

KNOWLEDGE DISTILLATION TECHNIQUES FOR MACHINE LEARNING

by

Kenneth Erbs Borup

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Statistics



Department of Mathematics

Aarhus University

October 2023

Knowledge Distillation Techniques for Machine Learning

Ph.D. dissertation by
Kenneth Erbs Borup

Department of Mathematics, Aarhus University
Ny Munkegade 118, 8000 Aarhus C, Denmark

Supervised by
Associate Professor Lars Nørvang Andersen, Aarhus University
Professor Henrik Karstoft, Aarhus University

Submitted to Graduate School of Natural Sciences, Aarhus, 23. October 2023



Preface

This dissertation is a tangible conclusion of my Ph.D. studies at the Department of Mathematics, Aarhus University as part of the Stochastics group and the Centre for Digitalisation, Big Data and Data Analytics (DIGIT). It was written during the period from February 2020 through October 2023 under the supervision of Lars Nørvang Andersen and Henrik Karstoft.

The dissertation comprises five self-contained papers, preceded by a general introduction to the research field. The papers, and their respective venues, are:

Paper A Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation.

Advances in Neural Information Processing Systems 34 (NeurIPS).

Paper B Self-Distillation for Gaussian Process Models.

Submitted to Journal of Machine Learning Research (JMLR).

Paper C Distilling from Similar Tasks for Transfer Learning on a Budget.

International Conference on Computer Vision (ICCV) 2023.

Paper D A Quest for Perfect Teacher-Student Agreement in Knowledge Distillation.

Working paper.

Paper E Automatic Sleep Scoring using Patient-Specific Ensemble Models and Knowledge Distillation for Ear-EEG Data.

Biomedical Signal Processing and Control (Volume 81).

Besides layout and minor edits, the papers are replications of the published or submitted versions at the respective venues. Thus, some differences in notation and terminology are inevitable, but any ambiguousness should be easily alleviated through the context of the respective papers and venues. Furthermore, due to the strict page requirement of most current machine learning venues, the main content of Papers A, B, C, and D is highly condensed and additional details and results are largely deferred to the supplementary material of each paper.

The introductory chapter primarily acts as a general introduction to the research field, as well as motivates and positions the five papers in the field of research. Paper A and Paper B are joint work with Lars Nørvang Andersen, and I have contributed significantly to both the research and writing of these papers. For Paper B the research and writing were more or less equally divided between us. In accordance with GSNS rules, large parts of Paper A were also part of the progress report for my qualifying examination. Paper C is the result of my research stay at Cornell University in Ithaca, New York, USA visiting Professor Bharath Hariharan during the spring of 2022. It was written in collaboration

Preface

with Cheng Perng Phoo and Bharath Hariharan, where I contributed significantly in both research and writing. Paper D is a single-author working paper initiated during the latter part of my Ph.D. studies and is yet to be finalized. Finally, Paper E is a joint work with Kaare Mikkelsen and Preben Kidmose of the Bioelectrical Instrumentation and Signal Processing research group at Aarhus University as well as Huy Phan from Queen Mary University of London. I undertook the majority of research and writing of this paper, and in particular, Kaare Mikkelsen contributed to conceptualizing the research.

* * *

Now, at the culmination of my studies with the completion of this treatise, I would like to express my sincerest gratitude for the journey that has been my Ph.D. studies. It has been both deeply challenging and highly rewarding — I owe a huge thank you to everyone who has supported me along this journey.

I thank my supervisors Lars Nørvang Andersen and Henrik Karstoft for allowing me to pursue a Ph.D. degree in such a stimulating and challenging field of research. I appreciate your guidance, support, and readiness to discuss my, at times endless amount of, research ideas.

In the spring of 2022, I had the great pleasure of visiting Bharath Hariharan at Cornell University in Ithaca, New York, USA. I am thankful to Cornell University for the hospitality, and to Bharath for great support and for allowing me to be a part of the department. Also, thank you to my co-author Cheng Perng Phoo for good company and engaging scientific discussions. I would also like to express my gratitude to Stibo-Fonden, Augustinusfonden, Otto Mønstedts Fond, William Demant Fonden, and Knud Højgaards Fond for the travel funds they have granted me. I am also grateful to the Department of Mathematics and DIGIT for providing financial support for my participation in conferences and my research visit, and for the research community I have been part of.

Conducting my Ph.D. studies at the Department of Mathematics at Aarhus University has been great, and my colleagues deserve a lot of recognition for this. In particular, I would like to thank my office mates Helene, Lota, Ragnhild, and Jakob, although our discussions have been less academic and more about traveling, career, and Formula 1.

Finally and most importantly, my family and friends deserve huge appreciation for listening to my relentless blabbering about my research, despite it most often being completely irrelevant and incomprehensible to you. I am thankful to my parents, Dan and Lone, for their patience and never-ending support, and to my brother, Daniel, for the many discussions on statistical matters. An especially heartfelt thank you is reserved for my wife Lea — words can not adequately express my gratitude for your unconditional support, encouragement, and love. Thank you for everything! Finally, a deep thank you to our son, Isak, who has made the last endeavors of my studies the most memorable of all.

Kenneth E. Borup
Aarhus, October 2023

Contents

Preface	i
Abstract	v
Resumé	vii
Introduction	1
1 Machine Learning	1
2 Knowledge Distillation	5
3 Introduction of Papers	11
Bibliography	19
Paper A Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation	29
<i>by Kenneth Borup and Lars N. Andersen</i>	
A.1 Introduction	30
A.2 Related Work	31
A.3 Problem Setup	32
A.4 Main Results	33
A.5 Approximate Optimal Weighting Parameter for Deep Learning	42
A.6 Conclusion	43
Bibliography	44
Supplementary material	48
Paper B Self-Distillation for Gaussian Process Models	59
<i>by Kenneth Borup and Lars N. Andersen</i>	
B.1 Introduction and Problem Setting	60
B.2 Preliminaries	62
B.3 Self-Distillation for GP Regression	63
B.4 Self-Distillation for GP Classification	66
B.5 Additional Experiments	70
B.6 Related Works	70
B.7 Conclusion	71
Bibliography	71
Supplementary material	75

Paper C Distilling from Similar Tasks for Transfer Learning on a Budget	97
<i>by Kenneth Borup, Cheng Perng Phoo, and Bharath Hariharan</i>	
C.1 Introduction	98
C.2 Related Work	100
C.3 Problem Setting	101
C.4 Cross-Task Distillation for Constructing Efficient Models from Foundation Models	102
C.5 Experiments and Results	105
C.6 Conclusion	110
Bibliography	111
Supplementary material	116
Paper D A Quest for Perfect Teacher-Student Agreement in Knowledge Distillation	127
<i>by Kenneth Borup</i>	
D.1 Introduction	128
D.2 Method	129
D.3 Experiments	130
D.4 Related Works	136
D.5 Conclusion	137
D.6 Future Directions of Research	137
Bibliography	137
Supplementary material	141
Paper E Automatic Sleep Scoring using Patient-Specific Ensemble Models and Knowledge Distillation for Ear-EEG Data	145
<i>by Kenneth Borup, Preben Kidmose, Huy Phan, and Kaare Mikkelsen</i>	
E.1 Introduction	146
E.2 Problem Setup and Methods	147
E.3 Results	152
E.4 Discussion and Conclusion	155
Bibliography	158
Supplementary material	162

Abstract

Knowledge distillation is a powerful and flexible machine learning technique that can be used to train smaller, more efficient models (called students) by mimicking larger trained models (called teachers). Such students can often achieve better predictive performance than models trained in a classical supervised manner. However, despite its empirical success, a rigorous foundation of knowledge distillation is still largely non-existent.

This dissertation investigates both the theoretical and empirical foundations of knowledge distillation. In the first two papers, we develop theoretical frameworks for understanding knowledge distillation in the simplified settings of self-distillation with kernel ridge regression and Gaussian process models, respectively. In these frameworks, we investigate the properties of iterative self-distillation and determine particular regularizing behaviors imposed by self-distillation.

In a third paper, we perform a rigorous empirical study of knowledge distillation with neural networks to support our theoretical findings. We investigate the efficiency of knowledge distillation under various controlled settings to determine under which conditions we can obtain perfect teacher-student agreement. In a fourth paper, we illustrate the real-world applicability of knowledge distillation by showing how to apply knowledge distillation for personalized automatic sleep scoring based on Ear-EEG measurements.

Finally, in a fifth paper, we address the challenge of exploiting diverse publicly available neural network models to improve the predictive performance on a given task under computational constraints. In particular, we propose a method to construct efficient models by identifying and distilling suitable pre-trained models with minimal access to these models.

Overall, this dissertation contributes to the theoretical and empirical foundations of knowledge distillation and proposes novel methods for adapting publicly available neural network models to specific tasks under constraints.

Resumé

Knowledge distillation er en effektiv og fleksibel maskinlæringsteknik, der kan bruges til at træne mindre, mere effektive modeller (kaldet *students*) ved at efterligne større trænede modeller (kaldet *teachers*). Sådanne *students* opnår ofte bedre prædiktiv nøjagtighed (*accuracy*) end tilsvarende modeller trænet med klassisk superviseret træning. På trods af den empiriske succes er et stringent grundlag for *knowledge distillation* stadig stort set ikke-eksisterende.

Denne afhandling undersøger både det teoretiske og empiriske grundlag for *knowledge distillation*. I de første to artikler opstilles teoretiske rammer til at opnå en forbedret forståelse af *knowledge distillation* i de forenkede tilfælde med *self-distillation* for kernel ridge regression og Gaussiske procesmodeller. I disse forenkede tilfælde undersøger vi egenskaberne ved iterativ *self-distillation* og udleder særlige de regulariserende egenskaber, der opstår ved gentagen *self-distillation*.

I en tredje artikel laves en struktureret empirisk undersøgelse af *knowledge distillation* med neurale netværk for at understøtte de teoretiske resultater. Vi undersøger effektiviteten af *knowledge distillation* i forskellige kontrollerede forsøg for at bestemme, under hvilke betingelser man kan opnå perfekt enighed mellem *teacher* og *student*. I en fjerde artikel illustrerer vi anvendeligheden af *knowledge distillation* på data fra den virkelige verden ved at anvende *knowledge distillation* til personlig og automatisk søvnscoring baseret på *Ear-EEG*-målinger.

Endelig, i en femte artikel behandles udfordringen med at udnytte forskellige offentligt tilgængelige neurale netværksmodeller til at forbedre den prædiktive nøjagtighed på en given opgave med begrænset computerkraft. Vi introducerer en metode til at konstruere effektive modeller ved at identificere og destillere relevante præ-trænede modeller med minimal adgang til disse modeller.

Denne afhandling bidrager både til det teoretiske og empiriske grundlag for *knowledge distillation* og introducerer nye metoder til at tilpasse offentligt tilgængelige neurale netværksmodeller til specifikke opgaver under begrænsninger på computerkraft.

Introduction

This chapter serves as an introduction to the field of research and the topics of the dissertation. Due to the rapid speed of research and publication by the machine learning research community, few widely established books on the foundations of machine learning exist, and the following will be a convolution of information from various references, and as is increasingly common in the field of machine learning, also curated preprints.

The chapter will be organized as follows. Initially, machine learning will be introduced in a general and common setting, followed by an overview of how to train and evaluate such models with various degrees of annotated and/or unannotated data. Afterward, the idea of knowledge distillation will be presented along with the associated terminology and aim as well as a brief overview of some extensions on the conventional knowledge distillation idea, both in terms of what *knowledge* to distill as well as what to distill it from. Finally, each paper included in this dissertation is individually motivated and the main results of each paper are introduced.

1 Machine Learning

Let \mathcal{X} be an input space and \mathcal{Y} an output space. Let $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \dots, n\}$ be an observed set of elements, where we assume that there exists some underlying unknown function, $f^* : \mathcal{X} \rightarrow \mathcal{Y}$, mapping samples $\mathbf{x} \in \mathcal{X}$ to values or labels $\mathbf{y} \in \mathcal{Y}$ possibly affected by some unknown noise. The aim of machine learning is to approximate the underlying true function f^* , by some function f , often parameterized by some parameters θ . This is typically achieved through *training* of the model: i.e. by the minimization of some objective function, quantifying the loss associated with the approximation, over the set of training samples $\mathcal{D}_{\text{train}}$. When f is parameterized as a neural network, we usually refer to the model and procedure as deep learning.

1.1 Deep learning

Deep learning is a somewhat vaguely defined subfield of machine learning, usually referring to research and applications of neural network models. Depending on the specific parameterization of the model, we subdivide neural networks into different *architectures*, and while originally neural networks referred to merely fully-connected multi-layer perceptrons (MLP), recently it encapsulates anything from MLPs, through Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), to Transformers amongst others. However, for ease of exposition, we will not introduce all of these architectures here but refer to e.g. [Goodfellow et al. \(2016\)](#) for details on the

most established architectures, and e.g. Vaswani et al. (2017) for details on the more modern transformer architecture. A fully-connected MLP is defined as the composition of a sequence of functions;

$$f_{\theta}(\mathbf{x}) = l_D \circ \dots \circ l_1(\mathbf{x}),$$

where $l_i(\mathbf{z}) \stackrel{\text{def}}{=} \psi(\mathbf{W}_i \mathbf{z} + \mathbf{b}_i)$ is a linear transformation with weights, \mathbf{W}_i , and bias, \mathbf{b}_i , such that $\theta = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \dots, \mathbf{W}_D, \mathbf{b}_D\}$, followed by a differentiable nonlinear activation function ψ . Different activation functions are commonly used, and while e.g. the sigmoid and hyperbolic tangent functions were historically popular modern models typically use e.g. the Rectified Linear Unit (ReLU) or adaptations hereof (Fukushima, 1975; Nair and Hinton, 2010; Maas et al., 2013; Hendrycks and Gimpel, 2016). Each function l_i is often referred to as a *layer* and the amount of such composed layers is the *depth* of the model. Furthermore, the output dimension of W_i is commonly called the *width* of the layer, and in turn, defines the width of the model. Both the depth and width of the model need to be predefined before the training of the model. Different architectures are typically comprised of different types of layers, skip-connections, pooling, attention, etc. and we refer to e.g. Goodfellow et al. (2016) for details on this.

1.2 Training and Evaluating Models

Since we aim to approximate the unknown underlying function f^* , we need to fit (i.e. train) our model on some observations produced by this mapping. Thus, in *supervised learning* we assume access to a labeled training dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_n^N$, where we assume that $\mathbf{y}_n = f^*(\mathbf{x}_n)$ for $n = 1, \dots, N$. We also select a reasonable evaluation measure of the goodness of fit. For instance, for classification, we typically aim to maximize the accuracy of our model. However, since accuracy is non-differentiable, we will in such cases need to optimize our model using a proxy for the evaluation measure. Therefore, given a pre-defined model f parameterized by $\theta \in \Theta$, the model is trained by minimization of some objective function, \mathcal{L} , quantifying the error (or loss) between the model predictions and the true targets over the training dataset, $\mathcal{D}_{\text{train}}$:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{N} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{train}}} \ell(f_{\theta}(\mathbf{x}), \mathbf{y}), \quad (1)$$

where ℓ is typically the cross-entropy for classification tasks and the mean squared error for regression tasks. In practice, many machine learning tasks are either overparameterized and/or ill-posed optimization problems, and some type of regularization is applied to restrict the space of possible solutions. Often one aims to obtain simple and sparse solutions to reduce the generalization error, by making the models less prone to overfitting the training data (Hastie et al., 2009; Goodfellow et al., 2016). For ease of exposition, we leave out explicit regularization terms here.

Unfortunately, (1) is intractable for neural networks and no analytical solution can be derived (Hastie et al., 2009). Thus, in practice, a solution, θ^* , for neural networks is typically computed using first-order numerical methods such as stochastic gradient descent (SGD)¹. Denote by $\ell_n(\theta) \stackrel{\text{def}}{=} \ell(f_{\theta}(\mathbf{x}_n), \mathbf{y}_n)$ the loss on the n 'th sample, when the

¹ A note on terminology. Typically, the procedure is called gradient descent (GD) when the gradient is computed over all samples at each iteration, stochastic gradient descent (SGD) when the gradient is

model is parameterized by θ , then SGD amounts to iteratively solving

$$\theta_{t+1} \stackrel{\text{def}}{=} \theta_t - \eta \frac{1}{b} \sum_{n \in \mathcal{B}_t} \nabla_{\theta} \ell_n(\theta_t) \quad \text{where } t = 0, \dots, T,$$

for $T > 1$ iterations where \mathcal{B}_t is a subset of $b > 1$ (*batch size*) indexes from $\{1, \dots, N\}$ called a *batch*, $\eta > 0$ is a step size typically denoted the *learning rate*, and θ_0 is randomly initialized. In practice, the batches \mathcal{B}_t typically traverse through a shuffled list of all the indexes, where each traversal of all N samples is called an *epoch* — hence, often $T \propto \lfloor \frac{N}{b} \rfloor$. Other more advanced optimization techniques e.g. utilize the empirical average of the first (Hinton et al., 2012) and second (Kingma and Ba, 2015) moments of the gradient to obtain faster convergence speed. However, research into improved optimization techniques for neural networks is an active research field (Anil et al., 2020; Schmidt et al., 2021). Furthermore, the learning rate η can also be iteration dependent, η_t , and e.g. be large initially and gradually decrease as the optimization approach minima for a more stable optimization procedure. Such a procedure is commonly called a *learning rate schedule*, where common choices are the cosine annealing schedule and linear decay (Smith, 2017, 2018; Loshchilov and Hutter, 2017).

Following the optimization, we denote by θ_T and f_{θ_T} the trained weights and trained model, respectively. To obtain an estimate of the generalization performance of f_{θ_T} it is evaluated on a test set $\mathcal{D}_{\text{test}}$ assumed to follow the same data distribution and underlying mapping as $\mathcal{D}_{\text{train}}$. Depending on the task various evaluation metrics could be of interest; e.g. accuracy, precision, or recall for classification, and (root) mean squared error or mean absolute error for regression tasks. When computing the evaluation metric over the training or test set, we refer to the computed metric as the training or test metric, respectively. Without loss of generality, we now consider accuracy, where a higher score is better.

Generally, if the test metric is low compared to the training metric the model is likely overfitting to the training data and is not able to generalize to similar but unseen data. However, if both the test metric and the training metric are low, the model is likely underfitting the training data and is not able to accurately represent the data. Both cases require modifications of the training scheme, and possibly any of the customizable elements (such as e.g. model architecture, learning rate, or batch size), which we collectively denote hyperparameters can be of importance. Yet, if we were to rerun our experiment multiple times with different sets of hyperparameters, and reevaluate our model on the test set each time, we could potentially overfit to the test set. Thus, we use a validation set \mathcal{D}_{val} under the same assumptions as the test set for model selection, and merely evaluate our final selected model on the test set.² In practice, hyperparameters are often selected either based on the researchers' experience, based on a grid search over all possible hyperparameter combinations (within some bounds), or by e.g. random search or Bayesian hyperparameter optimization (Snoek et al., 2012).

computed over merely a single sample at each iteration, and mini-batch gradient descent when a subsample of $b > 1$ samples is used at each iteration. However, an irrevocable practice in machine learning is to denote mini-batch gradient descent as SGD irrespective of $b > 1$. We will adopt this convention here.

² Ideally we would apply a more elaborate evaluation scheme such as cross-validation, but due to the computational costs of optimizing neural networks, such elaborate schemes are prohibitive.

1.3 Connections Between Neural Networks and Kernel Methods

Although the inherent nonlinearity of neural networks implies the need for numerical optimization, equivalences between sufficiently wide³ neural networks and certain kernel methods have been shown (Neal, 1996; Jacot et al., 2018; Arora et al., 2019; Lee et al., 2018a, 2019; Yang, 2019; Du et al., 2019b,a). Specifically, when each layer of a randomly initialized fully connected neural network tends to infinity one layer at a time, the neural network (when seen as merely a function) tends to a Kernel (Ridge) Regression (KRR) model with the Neural Tangent Kernel (NTK) as the kernel. The NTK can be explicitly derived for most neural network architectures (Arora et al., 2019; Yang, 2019; Hron et al., 2020), and through numerous existing theoretical results on KRR, the behavior of wide neural networks can be analyzed. However, despite theoretical equivalences, the empirical performance of KRR with the NTK still deviates notably from the empirical performance of neural networks (Yang and Hu, 2021). Similarly, we now consider a D -layer deep fully connected neural network with layers l_i of width n_i for $i = 1, \dots, D$ parameterized by weights \mathbf{W}_i and biases \mathbf{b}_i . We draw all elements of weights and biases independently with zero mean and variances σ_w^2/n_i and σ_b^2 respectively. Then Lee et al. (2019) show that the neural network is equivalent to a Gaussian process model with a particular covariance function when the width n_i of each layer tends to infinity (Neal, 1996; Matthews et al., 2018; Novak et al., 2019). This Gaussian process model is dubbed the Neural Network Gaussian Process (NNGP).

The equivalences of specific kernel ridge regression and Gaussian process models with certain neural networks establish the foundation and motivate our theoretical analysis of distillation for kernel ridge regression and Gaussian process models in Paper A and Paper B, respectively. For additional details on NTK and NNGP, we refer to e.g. Jacot et al. (2018) and Lee et al. (2019), respectively.

1.4 Different Degrees of Supervision

While the most common learning scheme is supervised learning, annotated data can be expensive and therefore sparse in practice. Thus, different approaches to training a neural network exist. Opposite to supervised learning introduced above, unsupervised or self-supervised learning has no access to targets during training. Both these fields of research aim at training models using merely input data, without any associated targets, often by creating auxiliary tasks, somewhat resembling the properties one might expect useful for a particular task (Goodfellow et al., 2016). For instance, in contrastive learning (a self-supervised approach) a computer vision model is trained to map two different augmentations of the same image to similar representations while ensuring different images yield representations further away (He et al., 2020; Grill et al., 2020; Chen et al., 2020a,b,c; Caron et al., 2020). Following such *pre-training* procedure, the model naturally has to be either fine-tuned on some labeled downstream task (which is now possible with less annotated data) or used for unsupervised tasks such as re-identification of particular individual objects. Semi-supervised learning is the trade-off in which some

³ Determining the exact width required for the equivalences to hold is still an active and open research question. Initially, the width was required to tend to infinity in a particular order (Jacot et al., 2018), but more recent research has shown equivalences for wide but finite width networks (Arora et al., 2019). We omit the details for ease of exposition and consider the infinite case here.

data has associated targets, while the remaining is unlabeled. Techniques for semi-supervised learning typically aim at effectively propagating target information from the annotated samples to the unannotated samples, through e.g. pseudo-labeling/self-training or consistency regularization (Lee, 2013; Xie et al., 2020a; Sohn et al., 2020; Tarvainen and Valpola, 2017; Berthelot et al., 2019; Phoo and Hariharan, 2021; Islam et al., 2021). As will be evident in the following sections, knowledge distillation can be posed with either of these different degrees of supervision.

2 Knowledge Distillation

This section serves as a general introduction to the idea and motivation of knowledge distillation. It includes a presentation of the unusual yet distinctive terminology and analogies associated with knowledge distillation, a mathematical definition of conventional knowledge distillation as introduced by Hinton et al. (2015) along with a mention of the immense flexibility of the method, as well as an overview of more recent alterations and modifications to the conventional knowledge distillation procedure, both in terms of what knowledge to transfer and what to transfer knowledge from.

2.1 Motivation

Despite commonly attributed to Hinton et al. (2015), the idea of knowledge distillation originates back to Bucila et al. (2006); Ba and Caruana (2014) under the more general term *model compression*. However, common for all three formulations is the aim of “compressing” a cumbersome model (e.g. an ensemble of models) into a smaller, faster, and more efficient model without a significant loss in predictive performance. The success of any such procedure would yield cheaper and faster inference in applications and allow for improved and widened utilization of machine learning in devices with limited compute such as e.g. smartphones, headphones, or cameras.

2.2 Terminology of the Classroom

The field of knowledge distillation has collectively adopted a lot of terminology from the common classroom, and thus often allows for easy interpretation through analogies and connections to such settings. For instance, consider the following setting; a student is about to learn a given topic and aims at performing as well as possible on an upcoming test. The student is provided both with a textbook on the topic, containing annotated examples and information to learn from and with access to a teacher knowledgeable on the subject. The student is allowed to ask the teacher questions with the aim of learning from the answers. This setting largely corresponds to the general knowledge distillation setting, where the teacher is a well-trained high-capacity model for a specific task, the student is a smaller untrained model, the textbook is a labeled training dataset, and answers from the teacher acts as soft targets for the student. Now, no two teaching situations are identical: teachers teach differently, students learn differently, the size and quality of the textbook might differ, different tasks and subjects require different knowledge to be taught, and so on. Most alterations of the teaching situation have an analogous knowledge distillation procedure or setting. For instance, whether the

student asks the teacher questions about examples from the textbook, previously unseen unannotated examples, or both, translates to either supervised, unsupervised, or semi-supervised knowledge distillation. Furthermore, we might consider how vague or open-ended answers we want from the teacher (i.e. how soft our teacher targets should be), whether to be provided with answers, or the thought process behind such answers (i.e. perform response-based or feature-based distillation), or how much weight to assign to the answers of the teacher compared to the annotated examples from the textbook (i.e. how to weight supervised and distillation-objectives during knowledge distillation).

Such analogies and connections often allow a faster entrance to the research field but also tend to ease the task of communicating state-of-the-art research in a technical field to laypeople. In the remainder of this dissertation, few such analogies will be made explicitly, but a strong encouragement, especially for the uninformed reader, to mentally use the analogies stand.

2.3 Original Aim and Formulation

In the following, the concept of knowledge distillation as formulated by [Hinton et al. \(2015\)](#) is restated in a slightly generalized manner. This particular distillation procedure is remarkably simple yet exceptionally flexible. With minor modifications, it allows for either supervised, semi-supervised, or unsupervised distillation, while being model agnostic for both the teacher and student, yet requiring minimal access to the teacher model. Despite its simplicity, it has shown great competitiveness to more intricate distillation procedures ([Beyer et al., 2022](#)).

We now consider a C -class classification setting for $C > 1$. Then, given an input example $\mathbf{x} \in \mathcal{X}$ it yields a vector of *logits*, $\mathbf{z} = f(\mathbf{x}) \in \mathbb{R}^C$, when passed through a model f . Typically, such logit vectors are transformed into a probability vector by applying the softmax function, but for distillation, we use the temperature-scaled softmax

$$\sigma(\mathbf{z}, \tau) = (\sigma_1(\mathbf{z}, \tau), \dots, \sigma_C(\mathbf{z}, \tau)), \quad \text{where} \quad \sigma_i(\mathbf{z}, \tau) = \frac{\exp\left(\frac{z_i}{\tau}\right)}{\sum_c^C \exp\left(\frac{z_c}{\tau}\right)},$$

and where $\tau > 0$ is a hyperparameter, called temperature. This corresponds to the classical softmax for $\tau = 1$, and in such cases, we will omit the τ from the notation, i.e. $\sigma_i(\mathbf{z}, 1) \stackrel{\text{def}}{=} \sigma_i(\mathbf{z})$. By adjusting τ , the distribution of the resulting probability vector can be adjusted. In particular, for $\tau < 1$ the vector is sharpened, thereby assigning a larger probability to the largest entry of the logits. Conversely, for $\tau > 1$, the probability is dispersed between the entries, yielding a *softened* probability vector. The extremes $\tau \rightarrow 0$ and $\tau \rightarrow \infty$, correspond to a one-hot encoded (hard) probability vector with all weight assigned to the largest entry of the logit and a completely uninformative probability vector assigning equal probability to all classes, respectively.

The knowledge distillation procedure formulated by [Hinton et al. \(2015\)](#) corresponds to utilization of a combination of two loss functions; an ordinary supervised cross-entropy loss between the student logits, $\mathbf{z} = f_s(\mathbf{x})$, on some examples $\mathbf{x} \in \mathcal{X}$, and the associated ground-truth target \mathbf{y} ,

$$\ell_L(\mathbf{z}_s, \mathbf{y}) \stackrel{\text{def}}{=} - \sum_{c=1}^C \mathbf{y}_c \log(\sigma_c(\mathbf{z}_s)), \quad (2)$$

and a distillation loss,

$$\ell_D(\mathbf{z}_s, \mathbf{z}_t, \tau) \stackrel{\text{def}}{=} -\tau^2 \sum_{c=1}^C \sigma_c(\mathbf{z}_t, \tau) \log(\sigma_c(\mathbf{z}_s, \tau)), \quad (3)$$

between the temperature-scaled logits, $\mathbf{z}_t = f_t(\mathbf{x})$, of the teacher and the student logits, \mathbf{z}_s . Note, the multiplier τ^2 on the distillation loss follows the presentation of [Hinton et al. \(2015\)](#) and ensures a consistently comparable scale between the supervised and distillation loss when altering the hyperparameters since the gradients of the distillation loss are scaled by τ^{-2} compared to the supervised loss. Thus, the full objective function is a linear combination of the above loss functions over the available labeled dataset, \mathcal{D}_L and distillation dataset, \mathcal{D}_D , respectively,

$$\mathcal{L} = \frac{\alpha}{|\mathcal{D}_L|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_L} \ell_L(\mathbf{z}_s(\mathbf{x}), \mathbf{y}) + \frac{1-\alpha}{|\mathcal{D}_D|} \sum_{\mathbf{x} \in \mathcal{D}_D} \ell_D(\mathbf{z}_s(\mathbf{x}), \mathbf{z}_t(\mathbf{x}), \tau), \quad (4)$$

where $\alpha \in [0, 1]$ is a hyperparameter adjusting the weight assigned to the teacher predictions and ground-truth labels respectively. A notable feature of (4) is its immense flexibility: choosing $\alpha = 1$ yields ordinary supervised training while $\alpha = 0$ yields a fully unsupervised training by utilization of the teacher-produced pseudo-targets, and for $\alpha \in (0, 1)$ the procedure becomes semi-supervised. Furthermore, if we denote by $\mathcal{D}^x \stackrel{\text{def}}{=} \{\mathbf{x} \mid (\mathbf{x}, \mathbf{y}) \in \mathcal{D}\}$, then one can have both that $\mathcal{D}_D \subseteq \mathcal{D}_L^x$, or $\mathcal{D}_D \supseteq \mathcal{D}_L^x$, or even that $\mathcal{D}_D \cap \mathcal{D}_L^x = \emptyset$. That is the input examples of \mathcal{D}_L and \mathcal{D}_D can be identical, any manner of overlapping, or completely disjoint. Furthermore, since the training procedure requires merely the logits of the teacher model, it is not only model agnostic, but also merely requires access to the output of the model, and not any of the intermediate activations or weights. Hence, utilizing this distillation scheme in principle allows for the distillation of proprietary black-box teacher models, potentially only accessible for inference through e.g. an online API (given logits can be obtained from such a model). However, in practice, obtaining a sufficient number of examples through such API might be infeasible due to costs, time, or policies restricting access and/or utilization of model predictions for the training of new models.

A note on model size. In the above the term *model size* has been used intentionally vague. While the size of neural networks is often measured by the number of parameters used for a particular architecture, it is merely one of many possible proxies for model size. In particular, model size might refer to e.g. the functional space spanned by a model, the memory requirements needed for training or deployment, inference and/or training speed, FLOPS required for inference, or any other measure of model size relevant to the application of interest. Throughout this introduction, the common procedure of using the number of model parameters as a proxy for model size will be adopted unless otherwise mentioned.

2.4 A Generalized Formulation of Knowledge Distillation

Although the objective function in (4) is already a slight generalization of the conventional distillation objective, more recently introduced distillation procedures require a yet more general formulation of distillation. Recent research on distillation techniques

has introduced an immense amount of flexibility and modifications to the idea of knowledge distillation, including variations on what *knowledge* to transfer from the teacher as well as what a teacher model can or should look like. In particular, in recent work ℓ_D is allowed to be any loss function that is able to compute a distance between two models; e.g. based on the intermediate layer activations, activation correlation, or inter-sample geometry. Furthermore, methods allowing teachers of identical architecture to the student, multiple homo- or heterogeneous models as teachers, or teachers trained on different tasks have also been proposed. Thus, in current literature, the term *distillation* is often used for any procedure that encompasses the training of a model on the output of another model. Thus, in general, one can consider distillation as the general procedure for solving

$$\begin{aligned} \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(f_\theta, g_\omega) &\stackrel{\text{def}}{=} \operatorname{argmin}_{\theta \in \Theta} \frac{\alpha}{|\mathcal{D}_L|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_L} \ell_L(f_\theta(\mathbf{x}), \mathbf{y}) + \frac{1 - \alpha}{|\mathcal{D}_D|} \sum_{\mathbf{x} \in \mathcal{D}_D} \ell_D(f_\theta(\mathbf{x}), g_\omega(\mathbf{x})) \\ &\stackrel{\text{def}}{=} \operatorname{argmin}_{\theta \in \Theta} (\alpha \mathcal{L}_L + (1 - \alpha) \mathcal{L}_D) \end{aligned} \quad (5)$$

where the latter equality is for notational convenience, and f_θ is the student model parameterized by $\theta \in \Theta$, g_ω the teacher model parameterized by $\omega \in \Omega$, ℓ_L is a supervised loss function between the student predictions and the original targets, and ℓ_D is a distillation loss function between any statistic of the student model and the same statistic of the teacher model. Either partial loss function can be evaluated over any relevant set of annotated and unannotated elements, respectively.

In the following, we provide additional details on some of the proposed distillation techniques, in particular on what information to transfer between the models, what the teacher model can look like, as well as how to make distillation work well in practice. For extended surveys on diverse distillation techniques see e.g. [Gou et al. \(2021\)](#); [Wang and Yoon \(2021\)](#).

2.5 Different Notions of Knowledge

While [Hinton et al. \(2015\)](#) focused on matching the teacher and student on the (softened) distribution of the pre-softmax logits, subsequent research has investigated the advantages of matching on other statistics. One can generally categorize these different approaches as either response-, feature-, or relational-based.

Response-based knowledge distillation. A response-based knowledge distillation technique aims at training the student to match the teacher on information obtained from the output layer of the model. Hence, the formulation by [Hinton et al. \(2015\)](#) is a response-based technique since it aims at making the student mimic the teacher’s logits. Response-based techniques have gained great recognition due to the simplicity and limited need for access to the teacher model, making them flexible procedures. Since the seminal distillation procedure introduced in Section 2.3 subsequent works have, amongst others, investigated different ways of capturing response-based information. Of other response-based knowledge distillation techniques [Tarvainen and Valpola \(2017\)](#); [Srinivas and Fleuret \(2018\)](#); [Kim and Rush \(2016\)](#); [Furlanello et al. \(2018\)](#); [Yang et al. \(2020\)](#); [Gao et al. \(2020\)](#) are worthy of an explicit mention.

Feature-based knowledge distillation. Different layers of a neural network learn different levels of feature representations (Kornblith et al., 2019; Raghu et al., 2021; Nguyen et al., 2021), and instead of merely training the student to mimic the output distribution of the teacher, feature-based knowledge distillation aims at mimicking the intricate and elaborate features of the teacher. Naturally, this requires access to intermediate activations of the teacher model limiting some practical applications on e.g. proprietary teacher models. Furthermore, care must be taken with regard to dimensional differences between teacher and student features.

Now we define \mathcal{L}_D to measure the squared Euclidean distance between intermediate activation vectors from both models (Romero et al., 2015) and take the form

$$\mathcal{L}_D \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_D|} \sum_{\mathbf{x} \in \mathcal{D}_D} \|\psi(f_\theta)(\mathbf{x}) - \beta(\psi(g_\omega)(\mathbf{x}))\|_2^2,$$

where $\psi(h)(\mathbf{x})$ is a function that extracts a flattened intermediate activation vector (features) of a model h evaluated on an element \mathbf{x} , and $\beta(\cdot)$ is a learnable linear transformation between the (likely different) dimensions of the teacher and student activation vectors. Other notable feature-based knowledge distillation techniques are Zagoruyko and Komodakis (2017); Heo et al. (2019); Tung and Mori (2019); Heo et al. (2019); Ahn et al. (2019).

Relation-based knowledge distillation. Where both response- and feature-based knowledge distillation utilize information from specific layers of the models, relation-based knowledge distillation employ the inter-layer or inter-sample relationships of the models. Now recall the general distillation loss in (5), then we could use \mathcal{L}_D to measure the mutual relations of data elements (Park et al., 2019), and take the form

$$\mathcal{L}_D \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_D^N|} \sum_{(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}^N) \in \mathcal{D}_D^N} \ell_{\text{huber}}(\phi(f_1, f_2, \dots, f_N), \phi(g_1, g_2, \dots, g_N)),$$

where \mathcal{D}_D^N is the set of N -dimensional tuples $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}^N)$, where each element is distinct and $\mathbf{x}^i \in \mathcal{D}_D$, and we use the notation $f_i \stackrel{\text{def}}{=} f_\theta(\mathbf{x}_i)$ and $g_i \stackrel{\text{def}}{=} g_\omega(\mathbf{x}_i)$. Furthermore, $\ell_{\text{huber}}(\mathbf{z}, \mathbf{w})$ is the Huber loss (Huber, 1964), and ϕ is a relational function that measures a geometric relational measure of the given N -tuple. For instance, let $N = 3$, and

$$\phi(\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k) \stackrel{\text{def}}{=} \frac{(\mathbf{z}_i - \mathbf{z}_j)^\top (\mathbf{z}_k - \mathbf{z}_j)}{\|\mathbf{z}_i - \mathbf{z}_j\|_2 \|\mathbf{z}_k - \mathbf{z}_j\|_2},$$

then we optimize the student to match the angle-wise relation between the teacher predictions. Other interesting approaches to relation-based knowledge distillation are Yim et al. (2017); Tung and Mori (2019); Lee et al. (2018b); Passalis et al. (2021).

2.6 What do teachers look like and how to learn from them

In the above, we investigated what knowledge to transfer between the teacher and student model, but we intentionally kept the teacher arbitrarily general. While indeed, the conventional formulation of knowledge distillation by Hinton et al. (2015) aimed at model compression, by matching a pre-trained teacher model, with a smaller student

Introduction

model, this formulation is not a necessity. In particular, with some adaptations to the distillation loss in (5) one can e.g. choose a teacher model a) of any (ensemble of) architecture(s) larger than the student (Hinton et al., 2015; Fukuda et al., 2017) b) of identical architecture as the student (Furlanello et al., 2018; Mobahi et al., 2020) or c) of smaller architecture than the student (Xie et al., 2020b). Additionally, a teacher can be trained on different datasets than the one currently at hand (Phoo and Hariharan, 2021), a completely different modality (Tian et al., 2020; Gupta et al., 2016), and multiple homo- or heterogeneous (both in architecture and dataset) teachers can be used in conjunction (You et al., 2017; Fukuda et al., 2017; Tan et al., 2019; Liu et al., 2020). The teacher can even be trained simultaneously with the student in online distillation or offline knowledge distillation can be combined with other model compression techniques such as quantization or pruning (Anil et al., 2018; Aghli and Ribeiro, 2021; Polino et al., 2018; Kim and Rush, 2016). However, the effectiveness of the conventional knowledge distillation has been observed to be dependent on the size gap between the teacher and student model (Mirzadeh et al., 2020; Cho and Hariharan, 2019). Furthermore, it has been shown that one can distill a teacher model without any access to data by synthesizing new data from the teacher with data-free distillation techniques (Chen et al., 2019; Fang et al., 2019; Lopes et al., 2017; Nayak et al., 2019).

Other than the conventional formulation of knowledge distillation, self-distillation is of particular interest for this dissertation, as its simplicity in formulation lends itself to a natural first step for analyzing the behavior of knowledge distillation. Self-distillation works by first training a teacher model on a labeled dataset and subsequently training a student model of identical architecture (or model class) to the teacher on the predictions of the teacher model, possibly using the original targets analogously to (6) as well.⁴ This procedure can be iterated multiple times and despite no additional information being provided to the system, it can yield an increase in predictive performance (Furlanello et al., 2018; Mobahi et al., 2020; Yang et al., 2018). This procedure does not aim at model compression, but rather at improving the predictive performance, and the idea of iteratively reusing a model as the teacher is the foundation of the theoretical investigation of Mobahi et al. (2020), and in extension also for our theoretical analyses in Paper A and Paper B. Mobahi et al. (2020) investigates self-distillation for particular constrained kernel regression models and finds that when no ground-truth targets are used, the models are progressively regularized more, yielding increasingly sparser models, eventually collapsing into a zero-solution. However, we find that this is not necessarily true when including the original targets in the distillation procedure or considering Gaussian process models.

Another relevant and interesting direction of research is on cross-domain distillation. While conventionally, Hinton et al. (2015) proposed to distill a teacher model trained on an identical dataset as the training dataset (or at least on a dataset assumed drawn from the same distribution), more recently the idea of cross-domain distillation has been proposed (Gupta et al., 2016; Tian et al., 2020; Phoo and Hariharan, 2021; Liu et al.,

⁴ Notably a few different approaches are often dubbed self-distillation, but unless otherwise mentioned, in this dissertation self-distillation will refer to the procedure described here.

2020). Cross-domain distillation allows the teacher model to be trained on any dataset and works under the assumption that the (intermediate) representations of the teacher model are informative on the training dataset. However, this assumption is hard to verify, and given multiple teacher models trained on data from different domains, it is unclear which teacher to choose. This challenge is investigated in Paper C.

2.7 Knowledge Distillation in Practice

Despite the empirical success of numerous knowledge distillation techniques, published empirical findings are often the result of highly specialized and optimized experiments on a set of benchmark datasets exploiting various optimization tricks. Some of these tricks include providing different augmentations of the same data to the teacher and student (possibly using less intrusive augmentations for the teacher data), obtaining teacher predictions once before training on clean data and providing the student with augmented data, or appending multiple weighted unsupervised or semi-supervised loss functions to the overall optimization problem (Faghri et al., 2023; Tian et al., 2020; Chen et al., 2020a,b,c; Gao et al., 2022; Grill et al., 2020). Such specialized tricks complicate comparisons and obscure the actual effectiveness and generalization of knowledge distillation techniques. Thus, Beyer et al. (2022) perform a large-scale empirical analysis of the conventional knowledge distillation procedure by Hinton et al. (2015) to identify a robust and effective recipe for effective knowledge distillation. Beyer et al. (2022) find that effective knowledge distillation generally requires consistent input data for the teacher and student model, exceptionally long training schemes compared to the majority of common deep learning training as well as strong data augmentation. These findings are in large part supported by another empirical, yet rigorous, analysis of the efficiency of knowledge distillation by Stanton et al. (2021). They also consider the conventional knowledge distillation setting by Hinton et al. (2015), and through numerous ablation studies conclude that the challenge of efficient knowledge distillation is solving an unusually hard optimization problem. The findings of both Beyer et al. (2022) and Stanton et al. (2021) motivate our empirical analysis in Paper D.

3 Introduction of Papers

In the following section, the five papers included in this dissertation are each introduced individually. Specifically, each is motivated and aligned with related research, and the methodological approach along with the main results are presented. The approach to investigate and improve knowledge distillation differs notably between the five papers. In particular, Paper A and Paper B are theoretical papers providing rigorous mathematical derivations, whereas Paper C, Paper D, and Paper E are empirical papers yielding experimental observations and/or provide novel procedures for certain empirical tasks. These differences will be clear in the following introductions and the papers themselves.

3.1 Paper A: Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation

Despite the significant empirical success of knowledge distillation, a theoretical justification of knowledge distillation lags behind mainly due to the immense challenges of applying mathematical rigor to neural networks and their training procedures. This paper provides theoretical results on the inner workings of the particular case of self-distillation in the simplified, yet mathematically rigorous setting of kernel ridge regression.

This paper is building on the seminal results of [Mobahi et al. \(2020\)](#), which analyses self-distillation of nonlinear functions in a Hilbert Space subject to ℓ_2 -regularization. By omission of details, this setting is largely similar to fitting kernel ridge regression models in a constrained optimization setting. In particular, [Mobahi et al. \(2020\)](#) first finds that the model, f_t , after any finite number, $t > 0$, of distillation steps, can be written as a weighted sum of some basis functions (see [Mobahi et al., 2020](#), equation (47)), and then shows that self-distillation without the original targets (except for the first fit) corresponds to applying a progressively amplified regularization to this solution (see [Mobahi et al., 2020](#), Theorem 5). Thus, self-distillation progressively sparsifies the set of basis functions used for the solution, eventually causing the solution to collapse to a zero-solution.

We propose to directly investigate self-distillation for kernel ridge regression and to include the original targets in each step of distillation in Paper A. That is, we consider the objective function

$$\begin{aligned} \mathcal{L}^{\text{distill}}(f(\mathbf{X}, \boldsymbol{\beta}), \mathbf{y}, \mathbf{y}^{(\tau-1)}) \\ = \frac{\alpha^{(\tau)}}{2} \|f(\mathbf{X}, \boldsymbol{\beta}) - \mathbf{y}\|_2^2 + \frac{1 - \alpha^{(\tau)}}{2} \|f(\mathbf{X}, \boldsymbol{\beta}) - \mathbf{y}^{(\tau-1)}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2, \end{aligned} \quad (6)$$

where $\alpha^{(\tau)} \in [0, 1]$, $\tau \geq 1$, $\lambda > 0$, $\mathbf{y} \in \mathbb{R}^n$ are the original targets, $\mathbf{y}^{(0)} \stackrel{\text{def}}{=} \mathbf{y}$, and $f(\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{\varphi}(\mathbf{X})\boldsymbol{\beta}$. Furthermore, $\mathbf{y}^{(\tau-1)} \stackrel{\text{def}}{=} f(\mathbf{X}, \hat{\boldsymbol{\beta}}^{(\tau-1)}) \in \mathbb{R}^n$ are the predictions of the previous iteration of the model where $\hat{\boldsymbol{\beta}}^{(\tau-1)} \stackrel{\text{def}}{=} \text{argmin}_{\boldsymbol{\beta}} \mathcal{L}^{\text{distill}}(f(\mathbf{X}, \boldsymbol{\beta}), \mathbf{y}, \mathbf{y}^{(\tau-2)})$. Thus, the objective in (6) is a weighted sum of two mean squared error objective functions with different targets and an ℓ_2 -regularization on the model weights. While indeed (6) can be solved analytically, doing so for multiple distillation steps quickly becomes computationally expensive. Thus, we derive an explicit expression for the solution at any distillation step, which can be computed efficiently based on the first ordinary fit to the original data. In particular, fix $\alpha^{(2)}, \dots, \alpha^{(\tau)} \in [0, 1)$, and let $\eta(i, \tau) \stackrel{\text{def}}{=} \prod_{j=i}^{\tau} (1 - \alpha^{(j)})$, then for $\tau \geq 1$, we have

$$\mathbf{y}^{(\tau)} = \left(\sum_{i=2}^{\tau} \alpha^{(i)} \eta(i+1, \tau) (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^{\tau-i+1} + \eta(2, \tau) (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^{\tau} \right) \mathbf{y} \quad (7)$$

$$f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(\tau)}) = \alpha^{(\tau)} f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(1)}) + (1 - \alpha^{(\tau)}) f(\mathbf{x}, \hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)}) \quad \text{for any } \mathbf{x} \in \mathbb{R}^d \quad (8)$$

where $\hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)}$ is the minimizer of (6) with $\alpha^{(\tau)} = 0$. Here we can reuse the computation of \mathbf{K} and the inverted matrix $(\mathbf{K} + \lambda \mathbf{I}_n)^{-1}$ for any $\tau > 1$. We now note that we can rewrite $\mathbf{K} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$ by the spectral decomposition, where $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix with the eigenvectors of \mathbf{K} as rows and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a non-negative diagonal matrix with the

associated eigenvalues in the diagonal. We can then rewrite (7) as

$$\mathbf{y}^{(\tau)} = \mathbf{V}\mathbf{B}^{(\tau)}\mathbf{V}^\top\mathbf{y}, \quad \text{where} \quad (9)$$

$$\mathbf{B}^{(\tau)} \stackrel{\text{def}}{=} \sum_{i=2}^{\tau} \alpha^{(i)} \eta(i+1, \tau) \mathbf{A}^{\tau-i+1} + \eta(2, \tau) \mathbf{A}^{\tau}, \quad \text{and} \quad (10)$$

$$\mathbf{A} \stackrel{\text{def}}{=} \mathbf{D}(\mathbf{D} + \lambda \mathbf{I}_n)^{-1}, \quad (11)$$

and $\mathbf{A}, \mathbf{B}^{(\tau)} \in \mathbb{R}^{n \times n}$ are diagonal matrices, and merely $\mathbf{B}^{(\tau)}$ depend on τ . Thus, any dynamic of the distillation procedure must be derivable from $\mathbf{B}^{(\tau)}$. By analyzing (9), we both recover the findings of [Mobahi et al. \(2020\)](#); the solutions are progressively sparsified for $\alpha = 0$, but also that we can provide no such guarantees when $\alpha \in (0, 1)$. In particular, we find that when we include the original targets in our distillation procedure the solutions are not consistently regularized stronger with each distillation step, but we can potentially obtain solutions which does not sparsify any further. We further provide a formula to compute how the solution will behave at the next immediate distillation step in such a setting.

Now, since the solutions do not necessarily collapse to a zero-solution for $\alpha \in (0, 1]$, we are able to derive

$$\lim_{\tau \rightarrow \infty} \mathbf{y}^{(\tau)} = \mathbf{K} \left(\mathbf{K} + \frac{\lambda}{\alpha} \mathbf{I}_n \right)^{-1} \mathbf{y} \quad (12)$$

as a simple closed-form expression for the solution after an infinite number of distillation steps. This solution merely corresponds to an ordinary kernel ridge regression solution with regularization parameter α^{-1} , despite the intermediate solutions showing different behavior as argued above. Furthermore, if we let $\tilde{\mathbf{X}} \in \mathbb{R}^{m \times d}$ be the matrix of validation inputs, $\tilde{\mathbf{y}} \in \mathbb{R}^m$ the associated vector of validation targets and allow $\alpha^{(\tau)} \in \mathbb{R}$, we find an *optimal* $\alpha^{(\tau)}$ at each step τ as the solution to

$$\alpha^{*(\tau)} = \underset{\alpha^{(\tau)} \in \mathbb{R}}{\operatorname{argmin}} \left\| \tilde{\mathbf{y}} - f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) \right\|_2^2$$

Finally, we provide a procedure to estimate the optimal weight efficiently for neural networks and support these theoretical findings with illustrative examples for kernel ridge regression and empirical experiments for neural networks.

3.2 Paper B: Self-Distillation for Gaussian Process Models

The aim of Paper B is to extend the kernel ridge regression setting investigated in [Mobahi et al. \(2020\)](#) and Paper A to Gaussian Process (GP) models. Unlike the deterministic predictions obtained from kernel ridge regression, GP models produce probabilistic predictions, and it is not immediately clear how one should incorporate such probabilistic information in a distillation setting. At the time, no investigation of distillation with GPs existed in the literature, and thus, in this paper, we propose two different types of self-distillation with GP models for both regression and classification tasks.

Specifically, we proposed a deterministic procedure, named d -GPSD, as well as a probabilistic procedure, named ρ -GPSD. d -GPSD discards the uncertainty estimates of the predictions and merely reuses the mean predictions for the training of the model in the succeeding iteration, whereas ρ -GPSD reuses the full posterior predictive distribution as a prior for the succeeding iteration of distillation, thereby incorporating the

Introduction

probabilistic nature of the predictions. For the sake of brevity, we restrict our analysis to the case where ground-truth targets are only used for the first model fit, and all subsequent fits are based on merely the predictions of the preceding iteration of the model. Furthermore, for simplicity, we consider one-dimensional input and targets (i.e. univariate regression or binary classification).

Generally, we assume a generic input-output pair $(\mathbf{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ is related by $y = f(\mathbf{x}) + \varepsilon$, where $f \sim \mathcal{GP}(m, k)$ with mean function $m : \mathbb{R}^d \rightarrow \mathbb{R}$ and positive semi-definite covariance function (kernel) $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, and independent $\varepsilon \sim \mathcal{N}(0, \gamma)$. However, for ease of exposition, we only consider univariate inputs, $x \in \mathbb{R}$. Given a training set $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, N\}$, where we denote $\mathbf{x} \stackrel{\text{def}}{=} (x_1, \dots, x_N)^\top$, and a test set $\mathbf{x}_* \stackrel{\text{def}}{=} (x_{*1}, \dots, x_{*M})^\top$, the posterior predictive distribution of $\mathbf{f}_* \stackrel{\text{def}}{=} (f(x_{*1}), \dots, f(x_{*M}))^\top$ given \mathcal{D} is

$$\begin{aligned} \mathbf{f}_* \mid \mathbf{x}, \mathbf{y}, \mathbf{x}_* &\sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), \quad \text{where} \\ \boldsymbol{\mu}_* &= m(\mathbf{x}_*) + k(\mathbf{x}_*, \mathbf{x}^\top)(\mathbf{K} + \gamma \mathbf{I}_N)^{-1}(\mathbf{y} - m(\mathbf{x})), \\ \boldsymbol{\Sigma}_* &= k(\mathbf{x}_*, \mathbf{x}_*^\top) - k(\mathbf{x}_*, \mathbf{x}^\top)(\mathbf{K} + \gamma \mathbf{I}_N)^{-1}k(\mathbf{x}, \mathbf{x}_*^\top). \end{aligned}$$

and where \mathbf{x} and \mathbf{x}_* are stacked training and test input, respectively. We use the notation \mathbf{K} for the matrix $\mathbf{K} = k(\mathbf{x}, \mathbf{x}^\top)$ which we assume is invertible, and we will notationally omit the conditioning on \mathbf{x} and \mathbf{x}_* . For classification, we assume the conditional distribution $y \mid f(x)$ is a Bernoulli distribution with probability $\sigma(f(x))$ where $\sigma(\cdot)$ is the logistic function and $f \sim \mathcal{GP}(m, k)$.

We first investigate the regression task, and find that for d -GPSD, the behavior of the mean function is equivalent to the kernel ridge regression solution from [Mobahi et al. \(2020\)](#) and Paper A. However, for ρ -GPSD, if we allow the noise terms γ_t to be different in each step indexed by $t = 1, 2, \dots$, then the distillation procedure yields a sequence of GP models, $\mathcal{GP}(m_t, k_t)$, where the mean and covariance functions satisfy the recursions

$$m_{t+1}(x) = m_t(x) + k_t(x, \mathbf{x}^\top)[\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1}(\mathbf{y} - m_t(\mathbf{x})) \quad (13)$$

$$k_{t+1}(x, y) = k_t(x, y) - k_t(x, \mathbf{x}^\top)[\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1}k_t(\mathbf{x}, y) \quad (14)$$

with $\mathbf{K}_t \stackrel{\text{def}}{=} k_t(\mathbf{x}, \mathbf{x}^\top)$, $m_0(x) \stackrel{\text{def}}{=} 0$, $k_0(x, y) \stackrel{\text{def}}{=} k(x, y)$ and $\gamma_t > 0$. Solving these recursions for t steps with $\gamma_0, \gamma_1, \dots, \gamma_{t-1}$ is equivalent to an ordinary GP regression model with penalty parameter $1/\gamma_t^-$ where $\gamma_t^- \stackrel{\text{def}}{=} \sum_{s=0}^t 1/\gamma_s$ and $\gamma_{-1}^- \stackrel{\text{def}}{=} 0$. In particular, for $t = 1, 2, \dots$ we show in Theorem B.3 that

$$\begin{aligned} m_t(x) &= k_0(x, \mathbf{x}^\top)(\mathbf{K} + \mathbf{I}_N/\gamma_{t-1}^-)^{-1}\mathbf{y} \\ k_t(x, y) &= k_0(x, y) - k_0(x, \mathbf{x}^\top)(\mathbf{K} + \mathbf{I}_N/\gamma_{t-1}^-)^{-1}k_0(\mathbf{x}, y). \end{aligned} \quad (15)$$

Additionally, we show that if we fix $\gamma_t = \gamma$ for all $t = 1, 2, \dots$ then fitting an ordinary GP regression model on a dataset, \mathcal{D}_t , consisting of t replicates of the original dataset yields the same solution as t steps of self-distillation, in a sense made precise in Corollary B.4.

Next, we investigate the classification setting, and as usual for GP classification we need to approximate the intractable $\mathbf{f} \mid \mathbf{y}$. We will use the Laplace approximation throughout ([Rasmussen and Williams, 2006](#)). For the ρ -GPSD setting, we find that t steps of probabilistic distillation on a replicated dataset, \mathcal{D}_t , correspond to an ordinary GP classification model with the covariance function scaled by t . Furthermore, t iterations of probabilistic distillation on the original dataset yield approximately the same posterior

as a GP classification model with the covariance function scaled by t . For the d -GPSD setting, we need additional care. In particular, while the predictions of the first GP classification model are deterministic, they represent probabilities and are in $[0, 1]$ rather than binary classes in $\{0, 1\}$. Thus, using the Bernoulli likelihood for distillation would yield a misspecified model, and we need to reformulate the GP model in the distillation steps. A natural choice of likelihood function would be the beta distribution (with equal parameters), but compared to the Bernoulli distribution, the parameter and variable switch places, making the analysis of such a setting peculiar. Thus, we propose to use the *Continuous* Bernoulli by [Loaiza-Ganem and Cunningham \(2019\)](#), denoted by $\mathcal{CB}(\lambda)$ for some probability parameter λ , which is a more natural extension on the Bernoulli distribution to continuous parameters compared to the beta distribution. This allows us to naturally model soft predictions, but to do so we first derive some properties of the normalizing term, $C(\lambda)$ of a $\mathcal{CB}(\lambda)$ distribution. In particular, let $\lambda = \sigma(a)$ for some $a \in \mathbb{R}$, then by utilizing trigonometric functions, we find simple expressions for both $C(\sigma(a))$, as well as the first- and second-order derivatives of $\log(C(\sigma(a)))$ when $a \neq 0$,

$$\begin{aligned} C(\lambda) &= a \operatorname{coth}\left(\frac{a}{2}\right), \\ \frac{d}{da} \log(C(\lambda)) &= \frac{1}{a} - \frac{1}{\sinh(a)}, \\ \frac{d^2}{d^2a} \log(C(\lambda)) &= -\frac{1}{a^2} + \frac{\operatorname{coth}(a)}{\sinh(a)}, \end{aligned}$$

where coth , and \sinh are the hyperbolic cotangent, and sine functions, respectively. See Proposition B.5 for details and the results when $a = 0$. Based on these results, we can apply the Laplace approximation and empirically find that this has a significant effect on the model predictions, especially compared to an improperly specified model using the binomial on soft targets.

While ideally, we would analyze properties of self-distillation with neural networks, we resolve to derive properties of self-distillation in the rigorous and tractable Gaussian Process models, which translates to certain wide neural networks through the NNGP ([Neal, 1996](#); [Matthews et al., 2018](#); [Novak et al., 2019](#)). Thus, intuitively t steps of self-distillation with wide neural networks should be (at least approximately) equivalent to t steps of self-distillation with the corresponding Gaussian Process model, the behavior of which is described through the results in Paper B.

3.3 Paper C: Distilling from Similar Tasks for Transfer Learning on a Budget

It has been observed that the performance of neural networks typically improves with the size of the model and the amount of data used for training ([Sun et al., 2017](#); [Zhai et al., 2022](#); [Kolesnikov et al., 2020](#)). This however is a challenge for specialized tasks in e.g. medicine (x-ray images) or science (e.g. satellite images), where annotated data and the available compute (both for training and inference) are often sparse. Thus, the aim of Paper C is to determine how to train accurate models on tight data and compute budgets without fine-tuning large pre-trained models.

A common attempt at alleviating the challenge of limited annotated data is by fine-tuning large models that are pre-trained on some other task. Unfortunately, evidence suggests that fine-tuning large models performs the best ([Zhai et al., 2022](#); [Dehghani](#)

et al., 2023; Kolesnikov et al., 2020), and even fine-tuning large models can be prohibitively expensive. Thus, specialized applications are left with a trade-off between expensive computational costs or inferior predictive performance from smaller models. Thus, to avoid fine-tuning large pre-trained models, we propose to utilize the large amount of publicly available models trained on specialized tasks in a semi-supervised cross-domain distillation procedure.

In particular, we assume a semi-supervised setting in which we have a target task specified by a limited set of annotated data \mathcal{D}_τ^l , but readily available unannotated data, \mathcal{D}_τ^u , associated with the target task. Furthermore, we assume the ability to perform inference with a set $\mathcal{S} = \{\mathcal{M}_s\}_{s=1}^S$ of S different *source* models, \mathcal{M}_s , of various architectures and trained on source tasks different from the target task. We assume all models \mathcal{M}_s are classification models that can be parameterized as a feature extractor ϕ_s followed by a classifier head, h_s , i.e. written as $M_s = h_s \circ \phi_s$.

The main idea of our methods is based on the following observation. Despite a source and target model typically do not share the same label space (or even dimension), the source model can still be informative of the target task if the label space of the source model and the true target tasks labels align well. Thus, we can ignore the actual interpretation of the source label space and utilize the pseudo-labels provided in the source space as proxies for the true labels in the target label space. For instance, if the target task aims at classifying dog breeds, and a source model specialized in cat breeds (or satellite images for that matter) consistently classifies certain dog breeds as certain cat breeds, then this classification (although nonsensical for interpretation) is informative on how to classify the dog breeds.

Concretely, we propose two methods `DISTILLNEAREST` and `DISTILLWEIGHTED`. The general strategy entails first computing a measure of “task similarity” between each source model and the target task, capturing whether the source model is discriminative of the target task. Then the source models are ranked based on the task similarity and for `DISTILLNEAREST` the most similar source model is selected, while for `DISTILLWEIGHTED` each source model is assigned a relative weight α_s based on the computed task similarity measure, such that $\sum_{s=1}^S \alpha_s = 1$. We then propose to minimize the objective function

$$\mathcal{L}_{\text{multi}} \stackrel{\text{def}}{=} \lambda \mathcal{L}^{\text{labeled}} + (1 - \lambda) \sum_{s=1}^S \alpha_s \mathcal{L}_s^{\text{distill}}, \quad (16)$$

where $\lambda, \alpha_1, \dots, \alpha_S \in [0, 1]$, and the first loss function is the standard supervised objective over the labeled data,

$$\mathcal{L}^{\text{labeled}} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_\tau^l|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_\tau^l} \ell_{CE}(h_\tau(\phi_\tau(\mathbf{x})), \mathbf{y}), \quad (17)$$

where $\ell_{CE}(\cdot, \cdot)$ is the cross-entropy loss. The second term consists of distillation losses over the unlabeled data,

$$\mathcal{L}_s^{\text{distill}} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_\tau^u|} \sum_{\mathbf{x} \in \mathcal{D}_\tau^u} \ell_{CE}(h_\tau^s(\phi_\tau(\mathbf{x})), \mathcal{M}_s(\mathbf{x})), \quad (18)$$

but since the source and target tasks do not share the same label space additional classifier heads, h_τ^s , mapping the features from the target task feature extractor, ϕ_τ , to

the label space of the source task are introduced. These additional classifier heads are all discarded after training, leaving the architecture of the original target model unchanged at inference time. We note, that for `DISTILLNEAREST` we have $\alpha_{s^*} = 1$, where s^* corresponds to the index of the single highest-ranked source model, and all other weights are zero.

Thus, it remains to determine how to define the task similarity measure such that it is informative of the discriminative power of a source model on a target task, but also dimension agnostic, since various source models produce predictions in different label spaces. We identify existing measures such as the PARC, CKA, and RSA (Cortes et al., 2012; Kornblith et al., 2019; Dwivedi and Roig, 2019; Bolya et al., 2021) that are dimension agnostic measures of similarity between similarity matrices and we find that these measures correlate well with the predictive performance of our target model after distillation. Thus, we use PARC as default and observe that the highest-ranked model by PARC is often also the best source model for distillation.

We evaluate the accuracy of both `DISTILLNEAREST` and `DISTILLWEIGHTED` over a set of 8 different target tasks, each with 28 different source models, and show superior performance averaged over all tasks compared to multiple baselines including fine-tuning an ImageNet pre-trained target model and the strong semi-supervised baseline FixMatch (Sohn et al., 2020). Furthermore, we find that while `DISTILLNEAREST` shows great improvements, it assumes a single optimal source model is available for the target task. This assumption might not be satisfied in practice and in such cases `DISTILLWEIGHTED` improves on or matches the performance of `DISTILLNEAREST`, by the ability to combine information from multiple (weighted) source models. In the paper, and the associated supplementary material, we perform numerous ablation studies and find that while our method is simple it performs well and is robust to hyperparameter selection. Thus, we find that under computational constraints and limited annotated data, our methods outperform strong baselines across different target tasks.

3.4 Paper D: A Quest for Perfect Teacher-Student Agreement in Knowledge Distillation

The common tale of why distillation works is that the student learns to match the predictions of the teacher, and thus improves over a student model trained supervised (Bucila et al., 2006; Hinton et al., 2015). However, Stanton et al. (2021) and Beyer et al. (2022) questions this understanding. In particular, Stanton et al. (2021) considers the agreement between the teacher and student under various conditions and shows that a randomly initialized student trained with distillation does not match the teacher predictions very often (only $\approx 80\%$ of the time) even though it has the model capacity to perfectly do so. However, when initialized close enough to the trained teacher, it can perfectly match the teacher. Beyer et al. (2022) finds that students trained with distillation benefit from excessively long training durations combined with very strong augmentation and interpolation between training samples to better match the function of the teacher.

Hence, the common conception of why distillation works appears to be ungrounded, and thus, in this working paper, we perform a set of experiments aimed at understanding under which conditions we do in fact obtain a perfect teacher-student agreement. If we can provide such conditions, we can adjust the conditions to imitate more realistic

distillation scenarios and get a better understanding of how and when distillation works.

Therefore, the starting point of Paper D is to investigate the self-distillation setting, where we explicitly know the trained teacher weights, and that the student has the functional capacity to perfectly match this teacher. In particular, we consider the knowledge distillation loss as formulated by [Hinton et al. \(2015\)](#) and presented in (4), but initialize our student models with weights interpolated between the teacher initialization and the trained teacher weights. Through multiple experiments, we find supporting evidence for the excessive training durations of [Beyer et al. \(2022\)](#). Additionally, we recover the observation of [Stanton et al. \(2021\)](#) that student initialization is highly important in order to achieve high agreement. In particular, students initialized slightly toward the teacher achieve notably higher agreement than students initialized at the teacher initialization.

Interesting directions for future research in this paper are to investigate other means of student initialization such as interpolation between random weights and trained teacher weights. Furthermore, analysis of the possible confounding effect of teacher accuracy on the achieved agreement is also of great importance.

3.5 Paper E: Automatic Sleep Scoring using Patient-Specific Ensemble Models and Knowledge Distillation for Ear-EEG Data

Knowledge distillation has shown great empirical results on controlled benchmark datasets. However, its effectiveness in less controlled settings has not been well-studied. Thus, the aim of Paper E is to illustrate the effectiveness of knowledge distillation in an automatic sleep scoring task, which is a challenging yet important problem due to the high subject variability of sleep patterns ([Mikkelsen et al., 2019, 2021, 2022](#)).

Specifically, we consider 4 full nights of ear-EEG measurements associated with 20 different subjects. Furthermore, half the subjects also have 12 additional full nights of ear-EEG measurements without annotations. Each night is split into a sequence of 30-second subsequences, and one of five sleep stages is manually assigned to each epoch by a clinician ([Mikkelsen et al., 2019](#); [Berry et al., 2017](#)). Thus, a single night of 8 hours of measurements corresponds to approximately 960 subsequences that need manual annotation. This time-consuming and labor-intensive task quickly makes such manual annotation financially prohibitive, and the need for reliable, effective, and accurate automatic sleep-scoring techniques is immediate.

We propose a two-phase approach for automatic sleep scoring using knowledge distillation by utilization of ensemble models and unannotated personal measurements. In the first phase, we train multiple neural network models of the specialized architecture SeqSleepNet ([Phan et al., 2019](#)). These models are then combined into an ensemble model, and we observe a monotonically increasing predictive performance (measured by Cohen’s kappa) with the size of the ensemble as well as a significant improvement over a single model. However, the cumbersome ensemble model is computationally expensive to deploy. Thus, we employ knowledge distillation to reduce the computational costs of model inference with little drop in predictive performance (namely Cohen’s kappa).

In the second phase, we use knowledge distillation to distill the ensemble model into a single SeqSleepNet model. However, due to the flexibility of the knowledge distillation setup, we are able to utilize unannotated data in the distillation procedure. We repeat the

distillation procedure with different amounts and types of data, both to investigate the possible performance gains of different types of data and to simulate realistic settings. In particular, we consider *general* students and various *personal* students. For the former, we utilize 108 nights of unannotated samples (i.e. all unannotated samples not associated with the test subject) and are able to recover about 40% of the performance improvement by the ensemble model compared to the single-model baseline. For the latter, we consider 4 different choices of unannotated samples: using the 4 test nights without manual annotations along with a) no other nights, b) the 12 unannotated nights for the test subject, c) all 120 unannotated nights, or d) only using the 12 unannotated nights associated with the test subject and not the 4 test nights. Evaluated over the 10 subjects with unannotated nights, we observe a consistent performance increase compared to the baseline, even surpassing the ensemble model when such is constructed of 2 base models. All these student models require identical compute costs to a baseline model at inference time.

We further support our findings with ablation studies on the choice of distillation temperature and the number of unannotated nights. We find that the performance of the distilled model improves with a small distillation temperature and monotonically increases with more unannotated nights. However, the improvements associated with additional nights diminish after about 10-12 nights.

Thus, Paper E demonstrates that knowledge distillation is a promising approach for automatic sleep scoring and illustrates a promising way to utilize large amounts of cheap unannotated data for personalized models.

Bibliography

- Aghli, N. and Ribeiro, E. (2021). Combining weight pruning and knowledge distillation for CNN compression. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 3185–3192. Cited on page 10.
- Ahn, S., Hu, S. X., Damianou, A., Lawrence, N. D., and Dai, Z. (2019). Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9155–9163. Cited on page 9.
- Anil, R., Gupta, V., Koren, T., Regan, K., and Singer, Y. (2020). Scalable Second Order Optimization for Deep Learning. *arXiv preprint arXiv:2002.09018*. Cited on page 3.
- Anil, R., Pereyra, G., Passos, A., Ormandi, R., Dahl, G. E., and Hinton, G. E. (2018). Large scale distributed neural network training through online distillation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. Cited on page 10.
- Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, volume 32. Cited on pages 4.
- Ba, J. L. and Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems*, volume 27, pages 2654–2662. Cited on page 5.

Introduction

- Berry, R. B., Brooks, R., Gamaldo, C., Harding, S. M., Lloyd, R. M., Quan, S. F., Troester, M. T., and Vaughn, B. V. (2017). AASM Scoring Manual Updates for 2017 (Version 2.4). *Journal of Clinical Sleep Medicine*, 13(5). Cited on page 18.
- Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. (2019). Mixmatch: A holistic approach to semi-supervised learning. In *Advances in neural information processing systems*, volume 32. Cited on page 5.
- Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., and Kolesnikov, A. (2022). Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10915–10924. Cited on pages 6, 11, 17, and 18.
- Bolya, D., Mittapalli, R., and Hoffman, J. (2021). Scalable Diverse Model Selection for Accessible Transfer Learning. *arXiv preprint arXiv:2111.06977*. Cited on page 17.
- Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pages 535–541. Cited on pages 5 and 17.
- Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*. Cited on page 4.
- Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., and Tian, Q. (2019). Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3514–3522. Cited on page 10.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *37th International Conference on Machine Learning (ICML)*, pages 1575–1585. Cited on pages 4 and 11.
- Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. (2020b). Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, pages 1–17. Cited on pages 4 and 11.
- Chen, X., Fan, H., Girshick, R., and He, K. (2020c). Improved Baselines with Momentum Contrastive Learning. *arXiv preprint arXiv:2003.04297*. Cited on pages 4 and 11.
- Cho, J. H. and Hariharan, B. (2019). On the Efficacy of Knowledge Distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Cited on page 10.
- Cortes, C., Mohri, M., and Rostamizadeh, A. (2012). Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828. Cited on page 17.
- Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M. P., Gritsenko, A., Birodkar, V., Vasconcelos, C., Tay, Y., Mensink, T.,

- Kolesnikov, A., Pavetić, F., Tran, D., Kipf, T., Lučić, M., Zhai, X., Keysers, D., Harmsen, J., and Houlsby, N. (2023). Scaling Vision Transformers to 22 Billion Parameters. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 7480–7512. Cited on page 15.
- Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. (2019a). Gradient descent finds global minima of deep neural networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 3003–3048. Cited on page 4.
- Du, S. S., Poczós, B., Zhai, X., and Singh, A. (2019b). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*. Cited on page 4.
- Dwivedi, K. and Roig, G. (2019). Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 12379–12388. Cited on page 17.
- Faghri, F., Pouransari, H., Mehta, S., Farajtabar, M., Farhadi, A., Rastegari, M., and Tuzel, O. (2023). Reinforce Data, Multiply Impact: Improved Model Accuracy and Robustness with Dataset Reinforcement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17032–17043. Cited on page 11.
- Fang, G., Song, J., Shen, C., Wang, X., Chen, D., and Song, M. (2019). Data-Free Adversarial Distillation. *arXiv preprint arXiv:1912.11006*. Cited on page 10.
- Fukuda, T., Suzuki, M., Kurata, G., Thomas, S., Cui, J., and Ramabhadran, B. (2017). Efficient knowledge distillation from an ensemble of teachers. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 3697–3701. Cited on pages 10.
- Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20(3-4):121–136. Cited on page 2.
- Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born Again Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1607–1616. PMLR. Cited on pages 8 and 10.
- Gao, M., Shen, Y., Li, Q., and Loy, C. C. (2020). Residual Knowledge Distillation. *arXiv preprint arXiv:2002.09168*. Cited on page 8.
- Gao, Y., Zhuang, J. X., Lin, S., Cheng, H., Sun, X., Li, K., and Shen, C. (2022). DisCo: Remediating Self-supervised Learning on Lightweight Models with Distilled Contrastive Learning. In *European Conference on Computer Vision (ECCV)*, pages 237–253. Cited on page 11.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. Cited on pages 1, 2, and 4.
- Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6):1789–1819. Cited on page 8.
- Grill, J. B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. In *Advances in Neural Information Processing Systems*. Cited on pages 4 and 11.

- Gupta, S., Hoffman, J., and Malik, J. (2016). Cross Modal Distillation for Supervision Transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836. Cited on pages 10.
- Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: data mining, inference, and prediction*. Springer Series in Statistics. Springer New York Inc., second edition. Cited on pages 2.
- He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9726–9735. Cited on page 4.
- Hendrycks, D. and Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*. Cited on page 2.
- Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., and Choi, J. Y. (2019). A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1921–1930. Cited on pages 9.
- Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural Networks for Machine Learning (RMSProp). Cited on page 3.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*. Cited on pages 5, 6, 7, 8, 9, 10, 11, 17, and 18.
- Hron, J., Bahri, Y., Sohl-Dickstein, J., and Novak, R. (2020). Infinite attention: NNGP and NTK for deep attention networks. *arXiv preprint arXiv:2006.10540*. Cited on page 4.
- Huber, P. J. (1964). Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101. Cited on page 9.
- Islam, A., Chen, C.-F. R., Panda, R., Karlinsky, L., Feris, R., and Radke, R. J. (2021). Dynamic Distillation Network for Cross-Domain Few-Shot Recognition with Unlabeled Data. In *Advances in Neural Information Processing Systems*, volume 34, pages 3584–3595. Cited on page 5.
- Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pages 8571–8580. Cited on pages 4.
- Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1317–1327. Cited on pages 8 and 10.
- Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. Cited on page 3.
- Kolesnikov, A., Beyler, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big Transfer (BiT): General Visual Representation Learning. In *Proceedings of the 16th European Conference on Computer Vision*, pages 491–507. Cited on pages 15 and 16.

- Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of neural network representations revisited. In *36th International Conference on Machine Learning (ICML)*, pages 6156–6175. Cited on pages 9 and 17.
- Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning (ICML)*, volume 3. Cited on page 5.
- Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2018a). Deep neural networks as Gaussian processes. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. Cited on page 4.
- Lee, J., Xiao, L., Schoenholz, S. S., Novak, Y. B. R., Sohl-Dickstein, J., Pennington, J., Bahri, Y., Dec, M. L., and Brain, G. (2019). Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In *Advances in Neural Information Processing Systems*. Cited on pages 4.
- Lee, S. H., Kim, D. H., and Song, B. C. (2018b). Self-supervised knowledge distillation using singular value decomposition. In *Proceedings of the European conference on computer vision (ECCV)*. Cited on page 9.
- Liu, Y., Zhang, W., and Wang, J. (2020). Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113. Cited on pages 10.
- Loaiza-Ganem, G. and Cunningham, J. P. (2019). The continuous bernoulli: Fixing a pervasive error in variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 32. Cited on page 15.
- Lopes, R. G., Fenu, S., and Starner, T. (2017). Data-Free Knowledge Distillation for Deep Neural Networks. *arXiv preprint arXiv:1710.07535*. Cited on page 10.
- Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Cited on page 3.
- Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28. Cited on page 2.
- Matthews, A. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:arXiv:1804.11271v2*. Cited on pages 4 and 15.
- Mikkelsen, K., Phan, H., Rank, M. L., Hemmsen, M. C., De Vos, M., and Kidmose, P. (2021). Sleep Monitoring Using Ear-Centered Setups: Investigating the Influence from Electrode Configurations. *IEEE Transactions on Biomedical Engineering*, 69(5):1564–1572. Cited on page 18.
- Mikkelsen, K. B., Tabar, Y. R., Kappel, S. L., Christensen, C. B., Toft, H. O., Hemmsen, M. C., Rank, M. L., Otto, M., and Kidmose, P. (2019). Accurate Whole-Night Sleep Monitoring with Dry-Contact Ear-EEG. *Scientific Reports*, 9(1). Cited on pages 18.

Introduction

- Mikkelsen, K. B., Tabar, Y. R., Toft, H. O., Hemmsen, M. C., Rank, M. L., and Kidmose, P. (2022). Self-applied ear-EEG for sleep monitoring at home. In *Proceedings of the 44th annual international conference of the IEEE engineering in Medicine & Biology Society (EMBC)*, pages 3135–3138. Cited on page 18.
- Mirzadeh, S.-I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. (2020). Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198. Cited on page 10.
- Mobahi, H., Farajtabar, M., and Bartlett, P. L. (2020). Self-Distillation Amplifies Regularization in Hilbert Space. In *Advances in Neural Information Processing Systems*. Cited on pages 10, 12, 13, and 14.
- Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*. Cited on page 2.
- Nayak, G. K., Mopuri, K. R., Shaj, V., Babu, R. V., and Chakraborty, A. (2019). Zero-shot knowledge distillation in deep networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 8317–8325. PMLR. Cited on page 10.
- Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer. Cited on pages 4 and 15.
- Nguyen, T., Raghu, M., and Kornblith, S. (2021). Do Wide and Deep Networks Learn the Same Things? Uncovering How Neural Network Representations Vary With Width and Depth. *Proceedings of the 9th International Conference on Learning Representations (ICLR)*. Cited on page 9.
- Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2019). Bayesian deep convolutional networks with many channels are Gaussian processes. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on pages 4 and 15.
- Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational Knowledge Distillation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Cited on page 9.
- Passalis, N., Tzelepi, M., and Tefas, A. (2021). Probabilistic Knowledge Transfer for Lightweight Deep Representation Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):2030–2039. Cited on page 9.
- Phan, H., Andreotti, F., Cooray, N., Chen, O. Y., and De Vos, M. (2019). SeqSleepNet: End-to-End Hierarchical Recurrent Neural Network for Sequence-to-Sequence Automatic Sleep Staging. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(3):400–410. Cited on page 18.
- Phoo, C. P. and Hariharan, B. (2021). Self-training For Few-shot Transfer Across Extreme Task Differences. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Cited on pages 5 and 10.

- Polino, A., Pascanu, R., and Alistarh, D. (2018). Model compression via distillation and quantization. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. Cited on page 10.
- Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. (2021). Do Vision Transformers See Like Convolutional Neural Networks? In *Advances in Neural Information Processing Systems*. Cited on page 9.
- Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*, volume 1. MIT Press, Cambridge, MA. Cited on page 14.
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). FitNets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. Cited on page 9.
- Schmidt, R. M., Schneider, F., and Hennig, P. (2021). Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 9367–9376. PMLR. Cited on page 3.
- Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Proceedings of the IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. Cited on page 3.
- Smith, L. N. (2018). A Disciplined Approach To Neural Network Hyper-Parameters: Part 1 – Learning Rate, Batch Size, Momentum, and Weight Decay. *arXiv preprint arXiv:1803.09820*. Cited on page 3.
- Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959. Cited on page 3.
- Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. (2020). FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *Advances in Neural Information Processing Systems*, volume 33, pages 596–608. Cited on pages 5 and 17.
- Srinivas, S. and Fleuret, F. (2018). Knowledge transfer with jacobian matching. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 7515–7523. Cited on page 8.
- Stanton, S., Izmailov, P., Kirichenko, P., Alemi, A. A., and Wilson, A. G. (2021). Does Knowledge Distillation Really Work? In *Advances in Neural Information Processing Systems*, pages 6906–6919. Cited on pages 11, 17, and 18.
- Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852. Cited on page 15.
- Tan, X., Ren, Y., He, D., Qin, T., Zhao, Z., and Liu, T. Y. (2019). Multilingual neural machine translation with knowledge distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on page 10.

- Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, volume 30, pages 1196–1205. Cited on pages 5 and 8.
- Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive Representation Distillation. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. Cited on pages 10 and 11.
- Tung, F. and Mori, G. (2019). Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1374. Cited on pages 9.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5999–6009. Cited on page 2.
- Wang, L. and Yoon, K.-J. (2021). Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):3048–3068. Cited on page 8.
- Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2020a). Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Cited on page 5.
- Xie, Q., Luong, M. T., Hovy, E., and Le, Q. V. (2020b). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10684–10695. Cited on page 10.
- Yang, C., Xie, L., Qiao, S., and Yuille, A. (2018). Knowledge Distillation in Generations: More Tolerant Teachers Educate Better Students. *arXiv preprint arXiv:1805.05551*. Cited on page 10.
- Yang, G. (2019). Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation. *arXiv preprint arXiv:1902.04760*. Cited on pages 4.
- Yang, G. and Hu, E. J. (2021). Tensor Programs IV: Feature Learning in Infinite-Width Neural Networks. *arXiv preprint arXiv:2310.02244*. Cited on page 4.
- Yang, J., Martinez, B., Bulat, A., and Tzimiropoulos, G. (2020). Knowledge distillation via adaptive instance normalization. *arXiv preprint arXiv:2003.04289*. Cited on page 8.
- Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7130–7138. Cited on page 9.
- You, S., Xu, C., Xu, C., and Tao, D. (2017). Learning from Multiple Teacher Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*, pages 1285–1294. Cited on page 10.
- Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Cited on page 9.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2022). Scaling Vision Transformers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 12094–12103. Cited on pages [15](#).

Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation

PUBLISHED IN ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 34 (NEURIPS)

Kenneth Borup
Aarhus University

Lars Nørvang Andersen
Aarhus University

Abstract. Knowledge distillation is classically a procedure where a neural network is trained on the output of another network along with the original targets in order to transfer knowledge between the architectures. The special case of self-distillation, where the network architectures are identical, has been observed to improve generalization accuracy. In this paper, we consider an iterative variant of self-distillation in a kernel regression setting, in which successive steps incorporate both model outputs and the ground-truth targets. This allows us to provide the first theoretical results on the importance of using the weighted ground-truth targets in self-distillation. Our focus is on fitting nonlinear functions to training data with a weighted mean square error objective function suitable for distillation, subject to ℓ_2 regularization of the model parameters. We show that any such function obtained with self-distillation can be calculated directly as a function of the initial fit and that infinite distillation steps yield the same optimization problem as the original with amplified regularization. Furthermore, we provide a closed-form solution for the optimal choice of weighting parameter at each step and show how to efficiently estimate this weighting parameter for deep learning and significantly reduce the computational requirements compared to a grid search.

A.1 Introduction

Knowledge distillation, most commonly known from [Hinton et al. \(2015\)](#), is a procedure to transfer *knowledge* from one neural network (teacher) to another neural network (student).¹ Often the student has fewer parameters than the teacher, and the procedure can be seen as a model compression technique. Originally, the distillation procedure achieves the knowledge transfer by training the student network using the original training targets, denoted as ground-truth targets, as well as a softened distribution of logits from the (already trained and fixed) teacher network.² Since the popularization of knowledge distillation by [Hinton et al. \(2015\)](#), the idea of knowledge distillation has been extended to a variety of settings.³ This paper will focus on the special case where the teacher and student are of identical architecture, called self-distillation, and where the aim is to improve predictive performance, rather than compressing the model.

The idea of self-distillation is to use outputs from a trained model together with the original targets as new targets for retraining the same model from scratch. We refer to this as one step of self-distillation, and one can iterate this procedure for multiple distillation steps (see Figure A.1). Empirically, it has been shown that this procedure often generalizes better than the model trained merely on the original targets, and achieves higher predictive performance on validation data, despite no additional information being provided during training ([Furlanello et al., 2018](#); [Ahn et al., 2019](#); [Yang et al., 2018](#)).

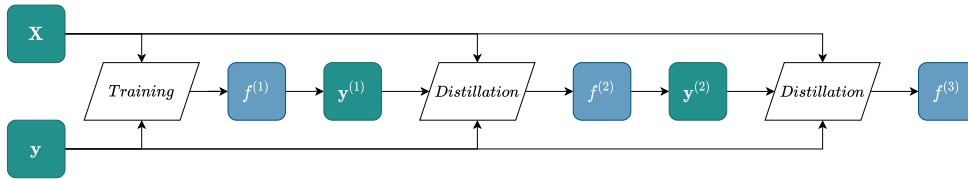


Figure A.1: Illustration of self-distillation for two steps after the initial training, where we use the notation $f^{(\tau)} = f(\cdot, \hat{\beta}^{(\tau)})$. See Section A.3 for details.

Modern deep neural networks are often trained in the over-parameterized regime, where the number of trainable parameters highly exceeds the number of training samples. Under simple first-order methods such as gradient descent, such large networks can fit any target, but in order to generalize well, such overfitting is usually undesirable ([Zhang et al., 2017](#); [Nakkiran et al., 2020](#)). Thus, some type of regularization is typically imposed during training, in order to avoid overfitting. A common choice is to add an ℓ_2 -regularization⁴ term to our objective function, which has been shown to perform comparably to early-stopping gradient descent training ([Yao et al., 2007](#)). However, in the theoretical study of the over-parameterized regime, regularization is often overlooked, but recent results have shown a connection between wide neural networks and kernel ridge regression through the Neural Tangent Kernel (NTK) ([Lee et al., 2019, 2020](#); [Hu](#)

¹ Often knowledge distillation is also referred to under the name *Teacher-Student learning*.

² We will refer to the weighted outputs of the penultimate layer, i.e. pre-activation of the last layer, as logits.

³ See Section A.2 for a brief overview, or see [Wang and Yoon \(2021\)](#) for a more exhaustive survey

⁴ With slight differences, ℓ_2 regularization is often referred to as weight decay and ridge regularization in deep learning and statistical learning literature, respectively. See e.g. [Loshchilov and Hutter \(2019\)](#) for details.

et al., 2020). We briefly elaborate on this connection in Section A.IV, which motivates our problem setup and connection to deep learning in Section A.5.

Our Contributions. Through a theoretical analysis, we show that

- the solution at any distillation step can easily be calculated as a function of the initial fit, and infinitely many steps of self-distillation (with fixed distillation weight) correspond to solving the usual kernel ridge regression problem with a specific amplified regularization parameter when the distillation weight is non-zero,
- for fixed distillation weights, self-distillation amplifies the regularization at each distillation step, and the ground-truth targets dampen the sparsification and regularization of the self-distilled solutions, ensuring non-zero solutions for any number of distillation steps,
- the optimal distillation weight has a closed-form solution for kernel ridge regression, and can be estimated efficiently for neural networks compared to a grid search.

Proofs of all our results can be found in Supplementary Material A.I, and code to reproduce our illustrative example in Section A.4.5 and experimental results in Section A.II can be found at https://github.com/Kennethborup/self_distillation.

A.2 Related Work

The idea of knowledge distillation dates back to Bucila et al. (2006), and was later brought to the deep learning setting by Ba and Caruana (2014) and more recently popularized by Hinton et al. (2015) in the context of compressing neural networks. Since the original formulation, various extensions have been proposed. Some approaches focus on matching the teacher and student models on statistics other than the distribution of the logits, such as intermediate representations (Romero et al., 2015), spacial attention maps (Zagoruyko and Komodakis, 2017), Jacobians (Srinivas and Fleuret, 2018), Gram matrices (Yim et al., 2017), or relational information between teacher outputs (Park et al., 2019). Other extensions focus on developing the transfer procedure, such as self-distillation (Furlanello et al., 2018), data-free distillation (Lopes et al., 2017; Nayak et al., 2019; Micaelli and Storkey, 2019; Chen et al., 2019; Fang et al., 2019), data distillation (Radosavovic et al., 2018), residual knowledge distillation (Gao et al., 2020), online distillation (Anil et al., 2018) or contrastive distillation (Ahn et al., 2019; Tian et al., 2020a).

The practical benefits of knowledge distillation have been proven countless times in a variety of settings, but the theoretical justification for knowledge distillation is still highly absent. Hinton et al. (2015) conjecture that the success of knowledge distillation should be attributed to the transfer of *dark knowledge* (e.g. inter-class relationships revealed in the soft labels). Müller et al. (2019); Tang et al. (2020) support this conjecture, and argue that knowledge distillation is similar to performing adaptive label smoothing weighted by the teacher’s confidence in the predictions. Dong et al. (2019) shows the importance of early stopping when training over-parameterized neural networks for distillation purposes

by arguing that neural networks tend to fit informative and simple patterns faster than noisy signals, and knowledge distillation utilizes these simple patterns for knowledge transfer. [Abnar et al. \(2020\)](#) empirically investigate how knowledge distillation can transfer inductive biases between student and teacher models, and [Gotmare et al. \(2019\)](#) empirically shows how the dark knowledge shared by the teacher mainly is disbursed to some of the deepest layers of the teacher.

To the best of our knowledge, few papers investigate knowledge distillation from a rigorous theoretical point of view, and those that do, do so with strong assumptions on the setting. [Phuong and Lampert \(2019\)](#) ignore the ground-truth targets during distillation and furthermore assume linear models. [Mobahi et al. \(2020\)](#) investigate self-distillation in a Hilbert space setting with kernel ridge regression models where the teacher is trained on the ground-truth targets, and the student (and subsequent iterations) is only trained on the predictions from the teacher without access to the ground-truth targets. They show that self-distillation progressively limits the number of basis functions used to represent the solutions, thus eventually causing the solutions to underfit. In this paper, we build on the theoretical results of [Mobahi et al. \(2020\)](#), but we include the weighted ground-truth targets in the self-distillation procedure, where we allow the weight to depend on the self-distillation step, and show how this drastically affects the behavior and effect of self-distillation.⁵

A.3 Problem Setup

Notation. Vectors and matrices are denoted by bold-faced letters; vectors are column vectors by default, and for a vector \mathbf{a} let $[\mathbf{a}]_i$ be the i -th entry, and for a matrix \mathbf{A} let $[\mathbf{A}]_{i,j}$ be the (i, j) -th entry. Let \mathbf{I}_n denote the identity matrix of dimension n , $[k] = \{1, 2, \dots, k\}$, and let $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the ℓ_2 -norm and the Frobenius norm, respectively. Finally, for a function $h: \mathbb{R}^n \rightarrow \mathbb{R}^d$ and $\mathbf{X} \in \mathbb{R}^{m \times n}$, we denote by $h(\mathbf{X})$ the $\mathbb{R}^{m \times d}$ matrix of outcomes, where the i 'th row of $h(\mathbf{X})$ is the function applied to the i 'th row of \mathbf{X} , i.e. $[h(\mathbf{X})]_{i,\cdot} = h(\mathbf{x}_i)$.

Consider the training dataset $\mathcal{D} \subseteq \mathbb{R}^d \times \mathbb{R}$, and let $\mathcal{X} = \{\mathbf{x} \mid (\mathbf{x}, y) \in \mathcal{D}\}$ and $\mathcal{Y} = \{y \mid (\mathbf{x}, y) \in \mathcal{D}\}$ denote the inputs and targets, respectively. Let $\mathbf{X} = [\mathbf{x}_i]_{i \in [n]} \in \mathbb{R}^{n \times d}$ be the matrix of inputs, $\mathbf{y} = [y_i]_{i \in [n]}$ the vector of targets, and $\tilde{\mathbf{X}} \in \mathbb{R}^{m \times d}$, $\tilde{\mathbf{y}} \in \mathbb{R}^m$ be the matrix and vector of validation inputs and targets, respectively. Given a feature map $\varphi: \mathbb{R}^d \rightarrow \mathcal{V}$, where \mathcal{V} has dimension D , we denote by $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X}) = [\kappa(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^n \in \mathbb{R}^{n \times n}$, where $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle$, the symmetric kernel (Gram) matrix associated with the feature map φ .⁶

A.3.1 Self-Distillation of Kernel Ridge Regressions

In order to avoid overfitting our training data, we will impose a regularization term on our weights, and thus investigate the kernel ridge regression functions $f \in \mathcal{F}$ mapping $f: \mathcal{X} \rightarrow \mathcal{Y}$, to construct a solution which best approximates the true underlying data generating map and generalize well to new unseen data from this underlying map. We

⁵ In Supplementary Material A.V we relate our problem setup to [Mobahi et al. \(2020\)](#) and extend some of our results to a constrained optimization setting with a regularization functional in Hilbert space.

⁶ Since the kernel trick makes the predictions depend only on inner products in the feature space, it is not a restriction if D is infinite. However, for ease of exposition, we assume D is finite.

consider self-distillation in the kernel ridge regression setup; i.e. consider the (self-distillation) objective function

$$\mathcal{L}^{\text{distill}}(f(\mathbf{X}, \beta), \mathbf{y}_1, \mathbf{y}_2) = \frac{\alpha}{2} \|f(\mathbf{X}, \beta) - \mathbf{y}_1\|_2^2 + \frac{1-\alpha}{2} \|f(\mathbf{X}, \beta) - \mathbf{y}_2\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2, \quad (\text{A.1})$$

where $\alpha \in [0, 1]$, $\lambda > 0$, $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^n$ and $f(\mathbf{X}, \beta) = \varphi(\mathbf{X})\beta$. The objective in (A.1) is a weighted sum of two Mean Square Error (MSE) objective functions with different targets⁷ and an ℓ_2 -regularization on the model weights. Minimization of (A.1) w.r.t. β is straightforward and yields the minimizer

$$\hat{\beta} \stackrel{\text{def}}{=} \underset{\beta}{\operatorname{argmin}} \mathcal{L}^{\text{distill}}(f(\mathbf{X}, \beta), \mathbf{y}_1, \mathbf{y}_2) \quad (\text{A.2})$$

$$= \varphi(\mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha \mathbf{y}_1 + (1-\alpha) \mathbf{y}_2) \quad (\text{A.3})$$

by Woodbury's matrix identity and definition of \mathbf{K} . This solution can also be seen as a direct application of the Representer Theorem (Schölkopf et al., 2001). Let $\mathbf{y}^{(0)} \stackrel{\text{def}}{=} \mathbf{y}$, i.e. the original targets, and recursively define for the steps $\tau \geq 1$,

$$\hat{\beta}^{(\tau)} \stackrel{\text{def}}{=} \underset{\beta}{\operatorname{argmin}} \mathcal{L}^{\text{distill}}(f(\mathbf{X}, \beta), \mathbf{y}, \mathbf{y}^{(\tau-1)}) \quad (\text{A.4})$$

$$= \varphi(\mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha^{(\tau)} \mathbf{y} + (1-\alpha^{(\tau)}) \mathbf{y}^{(\tau-1)}),$$

$$f(\mathbf{x}, \hat{\beta}^{(\tau)}) \stackrel{\text{def}}{=} \varphi(\mathbf{x})^\top \hat{\beta}^{(\tau)} \quad (\text{A.5})$$

$$= \kappa(\mathbf{x}, \mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha^{(\tau)} \mathbf{y} + (1-\alpha^{(\tau)}) \mathbf{y}^{(\tau-1)}),$$

$$\mathbf{y}^{(\tau)} \stackrel{\text{def}}{=} f(\mathbf{X}, \hat{\beta}^{(\tau)}), \quad (\text{A.6})$$

for fixed $\alpha^{(\tau)} \in [0, 1]$. Notice, the initial step ($\tau = 1$) corresponds to standard training by definition and as such is independent of $\alpha^{(1)}$. Self-distillation treats the weighted average of the predictions, $\mathbf{y}^{(1)}$, from this initial model on \mathbf{X} , and the ground-truth targets, \mathbf{y} , as targets. This procedure is repeated as defined in (A.4)-(A.6) and we obtain the self-distillation procedure as illustrated in Figure A.1. Note, the special cases $\alpha^{(\tau)} = 0$ and $\alpha^{(\tau)} = 1$ correspond to merely training on the predictions from the previous step, and only training on the original targets, respectively. Thus, $\alpha^{(\tau)} = 1$ is usually not of interest, as the solution is equal to a classical kernel ridge regression, and self-distillation plays no role in this scenario. We will often consider the special case of equal weights, $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$, and if $\alpha = 0$ this corresponds to the setting investigated in Mobahi et al. (2020) in a slightly different setup. Thus, some of the following results can be seen as a generalization of Mobahi et al. (2020) to step-wise and non-zero α .

A.4 Main Results

In this section, we present our main results for finitely and infinitely many distillation steps along with a closed-form solution for the optimal $\alpha^{(\tau)}$ as well as an illustrative example highlighting the effect of the chosen sequence of $(\alpha^{(t)})$ on the solutions.

⁷ It is straightforward to verify that minimizing (A.1) and the classic MSE objective with a weighted target, i.e. $\tilde{\mathcal{L}}^{\text{distill}}(f(\mathbf{X}, \beta), \mathbf{y}_1, \mathbf{y}_2) = \frac{1}{2} \|f(\mathbf{X}, \beta) - (\alpha \mathbf{y}_1 + (1-\alpha) \mathbf{y}_2)\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2$, are equivalent and that the objective functions are equal up to the additive constant $\alpha(\alpha-1) \|\mathbf{y}_1 - \mathbf{y}_2\|_2^2$.

A.4.1 Finite Self-Distillation Steps

Our first result, which follows from straightforward computations, states that the predictions obtained after any finite number of distillation steps can be expressed directly as a function of \mathbf{y} and the kernel matrix \mathbf{K} calculated at the initial fit ($\tau = 1$).

Theorem A.1. *Let $\mathbf{y}^{(\tau)}$, $\hat{\boldsymbol{\beta}}^{(\tau)}$, and $f(\cdot, \hat{\boldsymbol{\beta}}^{(\tau)})$ be defined as above. Fix $\alpha^{(2)}, \dots, \alpha^{(\tau)} \in [0, 1)$, and let $\eta(i, \tau) \stackrel{\text{def}}{=} \prod_{j=i}^{\tau} (1 - \alpha^{(j)})$, then for $\tau \geq 1$, we have that*

$$\mathbf{y}^{(\tau)} = \left(\sum_{i=2}^{\tau} \alpha^{(i)} \eta(i+1, \tau) (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^{\tau-i+1} + \eta(2, \tau) (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^{\tau} \right) \mathbf{y}, \quad (\text{A.7})$$

$$f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(\tau)}) = \alpha^{(\tau)} f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(1)}) + (1 - \alpha^{(\tau)}) f(\mathbf{x}, \hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)}) \quad (\text{A.8})$$

for any $\mathbf{x} \in \mathbb{R}^d$, where $\hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)}$ is the minimizer in (A.4) with $\alpha^{(\tau)} = 0$.

Proof. See proof on page 48. □

Since (A.7) and (A.8) are expressed only in terms of \mathbf{K} , $(\mathbf{K} + \lambda \mathbf{I}_n)^{-1}$, $\kappa(\mathbf{x}, \mathbf{X})$, and \mathbf{y} we are able to calculate the predictions for the training data as well as for any $\mathbf{x} \in \mathbb{R}^d$ based merely on the initial fit ($\tau = 1$) without the need for any additional fits. Hence, despite the calculations of \mathbf{K} , $\kappa(\mathbf{x}, \mathbf{X})$, and especially $(\mathbf{K} + \lambda \mathbf{I}_n)^{-1}$ being (potentially) highly computationally demanding, when obtained, we can calculate any distillation step directly by the equations in Theorem A.1. Furthermore, predictions at step τ can be seen as a weighted combination of two classical ridge regression solutions, based on the original targets and the predicted targets from step $\tau - 1$, respectively. However, choosing appropriate $\alpha^{(t)}$ for $t = 2, \dots, \tau$ is non-trivial. We explore these dynamics in Section A.4.3 and A.4.4. First, we use Theorem A.1 to analyze the regularization that self-distillation progressively imposes on the solutions.

A.4.2 Effective Sparsification of Self-Distillation Solutions

We now show that we can represent the solutions as a weighted sum of basis functions and that this basis sparsifies when we increase τ , but also that the amount of sparsification depends on the choice of α . A similar sparsification result for the special case of fixed $\alpha^{(\tau)} = 0$ for $\tau \geq 1$ was proved in [Mobahi et al. \(2020\)](#), and in particular, our (A.15) generalizes equation (47) in their paper.

Using the spectral decomposition of the symmetric matrix \mathbf{K} we write $\mathbf{K} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$, where $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix with the eigenvectors of \mathbf{K} as rows and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a non-negative diagonal matrix with the associated eigenvalues in the diagonal. Inserting the diagonalization yields

$$\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1} = \mathbf{V}\mathbf{D}\mathbf{V}^\top (\mathbf{V}\mathbf{D}\mathbf{V}^\top + \lambda \mathbf{I}_n)^{-1} \quad (\text{A.9})$$

$$= \mathbf{V}\mathbf{D}(\mathbf{D} + \lambda \mathbf{I}_n)^{-1} \mathbf{V}^\top, \quad (\text{A.10})$$

where $\lambda > 0$. By straightforward calculations using (A.7) and (A.10) we have

$$\mathbf{y}^{(\tau)} = \mathbf{V}\mathbf{B}^{(\tau)}\mathbf{V}^\top\mathbf{y}, \quad \text{where} \quad (\text{A.11})$$

$$\mathbf{B}^{(\tau)} \stackrel{\text{def}}{=} \sum_{i=2}^{\tau} \alpha^{(i)} \eta(i+1, \tau) \mathbf{A}^{\tau-i+1} + \eta(2, \tau) \mathbf{A}^\tau, \quad \text{and} \quad (\text{A.12})$$

$$\mathbf{A} \stackrel{\text{def}}{=} \mathbf{D}(\mathbf{D} + \lambda \mathbf{I}_n)^{-1}, \quad (\text{A.13})$$

and $\mathbf{A}, \mathbf{B}^{(\tau)} \in \mathbb{R}^{n \times n}$ are diagonal matrices for any τ . Furthermore, by (A.11) the only part of the solution depending on τ is the diagonal matrix, $\mathbf{B}^{(\tau)}$, and in the following we show how $\mathbf{B}^{(\tau)}$ determines the effective sparsification of the solution $f(\cdot, \hat{\boldsymbol{\beta}}^{(\tau)})$.

Lemma A.2. *Let $\mathbf{B}^{(\tau)}$, and \mathbf{A} be defined as above, and let $\mathbf{B}^{(0)} \stackrel{\text{def}}{=} \mathbf{I}_n$. Then we can express $\mathbf{B}^{(\tau)}$ recursively as*

$$\mathbf{B}^{(\tau)} = \mathbf{A} \left((1 - \alpha^{(\tau)}) \mathbf{B}^{(\tau-1)} + \alpha^{(\tau)} \mathbf{I}_n \right), \quad (\text{A.14})$$

and $[\mathbf{B}^{(\tau)}]_{k,k} \in [0, 1]$ is (strictly) decreasing in τ for all $k \in [n]$ and $\tau \geq 1$ if $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$.

Proof. See proof on page 48. □

Similarly to (A.11), if we use Lemma A.2 and Theorem A.1, we can show that for any $\mathbf{x} \in \mathbb{R}^p$

$$\begin{aligned} f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(\tau)}) &= \kappa(\mathbf{x}, \mathbf{X})^\top \mathbf{V} \mathbf{D}^{-1} \mathbf{B}^{(\tau)} \mathbf{V}^\top \mathbf{y} \\ &= \mathbf{p}(\mathbf{x})^\top \mathbf{B}^{(\tau)} \mathbf{z}, \quad \text{where} \end{aligned} \quad (\text{A.15})$$

$$\begin{aligned} \mathbf{p}(\mathbf{x}) &\stackrel{\text{def}}{=} \mathbf{D}^{-1} \mathbf{V}^\top \kappa(\mathbf{x}, \mathbf{X}), \quad \text{and} \\ \mathbf{z} &\stackrel{\text{def}}{=} \mathbf{V}^\top \mathbf{y}. \end{aligned} \quad (\text{A.16})$$

Thus, the solution $f(\cdot, \hat{\boldsymbol{\beta}}^{(\tau)})$ can be represented as a weighted sum of some basis functions, where the basis functions are the components of the orthogonally transformed and scaled basis $\mathbf{p}(\mathbf{x})$, and \mathbf{z} is an orthogonally transformed vector of targets.

Now assume $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$ for any $\tau \geq 2$ for the remaining of this section. In the following we show how the behaviour of $\mathbf{B}^{(\tau)}$, and in turn also the behaviour of $f(\cdot, \hat{\boldsymbol{\beta}}^{(\tau)})$, with τ is dependent on the choice of α . Lemma A.2 not only provides a recursive formula for $\mathbf{B}^{(\tau)}$ but also shows that each diagonal element of $\mathbf{B}^{(\tau)}$ is in $[0, 1]$ and is strictly decreasing in τ , which in turn implies that the self-distillation procedure progressively shrinks the coefficients of the basis functions. Using Lemma A.2 we can now show, that not only does $\mathbf{B}^{(\tau)}$ decrease in τ , smaller elements of $\mathbf{B}^{(\tau)}$ shrink faster than larger elements for $\alpha = 0$, as we elaborate on below the theorem.

Theorem A.3. *For any pair of diagonals of \mathbf{D} , i.e. d_k and d_j , where $d_k > d_j$, we have for all $\tau \geq 1$,*

$$\frac{[\mathbf{B}^{(\tau)}]_{k,k}}{[\mathbf{B}^{(\tau)}]_{j,j}} = \begin{cases} \frac{1 + \frac{\lambda}{d_j}}{1 + \frac{\lambda}{d_k}}, & \text{for } \alpha = 1, \\ \left(\frac{1 + \frac{\lambda}{d_j}}{1 + \frac{\lambda}{d_k}} \right)^\tau, & \text{for } \alpha = 0, \end{cases} \quad (\text{A.17})$$

and if we let $\text{sgn}(\cdot)$ denote the sign function⁸, then for $\alpha \in (0, 1)$ we have that

$$\begin{aligned} & \text{sgn}\left(\frac{[\mathbf{B}^{(\tau)}]_{k,k}}{[\mathbf{B}^{(\tau)}]_{j,j}} - \frac{[\mathbf{B}^{(\tau-1)}]_{k,k}}{[\mathbf{B}^{(\tau-1)}]_{j,j}}\right) \\ &= \text{sgn}\left(\left(\left(\frac{[\mathbf{B}^{(\tau-1)}]_{k,k}}{[\mathbf{B}^{(\tau-1)}]_{j,j}} - \frac{[\mathbf{A}]_{k,k}}{[\mathbf{A}]_{j,j}}\right) \frac{[\mathbf{A}]_{j,j}}{[\mathbf{B}^{(\tau-1)}]_{k,k}([\mathbf{A}]_{k,k} - [\mathbf{A}]_{j,j})} + 1\right)^{-1} - \alpha\right). \end{aligned} \quad (\text{A.18})$$

Proof. See proof on page 49. \square

If we consider a pair of diagonals of \mathbf{D} , where $d_k > d_j$, then for $\alpha = 0$, the fraction $[\mathbf{B}^{(\tau)}]_{k,k} / [\mathbf{B}^{(\tau)}]_{j,j}$ is strictly increasing in τ , due to the r.h.s. of (A.17) inside the parenthesis being strictly larger than 1. Hence, the diagonals corresponding to smaller eigenvalues shrink faster than the larger ones as τ increases. However, for $\alpha \in (0, 1)$ we can not ensure this behavior, but at step τ we are able to predict the behavior at step $\tau + 1$, by using (A.18). Thus, when we include the ground-truth targets in our distillation procedure we do not consistently increase the regularization with each distillation step, but can potentially obtain a solution that does not sparsify any further. We now turn our attention to the question of how to pick the $\alpha^{(\tau)}$'s in an optimal manner and find that it can be done if we relax the condition that the weights are restricted to the interval $[0, 1]$.

A.4.3 Closed Form Optimal Weighting Parameter

Recall, $\tilde{\mathbf{X}} \in \mathbb{R}^{m \times d}$ is the matrix of validation inputs and $\tilde{\mathbf{y}} \in \mathbb{R}^m$ the vector of validation targets. If we allow $\alpha^{(\tau)} \in \mathbb{R}$, we can find an *optimal* $\alpha^{(\tau)}$ (which is a non-trivial function of λ) at each step τ , denoted by $\alpha^{*(\tau)}$.⁹ Here, *optimal* denotes the value for which the validation MSE is minimized. Note, $\alpha^{*(\tau)}$ is optimal for a single distillation step, but not necessarily so for multiple distillation steps, however we may consider $\alpha^{*(\tau)}$ a greedy estimate of the optimal value across multiple steps.

Theorem A.4. Fix $\tau \geq 2$, $\lambda > 0$ and $\alpha^{(2)}, \dots, \alpha^{(\tau-1)} \in \mathbb{R}$, then

$$\alpha^{*(\tau)} = \underset{\alpha^{(\tau)} \in \mathbb{R}}{\text{argmin}} \|\tilde{\mathbf{y}} - f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)})\|_2^2 \quad (\text{A.19})$$

$$= 1 - \frac{(\tilde{\mathbf{y}}_{\alpha=0}^{(\tau)} - \tilde{\mathbf{y}}^{(1)})^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^{(1)})}{\|\tilde{\mathbf{y}}_{\alpha=0}^{(\tau)} - \tilde{\mathbf{y}}^{(1)}\|_2^2} \quad (\text{A.20})$$

where $\tilde{\mathbf{y}}^{(1)} = f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(1)})$, and $\tilde{\mathbf{y}}_{\alpha=0}^{(\tau)} = f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)})$.

Proof. See proof on page 50. \square

Since neither $\tilde{\mathbf{y}}^{(1)}$ nor $\tilde{\mathbf{y}}_{\alpha=0}^{(\tau)}$ depend on the choice of $\alpha^{(\tau)}$, we can calculate $\alpha^{*(\tau)}$ recursively as presented in Algorithm 1, where $\alpha^{*(\tau)}$ has the closed form in (A.19). In combination with the diagonalization results of Section A.4.2 we can efficiently calculate the solutions. This should be compared to performing grid-search for α with

⁸ Note, we use the definition of $\text{sgn}(\cdot)$ where $\text{sgn}(0) \stackrel{\text{def}}{=} 0$.

⁹ If $\alpha^{*(\tau)} \notin [0, 1]$, the sign of either the first or second term of (A.1) becomes negative, indicating either too strong or weak regularization of the previous distillation step, and one might fear this affects distillation performance. However, simply clipping of $\alpha^{*(\tau)}$ to be in $[0, 1]$ alleviates this, at the cost of requiring a larger τ .

g equidistant values on $[0, 1]$ in order to approximate the optimal α , which requires $g(\tau - 1) + 1$ model fits if one uses the same α for each sequence of $\tau \geq 2$ steps ($g^{\tau-1}$ if α is not fixed across distillation steps). However, by Algorithm 1 it is sufficient to perform $2(\tau - 1) + 1$ model fits, and obtain the exact optimal value at each step instead of an approximated value. In Section A.5 we apply Algorithm 1 to approximate $\alpha^{*(\tau)}$ in a deep learning setting.

Algorithm 1: Calculate $\hat{\beta}^{(\tau)}$ and $\alpha^{*(\tau)}$ for $\tau \geq 2$.

- 1 Calculate $\hat{\beta}^{(1)}$ from (A.4) (with any $\alpha^{(1)}$)
 - 2 Calculate $\tilde{\mathbf{y}}^{(1)} = f(\tilde{\mathbf{X}}, \hat{\beta}^{(1)})$
 - 3 **for** $t = 2$ **to** τ **do**
 - 4 Calculate $\hat{\beta}_{\alpha=0}^{(t)}$ from (A.4) and $\tilde{\mathbf{y}}_{\alpha=0}^{(t)} = f(\tilde{\mathbf{X}}, \hat{\beta}_{\alpha=0}^{(t)})$
 - 5 Solve $\alpha^{*(t)} = \operatorname{argmin}_{\alpha \in \mathbb{R}} \left\| \tilde{\mathbf{y}} - (\alpha \tilde{\mathbf{y}}^{(1)} + (1 - \alpha) \tilde{\mathbf{y}}_{\alpha=0}^{(t)}) \right\|_2^2$
 - 6 Calculate $\hat{\beta}^{(t)}$ from (A.4) with $\alpha^{*(t)}$
 - 7 **end**
-

A.4.4 Infinite Number of Self-Distillation Steps

We now prove that if we were to perform an infinite number of distillations steps ($\tau \rightarrow \infty$) with a fixed α (i.e. $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$) the solution would solve the classical kernel ridge regression problem, with an amplified regularization parameter (by α^{-1}) if $\alpha > 0$. Observe that, when $\alpha = 0$ and $\tau \rightarrow \infty$, (A.7) and (A.8) yield that the predictions $\mathbf{y}^{(\infty)}$ and $f(\mathbf{x}, \hat{\beta}^{(\infty)})$ collapse to the zero-solution for any $\mathbf{x} \in \mathbb{R}^p$ as expected from [Mobahi et al. \(2020\)](#).

Theorem A.5. Let $\mathbf{y}^{(\tau)}$, $\hat{\beta}^{(\tau)}$, and $f(\cdot, \hat{\beta}^{(\tau)})$ be defined as above, and $\alpha \in (0, 1]$, then the following limits hold

$$\begin{aligned} \mathbf{y}^{(\infty)} &\stackrel{\text{def}}{=} \lim_{\tau \rightarrow \infty} \mathbf{y}^{(\tau)} = \mathbf{K} \left(\mathbf{K} + \frac{\lambda}{\alpha} \mathbf{I}_n \right)^{-1} \mathbf{y} & (\text{A.21}) \\ f(\mathbf{x}, \hat{\beta}^{(\infty)}) &\stackrel{\text{def}}{=} \lim_{\tau \rightarrow \infty} f(\mathbf{x}, \hat{\beta}^{(\tau)}) = \alpha f(\mathbf{x}, \hat{\beta}^{(1)}) + (1 - \alpha) f(\mathbf{x}, \hat{\gamma}^{(\infty)}) \end{aligned}$$

where (A.21) corresponds to classical kernel ridge regression with amplified regularization parameter λ/α , and we let $\hat{\gamma}^{(\infty)}$ denote the kernel ridge regression weights associated with solving another kernel ridge regression on the targets $\mathbf{y}^{(\infty)}$ with regularization parameter λ . Furthermore, the convergence $\lim_{\tau \rightarrow \infty} \mathbf{y}^{(\tau)}$ is of linear rate.

Proof. See proof on page 51. □

If $\alpha > 0$, then by (A.10) and Theorem A.5, we have that $\mathbf{y}^{(\infty)} = \sum_{j=1}^p \mathbf{v}_j \frac{d_j}{d_j + \frac{\lambda}{\alpha}} \mathbf{v}_j^\top \mathbf{y}$ and we shrink the eigenvectors with smallest eigenvalues, corresponding to the directions with least variance, the most. Furthermore, if $\alpha > 0$ the limiting solution is a non-zero kernel ridge regression with regularization parameter $\lambda/\alpha \geq \lambda$, causing the eigenvectors associated with the smallest eigenvalues to shrink even more than in the original solution.

Our results give a theoretical explanation for why one should treat $\alpha^{(\tau)}$ as an adjustable hyperparameter to fine-tune the amount of regularization that self-distillation imposes for a particular problem, and that it can be chosen in an optimal way for kernel ridge regression. In the following we provide an illustrative example, and in Section A.5 we estimate the optimal weighting parameter for deep learning using an adaptation of Algorithm 1.

A.4.5 Illustrative Example

Consider the training dataset \mathcal{D} where $\mathcal{X} = \{0, 0.1, \dots, 0.9, 1\}$ and $\mathcal{Y} = \{\sin(2\pi x) + \varepsilon \mid x \in \mathcal{X}\}$, and ε is sampled from a zero-mean Gaussian random variable with standard deviation 0.5. Let φ be the Radial Basis Function kernel, i.e. $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|_2^2}$, where we choose $\gamma = \frac{1}{80}$, and let $\lambda = 0.2$ and consider the three cases; (a) $\alpha = 0$, (b) $\alpha = 0.25$, and (c) step-wise optimal $\alpha^{(\tau)}$.

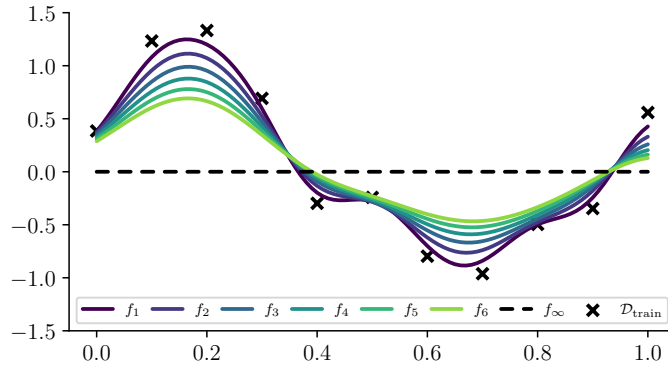
As illustrated in Figure A.2a for case (a), the regularization imposed by self-distillation initially improves the quality of the solution, but eventually overregularize and the solutions underfit the data, and will eventually converge to the zero-solution (see supplementary materials for the loss values). Using $\alpha > 0$ (see Figure A.2b), and more specifically $\alpha = 0.25$, reduce the imposed regularization and increase the stability of the distillation procedure; i.e. the solutions differ much less between each distillation step. This allows for a more dense exploration of solutions during iterated distillation steps, where increasing α reduces the difference between solutions from two consecutive steps, but also reduces the space of possible solutions as the limit, $f(\cdot, \hat{\beta}^{(\infty)})$, approaches the initial solution $f(\cdot, \hat{\beta}^{(1)})$ quickly.¹⁰ However, choosing the step-wise optimal $\alpha^{(\tau)}$ yields minuscule changes to the solution for $\tau > 2$, and a single step of distillation is effectively enough. Furthermore, for $\tau \geq 3$, all $\alpha^{(\tau)}$ are approximately equal, and the distillation procedure has reached an equilibrium.¹¹

As expected from Lemma A.2 and Theorem A.3, Figure A.3 verifies that both in case (a) and (b), the diagonal of $\mathbf{B}^{(\tau)}$ is decreasing in τ and the diagonal coordinates corresponding to smaller eigenvalues shrink faster than those corresponding to larger eigenvalues in case (a). Without loss of generality we can assume $d_1 < d_2 < \dots < d_n$, and for $k = 1, \dots, n-1$ and any $\tau \geq 1$ define $R_k^{(\tau)} \stackrel{\text{def}}{=} [\mathbf{B}^{(\tau)}]_{k+1, k+1} / [\mathbf{B}^{(\tau)}]_{k, k}$. We expect $R_k^{(\tau)}$ to be strictly increasing in τ for all k in case (a), but for case (b) we can make no such guarantee. Both of these properties are verified in Figure A.4.

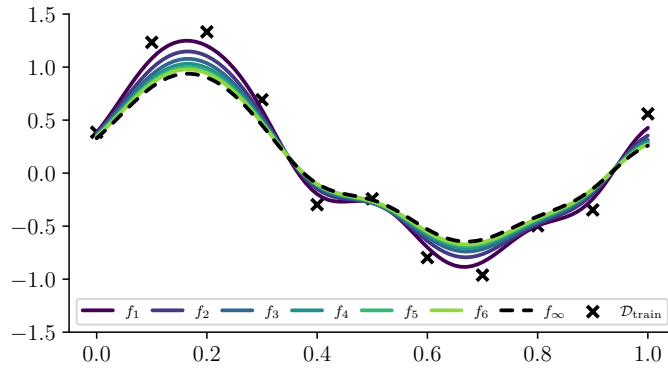
Finally, we observe that in case (a), the values of $\mathbf{B}^{(\tau)}$ shrink much faster than in case (b), and eventually collapse to all zeros, whereas the latter is nearly converged after six iterations. Furthermore, case (a) appears to obtain a more sparsified solution, as the smallest coordinates effectively diminish, which is not true for case (b). Furthermore, when directly comparing solutions from both cases with similar quality of fit, the solutions obtained with $\alpha = 0$ usually have smaller coordinates in $\mathbf{B}^{(\tau)}$ than those obtained with larger values of α .

¹⁰ As expected by Theorem A.5, we experience a fast convergence to the limit; usually less than 10 iterations are sufficient to converge

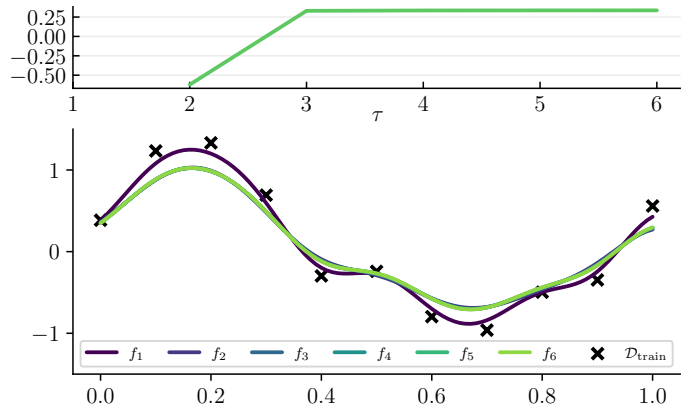
¹¹ If we clip $\alpha^{(\tau)}$ to be in $[0, 1]$, the $\alpha^{(\tau)}$ converges at $\tau = 4$ rather than $\tau = 3$.



(a) $\alpha = 0$

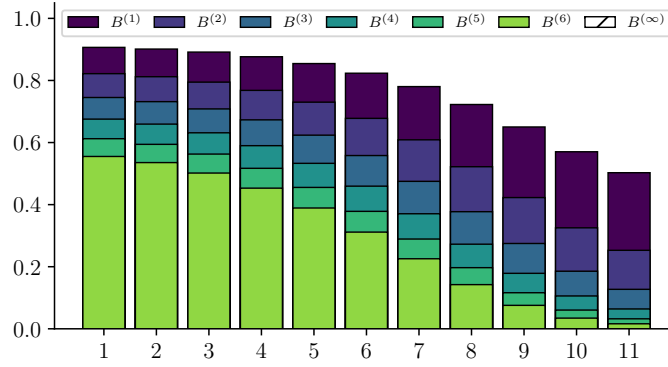


(b) $\alpha = 0.25$

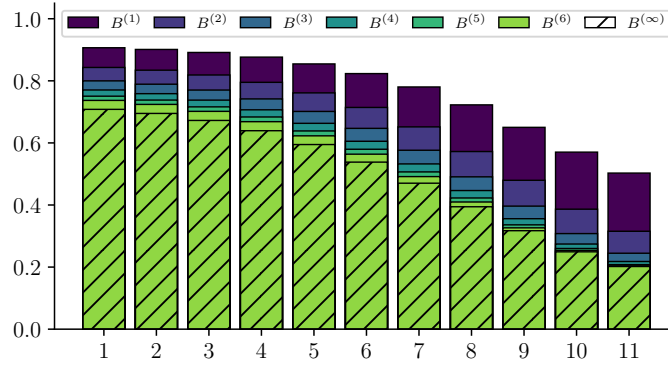


(c) $\alpha^{*(\tau)}$

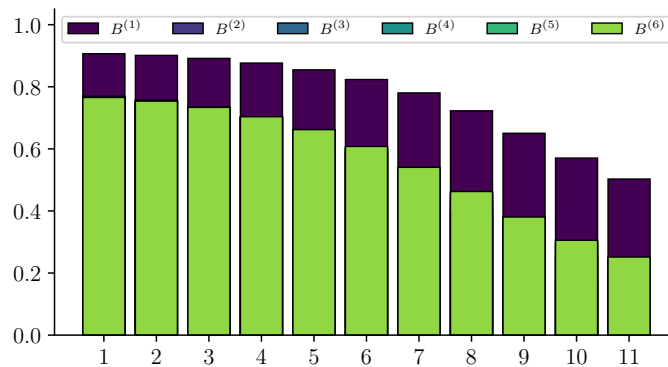
Figure A.2: Six steps of self-distillation with (a) zero limiting solution (dashed), (b) non-zero limiting solution (dashed), and (c) optimal step-wise $\alpha^{*(\tau)}$. Training examples are represented with \times and in (c) we also plot $\alpha^{*(\tau)}$ with τ .



(a) $\alpha = 0$

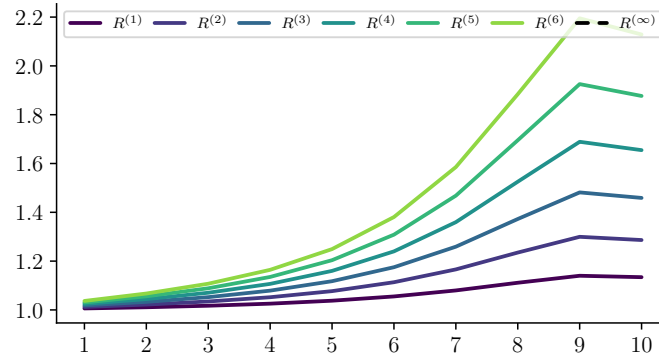


(b) $\alpha = 0.25$

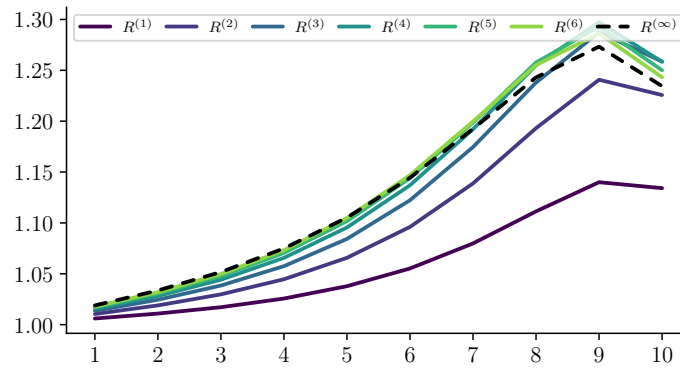


(c) $\alpha^{*(\tau)}$

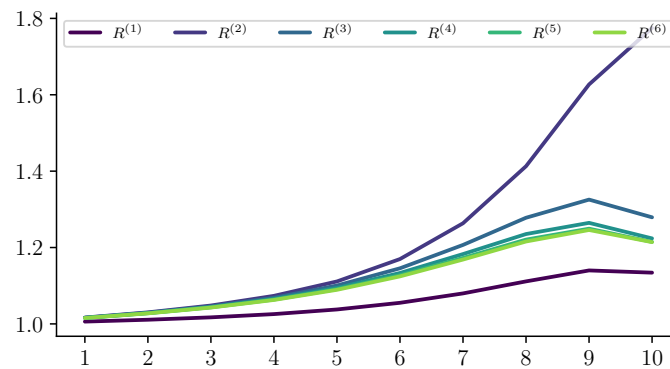
Figure A.3: Diagonal of $\mathbf{B}^{(\tau)}$ for $\tau = 1, \dots, 6$ associated with Figure A.2. Note, the plots are overlaid, but since the diagonal of $\mathbf{B}^{(\tau)}$ decreases in τ , all values until convergence are visible. In (a) we expect and observe strictly decreasing values in τ for all indices until collapsing at 0, but in (b) and (c) they converge to a non-zero limit.



(a) $\alpha = 0$



(b) $\alpha = 0.25$



(c) $\alpha^{(\tau)}$

Figure A.4: Ratios, $R_k^{(\tau)}$ of the ordered diagonal of $\mathbf{B}^{(\tau)}$ for all τ . In (a) we expect and observe strictly increasing values in τ for all k , but have no such guarantee in (b) or (c). The x-axis corresponds to indices $k = 1, \dots, n - 1$.

A.5 Approximate Optimal Weighting Parameter for Deep Learning

The following experiment aims to empirically evaluate the theoretical analysis above in a simple deep-learning setting. In (A.19) we find $\alpha^{*\tau}$ on closed form when $f(\cdot, \hat{\beta}^{(\tau)})$ is a (self-distilled) kernel ridge regression. No closed-form solution can be found for neural networks, but recent results show that (very) wide neural networks can be seen as kernel ridge regression solutions with the neural tangent kernel (Jacot et al., 2018; Arora et al., 2019; Lee et al., 2019, 2020).

Thus, inspired by (A.8) we propose to estimate α^{*t} for $t = 2, \dots, \tau$, denoted by $\hat{\alpha}^{(t)}$, for a neural network trained with self-distillation using an adapted Algorithm 1. Let $f_{\text{nn}}(\cdot, \theta) \in \mathbb{R}^p$ be a neural network with vector of weights θ , and recursively for $\tau \geq 1$ let $\hat{\theta}^{(\tau)}$ be the weights solving

$$\operatorname{argmin}_{\theta} \frac{\alpha^{(\tau)}}{2} \|f_{\text{nn}}(\mathbf{X}, \theta) - \mathbf{Y}^{(1)}\|_F^2 + \frac{1 - \alpha^{(\tau)}}{2} \|f_{\text{nn}}(\mathbf{X}, \theta) - \mathbf{Y}^{(\tau-1)}\|_F^2 + \frac{\lambda}{2} \|\theta\|_2^2, \quad (\text{A.22})$$

with $\alpha^{(\tau)} = \hat{\alpha}^{(\tau)}$ and where $\mathbf{Y}^{(\tau)} \in \mathbb{R}^{n \times p}$.¹² Furthermore, let $\hat{\theta}_{\alpha=0}^{(\tau)}$ be the weights associated with minimizing (A.22) with $\alpha^{(\tau)} = 0$, and $\tilde{\mathbf{Y}}_{\alpha=0}^{(\tau)} \stackrel{\text{def}}{=} f_{\text{nn}}(\tilde{\mathbf{X}}, \hat{\theta}_{\alpha=0}^{(\tau)})$ as well as $\tilde{\mathbf{Y}}^{(\tau)} \stackrel{\text{def}}{=} f_{\text{nn}}(\tilde{\mathbf{X}}, \hat{\theta}^{(\tau)})$ be the predictions on the validation input $\tilde{\mathbf{X}}$. Then, following Algorithm 1 with $\|\cdot\|_2$ replaced by $\|\cdot\|_F$, and (A.22) rather than (A.1) we can calculate the estimates $\hat{\alpha}^{(t)}$. These estimates yield comparable predictive performance to the best fixed $\alpha^{(\tau)}$ (found with time-consuming grid search) but only require one additional model fit per distillation step; i.e. $2(\tau - 1) + 1$ fits compared to $g(\tau - 1) + 1$ for a grid search over g values. See Figure A.5 for results and supplementary material for experimental details.

A.5.1 Experiment

We perform self-distillation with ResNet-50 (He et al., 2016) networks on CIFAR-10 (Krizhevsky et al., 2009), with minor pre-processing and augmentations. The model is initialized randomly at each step¹³ and trained according to the above with either estimated optimal parameters, $\hat{\alpha}^{(\tau)}$, or fixed α for all steps. We use the network weights from the last iteration of training at each distillation step for the next step, irrespective of whether a better model occurred earlier in the training. Our models are trained for fixed 75 epochs and each experiment is repeated with 4 different random seeds over 11 chains of distillation steps, corresponding to $\alpha \in \{0.0, 0.1, \dots, 0.9\}$ and $\hat{\alpha}^{(\tau)}$, with the first model initialized identically across all chains. The accuracy reported at the τ 'th step is based on comparing the training and validation predictions, $\mathbf{Y}^{(\tau)}$ and $f(\tilde{\mathbf{X}}, \hat{\beta}^{(\tau)})$ with the original training and validation targets; \mathbf{Y} and $\tilde{\mathbf{Y}}$.¹⁴

¹² We treat class labels as p -dimensional one-hot encoded vectors and use norm of the difference between the predicted class probabilities and the one-hot vectors.

¹³ Note, we initialize the models equally across all α for one experiment, but alter the seed for initialization between experiments.

¹⁴ The empirical experiments are constrained by the theoretical set-up and performed in a highly simple setting; e.g. using the weighted MSE loss from (A.22) (see supplementary materials for more details). Therefore, our accuracy measures are to be expected to be lower than for more fine-tuned training setups.

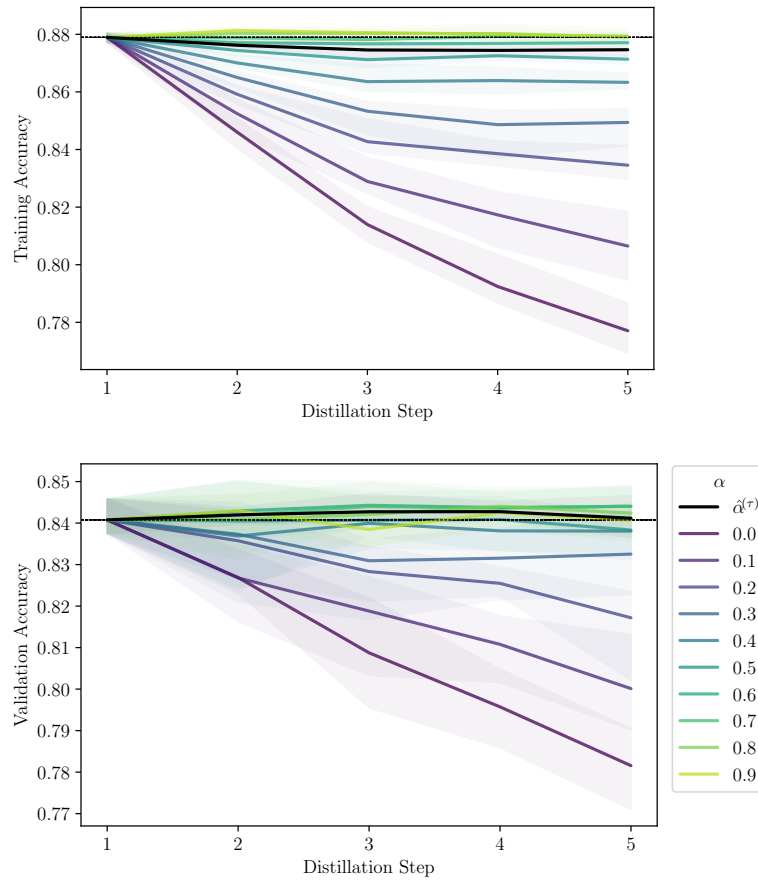


Figure A.5: Training and validation accuracy for five distillation steps with ResNet-50 models on CIFAR-10. Comparing fixed $\alpha^{(t)}$ for $t = 2, \dots, \tau$ and estimating optimal weight with $\hat{\alpha}^{(t)}$ at each step. The experiment is repeated four times and the mean (and max/min in shaded) is reported.

A.6 Conclusion

In this paper, we provided theoretical arguments for the importance of weighting the teacher outputs with the ground-truth targets when performing self-distillation with kernel ridge regressions along with a closed-form solution for the optimal weighting parameter. We proved how the solution at any (possibly infinite) distillation step can be calculated directly from the initial distillation step, and that self-distillation for an infinite number of steps corresponds to a classical kernel ridge regression solution with amplified regularization parameter. We showed both empirically and theoretically that the weighting parameter α determines the amount of regularization imposed by self-distillation, and empirically supported our results in a simple deep-learning setting.

Future Research Directions

Interesting directions of future research are on rigorously connecting neural networks and kernel methods in a knowledge distillation setting, extending to other objective functions than MSE as well as including intermediate model statistics in the distillation

procedure. Finally, a larger empirical study of the connection between the choice of α and the degree of overfitting is interesting as well.

Acknowledgments

We would like to thank GenomeDK and Aarhus University for providing computational resources that contributed to these research results. Furthermore, we would like to thank Daniel Borup and Ragnhild Ø. Laursen for comments and discussion, as well as Google Researcher Hossein Mobahi and Mehrdad Farajtabar (Deepmind) for clarifications on their experimental setup. We also thank the anonymous reviewers of the NeurIPS 2021 conference for their comments. Kenneth Borup is partly financed by Aarhus University Centre for Digitalisation, Big Data, and Data Analytics (DIGIT).

Bibliography

- Abnar, S., Dehghani, M., and Zuidema, W. (2020). Transferring Inductive Biases through Knowledge Distillation. *arXiv preprint arXiv:2006.00555*. Cited on page 32.
- Ahn, S., Hu, S. X., Damianou, A., Lawrence, N. D., and Dai, Z. (2019). Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9155–9163. Cited on pages 30, 31, and 52.
- Anil, R., Pereyra, G., Passos, A., Ormandi, R., Dahl, G. E., and Hinton, G. E. (2018). Large scale distributed neural network training through online distillation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. Cited on page 31.
- Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, volume 32. Cited on pages 42 and 55.
- Ba, J. L. and Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems*, volume 27, pages 2654–2662. Cited on page 31.
- Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pages 535–541. Cited on page 31.
- Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., and Tian, Q. (2019). Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3514–3522. Cited on page 31.
- Dong, B., Hou, J., Lu, Y., and Zhang, Z. (2019). Distillation \approx Early Stopping? Harvesting Dark Knowledge Utilizing Anisotropic Information Retrieval For Overparameterized Neural Network. *arXiv preprint arXiv:1910.01255*. Cited on pages 31 and 53.
- Falcon, W. (2019). PyTorch Lightning. github.com/PyTorchLightning/pytorch-lightning. Cited on page 53.
- Fang, G., Song, J., Shen, C., Wang, X., Chen, D., and Song, M. (2019). Data-Free Adversarial Distillation. *arXiv preprint arXiv:1912.11006*. Cited on page 31.

- Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born Again Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1607–1616. PMLR. Cited on pages 30, 31, and 52.
- Gao, M., Shen, Y., Li, Q., and Loy, C. C. (2020). Residual Knowledge Distillation. *arXiv preprint arXiv:2002.09168*. Cited on page 31.
- Gotmare, A., Shirish Keskar, N., Xiong, C., and Socher, R. (2019). A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on page 32.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 770–778. Cited on pages 42 and 52.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*. Cited on pages 30 and 31.
- Hu, W., Li, Z., and Yu, D. (2020). Simple and Effective Regularization Methods for Training on Noisily Labeled Data With Generalization Guarantee. *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. Cited on pages 30 and 54.
- Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pages 8571–8580. Cited on pages 42 and 55.
- Krizhevsky, A., Nair, V., and Hinton, G. (2009). Learning multiple layers of features from tiny images. Cited on pages 42 and 52.
- Lee, J., Xiao, L., Schoenholz, S. S., Novak, Y. B. R., Sohl-Dickstein, J., Pennington, J., Bahri, Y., Dec, M. L., and Brain, G. (2019). Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In *Advances in Neural Information Processing Systems*. Cited on pages 30, 42, 54, and 55.
- Lee, J. D., Shen, R., Song, Z., Wang, M., and Yu, Z. (2020). Generalized leverage score sampling for neural networks. *Advances in Neural Information Processing Systems*, 33:10775–10787. Cited on pages 30, 42, 54, and 55.
- Lopes, R. G., Fenu, S., and Starner, T. (2017). Data-Free Knowledge Distillation for Deep Neural Networks. *arXiv preprint arXiv:1710.07535*. Cited on page 31.
- Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on page 30.
- Micaelli, P. and Storkey, A. (2019). Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems*, volume 32, pages 9551–9561. Cited on page 31.
- Mobahi, H., Farajtabar, M., and Bartlett, P. L. (2020). Self-Distillation Amplifies Regularization in Hilbert Space. In *Advances in Neural Information Processing Systems*. Cited on pages 32, 33, 34, 37, 52, 55, 56, 57, and 58.

- Müller, R., Kornblith, S., and Hinton, G. (2019). When does label smoothing help? In *Advances in Neural Information Processing Systems*, volume 32. Cited on page 31.
- Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2020). Deep Double Descent: Where Bigger Models and More Data Hurt. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. Cited on page 30.
- Nayak, G. K., Mopuri, K. R., Shaj, V., Babu, R. V., and Chakraborty, A. (2019). Zero-shot knowledge distillation in deep networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 8317–8325. PMLR. Cited on page 31.
- Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational Knowledge Distillation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Cited on page 31.
- Phuong, M. and Lampert, C. H. (2019). Towards understanding knowledge distillation. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 8993–9007. Cited on page 32.
- Radosavovic, I., Dollar, P., Girshick, R., Gkioxari, G., and He, K. (2018). Data Distillation: Towards Omni-Supervised Learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4119–4128. Cited on page 31.
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). FitNets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. Cited on page 31.
- Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426, Berlin, Heidelberg. Springer. Cited on page 33.
- Srinivas, S. and Fleuret, F. (2018). Knowledge transfer with jacobian matching. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 7515–7523. Cited on page 31.
- Tang, J., Shivanna, R., Zhao, Z., Lin, D., Singh, A., Chi, E. H., and Jain, S. (2020). Understanding and Improving Knowledge Distillation. *arXiv preprint arXiv:2002.03532*. Cited on page 31.
- Tian, Y., Krishnan, D., and Isola, P. (2020a). Contrastive Representation Distillation. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. Cited on page 31.
- Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. (2020b). Rethinking Few-Shot Image Classification: A Good Embedding is All You Need? *arXiv preprint arXiv:2003.11539*. Cited on page 52.
- Wang, L. and Yoon, K.-J. (2021). Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):3048–3068. Cited on page 30.

- Yang, C., Xie, L., Qiao, S., and Yuille, A. (2018). Knowledge Distillation in Generations: More Tolerant Teachers Educate Better Students. *arXiv preprint arXiv:1805.05551*. Cited on pages 30 and 52.
- Yao, Y., Rosasco, L., and Caponnetto, A. (2007). On Early Stopping in Gradient Descent Learning. *Constructive Approximation*, 26:289–315. Cited on page 30.
- Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7130–7138. Cited on page 31.
- Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Cited on page 31.
- Zhang, C., Recht, B., Bengio, S., Hardt, M., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Cited on page 30.

Supplementary material

A.I Proofs

This section includes all proofs referenced in the main part of the paper, along with the associated theorems and lemmas for completeness.

Theorem A.1 (from page 34). *Let $\mathbf{y}^{(\tau)}$, $\hat{\boldsymbol{\beta}}^{(\tau)}$, and $f(\cdot, \hat{\boldsymbol{\beta}}^{(\tau)})$ be defined as above. Fix $\alpha^{(2)}, \dots, \alpha^{(\tau)} \in [0, 1)$, and let $\eta(i, \tau) \stackrel{\text{def}}{=} \prod_{j=i}^{\tau} (1 - \alpha^{(j)})$, then for $\tau \geq 1$, we have that*

$$\mathbf{y}^{(\tau)} = \left(\sum_{i=2}^{\tau} \alpha^{(i)} \eta(i+1, \tau) (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^{\tau-i+1} + \eta(2, \tau) (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^{\tau} \right) \mathbf{y},$$

$$f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(\tau)}) = \alpha^{(\tau)} f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(1)}) + (1 - \alpha^{(\tau)}) f(\mathbf{x}, \hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)})$$

for any $\mathbf{x} \in \mathbb{R}^d$, where $\hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)}$ is the minimizer in (A.4) with $\alpha^{(\tau)} = 0$.

Proof. We prove the theorem by induction, where we let $\tilde{\mathbf{K}} \stackrel{\text{def}}{=} \mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1}$. For $\tau = 1$, the result holds trivially, and thus, assume it holds for $\tau = t$. Since $\boldsymbol{\beta}^{(t+1)} = \varphi(\mathbf{X})^{\top} (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha^{(t+1)} \mathbf{y} + (1 - \alpha^{(t+1)}) \mathbf{y}^{(t)})$ we have that

$$\begin{aligned} \mathbf{y}^{(t+1)} &= \varphi(\mathbf{X}) \varphi(\mathbf{X})^{\top} (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha^{(t+1)} \mathbf{y} + (1 - \alpha^{(t+1)}) \mathbf{y}^{(t)}) \\ &= \alpha^{(t+1)} \tilde{\mathbf{K}} \mathbf{y} + (1 - \alpha^{(t+1)}) \tilde{\mathbf{K}} \left(\sum_{i=2}^t \alpha^{(i)} \eta(i+1, t) \tilde{\mathbf{K}}^{t-i+1} + \eta(2, t) \tilde{\mathbf{K}}^t \right) \mathbf{y} \\ &= \alpha^{(t+1)} \tilde{\mathbf{K}} \mathbf{y} + \left(\sum_{i=2}^t \alpha^{(i)} \eta(i+1, t+1) \tilde{\mathbf{K}}^{(t+1)-i+1} + \eta(2, t+1) \tilde{\mathbf{K}}^{t+1} \right) \mathbf{y} \\ &= \left(\sum_{i=2}^{t+1} \alpha^{(i)} \eta(i+1, t+1) \tilde{\mathbf{K}}^{(t+1)-i+1} + \eta(2, t+1) \tilde{\mathbf{K}}^{t+1} \right) \mathbf{y} \end{aligned}$$

which finalizes our induction proof for the first part. For the second part, note that it also holds trivially for $\tau = 1$. Thus assume, it holds for $\tau = t$, then by direct manipulations

$$\begin{aligned} f(\mathbf{x}, \boldsymbol{\beta}^{(t+1)}) &= \kappa(\mathbf{x}, \mathbf{X})^{\top} (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha^{(t+1)} \mathbf{y} + (1 - \alpha^{(t+1)}) \mathbf{y}^{(t)}) \\ &= \alpha^{(t+1)} f(\mathbf{x}, \boldsymbol{\beta}^{(1)}) + (1 - \alpha^{(t+1)}) \kappa(\mathbf{x}, \mathbf{X})^{\top} (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{y}^{(t)} \\ &= \alpha^{(t+1)} f(\mathbf{x}, \boldsymbol{\beta}^{(1)}) + (1 - \alpha^{(t+1)}) f(\mathbf{x}, \hat{\boldsymbol{\beta}}_{\alpha=0}^{(t+1)}), \end{aligned}$$

where we let $\hat{\boldsymbol{\beta}}_{\alpha=0}^{(t+1)}$ denote the minimizer (A.4) with $\alpha^{(t+1)} = 0$; i.e. minimizing the classical kernel ridge regression problem with targets $\mathbf{y}^{(t)}$. \square

Lemma A.2 (from page 35). *Let $\mathbf{B}^{(\tau)}$, and \mathbf{A} be defined as above, and let $\mathbf{B}^{(0)} \stackrel{\text{def}}{=} \mathbf{I}$. Then we can express $\mathbf{B}^{(\tau)}$ recursively as*

$$\mathbf{B}^{(\tau)} = \mathbf{A} \left((1 - \alpha^{(\tau)}) \mathbf{B}^{(\tau-1)} + \alpha^{(\tau)} \mathbf{I}_n \right),$$

and $[\mathbf{B}^{(\tau)}]_{k,k} \in [0, 1]$ is (strictly) decreasing in τ for all $k \in [n]$ and $\tau \geq 1$ if $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$.

Proof. The case, $\tau = 1$, is easy to verify, and we assume the claim holds for $\tau = t$. Then note that

$$\begin{aligned} & \mathbf{A} \left((1 - \alpha^{(t+1)}) \mathbf{B}^{(t)} + \alpha^{(t+1)} \mathbf{I}_n \right) \\ &= \sum_{i=2}^t \alpha^{(i)} \eta(i+1, t+1) \mathbf{A}^{(t+1)-i+1} + \eta(2, t+1) \mathbf{A}^{t+1} + \alpha^{(t+1)} \mathbf{A} \\ &= \sum_{i=2}^{t+1} \alpha^{(i)} \eta(i+1, t+1) \mathbf{A}^{(t+1)-i+1} + \eta(2, t+1) \mathbf{A}^{t+1} \\ &= \mathbf{B}^{(t+1)}, \end{aligned}$$

finalizing the induction proof. Now, assume $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$ and note that for any k and $\tau \geq 1$, then

$$\begin{aligned} [\mathbf{A}]_k \left((1 - \alpha) [\mathbf{B}^{(\tau-1)}]_{k,k} + \alpha \right) &= [\mathbf{B}^{(\tau)}]_{k,k} \\ &\leq [\mathbf{B}^{(\tau-1)}]_{k,k} \\ &= [\mathbf{A}]_k \left((1 - \alpha) [\mathbf{B}^{(\tau-2)}]_{k,k} + \alpha \right), \end{aligned}$$

if and only if $[\mathbf{B}^{(\tau-1)}]_{k,k} \leq [\mathbf{B}^{(\tau-2)}]_{k,k}$, and iteratively, if and only if $[\mathbf{B}^{(1)}]_{k,k} \leq [\mathbf{B}^{(0)}]_{k,k}$. The latter is indeed true, since $\mathbf{B}^{(1)} = \mathbf{A}$, and finally, $\mathbf{A} = \mathbf{I}_n$ if and only if $\lambda = 0$. \square

Theorem A.3 (from page 35). Assume $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$. Then, for any pair of diagonals of \mathbf{D} , i.e. d_k and d_j , where $d_k > d_j$, we have that for all $\tau \geq 1$,

$$\frac{[\mathbf{B}^{(\tau)}]_{k,k}}{[\mathbf{B}^{(\tau)}]_{j,j}} = \begin{cases} \frac{1 + \frac{\lambda}{d_j}}{1 + \frac{\lambda}{d_k}}, & \text{for } \alpha = 1, \\ \left(\frac{1 + \frac{\lambda}{d_j}}{1 + \frac{\lambda}{d_k}} \right)^\tau, & \text{for } \alpha = 0, \end{cases}$$

and if we let $\text{sgn}(\cdot)$ denote the sign function, i.e.

$$\text{sgn}(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0 \end{cases}$$

then for $\alpha \in (0, 1)$ we have that

$$\begin{aligned} & \text{sgn} \left(\frac{[\mathbf{B}^{(\tau)}]_{k,k}}{[\mathbf{B}^{(\tau)}]_{j,j}} - \frac{[\mathbf{B}^{(\tau-1)}]_{k,k}}{[\mathbf{B}^{(\tau-1)}]_{j,j}} \right) \\ &= \text{sgn} \left(\left(\left(\frac{[\mathbf{B}^{(\tau-1)}]_{k,k}}{[\mathbf{B}^{(\tau-1)}]_{j,j}} - \frac{[\mathbf{A}]_{k,k}}{[\mathbf{A}]_{j,j}} \right) \frac{[\mathbf{A}]_{j,j}}{[\mathbf{B}^{(\tau-1)}]_{k,k}([\mathbf{A}]_{k,k} - [\mathbf{A}]_{j,j})} + 1 \right)^{-1} - \alpha \right). \end{aligned}$$

Proof. First note that

$$\frac{[\mathbf{A}]_{k,k}}{[\mathbf{A}]_{j,j}} = \frac{\frac{d_k}{d_k + \lambda}}{\frac{d_j}{d_j + \lambda}} = \frac{1 + \frac{\lambda}{d_j}}{1 + \frac{\lambda}{d_k}},$$

and for $\alpha = 1$, (A.14) amounts to $\mathbf{B}^{(\tau)} = \mathbf{A}$, which gives the first result. For $\alpha = 0$, (A.14) amounts to $\mathbf{B}^{(\tau)} = \mathbf{A}^\tau$, and the second result follows. For the remainder we denote

$[\mathbf{B}^{(\tau-1)}]_{k,k}$ by \mathbf{B}_k and $[\mathbf{A}]_{k,k}$ by \mathbf{A}_k to simplify notation. We investigate the case where both r.h.s. and l.h.s. equals zero. Thus, for $\alpha \in (0, 1)$, we observe that if

$$\frac{\mathbf{B}_k}{\mathbf{B}_j} = \frac{\mathbf{A}_k (1 - \alpha)\mathbf{B}_k + \alpha}{\mathbf{A}_j (1 - \alpha)\mathbf{B}_j + \alpha} = \frac{\mathbf{A}_k \frac{1-\alpha}{\alpha}\mathbf{B}_k + 1}{\mathbf{A}_j \frac{1-\alpha}{\alpha}\mathbf{B}_j + 1},$$

then we have that

$$\begin{aligned} \mathbf{B}_j &= \frac{1}{\frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) - \frac{1-\alpha}{\alpha}} = \frac{\mathbf{B}_k}{\frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) - \frac{1-\alpha}{\alpha}\mathbf{B}_k} \\ \frac{1-\alpha}{\alpha}\mathbf{B}_j + 1 &= \frac{\frac{1-\alpha}{\alpha}\mathbf{B}_k}{\frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) - \frac{1-\alpha}{\alpha}\mathbf{B}_k} + 1 = \frac{\frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right)}{\frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) - \frac{1-\alpha}{\alpha}\mathbf{B}_k}, \end{aligned}$$

which in turn yields that

$$\begin{aligned} \frac{\mathbf{B}_k}{\mathbf{B}_j} &= \frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) \left(\frac{\frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) - \frac{1-\alpha}{\alpha}\mathbf{B}_k}{\frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right)} \right) \\ &= \frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) - \frac{1-\alpha}{\alpha}\mathbf{B}_k. \end{aligned}$$

Now, observe that $0 = \alpha - \left(\left(\frac{\mathbf{B}_k}{\mathbf{B}_j} - \frac{\mathbf{A}_k}{\mathbf{A}_j} \right) \frac{\mathbf{A}_j}{\mathbf{B}_k(\mathbf{A}_k - \mathbf{A}_j)} + 1 \right)^{-1}$ yield that

$$\frac{\mathbf{B}_k}{\mathbf{B}_j} = \frac{1-\alpha}{\alpha} \frac{\mathbf{B}_k(\mathbf{A}_k - \mathbf{A}_j)}{\mathbf{A}_j} + \frac{\mathbf{A}_k}{\mathbf{A}_j} = \frac{\mathbf{A}_k}{\mathbf{A}_j} \left(\frac{1-\alpha}{\alpha}\mathbf{B}_k + 1 \right) - \frac{1-\alpha}{\alpha}\mathbf{B}_k.$$

Thus, similar calculations with $>$ and $<$ instead of $=$, completes the claim. \square

Theorem A.4 (from page 36). *Fix $\tau \geq 2$, $\lambda > 0$ and $\alpha^{(2)}, \dots, \alpha^{(\tau-1)} \in \mathbb{R}$, then*

$$\begin{aligned} \alpha^{(\tau)} &= \operatorname{argmin}_{\alpha^{(\tau)} \in \mathbb{R}} \|\tilde{\mathbf{y}} - f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)})\|_2^2 \\ &= \frac{\left(\frac{\partial}{\partial \alpha^{(\tau)}} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) \right)^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^{(1)})}{\left\| \frac{\partial}{\partial \alpha^{(\tau)}} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) \right\|_2^2} + 1, \\ &= 1 - \frac{(\tilde{\mathbf{y}}_{\alpha=0}^{(\tau)} - \tilde{\mathbf{y}}^{(1)})^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^{(1)})}{\left\| \tilde{\mathbf{y}}_{\alpha=0}^{(\tau)} - \tilde{\mathbf{y}}^{(1)} \right\|_2^2} \end{aligned}$$

where $\tilde{\mathbf{y}}^{(1)} = f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(1)})$, and $\tilde{\mathbf{y}}_{\alpha=0}^{(\tau)} = f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}_{\alpha=0}^{(\tau)})$.

Proof. Let $\mathcal{L}(\alpha^{(\tau)}, \lambda) = \|\tilde{\mathbf{y}} - f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)})\|_2^2$, where f depends on $\alpha^{(\tau)}$ and λ through $\hat{\boldsymbol{\beta}}^{(\tau)}$. Note that,

$$\begin{aligned} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) &= \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda\mathbf{I})^{-1} (\alpha^{(\tau)}\mathbf{y} + (1 - \alpha^{(\tau)})\mathbf{y}^{(\tau-1)}) \\ \frac{\partial}{\partial \alpha^{(\tau)}} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) &= \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda\mathbf{I})^{-1} (\mathbf{y} - \mathbf{y}^{(\tau-1)}). \end{aligned}$$

Then for fixed $\lambda > 0$, we have that

$$\frac{\partial}{\partial \alpha^{(\tau)}} \mathcal{L}(\alpha, \lambda) = \left(\frac{\partial}{\partial \alpha^{(\tau)}} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) \right)^\top (2f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) - 2\tilde{\mathbf{y}})$$

and since we can decompose $f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)})$ as

$$\begin{aligned} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) &= \alpha^{(\tau)} \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} (\mathbf{y} - \mathbf{y}^{(\tau-1)}) + \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}^{(\tau-1)} \\ &= \alpha^{(\tau)} \frac{\partial}{\partial \alpha^{(\tau)}} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) + \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}^{(\tau-1)}, \end{aligned}$$

and set $\frac{\partial}{\partial \alpha^{(\tau)}} \mathcal{L}(\alpha^{(\tau)}, \lambda) = 0$, we can solve as follows

$$\begin{aligned} (\partial f^{(\tau)})^\top \tilde{\mathbf{y}} - (\partial f^{(\tau)})^\top \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}^{(\tau-1)} &= \alpha^{(\tau)} (\partial f^{(\tau)})^\top (\partial f^{(\tau)}) \\ &= \alpha^{(\tau)} \|\partial f^{(\tau)}\|^2, \end{aligned}$$

where we use the notation $\partial f^{(\tau)} \stackrel{\text{def}}{=} \frac{\partial}{\partial \alpha^{(\tau)}} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)})$ for brevity. Now since

$$-\kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}^{(\tau-1)} = \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} (\mathbf{y} - \mathbf{y}^{(\tau-1)}) - \kappa(\tilde{\mathbf{X}}, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y},$$

we can finalize the proof with

$$\alpha^{\star(\tau)} = \frac{\left(\frac{\partial}{\partial \alpha} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) \right)^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^{(1)})}{\left\| \frac{\partial}{\partial \alpha} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) \right\|^2} + 1,$$

and noting that $\frac{\partial}{\partial \alpha^{(\tau)}} f(\tilde{\mathbf{X}}, \hat{\boldsymbol{\beta}}^{(\tau)}) = \tilde{\mathbf{y}}^{(1)} - \tilde{\mathbf{y}}_{\alpha=0}^{(\tau)}$. \square

Note, in the following we state and prove a slightly more general result than Theorem A.5.

Theorem A.5 (from page 37). *Let $\mathbf{y}^{(\tau)}$, $\hat{\boldsymbol{\beta}}^{(\tau)}$, and $f(\cdot, \hat{\boldsymbol{\beta}}^{(\tau)})$ be defined as above, and $\alpha \in [0, 1]$, then the following limits hold*

$$\begin{aligned} \mathbf{y}^{(\infty)} &\stackrel{\text{def}}{=} \lim_{\tau \rightarrow \infty} \mathbf{y}^{(\tau)} = \alpha \mathbf{K} (\alpha \mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{y} \\ f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(\infty)}) &\stackrel{\text{def}}{=} \lim_{\tau \rightarrow \infty} f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(\tau)}) = \alpha \kappa(\mathbf{x}, \mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\mathbf{I}_n + (1 - \alpha) \mathbf{K} (\alpha \mathbf{K} + \lambda \mathbf{I}_n)^{-1}) \mathbf{y} \end{aligned}$$

and if $\alpha > 0$, then

$$\begin{aligned} \mathbf{y}^{(\infty)} &= \mathbf{K} \left(\mathbf{K} + \frac{\lambda}{\alpha} \mathbf{I}_n \right)^{-1} \mathbf{y} \\ f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(\infty)}) &= \alpha f(\mathbf{x}, \hat{\boldsymbol{\beta}}^{(1)}) + (1 - \alpha) f(\mathbf{x}, \hat{\boldsymbol{\gamma}}^{(\infty)}) \end{aligned}$$

where (A.21) corresponds to classical kernel ridge regression with amplified regularization parameter $\frac{\lambda}{\alpha}$, and we let $\hat{\boldsymbol{\gamma}}^{(\infty)}$ denote the kernel ridge regression parameter associated with solving another kernel ridge regression on the targets $\mathbf{y}^{(\infty)}$ with regularization parameter λ . Furthermore, the convergence $\lim_{\tau \rightarrow \infty} \mathbf{y}^{(\tau)}$ is of linear rate.

Proof. By (A.10) we have that $\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1} = \mathbf{V} \mathbf{D} (\mathbf{D} + \lambda \mathbf{I}_n)^{-1} \mathbf{V}^\top$ where $\lambda > 0$, \mathbf{D} is positive diagonal and \mathbf{V} is orthogonal, and hence, the eigenvalues of $\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1}$ are all smaller than 1 in absolute value, and thus $(1 - \alpha)^{\tau-1} (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^\tau$ converge to the zero-matrix when $\tau \rightarrow \infty$. Thus, using the limit for a geometric series of matrices we get that

$$\begin{aligned} \lim_{\tau \rightarrow \infty} \mathbf{y}^{(\tau)} &= \left(\frac{\alpha}{1 - \alpha} \sum_{i=1}^{\infty} \left((1 - \alpha) \mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \right)^i \right) \mathbf{y} \\ &= \frac{\alpha}{1 - \alpha} (1 - \alpha) \mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\mathbf{I}_n - (1 - \alpha) \mathbf{K}(\mathbf{K} + \lambda \mathbf{I}_n)^{-1})^{-1} \mathbf{y} \\ &= \alpha \mathbf{K} (\alpha \mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{y}. \end{aligned}$$

If $\alpha > 0$, the remaining result for $\lim_{\tau \rightarrow \infty} \mathbf{y}^{(\tau)}$ follows directly. Now, by inserting $\mathbf{y}^{(\infty)}$ and manipulating the result, we get that

$$\begin{aligned} f(\mathbf{x}, \boldsymbol{\beta}^{(\infty)}) &= \kappa(\mathbf{x}, \mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha \mathbf{y} + (1 - \alpha) \mathbf{y}^{(\infty)}) \\ &= \kappa(\mathbf{x}, \mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\alpha \mathbf{I}_n + (1 - \alpha) \alpha \mathbf{K} (\alpha \mathbf{K} + \lambda \mathbf{I}_n)^{-1}) \mathbf{y} \\ &= \alpha \kappa(\mathbf{x}, \mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} (\mathbf{I}_n + (1 - \alpha) \mathbf{K} (\alpha \mathbf{K} + \lambda \mathbf{I}_n)^{-1}) \mathbf{y}, \end{aligned}$$

and if $\alpha > 0$, then

$$\begin{aligned} f(\mathbf{x}, \boldsymbol{\beta}^{(\infty)}) &= \alpha f(\mathbf{x}, \boldsymbol{\beta}^{(1)}) + (1 - \alpha) \kappa(\mathbf{x}, \mathbf{X})^\top (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{K} \left(\mathbf{K} + \frac{\lambda}{\alpha} \mathbf{I}_n \right)^{-1} \mathbf{y} \\ &= \alpha f(\mathbf{x}, \boldsymbol{\beta}^{(1)}) + (1 - \alpha) f(\mathbf{x}, \hat{\boldsymbol{\gamma}}^{(\infty)}), \end{aligned}$$

where we let $\hat{\boldsymbol{\gamma}}^{(\infty)}$ denote the kernel ridge regression parameter associated with the classical kernel ridge regression problem on the targets $\mathbf{y}^{(\infty)}$ with regularization parameter λ . Finally, denote by $\mathbf{C} \stackrel{\text{def}}{=} (1 - \alpha) \mathbf{K} (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}$, then we have that

$$\mathbf{E}(t) \stackrel{\text{def}}{=} \sum_{i=1}^t \mathbf{C}^i - \sum_{i=1}^{\infty} \mathbf{C}^i = \mathbf{C}^{t+1} (\mathbf{C} - \mathbf{I}_n)^{-1},$$

and thus for an additional s steps we have $\mathbf{E}(t+s) = \mathbf{C}^{t+s+1} (\mathbf{C} - \mathbf{I}_n)^{-1} = \mathbf{C}^s \mathbf{E}(t)$. Hence, the convergence is of linear rate as claimed. \square

A.II Experiments

In the following, we show empirical results of performing a simple self-distillation procedure with deep neural networks with varying choices of α to investigate the large-scale effects. The experiments are adapted from [Mobahi et al. \(2020\)](#) with the additional introduction of the α -parameter. For stronger baselines of the possible performance gains from self-distillation see e.g. [Furlanello et al. \(2018\)](#); [Tian et al. \(2020b\)](#); [Ahn et al. \(2019\)](#); [Yang et al. \(2018\)](#). The following sections provide additional details to that of Section A.5.

A.II.1 Experimental Setup

We perform self-distillation with ResNet-50 ([He et al., 2016](#)) networks on CIFAR-10 ([Krizhevsky et al., 2009](#)), with minor pre-processing and augmentations.¹⁵ The model is initialized randomly at each step¹⁶ and trained as described in Section A.5 with either estimated optimal parameters, $\hat{\alpha}^{(\tau)}$, or fixed α for all steps. We use Adam optimizer with a learning rate of 10^{-4} , ℓ_2 regularization with regularization coefficient 10^{-4} , and train on the full 50000 training images and validate our generalization performance on the 10000 test images. We use the weights from the last step of optimization at each

¹⁵ Training: We randomly flip an image horizontally with probability $\frac{1}{2}$, followed by a random 32×32 crop of the 40×40 zero padded image. Finally, we normalize the image to have mean 0 and standard deviation 1. Validation: We normalize the image with the empirical mean and standard deviation from the training data.

¹⁶ Note, we initialize the models equally across all α for one experiment, but alter the seed for initialization between experiments.

distillation step for the next step, irrespective of whether a better model occurred earlier in the training. Our models are trained for fixed 75 epochs, which does not allow our models to overfit the training data, which is important for our models to be suitable for distillation procedures (Dong et al., 2019). The experiments are performed on a single Nvidia Tesla V100 16GB GPU with the PyTorch Lightning framework (Falcon, 2019).

A.II.2 Results

We repeat our experiment 4 times and illustrate the mean, minimum and maximum at each distillation step in Figure A.5. Each experiment is 11 chains of distillation steps, corresponding to $\alpha \in \{0.0, 0.1, \dots, 0.9\}$, with the first model initialized identically across all chains. The accuracy reported at the τ 'th step is based on comparing the training and validation predictions, $\mathbf{Y}^{(\tau)}$ and $f(\mathbf{X}, \hat{\beta}^{(\tau)})$ with the original training and validation targets; \mathbf{Y} and $\hat{\mathbf{Y}}$.

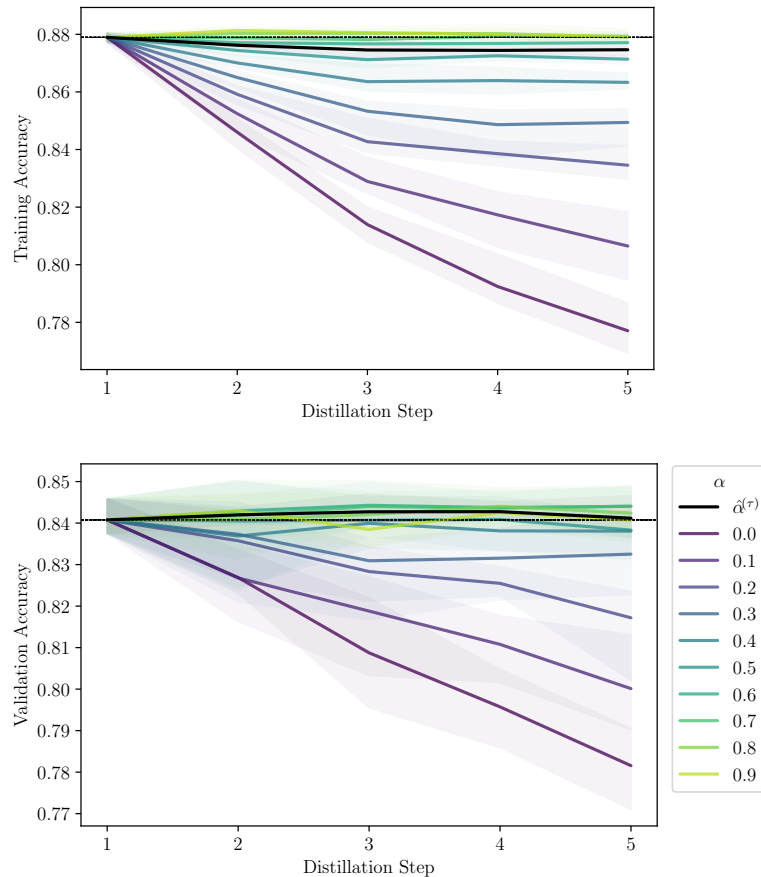


Figure A.6: (Identical to Figure A.5) Training and validation accuracy for five distillation steps with ResNet-50 models on CIFAR-10. Comparing fixed $\alpha^{(t)}$ for $t = 2, \dots, \tau$ and estimating optimal weight with $\hat{\alpha}^{(t)}$ at each step. The experiment is repeated four times and the mean (and max/min in shaded) is reported.

The theory introduced in Section A.4 suggests that self-distillation corresponds to a progressively amplified regularization of the solution, and larger α dampens the amount of regularization imposed by the procedure more than small values of α . Thus, for small

α we expect the training accuracy to decrease with each distillation, but for larger α we might experience an increase in training accuracy, due to additional training iterations and a sufficient amount of ground-truth target information being kept in the optimization problem, which could prove beneficial. However, depending on the need for increased regularization, we expect the validation accuracy to increase for some α , and possibly decrease for α values either too small or too large. The above properties are observed in Figure A.5, where $\alpha \leq 0.4$ generally overregularize the solution, and performance drops with distillation steps. For $\alpha > 0.4$ the performance generally improves with distillation, but for α close to one, the gains reduce, and a suitably balanced α for this experiment would be in $[0.5, 0.7]$. This aligns well with the optimal $\hat{\alpha}^{(\tau)}$ estimated at approx. 0.6 for each step of self-distillation.

A.III Illustrative Example

In Figure A.7 we show the training and validation loss associated with the illustrative example in Section A.4.5. Although subtly, we observe a performance improvement in the first few distillation steps across all three choices of α . However, for $\alpha = 0$ the solutions progressively underfit the data as τ increases further.

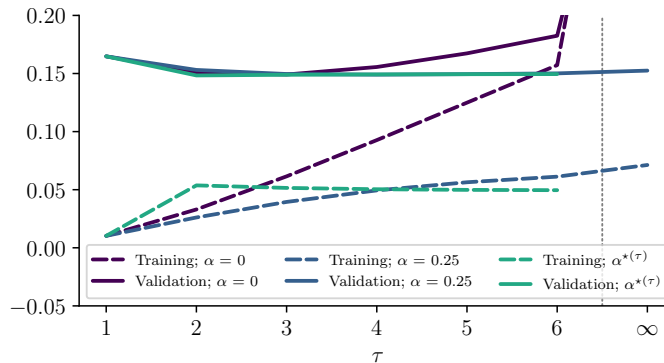


Figure A.7: Training (dashed lines) and validation (solid lines) loss associated with $\alpha = 0$, $\alpha = 0.25$, and $\alpha^*(\tau)$. Both losses associated with $\alpha = 0$ for $\tau \rightarrow \infty$ are huge compared to finite τ and the plot is bounded accordingly.

A.IV Connection to Neural Networks

This paper theoretically investigates self-distillation of kernel ridge regression models, but distillation procedures are more commonly used in a deep learning setting. However, recent research in the over-parameterized regime has shown great progress and connected wide neural networks with kernel ridge regression using the Neural Tangent Kernel (NTK) (Lee et al., 2019, 2020; Hu et al., 2020). The following is a brief and informal connection between kernel ridge regression and wide¹⁷ neural networks, motivating our problem setup and approach to estimate $\hat{\alpha}^{(\tau)}$ in Section A.5.

¹⁷ Note, we refer to width as the number of hidden nodes in a fully connected neural network or channels in a convolutional neural network.

Consider a neural network with scalar output $f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{R}$, where $\boldsymbol{\theta}(t) \in \mathbb{R}^D$ is the vector of all network parameters at training iteration $t \geq 0$, and $\mathbf{x} \in \mathbb{R}^d$ some input. We consider the case where we use gradient descent on the MSE objective, $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^n (f_{\text{nn}}(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)^2$ over some training dataset $\mathcal{D}_{\text{train}} \subseteq \mathbb{R}^d \times \mathbb{R}$. Consider the first-order Taylor-expansion of $f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta})$ w.r.t its parameters at initialization, $\boldsymbol{\theta}(0)$,

$$f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta}) \approx f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta}(0)) + \langle \nabla_{\boldsymbol{\theta}} f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta}(0)), \boldsymbol{\theta} - \boldsymbol{\theta}(0) \rangle \quad (\text{A.23})$$

where $f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta}(0))$ and $\nabla_{\boldsymbol{\theta}} f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta}(0))$ are constants w.r.t. $\boldsymbol{\theta}$. For sufficiently wide networks, (A.23) holds, and we say that we are in the *(NTK) regime* (Arora et al., 2019; Lee et al., 2019). Now, let $\varphi(\mathbf{x}) \stackrel{\text{def}}{=} \nabla_{\boldsymbol{\theta}} f_{\text{nn}}(\mathbf{x}, \boldsymbol{\theta}(0))$ for any $\mathbf{x} \in \mathbb{R}^d$, and denote the random kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) \stackrel{\text{def}}{=} \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle$ for any $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ (Jacot et al., 2018). For sufficiently wide networks, the random kernel converges to a deterministic kernel, and since the r.h.s. of (A.23) is linear, one can show that minimizing \mathcal{L} with gradient descent leads to the solution of the kernel regression problem, with the NTK; $\mathbf{x} \mapsto \kappa(\mathbf{x}, \mathbf{X})^\top \kappa(\mathbf{X}, \mathbf{X})^{-1} \mathbf{y}$, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the matrix of training inputs, and $\mathbf{y} \in \mathbb{R}^n$ the vector of training targets (Arora et al., 2019; Lee et al., 2019). It has been shown that when minimizing the ℓ_2 -regularized MSE loss, the solution becomes the kernel ridge regression solution (Lee et al., 2020).

The connections between neural networks and kernel ridge regressions in knowledge distillation settings have, to the best of our knowledge, not been explicitly investigated yet, but we hope that the results of this paper will improve the understanding of self-distillation of neural networks once such a connection is made rigorously.

A.V Connections to Constrained Optimization Problem

The setup investigated in this paper is the unconstrained optimization problem presented in (A.2), but some of the results can easily be extended to a constrained optimization problem, with a general regularization functional in Hilbert spaces, namely the natural extension of the setup proposed by Mobahi et al. (2020). For the rest of this section we assume $\alpha^{(2)} = \dots = \alpha^{(\tau)} = \alpha$. Mobahi et al. (2020) propose to solve the problem

$$f^{(\tau)} \stackrel{\text{def}}{=} \underset{f \in \mathcal{F}}{\text{argmin}} \int_{\mathcal{X}} \int_{\mathcal{X}} u(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \quad \text{s.t.} \quad (\text{A.24})$$

$$\frac{1}{N} \sum_{n=1}^N (f(\mathbf{x}_n) - y_n)^2 \leq \varepsilon,$$

where $\varepsilon > 0$ is a desired loss tolerance, $\tau \geq 1$, $f^{(0)}(\mathbf{x}_n) = y_n$ for $n = 1, \dots, N$, and u being symmetric and such that $\forall f \in \mathcal{F}$ the double integral is greater than or equal to 0 with equality only when $f(\mathbf{x}) = 0$.¹⁸ See Mobahi et al. (2020) for details.

The natural extension of this problem is to include ground-truth labels and solve the weighted problem

$$f^{(\tau)} = \underset{f \in \mathcal{F}}{\text{argmin}} \int_{\mathcal{X}} \int_{\mathcal{X}} u(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \quad \text{s.t.} \quad (\text{A.25})$$

$$\frac{\alpha}{N} \sum_{n=1}^N (f(\mathbf{x}_n) - y_n)^2 + \frac{1-\alpha}{N} \sum_{n=1}^N (f(\mathbf{x}_n) - f^{(\tau-1)}(\mathbf{x}_n))^2 \leq \varepsilon,$$

¹⁸ For a given u the function space \mathcal{F} is the space of functions f for which the double integral in (A.24) is bounded.

for $\tau \geq 1$, where $\alpha \in [0, 1]$ and $f^{(0)}(\mathbf{x}_n) = y_n$ for $n = 1, \dots, N$. In [Mobahi et al. \(2020\)](#), $\alpha = 0$, and this problem completely ignores the ground truth data after the first model fit, and it is easy to see that consecutive self-fits will be penalized increasingly stronger, and eventually collapse to zero, whenever $\frac{1}{N} \sum_{n=1}^N (f(\mathbf{x}_n) - y_n)^2 \leq \varepsilon$. The case $\alpha = 1$, corresponds to fitting to the ground-truth at each iteration, and do not benefit from distillation, and thus is without interest here.

A.V.1 Collapsing and Converging Conditions

The regularization functional of (A.25), is clearly minimized by $f_t(\mathbf{x}) = 0$, but in order for this to be a solution for some $\tau \geq 1$, it must hold that

$$\frac{\alpha}{N} \|\mathbf{y}\|_2^2 + \frac{1-\alpha}{N} \|\mathbf{y}^{(\tau-1)}\|_2^2 \leq \varepsilon,$$

where we use the notation that $\mathbf{y}^{(\tau)} = (f^{(\tau)}(\mathbf{x}_1), \dots, f^{(\tau)}(\mathbf{x}_N))^T$ and $\mathbf{y} = (y_1, \dots, y_N)$. For $\tau = 1$, this amounts to $\frac{1}{N} \|\mathbf{y}\|_2^2 \leq \varepsilon$, and for $\tau > 1$

$$\frac{1-\alpha}{N} \|\mathbf{y}^{(\tau-1)}\|_2^2 \leq \varepsilon - \frac{\alpha}{N} \|\mathbf{y}\|_2^2. \quad (\text{A.26})$$

But since the l.h.s. is non-negative, it is required that $\frac{\alpha}{N} \|\mathbf{y}\|_2^2 \leq \varepsilon$ in order for $f^{(\tau)}(\mathbf{x}) = 0$ to be a solution. Hence, we can construct the following settings, that determine the behavior of the solutions:

1. Collapsed solution:

$$\frac{1}{N} \|\mathbf{y}\|_2^2 \in [0, \varepsilon] \quad \implies \quad \|\mathbf{y}^{(\tau)}\|_2 = 0 \quad \forall \tau \geq 1$$

2. Converging to collapsed solution:

$$\frac{1}{N} \|\mathbf{y}\|_2^2 \in \left(\varepsilon, \frac{\varepsilon}{\alpha} \right] \quad \implies \quad \exists \underline{\tau} \geq 1 \text{ such that } \begin{cases} \|\mathbf{y}^{(\tau)}\|_2 > 0 & \forall \tau < \underline{\tau}, \\ \|\mathbf{y}^{(\tau)}\|_2 = 0 & \forall \tau \geq \underline{\tau}, \end{cases}$$

3. Converging to non-collapsed solution:

$$\frac{1}{N} \|\mathbf{y}\|_2^2 \in \left(\frac{\varepsilon}{\alpha}, \infty \right) \quad \implies \quad \|\mathbf{y}^{(\tau)}\|_2 > 0 \quad \forall \tau \geq 1.$$

If we let $\alpha \rightarrow 0$ the interval $\left(\varepsilon, \frac{\varepsilon}{\alpha} \right]$ effectively becomes (ε, ∞) , and any solution will collapse at some point ([Mobahi et al., 2020](#)). Analogously, if we let $\alpha \rightarrow 1$, the interval $\left(\varepsilon, \frac{\varepsilon}{\alpha} \right]$ effectively becomes empty, and all non-collapsed solutions will converge to a non-zero solution. Hence, if $\alpha > 0$, one can obtain non-collapsing convergence with infinite iterations. Furthermore, if we let $\varepsilon \rightarrow 0$, then $[0, \varepsilon]$ and $\left(\varepsilon, \frac{\varepsilon}{\alpha} \right]$ will practically collapse to empty intervals, and we will always obtain convergence to non-collapsing solutions, which will correspond to an interpolating solution. For the remainder we assume $\alpha \in (0, 1)$, since the boundary cases are covered in [Mobahi et al. \(2020\)](#) ($\alpha = 0$) or is without interest ($\alpha = 1$). Furthermore, we assume that $\|\mathbf{y}\|_2 > \sqrt{N\varepsilon}$ to avoid a collapsed

solution from the beginning. Utilizing the Karush-Kuhn-Tucker (KKT) conditions for this problem, we can rephrase our optimization problem as

$$f^{(\tau)} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{\alpha}{N} \sum_{n=1}^N (f(\mathbf{x}_n) - y_n)^2 + \frac{1-\alpha}{N} \sum_{n=1}^N (f(\mathbf{x}_n) - f^{(\tau-1)}(\mathbf{x}_n))^2 + \lambda_\tau \int_{\mathcal{X}} \int_{\mathcal{X}} u(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}',$$

where $\lambda_\tau \geq 0$. For suitably chosen λ_τ , one can show that $f^{(\tau)}$ is an optimal solution to our problem.¹⁹

A.V.2 Extending Our Results

By direct calculations similar to those of [Mobahi et al. \(2020\)](#) one can obtain the closed form solution of (A.27), but first, we will repeat some definitions from [Mobahi et al. \(2020\)](#). Let the Green's Function $g(x, t)$ be such that $\int_{\mathcal{X}} u(x, x') g(x', t) dx' = \delta(x - t)$, where δ is the Dirac delta, and let $[\mathbf{G}]_{j,k} = \frac{1}{N} g(\mathbf{x}_j, \mathbf{x}_k)$ and $[\mathbf{g}(\mathbf{x})]_k = \frac{1}{N} g(\mathbf{x}, \mathbf{x}_k)$, where \mathbf{G} is a matrix and $\mathbf{g}(\mathbf{x})$ a vector dependent on \mathbf{x} . Now we can present the proposition.

Proposition A.6. *For any $\tau \geq 1$, the problem (A.27) has a solution of the form*

$$\mathbf{y}^{(\tau)} = \mathbf{g}(\mathbf{x})^\top (\mathbf{G} + \lambda_\tau \mathbf{I})^{-1} (\alpha \mathbf{y} + (1 - \alpha) \mathbf{y}^{(\tau-1)}),$$

where $\mathbf{y}^{(0)} \stackrel{\text{def}}{=} \mathbf{y}$.

Since \mathbf{G} is positive semi-definite, we can decompose it as $\mathbf{G} = \mathbf{V} \mathbf{D} \mathbf{V}^\top$. Define $\mathbf{B}^{(0)} \stackrel{\text{def}}{=} \mathbf{I}$, then for $\tau \geq 1$, we have that $\mathbf{y}^{(\tau)} = \mathbf{V} \mathbf{B}^{(\tau)} \mathbf{V}^\top \mathbf{y}$, where we set

$$\mathbf{B}^{(\tau)} \stackrel{\text{def}}{=} \frac{\alpha}{1-\alpha} \sum_{i=1}^{\tau-1} (1-\alpha)^{\tau-i} \prod_{j=i}^{\tau-1} \mathbf{A}^{(j+1)} + (1-\alpha)^{\tau-1} \prod_{j=1}^{\tau} \mathbf{A}^{(j)},$$

$$\mathbf{A}^{(\tau)} \stackrel{\text{def}}{=} \mathbf{D} (\mathbf{D} + \lambda_\tau \mathbf{I})^{-1}.$$

By equivalent calculations as those of Lemma A.2, we have that

$$\mathbf{B}^{(\tau)} = \mathbf{A}^{(\tau)} ((1-\alpha) \mathbf{B}^{(\tau-1)} + \alpha \mathbf{I}_N) \quad \forall \tau \geq 1,$$

and we can now formulate the following theorem, similar to Theorem A.3.

Theorem A.7. *For any pair of diagonals of \mathbf{D} , i.e. d_k and d_j , where $d_k > d_j$, we have that for all $\tau \geq 1$ and for $\alpha \in (0, 1)$, then*

$$\operatorname{sign} \left(\frac{[\mathbf{B}^{(\tau)}]_{k,k}}{[\mathbf{B}^{(\tau)}]_{j,j}} - \frac{[\mathbf{B}^{(\tau-1)}]_{k,k}}{[\mathbf{B}^{(\tau-1)}]_{j,j}} \right) \quad (\text{A.27})$$

$$= \operatorname{sign} \left(\left(\left(\frac{[\mathbf{B}^{(\tau-1)}]_{k,k}}{[\mathbf{B}^{(\tau-1)}]_{j,j}} - \frac{[\mathbf{A}^{(\tau)}]_{k,k}}{[\mathbf{A}^{(\tau)}]_{j,j}} \right) \frac{[\mathbf{A}^{(\tau)}]_{j,j}}{[\mathbf{B}^{(\tau-1)}]_{k,k} ([\mathbf{A}^{(\tau)}]_{k,k} - [\mathbf{A}^{(\tau)}]_{j,j})} + 1 \right)^{-1} - \alpha \right). \quad (\text{A.28})$$

Proof. The proof follows analogously to the proof of Theorem A.3. \square

¹⁹ See [Mobahi et al. \(2020\)](#) for a detailed argument.

For the case $\alpha = 1$, the results are identical to Theorem A.3, since no distillation is actually performed. However, the case $\alpha = 0$ is more involved and we refer the reader to [Mobahi et al. \(2020\)](#) for the treatment of this case since our setups are identical when $\alpha = 0$.

Finally, due to the dependency of λ_τ on the solution from the previous step of distillation in this constrained optimization problem, we are unable to obtain a simple recurrent expression as (A.7) and a limiting solution as in Theorem A.5. However, by the results in Section A.V.1, we can determine, from the norm of the targets, whether the solution collapses or not.

Self-Distillation for Gaussian Process Models

SUBMITTED TO JOURNAL OF MACHINE LEARNING RESEARCH (JMLR)

Kenneth Borup

Aarhus University

Lars Nørvang Andersen

Aarhus University

Abstract. Knowledge distillation has empirically proven to be an effective technique for training a student model to reproduce a teacher model. However, a rigorous theoretical understanding of why distillation techniques work is largely absent. In this paper, we aim to remedy this gap between theory and practice by theoretically analyzing self-distillation for Gaussian process regression (GPR) and Gaussian process classification (GPC). We propose two approaches to extend the notion of self-distillation to Gaussian Processes, which we refer to as deterministic GPSD (d -GPSD) and probabilistic GPSD (ρ -GPSD). The d -GPSD approach resembles most current distillation techniques for machine learning, and refits a model on deterministic predictions from the teacher, while the ρ -GPSD approach, re-uses the full probabilistic posterior for the next iteration. By analyzing the properties of these methods, we show that the d -GPSD approach for GPR closely relates to known results for self-distillation and that the ρ -GPSD approach for GPR corresponds to ordinary GPR with a particular choice of hyperparameters. Furthermore, we demonstrate that the ρ -GPSD approach for GPC approximately corresponds to data duplication and a particular scaling of the covariance and that the d -GPSD approach for GPC requires redefining the model from a Binomial likelihood to a continuous Bernoulli likelihood to be well-specified. We illustrate our theoretical results with empirical examples. To our knowledge, our proposed approaches are the first to formulate self-distillation specifically for GP models.

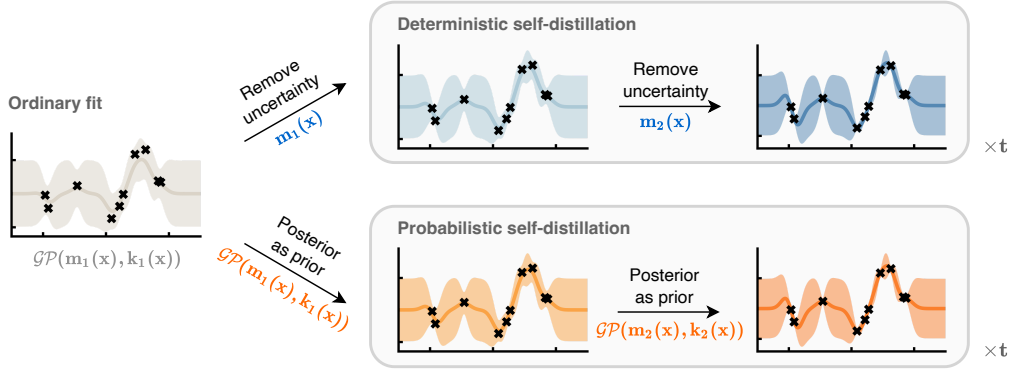


Figure B.1: Conceptual illustration of our two proposed methods of performing distillation for Gaussian processes. In deterministic self-distillation, we re-use the mean function of the previous solution in the next step, and in probabilistic self-distillation, we re-use the full distribution.

B.1 Introduction and Problem Setting

In this paper, we propose a notion of knowledge distillation (KD) in a Gaussian process (GP) context. Despite numerous research and applications of (extensions of) KD in the context of deep learning, research on knowledge distillation for other methods of machine learning than deep learning is lacking. Recent work by [Borup and Andersen \(2021\)](#); [Mobahi et al. \(2020\)](#); [Phuong and Lampert \(2019\)](#); [Frosst and Hinton \(2017\)](#) amongst others, provide results on KD for simplified settings, in particular for kernel ridge regression. A natural extension of these results is to investigate KD for Gaussian processes, and since no prior work has been performed in this context, we propose and analyze approaches on how one could define knowledge distillation for GP models. In this paper, we restrict our analysis to self-distillation, i.e. where the student and teacher models are of an identical class, which in practice corresponds to the chosen kernel function being identical for both models.

In ordinary GP models, one defines a prior, and, based on a set of observations \mathcal{D} , a posterior model is obtained which can be used for inference on previously unseen samples. We propose two approaches, illustrated in Figure B.1 and B.2, for using the posterior model in a self-distillation step: d -GPSD and ρ -GPSD.

Deterministic self-distillation (d -GPSD). In deterministic distillation, we treat the output predictions of a model at step t as the input to the model at step $t + 1$. This procedure aligns well with most current distillation methods known from deep learning ([Hinton et al., 2015](#)). However, unlike in deep learning, the output of our models is not merely scalars, but distributions. We propose to discard the stochasticity of our predictions at step t and merely keep the mean predictions for the training of the model at step $t + 1$. We show that for GP regression the mean after self-distillation is closely related to distillation with kernel ridge regression as in [Borup and Andersen \(2021\)](#); [Mobahi et al. \(2020\)](#). Furthermore, we argue that for distillation in a classification setting, one needs to adapt the likelihood function to support continuous targets rather than binary targets. We address this by utilization of the continuous Bernoulli distribution ([Loaiza-Ganem and Cunningham, 2019](#)). We investigate deterministic distillation for regression in Section B.3.1 and for classification in Section B.4.1.

Probabilistic self-distillation (ρ -GPSD). In probabilistic distillation, we replace the prior at a given step, t , with the posterior of the previous step, $t - 1$. In GP regression the posterior distribution is a GP itself, and we can directly consider it as a prior for the succeeding step. This yields an iteratively refined and data-dependent prior with each distillation step. For classification, more care has to be taken as the posterior is not known analytically. However, by utilizing the Laplace approximation of the posterior as the prior of the latent GP model, we are able to define a meaningful notion of probabilistic distillation in the classification setup as well. We investigate probabilistic for regression in Section B.3.2 and for classification in Section B.4.2.

We summarize our **contributions** as:

- We propose two different approaches to self-distillation for both GP regression and GP classification, reusing either the mean predictions (d -GPSD) or full predictive distribution (d -GPSD) of the previous iteration of the model.
- We show a connection between deterministic self-distillation for GP regression and self-distillation of kernel ridge regression. Furthermore, we prove that probabilistic self-distillation for GP regression is equivalent to ordinary GP regression with a particular choice of hyperparameter.
- We find that a naïve approach to deterministic self-distillation for GP classification yields a misspecified model due to the continuous form of the predictions, and we alleviate this by utilization of the continuous Bernoulli distribution. Furthermore, we find that probabilistic self-distillation can be efficiently approximated by a particular scaling of the covariance function for any number of distillation steps.

Limitations. The work in this paper should be seen as a theoretical contribution to the understanding of knowledge distillation by extending and analyzing the tractable kernel methods to their natural generalization (Phuong and Lampert, 2019; Mobahi et al., 2020; Borup and Andersen, 2021). We view this as a step towards understanding KD in neural networks, due to the connection between kernel methods and neural networks given by the Neural Tangent Kernel, see e.g. Jacot et al. (2018) and Arora et al. (2019). Hence, it is not the aim of the paper to provide numerical methods, which are superior to existing benchmarks.

Paper outline. The structure of the paper is as follows. In Section B.2 we present preliminary results on GP regression and GP classification. Next, we focus on self-distillation for GP regression in Section B.3, and in particular on deterministic distillation in Section B.3.1 and probabilistic distillation in Section B.3.2. Afterward, in Section B.4 we investigate the classification setup and in particular the deterministic setting in Section B.4.1 and probabilistic setting in Section B.4.2. Finally, we relate our work to existing research in Section B.6. Additionally, in the appendix, we provide additional mathematical details, implementation details, experiment ablations, as well as all proofs.

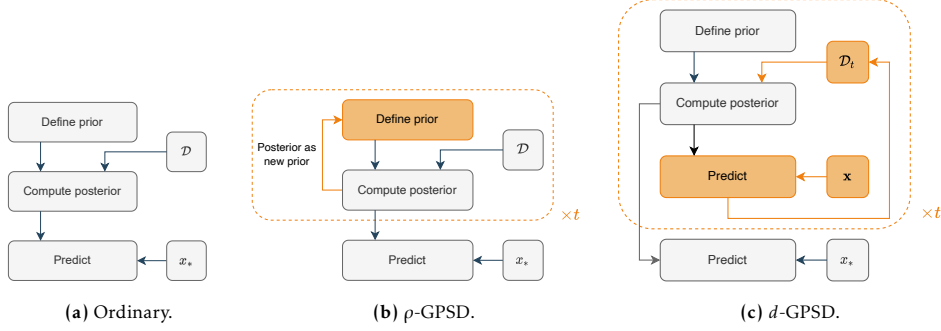


Figure B.2: In (a) we illustrate a typical procedure to fit a GP to an observed dataset, $\mathcal{D} = \{(x_i, y_i)\}_{i=0}^n$, and compute the output distribution for some unseen sample x_* . In (b) we illustrate the probabilistic self-distillation procedure, where we reuse the posterior as a prior. In (c), we illustrate the deterministic self-distillation procedure, where we iteratively replace the training observations $\mathcal{D}_t = \{(x_i, y_{t,i})\}_{i=0}^n$ with the posterior predictions of the current model, $\mathcal{D}_{t+1} = \{(x_i, y_{t+1,i})\}_{i=0}^n$, and refit the model to these samples.

B.2 Preliminaries

Our starting point is the standard set-up for Gaussian Process regression and classification. We use the standard results and notation, see [Bishop \(2006\)](#) and [Rasmussen and Williams \(2006\)](#).

B.2.1 Gaussian Process Regression

We assume a generic input-output pair $(\mathbf{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ is related by $y = f(\mathbf{x}) + \varepsilon$, where $f(\cdot)$ is a function, which we wish to infer, based on the assumption that f is a random function whose prior distribution is $f \sim \mathcal{GP}(m, k)$ for mean function $m: \mathbb{R}^d \rightarrow \mathbb{R}$ and positive semi-definite covariance function (kernel) $k: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, and ε is an independent noise term with $\varepsilon \sim \mathcal{N}(0, \gamma)$. For notational convenience, we will assume that our input is univariate, i.e. $d = 1$. For a training set $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, N\}$ and test set $(x_{*1}, \dots, x_{*M})^\top$, the first task is to describe the posterior distribution of $\mathbf{f}_* = (f(x_{*1}), \dots, f(x_{*M}))^\top$ given \mathcal{D} . The posterior predictive distribution of \mathbf{f}_* given \mathcal{D} is

$$\begin{aligned} \mathbf{f}_* \mid \mathbf{x}, \mathbf{y}, \mathbf{x}_* &\sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), \quad \text{where} \\ \boldsymbol{\mu}_* &= m(\mathbf{x}_*) + k(\mathbf{x}_*, \mathbf{x}^\top)(\mathbf{K} + \gamma \mathbf{I}_N)^{-1}(\mathbf{y} - m(\mathbf{x})), \\ \boldsymbol{\Sigma}_* &= k(\mathbf{x}_*, \mathbf{x}_*^\top) - k(\mathbf{x}_*, \mathbf{x}^\top)(\mathbf{K} + \gamma \mathbf{I}_N)^{-1}k(\mathbf{x}, \mathbf{x}_*^\top). \end{aligned}$$

and where \mathbf{x} and \mathbf{x}_* are stacked training and test input, respectively.

Throughout the paper, we will write \mathbf{K} for the matrix $\mathbf{K} = k(\mathbf{x}, \mathbf{x}^\top)$ which we assume is invertible. Furthermore, we will often notationally omit the conditioning on \mathbf{x} and \mathbf{x}_* for ease of exposition.

B.2.2 Gaussian Process Classification

We follow the usual setup for classification with Gaussian Processes in which each input-output pair (\mathbf{x}, y) is related by the assumption that the conditional distribution $y \mid f(\mathbf{x})$ is a Bernoulli distribution with probability $\sigma(f(\mathbf{x}))$ where $\sigma(\cdot)$ is the logistic function and the prior distribution of f is $\mathcal{GP}(m, k)$.

Unlike the regression case, the posterior distribution $\mathbf{f} | \mathbf{y}$ is not analytically tractable and needs to be approximated. This issue is not specific to distillation but arises in any practical application involving Gaussian process classification, and a number of approaches have been suggested for obtaining an approximation of the posterior distribution. In this paper, we will focus on the Laplace approximation, where the posterior distribution is approximated by

$$q(\mathbf{f} | \mathbf{y}) = \mathcal{N}(\mathbf{f} | \hat{\mathbf{f}}, (\mathbf{K}^{-1} + \mathbf{W})^{-1})$$

where $\hat{\mathbf{f}} = (\hat{f}_n)_{n=1}^N$ is the mode of $\psi(\mathbf{f})$ and \mathbf{W} is the $N \times N$ matrix $\text{diag}_n\{\sigma(\hat{f}_n)(1 - \sigma(\hat{f}_n))\}$. The approximate posterior predictive distribution is found in [Rasmussen and Williams \(2006, Section 3.4.2\)](#) when $m = 0$ and for a general prior mean m , we have $p(f_* | \mathbf{x}, \mathbf{y}) \approx \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$ where

$$\boldsymbol{\mu}_* = \mathbb{E}_q[\mathbf{f}_* | \mathbf{y}, \mathbf{x}_*] = m(\mathbf{x}_*) + k(\mathbf{x}_*, \mathbf{x}^\top) \mathbf{K}^{-1} (\hat{\mathbf{f}} - m(\mathbf{x})) \quad (\text{B.1})$$

$$\boldsymbol{\Sigma}_* = \text{Cov}_q(\mathbf{f}_* | \mathbf{y}, \mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*^\top) - k(\mathbf{x}_*, \mathbf{x}^\top) (\mathbf{K} + \mathbf{W}^{-1})^{-1} k(\mathbf{x}, \mathbf{x}_*^\top). \quad (\text{B.2})$$

See Section B.II.1 for additional details on Gaussian Process classification. We return to the classification setting in Section B.4.

B.3 Self-Distillation for GP Regression

We now consider deterministic and probabilistic self-distillation for GP regression (GPR) separately.

B.3.1 Deterministic Self-Distillation (d -GPSD)

For the general setup of deterministic distillation, we consider a mean-zero prior distribution, $f \sim \mathcal{GP}(0, k)$, and the posterior distribution on $\mathbf{x} = (x_1, \dots, x_N)^\top$ at step $t \geq 1$ as $\mathbf{f}_t | \mathbf{y}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, where

$$\boldsymbol{\mu}_t = \mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{y}_{t-1}, \quad \text{and} \quad \boldsymbol{\Sigma}_t = \mathbf{K} - \mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{K},$$

and $\gamma_t > 0$ for $t \geq 1$. Deterministic self-distillation corresponds to using the mean predictions $\boldsymbol{\mu}_t$ in place of \mathbf{y}_t when fitting the succeeding step, thereby fitting the subsequent model to the mean predictions of the current model. In this setting the behavior of the mean function becomes equivalent to that of [Mobahi et al. \(2020\)](#) and [Borup and Andersen \(2021\)](#) as evident from the following result.

Proposition B.1. *Define $\mathbf{y}_t \stackrel{\text{def}}{=} \boldsymbol{\mu}_t$ and $\mathbf{y}_0 \stackrel{\text{def}}{=} \mathbf{y}$. Then it holds for any $t \geq 1$ that $\mathbf{f}_t | \boldsymbol{\mu}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, where*

$$\mathbf{y}_t = \left(\prod_{s=1}^t \mathbf{K}(\mathbf{K} + \gamma_s \mathbf{I}_N)^{-1} \right) \mathbf{y}, \quad (\text{B.3})$$

$$\mathbb{E}[\mathbf{f}_{t,*} | \boldsymbol{\mu}_{t-1}] = k(\mathbf{x}_*, \mathbf{x}) (\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{y}_{t-1}, \quad (\text{B.4})$$

and $\boldsymbol{\Sigma}_t = \mathbf{K} - \mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{K}$ as well as \mathbf{y}_t is only affected by the choice of γ_t .

Proof. See proof on page 88. □

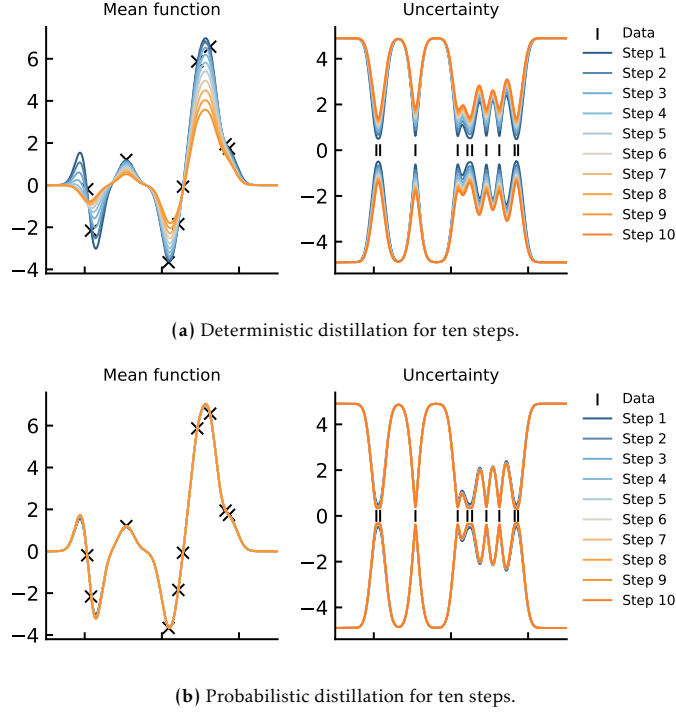


Figure B.3: Ten steps of self-distillation with fixed hyperparameters and $(\gamma_1, \dots, \gamma_{10}) = (0.1, \dots, 1)$ for both the deterministic and probabilistic approach. We plot both the mean function and the 2.5 and 97.5 percentiles of the uncertainty estimate at each step.

It can be shown that the behavior of \mathbf{y}_t as t increases corresponds to progressively sparsifying the underlying basis functions by shrinking the basis functions with the smallest eigenvalues the most, thereby imposing a regularization effect on the predictions (Mobahi et al., 2020; Borup and Andersen, 2021). Furthermore, since \mathbf{y}_t only depend on t through $\{\gamma_s\}_{s=1}^t$, once can efficiently compute (B.3) for any choice of t , by using a spectral decomposition of \mathbf{K} .

Furthermore, following Borup and Andersen (2021) the results can be extended to include a weighting (by α) of the original training observations and (by $1 - \alpha$) of the predictions from the previous iteration of the model.

B.3.2 Probabilistic Self-Distillation (ρ -GPSD)

The central observation of the probabilistic approach is that the posterior distribution is itself a Gaussian Process, namely $\mathcal{GP}(m_*, k_*)$, where

$$m_*(x) = m(x) + k(x, \mathbf{x}^\top) (\mathbf{K} + \gamma \mathbf{I}_N)^{-1} (\mathbf{y} - m(\mathbf{x}))$$

$$k_*(x, y) = k(x, y) - k(x, \mathbf{x}^\top) (\mathbf{K} + \gamma \mathbf{I}_N)^{-1} k(\mathbf{x}, y).$$

Since the posterior distribution is a Gaussian process, we may “distill” the posterior distribution, by using it as a prior distribution in our initial set-up, and we may iterate this procedure for any finite number of steps, see Figure 1 (b). Furthermore, we may select the noise terms γ_t to be different in each step, and we see that the distillation procedure

leads to a series of Gaussian Processes $\mathcal{GP}(m_t, k_t)$ where the mean- and covariance function satisfy the recursions

$$m_{t+1}(x) = m_t(x) + k_t(x, \mathbf{x}^\top) [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} (\mathbf{y} - m_t(\mathbf{x})) \quad (\text{B.5})$$

$$k_{t+1}(x, y) = k_t(x, y) - k_t(x, \mathbf{x}^\top) [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} k_t(\mathbf{x}, y) \quad (\text{B.6})$$

where $\mathbf{K}_t \stackrel{\text{def}}{=} k_t(\mathbf{x}, \mathbf{x}^\top)$, $m_0(x) \stackrel{\text{def}}{=} 0$, $k_0(x, y) \stackrel{\text{def}}{=} k(x, y)$ and $\gamma_t > 0$, $t = 0, 1, \dots$

The solutions to the above recursions (B.5) and (B.6) are given in Theorem B.3 below, where we show that iterating the recursions t times with penalty parameters $\gamma_0, \gamma_1, \dots, \gamma_{t-1}$ yields the same result as performing a single iteration with penalty parameter $1/\gamma_{t-1}^-$ where $\gamma_t^- \stackrel{\text{def}}{=} \sum_{s=0}^t 1/\gamma_s$ and $\gamma_{-1}^- \stackrel{\text{def}}{=} 0$. To show this, we first need a lemma.

Lemma B.2. *Let $\lambda_i^{(t)}$, for $i = 1, \dots, N$, denote the eigenvalues of \mathbf{K}_t . Write $\mathbf{K}_0 = \mathbf{O} \text{diag}_i(\lambda_i^{(0)}) \mathbf{O}^\top$ for the spectral decomposition of \mathbf{K}_0 . Then we have for $t = 0, 1, \dots$ that*

$$\mathbf{K}_t = \mathbf{O} \text{diag}_i(\lambda_i^{(0)} \zeta_i^{(t)}) \mathbf{O}^\top \quad (\text{B.7})$$

where $\zeta_i^{(t)} \stackrel{\text{def}}{=} \frac{1}{\lambda_i^{(0)} \gamma_{t-1}^- + 1}$. For $k_t(x, \mathbf{x}^\top)$ and $m_t(\mathbf{x})$ we have

$$k_t(x, \mathbf{x}^\top) = k_0(x, \mathbf{x}^\top) \mathbf{O} \text{diag}_i(\zeta_i^{(t)}) \mathbf{O}^\top \quad (\text{B.8})$$

$$m_t(\mathbf{x}) = \mathbf{O} \text{diag}_i(\lambda_i^{(0)} \gamma_{t-1}^- \zeta_i^{(t)}) \mathbf{O}^\top \mathbf{y}. \quad (\text{B.9})$$

Proof. See proof on page 88. □

We may now prove the main result of this section.

Theorem B.3. *The solution to the recursions (B.5) and (B.6) is given by*

$$m_t(x) = k_0(x, \mathbf{x}^\top) (\mathbf{K} + \mathbf{I}_N / \gamma_{t-1}^-)^{-1} \mathbf{y} \quad (\text{B.10})$$

$$k_t(x, y) = k_0(x, y) - k_0(x, \mathbf{x}^\top) (\mathbf{K} + \mathbf{I}_N / \gamma_{t-1}^-)^{-1} k_0(\mathbf{x}, y)$$

for $t = 1, 2, \dots$

Proof. See proof on page 91. □

It follows from Theorem B.3, that performing multiple steps of probabilistic self-distillation is equivalent to fitting an ordinary GP with a particular choice of noise parameter. Next, if we assume that the noise parameter is identical in all distillation steps, i.e. $\gamma_t = \gamma$ for $t = 0, 1, \dots$, then $\gamma_{t-1}^- = t/\gamma$ which corresponds to fitting a GPR to a dataset \mathcal{D}_t , which consists of t replications of \mathcal{D} with noise parameter γ , in a sense made precise by the following corollary.

Corollary B.4. *If we let $\mathcal{D}_t = \{(x_{ij}, y_{ij}) \mid i = 1, \dots, N, \text{ and } j = 1, \dots, t\}$ where $x_{ij} = x_i$ for all i and j , then the posterior distribution of \mathbf{f} is $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ where*

$$\boldsymbol{\mu}_t = \begin{pmatrix} \mathbf{K}(\mathbf{K} + \gamma/t \mathbf{I}_N)^{-1} \mathbf{y} \\ \vdots \\ \mathbf{K}(\mathbf{K} + \gamma/t \mathbf{I}_N)^{-1} \mathbf{y} \end{pmatrix} \quad (\text{B.11})$$

$$\boldsymbol{\Sigma}_t = \mathbf{1}\mathbf{1}^\top \otimes (\mathbf{K} - \mathbf{K}(\mathbf{K} + \gamma/t \mathbf{I}_N)^{-1} \mathbf{K}) \quad (\text{B.12})$$

where \otimes is the Kronecker product.

Proof. See proof on page 92. □

B.3.3 Illustratory Example

We now consider an illustratory example, where we assume a true function $g(z) = z \sin(z)$, 10 training samples \mathbf{x} equidistantly distributed between 0 and 10, and observed values $y_i = g(x_i) + \varepsilon_i$, where ε_i is standard normally distributed. We use a scaled radial basis kernel with two hyperparameters σ_f and l as well as noise parameters γ_t . See Section B.III for further details on the setup.

We now consider 10 steps of both deterministic and probabilistic self-distillation with fixed hyperparameters for all steps. In Figure B.3a and Figure B.3b we plot the results for deterministic distillation with $(\gamma_1, \dots, \gamma_{10}) = (0.1, \dots, 1)$. Here, we observe the expected increase in regularization of the mean, and that the uncertainty increases with each step as expected from the choice of $(\gamma_1, \dots, \gamma_{10})$. This aligns well with our theory. However, for the probabilistic case, the mean function is nearly unaffected by the distillation procedure, and the uncertainty estimate does not change much either. This is due to the relatively small change in $1/\gamma_t^-$ when t goes from 1 to 10 as $1/\gamma_1^- = 0.1$ and $1/\gamma_{10}^- = 0.034$, which yield a small change for each distillation step. See Appendix B.III for additional examples with different choices of parameters.

B.4 Self-Distillation for GP Classification

In the following, we separately investigate deterministic and probabilistic self-distillation for GP classification.

B.4.1 Deterministic Self-Distillation (d -GPSD)

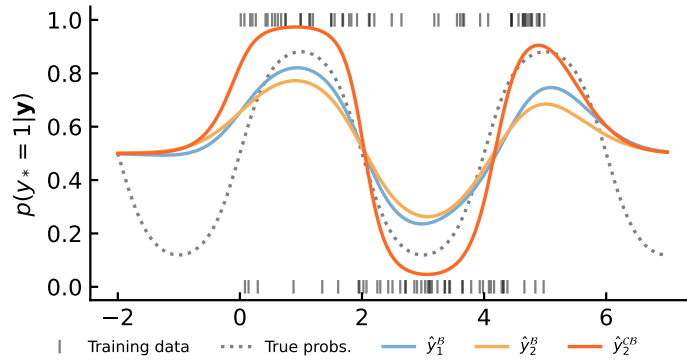


Figure B.4: Ordinary GPC model and one step of deterministic distillation, with both continuous and ordinary Bernoulli likelihood. We note that the Bernoulli likelihood, although well-performing, does not constitute a well-specified model.

Similarly to the deterministic approach for GPR, we will in this section consider the posterior mean predictions as deterministic targets for the next iteration of the model. In particular, at each step, t , we assume the prior of f_t to be $\mathcal{GP}(0, k)$, and for $t = 1$ we will obtain predictions, $\hat{\mathbf{y}}_0$ as the average posterior prediction based on the Laplace approximation e.g. $\hat{\mathbf{y}}_0 = \mathbb{E}_q[\sigma(\mathbf{f}_1) | \mathbf{y}_0]$. These predictions will then be used in place of the original observations for step $t = 2$, which will lead to new predictions $\hat{\mathbf{y}}_2$, that we can use in the succeeding step, and so on.

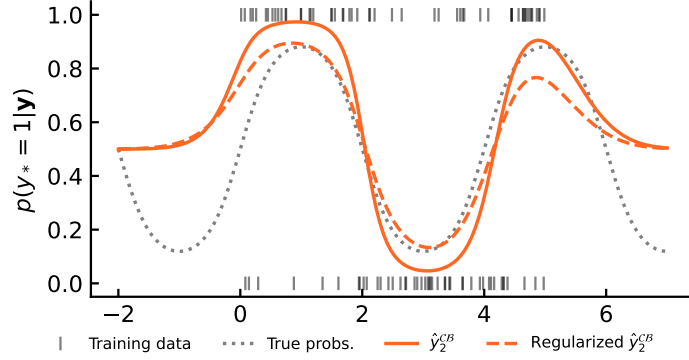


Figure B.5: One step of deterministic distillation with continuous Bernoulli likelihood as in Figure B.4, both with and without regularization imposed by a noise assumption on the observations. Adjusting the parameter γ_1 adjusts the regularization.

Since the original observations, \mathbf{y}_0 , are in $\{0, 1\}^N$ and the predictions, $\hat{\mathbf{y}}_1$, are in $[0, 1]^N$, directly reusing the predictions, $\hat{\mathbf{y}}_1$, in place of the observations, \mathbf{y}_0 , in the subsequent step yields a misspecified model as the Bernoulli distribution only has support on $\{0, 1\}$. To alleviate this misspecification, we redefine our model for $t \geq 2$ to assume a conditional *continuous* Bernoulli distribution rather than the usual conditional Bernoulli (Loaiza-Ganem and Cunningham, 2019). We will occasionally refer to this method as C-GPC, and in practice, this requires that we multiply each element of (B.15) by the appropriate normalizing constant, C (see (B.20) and Appendix B.IV). That is, for $t \geq 2$ we assume

$$\mathbf{y}_{t-1} | \mathbf{f}_t \sim \prod_{n=1}^N C(\sigma(f_{t,n})) \sigma(f_{t,n})^{y_{t-1,n}} (1 - \sigma(f_{t,n}))^{1-y_{t-1,n}},$$

which in turn affects the log posterior $\log \tilde{p}(\mathbf{f}_t | \mathbf{y}_{t-1})$ through $\log \tilde{p}(\mathbf{y}_{t-1} | \mathbf{f}_t)$ which now becomes

$$\log \tilde{p}(\mathbf{y}_{t-1} | \mathbf{f}_t) = \log p(\mathbf{y}_{t-1} | \mathbf{f}_t) + \sum_{n=1}^N \log C(\sigma(f_{t,n})).$$

Furthermore, this also affects the gradient and hessian of $\log \tilde{p}(\mathbf{f}_t | \mathbf{y}_{t-1})$ used to obtain the Laplace approximation, and although $\log C(\lambda)$ nor $\frac{d}{d\lambda} \log C(\lambda)$ attains no simple expressions, we show in Proposition B.5 that both $C(\sigma(a))$ and the derivatives $\frac{d}{da} \log C(\sigma(a))$ and $\frac{d^2}{da^2} \log C(\sigma(a))$ yield surprisingly simple expressions of standard hyperbolic functions (see also Figure B.14 for plots of the functions).

Proposition B.5. *Let $C(\sigma(a))$ be defined as in (B.20) with $\lambda = \sigma(a)$, then we have that*

$$C(\sigma(a)) = \begin{cases} 2 & \text{if } a = 0 \\ \operatorname{acoth}\left(\frac{a}{2}\right) & \text{otherwise,} \end{cases}$$

$$\frac{d}{da} \log(C(\sigma(a))) = \begin{cases} 0 & \text{if } a = 0 \\ \frac{1}{a} - \frac{1}{\sinh(a)} & \text{otherwise,} \end{cases}$$

$$\frac{d^2}{da^2} \log(C(\sigma(a))) = \begin{cases} \frac{1}{6} & \text{if } a = 0 \\ -\frac{1}{a^2} + \frac{\operatorname{coth}(a)}{\sinh(a)} & \text{otherwise,} \end{cases}$$

where \coth , and \sinh are the hyperbolic cotangent, and sine functions, respectively.

Proof. See proof on page 93. □

Thus, if we denote $\mathbf{C}_t \stackrel{\text{def}}{=} \sum_{n=1}^N \log(C(\sigma(f_{t,n})))$, we get that the gradient and hessian of the log posterior becomes

$$\begin{aligned}\nabla \tilde{\psi}(\mathbf{f}_t) &= \mathbf{y}_{t-1} - \sigma_{t-1} - \mathbf{K}^{-1} \mathbf{f}_t + \nabla \mathbf{C}_t, \\ -\nabla \nabla \tilde{\psi}(\mathbf{f}_t) &= \mathbf{W}_t + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_t,\end{aligned}$$

where $\mathbf{W}_t = \text{diag}_n\{\sigma(f_{t,n})(1 - \sigma(f_{t,n}))\}$ and $\nabla \nabla \mathbf{C}_t = \text{diag}_n\{\frac{d^2}{df_{t,n}^2} \log C(\sigma(f_{t,n}))\}$, respectively. This also affects the Laplace approximation of the posterior as evident from Figure B.4.

Unfortunately, unlike deterministic self-distillation for GPR, we are unable to obtain a closed-form solution for the distilled models due to the numerical approximations necessary to obtain the posterior (predictions). However, we do empirically investigate the effect of using the continuous Bernoulli rather than the discrete Bernoulli distribution in Figure B.4. In particular, we plot the solutions for a single step of distillation with C-GPC and with the discrete Bernoulli where we use continuous observations despite the support of the Bernoulli distribution being discrete. We expand on this example in Section B.4.3 below.

B.4.2 Probabilistic Self-Distillation (ρ -GPSD)

In the probabilistic approach, we proceed in a similar manner as in Section B.3.2. Firstly, we observe from (B.1) and (B.2) that our approximate posterior distribution is a Gaussian process $\mathcal{GP}(m_*, k_*)$ with

$$\begin{aligned}m_*(x) &= m(\mathbf{x}) + k(\mathbf{x}, \mathbf{x}^\top) \mathbf{K}^{-1} (\hat{\mathbf{f}} - m(\mathbf{x})) \\ k_*(x, y) &= k(x, y) - k(x, \mathbf{x}^\top) (\mathbf{K} + \mathbf{W}^{-1})^{-1} k(\mathbf{x}, y).\end{aligned}$$

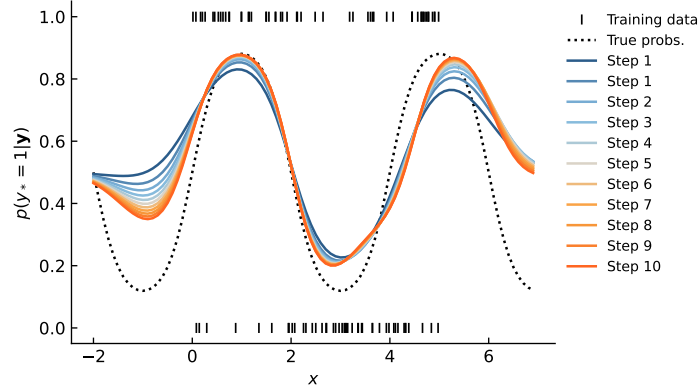
This may then be used as a prior distribution, which again implies a series of Gaussian processes $\mathcal{GP}(m_t, k_t)$, $t = 0, 1, 2, \dots$ where analogue to equations (B.5) and (B.6) in this case becomes

$$m_{t+1}(x) = m_t(x) + k_t(x, \mathbf{x}^\top) \mathbf{K}_t^{-1} (\hat{\mathbf{f}}_t - m_t(\mathbf{x})) \tag{B.13}$$

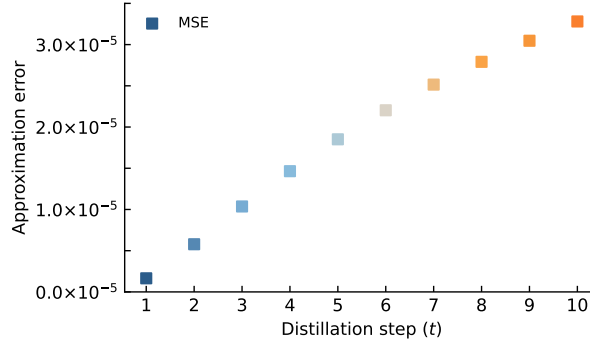
$$k_{t+1}(x, y) = k_t(x, y) - k_t(x, \mathbf{x}^\top) (\mathbf{K}_t + \mathbf{W}_t^{-1})^{-1} k_t(\mathbf{x}, y). \tag{B.14}$$

Here, $\hat{\mathbf{f}}_t$ is the mode of $\psi_t(\mathbf{f})$, the log posterior for the t 'th iteration, and similarly $\mathbf{W}_t = \text{diag}_n\{\sigma(\hat{f}_{t,n})(1 - \sigma(\hat{f}_{t,n}))\}$.

Since $\hat{\mathbf{f}}_t$ is not given analytically, we cannot solve the above recursions in the classification case, as we could in the regression setting and must resort to numerical results. However, we show an analogous result to the data duplication result from Corollary B.4 and it follows that iterated distillation in the probabilistic setup may be approximated by scaling the covariance in the first distillation step. These observations are summarized in the following proposition, where \mathcal{D}_t as in Corollary B.4 consists of the dataset \mathcal{D} replicated t times, and an example is illustrated in Figure B.6a and B.6b.



(a) Solutions for 10 steps of distillation.



(b) Error between iterated and scaled solutions.

Figure B.6: Ten steps of probabilistic self-distillation for GP classification. In (a) we plot the solutions using the scaled covariance approximation, and in (b) we plot the approximation error between the iterated distillation procedure and scaled approximation over the test set of 90 equidistant point on the interval $[-2, 7]$.

Proposition B.6. (A) A single step of probabilistic distillation using \mathcal{D}_t and a Gaussian prior $\mathcal{GP}(0, k)$ yields the same posterior distribution as a single step of probabilistic distillation using a Gaussian prior $\mathcal{GP}(0, tk)$. (B) Performing t iterations of the probabilistic distillation using \mathcal{D} and starting with an initial Gaussian prior $\mathcal{GP}(0, k)$ yields approximately the same posterior as a single step of probabilistic distillation using a Gaussian prior $\mathcal{GP}(0, tk)$.

Proof. See proof on page 95. □

B.4.3 Illustratory Example

We now consider an illustratory example, where we consider the true underlying function $g(x) = 2\sin(x\pi/2)$. We sample training observations, \mathbf{x} , from an $\mathcal{U}(0, 5)$ distribution and binary targets from a Bernoulli distribution with probability parameterized by $g(x)$.

d -GPSD for GPC. For d -GPSD, we initially fit an ordinary GPC model with Bernoulli distribution to the binary observations and fit a secondary model to the continuous predictions of this first model. In Figure B.4 we plot the predictions of both these

models when we use either a continuous Bernoulli distribution (as argued in Section B.4) or a discrete Bernoulli distribution (where we intentionally violate the discrete support of the distribution) for the second model. The hyperparameters of the models minimize the negative log-likelihood (NLL) over the input data, and while the distilled Bernoulli model is slightly flattened, compared to the initial fit, the C-GPC model yields more “extreme” predictions. We conjecture that the implicit regularization imposed by the misspecification of the distilled Bernoulli model is absent in the C-GPC model. Furthermore, by Section B.IV we expect the continuous Bernoulli to favor more extreme values compared to the ordinary Bernoulli, as is also evident in Figure B.4. To dampen the degree of overfitting in the C-GPC model we reintroduce the noise parameter $\gamma_t > 0$ along the diagonal of the kernel matrix of the training data and plot both the regularized and non-regularized solutions in Figure B.5. We observe that with the noise parameter, we get not only a well-specified model but also get more control over the amount of regularization. See Appendix B.III for more experiments.

ρ -GPSD for GPC. Again, we initially fit an ordinary GPC model, but unlike the d -GPSD case, for ρ -GPSD distillation, we do not change the model assumptions for the distillation step. In Figure B.6a we plot the solutions for 10 steps of distillation and observe a significantly different behavior compared to d -GPSD; the effect of distillation is much more subtle and more closely resembles that of distillation in the regression case where the solutions are getting progressively closer to the original data. Furthermore, in Figure B.6b we plot the mean squared approximation error between the solutions obtained from (B.5)-(B.6) and those obtained by scaling the kernel matrix as argued in Proposition B.6. Although the error is almost linearly increasing, it does so by less than 0.5×10^{-5} per step, and the approximation error is on the order of 10^{-5} for 10 steps. Thus, since the scaling approximation is much faster (especially for many distillation steps) it is a great alternative for naïve computations.

B.5 Additional Experiments

To further illustrate our theoretical results on empirical data, we include additional experiments on both regression and classification tasks on realistic data in Appendix B.VI. For these experiments, we apply common training strategies such as hyperparameter optimization by minimization of the negative log-likelihood and a cross-validated grid search over the noise parameter, kernels, and the number of distillation steps.

B.6 Related Works

To the best of our knowledge, no literature on knowledge distillation for Gaussian process models exists. However, for good measure in the following, we relate our proposed methods to the existing related literature on knowledge distillation and Gaussian process regression/classification.

Knowledge Distillation. The idea of knowledge distillation originates back to [Ba and Caruana \(2014\)](#); [Bucila et al. \(2006\)](#), but is typically most known from [Hinton et al. \(2015\)](#). It was originally aimed at model compression, by training a small (in an

appropriate measure of model size) *student* model on the outputs of a larger *teacher* model.

In recent years, the idea of distillation has expanded in numerous directions, and for various types of applications. Of particular relevance is the direction of self-distillation, where the student model is considered to be of the same model class/architecture as the teacher model (Furlanello et al., 2018; Zhang et al., 2019; Allen-Zhu and Li, 2022). Other interesting directions of research on distillation are on the impact of the size-gap between teacher and student (Mirzadeh et al., 2020), matching on other information than the output (Park et al., 2019; Romero et al., 2015; Srinivas and Fleuret, 2018), as well as multi-teacher and multi-source distillation (Li et al., 2021; Liu et al., 2020; You et al., 2017; Borup et al., 2023b). Finally, due to the domain-agnostic nature of most distillation methods, they are applied in a wide range of domains (Parthasarathi and Strom, 2019; Borup et al., 2023a).

Despite much empirical success of a wide array of distillation techniques, a rigorous understanding of distillation is still lacking behind. However, recent work has provided insights into simplified settings such as linear models, kernel ridge regression, and decision trees (Phuong and Lampert, 2019; Borup and Andersen, 2021; Frosst and Hinton, 2017).

Gaussian Processes. In recent years, an increasing amount of research on extending the capabilities of GP models and adapting them to modern compute availability and dataset sizes has been made. This includes theoretical connections between neural networks and Gaussian processes (Lee et al., 2018; Yang, 2019; Garriga-Alonso et al., 2019), stochastic and sparse GPs (Titsias, 2009; Hensman et al., 2013, 2015), and the possibility of exploiting the successes of neural networks in combination with Gaussian processes e.g. deep kernel learning (Wilson et al., 2016a,b).

B.7 Conclusion

In conclusion, we propose two approaches to extend knowledge distillation to Gaussian process regression and classification; deterministic and probabilistic. We show that some of these approaches are closely related to known results or very particular choices of hyperparameters, but also that some methods require careful redefinition of our model assumptions to be appropriate. To the best of our knowledge, this paper is the first attempt to propose knowledge distillation specifically for Gaussian Process models, opening new avenues for further research with potential implications in machine learning. Interesting directions of future research include, amongst others, the utilization of a convex combination of both the original and predicted targets in the distillation steps, as well as combinations of d -GPSD and ρ -GPSD distillation approaches.

Bibliography

Allen-Zhu, Z. and Li, Y. (2022). Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*. Cited on page 71.

- Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, volume 32. Cited on page 61.
- Ba, J. L. and Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems*, volume 27, pages 2654–2662. Cited on page 70.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, volume 1. Springer, New York, NY. Cited on pages 62 and 76.
- Blackard, J. (1998). Covertypes. UCI Machine Learning Repository. Cited on page 86.
- Borup, K. and Andersen, L. N. (2021). Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation. In *Advances in Neural Information Processing Systems*, volume 34, pages 5316–5327. Cited on pages 60, 61, 63, 64, 71, and 84.
- Borup, K., Kidmose, P., Phan, H., and Mikkelsen, K. (2023a). Automatic sleep scoring using patient-specific ensemble models and knowledge distillation for ear-EEG data. *Biomedical Signal Processing and Control*, 81:104496. Cited on page 71.
- Borup, K., Phoo, C. P., and Hariharan, B. (2023b). Distilling from Similar Tasks for Transfer Learning on a Budget. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11431–11441. Cited on page 71.
- Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pages 535–541. Cited on page 70.
- Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least Angle Regression. *The Annals of Statistics*, 32(2):407–499. Cited on page 86.
- Frosst, N. and Hinton, G. (2017). Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*. Cited on pages 60 and 71.
- Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born Again Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1607–1616. PMLR. Cited on page 71.
- Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. (2019). Deep convolutional networks as shallow Gaussian processes. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on page 71.
- Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290. Cited on page 71.
- Hensman, J., Matthews, A. G., and Ghahramani, Z. (2015). Scalable variational Gaussian process classification. *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 351–360. Cited on page 71.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*. Cited on pages 60 and 70.

- Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. *Cited on page 86.*
- Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pages 8571–8580. *Cited on page 61.*
- Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2018). Deep neural networks as Gaussian processes. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. *Cited on page 71.*
- Li, Z., Ravichandran, A., Fowlkes, C., Polito, M., Bhotika, R., and Soatto, S. (2021). Representation Consolidation for Training Expert Students. *arXiv preprint arXiv:2107.08039*. *Cited on page 71.*
- Liu, Y., Zhang, W., and Wang, J. (2020). Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113. *Cited on page 71.*
- Loaiza-Ganem, G. and Cunningham, J. P. (2019). The continuous bernoulli: Fixing a pervasive error in variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 32. *Cited on pages 60, 67, 76, 78, and 79.*
- Mirzadeh, S.-I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. (2020). Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198. *Cited on page 71.*
- Mobahi, H., Farajtabar, M., and Bartlett, P. L. (2020). Self-Distillation Amplifies Regularization in Hilbert Space. In *Advances in Neural Information Processing Systems*. *Cited on pages 60, 61, 63, and 64.*
- Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational Knowledge Distillation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. *Cited on page 71.*
- Parthasarathi, S. H. K. and Strom, N. (2019). Lessons from Building Acoustic Models with a Million Hours of Speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6670–6674. *Cited on page 71.*
- Phuong, M. and Lampert, C. H. (2019). Towards understanding knowledge distillation. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 8993–9007. *Cited on pages 60, 61, and 71.*
- Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*, volume 1. MIT Press, Cambridge, MA. *Cited on pages 62, 63, 75, and 85.*
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). FitNets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. *Cited on page 71.*
- Srinivas, S. and Fleuret, F. (2018). Knowledge transfer with jacobian matching. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 7515–7523. *Cited on page 71.*

- Titsias, M. K. (2009). Variational Learning of Inducing Variables in Sparse Gaussian Processes Michalis. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 567–574. Cited on page 71.
- Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016a). Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 370–378. Cited on page 71.
- Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016b). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, volume 29, pages 2594–2602. Cited on page 71.
- Wolberg William, M. O. S. N. and Street, W. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. Cited on page 86.
- Yang, G. (2019). Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation. *arXiv preprint arXiv:1902.04760*. Cited on page 71.
- You, S., Xu, C., Xu, C., and Tao, D. (2017). Learning from Multiple Teacher Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*, pages 1285–1294. Cited on page 71.
- Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., and Ma, K. (2019). Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3712–3721. Cited on page 71.

Supplementary material

B.I Notation

In the main text, we adopt the notation of [Rasmussen and Williams \(2006\)](#) and use the following. In the expressions for $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$, we have used the notation

$$g(\mathbf{x}, \mathbf{y}^\top) = \left\{ g(x_i, y_j) \right\}_{i,j=1}^{N,M} \in \mathbb{R}^{N \times M}$$

where $g : \mathbb{R}^2 \ni (x, y) \mapsto g(x, y) \in \mathbb{R}$, and $\mathbf{x} = (x_1, \dots, x_N)^\top \in \mathbb{R}^N$, $\mathbf{y} = (y_1, \dots, y_M)^\top \in \mathbb{R}^M$ are column vectors. Notice, that if either argument is a scalar we have

$$\begin{aligned} g(x, \mathbf{x}^\top) &= (g(x, x_1), \dots, g(x, x_N)) \in \mathbb{R}^{1 \times N} \\ g(\mathbf{x}, y) &= (g(x_1, y), \dots, g(x_N, y))^\top \in \mathbb{R}^{N \times 1} \end{aligned}$$

where $g(x, \mathbf{x}^\top)$ is a row-vector and $g(\mathbf{x}, y)$ a column vector.

B.II Additional Details on Self-Distillation for GP Classification

In the following, we restate some known results for GP classification with more details than in the main paper. We also provide some additional details on our self-distillation results for GP classification.

B.II.1 Gaussian Process Classification

Following the usual setup for classification with Gaussian Processes, we assume $f \sim \mathcal{GP}(m, k)$, and for a training set $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, N\}$ we denote $\mathbf{f} = (f_n)_{n=1}^N = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_N))^\top$ and $\mathbf{y} = (y_1, \dots, y_N)^\top$ so that

$$\mathbf{y} \mid \mathbf{f} \sim \prod_{n=1}^N \sigma(f_n)^{y_n} (1 - \sigma(f_n))^{1-y_n}. \quad (\text{B.15})$$

The log posterior has the form

$$\psi(\mathbf{f}) \stackrel{\text{def}}{=} \log p(\mathbf{f} \mid \mathbf{y}) \propto \log p(\mathbf{f}) + \log p(\mathbf{y} \mid \mathbf{f}),$$

where $p(\mathbf{f})$ is the pdf of the multivariate Gaussian distribution and it follows from (B.15) that

$$\log p(\mathbf{y} \mid \mathbf{f}) = \mathbf{y}^\top \mathbf{f} - \sum_{n=1}^N \log(1 + \exp(f_n)).$$

so that (up to the proportionality in \mathbf{f}):

$$\psi(\mathbf{f}) = -\frac{1}{2}(\mathbf{f}_N - \mathbf{m})^\top \mathbf{K}^{-1}(\mathbf{f} - \mathbf{m}) + \mathbf{y}^\top \mathbf{f} - \sum_{n=1}^N \log(1 + \exp(f_n)). \quad (\text{B.16})$$

Since the posterior is non-Gaussian, we apply Laplace approximation to get a Gaussian approximation. Thus, we need the gradient and Hessian to obtain $\hat{\mathbf{f}}$ (the MAP estimate) and the covariance matrix. It follows from (B.16) that

$$\begin{aligned} \nabla \psi(\mathbf{f}) &= \mathbf{y} - \sigma - \mathbf{K}^{-1} \mathbf{f} + \mathbf{K}^{-1} \mathbf{m} \\ -\nabla \nabla \psi(\mathbf{f}) &= \mathbf{W} + \mathbf{K}^{-1} \end{aligned}$$

where $(\sigma)_n = \sigma(f_n)$ (this term comes from $\sum_{n=1}^N \log(1+e^{f_n})$) and $\mathbf{W} = \text{diag}_n\{\sigma(f_n)(1-\sigma(f_n))\}$.

Using these can use the Newton-Raphson procedure to find $\hat{\mathbf{f}}$, and it follows from the expression for the gradient, that $\hat{\mathbf{f}}$ will be the solution to

$$\hat{\mathbf{f}} = \mathbf{m} + \mathbf{K}(\mathbf{y} - \sigma). \quad (\text{B.17})$$

The Gaussian approximation of $p(\mathbf{f} | \mathbf{y})$ comes out to

$$q(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \hat{\mathbf{f}}, (\mathbf{W} + \mathbf{K}^{-1})^{-1}).$$

The posterior predictive distribution is computed as

$$p(f_* | \mathbf{x}, \mathbf{y}, \mathbf{x}_*) = \int p(f_* | \mathbf{f})p(\mathbf{f} | \mathbf{y})d\mathbf{f} \quad (\text{B.18})$$

(see [Bishop \(2006\)](#)) where the $\mathcal{GP}(m, k)$ prior gives the conditional distribution

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* | m(x) + \mathbf{k}^T \mathbf{K}^{-1}(\mathbf{f} - \mathbf{m}), k - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}). \quad (\text{B.19})$$

We can obtain predictions in two different ways: Either by substituting $q(\mathbf{f})$ for $p(\mathbf{f} | \mathbf{y})$ in (B.19) (this leads to (B.1) and (B.2)), or simply as the predictions of the mean

$$\sigma(\mathbb{E}_q[\mathbf{f} | \mathbf{y}]) = \sigma(\hat{\mathbf{f}}).$$

We note that while the two are different in general, the decision boundary is identical for binary classification ([Bishop, 2006](#), Sec. 10.3).

B.II.2 Deterministic Self-distillation for GP Classification

For a single step of deterministic distillation, we can choose either of the above predictions as our new targets and will denote them as \mathbf{y}_1 irrespective of the choice. However, while $\mathbf{y} \in \{0, 1\}^N$ we now have $\mathbf{y}_1 \in [0, 1]^N$, and we can no longer assume a conditional Bernoulli distribution over \mathbf{y}_1 . However, by extending to the continuous Bernoulli distribution ([Loaiza-Ganem and Cunningham, 2019](#)), we can avoid this problem, by introducing the appropriate normalizing constant

$$C(\lambda) \stackrel{\text{def}}{=} \begin{cases} 2 & \text{if } \lambda = \frac{1}{2}, \\ \frac{2 \tanh^{-1}(1-2\lambda)}{1-2\lambda} & \text{otherwise.} \end{cases} \quad (\text{B.20})$$

See Figure B.14 for plots of the normalizing constant, its gradient, and Hessian. We now assume $f_1 \sim \mathcal{GP}(0, k)$ and

$$\mathbf{y}_1 | \mathbf{f}_1 \sim \prod_{n=1}^N C(\sigma(f_{1,n})) \sigma(f_{1,n})^{y_{1,n}} (1 - \sigma(f_{1,n}))^{1-y_{1,n}},$$

which in turn yields that

$$\log p(\mathbf{y}_1 | \mathbf{f}_1) = \mathbf{y}_1^T \mathbf{f}_1 - \sum_{n=1}^N \log(1 + \exp(f_{1,n})) + \sum_{n=1}^N \log C(\sigma(f_{1,n})),$$

where we note that $\sum_{n=1}^N \log C(\sigma(f_{1,n})) > 0$. In Proposition B.5 we find the derivative and second derivative of $\log C(\sigma(a))$. Thus, if we denote $\mathbf{C}_1 \stackrel{\text{def}}{=} \sum_{n=1}^N \log(C(\sigma(f_{1,n})))$, we get that the gradient of the conditional density is

$$\nabla \psi_1(\mathbf{f}_1) = \mathbf{y}_1 - \sigma_1 - \mathbf{K}^{-1} \mathbf{f}_1 + \nabla \mathbf{C}_1,$$

where we can compute the last term by use of the derivative above. Similarly, it follows that the Hessian can be computed as

$$-\nabla \nabla \psi_1(\mathbf{f}_1) = \mathbf{W}_1 + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_1,$$

where we note that $\nabla \nabla \mathbf{C}_1$ is a diagonal matrix, and that σ_1 and \mathbf{W}_1 are defined analogously to the classical case. With the gradient and Hessian, we can follow the usual setup and compute the MAP estimate for the mean of the Laplace approximation, and use the inverse Hessian as covariance. That is, we use

$$\begin{aligned} \mathbf{f}_1^{\text{new}} &= \mathbf{f}_1 - (-\mathbf{W}_1 - \mathbf{K}^{-1} + \nabla \nabla \mathbf{C}_1)^{-1} (\mathbf{y}_1 - \sigma_1 - \mathbf{K}^{-1} \mathbf{f}_1 + \nabla \mathbf{C}_1) \\ &= (\mathbf{W}_1 + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_1)^{-1} ((\mathbf{W}_1 + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_1) \mathbf{f}_1 + \mathbf{y}_1 - \sigma_1 - \mathbf{K}^{-1} \mathbf{f}_1 + \nabla \mathbf{C}_1) \\ &= \mathbf{K} \mathbf{K}^{-1} (\mathbf{W}_1 + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_1)^{-1} ((\mathbf{W}_1 - \nabla \nabla \mathbf{C}_1) \mathbf{f}_1 + \mathbf{y}_1 - \sigma_1 + \nabla \mathbf{C}_1) \\ &= \mathbf{K} ((\mathbf{W}_1 - \nabla \nabla \mathbf{C}_1) \mathbf{K} + \mathbf{I})^{-1} ((\mathbf{W}_1 - \nabla \nabla \mathbf{C}_1) \mathbf{f}_1 + \mathbf{y}_1 - \sigma_1 + \nabla \mathbf{C}_1) \end{aligned}$$

That is, we now have the Laplace approximation

$$p(\mathbf{f}_1 | \mathbf{y}) \approx q(\mathbf{f}_1 | \mathbf{y}) = \mathcal{N}(\mathbf{f}_1 | \hat{\mathbf{f}}_1, (\mathbf{W}_1 + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_1)^{-1}),$$

where $\hat{\mathbf{f}}_1$ is the MAP estimate from above, and we can also use the Laplace approximation to get the marginal log-likelihood. First by a Taylor approximation of $\psi_1(\mathbf{f}_1)$ at $\hat{\mathbf{f}}_1$, we get that $\psi_1(\mathbf{f}_1) \approx \psi_1(\hat{\mathbf{f}}_1) - \frac{1}{2} (\mathbf{f}_1 - \hat{\mathbf{f}}_1)^\top \mathbf{H}^{-1} (\mathbf{f}_1 - \hat{\mathbf{f}}_1)$, where \mathbf{H}^{-1} is the covariance matrix of q , i.e. $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{W}_1 + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_1$. Thus, we have that

$$\begin{aligned} \log p(\mathbf{y}_1) &= \int \psi(\mathbf{f}_1) d\mathbf{f}_1 \\ &\approx \psi_1(\hat{\mathbf{f}}_1) + \int -\frac{1}{2} (\mathbf{f}_1 - \hat{\mathbf{f}}_1)^\top \mathbf{H}^{-1} (\mathbf{f}_1 - \hat{\mathbf{f}}_1) d\mathbf{f}_1 \\ &= \psi_1(\hat{\mathbf{f}}_1) + \frac{N}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{H}| \\ &= \mathbf{y}_1^\top \hat{\mathbf{f}}_1 - \sum_{n=1}^N \log(1 + \exp(\hat{f}_{1,n})) + \sum_{n=1}^N \log C(\sigma(\hat{f}_{1,n})) - \frac{1}{2} \hat{\mathbf{f}}_1^\top \mathbf{K}^{-1} \hat{\mathbf{f}}_1 \\ &\quad - \frac{1}{2} \log |\mathbf{K}| - \frac{N}{2} \log(2\pi) + \frac{N}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{H}| \\ &= \mathbf{y}_1^\top \hat{\mathbf{f}}_1 - \sum_{n=1}^N \log(1 + \exp(\hat{f}_{1,n})) + \sum_{n=1}^N \log C(\sigma(\hat{f}_{1,n})) \\ &\quad - \frac{1}{2} \hat{\mathbf{f}}_1^\top \mathbf{K}^{-1} \hat{\mathbf{f}}_1 - \frac{1}{2} \log(|\mathbf{K}|) - \frac{1}{2} \log(|\mathbf{W}_1 + \mathbf{K}^{-1} - \nabla \nabla \mathbf{C}_1|) \end{aligned}$$

The marginal log-likelihood is useful for hyperparameter optimization.

We can then iterate this distillation procedure any number of times. Usually, we consider the inversion of \mathbf{K} the most computationally expensive step of fitting a GP, and since we can re-use \mathbf{K}^{-1} for all our steps, the addition of distillation steps is, computationally, not very demanding. In the above MAP estimation, we even rewrite the optimization to avoid the use of the inverse of \mathbf{K} .

B.III Setup for Illustratory Examples and Additional Results

Throughout the main paper, we have included illustrations and plots based on various illustrator examples. However, all these illustratory examples have been computed with a radial basis kernel function

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2l}\right),$$

where $\sigma_f^2 > 0$ and $l > 0$ are both hyperparameters tunable for the particular example. We also (mainly for the regression setup) consider a noise parameter $\gamma \geq 0$ which is added to the diagonal of \mathbf{K} , i.e. assume noise on the training observations. Even without this assumption we typically use a small $\gamma \approx 10^{-8}$ to avoid numerical instability both in regression and classification examples.

In Figure B.7 we plot the negative log-likelihood over a grid of values for σ_f and l both in the case of using the ordinary GP classification with Bernoulli likelihood and the adjusted setting with continuous Bernoulli likelihood when the input is the true continuous underlying function $g(x) = 2 \sin(x\pi/2)$.

B.III.1 Additional Experiments and Results

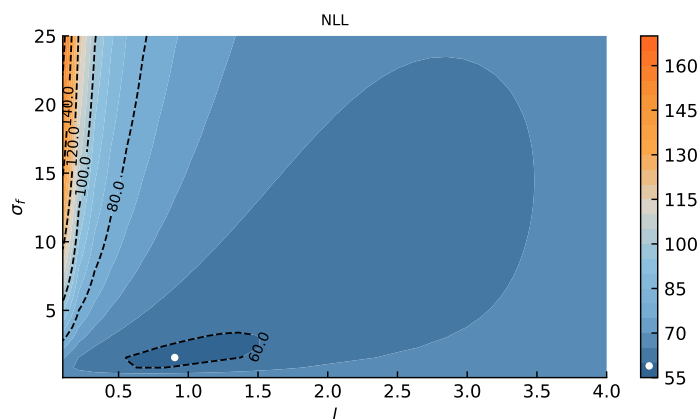
In the following, we present additional results on the same illustratory experiments as in the main paper, but with changes to e.g. the model parameters or data assumptions. In particular, in Figure B.8 we present different solutions to the deterministic GPR example in the main text when varying the choices of $(\gamma_1, \dots, \gamma_{10})$. Similarly, in Figure B.9 we repeat the same experiments for probabilistic GPR. Finally, in Figure B.10 we plot the solutions of deterministic GPC, when we fit the distilled model, not to the continuous predictions of the initial model, but to the $(0, 1)$ -encoded targets obtained by thresholding the initial model at 0.5. In this setting, using an ordinary Bernoulli assumption does not result in a misspecified model, but the thresholded neglects some of the information contained in the continuous targets, and as evident from Figure B.10, the distilled models, in this case, is largely similar to that of a C-GPC on the continuous targets.

B.IV Some Details on the Continuous Bernoulli Distribution

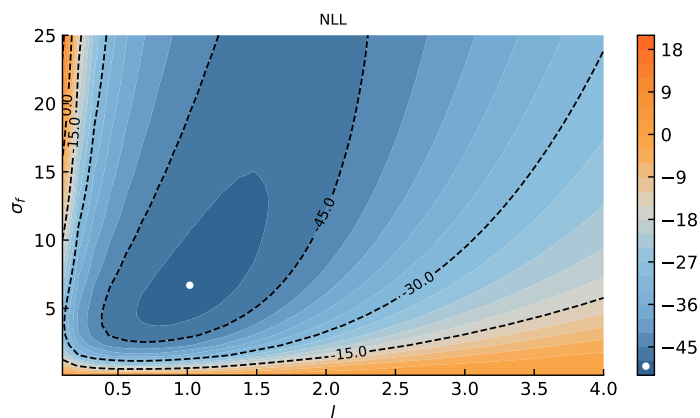
The continuous Bernoulli distribution (denoted by $\mathcal{CB}(\lambda)$) was introduced in [Loaiza-Ganem and Cunningham \(2019\)](#) and dealt with the relatively common case, where data is continuous values in $[0, 1]$ rather than discrete values in $\{0, 1\}$. When performing distillation this is indeed the case, and merely using a (implicit or explicit) Bernoulli assumption is not suitable. For good measure, we repeat some key results of \mathcal{CB} from [Loaiza-Ganem and Cunningham \(2019\)](#) in the following section. Note, in the following, we will treat Bernoulli variables as if they could attain continuous values in $[0, 1]$, which is the incorrect approach currently often used in practice.

First, we recall that the density of the Bernoulli distribution with parameter $\lambda \in (0, 1)$ is given by

$$X \sim \mathcal{B}(\lambda) \iff \tilde{p}(x | \lambda) = \lambda^x (1 - \lambda)^{1-x}.$$



(a) Ordinary GPC



(b) C-GPC

Figure B.7: Grid search of hyperparameters of two different types of GP classification. We indicate the minimum of each grid with a white dot. Note, the absolute values can not be compared between the two models due to different likelihood functions.

Second, by [Loaiza-Ganem and Cunningham \(2019\)](#) the density of CB differs from the density of \mathcal{B} merely by a multiplicative normalizing constant (dependent on the parameter λ), denoted by $C(\lambda)$. See Figure B.11 for a plot of both densities. In particular, we have that for $\lambda \in (0, 1)$ then

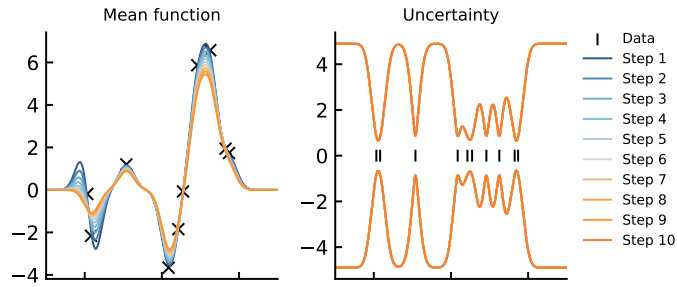
$$X \sim CB(\lambda) \iff p(x | \lambda) = C(\lambda)\lambda^x(1 - \lambda)^{1-x}$$

where the normalizing constant can be shown to be

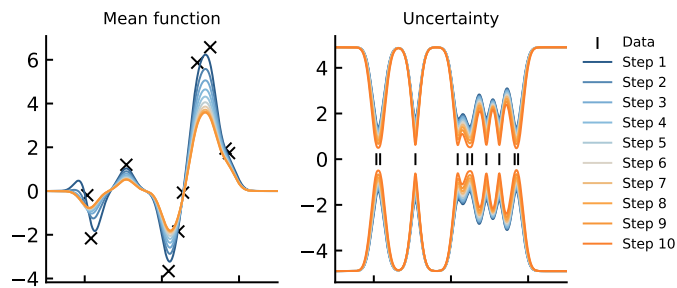
$$C(\lambda) \stackrel{\text{def}}{=} \begin{cases} 2 & \text{if } \lambda = \frac{1}{2}, \\ \frac{2 \tanh^{-1}(1-2\lambda)}{1-2\lambda} & \text{otherwise,} \end{cases}$$

and \tanh^{-1} is the inverse hyperbolic tangent function.

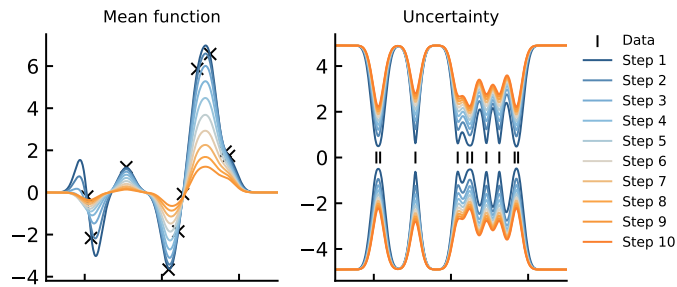
In Figure B.13 we plot the normalizing constant for both the ordinary (equal to 1 for all λ) and continuous Bernoulli distribution, and we also observe that since the mean of a $CB(\lambda)$ distributed variable is not merely λ , the optimal λ (i.e. maximizing the density) for a single observation x is different from λ .



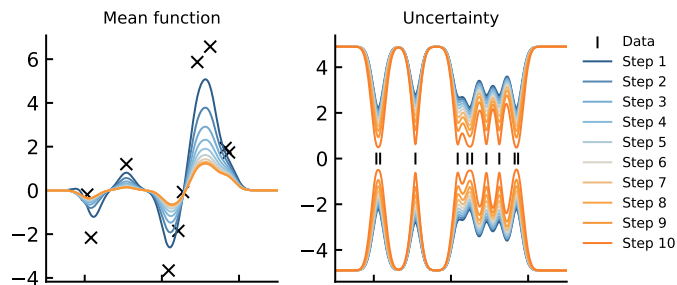
(a) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (0.2, 0.2, \dots, 0.2)$



(b) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (1, 0.9, \dots, 0.1)$



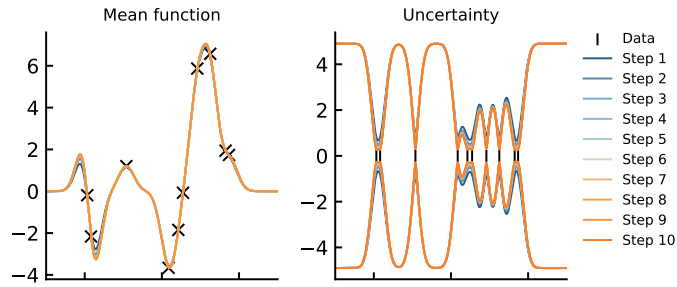
(c) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (0.1, 0.42, \dots, 3)$



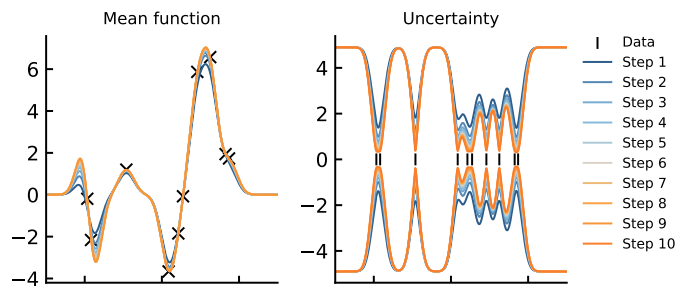
(d) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (3, 2.68, \dots, 0.1)$

Figure B.8: Ten steps of d -GPSD for GPR with fixed hyperparameters, but with varying choices of noise parameters. We note the progressively increasing regularization present in all examples.

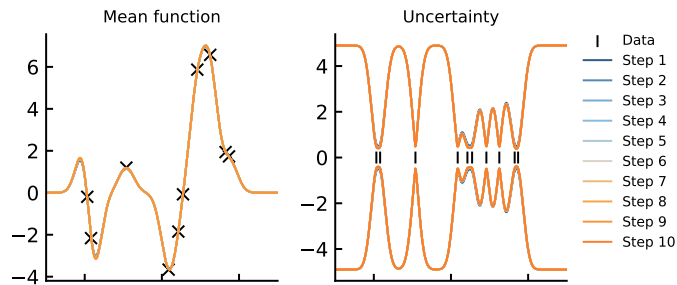
B.IV · Some Details on the Continuous Bernoulli Distribution



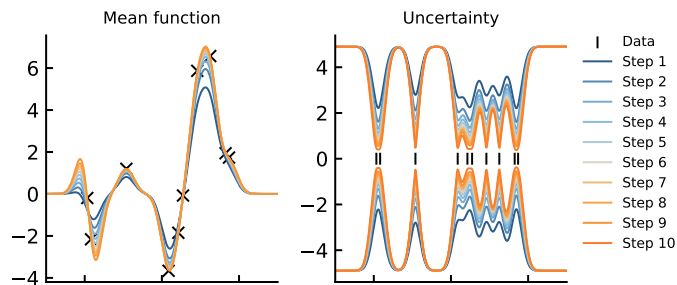
(a) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (0.2, 0.2, \dots, 0.2)$



(b) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (1, 0.9, \dots, 0.1)$



(c) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (0.1, 0.42, \dots, 3)$



(d) Noise parameters: $(\gamma_1, \dots, \gamma_{10}) = (3, 2.68, \dots, 0.1)$

Figure B.9: Ten steps of ρ -GPSD for GPR with fixed hyperparameters, but with varying choices of noise parameters. We note that while (c) and (d) yield the same solution at step 10, the path to that solution varies.

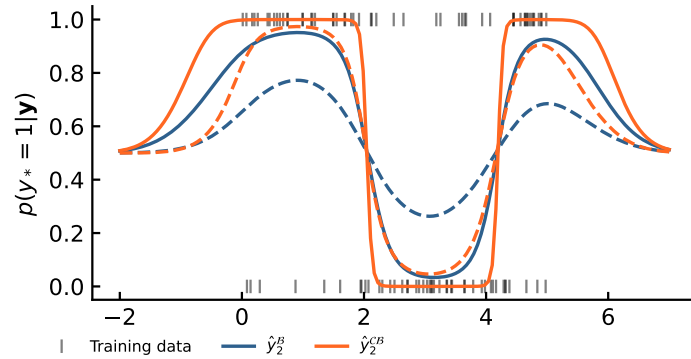


Figure B.10: Same setup as in Figure B.4, but instead of fitting the distilled model to the soft output of the first, we also include (in dashed) the ordinary GPC and C-GPC when fitted to the binary predictions (thresholded at 0.5). We see that the overfitting of the C-GPC is amplified, while the ordinary GPC closely resembles the initial C-GPC on soft targets.

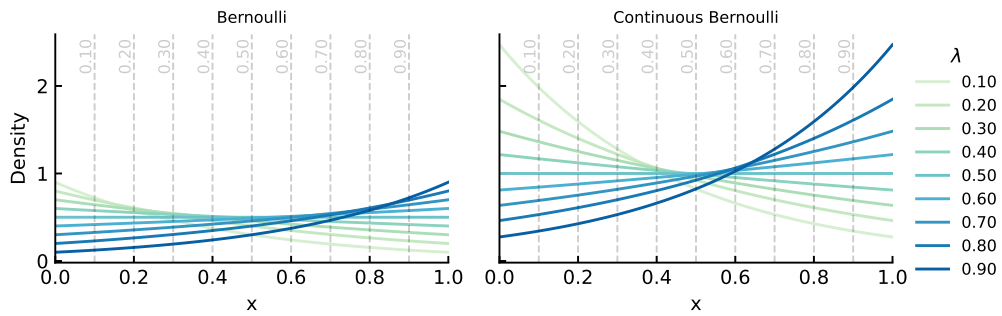


Figure B.11: Density of Bernoulli and continuous Bernoulli distributions with different choices of parameter λ .

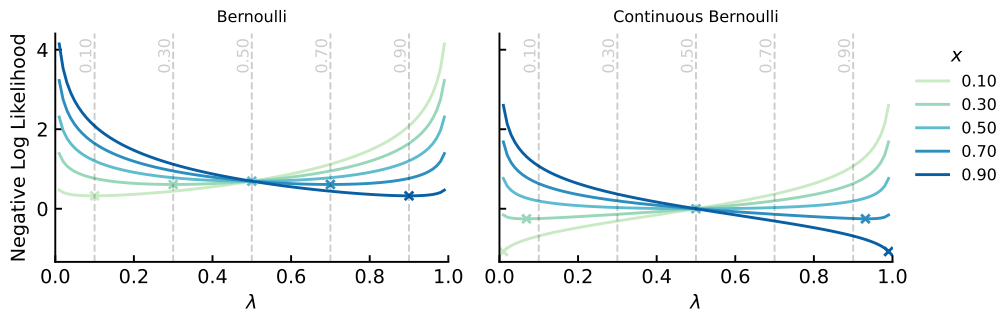


Figure B.12: Negative log-likelihood of Bernoulli and continuous Bernoulli distributions with different choices of parameter λ . We illustrate the optimizing λ for each x by a cross \times , and note that for a given observation x the optimal choice of λ for the Bernoulli is x , while it for the continuous Bernoulli is a non-linear function of x , which is generally biased towards more extreme values of λ . See also Figure B.13.

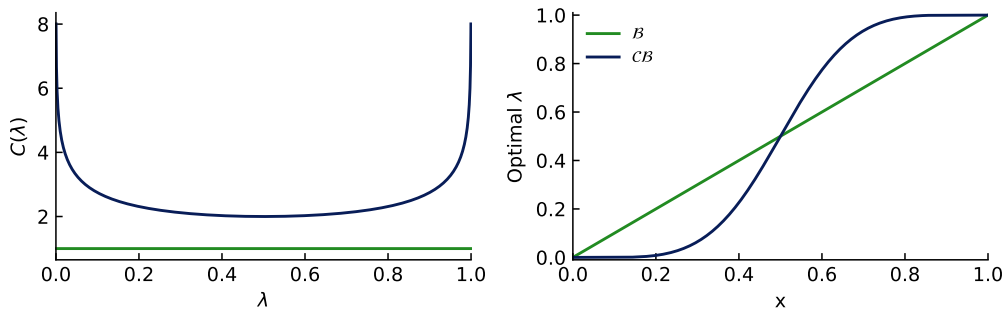


Figure B.13: Normalizing constant and optimal λ values (i.e. maximizing likelihood for a single observation of x) for both ordinary and continuous Bernoulli distributions.

Finally, in Figure B.14 we both plot the log-normalizing constant, as well as the first and second derivatives with respect to a versus either $\lambda = \sigma(a)$ or a . See Proposition B.5 for the derivations of these.

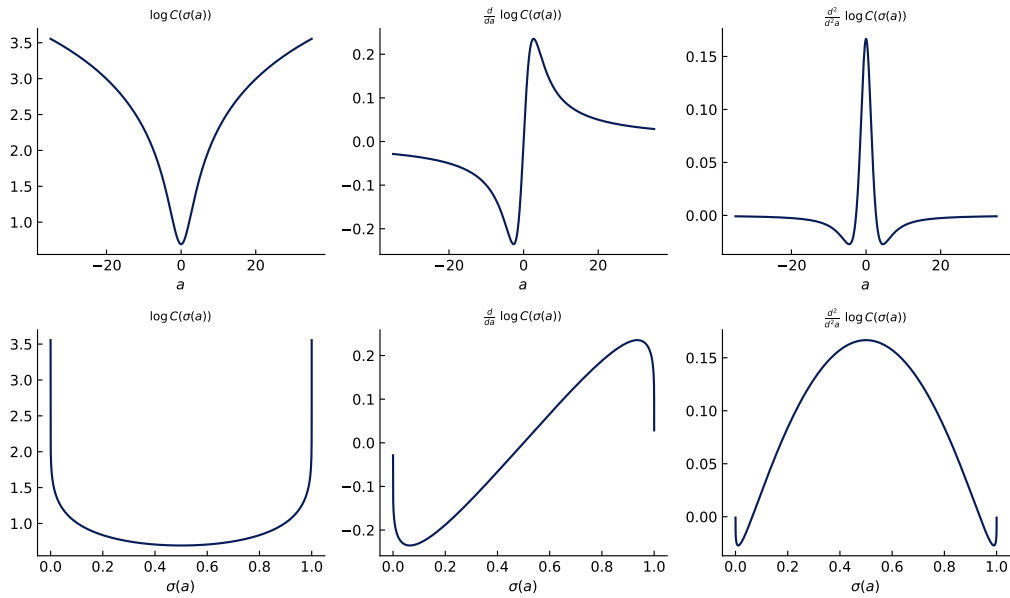


Figure B.14: Log-normalizing constant for the continuous Bernoulli distribution $C(\lambda)$, along with first and second derivatives with respect to a .

B.V Implementation and Time/Compute Analysis

Despite the main aim of our proposed self-distillation methods being of a theoretical nature, in the following, we investigate the compute requirements of our proposed methods.

B.V.1 (Efficient) Implementation

We provide a public implementation of our methods in Python on https://github.com/Kennethborup/gaussian_process_self_distillation. We provide four classes; deterministic and probabilistic self-distillation for both GP regression and GP classification. The classes are named straightforwardly as `DataCentricGPR`, `DistributionCentricGPR`, `DataCentricGPC`, and `DistributionCentricGPC`.¹ Each method has `.fit`, and `.predict` methods that are similar to the `scikit-learn` API. For `DataCentricGPR` we provide both a naïve and efficient implementation, which significantly improves the training speed for multiple steps of distillation. In particular, inspired by [Borup and Andersen \(2021\)](#) we utilize the singular values decomposition of \mathbf{K} as \mathbf{VDV}^\top (here $\mathbf{V} \in \mathbb{R}^{N \times N}$ is an orthogonal matrix with the eigenvectors of \mathbf{K} as rows and $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a non-negative diagonal matrix with the associated eigenvalues in the diagonal), to rewrite (B.3) for $t = 1$ into

$$\mathbf{y}_1 = \mathbf{VD}(\mathbf{D} + \gamma_1 \mathbf{I}_N)^{-1} \mathbf{V}^\top \mathbf{y}$$

which by iterative inserting yields the more general expression

$$\begin{aligned} \mathbf{y}_t &= \left(\prod_{s=1}^t \mathbf{K}(\mathbf{K} + \gamma_s \mathbf{I}_N)^{-1} \right) \mathbf{y}, \\ &= \mathbf{V} \left(\prod_{s=1}^t \mathbf{A}_s \right) \mathbf{V}^\top \mathbf{y}, \end{aligned}$$

where $\mathbf{A}_s = \mathbf{D}(\mathbf{D} + \gamma_s \mathbf{I}_N)^{-1} = \text{diag}_{n=1, \dots, N} \left(\frac{d_n}{d_n + \gamma_s} \right)$ for $s = 1, \dots, t$ are diagonal matrices. Thus, when \mathbf{V} and \mathbf{D} are computed, the subsequent distillation steps are computationally cheap due to the diagonal structure of the \mathbf{A}_s 's. One can similarly rewrite the prediction formula (B.3) — see the source code for more implementation details.

B.V.2 Time Requirements

To evaluate the practical implications of our proposed methods, we compared the time it takes to fit our methods (for a varying number of distillation steps) with an ordinary fit using the standard implementation of GP regression and GP classification in `scikit-learn`; namely the `GaussianProcessRegressor` and `GaussianProcessClassifier`. In particular, we fit a model with each of our methods for a particular number of steps and divide the fitting time with the fitting time of the corresponding ordinary `scikit-learn` method. We repeat our experiments 30 times and report the mean and 10 and 90 empirical percentiles in Figure B.15. All experiments are performed on a Mac M1 Pro CPU.

For regression, we observe that probabilistic self-distillation (as expected from Theorem B.3) requires no more computation than ordinary GPR with the `scikit-learn` implementation of GP regression. This is simply due to probabilistic self-distillation for regression stays within the GP setting. For deterministic self-distillation the story is different; the time complexity of a naïve iterative implementation where each successive model is fitted to the predictions of the preceding model scales linearly with the number

¹ The naming follows an earlier version of the paper, where data-centric refers to the deterministic methods, and distribution-centric refers to the probabilistic methods.

of distillation steps (at a rate of ≈ 0.11). However, by utilization of an eigendecomposition of \mathbf{K} in (B.3), one can rewrite the solution to optimize the fitting speed significantly. In particular, each additional step of distillation merely requires updating a diagonal matrix and a few matrix products (of re-used matrices), which is much cheaper. Thus, with this efficient implementation, we are able to fit deterministic self-distillation to any number of distillation steps at a constant time complexity. Thus, both deterministic and probabilistic self-distillation for regression does not take longer to fit than an ordinary GP regression from `scikit-learn`.

For classification, we observe that deterministic self-distillation scales linearly with the number of distillation steps at a rate of ≈ 0.26 , and multiple steps of self-distillation in this setting are quickly computationally costly. However, the time to fit with probabilistic self-distillation is (as expected) constant with the number of distillation steps. This is clearly expectable from Proposition B.6, and fits as quickly as the `scikit-learn` implementation.

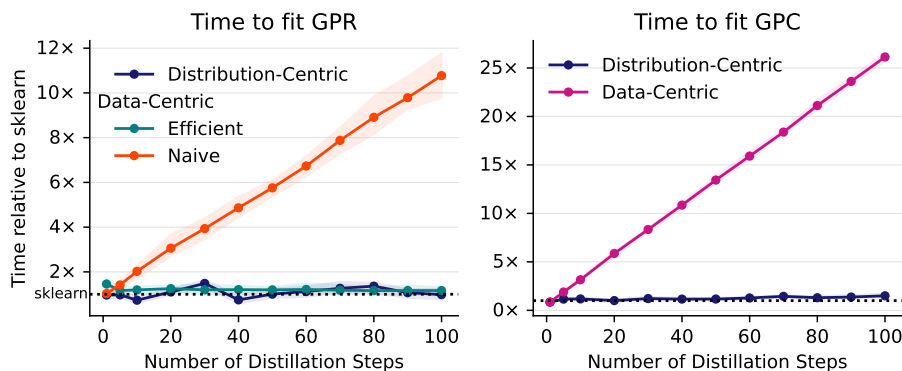


Figure B.15: Relative time to fit various self-distilled GP models compared to a single fit with the standard `scikit-learn` implementation of the regression and classification model respectively. On the (left) we plot the relative time to fit regression models. For deterministic self-distillation, we use both a naïve and efficient approach. While the naïve approach scales linearly with the distillation steps, the efficient approach is constant in relative time for all distillation steps. On the (right) we plot the relative time for the classification models compared to the `scikit-learn` implementation and observe that the deterministic approach scales linearly with distillation steps, while (as expected) the probabilistic approach is constant with distillation steps. The shaded regions represent the 10 and 90 empirical quantiles of 30 repetitions for each method and the number of distillation steps. We note that these experiments are performed with hyperparameters optimized at the first step only.

B.V.3 Computational Requirements

In the following, we consider the computational requirements of our methods. We will consider the inversion of matrices, matrix products, Cholesky decomposition, and SVD to all be $\mathcal{O}(N^3)$ ignoring constants, although some algorithms exist to reduce the order of computation for different methods to $\mathcal{O}(N^\omega)$ where typically $2 < \omega \leq 3$. Our implementation is based on Algorithm 3.1, 3.2, and 5.1 in [Rasmussen and Williams \(2006\)](#), which largely depends on using the Cholesky decomposition for back-substitution.

We note that while the framing of our setup differs notably from the ordinary GP regression and GP classification setups, our proposed methods still stay close to or within

the ordinary GP framework simplifying the analysis. In particular, for probabilistic GPR, the solution is an ordinary GPR model with a particularly scaled noise parameter (see Theorem B.3), and any existing GPR implementation can be used. Thus, the complexity of this method is $\mathcal{O}(N^3)$ as usual. For probabilistic GP classification, we consider the approximate method from Proposition B.6, and note, that this too is equivalent to an ordinary GPC model, and is in $\mathcal{O}(N^3)$. For deterministic GP regression, a naive implementation would require t iterative fits of an ordinary GPR model, but utilizing the efficient implementation above, we replace the Cholesky decomposition with the SVD, and can thus reuse \mathbf{V} , and \mathbf{D} for multiple steps of distillation. Thus, each additional distillation step merely requires matrix products of diagonal matrices and with the $N \times N$ matrices, which makes this method $\mathcal{O}(N^3)$ as well. Finally, due to the iterative nature of deterministic GP classification, which does not allow a simplified solution, this method, unfortunately, scales linearly with the number of distillation steps (although still in $\mathcal{O}(N^3)$). In particular, for $t > 1$ we change the likelihood function to the continuous Bernoulli, which is a negligent change due to the simple and closed-form solutions provided in Proposition B.5, but since we do not have a closed form solution for our model, we need to fit each model in an ordinary fashion, thus requiring $\mathcal{O}(tN^3)$ with t the number of distillation steps. This can be computationally demanding for large t . Generally, we observe that, despite deterministic GP classification scaling linearly with t , all our proposed methods are still primarily limited by the scaling of N . Finally, due to the minor changes in our methods, methods to e.g. efficiently compute/approximate \mathbf{K} should largely still be compatible with our methods.

B.VI Experiments on Realistic Data

To supplement our theoretical analysis we provide results on empirical data in Table B.1 for both regression and classification tasks. We use the implementation presented in Section B.V, and provide results for ordinary GP models and both ρ -GPSD and d -GPSD for comparison.

We investigate three regression tasks and three classification tasks. In particular, for regression, we analyze the Diabetes dataset (Efron et al., 2004), the California Housing dataset², and the Linnerud dataset (we only model the second output dimension)³. For classification, we need to adapt any task as a binary task per the restrictions of the implementation. Thus, we consider the Breast Cancer dataset (Wolberg William and Street, 1995), a binary version of the Cover Type dataset (only compare classes 5 and 6) (Blackard, 1998), and the Credit-G dataset (Hofmann, 1994). To keep the computational requirements at a reasonable scale to perform a search of a large grid of hyperparameters, we subsample the California Housing and Cover Type datasets to 1000 samples. We apply a 5-fold cross-validated grid-search over the hyperparameters; noise parameter (for regression only), number of distillation steps, and kernel function. We report the predictive performance as the average RMSE or accuracy on the 5 validation sets for regression and classification, respectively. For all methods (ordinary, ρ -GPSD, and d -GPSD) we optimize the kernel parameters by minimizing the negative log-likelihood on

² https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

³ https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_linnerud.html

	Diabetes (\downarrow)	Cal. Housing (\downarrow)	Linnerud (\downarrow)
Ordinary	54.095	0.430	2.733
Single-optimization			
ρ -GPSD	54.095	0.430	2.733
d -GPSD	54.492	0.430	2.522
Multi-optimization			
d -GPSD	53.677	0.430	2.709

(a) RMSE for GPR on three different datasets.

	Breast Cancer (\uparrow)	Cover Type (\uparrow)	Credit-G (\uparrow)
Ordinary	0.975	0.991	0.763
Single-optimization			
ρ -GPSD	0.975	0.991	0.763
d -GPSD	0.975	0.991	0.763

(b) Accuracy for GPC on three different datasets.

Table B.1: Predictive performance through 5-fold cross-validation on 3 different tasks for both regression and classification. For GPR we include both ordinary GPR, and ρ -GPSD and d -GPSD with hyperparameter optimization at the first step, as well as d -GPSD with hyperparameter optimization at each distillation step. For GPC, we include ordinary GPC as well as ρ -GPSD and d -GPSD with hyperparameter optimization at the first step only.

the original data at the first fit, while we for ρ -GPSD and d -GPSD reuse these parameters for all distillation steps in the *Single-optimization* rows of the table, and refit these kernel parameters (by minimizing the NLL on the previous predictions) at each distillation step for d -GPSD on regression in the *Multi-optimization* row. Multi-optimization solutions tend to perform slightly better than single-optimization but do not warrant the increased computational requirements. Due to the limitations of our implementation, we only provide multi-optimization results for d -GPSD.

As expected from the theoretical results, ρ -GPSD is equivalent to an ordinary fit, where the constant kernel parameter is scaled by 1 over the number of distillation steps, as long as we search over an equivalent set of noise parameters. Hence, when we optimize the kernel parameters on the NLL at the initial fit, we expect them to converge to an equivalent solution. We also observe that for classification, self-distillation does not improve the predictive performance for d -GPSD. However, for the regression tasks, we observe that while d -GPSD with re-used hyperparameters (from the first fit) is inferior to both ordinary GPR and ρ -GPSD on the Diabetes dataset, it is superior on Linnerud and equivalent on California Housing. Furthermore, we observe that while d -GPSD with hyperparameters updated at each step is superior to all other methods on Diabetes, and to GPR and ρ -GPSD on Linnerud, it is still outperformed by a single-step optimized d -GPSD on Linnerud. We stress that, while self-distillation for GP models appears to provide some possibilities for empirical improvements, it is not the aim of this paper to provide a superior empirical method, but rather to provide a theoretical framework on how to analyze self-distillation in a mathematically tractable setting of GP models.

B.VII Proofs From the Main Paper

In the following, we restate the results of the main paper and provide the associated proofs.

Proposition B.1 (from page 63). *Define $\mathbf{y}_t \stackrel{\text{def}}{=} \boldsymbol{\mu}_t$ and $\mathbf{y}_0 \stackrel{\text{def}}{=} \mathbf{y}$. Then it holds for any $t \geq 1$ that $\mathbf{f}_t | \boldsymbol{\mu}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, where*

$$\mathbf{y}_t = \left(\prod_{s=1}^t \mathbf{K}(\mathbf{K} + \gamma_s \mathbf{I}_N)^{-1} \right) \mathbf{y}, \quad (\text{B.3})$$

$$\mathbb{E}[\mathbf{f}_{t,*} | \boldsymbol{\mu}_{t-1}] = k(\mathbf{x}_*, \mathbf{x})(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{y}_{t-1}, \quad (\text{B.21})$$

and $\boldsymbol{\Sigma}_t = \mathbf{K} - \mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{K}$ as well as \mathbf{y}_t is only affected by the choice of γ_t .

Proof. We assume the model $y_{t-1} = f_t(\mathbf{x}) + \varepsilon_t$, with $f_t \sim \mathcal{GP}(0, k)$, and $\varepsilon_t \sim \mathcal{N}(0, \gamma_t)$ for all $t \geq 1$. This yields

$$\begin{aligned} p(\mathbf{f}_t) &= \mathcal{N}(\mathbf{f}_t | 0, \mathbf{K}) \\ p(\mathbf{y}_{t-1} | \mathbf{f}_t) &= \mathcal{N}(\mathbf{y}_{t-1} | \mathbf{f}_t, \gamma_t \mathbf{I}_N) \\ p(\mathbf{y}_{t-1}) &= \int p(\mathbf{y}_{t-1} | \mathbf{f}_t) p(\mathbf{f}_t) d\mathbf{f}_t = \mathcal{N}(\mathbf{y}_{t-1} | 0, \mathbf{K} + \gamma_t \mathbf{I}_N), \end{aligned}$$

where we use the usual bolded notation for functions evaluated in $\mathbf{x}_1, \dots, \mathbf{x}_N$; i.e. $\mathbf{y}_t = (y_t(\mathbf{x}_1), \dots, y_t(\mathbf{x}_N))^T \in \mathbb{R}^N$ and $\mathbf{f}_t = (f_t(\mathbf{x}_1), \dots, f_t(\mathbf{x}_N))^T \in \mathbb{R}^N$. Which combined yields the prior

$$\begin{bmatrix} \mathbf{y}_{t-1} \\ \mathbf{f}_t \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \gamma_t \mathbf{I} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{bmatrix} \right).$$

Thus, following the standard results for conditional multivariate Gaussian distributions, we get the posterior

$$\mathbf{f}_t | \mathbf{y}_{t-1} \sim \mathcal{N} \left(\mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{y}_{t-1}, \mathbf{K} - \mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{K} \right). \quad (\text{B.22})$$

Recall that we use $\mathbf{y}_t \stackrel{\text{def}}{=} \boldsymbol{\mu}_t = \mathbb{E}[\mathbf{f}_t | \mathbf{y}_{t-1}]$, and by iteratively inserting $\mathbf{y}_{t-1} = \mathbf{K}(\mathbf{K} + \gamma_{t-1} \mathbf{I}_N)^{-1} \mathbf{y}_{t-2}$ into (B.22), we get

$$\begin{aligned} \mathbf{y}_t = \mathbb{E}[\mathbf{f}_t | \mathbf{y}_{t-1}] &= \mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{y}_{t-1} \\ &= \mathbf{K}(\mathbf{K} + \gamma_t \mathbf{I}_N)^{-1} \mathbf{K}(\mathbf{K} + \gamma_{t-1} \mathbf{I}_N)^{-1} \mathbf{y}_{t-2} \\ &= \mathbf{A}_t \mathbf{A}_{t-1} \mathbf{y}_{t-2} \\ &= \left(\prod_{m=1}^t \mathbf{A}_m \right) \mathbf{y}_0, \end{aligned}$$

where we use the notation $\mathbf{A}_i = \mathbf{K}(\mathbf{K} + \gamma_i \mathbf{I}_N)^{-1}$. Note, the claim for $\mathbb{E}[f_t(x)]$ follows directly from analogous results for arbitrary \mathbf{x} . \square

Lemma B.2 (from page 65). *Let $\lambda_i^{(t)}$, for $i = 1, \dots, N$, denote the eigenvalues of \mathbf{K}_t . Write $\mathbf{K}_0 = \mathbf{O} \text{diag}_i(\lambda_i^{(0)}) \mathbf{O}^T$ for the spectral decomposition of \mathbf{K}_0 . Then we have for $t = 0, 1, \dots$ that*

$$\mathbf{K}_t = \mathbf{O} \text{diag}_i \left(\lambda_i^{(0)} \zeta_i^{(t)} \right) \mathbf{O}^T \quad (\text{B.7})$$

where $\zeta_i^{(t)} \stackrel{\text{def}}{=} \frac{1}{\lambda_i^{(0)} \gamma_{i-1}^- + 1}$. For $k_t(x, \mathbf{x}^\top)$ and $m_t(\mathbf{x})$ we have

$$k_t(x, \mathbf{x}^\top) = k_0(x, \mathbf{x}^\top) \mathbf{O} \text{diag}_i \left(\zeta_i^{(t)} \right) \mathbf{O}^\top \quad (\text{B.8})$$

$$m_t(\mathbf{x}) = \mathbf{O} \text{diag}_i \left(\lambda_i^{(0)} \gamma_{i-1}^- \zeta_i^{(t)} \right) \mathbf{O}^\top \mathbf{y}. \quad (\text{B.9})$$

Proof. First we observe that (B.6) gives the following recursion for \mathbf{K}_t :

$$\mathbf{K}_{t+1} = \mathbf{K}_t - \mathbf{K}_t (\mathbf{K}_t + \gamma_t \mathbf{I}_N)^{-1} \mathbf{K}_t \quad (\text{B.23})$$

If we let (λ, \mathbf{o}) be an eigenvalue-eigenvector pair for \mathbf{K}_t , it follows from (B.23), that $(\lambda - \lambda^2 / (\lambda + \gamma_t), \mathbf{o})$ is an eigenvalue-eigenvector pair for \mathbf{K}_{t+1} , so that we have the following recursion for eigenvalues of the \mathbf{K}_t -matrices:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} - \frac{(\lambda_i^{(t)})^2}{\lambda_i^{(t)} + \gamma_t} = \frac{\lambda_i^{(t)} \gamma_t}{\lambda_i^{(t)} + \gamma_t} \quad i = 1, \dots, N.$$

From this it can be easily verified by induction, that

$$\lambda_i^{(t)} = \frac{\lambda_i^{(0)}}{\lambda_i^{(0)} \gamma_{i-1}^- + 1} \quad i = 1, \dots, N,$$

and we see that the solution to the recursion (B.7) is

$$\mathbf{K}_t = \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)}}{\lambda_i^{(0)} \gamma_{i-1}^- + 1} \right) \mathbf{O}^\top. \quad (\text{B.24})$$

Notice that (B.24) trivially holds for $t = 0$. Next, we note the following consequences of (B.24): Firstly, it follows from (B.23) that

$$(\mathbf{K}_t + \gamma_t \mathbf{I})^{-1} = \mathbf{O} \text{diag}_i \left(\frac{1}{\frac{\lambda_i^{(0)}}{\lambda_i^{(0)} \gamma_{i-1}^- + 1} + \gamma_t} \right) \mathbf{O}^\top = \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)} \gamma_{i-1}^- + 1}{\gamma_t (\lambda_i^{(0)} \gamma_{i-1}^- + 1)} \right) \mathbf{O}^\top. \quad (\text{B.25})$$

Secondly, combining (B.24) and (B.25) gives

$$\mathbf{K}_t (\mathbf{K}_t + \gamma_t \mathbf{I})^{-1} = (\mathbf{K}_t + \gamma_t \mathbf{I})^{-1} \mathbf{K}_t = \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)}}{\gamma_t (\lambda_i^{(0)} \gamma_{i-1}^- + 1)} \right) \mathbf{O}^\top. \quad (\text{B.26})$$

Finally, we see that

$$\begin{aligned} \mathbf{I}_N - [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} \mathbf{K}_t &= \mathbf{O} \text{diag}_i \left(1 - \frac{\lambda_i^{(0)} / \gamma_t}{\lambda_i^{(0)} \gamma_{i-1}^- + 1} \right) \mathbf{O}^\top \\ &= \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)} \gamma_{i-1}^- + 1}{\lambda_i^{(0)} \gamma_{i-1}^- + 1} \right) \mathbf{O}^\top. \end{aligned} \quad (\text{B.27})$$

Now, we can turn our attention to $k_t(x, \mathbf{x}^\top)$. Here we have

$$\begin{aligned}
 k_{t+1}(x, \mathbf{x}^\top) &= (k_{t+1}(x, x_1), k_{t+1}(x, x_2), \dots, k_{t+1}(x, x_N)) \\
 &= (k_t(x, x_1) - k_t(x, \mathbf{x}^\top)[\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} k_t(\mathbf{x}, x_1), \\
 &\quad \dots, k_t(x, x_N) - k_t(x, \mathbf{x}^\top)[\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} k_t(\mathbf{x}, x_N)) \\
 &= k_t(x, \mathbf{x}^\top) - (k_t(x, \mathbf{x}^\top)[\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} k_t(\mathbf{x}, x_1), \\
 &\quad \dots, k_t(x, \mathbf{x}^\top)[\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} k_t(\mathbf{x}, x_N)) \\
 &= k_t(x, \mathbf{x}^\top) - k_t(x, \mathbf{x}^\top)[\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} \mathbf{K}_t \\
 &= k_t(x, \mathbf{x}^\top) (\mathbf{I}_N - [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} \mathbf{K}_t)
 \end{aligned}$$

Iterating the recursion for $k_t(x, \mathbf{x})$ above and using (B.27) we obtain for $t = 1, \dots$,

$$\begin{aligned}
 k_t(x, \mathbf{x}^\top) &= k_0(x, \mathbf{x}^\top) \prod_{j=0}^{t-1} \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)} \gamma_{j-1}^- + 1}{\lambda_i^{(0)} \gamma_j^- + 1} \right) \mathbf{O}^\top \\
 &= k_0(x, \mathbf{x}^\top) \mathbf{O} \text{diag}_i \left(\prod_{j=0}^{t-1} \frac{\lambda_i^{(0)} \gamma_{j-1}^- + 1}{\lambda_i^{(0)} \gamma_j^- + 1} \right) \mathbf{O}^\top \\
 &= k_0(x, \mathbf{x}^\top) \mathbf{O} \text{diag}_i \left(\frac{1}{\lambda_i^{(0)} \gamma_{t-1}^- + 1} \right) \mathbf{O}^\top.
 \end{aligned} \tag{B.28}$$

Next, we prove (B.9). First, we note that (B.5) gives the following recursion for $m_{t+1}(\mathbf{x})$:

$$\begin{aligned}
 m_{t+1}(\mathbf{x}) &= m_t(\mathbf{x}) + k_t(\mathbf{x}, \mathbf{x}^\top)[k_t(\mathbf{x}, \mathbf{x}^\top) + \gamma_t \mathbf{I}_N]^{-1} (\mathbf{y} - m_t(\mathbf{x})) \\
 &= m_t(\mathbf{x}) + \mathbf{K}_t [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} (\mathbf{y} - m_t(\mathbf{x})) \\
 &= (\mathbf{I}_t - \mathbf{K}_t [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1}) m_t(\mathbf{x}) + \mathbf{K}_t [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} \mathbf{y}
 \end{aligned} \tag{B.29}$$

It follows by induction that

$$m_t(\mathbf{x}) = \mathbf{O} \text{diag}_i \left(\frac{\gamma_{t-1}^- \lambda_i^{(0)}}{\gamma_{t-1}^- \lambda_i^{(0)} + 1} \right) \mathbf{O}^\top \mathbf{y} \tag{B.30}$$

The claim is trivial for $t = 0$. Assuming (B.30) is true and using (B.29) together with (B.25) and (B.27), we find

$$\begin{aligned}
 m_{t+1}(\mathbf{x}) &= \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)} \gamma_{t-1}^- + 1}{\lambda_i^{(0)} \gamma_t^- + 1} \right) \mathbf{O}^\top \mathbf{O} \text{diag}_i \left(\frac{\gamma_{t-1}^- \lambda_i^{(0)}}{\gamma_{t-1}^- \lambda_i^{(0)} + 1} \right) \mathbf{O}^\top \mathbf{y} \\
 &\quad + \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)}}{\gamma_t (\lambda_i^{(0)} \gamma_t^- + 1)} \right) \mathbf{O}^\top \mathbf{y} \\
 &= \mathbf{O} \text{diag}_i \left(\frac{\gamma_{t-1}^- \lambda_i^{(0)}}{\lambda_i^{(0)} \gamma_t^- + 1} \right) \mathbf{O}^\top \mathbf{y} + \mathbf{O} \text{diag}_i \left(\frac{\lambda_i^{(0)}}{\gamma_t (\lambda_i^{(0)} \gamma_t^- + 1)} \right) \mathbf{O}^\top \mathbf{y} \\
 &= \mathbf{O} \text{diag}_i \left(\frac{\gamma_{t-1}^- \lambda_i^{(0)}}{\lambda_i^{(0)} \gamma_t^- + 1} + \frac{\lambda_i^{(0)}}{\gamma_t (\lambda_i^{(0)} \gamma_t^- + 1)} \right) \mathbf{O}^\top \mathbf{y}
 \end{aligned}$$

$$\begin{aligned}
 &= \mathbf{O} \operatorname{diag}_i \left(\frac{\gamma_t \gamma_{t-1}^- \lambda_i^{(0)} + \lambda_i^{(0)}}{\gamma_t (\lambda_i^{(0)} \gamma_t^- + 1)} \right) \mathbf{O}^\top \mathbf{y} \\
 &= \mathbf{O} \operatorname{diag}_i \left(\frac{\lambda_i^{(0)} (\gamma_{t-1}^- + 1/\gamma_t)}{\lambda_i^{(0)} \gamma_t^- + 1} \right) \mathbf{O}^\top \mathbf{y} \\
 &= \mathbf{O} \operatorname{diag}_i \left(\frac{\lambda_i^{(0)} \gamma_t^-}{\lambda_i^{(0)} \gamma_t^- + 1} \right) \mathbf{O}^\top \mathbf{y}
 \end{aligned}$$

as required. \square

Theorem B.3 (from page 65). *The solution to the recursions (B.5) and (B.6) is given by*

$$\begin{aligned}
 m_t(x) &= k_0(x, \mathbf{x}^\top) (\mathbf{K} + \mathbf{I}_N / \gamma_{t-1}^-)^{-1} \mathbf{y} \\
 k_t(x, y) &= k_0(x, y) - k_0(x, \mathbf{x}^\top) (\mathbf{K} + \mathbf{I}_N / \gamma_{t-1}^-)^{-1} k_0(\mathbf{x}, y)
 \end{aligned} \tag{B.10}$$

for $t = 1, 2, \dots$

Proof. We first consider k_t . By combining (B.6), (B.24), (B.25) and (B.28), we find

$$\begin{aligned}
 k_{t+1}(x, y) &= k_t(x, y) - k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\gamma_t (\lambda_i^{(0)} \gamma_t^- + 1) (\lambda_i^{(0)} \gamma_{t-1}^- + 1)} \right) \mathbf{O}^\top k_0(\mathbf{x}, y) \\
 &= k_t(x, y) - k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)}} \left(\frac{1}{\lambda_i^{(0)} \gamma_{t-1}^- + 1} - \frac{1}{\lambda_i^{(0)} \gamma_t^- + 1} \right) \right) \mathbf{O}^\top k_0(\mathbf{x}, y).
 \end{aligned} \tag{B.31}$$

Iterating the equation above gives

$$\begin{aligned}
 k_t(x, y) &= k_0(x, y) - \sum_{j=0}^{t-1} k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)}} \left(\frac{1}{\lambda_i^{(0)} \gamma_{j-1}^- + 1} - \frac{1}{\lambda_i^{(0)} \gamma_j^- + 1} \right) \right) \mathbf{O}^\top k_0(\mathbf{x}, y) \\
 &= k_0(x, y) - k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)}} \sum_{j=0}^{t-1} \left(\frac{1}{\lambda_i^{(0)} \gamma_{j-1}^- + 1} - \frac{1}{\lambda_i^{(0)} \gamma_j^- + 1} \right) \right) \mathbf{O}^\top k_0(\mathbf{x}, y) \\
 &= k_0(x, y) - k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)}} \left(\frac{1}{\lambda_i^{(0)} \gamma_{-1}^- + 1} - \frac{1}{\lambda_i^{(0)} \gamma_{t-1}^- + 1} \right) \right) \mathbf{O}^\top k_0(\mathbf{x}, y)
 \end{aligned}$$

and since $\gamma_{-1}^- = 0$ we see that

$$\begin{aligned}
 k_t(x, y) &= k_0(x, y) - k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)}} \left(1 - \frac{1}{\lambda_i^{(0)} \gamma_{t-1}^- + 1} \right) \right) \mathbf{O}^\top k_0(\mathbf{x}, y) \\
 &= k_0(x, y) - k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)} + 1/\gamma_{t-1}^-} \right) \mathbf{O}^\top k_0(\mathbf{x}, y) \\
 &= k_0(x, y) - k_0(x, \mathbf{x}^\top) (\mathbf{K} + \mathbf{I}_N / \gamma_{t-1}^-)^{-1} k_0(\mathbf{x}, y)
 \end{aligned}$$

as required.

Next, we consider (B.10). From (B.5) we have

$$m_{t+1}(x) = m_t(x) + k_t(x, \mathbf{x}^\top) [\mathbf{K}_t + \gamma_t \mathbf{I}_N]^{-1} (\mathbf{y} - m_t(\mathbf{x}))$$

Using (B.8), (B.9) and (B.25), we may rewrite this as

$$\begin{aligned}
 m_{t+1}(x) &= m_t(x) + k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\gamma_t(\lambda_i^{(0)} \gamma_t^- + 1)} \right) \mathbf{O}^\top \left(\mathbf{y} - \mathbf{O} \operatorname{diag}_i \left(\frac{\lambda_i^{(0)} \gamma_{t-1}^-}{\lambda_i^{(0)} \gamma_{t-1}^- + 1} \right) \mathbf{O}^\top \mathbf{y} \right) \\
 &= m_t(x) + k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\gamma_t(\lambda_i^{(0)} \gamma_t^- + 1)} - \frac{1}{\gamma_t(\lambda_i^{(0)} \gamma_t^- + 1)} \frac{\lambda_i^{(0)} \gamma_{t-1}^-}{\lambda_i^{(0)} \gamma_{t-1}^- + 1} \right) \mathbf{O}^\top \mathbf{y} \\
 &= m_t(x) + k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\gamma_t(\lambda_i^{(0)} \gamma_t^- + 1)(\lambda_i^{(0)} \gamma_{t-1}^- + 1)} \right) \mathbf{O}^\top \mathbf{y} \tag{B.32}
 \end{aligned}$$

Notice, that the entries of the diagonal-matrix appearing in (B.32) are identical to those appearing in (B.31). Using the same calculations as above, we find

$$\begin{aligned}
 m_{t+1}(x) &= m_0(x) + k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)} + 1/\gamma_{t-1}^-} \right) \mathbf{O}^\top \mathbf{y} \\
 &= k_0(x, \mathbf{x}^\top) \mathbf{O} \operatorname{diag}_i \left(\frac{1}{\lambda_i^{(0)} + 1/\gamma_{t-1}^-} \right) \mathbf{O}^\top \mathbf{y},
 \end{aligned}$$

from which (B.10) follows. \square

Corollary B.4 (from page 65). *If we let $\mathcal{D}_t = \{(x_{ij}, y_{ij}) \mid i = 1, \dots, N, \text{ and } j = 1, \dots, t\}$ where $x_{ij} = x_i$ for all i and j , then the posterior distribution of \mathbf{f} is $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ where*

$$\boldsymbol{\mu}_t = \begin{pmatrix} \mathbf{K}(\mathbf{K} + \gamma/t\mathbf{I}_N)^{-1}\mathbf{y} \\ \vdots \\ \mathbf{K}(\mathbf{K} + \gamma/t\mathbf{I}_N)^{-1}\mathbf{y} \end{pmatrix} \tag{B.33}$$

$$\boldsymbol{\Sigma}_t = \mathbf{1}\mathbf{1}^\top \otimes (\mathbf{K} - \mathbf{K}(\mathbf{K} + \gamma/t\mathbf{I}_N)^{-1}\mathbf{K}) \tag{B.34}$$

where \otimes is the Kronecker product.

Proof. For simplicity, we only prove the statement for $t = 2$, as the general proof is similar. The key observation is that the prior covariance matrix may be written as

$$\begin{pmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{pmatrix} + \gamma\mathbf{I}_{2N} = \begin{pmatrix} \mathbf{K} + \gamma\mathbf{I}_N & \mathbf{K} \\ \mathbf{K} & \mathbf{K} + \gamma\mathbf{I}_N \end{pmatrix}$$

Using the formula for the inverse of a partitioned matrix, we find

$$\begin{pmatrix} \mathbf{K} + \gamma\mathbf{I}_N & \mathbf{K} \\ \mathbf{K} & \mathbf{K} + \gamma\mathbf{I}_N \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}$$

where

$$\begin{aligned}
 \mathbf{A}_{11} &= \mathbf{A}_{22} = (\mathbf{K} + \gamma\mathbf{I}_N - \mathbf{K}(\mathbf{K} + \gamma\mathbf{I}_N)^{-1}\mathbf{K})^{-1} \\
 \mathbf{A}_{12} &= \mathbf{A}_{21} = -(\mathbf{K} + \gamma\mathbf{I}_N - \mathbf{K}(\mathbf{K} + \gamma\mathbf{I}_N)^{-1}\mathbf{K})^{-1}\mathbf{K}(\mathbf{K} + \gamma\mathbf{I}_N)^{-1}
 \end{aligned}$$

and these may be further simplified to obtain

$$\left(\begin{pmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{pmatrix} + \mathbf{I}_{2N} \right)^{-1} = \begin{pmatrix} -(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}(\mathbf{K} + \gamma\mathbf{I}_N) & (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\mathbf{K} \\ -(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\mathbf{K} & (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}(\mathbf{K} + \gamma\mathbf{I}_N) \end{pmatrix},$$

so we find

$$\begin{aligned}\mu_2 &= \begin{Bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{Bmatrix} \begin{Bmatrix} (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}(\mathbf{K} + \gamma\mathbf{I}_N) & -(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\mathbf{K} \\ -(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\mathbf{K} & (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}(\mathbf{K} + \gamma\mathbf{I}_N) \end{Bmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{y} \end{pmatrix} \\ &= \begin{Bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{Bmatrix} \begin{Bmatrix} (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\gamma\mathbf{y} \\ (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\gamma\mathbf{y} \end{Bmatrix} \\ &= \begin{pmatrix} 2\mathbf{K}(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\gamma\mathbf{y} \\ 2\mathbf{K}(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\gamma\mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{K}(\mathbf{K} + (\gamma/2)\mathbf{I}_N)^{-1}\mathbf{y} \\ \mathbf{K}(\mathbf{K} + (\gamma/2)\mathbf{I}_N)^{-1}\mathbf{y} \end{pmatrix}.\end{aligned}$$

Similarly, the covariance matrix is found as

$$\Sigma_2 = \begin{Bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{Bmatrix} - \begin{Bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{Bmatrix} \begin{Bmatrix} (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}(\mathbf{K} + \gamma\mathbf{I}_N) & -(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\mathbf{K} \\ -(2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}\mathbf{K} & (2\mathbf{K}\gamma + \gamma^2\mathbf{I}_N)^{-1}(\mathbf{K} + \gamma\mathbf{I}_N) \end{Bmatrix} \begin{Bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{Bmatrix}$$

which may be simplified to give

$$\Sigma_2 = \begin{Bmatrix} \mathbf{K} - \mathbf{K}(\mathbf{K} + (\gamma/2)\mathbf{I}_N)^{-1}\mathbf{K} & \mathbf{K} - \mathbf{K}(\mathbf{K} + (\gamma/2)\mathbf{I}_N)^{-1}\mathbf{K} \\ \mathbf{K} - \mathbf{K}(\mathbf{K} + (\gamma/2)\mathbf{I}_N)^{-1}\mathbf{K} & \mathbf{K} - \mathbf{K}(\mathbf{K} + (\gamma/2)\mathbf{I}_N)^{-1}\mathbf{K} \end{Bmatrix},$$

as desired. \square

Proposition B.5 (from page 67). *Let $C(\sigma(a))$ be defined as in (B.20) with $\lambda = \sigma(a)$, then we have that*

$$\begin{aligned}C(\sigma(a)) &= \begin{cases} 2 & \text{if } a = 0 \\ \operatorname{acoth}\left(\frac{a}{2}\right) & \text{otherwise,} \end{cases} \\ \frac{d}{da} \log(C(\sigma(a))) &= \begin{cases} 0 & \text{if } a = 0 \\ \frac{1}{a} - \frac{1}{\sinh(a)} & \text{otherwise,} \end{cases} \\ \frac{d^2}{da^2} \log(C(\sigma(a))) &= \begin{cases} \frac{1}{6} & \text{if } a = 0 \\ -\frac{1}{a^2} + \frac{\operatorname{coth}(a)}{\sinh(a)} & \text{otherwise,} \end{cases}\end{aligned}$$

where coth , and \sinh are the hyperbolic cotangent, and sine functions, respectively.

Proof. First we note that for $a \neq 0$ it can be shown that

$$C(\sigma(a)) = \frac{2\tanh^{-1}(1 - 2\sigma(a))}{1 - 2\sigma(a)} = \operatorname{acoth}\left(\frac{a}{2}\right), \quad (\text{B.35})$$

which follows from the fact that $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$

$$1 - 2\sigma(a) = 1 - \frac{2e^a}{1+e^a} = \frac{1+e^a}{1+e^a} - \frac{2e^a}{1+e^a} = \frac{1-e^a}{1+e^a} = \tanh\left(-\frac{a}{2}\right)$$

and thus it follows directly that

$$2\tanh^{-1}(1 - 2\sigma(a)) = 2\tanh^{-1}\left(\tanh\left(-\frac{a}{2}\right)\right) = \frac{-2a}{2} = -a$$

and finally by $\operatorname{coth}(x) = \frac{1}{\tanh(x)}$ and $\tanh(-x) = -\tanh(x)$ that

$$\frac{1}{1 - 2\sigma(a)} = \frac{1}{-\tanh\left(\frac{a}{2}\right)} = -\operatorname{coth}\left(\frac{a}{2}\right),$$

which combines to the claim in (B.35). Now, By properties of the hyperbolic functions⁴ we get by direct computation that

$$\begin{aligned}
 \frac{d}{da} \log(C(\sigma(a))) &= \frac{d}{da} \log(a) + \frac{d}{da} \log \coth\left(\frac{a}{2}\right) && \text{for } a \neq 0 \\
 &= \frac{1}{a} + \frac{1}{\coth\left(\frac{a}{2}\right)} \frac{d}{da} \coth\left(\frac{a}{2}\right) && \text{for } a \neq 0 \\
 &= \frac{1}{a} + \frac{\sinh\left(\frac{a}{2}\right)}{\cosh\left(\frac{a}{2}\right)} \left(-\frac{1}{2\sinh^2\left(\frac{a}{2}\right)}\right) && \text{for } a \neq 0 \\
 &= \frac{1}{a} - \frac{1}{2\cosh\left(\frac{a}{2}\right)\sinh\left(\frac{a}{2}\right)} && \text{for } a \neq 0 \\
 &= \frac{1}{a} - \frac{1}{\sinh(a)} && \text{for } a \neq 0 \\
 &= \frac{1}{a} - \operatorname{csch}(a) && \text{for } a \neq 0.
 \end{aligned}$$

Additionally, by further properties of the hyperbolic functions⁵ it follows directly that

$$\begin{aligned}
 \frac{d^2}{d^2a} \log(C(\sigma(a))) &= -\frac{1}{a^2} - \frac{d}{da} \operatorname{csch}(x) && \text{for } a \neq 0 \\
 &= -\frac{1}{a^2} + \operatorname{csch}(a)\coth(a) && \text{for } a \neq 0. \\
 &= -\frac{1}{a^2} + \frac{1}{\sinh(a)} \frac{1}{\tanh(a)} && \text{for } a \neq 0.
 \end{aligned}$$

It remains to show both derivatives for $a = 0$. First let $f(a) = \log(C(\sigma(a))) = \log a \coth\left(\frac{a}{2}\right)$ for $a \neq 0$ and $\log(2)$ elsewhere, then we have that

$$f(-x) = \log\left(-x \coth\left(\frac{-x}{2}\right)\right) = \log\left(x \coth\left(\frac{x}{2}\right)\right) = f(x),$$

and we can without loss of generality merely consider limits from the right. See that

$$\begin{aligned}
 \lim_{h \downarrow 0} \frac{f(h) - f(0)}{h} &= \lim_{h \downarrow 0} \frac{d}{dx} f(h) \\
 &= \lim_{h \downarrow 0} \left(\frac{1}{h} - \frac{1}{\sinh(h)}\right) \\
 &= \lim_{h \downarrow 0} \left(\frac{\sinh(h) - h}{h \sinh(h)}\right),
 \end{aligned}$$

and by applying L'Hôpital's rule for 0/0 limits we have that

$$\lim_{h \downarrow 0} \frac{f(h) - f(0)}{h} = \lim_{h \downarrow 0} \left(\frac{\cosh(h) - 1}{\sinh(h) + x \cosh(h)}\right) = 0.$$

By a similar argument we get that since $\sinh(-x) = -\sinh(x)$ then $\frac{d}{dx} f(-x) = -\frac{d}{dx} f(x)$, and we can w.l.o.g. look only at right limits and apply L'Hôpital's rule, which yields that

$$\lim_{h \downarrow 0} \frac{\frac{d}{dx} f(h) - \frac{d}{dx} f(0)}{h} = \lim_{h \downarrow 0} \frac{d^2}{d^2x} f(h) = \lim_{h \downarrow 0} \left(-\frac{1}{a^2} + \frac{1}{\sinh(a)} \frac{1}{\tanh(a)}\right) = \frac{1}{6}$$

⁴ In particular that $\sinh(2x) = 2\sinh(x)\cosh(x)$ and that $\frac{d}{da} \coth(x) = -\frac{1}{\sinh^2(x)}$ for $x \neq 0$.

⁵ Here, in particular $\frac{d}{da} \operatorname{csch}(x) = -\operatorname{csch}(x)\coth(x)$, $\coth(x) = \frac{1}{\tanh(x)}$ and $\operatorname{csch}(x) = \frac{1}{\sinh(x)}$.

and we are done. \square

Proposition B.6 (from page 69). **(A)** A single step of probabilistic distillation using \mathcal{D}_t and a Gaussian prior $\mathcal{GP}(0, k)$ yields the same posterior distribution as a single step of probabilistic distillation using a Gaussian prior $\mathcal{GP}(0, tk)$. **(B)** Performing t iterations of the probabilistic distillation using \mathcal{D} and starting with an initial Gaussian prior $\mathcal{GP}(0, k)$ yields approximately the same posterior as a single step of probabilistic distillation using a Gaussian prior $\mathcal{GP}(0, tk)$.

Proof. **(A)** First, we notice that duplicating the data simply implies that the term corresponding to the conditional distribution of the data given the latent variables is repeated t times. Using the notation from section B.II.1, this implies that the log posterior will satisfy

$$\psi(\mathbf{f}) \stackrel{\text{def}}{=} \log p(\mathbf{f} | \mathbf{y}) \propto \log p(\mathbf{f}) + t \log p(\mathbf{y} | \mathbf{f})$$

with gradient

$$\nabla \psi(\mathbf{f}) = -\mathbf{K}^{-1} \mathbf{f} + t(\mathbf{y} - \sigma)$$

and by setting this equal to zero, we see that the maximizer $\hat{\mathbf{f}}$ will satisfy

$$\hat{\mathbf{f}} = (t\mathbf{K})(\mathbf{y} - \sigma).$$

Compared to the same equation for single-step distillation implied by (B.17), we see that the two methods give the same result.

Next, we wish to show **(B)**. For simplicity, we consider $t = 2$ as the argument for general t is similar. From **(A)** we see that we can obtain the desired result by showing that 2 steps of distillation will give approximately the same result as a single step of distillation based on \mathcal{D}_2 . For the latter, we see from the proof of **(A)** that the posterior distribution is proportional to

$$\left(\prod_{n=1}^N \sigma(f_n)^{t_n} (1 - \sigma(f_n))^{1-t_n} \right)^2 \exp\left(-\frac{1}{2} \mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f}\right). \quad (\text{B.36})$$

If we define

$$\varphi(\mathbf{f}) \stackrel{\text{def}}{=} \left(\prod_{n=1}^N \sigma(f_n)^{t_n} (1 - \sigma(f_n))^{1-t_n} \right) \exp\left(-\frac{1}{2} \mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f}\right)$$

we may rewrite (B.36) as

$$\left(\prod_{n=1}^N \sigma(f_n)^{t_n} (1 - \sigma(f_n))^{1-t_n} \right) \varphi(\mathbf{f}). \quad (\text{B.37})$$

Now we simply observe that 2-step distillation will correspond to replacing $\varphi(\mathbf{f})$ in (B.37) with the appropriate Gaussian distribution coming from the Laplace distribution, and hence the approximation comes from the fact that $\varphi(\mathbf{f})$ is not itself exactly proportional to a Gaussian distribution. \square

Distilling from Similar Tasks for Transfer Learning on a Budget

PUBLISHED IN INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV) 2023

Kenneth Borup

Aarhus University

Cheng Perng Phoo

Cornell University

Bharath Hariharan

Cornell Univeristy

Abstract. We address the challenge of getting efficient yet accurate recognition systems with limited labels. While recognition models improve with model size and amount of data, many specialized applications of computer vision have severe resource constraints both during training and inference. Transfer learning is an effective solution for training with few labels, however often at the expense of a computationally costly fine-tuning of large base models. We propose to mitigate this unpleasant trade-off between compute and accuracy via semi-supervised cross-domain distillation from a set of diverse source models. Initially, we show how to use task similarity metrics to select a single suitable source model to distill from, and that a good selection process is imperative for good downstream performance of a target model. We dub this approach `DISTILLNEAREST`. Though effective, `DISTILLNEAREST` assumes a single source model matches the target task, which is not always the case. To alleviate this, we propose a weighted multi-source distillation method to distill multiple source models trained on different domains weighted by their relevance for the target task into a single efficient model (named `DISTILLWEIGHTED`). Our methods need no access to source data and merely need features and pseudo-labels of the source models. When the goal is accurate recognition under computational constraints, both `DISTILLNEAREST` and `DISTILLWEIGHTED` approaches outperform both transfer learning from strong ImageNet initializations as well as state-of-the-art semi-supervised techniques such as `FixMatch`. Averaged over 8 diverse target tasks our multi-source method outperforms the baselines by 5.6%-points and 4.5%-points, respectively.

C.1 Introduction

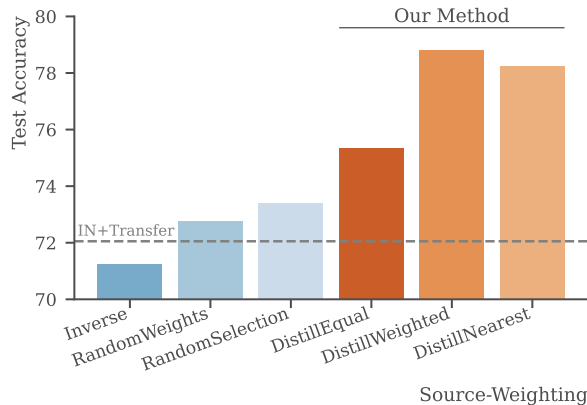


Figure C.1: Average test accuracy over five target tasks with different methods for weighting source models for distillation. Our methods outperform the baselines and transfer learning from ImageNet. See Section C.5.3 for details.

Recognition models get more accurate the larger they are and the more data they are trained on [Sun et al. \(2017\)](#); [Zhai et al. \(2022\)](#); [Kolesnikov et al. \(2020\)](#). This is a problem for many applications of interest in medicine (e.g. X-ray analysis) or science (e.g. satellite-image analysis) where both labeled training data, as well as computational resources needed to train such large models, are lacking.

The challenge of limited labeled data can potentially be alleviated by fine-tuning large-scale “foundation models” [Zhai et al. \(2022\)](#); [Dehghani et al. \(2023\)](#); [Kolesnikov et al. \(2020\)](#). However, fine-tuning is computationally expensive, especially when one looks at foundation models with billions of parameters [Dehghani et al. \(2023\)](#). Unfortunately, all evidence suggests that larger foundation models perform better at fine-tuning [Kolesnikov et al. \(2020\)](#); [Zhai et al. \(2022\)](#). This leaves downstream applications the unpleasant trade-off of expensive computational hardware for fine-tuning large models or inaccurate results from smaller models. Motivated by this challenge, we ask *can we train accurate models on tight data and compute budgets without fine-tuning large foundation models?*

To set the scene, we assume the existence of a diverse set (both in architecture and task) of pre-trained source models (or foundation models). We do not have the resources to fine-tune these models, but we assume we can perform inference on these models and extract features, e.g. through APIs on cloud services [Bisong \(2019\)](#); [Rekognition \(2023\)](#). For the target task, we assume that labeled data is very limited, but unlabeled data is available. We then propose a simple and effective strategy for building an accurate model for the target task: `DISTILLNEAREST`. Concretely, we first compute a measure of “task similarity” between our target task and each source model and rank the source models accordingly. Then we pseudo-label the unlabeled data using the most similar source model. These pseudo-labels may not even be in the same label space as the target task, but we conjecture that due to the similarity between the source and target tasks, the pseudo-labels will still *group* the target data points in a task-relevant manner. Finally, we train the target model using the pseudo-labels and the available ground truth labeled

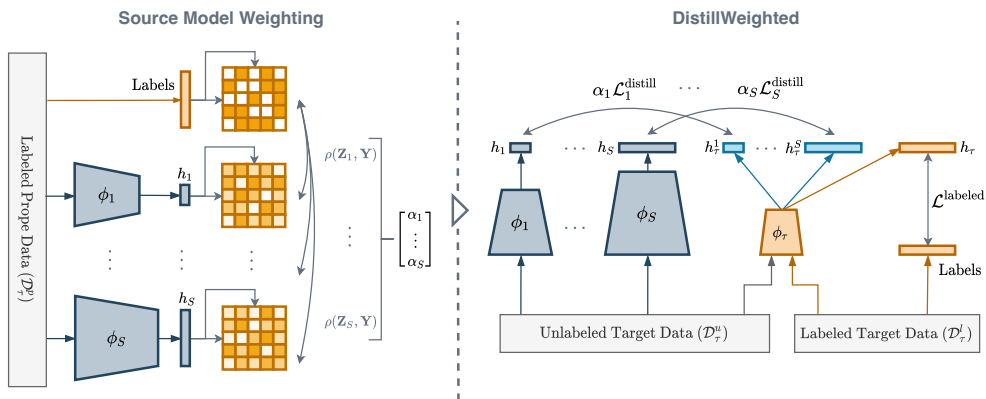


Figure C.2: We propose to weigh a set of S source models, $M_s = h_s \circ \phi_s$, by using task similarity metrics to estimate the alignment of each source model with the particular target task using a small probe set of labeled data, \mathcal{D}_t^p . Since the task similarity metrics are independent of feature dimension, we can utilize source models of any architecture and from any source task. We show that by choosing the weighting, $\alpha_1, \dots, \alpha_S$, this way we are able to improve performance over transfer from ImageNet and training with FixMatch amongst others (see e.g. Table C.1 and Figure C.3).

data. This allows us to bypass the large computations required to fine-tune source models and directly work on the target model. At the same time, we get to effectively use the knowledge of the large source model even if it is trained on a different task.

DISTILLNEAREST assumes that a *single* best source model exists. But for some target tasks, we might need to combine multiple source models to achieve a sufficiently diverse representation to distill. We, therefore, propose an extension of our approach that distills *multiple (diverse) source models* trained on different domains, weighted by their relevance for the target task. This extension obtains even further improvements on our target performance (see Figure C.1). We dub this method DISTILLWEIGHTED.

We summarize our contributions as follows:

- We train more than 200 models across a diverse set of source and target tasks using single-source distillation, and extensively show that the choice of source model is imperative for the predictive performance of the target model. To the best of our knowledge, no previous work has addressed how to efficiently select a teacher model for (cross-domain) distillation.
- We find that *task similarity metrics* correlate well with predictive performance and can be used to efficiently select and weight source models for single- and multi-source distillation without access to any source data.
- We show that our approaches yield the best accuracy on multiple target tasks under compute and data constraints. We compare our DISTILLNEAREST and DISTILLWEIGHTED methods to two baselines (transfer learning and FixMatch), as well as the naïve case of DISTILLWEIGHTED with *equal* weighting (called DISTILLEQUAL), among others. Averaged over 8 diverse datasets, our DISTILLWEIGHTED outperforms the baselines with at least 4.5% and in particular 17.5% on CUB200.

C.2 Related Work

Knowledge Distillation. One key aspect of our problem is to figure out how to compress single or multiple large foundation models into an efficient target model. A common approach is knowledge distillation (Ba and Caruana, 2014; Hinton et al., 2015) where an efficient student model is trained to mimic the output of a larger teacher model. However, most single-teacher (Romero et al., 2015; Mirzadeh et al., 2020; Park et al., 2019; Cho and Hariharan, 2019; Borup and Andersen, 2021) or multi-teacher knowledge distillation (You et al., 2017; Fukuda et al., 2017; Tan et al., 2019; Liu et al., 2020) research focuses on the closed set setup, where the teacher(s) and the student both attempts to tackle the same task. To the best of our knowledge, compressing models specializing in various tasks different from the target task has rarely been explored in the literature. Our paper explores this setup and illustrates that carefully distilling source models trained on different tasks can bring forth efficient yet accurate models.

Semi-Supervised Learning and Transfer. Given our target tasks are specified in a semi-supervised setting, it is customary to review methods for semi-supervised learning (SSL). The key to SSL approaches is how to effectively propagate label information from a small labeled dataset to a large unlabeled dataset. Along this vein, methods such as pseudo-labeling/self-training (Lee, 2013; Xie et al., 2020) or consistency regularization (Tarvainen and Valpola, 2017; Berthelot et al., 2019; Sohn et al., 2020) have shown remarkable results in reducing deep networks dependencies on large labeled datasets via unlabeled data. However, most SSL approaches focus on training models from scratch without considering the availability of pre-trained models. Given the increasing availability of large pre-trained models (Paszke et al., 2019; Wolf et al., 2020), recent work has started exploring the intersection between transfer learning and SSL (Phoo and Hariharan, 2021; Islam et al., 2021; Abuduweili et al., 2021). However, most of these works focus on how to transfer from a single pre-trained model to the target task. Our paper, however, explores an even more practical setup: how to transfer from multiple pre-trained models to a downstream task where in-domain unlabeled data are available. In principle, we could combine our approach with a lot of previous work on SSL to (potentially) gain even larger improvements, but to keep our method simple we leave such exploration to future work and focus on how to better utilize an available set of pre-trained models.

Multi-Source Domain Adaptation. Our setup also bears a resemblance with multi-source domain adaptation (MSDA) (Peng et al., 2019) in which the goal is to create a target model by leveraging multiple source models. However, MSDA methods often assume the source and target models share the same label space to perform domain alignment. We do not make such an assumption and in fact, focus on the case where the label space of source and target tasks have minimal to no overlap. Besides, a lot of the MSDA approaches (Zhao et al., 2018; Xu et al., 2018; Peng et al., 2019; Zhao et al., 2020) rely on the availability of source data or the fact that the source and target tasks share the same model architecture to build domain invariant features. Given the discrepancy in assumptions between MSDA and our setup, we do not consider any methods from this line of work as baselines.

Transfer Learning From Multiple Sources. Transfer learning from multiple different pre-trained models has been explored in different setups. [Bolya et al. \(2021\)](#) focuses on how to select a single good pre-trained model to use as a model initialization whereas we explore how to distill an efficient model from the pre-trained models (i.e. our target architecture could be different from those of the source models). [Agostinelli et al. \(2022\)](#) focuses on how to select a subset of pre-trained models to construct an (fine-tuned) ensemble, whereas we focus on creating a single model. [Li et al. \(2021\)](#) focuses on creating a generalist representation by equally distilling multiple pre-trained models using proxy/source data (which often requires high-capacity models) whereas our goal is to construct an efficient specialist model using the target data. All these works have indicated the importance of exploring how to best leverage a large collection of pre-trained models but due to differences in setup and assumptions, we do not (and could not) compare to them.

Task Similarity / Transferability Metrics. A key insight of our approach is to leverage the similarity between the target and source tasks to compare and weigh different pre-trained source models during distillation. Characterizing tasks (or similarities between tasks) is an open research question with various successes. A common approach is to embed tasks into a common vector space and characterize similarities in said space. Representative research along this line of work include [Achille et al. \(2019\)](#); [Peng et al. \(2020\)](#); [Wallace et al. \(2021\)](#). Another related line of work investigates transferability metrics ([Tran et al., 2019](#); [Bao et al., 2019](#); [Nguyen et al., 2020](#); [Dwivedi et al., 2020](#); [Dwivedi and Roig, 2019](#); [Bolya et al., 2021](#)). After all, one of the biggest use cases of task similarities is to predict how well a model transfers to new tasks. Since it is not our intention to define new task similarity/transferability metrics for distillation, we use already established metrics that capture the similarity between source representations and one-hot labels to weigh the source models. Under this purview, metrics that characterize similarities between features such as CKA ([Cortes et al., 2012](#); [Kornblith et al., 2019](#)) and transferability metrics based on features ([Dwivedi and Roig, 2019](#); [Bolya et al., 2021](#)) suffice.

C.3 Problem Setting

The aim of this paper is to train an accurate model for a given target task, subject to limited labeled data and computational constraints (e.g. limited compute resources). Formally, we assume that our target task is specified via a small labeled training set \mathcal{D}_τ^l . Furthermore, we assume (a) the availability of a set of unlabeled data, \mathcal{D}_τ^u , associated with the target task, and (b) the ability to perform inference on a set $\mathcal{S} = \{\mathcal{M}_s\}_{s=1}^S$ of S different *source* models, \mathcal{M}_s , trained on various source tasks different from the target task. We emphasize that we have no access to any source data which could be practical due to storage, privacy, and computational constraints. Neither do we need full access to the source models provided we can perform inference on the models anyway (e.g. through an API).

We assume that the architecture of the target model, \mathcal{M}_τ , must be chosen to meet any applicable computational constraints. This can imply that no suitable target architecture is available in the set of source models \mathcal{S} , making classical transfer learning impossible.

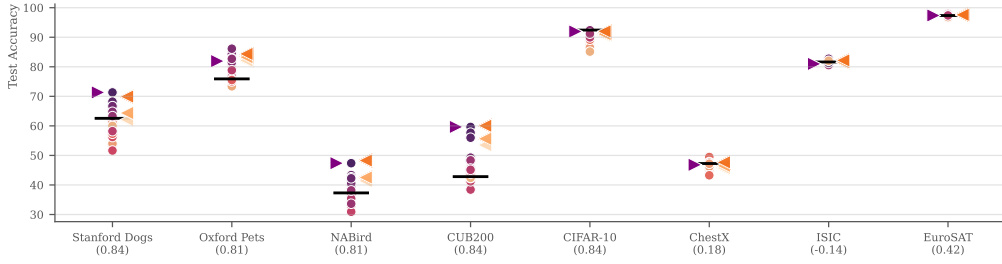



Figure C.3: Test accuracy for distillation with each dot representing single-source distillation from different source models. The colors represent the task similarity for the source models (from small to large; ). We include the performance from fine-tuning ImageNet (—), **DISTILLNEAREST**; i.e. distillation of the highest ranked source model (\blacktriangleright) as well as **DISTILLEQUAL** (\blacktriangleleft), and **DISTILLWEIGHTED**(p) where weights are proportional to task similarity with power $p = 1$ (\blacktriangleleft), and $p = 12$ (\blacktriangleleft), respectively. The numbers in parentheses at the bottom are Spearman correlations between the task similarity and test accuracy for single-source distillation.

For simplicity, we restrict our models (regardless of source or target) to classification models that can be parameterized as $\mathcal{M} = h \circ \phi$; the feature extractor ϕ embeds input \mathbf{x} into a feature representation, and the classifier head, h , maps the feature $\phi(\mathbf{x})$ into predicted conditional class probabilities, $P(\mathbf{y} | \mathbf{x})$.

C.4 Cross-Task Distillation for Constructing Efficient Models from Foundation Models

To construct an efficient model, we propose to distill large foundation models. Along this vein, we propose two variants: (a) **DISTILLNEAREST** that distills the single nearest source model (Section C.4.1) and (b) **DISTILLWEIGHTED** that distills a weighted collection of source models (Section C.4.2).

C.4.1 DISTILLNEAREST

To construct a single efficient target model, **DISTILLNEAREST** undergoes two steps sequentially: (a) selecting an appropriate source model and (b) distilling the knowledge from the selected source model into the target model. For ease of exposition, we start by explaining the distillation process and then discuss how to select an appropriate source model.

Distilling a selected source model. Given a selected source model \mathcal{M}_s , the target model $\mathcal{M}_\tau = h_\tau \circ \phi_\tau$ is trained by minimizing a weighted sum of two loss functions,

$$\mathcal{L}_{\text{single}} \stackrel{\text{def}}{=} \lambda \mathcal{L}^{\text{labeled}} + (1 - \lambda) \mathcal{L}_s^{\text{distill}}, \quad (\text{C.1})$$

where $\lambda \in [0, 1]$. The first loss function is the standard supervised objective over the labeled data,

$$\mathcal{L}^{\text{labeled}} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_\tau^l|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_\tau^l} \ell_{\text{CE}}(h_\tau(\phi_\tau(\mathbf{x}_i)), \mathbf{y}_i), \quad (\text{C.2})$$

where $\ell_{\text{CE}}(\cdot, \cdot)$ is the cross-entropy loss. The second loss function is a distillation loss over the unlabeled data,

$$\mathcal{L}_s^{\text{distill}} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_\tau^u|} \sum_{\mathbf{x}_i \in \mathcal{D}_\tau^u} \ell_{\text{CE}}(h_\tau^s(\phi_\tau(\mathbf{x}_i)), \mathcal{M}_s(\mathbf{x}_i)). \quad (\text{C.3})$$

Note, the source and target tasks do not share the same label space so we introduce an additional classifier head, h_τ^s , which maps the features from the target task feature extractor, ϕ_τ , to the label space of the source task. This additional classifier head, h_τ^s , is discarded after training and only the target classifier head, h_τ , is used for inference.

In principle, we could add additional semi-supervised losses, such as the FixMatch loss (Sohn et al., 2020) to propagate label information from the labeled set to the unlabeled set for better performance, but this would add additional hyperparameters and entangle the effect of our methods. We leave such explorations to future work.

Selecting the nearest source model for distillation. Selecting a source model for distillation is an under-explored problem. Given the recent success of using task similarity metrics (Bolya et al., 2021) for selecting foundation models for fine-tuning, we conjecture that high similarities between a source model and the target task could indicate better performance of the distilled model (we verify this in Section C.5.2). However, quantifying similarities between tasks/models is an open research question with various successes (Achille et al., 2019; Nguyen et al., 2020). For simplicity, we pick our similarity based on one simple intuition: target examples with identical labels should have similar source representations and vice versa. Along this vein, the recently introduced metric, PARC (Bolya et al., 2021) fits the bill.

For convenience, we briefly review PARC. Given a small labeled probe set $\mathcal{D}_\tau^p = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n \subseteq \mathcal{D}_\tau^l$ and a source representation of interest ϕ_s , PARC first constructs two distance matrices D_{ϕ_s} , D_Y based on the Pearson correlations between every pair of examples in the probe set;

$$\begin{aligned} D_{\phi_s} &= 1 - \text{pearson}(\{\phi_s(\mathbf{x}_i)\}_{i=1}^n), \\ D_Y &= 1 - \text{pearson}(\{\mathbf{y}_i\}_{i=1}^n). \end{aligned}$$

PARC is computed as the Spearman correlation between the lower triangles of the distance matrices;

$$\text{PARC}(\phi_s, Y) = \text{spear}(\{D_{\phi_s}[i, j]\}_{i < j}, \{D_Y[i, j]\}_{i < j}).$$

Intuitively, PARC quantifies the similarity of representations by comparing the (dis)similarity structures of examples within different feature spaces: if two representations are similar, then (dis)similar examples in one feature space should stay (dis)similar in the other feature space. In Figure C.3 and C.4 we show that ranking source models by PARC correlates well with test accuracy and that selecting an appropriate source model can yield significant improvements.

C.4.2 DISTILLWEIGHTED

Above, DISTILLNEAREST assumes a single optimal source model exists for the target task, but what if no single source model aligns well with our target task? To alleviate this issue, we propose to distill multiple source models, weighted according to their similarities

		Target Data		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
		Labeled	Unlabeled									
MobileNetV3 (0.24 GFLOPs)	IN+Transfer	✓	–	92.4	42.8	47.3	97.4	81.6	37.3	75.9	62.6	67.2
	IN+FixMatch	✓	✓	93.5	41.9	38.5	98.1	82.6	<u>42.8</u>	83.4	<u>65.8</u>	68.3
	DISTILLRANDOMSELECTION	✓	✓	89.6	46.5	46.6	97.4	<u>81.8</u>	39.0	79.4	61.9	67.8
	(Ours) DISTILLNEAREST	✓	✓	92.0	59.6	46.8	97.4	81.0	47.4	81.9	71.3	72.2
	DISTILLEQUAL	✓	✓	90.8	<u>53.5</u>	45.7	97.5	81.5	41.4	<u>82.1</u>	62.1	<u>69.3</u>
	DISTILLRANDOMWEIGHTS	✓	✓	87.9	44.9	<u>46.9</u>	97.8	81.6	39.6	80.2	59.2	67.3
(Ours) DISTILLWEIGHTED	✓	✓	<u>92.0</u>	60.0	47.7	<u>97.6</u>	82.2	48.3	84.4	69.9	72.8	
AlexNet (0.71 GFLOPs)	IN+Transfer	✓	–	85.0	18.4	46.2	91.9	67.8	13.0	50.9	29.1	50.3
	Fine-tune Selected Source	✓	–	88.0	30.4	42.9	89.8	74.5	17.9	66.8	41.3	56.5
GoogLeNet (1.51 GFLOPs)	IN+Transfer	✓	–	91.8	42.8	41.4	96.8	80.5	36.5	84.8	65.9	67.6
	Fine-tune Selected Source	✓	–	91.6	61.2	48.6	96.9	78.3	33.0	87.8	71.8	71.2
ResNet-18 (1.83 GFLOPs)	IN+Transfer	✓	–	92.2	37.8	45.2	96.6	80.2	34.0	80.2	58.2	65.6
	Fine-tune Selected Source	✓	–	91.3	58.2	46.4	97.0	75.8	35.4	80.7	69.3	69.3
ResNet-50 (4.14 GFLOPs)	IN+Transfer	✓	–	92.9	42.0	43.4	96.8	79.9	39.9	83.3	65.9	68.0
	Fine-tune Selected Source	✓	–	93.0	70.8	43.9	97.2	81.3	47.4	84.8	79.3	74.7

Table C.1: Cross-task distillation compared to baselines. MobileNetV3 models (target architecture) trained with our methods are highly competitive with baseline methods on MobileNetV3 as well as baseline methods for more demanding model architectures (source architectures: Alexnet, GoogLeNet, ResNet-18, ResNet-50). We highlight the top 3 methods, which comply with compute requirements (i.e. MobileNetV3) for each target task by **bold**, **blue**, and underline, respectively. We also indicate the target data used by different methods.

with the target tasks. In the following, we explain our weighted distillation objective and how the weights are constructed. Figure C.2 is a schematic depiction of the approach `DISTILLWEIGHTED`.

Weighted objective for distilling multiple sources. Given a set of source models $\mathcal{S} = \{M_s\}_{s=1}^S$, we modify the distillation loss of (C.1) with a weighted sum of multiple distillation losses (one for each source model):

$$\mathcal{L}_{\text{multi}} \stackrel{\text{def}}{=} \lambda \mathcal{L}^{\text{labeled}} + (1 - \lambda) \sum_{s=1}^S \alpha_s \mathcal{L}_s^{\text{distill}}, \quad (\text{C.4})$$

where $\lambda, \alpha_1, \dots, \alpha_S \in [0, 1]$ ($\mathcal{L}^{\text{labeled}}$ and $\mathcal{L}_s^{\text{distill}}$ are as defined in (C.2) and (C.3), respectively). Here α_s is the relative weight assigned to each source model such that $\sum_{s=1}^S \alpha_s = 1$. Once again, we could add additional semi-supervised losses, such as the FixMatch loss, but to ensure simplicity, we leave such explorations for future research.

Task similarity weighting of source models. Simply assigning equal weight to all source models is sub-optimal (e.g. weighing source models trained on ImageNet and Chest X-ray equally might not be optimal for recognizing birds). As such, we propose to compute the source weight α_s from a task similarity metric between the s -th source model and the target task. In particular, let e_s be such a similarity metric, then we

compute the source weights $\{\alpha_i\}_{i \in [S]}$ as

$$\alpha_i = \frac{e_i^p}{\sum_{s=1}^S e_s^p}, \quad \text{where } e_j = \max(0, e_j) \quad (\text{C.5})$$

for $j = 1, \dots, S$. Here p is a hyperparameter to re-scale the distribution of the weights. Larger p assigns more weight to the most similar source models, while $p = 0$ corresponds to equal weights for all models (denoted `DISTILLEQUAL`), and $p \rightarrow \infty$ assigns all weight to the most similar source model (i.e. `DISTILLNEAREST`). When relevant, we use the notation `DISTILLWEIGHTED(p)` to indicate the choice of p .

Scalability. For `DISTILLWEIGHTED` to be feasible, compared to `DISTILLNEAREST`, we need to ensure that the training procedure scales well with the size of \mathcal{S} . Since the computation of the weights $\{\alpha_s\}_{s=1}^S$ is based on the small probe set and is almost identical to the selection procedure for `DISTILLNEAREST` this is a negligible step. When training the target model, we merely require one forward pass on the unlabeled target dataset with each source model (to obtain pseudo-labels) as well as training of a one-layer classifier head per source model, both of which are cheap compared to the full training procedure of the target model. Nonetheless, one could employ a pre-selection of the top- k source models with the largest task similarity, thereby reducing the number of classifier heads and forward passes required. However, doing so introduces another hyperparameter, k , (i.e. how many models to use) complicating the analysis. Moreover, since large p induces such pre-selection in a *soft* manner, we leave it to future research to determine how to select the appropriate k .

C.5 Experiments and Results

C.5.1 Experimental Setup

Benchmark. Despite our methods being designed with the interest of using large vision models (that are potentially only available for inference), such a setting is intractable for our research. Thus, to allow for controlled experimentation we restrict our source models to a more tractable scale. In particular, we modify an existing transfer learning benchmark: Scalable Diverse Model Selection by [Bolya et al. \(2021\)](#), and use the publicly available models to construct a set of source models for each target task. Thus, we consider a set consisting of 28 models: 4 architectures (AlexNet, GoogLeNet, ResNet-18, and ResNet-50 ([Russakovsky et al., 2015](#); [He et al., 2016](#))) trained on 7 different source tasks (CIFAR-10, Caltech101, CUB200, NABird, Oxford Pets, Stanford Dogs, and VOC2007). For the target tasks, we consider 8 different tasks covering various image domains (Natural images: CIFAR-10, CUB200, NABird, Oxford Pets, Stanford Dogs; X-ray: ChestX; Skin Lesion Images: ISIC; Satellite Images: EuroSAT). We treat 20% of the samples as labeled, and the remaining as unlabeled. We carefully remove any source models associated with a particular target task, if such exists, in order to avoid information leakage between source and target tasks. For the target architecture, we use MobileNetV3 ([Howard et al., 2019](#)) due to its low computational requirements compared to any of the source models. Furthermore, unless otherwise mentioned $\lambda = 0.8$ and $p = 12$. We refer the reader to the supplementary material for further details.

Baselines. We consider a set of different baselines: based on ImageNet initializations we consider IN+TRANSFER (fine-tunes ImageNet representations using only the labeled data), and IN+FIXMATCH (Sohn et al., 2020) (fine-tunes the ImageNet representation using labeled and unlabeled data), and based on source model initializations we fine-tune the highest-ranked source model of each source architecture. To show the importance of using the right source model(s) to distill, we also compare DISTILLNEAREST to DISTILLRANDOMSELECTION which is the average of distilling from a randomly selected source, and for comparison to DISTILLWEIGHTED we also construct distilled models using the multi-source objective (C.4) with a random weight (DISTILLRANDOMWEIGHTS) and equal weights (DISTILLEQUAL). For ease of exposition, we present results for DISTILLNEAREST (Section C.5.2) and DISTILLWEIGHTED (Section C.5.3) in separate sections.

C.5.2 Results for DISTILLNEAREST

We compare DISTILLNEAREST with the baselines in Table C.1 and Figure C.3. Our observations are as follows.

Distillation with the right source model is better than fine-tuning from ImageNet.

We observe that within the same target architecture (MobileNetv3), simply fine-tuning ImageNet representations (IN+TRANSFER) is less optimal than distilling from the most similar single model (DISTILLNEAREST). In fact, for fine-grained datasets such as CUB200, NABird, Oxford Pets, and Stanford Dogs, we observe that distilling from an appropriate source model (DISTILLNEAREST) could yield much better performance than fine-tuning from a generalist ImageNet representation. More surprisingly, even with the aid of unlabeled data, models fine-tuned from ImageNet representations using a label propagation style approach (IN+FIXMATCH) still underperform distillation-based methods by at least 3.9% on average. These observations indicate the importance of selecting the right source model for transfer/distillation.

Distilling to efficient architecture could be better than fine-tuning larger models.

In Table C.1, we include the performance when fine-tuning larger architectures trained on ImageNet (IN+TRANSFER) and the source model (of the same architecture) most similar to each target task selected using PARC (FINE-TUNE SELECTED SOURCE). A few observations are immediate: (a) our choice of task similarity metric is effective for transfer; across all 4 architectures, we observe at least 4% improvement over simple fine-tuning from ImageNet, which validates the results by Bolya et al. (2021), and (b) with the aid of unlabeled data and distillation, the computationally efficient architecture MobileNetV3 can outperform larger architectures fine-tuned on labeled data from the target task (i.e. AlexNet, GoogLeNet, ResNet-18). Although underperforming fine-tuning a ResNet-50 initialized with the most similar ResNet-50 source model by a mere average of 2.5%-points (FINE-TUNE SELECTED SOURCE), using a ResNet-50 would require 17.5× more computations during inference to achieve such improvements.

Task Similarity Metrics for DISTILLNEAREST

One key component of DISTILLNEAREST is to select the source model to perform cross-task distillation on using task similarity metrics. Despite many existing metrics for

		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
Pseudo	CKA	0.72	0.62	0.23	0.39	-0.04	0.31	0.69	0.11	0.38
	PARC	0.79	0.79	0.02	0.17	0.06	0.48	0.72	0.54	0.45
	RSA	0.82	0.31	-0.11	0.30	0.10	-0.03	0.65	0.38	0.30
Feature	CKA	0.82	0.39	0.36	0.21	-0.04	0.47	0.69	0.55	0.43
	PARC	0.84	0.84	0.18	0.42	-0.14	0.81	0.81	0.84	0.58
	RSA	0.86	0.81	0.03	0.38	0.03	0.28	0.89	0.85	0.52

Table C.2: Spearman correlation between test accuracy after all possible single-source distillations and task similarities associated with the source models. Generally feature representations correlate better with distillation performance compared to pseudo-label representations.

quantifying task similarities, their effectiveness for distillation remains unclear. Given the myriads of metrics, we restrict our focus to metrics that can capture similarities between a source representation of a target example and its one-hot label representation. Along this vein, two questions arise: which metric to use for comparing representations, and which representations from a source model should be used to represent a target example?

For the first question, we look into multiple metrics in the literature that compare various representations: CKA (Cortes et al., 2012), RSA (Dwivedi and Roig, 2019), and PARC (Bolya et al., 2021). For the second question, we look into the common representations from a source model: the features ϕ and the probabilistic outputs $h \circ \phi$.

To establish the effectiveness of our choice of similarity metric, we report the Spearman correlation between the task similarities and the test accuracy of the distilled models in Table C.2. We see that features from the source models can better capture the correlation between the source models and the test accuracy of the distilled models, than the probabilistic pseudo-labels. In addition, we also see a much higher correlation among natural tasks (compared to specialized tasks such as ChestX, EuroSAT, and ISIC) which suggests that our choice of task similarity is effective at selecting similar tasks. Besides, we also observe a higher correlation when using PARC compared to the other metrics, thus validating our choice of using PARC as the default metric.

To further establish the effectiveness of our metrics to rank various source models, we compute the relative test accuracy between the top-3 models most similar to the target task and the top-3 best models after distillation (see Table C.3). Again, we observe that all three metrics are capable of ranking affinity between source models, but ranking the models with PARC outperforms the other two metrics.

C.5.3 Results for DISTILLWEIGHTED

From Table C.1, we observe that DISTILLWEIGHTED compares favorably to DISTILLNEAREST, thus the conclusions for DISTILLNEAREST translates to DISTILLWEIGHTED. Yet, one particular task, Oxford Pets, is worth more attention. On Oxford Pets (classification of different breeds of cats and dogs), we observe that distilling from multiple weighted sources (DISTILLWEIGHTED) is much better than distilling from the single most similar

		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
Pseudo	CKA	99.1	95.6	97.4	99.6	98.8	89.4	100.0	97.6	97.2
	PARC	99.5	100.0	95.5	99.6	98.5	99.7	98.8	99.7	98.9
	RSA	100.0	77.7	96.5	99.7	98.5	87.2	98.6	97.6	94.5
Feature	CKA	100.0	95.6	97.0	99.8	99.0	93.3	100.0	96.4	97.6
	PARC	100.0	100.0	97.8	99.7	98.3	100.0	97.1	98.5	98.9
	RSA	100.0	100.0	96.7	99.8	98.9	94.9	98.9	98.8	98.5

Table C.3: Relative accuracy of top-3 single-source distilled models selected by task similarity over the average of the 3 actual best models. We compute the average test accuracy of the top-3 highest ranked target models and divide it by the average of the 3 actually best-performing target models.

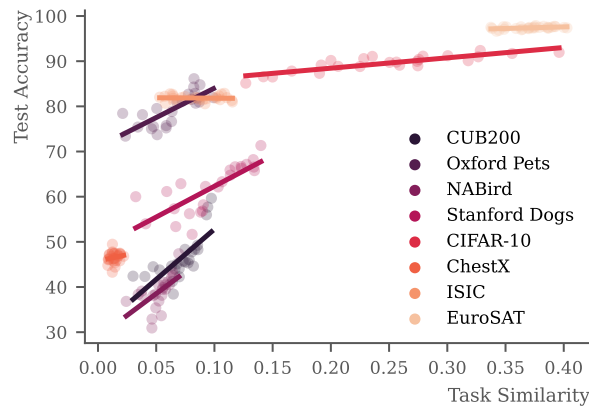


Figure C.4: Test accuracy of single-source distillation and raw task similarity score using PARC on the feature representations. The scores are on different scales for different tasks, but almost all tasks have a positive correlation between test accuracy and task similarity.

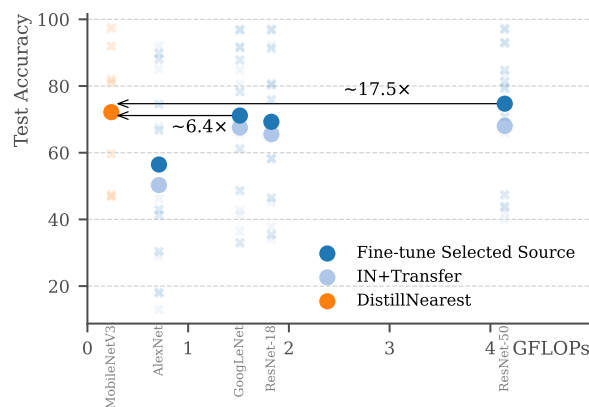


Figure C.5: Average test accuracy over the 8 target tasks vs. compute requirements for a single forward pass at inference. Using DISTILLNEAREST with an efficient target architecture outperforms (or is comparable to) fine-tuning larger models.

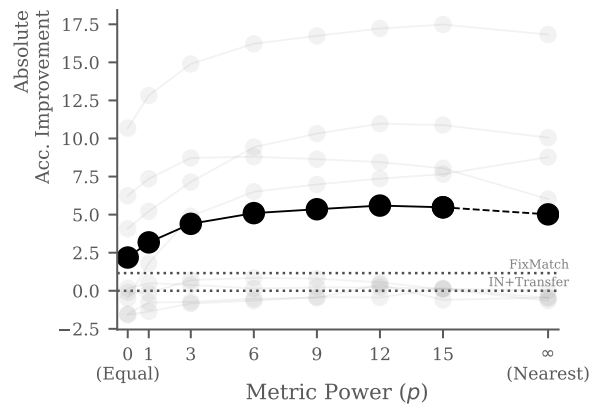


Figure C.6: Improvement over IN+TRANSFER. Here \bullet is the average improvement over all eight target tasks and \circ represents the performance on a target task. Note, $p = 0$ corresponds to DISTILLEQUAL, and $p = \infty$ corresponds to DISTILLNEAREST.

source (DISTILLNEAREST), which is a ResNet-18 trained on Caltech101 (that can recognize concepts such as Dalmatian dog, spotted cats, etc.). Although the most similar source model contains relevant information for recognizing different breeds of dogs and cats, it might not contain all relevant knowledge from the set of source models that could be conducive to recognizing all visual concepts in Oxford Pets. In fact, we observe that the second most similar model is a GoogLeNet model trained on Stanford Dogs to recognize more dog breeds than the most similar source model (but incapable of recognizing cats). In this case, DISTILLWEIGHTED allows aggregation of knowledge from multiple sources and can effectively combine knowledge from different source models for a more accurate target model than distillation from a single source. This suggests that *under certain conditions such as high heterogeneity in data, distilling from multiple source models can outperform distilling a single best source model.*

Task Similarity Metrics for Weighing Sources

We have established that our task similarity metric can capture the correlation between the source model representations and the test accuracy of the distilled models. However, it is not a priori clear that weighing source models based on the ranking of their affinity to the target task would yield better performance for multi-source distillation. As such, we investigate alternative choices of weighing schemes for a subset of 5 target tasks (CUB200, EuroSAT, ISIC, Oxford Pets, Stanford Dogs): INVERSE (weights are inversely proportional to task similarity), DISTILLRANDOMWEIGHTS (weights are sampled uniformly on a 4-simplex), DISTILLRANDOMSELECTION (randomly selecting a single source model), and DISTILLEQUAL (equal weights for all models).

Through Figure C.1, we find that distilling from a single or set of source models ranked using the similarity metric is much more effective than distilling from source models that are weighted randomly or equally (DISTILLRANDOMWEIGHTS or DISTILLEQUAL). In addition, the fact that INVERSE underperforms IN+TRANSFER on average suggests that it is crucial to follow the ranking induced by the similarity metrics when distilling the sources and that the metric ranks both the most similar source models and the least similar source models appropriately.

Effect of p

Our task similarity metrics give a good ranking of which source models to select for distillation but it is unclear whether the similarity score could be used directly without any post-processing. To investigate, we visualize the relationship between the test accuracy of the models distilled from a single source and our task similarity. From Figure C.4, it is clear that the distribution of task similarities depends on the target task, which motivates our normalization scheme.

In addition, it is not a priori clear that the weights should scale linearly with the similarity scores. Thus, we investigate the effect of the rescaling factor, p , for constructing the weights. In Figure C.6, we see that although no rescaling ($p = 1$) outperforms equal weighting, it is less optimal than e.g. $p = 12$ (our default). This suggests that task similarity and good weights have a monotonic, but non-linear relationship.

C.5.4 Additional Ablations and Analyses

Due to space constraints, we include additional ablations and analyses in the supplementary materials. We summarize the main findings as follows.

ResNet-50 as target model. Averaged over 8 tasks, `DISTILLWEIGHTED` outperforms both `IN+TRANSFER` and `DISTILLEQUAL` by 5.6% and 3.8%, respectively. Also, compared to ImageNet initialization, using `DISTILLWEIGHTED` with the most similar ResNet-50 source model as target model initialization improves accuracy by 1.0%.

Improvements on VTAB. `DISTILLWEIGHTED` outperforms `IN+TRANSFER` averaged over the ● *Natural* and ● *Specialized* tasks of VTAB, by 5.1% and 0.8%, respectively. `DISTILLNEAREST` outperforms by 4.8% and 0.6%, respectively.

Fewer labels. `DISTILLWEIGHTED` and `DISTILLNEAREST` outperform `IN+TRANSFER` (by 6.8% and 4.4%, respectively) under a setup with even fewer labeled samples.

Additional analysis of task similarity metrics. We consider additional correlation metrics and top- k relative accuracies of the selected models — all supporting the usefulness of task similarity to weigh and select source models.

C.6 Conclusion

We investigate the use of diverse source models to obtain efficient and accurate models for visual recognition with limited labeled data. In particular, we propose to distill multiple diverse source models from different domains weighted by their relevance to the target task without access to any source data. We show that under computational constraints and averaged over a diverse set of target tasks, our methods outperform both transfer learning from ImageNet initializations and state-of-the-art semi-supervised techniques.

Acknowledgments. This work was funded in part by NSF 2144117, and in part by Aarhus University Centre for Digitalisation, Big Data and Data Analytics (DIGIT).

Bibliography

- Abuduweili, A., Li, X., Shi, H., Xu, C. Z., and Dou, D. (2021). Adaptive Consistency Regularization for Semi-Supervised Transfer Learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6919–6928. Cited on page 100.
- Achille, A., Lam, M., Tewari, R., Ravichandran, A., Maji, S., Fowlkes, C., Soatto, S., and Perona, P. (2019). Task2Vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6429–6438. Cited on pages 101 and 103.
- Agostinelli, A., Uijlings, J., Mensink, T., and Ferrari, V. (2022). Transferability Metrics for Selecting Source Model Ensembles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7936–7946. Cited on pages 101 and 121.
- Ba, J. L. and Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems*, volume 27, pages 2654–2662. Cited on page 100.
- Bao, Y., Li, Y., Huang, S. L., Zhang, L., Zheng, L., Zamir, A., and Guibas, L. (2019). An Information-Theoretic Approach to Transferability in Task Transfer Learning. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 2309–2313. IEEE. Cited on page 101.
- Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. (2019). Mixmatch: A holistic approach to semi-supervised learning. In *Advances in neural information processing systems*, volume 32. Cited on page 100.
- Bisong, E. (2019). *Google Colaboratory*, pages 59–64. Apress, Berkeley, CA. Cited on page 98.
- Bolya, D., Mittapalli, R., and Hoffman, J. (2021). Scalable Diverse Model Selection for Accessible Transfer Learning. *arXiv preprint arXiv:2111.06977*. Cited on pages 101, 103, 105, 106, 107, 119, 122, and 123.
- Borup, K. and Andersen, L. N. (2021). Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation. In *Advances in Neural Information Processing Systems*, volume 34, pages 5316–5327. Cited on page 100.
- Cho, J. H. and Hariharan, B. (2019). On the Efficacy of Knowledge Distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Cited on page 100.
- Cortes, C., Mohri, M., and Rostamizadeh, A. (2012). Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828. Cited on pages 101, 107, and 122.
- Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M. P., Gritsenko, A., Birodkar, V., Vasconcelos, C., Tay, Y., Mensink, T.,

- Kolesnikov, A., Pavetić, F., Tran, D., Kipf, T., Lučić, M., Zhai, X., Keysers, D., Harmsen, J., and Houlsby, N. (2023). Scaling Vision Transformers to 22 Billion Parameters. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 7480–7512. Cited on pages 98.
- Dwivedi, K., Huang, J., Cichy, R. M., and Roig, G. (2020). Duality Diagram Similarity: A Generic Framework for Initialization Selection in Task Transfer Learning. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 497–513. Cited on page 101.
- Dwivedi, K. and Roig, G. (2019). Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 12379–12388. Cited on pages 101, 107, and 122.
- Fukuda, T., Suzuki, M., Kurata, G., Thomas, S., Cui, J., and Ramabhadran, B. (2017). Efficient knowledge distillation from an ensemble of teachers. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 3697–3701. Cited on page 100.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 770–778. Cited on page 105.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*. Cited on page 100.
- Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L. C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Le, Q., and Adam, H. (2019). Searching for mobileNetV3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324. Cited on page 105.
- Islam, A., Chen, C.-F. R., Panda, R., Karlinsky, L., Feris, R., and Radke, R. J. (2021). Dynamic Distillation Network for Cross-Domain Few-Shot Recognition with Unlabeled Data. In *Advances in Neural Information Processing Systems*, volume 34, pages 3584–3595. Cited on page 100.
- Jia, M., Tang, L., Chen, B. C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S. N. (2022). Visual Prompt Tuning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 709–727. Cited on page 124.
- Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big Transfer (BiT): General Visual Representation Learning. In *Proceedings of the 16th European Conference on Computer Vision*, pages 491–507. Cited on pages 98 and 124.
- Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of neural network representations revisited. In *36th International Conference on Machine Learning (ICML)*, pages 6156–6175. Cited on page 101.
- Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning (ICML)*, volume 3. Cited on page 100.

- Li, Z., Ravichandran, A., Fowlkes, C., Polito, M., Bhotika, R., and Soatto, S. (2021). Representation Consolidation for Training Expert Students. *arXiv preprint arXiv:2107.08039*. Cited on page 101.
- Liu, Y., Zhang, W., and Wang, J. (2020). Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113. Cited on page 100.
- Mirzadeh, S.-I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. (2020). Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198. Cited on page 100.
- Nguyen, C. V., Hassner, T., Seeger, M., and Archambeau, C. (2020). LEEP: A new measure to evaluate transferability of learned representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 7250–7261. PMLR. Cited on pages 101 and 103.
- Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational Knowledge Distillation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Cited on page 100.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Cited on page 100.
- Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019). Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415. Cited on pages 100.
- Peng, X., Li, Y., and Saenko, K. (2020). Domain2Vec: Domain Embedding for Unsupervised Domain Adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 756–774, Cham. Springer International Publishing. Cited on page 101.
- Phoo, C. P. and Hariharan, B. (2021). Self-training For Few-shot Transfer Across Extreme Task Differences. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Cited on page 100.
- Rekognition, A. (2023). Rekognition. Cited on page 98.
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). FitNets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. Cited on page 100.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 211–252. Cited on page 105.
- Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. (2020). FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *Advances in Neural Information Processing Systems*, volume 33, pages 596–608. Cited on pages 100, 103, and 106.

- Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852. Cited on page 98.
- Tan, X., Ren, Y., He, D., Qin, T., Zhao, Z., and Liu, T. Y. (2019). Multilingual neural machine translation with knowledge distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on page 100.
- Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, volume 30, pages 1196–1205. Cited on page 100.
- Tran, A., Nguyen, C., and Hassner, T. (2019). Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1395–1405. Cited on page 101.
- Wallace, B., Wu, Z., and Hariharan, B. (2021). Can We Characterize Tasks Without Labels or Features? In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1245–1254. Cited on page 101.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45. Cited on page 100.
- Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2020). Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Cited on page 100.
- Xu, R., Chen, Z., Zuo, W., Yan, J., and Lin, L. (2018). Deep Cocktail Network: Multi-source Unsupervised Domain Adaptation with Category Shift. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3964–3973. Cited on page 100.
- You, S., Xu, C., Xu, C., and Tao, D. (2017). Learning from Multiple Teacher Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*, pages 1285–1294. Cited on page 100.
- Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2022). Scaling Vision Transformers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 12094–12103. Cited on pages 98.
- Zhai, X., Puigcerver, J., Kolesnikov, A., Ruysen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., Beyer, L., Bachem, O., Tschannen, M., Michalski, M., Bousquet, O., Gelly, S., and Houlsby, N. (2019). A Large-scale Study of Representation Learning with the Visual Task Adaptation Benchmark. *arXiv preprint arXiv:1910.04867*. Cited on pages 116 and 124.

- Zhao, H., Zhang, S., Wu, G., Costeira, J. P., Moura, J. M., and Gordon, G. J. (2018). Multiple source domain adaptation with adversarial learning. In *Workshop proceedings of the 6th International Conference on Learning Representations (ICLR)*. Cited on page 100.
- Zhao, S., Wang, G., Zhang, S., Gu, Y., Li, Y., Song, Z., Xu, P., Hu, R., Chai, H., and Keutzer, K. (2020). Multi-source distilling domain adaptation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, volume 34, pages 12975–12983. Cited on page 100.

Supplementary material

C.I Additional Ablations and Analyses

C.I.1 Results on VTAB

We report the results of our VTAB (Zhai et al., 2019) experiment in Table C.4. On VTAB, We find that both `DISTILLWEIGHTED` and `DISTILLNEAREST` distillation outperform `IN+TRANSFER` on each of the ● *Natural* tasks. Particularly, `DISTILLWEIGHTED` outperforms `IN+TRANSFER` with 13.9%-points on CIFAR-10 and 10.6%-points on Sun397 and averaged across ● *Natural* `DISTILLWEIGHTED` outperforms `IN+TRANSFER` with 5.1%-points. Average over ● *Specialized* both `DISTILLWEIGHTED` and `DISTILLNEAREST` outperform `IN+TRANSFER`, although with a small margin. Finally, averaged over ● *Structured* `IN+TRANSFER` outperforms our methods, but due to the nature of these tasks, we do not expect source models to transfer well to these tasks.¹ Yet, we still obtain the best accuracy on DMLab, dSpr-Loc, and sNORB-Azimuth.

	● Caltech101	● CIFAR-100	● DTD	● Flowers102	● Pets	● SVHN	● Sun397	● Natural	● Camelyon	● EuroSAT	● Resisc45	● Retinopathy	● Specialized	● Clevr-Count	● Clevr-Dist	● DMLab	● KITTI-Dist	● dSpr-Loc	● dSpr-Ori	● sNORB-Azimuth	● sNORB-Elev	● Structured	● Mean
<code>IN+Transfer</code>	88.1	47.0	57.4	85.8	82.8	75.3	27.8	66.3	81.0	95.0	80.0	72.7	82.2	73.1	55.9	43.6	75.7	18.7	58.6	21.2	46.0	49.1	62.4
<code>DISTILLWEIGHTED</code>	88.6	60.9	62.4	86.1	84.4	79.0	38.4	71.4	80.6	95.9	83.3	72.2	83.0	57.4	45.6	44.6	67.7	27.4	44.9	23.9	38.2	43.7	62.2
<code>DISTILLNEAREST</code>	88.9	59.5	61.9	86.2	84.5	79.5	37.6	71.1	80.5	95.8	83.2	71.7	82.8	60.5	45.4	45.2	67.9	20.8	40.6	24.2	36.5	42.6	61.6

Table C.4: Top-1 accuracy by dataset in VTAB. The accuracy for each task is in grey, and the average accuracy for each category of tasks is in black. Note, the ● *Mean* is the average across all tasks, not categories. The largest value in each column is marked in bold. Here `DISTILLWEIGHTED` is with $p = 9$.

C.I.2 Relative Accuracy of Single-source Distillation

Similarly to Table C.3, we extend our evaluation of how well the task similarity selects the best source models for single-source distillation. We report the ratio between the average test accuracy of the top- k target models ranked using the task similarity and the average test accuracy for the actual top- k target models found after the fact in Table C.5, Table C.6, and Table C.7 for $k = 1$, $k = 3$, and $k = 5$, respectively.

We find that generally, using task similarity on feature representations rather than the corresponding pseudo-labels yields better rankings, but also that PARC shows very little difference between features and pseudo-labels for all considered $k \in \{1, 3, 5\}$.

Relative accuracy over all k . The relative accuracy measure reported above is sensitive to k and the actual accuracy values of the models. That is, if a metric flips the order of the best and second best model when there is a notable performance gap between the two models, the relative accuracy for $k = 1$ will be low, and we might be mistaken to believe the metric is not working well. However, the metric might rank every model for

¹ The ● *Structured* tasks are mainly (ordinal) regression tasks transformed into classification tasks, and thus it seems reasonable to expect very general features (such as those from an ImageNet pre-trained model) to generalize better to such constructed tasks than specialized source models.

$k > 2$ perfectly correct, and since we typically utilize the full set of source models, the initial mistake should not be detrimental to the selection of the task similarity metric. Thus, in Figure C.7 we plot the relative accuracy for each task similarity metric and all $k \in \{1, \dots, S\}$. We find that while PARC on feature representations is outperformed by both PARC and CKA on pseudo-labels for $k < 3$, PARC on feature representations outperforms all the other metrics for $k \geq 3$. In particular, from Table C.8 we have that on average over all $k < S$, PARC, performs the best.

		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
Pseudo	CKA	99.6	100.0	96.1	99.5	98.1	100.0	100.0	100.0	99.2
	PARC	99.3	100.0	93.6	99.5	98.3	100.0	98.4	100.0	98.6
	RSA	99.3	74.8	94.8	99.5	98.3	86.6	97.8	95.6	93.4
Feature	CKA	99.6	81.0	92.6	99.8	98.3	100.0	100.0	100.0	96.4
	PARC	99.6	100.0	94.6	99.5	97.7	100.0	95.1	100.0	98.3
	RSA	99.6	100.0	92.6	99.5	98.3	80.6	100.0	100.0	96.3

Table C.5: Relative accuracy of top-1 single-source distilled target model selected by task similarity over the best model found in hindsight. We compute the test accuracy of the highest-ranked target model (ranked by some task similarity) and divide this by the test accuracy of the best-performing target model.

		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
Pseudo	CKA	99.1	95.6	97.4	99.6	98.8	89.4	100.0	97.6	97.2
	PARC	99.5	100.0	95.5	99.6	98.5	99.7	98.8	99.7	98.9
	RSA	100.0	77.7	96.5	99.7	98.5	87.2	98.6	97.6	94.5
Feature	CKA	100.0	95.6	97.0	99.8	99.0	93.3	100.0	96.4	97.6
	PARC	100.0	100.0	97.8	99.7	98.3	100.0	97.1	98.5	98.9
	RSA	100.0	100.0	96.7	99.8	98.9	94.9	98.9	98.8	98.5

Table C.6: (Identical to Table C.3) Relative accuracy of top-3 single-source distilled target models selected by task similarity over the average of the 3 best models found in hindsight. We compute the average test accuracy of the top-3 highest ranked target models and divide this average by the average test accuracy of the 3 best-performing target models.

C.I.3 Ablation of p for DISTILLWEIGHTED

We report the values associated with Figure C.6 for each target task and all considered choices of p in Table C.9.

C.I.4 DISTILLWEIGHTED with ResNet-50 as Target Architecture

In the main part of the article, we consider the computationally constrained setting, where some compute budget restricts the possible size of our target model. Thus, we use MobileNetV3 models as target models throughout the main paper. However, in

		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
Pseudo	CKA	99.3	98.7	98.3	99.7	99.0	92.9	99.2	98.4	98.2
	PARC	99.7	100.0	96.7	99.7	98.9	94.5	99.4	98.4	98.4
	RSA	99.7	83.2	97.6	99.8	99.0	84.9	99.2	92.8	94.5
Feature	CKA	99.7	97.4	97.7	99.8	98.9	96.5	99.2	97.8	98.4
	PARC	99.7	100.0	97.9	99.8	99.1	99.7	97.5	99.7	99.2
	RSA	99.7	99.7	97.9	99.8	99.2	97.9	98.9	99.7	99.1

Table C.7: Relative accuracy of top-5 single-source distilled target models selected by task similarity over the average of the 5 best models found in hindsight. We compute the results analogously to Table C.6 with $k = 5$.

	CKA	PARC	RSA
Pseudo	0.985	0.990	0.974
Feature	0.986	0.993	0.991

Table C.8: The mean relative accuracy, across all k , for each metric in Figure C.7. The average is bounded in $(0,1]$, and 1 corresponds to perfect ordering by task similarity. We find that using feature representations consistently outperforms pseudo-labels and that for both feature representations and pseudo-labels PARC performs the best.

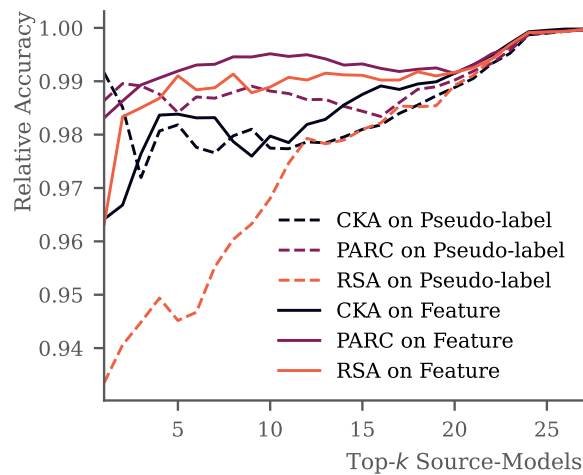


Figure C.7: Relative accuracy of top- k single-source distilled target models selected by task similarity over the average of the top- k actual best target models found in hindsight. If the ordering by task similarity were perfectly correct, the relative accuracy would be 1 for all k . See Table C.8 for the average of each metric across all k .

	CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
IN+Transfer	92.4	42.8	47.3	97.4	81.6	37.3	75.9	62.6	67.2
IN+FixMatch	93.5	41.9	38.5	98.1	82.6	42.8	83.4	65.8	68.3
DISTILLEQUAL	90.8	53.5	45.7	97.5	81.5	41.4	82.1	62.1	69.3
DISTILLWEIGHTED(1)	91.1	55.6	46.5	97.9	81.5	42.5	83.3	64.4	70.3
DISTILLWEIGHTED(3)	91.6	57.7	46.5	97.7	82.3	44.5	84.6	67.4	71.6
DISTILLWEIGHTED(6)	91.8	59.0	46.7	97.5	82.5	46.7	84.7	69.1	72.3
DISTILLWEIGHTED(9)	92.0	59.6	46.8	97.6	82.4	47.6	84.5	69.5	72.5
DISTILLWEIGHTED(12)	92.0	60.0	47.7	97.6	82.2	48.3	84.4	69.9	72.8
DISTILLWEIGHTED(15)	92.6	60.3	46.7	97.5	81.7	48.2	83.9	70.2	72.6
DISTILLNEAREST	92.0	59.6	46.8	97.4	81.0	47.4	81.9	71.3	72.2

Table C.9: Test accuracy of DISTILLWEIGHTED with various choices of p , compared to the baseline methods of IN+TRANSFER and IN+FIXMATCH. We highlight the largest value for each target task in **bold** and shade our default options. The results are also visualized in Figure C.6.

	Model Init.	CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
IN+Transfer	ImageNet	92.9	42.0	43.4	96.8	79.9	39.9	83.3	65.9	68.0
Fine-tune Source	Source	93.0	70.8	43.9	97.2	81.3	47.4	84.8	79.3	74.7
DISTILLEQUAL	ImageNet	87.8	57.3	46.1	97.0	78.9	42.4	84.1	64.5	69.8
DISTILLWEIGHTED(12)	ImageNet	91.5	64.5	45.4	97.0	78.9	49.8	87.1	74.2	73.6
DISTILLEQUAL	Source	87.5	68.8	45.5	97.4	81.2	43.2	81.9	65.1	71.3
DISTILLWEIGHTED(12)	Source	91.6	70.0	47.6	97.0	80.8	50.0	85.7	73.8	74.6

Table C.10: DISTILLWEIGHTED with ResNet-50 as target model architecture. We compare fine-tuning of the highest ranked source model (Bolya et al., 2021) with DISTILLWEIGHTED to both ImageNet-initialized target models and target models initialized from the highest ranked ResNet-50 source model. For $p = 12$, DISTILLWEIGHTED performs on par with fine-tuning the selected source model. The largest value for each target task is in **bold**.

Table C.10 we remove the computational budget and allow the target model to be of any architecture, and particularly we use a ResNet-50 as the target model.

We compare DISTILLWEIGHTED (with $p = 0$ and $p = 12$) initialized with either ImageNet pre-trained weights or the weights of the highest ranked ResNet-50 source model to IN+TRANSFER and FINE-TUNE SELECTED SOURCE. We find that DISTILLWEIGHTED initialized from ImageNet outperforms IN+TRANSFER on average for both equal weighting and $p = 12$, but underperforms FINE-TUNE SELECTED SOURCE for both p . However, since FINE-TUNE SELECTED SOURCE is initialized from well-selected source model weights, the comparison is not entirely fair. Thus, we also consider the case where we initialize the target model for DISTILLWEIGHTED with the weights of the highest ranked ResNet-50 source model, and find that for $p = 12$ DISTILLWEIGHTED performs on par with FINE-TUNE SELECTED SOURCE.

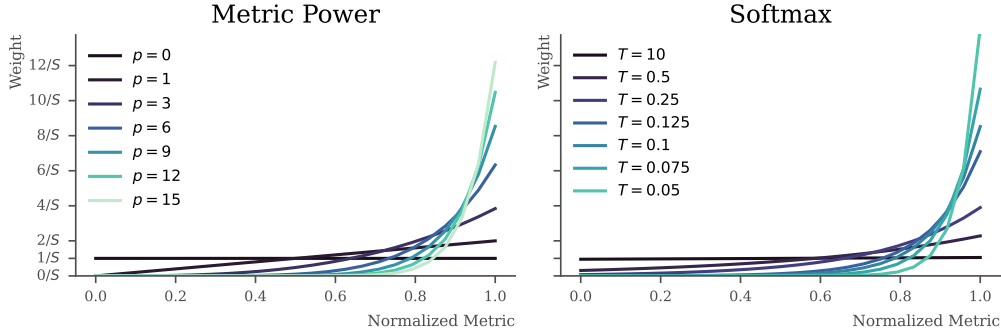


Figure C.8: Transformation of weights for various choices of power (left) or softmax temperature (right). Here S is the number of source models, and we consider equidistantly distributed normalized metrics.

C.I.5 Normalization of Task Similarity for Source Model Weighting

We propose to choose the weights $\alpha = (\alpha_1, \dots, \alpha_S)$ as

$$\alpha_i = \frac{e_i^p}{\sum_{s=1}^S e_s^p}, \quad \text{where } e_j = \mathbb{1}_{(e_j > 0)} e_j$$

for $j = 1, \dots, S$, and e_s is the task similarity for source model \mathcal{M}_s , evaluated on the target task, normalized to satisfy $e_s \in [0, 1]$ with min-max normalization over all e_s . Here, the hyperparameter, p can be used to increase/decrease the relative weight on the highest ranked source models, with the extremes $p = 0$ and $p \rightarrow \infty$ corresponding to equal weight and single-source distillation, respectively. An alternative way to obtain our normalization is to use the softmax function on the task similarities,

$$\alpha_i = \frac{\exp\left(\frac{e_i}{T}\right)}{\sum_{s=1}^S \exp\left(\frac{e_s}{T}\right)}.$$

This does not require clipping the task similarity at 0, and with the temperature, T , we can adjust the relative weight on particular source models. Here, large T flattens the weights, and $T \rightarrow \infty$ corresponds to an equal weighting of all source models, while small T increases the weight on the highest-ranked source models. Quantitatively, the two normalization methods can yield similar transformations with appropriate choices of p and T — see Figure C.8.

C.I.6 Smaller Amount of Labeled Data

We now repeat the experiment of the main paper across the 8 target datasets with a reduced amount of labeled samples. Here, we reduce the number of labeled samples to 5% (rather than 20%) of the training set and report the accuracy in Table C.11. We find a similar pattern as observed in the main experiment, where `DISTILLWEIGHTED` distillation on average outperforms `IN+TRANSFER` irrespective of the choice of p . For $p = 9$ `DISTILLWEIGHTED` outperforms `IN+TRANSFER` by 6.8%-point on average and in particular 15.5%-points on CUB200, whereas the only loss in performance is on ChestX with a drop of 0.9%-point.

	CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
IN+Transfer	88.0	16.8	43.5	94.8	73.9	14.4	55.0	38.9	53.2
DISTILLWEIGHTED(1)	88.1	29.2	42.3	95.9	76.3	20.5	66.6	42.1	57.6
DISTILLWEIGHTED(9)	90.2	32.3	42.6	95.9	76.7	24.8	68.2	49.0	60.0
DISTILLNEAREST	87.2	31.4	39.7	95.1	75.4	24.0	58.9	49.7	57.7

Table C.11: Distillation on the eight target tasks with only 5% labeled samples per task. Again, we compare to the baseline of IN+TRANSFER. The largest value for each target task is in **bold**.

C.I.7 Different Measures of Correlation

In order to evaluate the quality of a task similarity metric to estimate the performance of a target model after distillation, we consider the correlation between the computed metric and the actual observed performance after distillation. However, since we have no reason to believe that the relationship is linear, we consider the Spearman correlation in the main paper. However, for completeness of exposition, we report Pearson correlation and Kendall’s Tau in Table C.12 and Table C.13, respectively. For both these correlation measures, the overall conclusions are the same: Using feature representations is preferable to pseudo-labels, and PARC generally outperforms both CKA and RSA, albeit not by much over CKA.

		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
Pseudo	CKA	0.62	0.85	0.07	0.30	-0.06	0.33	0.67	0.21	0.37
	PARC	0.75	0.74	-0.03	0.27	-0.00	0.36	0.63	0.51	0.40
	RSA	0.75	0.13	-0.07	0.38	0.04	-0.09	0.66	0.40	0.27
Feature	CKA	0.84	0.60	0.39	0.29	0.00	0.30	0.71	0.54	0.46
	PARC	0.86	0.73	0.17	0.46	-0.06	0.58	0.77	0.78	0.54
	RSA	0.90	0.85	0.07	0.45	0.04	0.27	0.87	0.83	0.54

Table C.12: Pearson correlation between test accuracy after all possible single-source distillations and task similarity associated with the source models. Similar to Table C.2.

C.I.8 Choice of Task Similarity Metrics

Recently, multiple measures intended to estimate the transferability of a source model have been proposed. However, despite the very recently published Multi-Source Leep (MS-LEEP) and Ensemble Leep (E-Leep) no task similarity metric considers the estimation over multiple models at once (Agostinelli et al., 2022). Thus, we consider each source model separately and compute the metrics independent of other source models. This has the added benefit of reducing the number of metric computations required as we do not need to compute the task similarity for all possible combinations of n models from S possible (i.e. $\binom{n}{S}$), which grows fast with S .

		CIFAR-10	CUB200	ChestX	EuroSAT	ISIC	NABird	Oxford Pets	Stanford Dogs	Mean
Pseudo	CKA	0.51	0.46	0.16	0.28	-0.05	0.24	0.49	0.07	0.27
	PARC	0.61	0.64	0.01	0.12	0.02	0.36	0.54	0.39	0.34
	RSA	0.62	0.17	-0.07	0.22	0.08	-0.01	0.48	0.29	0.22
Feature	CKA	0.67	0.34	0.25	0.14	-0.05	0.40	0.50	0.38	0.33
	PARC	0.69	0.67	0.14	0.31	-0.10	0.65	0.62	0.67	0.46
	RSA	0.72	0.65	0.02	0.28	0.02	0.19	0.72	0.67	0.41

Table C.13: Kendall Tau correlation between test accuracy after all possible single-source distillations and task similarity associated with the source models. Similar to Table C.2.

Assume $\mathbf{X} \in \mathbb{R}^{N \times d_x}$ and $\mathbf{Y} \in \mathbb{R}^{N \times d_y}$, and that $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for and $\mathbf{L}_{ij} = l(\mathbf{y}_i, \mathbf{y}_j)$ where k , and l are two (similarity) kernels as well as $\mathbf{x}_i, \mathbf{x}_j$ and $\mathbf{y}_i, \mathbf{y}_j$ are rows of \mathbf{X} and \mathbf{Y} , respectively. Then we have that CKA is defined as

$$\rho_{\text{CKA}}(\mathbf{X}, \mathbf{Y}) \stackrel{\text{def}}{=} \frac{\text{HSIC}(\mathbf{K}, \mathbf{L})}{\sqrt{\text{HSIC}(\mathbf{K}, \mathbf{K})\text{HSIC}(\mathbf{L}, \mathbf{L})}},$$

where $\mathbf{K}, \mathbf{L} \in \mathbb{R}^{N \times N}$ and HSIC is the Hilbert-Schmidt Independence Criterion,

$$\text{HSIC}(\mathbf{K}, \mathbf{L}) \stackrel{\text{def}}{=} \frac{1}{(N-1)^2} \text{tr}(\mathbf{K}\mathbf{H}_N\mathbf{L}\mathbf{H}_N), \quad \text{with}$$

$$\mathbf{H}_N \stackrel{\text{def}}{=} \mathbf{I}_N - \frac{1}{N}\mathbf{1}\mathbf{1}^\top.$$

In particular, if both k and l are linear kernels, then

$$\rho_{\text{CKA}}(\mathbf{X}, \mathbf{Y}) = \frac{\|\mathbf{Y}^\top \mathbf{X}\|_F^2}{\|\mathbf{X}^\top \mathbf{X}\|_F \|\mathbf{Y}^\top \mathbf{Y}\|_F},$$

where $\|\cdot\|_F$ is the Frobenius norm. We use the linear kernel throughout this paper and refer to [Cortes et al. \(2012\)](#) for additional details on CKA.

For RSA, we consider the dissimilarity matrices given by

$$\mathbf{K}_{ij} \stackrel{\text{def}}{=} 1 - \text{pearson}(\mathbf{x}_i, \mathbf{x}_j) \quad \text{and}$$

$$\mathbf{L}_{ij} \stackrel{\text{def}}{=} 1 - \text{pearson}(\mathbf{y}_i, \mathbf{y}_j),$$

where \mathbf{X} and \mathbf{Y} are assumed normalized to have mean 0 and variance 1. We then compute RSA as the Spearman correlation between the lower triangles of \mathbf{K} and \mathbf{L} ,

$$\rho_{\text{RSA}}(\mathbf{X}, \mathbf{Y}) \stackrel{\text{def}}{=} \text{spearman}(\{\mathbf{K}_{ij} \mid i < j\}, \{\mathbf{L}_{ij} \mid i < j\}).$$

For additional details on RSA, we refer the reader to [Dwivedi and Roig \(2019\)](#). While [Bolya et al. \(2021\)](#) introduces PARC alongside a heuristic and feature reduction, the PARC metric is almost identical to RSA. However, RSA was introduced to compute similarities between two sets of representations, and PARC was aimed at computing similarities between a set of representations and a set of labels associated with the dataset. Thus, in our use of PARC, it merely differs from RSA in the lack of normalization of \mathbf{Y} , which is assumed to be one-hot encoded vectors of class labels from the probe dataset.

C.I.9 Compute Requirements and Scalability

For `DISTILLNEAREST` and `DISTILLWEIGHTED` to be feasible in practice, we need to ensure that the computational costs of training and inference for both methods are reasonable and that it scales well with the size of \mathcal{S} .

Inference. We note, that while both `DISTILLNEAREST` and `DISTILLWEIGHTED` use additional classifier head(s) during training, these are discarded at inference time, and no additional compute overhead remains. Thus, memory and compute requirements at inference time are identical to those of the original target model, and thereby the equivalent target model trained supervised.

Training. We can separate the training procedure into two phases; a) estimation of task similarity metrics, and b) training of the target model with `DISTILLNEAREST` or `DISTILLWEIGHTED`. The majority of compute is typically needed for b) as is expected for the training of neural networks. For a) we estimate the task similarity metric for a single model based on the small annotated probe set (we use 500 samples). The computation of the metric itself is dominated by the forward pass, which is a single forward pass on each of the 500 samples, thus corresponding to less compute than 4 batches of training, where we typically train for thousands of batches. We thus consider phase a) as negligible as it is also reusable across multiple experiments for the target task. Furthermore, for b) we use additional compute in two parts of the training; 1) training of an additional classifier head per source model used (thus 1 for `DISTILLNEAREST` and $|\mathcal{S}|$ for `DISTILLWEIGHTED`), and 2) obtaining pseudo-labels for the unlabeled data. We note that for 224×224 inputs, a `MobileNetV3` uses 0.24 GFLOPs as default, and each additional classifier head requires an additional approx. 0.0013 GFLOPs (for 1000 classes). Thus, we can attach about 3000 classifier heads (and thereby 3000 source models) to a `MobileNetV3` before we require the same GFLOPs as a `ResNet-50`. Regarding 2), the pseudo-labels are obtained by a single forward pass by each source model over the unlabeled data. For very large source models, this can potentially be expensive, but the pseudo-labels can be reused across multiple experiments for the same target task. However, the computational requirements for this step highly depend on the set of source models \mathcal{S} , and is thereby hard to quantify.

C.II Experimental Details

In the following, we provide some experimental details.

C.II.1 Main Experiments

Unless otherwise mentioned, we use SGD with a learning rate of 0.01, weight decay of 0.0001, batch size of 128, and loss weighting of $\lambda = 0.8$. We initialize our target models with the ImageNet pre-trained weights available in torchvision (<https://pytorch.org/vision/stable/models>) and consider 28 fine-tuned models from Bolya et al. (2021) publicly available at <https://github.com/dbolya/parc> as our set of source models. The source models consist of each of the architectures (AlexNet, GoogLeNet, ResNet-18, and ResNet-50) trained on CIFAR-10, Caltech101, CUB200, NABird, Oxford Pets, Stanford Dogs, and VOC2007. Note, we always exclude any source model trained on the particular

target task, thus effectively reducing the number of source models for some target tasks. For FixMatch we use a batch size of 128 (with a 1:1 ratio of labeled to unlabeled samples for each batch) and fix the confidence threshold at 0.95 and the temperature at 1. We keep the loss weighting between the supervised loss and the unlabeled FixMatch loss at $\lambda = 0.8$.

C.II.2 VTAB Experiments

For each VTAB experiment, we consider the full training set (as introduced in [Zhai et al. \(2019\)](#)) as the unlabeled set, \mathcal{D}_τ^u , and the VTAB-1K subset as the labeled set, \mathcal{D}_τ^l . We use the Pytorch implementation from [Jia et al. \(2022\)](#) available at <https://github.com/KMnP/vpt>.

We use SGD with a learning rate of 0.005, weight decay of 0.0001, batch size of 128 equally split in 64 labeled and unlabeled samples, and loss weighting of $\lambda = 0.9$. We train our models for 100 epochs, where we define one epoch as the number of steps required to traverse the set of unlabeled target data, \mathcal{D}_τ^u when using semi-supervised methods, or merely as the number of steps to traverse the labeled set, \mathcal{D}_τ^l , for supervised transfer methods. We initialize our target models with the BiT-M ResNet-50x1 model fine-tuned on ILSVRC-2012 from BiT ([Kolesnikov et al., 2020](#)) publicly available at https://github.com/google-research/big_transfer.

We consider the 19 BiT-M ResNet-50x1 models fine-tuned on the VTAB-1K target tasks from [Kolesnikov et al. \(2020\)](#) as the set of source models. We always exclude the source model associated with the target task from the set of source models, and thus effectively have 18 source models available for each target task in VTAB. We use the PARC metric on the source model features to compute the source weighting, but also only use the top-5 highest-ranked source models to reduce the computational costs of training. Furthermore, we use $p = 9$ for DISTILLWEIGHTED.

C.III Domain Gap Between Source Tasks, Targets Tasks and ImageNet

As is evident from Figure C.3 and Table C.1, both DISTILLNEAREST and DISTILLWEIGHTED do not yield notable improvements on e.g. ChestX and ISIC, but yield significant improvements on e.g. CUB200 and Oxford Pets. Notably, for the latter target tasks there are semantically similar source tasks present in our set of source models, while this is not true for the former target tasks. Hence, as one would expect, the availability of a source model trained on source tasks similar to the target tasks is important for cross-domain distillation to work well, which is expected to be true for both DISTILLNEAREST and DISTILLWEIGHTED. Indeed, the task similarity metrics considered in this paper all aim at measuring alignment between tasks, and if the alignment between source and target tasks is small, we do not expect to gain much from distillation. This is affirmed by our experiments in e.g. Table C.1.

C.III.1 A Note on Potential Data Overlap Between Source and Target Tasks

Whenever any type of transfer learning is applied, including using ImageNet initializations, we (often implicitly) assume that the model we transfer from has not been trained on any data from the target test set. Although this assumption is often satisfied in practice due to domain gaps between the source and target task, utilizing initializations trained on e.g. ImageNet can potentially violate the assumption. This is due to the fact that ImageNet and many other modern publicly available datasets are gathered from various public websites and overlaps between samples in different datasets might occur.

Thus, it is natural to question whether the observed improvements are due to methodological advances or information leakage between source and target tasks. To ensure our advancements are valid we carefully remove any source model associated with the target task from the set of source models, \mathcal{S} . However, information leakage might still appear if e.g. there are identical samples in the target dataset and the source dataset or ImageNet. Despite large overlaps being improbable, it has been shown that there e.g. is a minor overlap (of at least 43 samples) between the training set of ImageNet and the test set of CUB200 (see e.g. <https://gist.github.com/arunmallya/a6889f151483dcb348fa70523cb4f578>). However, since the test set of CUB200 consists of 5794 samples, the presence of such a minor overlap should not affect the true performance of a model much.

In our experiments, we consistently compare our target models (initialized with ImageNet weights) to either identically initialized target models or source models initialized with either ImageNet weights or with weights from a source task. Hence, any potential gain from information leakage between ImageNet and a target task would bias both our results and the baselines, and thereby not affect our overall results. Furthermore, while an overlap between a source and target task might unfairly benefit the performance of our methods compared to IN+TRANSFER and IN+FixMATCH, such an overlap would likely benefit the fine-tuned source models even more making this baseline even harder to outperform (see e.g. Figure C.5 and Table C.1). Thus, our results should be at most as biased as the baselines.

A Quest for Perfect Teacher-Student Agreement in Knowledge Distillation

WORKING PAPER

Kenneth Borup
Aarhus University

Abstract. Knowledge distillation has shown great success at training a small *student* model on the predictions of a larger *teacher* model, thereby achieving improved generalization performance of the student. However, while it is commonly expected that knowledge distillation works by the student learning to match the predictions of the teacher, this claim has yet to be confirmed. In fact, it has been shown that even a student with sufficient capacity to match a known optimal solution, often fails to do so.

We investigate the effect of different implicit design choices, with the aim of obtaining perfect teacher-student agreement. Thus, while we do not aim to propose a new distillation technique, we strive to identify the effect of particular choices of hyperparameters and settings on the effectiveness of knowledge distillation. We do this through empirical experiments, and amongst others find that the teacher-student agreement sometimes negatively correlates with the generalization performance. Furthermore, we confirm the hypothesis that excessively long training schedules are needed to obtain perfect agreement in some cases.

D.1 Introduction

Currently, many areas of computer vision are dominated by large-scale vision models trained on huge datasets (Kolesnikov et al., 2020; Sun et al., 2017; Zhai et al., 2022). Thus, state-of-the-art models push the limits of current hardware for classification, object detection, and semantic segmentation tasks. Yet, such computationally demanding models are often infeasible for deployment in practice due to high computational costs. Thus, state-of-the-art performance improvements often do not translate into practical real-world applications. Two widely used paradigms attempting to alleviate the gap between research and applications are model pruning and knowledge distillation. Both start with a well-performing large model and aim at reducing the computational requirements by reducing the model size. Model pruning achieves this by selectively pruning away parts of the model (Aghli and Ribeiro, 2021; Anwar et al., 2017; Molchanov et al., 2016; Frankle and Carbin, 2019). However, this restricts the final model to be an altered version of the original architecture and can be challenging due to internal dependencies between e.g. weights and normalization statistics. Knowledge distillation is more flexible, avoids these inconveniences, and allows nearly any architecture to be used as the final model (Hinton et al., 2015).

The idea of knowledge distillation is to *distill* information from a large cumbersome and pre-trained *teacher* model into a smaller and more efficient *student* model. Recent research has proposed various notions of information to transfer from the teacher to the student model (Park et al., 2019; Srinivas and Fleuret, 2018; Zagoruyko and Komodakis, 2017; Romero et al., 2015; Yim et al., 2017), but we restrict ourselves to one of the conventional formulations by Hinton et al. (2015) and Ba and Caruana (2014). Despite the large empirical success of this simple approach, a thorough understanding of how, why, and when this procedure works is still largely lacking.

A widespread, yet largely unsupported, expectation for the empirical successes of a multitude of distillation techniques is that a student model trained with distillation learns to mimic the predictions of the teacher model (Bucila et al., 2006; Hinton et al., 2015; Gotmare et al., 2019; Tang et al., 2020; Dong et al., 2019). However, evidence of such ability is lacking at large, and merely little research investigates this claim. Two branches of research attempt to answer how, when, and why knowledge distillation works. One branch investigates knowledge distillation with theoretical mathematical rigor under somewhat strong assumptions (Phuong and Lampert, 2019; Mobahi et al., 2020; Borup and Andersen, 2021, 2023), and another branch performs empirical analyses of knowledge distillation (Stanton et al., 2021; Beyer et al., 2022). In particular, for the latter, Stanton et al. (2021) explicitly investigates the agreement between the teacher and student, while Beyer et al. (2022) finds that excessively long training durations and very strong augmentation schemes are necessary for a student to accurately match a teacher. In this paper, we delve into the question;

“Under which conditions can we obtain perfect teacher-student agreement?”

Specifically, we perform numerous experiments with conventional logit-based distillation aimed at discovering under which conditions one can obtain perfect agreement and how interventions to these conditions affect the agreement.

D.2 Method

To illuminate parts of the inner workings of knowledge distillation we perform a variety of experiments. Unless otherwise mentioned, we will keep the teacher model fixed and denote it as \mathcal{M}_t and will analyze students \mathcal{M}_s trained with knowledge distillation. We will refer to pre-softmax predictions of \mathcal{M}_s (or \mathcal{M}_t) by \mathbf{z}_s (or \mathbf{z}_t), and denote them as logits.

We consider the conventional knowledge distillation loss function introduced in [Hinton et al. \(2015\)](#), over a labeled dataset, \mathcal{D}_L , and a distillation dataset, \mathcal{D}_D ,¹

$$\mathcal{L} = \frac{\alpha}{|\mathcal{D}_L|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_L} \ell_{\text{CE}}(\mathcal{M}_s(\mathbf{x}), \mathbf{y}) + \frac{1-\alpha}{|\mathcal{D}_D|} \sum_{\mathbf{x} \in \mathcal{D}_D} \ell_{\text{distill}}(\mathcal{M}_s(\mathbf{x}), \mathcal{M}_t(\mathbf{x})), \quad (\text{D.1})$$

where we define the two partial loss functions as usual;

$$\ell_{\text{CE}}(\mathbf{z}_s, \mathbf{y}) = - \sum_{c=1}^C \mathbf{y}_c \log(\sigma_c(\mathbf{z}_s)), \quad (\text{D.2})$$

$$\ell_{\text{distill}}(\mathbf{z}_s, \mathbf{z}_t) = -\tau^2 \sum_{c=1}^C \sigma_c\left(\frac{\mathbf{z}_t}{\tau}\right) \log\left(\sigma_c\left(\frac{\mathbf{z}_s}{\tau}\right)\right), \quad (\text{D.3})$$

with $\alpha \in [0, 1]$. We note that the multiplier τ^2 on the distillation loss merely scales the distillation loss to keep the gradients of both loss functions comparable between different choices of temperature, τ .

Agreement metric. Throughout this paper, we evaluate the *agreement* between the teacher and the student by the top-1 accuracy between the teacher’s predictions and the student’s predictions. That is, whether the class predicted by the student matches the class predicted by the teacher. Specifically, we use

$$\text{Agg}(\mathcal{M}_s, \mathcal{M}_t) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}^x} \mathbb{1}\left(\underset{c \in \{1, \dots, C\}}{\text{argmax}} \mathcal{M}_t(\mathbf{x})_c = \underset{c \in \{1, \dots, C\}}{\text{argmax}} \mathcal{M}_s(\mathbf{x})_c\right),$$

where \mathcal{D} can be either a training, validation, or test dataset. Throughout this paper, we denote an agreement of at least 95% as *perfect* and often shade this area with light gray in plots.

Assumptions and limitations. We restrict our analysis by the following assumptions: a) our teacher model is fixed throughout both in architecture and trained weights, and b) we will merely consider distillation in its simplest form as introduced in [Hinton et al. \(2015\)](#). However, we will investigate some alterations to this setup and the training scheme for an improved understanding of distillation under various interventions.

Naturally, we can not provide universally applicable conclusions from this constrained setup, nor can we investigate every possible distillation setup. However, we believe our methodical analysis and results can provide some useful insights into the (at times elusive) behavior of distillation, and leave it to future research to generalize these findings to even more general settings. While it would be seminal to provide closed-form

¹ Note, one can have $\mathcal{D}_L^x \subseteq \mathcal{D}_D$, or $\mathcal{D}_L^x \supseteq \mathcal{D}_D$, or even $\mathcal{D}_L^x \cap \mathcal{D}_D = \emptyset$, where we use the notation $\mathcal{D}^x = \{\mathbf{x} \mid \mathbf{x} \in \mathcal{D}\}$ — i.e. the input of the samples without the associated targets.

mathematical proofs for the conclusions of our analysis it is significantly more difficult (and sometimes impossible with current methods) than a structured empirical analysis, and merely few such results exist in simplified setups, see e.g. [Borup and Andersen \(2023, 2021\)](#); [Mobahi et al. \(2020\)](#); [Phuong and Lampert \(2019\)](#). Finally, we acknowledge that our analysis bears a resemblance to those of [Stanton et al. \(2021\)](#) and [Beyer et al. \(2022\)](#), and stress that this analysis aims to extend on (and at the very least recover some of) the conclusions by these previous works. Our starting point will be from the conclusion of [Stanton et al. \(2021\)](#): *“Optimization is challenging in knowledge distillation: even in cases when the student has sufficient capacity to match the teacher on the distillation data, it is unable to do so.”*

In the following section, we pose numerous different hypotheses on the “inner workings” of knowledge distillation and present various experiments performed in the quest of elucidating these hypotheses.

D.3 Experiments

One of the main conclusions of [Stanton et al. \(2021\)](#) and [Beyer et al. \(2022\)](#) is that the main challenge of knowledge distillation is to solve the unusually difficult optimization problem sufficiently well. This conclusion is supported by a lack of agreement between student and teacher models in a self-distillation setup, where the student has the capacity to perfectly represent the teacher, and by students consistently improving in accuracy with long training schedules when distilling large teachers. In particular, [Stanton et al. \(2021\)](#) finds that a randomly initialized student or even a student initialized at the teacher initialization is unable to achieve training agreements exceeding 80% on CIFAR-100 ([Krizhevsky et al., 2009](#)).

Our initial aim is to determine under which conditions, we can obtain perfect agreement, and then to analyze the effect of various interventions on such conditions. Thus, we naturally start in the self-distillation setup, and in the consecutive, we minimize (D.1) with $\alpha = 0$ unless otherwise mentioned. We train a ResNet-20 teacher, and distill it into various ResNet-20 students initialized with weights interpolated between the trained teacher weights θ_t^* , and the initial teacher weights $\theta_t^{(0)}$. See the supplementary materials for additional details and ablation experiments.

Naïve solutions. By direct analysis of (D.1), (D.2), and (D.3), a few naive solutions obtaining perfect agreement are immediate. First, for $\alpha = 1$, initialization of the student with the teacher initialization (i.e. $\theta_t^{(0)}$), and weight decay of 10^{-4} , will make the distillation procedure identical to the teacher training and yield a perfect agreement. Furthermore, if $\alpha = 0$, no weight-decay is used, and we let the student initialization be the trained teacher (i.e. θ_t^*) then (D.1) is minimized at initialization and we obtain perfect agreement without any training. However, if we deviate from any of these settings, the effect on the agreement is not clear and the following sections investigate exactly this.

D.3.1 Linearly Interpolated Model Weights

We now consider the case where the student initialization, $\theta_s^{(0)}$, is linearly interpolated between the initial teacher weights $\theta_t^{(0)}$, and the trained teacher weights θ_t^* , i.e.

$$\theta_s^{(0)} = (1 - \lambda)\theta_t^{(0)} + \lambda\theta_t^*.$$

In the below, we propose variations to this, and our initial goal is to obtain perfect agreement across all $\lambda \in [0, 1]$, but in particular for $\lambda = 0$. In practice, we use $\lambda \in \{0.0, 0.125, 0.25, \dots, 1.0\}$. Finally, we adopt the procedure from [Stanton et al. \(2021\)](#) and replace all batch-normalization layers ([Ioffe and Szegedy, 2015](#)) with layer-normalization layers ([Ba et al., 2016](#)), in order to avoid the non-parametric running batch statistics affecting the predictions of identically parameterized models due to the shuffling of samples.

Teacher weights are not optimal weights for self-distillation with weight decay.

While self-distillation has the nice property, that we know the student has the capacity to match the teacher, we also know an optimal solution to (D.3); namely the teacher weights, θ_t^* . However, [Stanton et al. \(2021\)](#) finds that if the student weights, θ_s , are initialized too far away from the optimal weights, then the agreement between the trained student and the teacher is still low. One reason for this could be that, while indeed θ_t^* does minimize (D.3), this is not generally true when using weight decay. That is, with SGD and weight decay we no longer minimize (D.3), but rather $\mathcal{L}_{\text{distill}} + \beta\|\theta_s\|^2$. Thus, for $\beta > 0$ then θ_t^* is likely not an optimal solution, and expecting perfect agreement from θ_t^* is unreasonable.

To verify this, we train students both with a weight-decay of 10^{-4} and without any weight-decay. From Figure D.1 we observe a) when using weight-decay students with $\lambda \geq 0.25$ all reach a lower agreement with the teacher compared to not using weight-decay, verifying that θ_t^* is indeed not optimal weights, and b) when no weight-decay is used, students initialized close to the optimal reach almost 100% agreement, but the agreement also slowly taper off when initialized further away from the trained teacher. These observations hold for both training and test agreements.

Softened and sharpened labels make distillation more challenging. In the above, we have adopted the default temperature of $\tau = 4$ from [Stanton et al. \(2021\)](#), but we found that even for known optimal weights, we are unable to perfectly match the teacher if we are not initialized somewhat in the direction of the optimal weights. Thus, one might ask, if the task of matching the soft labels is too difficult a task, and if we should consider changing the temperature. Thus, we train students without weight-decay and with varying temperatures $\tau \in \{0.2, 1, 4, 8\}$, corresponding to either a sharpening of the labels, using the original predictions, or two different degrees of softening the labels. From Figure D.2 we observe that a) softening labels slightly reduces the agreement for small λ compared to the original predictions ($\tau = 1$), and b) that sharpening of the labels ($\tau = 0.2$) is significantly worse than using softened labels with temperature $\tau \in \{1, 4, 8\}$ unless we initialize the student very close to the trained teacher, as well as c) a temperature of $\tau = 1$ yields near-perfect agreement for all λ , except for $\lambda = 0$.

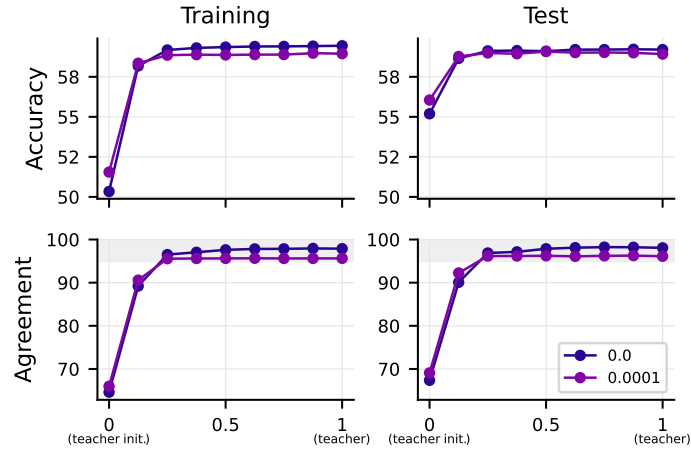


Figure D.1: Accuracy and agreement for ResNet-20 students initialized at linearly interpolated weights and distilled with different choices of weight-decay. Without weight-decay the trained teacher initialization is optimal. We use $\alpha = 0$ and $\tau = 4$ here.

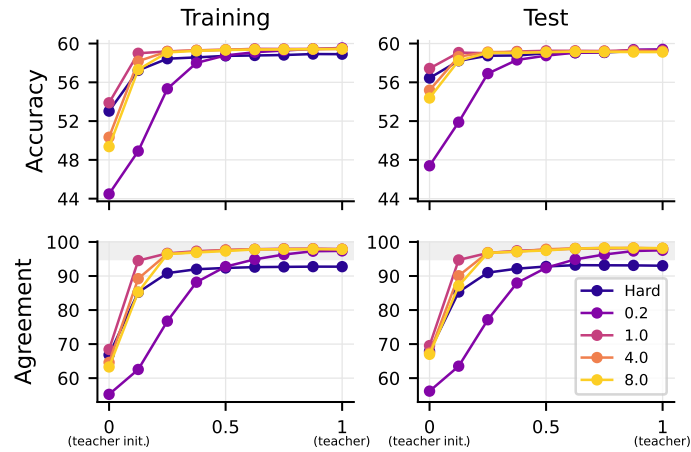


Figure D.2: Accuracy and agreement for ResNet-20 students initialized at linearly interpolated weights and distilled with different choices of temperature, τ , and without weight-decay. Hard labels are one-hot encoded labels. We use $\alpha = 0$ for all experiments here.

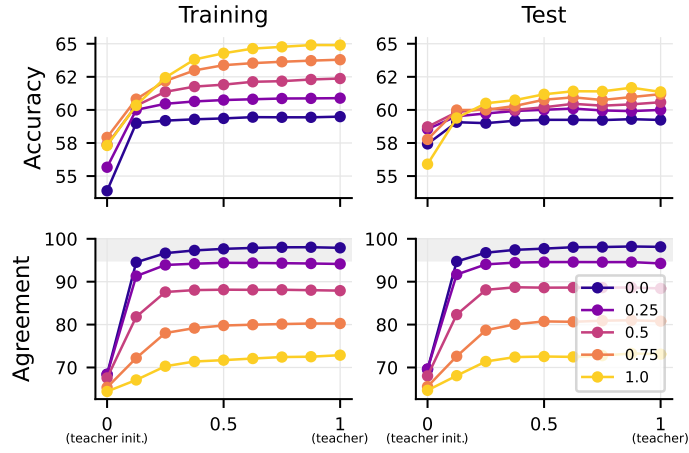
Hard labels are not the solution. Since soft labels appear to be a challenge for self-distillation, one might suspect the possible noise in the non-max indices of the soft label is too hard to fit for the student yielding subpar agreement. However, as evident both from the sharpened labels ($\tau = 0.2$) in Figure D.2 and the hard targets where we train students with temperature $\tau = 1$, no weight-decay, one-hot encoded (hard) teacher predictions, this is not the case. In fact, assigning all weight to the top-1 teacher prediction (equivalent to $\tau \rightarrow 0$) consistently underperforms ordinary soft labels (i.e. $\tau \geq 1$) in agreement for most λ . Thus, while too-soft labels might be too uninformative, so might hard labels. This observation is further supported by the hard labels underperforming sharpened labels for $\lambda > 0.5$ and not obtaining perfect agreement irrespective of initialization.

Accuracy and agreement are negatively correlated when changing α . While in the above, we merely consider a fixed $\alpha = 0$, alternating $\alpha \in [0, 1]$, yields a negative correlation between accuracy and agreement (see Figure D.3). In particular, when increasing α from 0 to 1, the accuracy consistently and gradually increases for all interpolation weights $\lambda \in [0, 1]$, while the agreement consistently and gradually decreases. Thus, this poses a trade-off between agreement and accuracy when choosing α . We observe this general pattern both when distilling without (see Figure D.3a) and with (see Figure D.3b) weight-decay.

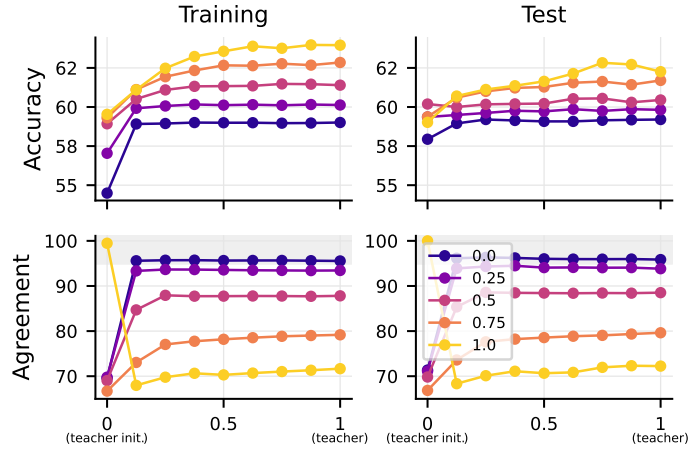
Training for a long time is helpful. In the previous experiments, all students were trained for an identical amount of 300 epochs, but models initialized closer to the teacher, i.e. large λ are closer to a known fitted model, and one would expect fewer training steps would be needed to reach a reasonable solution in such cases, compared to models initialized at random (i.e. smaller λ) making the comparison across different λ unfair. Thus, we now train models initialized from the teacher initialization ($\lambda = 0$) for 100, 300 (default), 500, 1000, and 5000 epochs, respectively. From Figure D.4 we find that longer training yields notable improvements in agreement both with and without weight decay. Notably, we observe evanescent gains when exceeding 1000 steps (which is significantly longer training than commonly used) without weight decay, but not when using weight decay. Hence, we find that if our student is trained for significantly longer durations, with weight decay, a temperature of $\tau = 1$ and $\alpha = 0$, we are able to achieve (near) perfect agreement for all $\lambda \in [0, 1]$.

D.3.2 Training Supervised Students for Comparison

We now consider the baseline of fully supervised students (i.e. $\alpha = 1$) initialized at the linearly interpolated weights. This training procedure is identical to how the teacher is trained, and thus, when $\lambda = 0$ and weight-decay are used, the student is trained identically to the teacher, and we obtain perfect agreement and identical accuracy — see Figure D.5. Furthermore, from Figure D.5 we observe a near-consistent improvement in both accuracy (even outperforming the teacher) and agreement when the student is initialized at weights increasingly closer to the trained teacher model and trained for 300 epochs. Yet, it only reaches about 72% agreement at best. However, any student initialized with $\lambda > 0$ is essentially pre-trained for 300λ epochs, yielding an unfair comparison. Thus, we also train identical student models, with reduced training schemes depending on how close to the teacher they are initialized. Specifically, they are trained



(a) Varying α and not using weight decay.



(b) Varying α and using weight decay.

Figure D.3: Accuracy and agreement for ResNet-20 students initialized at linearly interpolated weights and distilled with different choices of distillation weight, α . Here, $\alpha = 1$ corresponds to supervised training, while $\alpha = 0$ corresponds to only using the distillation loss. We use $\tau = 1$ for all experiments here.

for $(1 - \lambda)300$ epochs,² and we observe a notable change in performance both with and without weight decay. While initially improving in accuracy when getting closer to the trained teacher, the performance eventually drops off when initialized too close to the final teacher and only trained for a few epochs. However, for $\lambda \in (0, 1)$ the agreement monotonically increases as we initialize closer to the trained teacher. Yet, when initialized and trained for one epoch from the final trained teacher, the training agreement suddenly drops to the level observed when training with long schedules, while the test agreement is largely unchanged from $\lambda = 0.875$. Notably, with no training and initialization at $\lambda = 1$, the agreement would be 100%, just as observed for $\lambda = 0$ and 300 epochs.

² If $\lambda = 1$, we use a single epoch.

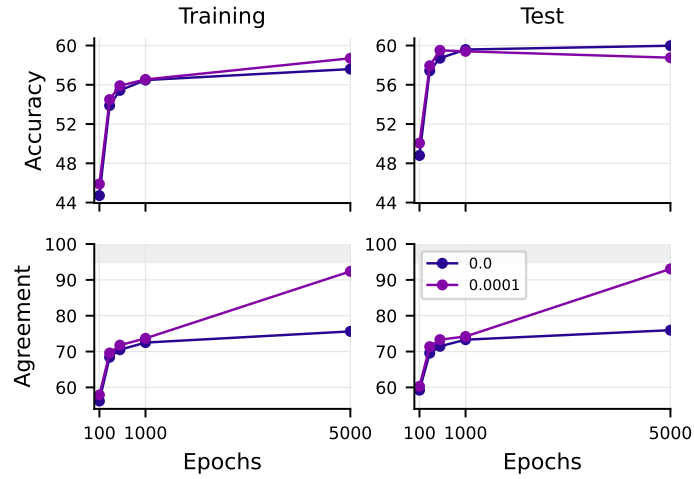


Figure D.4: Accuracy and agreement for ResNet-20 students initialized at the teacher initialization and distilled for different numbers of epochs with different choices of weight-decay. We use $\alpha = 0$ and $\tau = 1$.

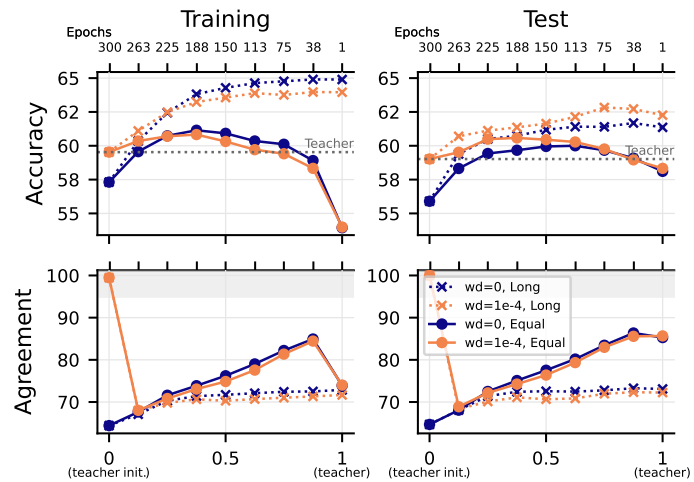


Figure D.5: Students trained fully supervised but initialized at different interpolated weights of the teacher. For each initialization, both a student with and without weight decay are trained with both a *long* training schedule of 300 epochs and an *equal* schedule of $(1 - \lambda)300$ epochs. We measure the agreement towards the fully trained teacher model.

D.3.3 Teacher Training-trajectory Interpolated Model Weights

In the above, we assume that the distance of model weights on the linear path between the teacher initialization and the final trained teacher is representative of the distance of the predictions produced by such models. However, this is not immediately clear, and in Section D.II.1 we provide additional experiments with students initialized at model weights interpolated along the training-trajectory of the teacher model. In summary, the overall findings under this interpolation scheme do not change.

D.4 Related Works

Knowledge distillation has shown great empirical effectiveness in obtaining well-performing student models for improved model efficiency, model transparency, as well as semi- and unsupervised training. First introduced by [Ba and Caruana \(2014\)](#); [Bucila et al. \(2006\)](#), their seminal work showed that large cumbersome ensemble models could be compressed into a single small student model with comparable predictive performance to the ensemble teacher. More recent work on knowledge distillation originates in the work of [Hinton et al. \(2015\)](#) and focuses on matching the softened probabilistic class predictions of a teacher model. Since then research into different alterations on this setting has been conducted with varying success ([Romero et al., 2015](#); [Mirzadeh et al., 2020](#); [Park et al., 2019](#); [Cho and Hariharan, 2019](#); [Beyer et al., 2023](#); [Borup et al., 2023](#); [Srinivas and Fleuret, 2018](#)).

Most of this research focuses on empirically improving some metric of predictive performance (e.g. accuracy) over a (set of) benchmark dataset(s), by changing either what statistic from the teacher to match ([Park et al., 2019](#); [Srinivas and Fleuret, 2018](#); [Zagoruyko and Komodakis, 2017](#); [Romero et al., 2015](#); [Yim et al., 2017](#)), the available labeled and unlabeled data ([Phoo and Hariharan, 2021](#); [Caron et al., 2021](#); [Chen et al., 2019](#); [Fang et al., 2019](#); [Nayak et al., 2019](#)), or what model(s) to use as the teacher ([Furlanello et al., 2018](#); [Hinton et al., 2015](#); [Xie et al., 2020](#); [Gupta et al., 2016](#); [You et al., 2017](#); [Liu et al., 2020](#); [Borup et al., 2023](#)). Such proposed adaptations are often based on the belief that the student actually does match the corresponding statistic from the teacher, which is largely unsupported. To the best of our knowledge, merely [Stanton et al. \(2021\)](#) explicitly investigates this claim and argues that this is indeed not generally true. This is further supported by [Beyer et al. \(2022\)](#) which finds that care must be taken when defining the distillation setup, and one should expect excessively long training schemes in order to succeed with distillation.

The theoretical justification of the efficacy of knowledge distillation is largely lacking, with recent work showing that the special case of self-distillation corresponds to particular types of regularization when considering analytically tractable settings ([Mobahi et al., 2020](#); [Phuong and Lampert, 2019](#); [Borup and Andersen, 2021, 2023](#)). However, to the best of our knowledge, no rigorous theoretical justification for knowledge distillation with neural networks has been published to this date, making rigorous controlled empirical investigations such as those by [Beyer et al. \(2022\)](#); [Stanton et al. \(2021\)](#) important for the understanding of knowledge distillation.

D.5 Conclusion

In this paper, we empirically investigate the behavior of knowledge distillation under various interventions with the aim of optimizing the fidelity of the student models. In particular, we aim to obtain perfect agreement between student and teacher models and perform several interventions to our setup to examine the effect of these changes on the agreement. Amongst others, we recover the findings of [Stanton et al. \(2021\)](#); students with sufficient capacity to match the teacher are not necessarily able to do so in practice due to a challenging optimization problem. Furthermore, we observe support for the findings of [Beyer et al. \(2022\)](#); excessively long training schemes are important for the success of knowledge distillation. We also find that when weighting the ground-truth targets and the teacher predictions by α agreement and accuracy become negatively correlated over α (i.e. larger α yields larger accuracy, but lower agreement, and vice versa).

D.6 Future Directions of Research

In this paper, we have investigated and established the challenges of obtaining perfect agreement between a teacher model and students initialized at various interpolations between the trained and untrained teacher. However, it remains to extend this analysis to include randomly initialized students as well as use teachers of different sizes than the student to more closely resemble the usual knowledge distillation setup. Furthermore, we will consider additional measures of alignment between teachers and students, such as e.g. centered kernel alignment ([Cortes et al., 2012](#)) and KL-divergence, thereby not relying merely on the top-1 prediction of the models. We plan to investigate these directions in the future.

Bibliography

- Aghli, N. and Ribeiro, E. (2021). Combining weight pruning and knowledge distillation for CNN compression. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 3185–3192. Cited on page 128.
- Anwar, S., Hwang, K., and Sung, W. (2017). Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3). Cited on page 128.
- Ba, J. L. and Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems*, volume 27, pages 2654–2662. Cited on pages 128 and 136.
- Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization. In *Proceedings of the NIPS 2016 Deep Learning Symposium*. Cited on page 131.
- Beyer, L., Izmailov, P., Kolesnikov, A., Caron, M., Kornblith, S., Zhai, X., Minderer, M., Tschannen, M., Alabdulmohsin, I., and Pavetic, F. (2023). FlexiViT: One Model for All Patch Sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14496–14506. Cited on page 136.

- Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., and Kolesnikov, A. (2022). Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10915–10924. Cited on pages 128, 130, 136, and 137.
- Borup, K. and Andersen, L. N. (2021). Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation. In *Advances in Neural Information Processing Systems*, volume 34, pages 5316–5327. Cited on pages 128, 130, and 136.
- Borup, K. and Andersen, L. N. (2023). Self-Distillation for Gaussian Process Models. *arXiv preprint arXiv:2304.02641*. Cited on pages 128, 130, and 136.
- Borup, K., Phoo, C. P., and Hariharan, B. (2023). Distilling from Similar Tasks for Transfer Learning on a Budget. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11431–11441. Cited on pages 136.
- Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pages 535–541. Cited on pages 128 and 136.
- Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9630–9640. Cited on page 136.
- Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., and Tian, Q. (2019). Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3514–3522. Cited on page 136.
- Cho, J. H. and Hariharan, B. (2019). On the Efficacy of Knowledge Distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Cited on page 136.
- Cortes, C., Mohri, M., and Rostamizadeh, A. (2012). Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828. Cited on page 137.
- Dong, B., Hou, J., Lu, Y., and Zhang, Z. (2019). Distillation \approx Early Stopping? Harvesting Dark Knowledge Utilizing Anisotropic Information Retrieval For Overparameterized Neural Network. *arXiv preprint arXiv:1910.01255*. Cited on page 128.
- Fang, G., Song, J., Shen, C., Wang, X., Chen, D., and Song, M. (2019). Data-Free Adversarial Distillation. *arXiv preprint arXiv:1912.11006*. Cited on page 136.
- Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on page 128.
- Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born Again Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1607–1616. PMLR. Cited on page 136.

- Gotmare, A., Shirish Keskar, N., Xiong, C., and Socher, R. (2019). A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. Cited on page 128.
- Gupta, S., Hoffman, J., and Malik, J. (2016). Cross Modal Distillation for Supervision Transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836. Cited on page 136.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 770–778. Cited on page 141.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*. Cited on pages 128, 129, and 136.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456. pmlr. Cited on page 131.
- Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big Transfer (BiT): General Visual Representation Learning. In *Proceedings of the 16th European Conference on Computer Vision*, pages 491–507. Cited on page 128.
- Krizhevsky, A., Nair, V., and Hinton, G. (2009). Learning multiple layers of features from tiny images. Cited on pages 130 and 141.
- Liu, Y., Zhang, W., and Wang, J. (2020). Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113. Cited on page 136.
- Mirzadeh, S.-I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. (2020). Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198. Cited on page 136.
- Mobahi, H., Farajtabar, M., and Bartlett, P. L. (2020). Self-Distillation Amplifies Regularization in Hilbert Space. In *Advances in Neural Information Processing Systems*. Cited on pages 128, 130, and 136.
- Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*. Cited on page 128.
- Nayak, G. K., Mopuri, K. R., Shaj, V., Babu, R. V., and Chakraborty, A. (2019). Zero-shot knowledge distillation in deep networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 8317–8325. PMLR. Cited on page 136.
- Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational Knowledge Distillation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Cited on pages 128 and 136.
- Phoo, C. P. and Hariharan, B. (2021). Self-training For Few-shot Transfer Across Extreme Task Differences. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Cited on page 136.

- Phuong, M. and Lampert, C. H. (2019). Towards understanding knowledge distillation. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 8993–9007. Cited on pages [128](#), [130](#), and [136](#).
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). FitNets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. Cited on pages [128](#) and [136](#).
- Srinivas, S. and Fleuret, F. (2018). Knowledge transfer with jacobian matching. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 7515–7523. Cited on pages [128](#) and [136](#).
- Stanton, S., Izmailov, P., Kirichenko, P., Alemi, A. A., and Wilson, A. G. (2021). Does Knowledge Distillation Really Work? In *Advances in Neural Information Processing Systems*, pages 6906–6919. Cited on pages [128](#), [130](#), [131](#), [136](#), and [137](#).
- Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852. Cited on page [128](#).
- Tang, J., Shivanna, R., Zhao, Z., Lin, D., Singh, A., Chi, E. H., and Jain, S. (2020). Understanding and Improving Knowledge Distillation. *arXiv preprint arXiv:2002.03532*. Cited on page [128](#).
- Xie, Q., Luong, M. T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10684–10695. Cited on page [136](#).
- Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7130–7138. Cited on pages [128](#) and [136](#).
- You, S., Xu, C., Xu, C., and Tao, D. (2017). Learning from Multiple Teacher Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*, pages 1285–1294. Cited on page [136](#).
- Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Cited on pages [128](#) and [136](#).
- Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2022). Scaling Vision Transformers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 12094–12103. Cited on page [128](#).

Supplementary material

D.I Experimental Details

Throughout our experiment, we have a default setup for our training and distillation. In particular, we consider the CIFAR-100 (Krizhevsky et al., 2009) dataset, where we randomly split the default training set into 90% training samples and 10% validation samples. We apply simple augmentations in the form of first normalizing the data, followed by a random ($p = 0.5$) horizontal flip and a random resize and crop to 32×32 images. We use a batch size of 256 and train for 300 epochs with a cosine-annealing learning rate schedule starting at 0.05 and ending at 0. We use stochastic gradient descent with Nesterov momentum of 0.9 and weight decay of either 0 or 10^{-4} . We consistently use preactivation ResNet-20 models (He et al., 2016) with layer-norms instead of batch-norms and 16, 32, and 64 filters in each stage. We use a single Nvidia RTX 3060 for each experiment and employ mixed-integer (bf16) training with gradient clipping at norms of 1. Unless otherwise mentioned we use a temperature of $\tau = 4$ and a distillation weight of $\alpha = 0$.

D.II Additional Experiments and Results

In the following, we include additional ablations and experiments to supplement the findings presented in the main paper. Specifically, we will consider experiments with interpolations along the teacher training trajectory rather than linearly interpolated.

D.II.1 Interpolation Along the Teacher Training Trajectory

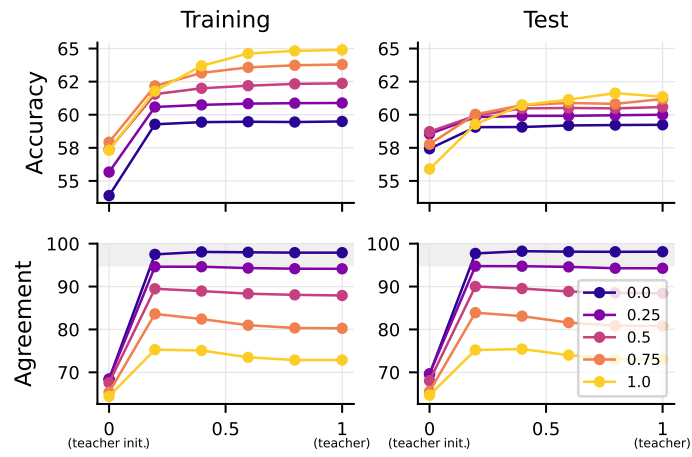
In the main paper, we work under the assumption that linearly interpolating weights is representative of how close the models associated with the weights are when producing predictions. However, it is not immediately clear that this equivalence between weight space and prediction space should hold in general, and in the following, we use checkpointed weights along the training trajectory of the teacher instead. Thus, the interpolation coefficient $\lambda \in [0, 1]$ is now interpolating along the training trajectory, i.e.

$$\theta_s^{(0)} = \theta_t^{(\lambda T)},$$

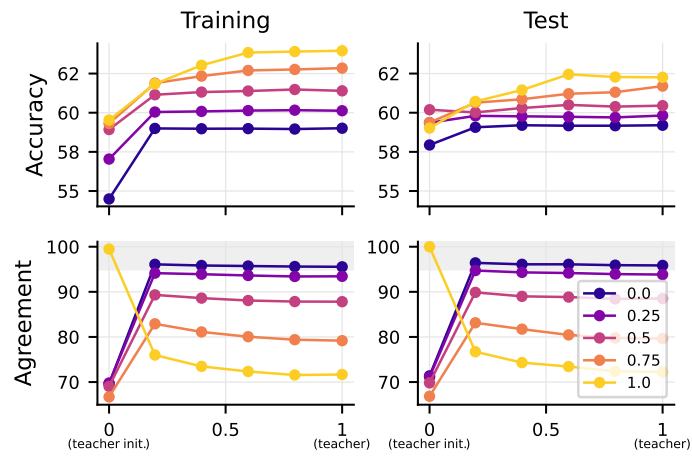
where $T > 1$ is the total number of training epochs of the teacher (default is $T = 300$), and thus $\theta_t^{(T)} = \theta_t^*$. In practice, we choose $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ such that λT are multiples of 60.

Results

The overall findings are equivalent to those of the main paper, and using this training trajectory interpolation procedure yields minor changes to the results. However, as evident from Figure D.6 and Figure D.7 using this interpolation procedure produces slightly higher accuracy and agreement — this is mostly evident for small λ . Furthermore, we observe that both the agreement and accuracy appear to be slightly less sensitive to changes in the temperature for small λ than for linear interpolation of weights.

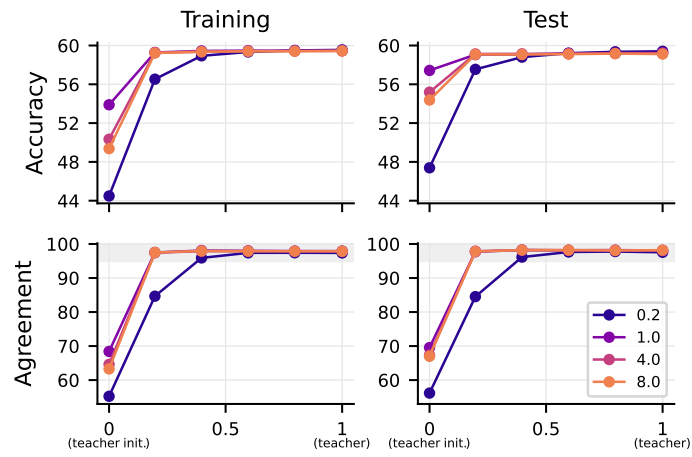


(a) Varying α and not using weight decay.

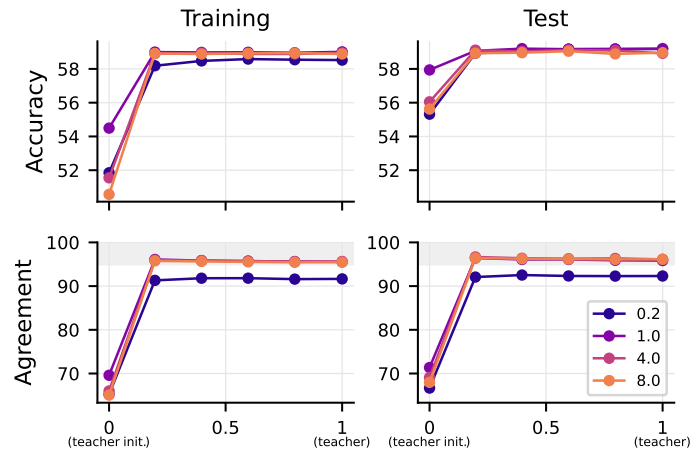


(b) Varying α and using weight decay.

Figure D.6: Accuracy and agreement for ResNet-20 students initialized at interpolated weights along the teacher training trajectory and distilled with different choices of distillation weight, α . We use $\tau = 1$ here.



(a) Varying the temperature τ and not using weight decay.



(b) Varying the temperature τ and using weight decay.

Figure D.7: Accuracy and agreement for ResNet-20 students initialized at interpolated weights along the teacher training trajectory and distilled with different choices of temperature, τ . We use $\alpha = 0$ here.

Automatic Sleep Scoring using Patient-Specific Ensemble Models and Knowledge Distillation for Ear-EEG Data

PUBLISHED IN BIOMEDICAL SIGNAL PROCESSING AND CONTROL (VOLUME 81)

Kenneth Borup

Aarhus University

Preben Kidmose

Aarhus University

Huy Phan

Queen Mary University of London

Alan Turing Institute

Kaare Mikkelsen

Aarhus University

Abstract. Human sleep can be described as a series of transitions between distinct states. This makes automatic sleep analysis (*scoring*) suitable for an automatic implementation using machine learning. However, the task becomes harder when data is sampled using more lightweight or mobile equipment, often chosen due to greater comfort for the patient. In this study we investigate the improvement in sleep scoring when multiple state-of-the-art neural networks are joined into an ensemble, and subsequently *distilled* into a single model of identical network architecture, but with improved predictive performance. In this study, we investigate ensembles of up to 10 networks and show that, on the same data, ensembles of neural networks perform better than each single subject model (improvement: 2.4%) and that this improvement can be transferred back into a single network using a combination of patient-specific data and *knowledge distillation*. The study demonstrates both a way to further improve automatic sleep scoring from mobile devices, which in itself is interesting, but also highlights the great potential of the vast amounts of unlabeled personal data which will become available from personal recording devices.

E.1 Introduction

Humans sleep for one-third of their lives. Our sleep both affects and is impacted by our health and as such knowledge about patient sleep is recognized as a valuable ingredient in clinical care and diagnosis (Berry et al., 2017; van Gilst et al., 2019). However, the current gold standard for sleep monitoring builds on manual classification (*scoring*) of *polysomnography* (PSG) recordings, the entire process of which is both expensive and intrusive on the patient’s actual sleep. This has led to repeated attempts to update the process, both through new and more light-weight recording devices (Arnal et al., 2019; Mikkelsen et al., 2019; Gangstad et al., 2019; Mikkelsen et al., 2019; Miettinen et al., 2018) and automatic algorithms for analyzing the data (Phan et al., 2019; Stephansen et al., 2018). The present study falls into both categories, in that we explore automatic algorithms specifically for scoring light-weight recordings.

Multiple studies (Koley and Dey, 2012; Mikkelsen et al., 2019; Boostani et al., 2017) have shown the efficiency of ensemble models for the classification of electroencephalography (EEG) data. At the same time, all state-of-the-art algorithms for automatic sleep scoring in the past few years have been built on neural networks (Phan and Mikkelsen, 2022). A natural question then becomes to which extent the combination of these methods will lead to even better performance.

In this paper, we train ensembles of neural networks for sleep scoring and perform a thorough investigation of the possible benefits and realistic applications of this method. We focus on an established deep neural network for automatic sleep scoring, the SE-QSLEEPNET (Phan et al., 2019), in the specific context of a proven light-weight sleep monitoring technology, the ear-EEG (Mikkelsen et al., 2015). A major limitation of neural ensembles is the added memory and computational requirements for such models. This leads us to a further investigation of the benefits of *knowledge distillation* (Hinton et al., 2015; Ba and Caruana, 2014; Bucila et al., 2006), which was specifically introduced to alleviate this problem by *distilling* the ensemble of models into a single model at the cost of a small loss in predictive performance.

The idea of *knowledge distillation* (or just *distillation*) originates back to Bucila et al. (2006), and was later brought to the deep learning setting by Ba and Caruana (2014), but it is most commonly known as a model compression technique popularized by Hinton et al. (2015). It is a procedure to transfer some statistic (often called *knowledge*) from one model (teacher) to another model (student). Originally the student was considered smaller¹ than the teacher, and the distillation procedure aimed at training the student to mimic the softened probability distribution over the logits of the (trained and fixed) teacher model alongside the original training data. However, since the formulation by Hinton et al. (2015) an extensive amount of alterations to the procedure has been proposed. A branch of research proposes mimicking the teacher on other statistics than the distribution of logits (Romero et al., 2015; Zagoruyko and Komodakis, 2017; Srinivas and Fleuret, 2018; Park et al., 2019; Yim et al., 2017), while another branch focuses on developing the transfer procedure and the choice of data used for distillation (Furlanello et al., 2018; Ahn et al., 2019; Tian et al., 2020; Lopes et al., 2017; Micaelli and Storkey, 2019; Fang et al., 2019; Anil et al., 2018; Caron et al., 2021).

¹ Depending on the task at hand different measures of model size can be relevant; e.g. model parameters, inference time, memory requirements or model complexity. However, often model parameters are considered a reasonable proxy for model size.

Exactly why knowledge distillation works well is still an open research question and an active field of research, but [Mobahi et al. \(2020\)](#) show that self-distillation² with kernel ridge regression models progressively shrinks the number of basis functions used to represent the solution, thus acting as a method of regularization. Furthermore, [Borup and Andersen \(2021\)](#) shows that this behavior is highly dependent on the weighting between labeled ground-truth data and teacher outputs used during distillation. Our application of knowledge distillation builds on the empirical successes of distillation techniques and is closely related to self-distillation.

In order to reduce the computational burden of ensemble models at inference time, we utilize knowledge distillation to distill the cumbersome ensemble into a single model. Our distillation framework is very flexible, and distillation can be performed in supervised, semi-supervised, or unsupervised settings depending on the available data.

We find that forming ensembles of neural networks does indeed improve performance relative to single networks and that by using unlabeled data from the individual patient, we can transfer some of that improvement back into a single network using knowledge distillation.

Our contributions. In this study, we present a number of important contributions to the field of automatic sleep scoring:

- To our knowledge, this is the first study successfully leveraging unlabeled, personal data, which is likely to be important in long-term sleep monitoring.
- We show that simple ensembles of 10 SEQSLEEPNET models trained independently improve predictive performance, and only differ by 0.04 in Cohen’s kappa compared to the best-case scenario of two manual scorers.
- Despite no change in model architecture, we show that a single SEQSLEEPNET model trained with our semi-supervised distillation setup retains between 50% and 100% of the improvements obtained by ensemble models (of various sizes) when trained with personal data.

Details on our experimental setup and the code to reproduce our experimental results can be found in the supplementary material.

E.2 Problem Setup and Methods

E.2.1 Data

In this study, the input data to the algorithm is a bilateral ear-EEG derivation (specifically, the average of the left ear electrodes relative to the average of the right ear electrodes), while labels come from manual scoring of a reduced PSG montage. See Figure E.1 for visualizations of the two methods. The *left-right* ear-EEG derivation is used because it has been thoroughly studied for sleep scoring, and has been shown to be a strong candidate for clinical-grade home sleep monitoring ([Mikkelsen et al., 2021](#)). We recommend reading [Mikkelsen et al. \(2019\)](#) for a detailed description of the recording platform.

² Self-distillation often refers to the use of identical teacher and student models, which is not entirely true for our setup as our teacher is an ensemble.

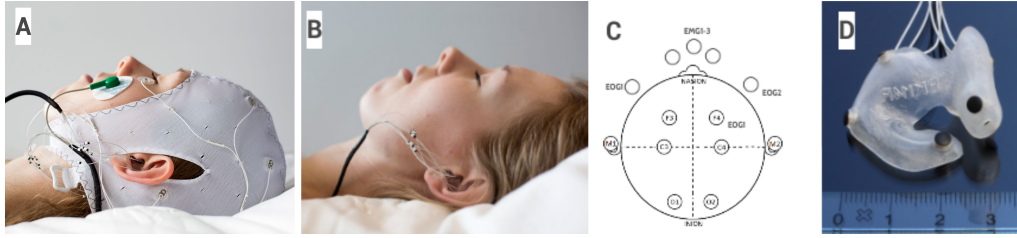


Figure E.1: The recording setup used in the data set. (A): the setup used for the labeled recordings, where the data from the electrodes in the cap is used for the manual labeling. (B): the setup used for the unlabeled recordings. There are only electrodes inside the ears and next to the right eye. (C): Positions of the electrodes in picture (A), excluding the ear electrodes. Note the two positions next to the eyes and the three on the chin. (D): an example of the soft ear pieces with dry electrodes placed inside the ears. Note that the ruler at the bottom goes from 0 to 3 cm.

The specific recordings used are presented in [Mikkelsen et al. \(2019\)](#) and [Mikkelsen et al. \(2022\)](#). Combined, they constitute a data set of 20 subjects recorded using the same equipment. Each subject has four nights of labeled recordings. Half of the subjects also have a further 12 nights of unlabeled recordings each. We shall refer to the two groups of subjects as respectively *short* and *long* subjects. See Figure E.2 for an overview.

In accordance with standard sleep scoring practice, the sleep recordings have been partitioned into 30-second epochs. For the labeled recordings, they have been manually scored by the same sleep technician according to the five-stage scoring described in the AASM manual ([Berry et al., 2017](#)): *Wake*, *REM*, *Non-REM 1*, *Non-REM 2*, and *Non-REM 3*.

E.2.2 Cohen’s Kappa Score

As is established practice when quantifying sleep scoring performance, we measure the performance of our automatic classifier by calculating *Cohen’s kappa* ([Cohen, 1960](#)) between the predicted and manual labels.

E.2.3 Model Architecture (SEQSLEEPNET Classifier)

In this paper, we use the sequence-to-sequence neural network architecture SEQSLEEPNET introduced in [Phan et al. \(2019\)](#). SEQSLEEPNET takes a sequence of L consecutive epochs as input and outputs a sequence of L 5-dimensional probability vectors. The input can be either single- or multichannel log-scale spectrograms, where the data of each channel is (approximately) normalized to zero mean and unit variance for each frequency bin. The output probability vectors are the predicted class probability for each of the $P = 5$ sleep stages. In this paper, we follow the settings of [Mikkelsen et al. \(2021\)](#) and use spectrograms with $T = 29$ time bins (spanning 30 seconds), with $F = 129$ frequency bins, and with a single, $C = 1$, channel. Furthermore, we will use a sequence length of $L = 20$ as in [Mikkelsen et al. \(2021\)](#) and [Phan et al. \(2019\)](#). We denote each epoch by $\mathbf{z}_i \in \mathbb{R}^{T \times F \times C}$, and the sequence of L epochs by $\mathbf{x}_n = (\mathbf{z}_n, \mathbf{z}_{n+1}, \dots, \mathbf{z}_{n+L-1}) \in \mathbb{R}^{L \times T \times F \times C}$.³ For more details on the SEQSLEEPNET architecture we refer to [Phan et al. \(2019\)](#) and our implementation

³ When using multiple nights for training or evaluation, we assume no gap between the nights, and thus, some sequences of epochs might overlap two different nights or even subjects. However, the effect of this on the overall predictive performance is low from our experience.

in PyTorch available on GitHub: <https://github.com/Kennethborup/SeqSleepNet>. For details on training, we refer to Section E.2.4 and to Appendix E.I for experimental details and additional results.

Sliding window average prediction. The sequence-to-sequence nature of the SEQSLLEEPNET allows us to obtain predictions on a sequence of epochs with a sliding window approach. More specifically, we first apply our model on $\mathbf{x}_1 = (\mathbf{z}_1, \dots, \mathbf{z}_{1+L-1})$ followed by $\mathbf{x}_2 = (\mathbf{z}_2, \dots, \mathbf{z}_{2+L-1})$ and so on. Thus, by *sliding* our model across a sequence of L epochs by increments in the index of one, we obtain L predictions for each epoch, and averaging these predictions for each epoch yields a new probability vector.⁴ Throughout this study, we will always be utilizing this sliding window average and therefore refer to this procedure merely as predicting. Note, that while this procedure improves predictive performance, it also requires L times as many steps of predictions, which is computationally expensive at inference time, especially for large L .

E.2.4 Model Training

We perform our experiments using Leave-One-Subject-Out-Cross-Validation (LOSO-CV) across all 20 subjects (both *short* and *long*). For each CV-step we divide the subjects into training (15 subjects), validation (4 subjects), and test (one subject) sets, irrespective of the subject type (long/short) - see Figure E.2 for an illustration of this. Thus, for each random initialization of our model, we train 20 different models, but will merely refer to it as one model and will report the predictive performance of this model as the average Cohen’s kappa on the 4 scored nights of the test subjects across all 20 subjects. Our study is split into two phases; in Phase 1, we train a set of single SEQSLLEEPNET models in the classical supervised way and denote these models as *baseline* models. In Phase 2, we collect these baseline models into a large and computationally demanding ensemble model (called a *teacher* model) which in turn is distilled into a single SEQSLLEEPNET model by utilization of knowledge distillation, thereby reducing the computational requirements at inference time significantly.

Baseline models and training (Phase 1). We independently train M random initializations of the SEQSLLEEPNET model and refer to these models as baseline models, each denoted by \mathcal{B}_j for $j = 1, \dots, M$. We train each model to minimize the cross-entropy loss on the 4 scored nights for all training subjects. That is, let $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ be the training dataset, then we minimize

$$\mathcal{L}_{\text{CE}}(\mathcal{B}_j) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{train}}} \ell_{\text{CE}}(\mathbf{y}, \mathcal{B}_j(\mathbf{x})), \quad \text{where}$$

$$\ell_{\text{CE}}(\mathbf{t}, \mathbf{s}) \stackrel{\text{def}}{=} - \sum_{l=1}^L \sum_{p=1}^P [\mathbf{t}]_p \log([\mathbf{s}]_p), \quad \text{for } \mathbf{t}, \mathbf{s} \in \mathbb{R}^{L \times P},$$

and where $\mathcal{B}_j(\mathbf{x})$ is the sequence of predicted class-probabilities on \mathbf{x} , and \mathbf{y} the sequence of associated one-hot encoded ground-truth labels. We refer to the training of the

⁴ Note, for the initial and final $L-1$ epochs we will not obtain L predictions, due to missing data prior and after the sequence, and we will merely consider the average predictions of all the possible predictions at these epochs.

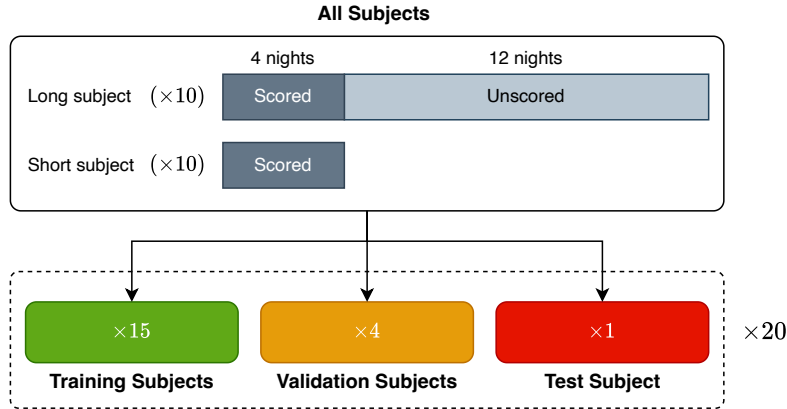


Figure E.2: We perform our experiments using Leave-One-Subject-Out-Cross-Validation (LOSO-CV), and our dataset consists of 20 subjects. 10 subjects (denoted *short* subjects) have 4 nights of scored observations, and the remaining 10 subjects (denoted *long* subjects) have 4 nights of scored observations along with 12 nights of *unscored* observations. For each CV-step, we divide the subjects into a **training** (15 subjects), **validation** (4 subjects), and **test** (one subject) set, irrespective of the subject type (long/short). Thus, each set can consist of both long and short subjects, but whether the unscored recordings are used or not, depends on the experiment.

baseline models as supervised training or Phase 1 (see Figure E.3) and remind that by \mathcal{B}_j we in fact refer to the 20 underlying models trained in a LOSO-CV setup.

Ensemble models. Based on the M baseline models, $\{\mathcal{B}_j\}_{j=1}^M$, we can construct ensemble models of size m , where $m \in \{1, \dots, M\}$ is the number of baseline models used in the ensemble. We construct the ensemble models as the unweighted average of the m individual predictions on some sample \mathbf{x} , i.e. as $\mathcal{T}(\mathbf{x}) = \frac{1}{m} \sum_{j=1}^m \mathcal{B}_j(\mathbf{x})$, and thus $\mathcal{T}(\mathbf{x})$ is still a probability vector. We denote an ensemble model of size m by \mathcal{T}_m , or merely \mathcal{T} if the size is unambiguous.⁵ By using an unweighted average ensemble of baseline models, no additional training is required to construct the ensemble, but m times more prediction steps are required in order to perform inference. Due to the L times more steps required by the sliding window prediction of each baseline model, prediction with an ensemble model requires Lm times more prediction steps compared to naïve prediction using a single baseline model. In Section E.3 we report the predictive performance of all possible ensemble models constructed of unique sets of m baseline models, and in the following we investigate a semi-supervised adaptation of knowledge distillation as a way to reduce the computational requirements of ensemble models at inference time. We stress the fact, that distilling an ensemble of m baseline models into a single student model, reduces the computational requirements at inference time by $m \times$ at a small loss in predictive performance (see e.g. Figure E.4).

⁵ We remind that by \mathcal{T}_m we in fact refer to the 20 underlying models trained in a LOSO-CV setup. Thus, each of the 20 underlying ensemble models is the unweighted combination of the m underlying baseline models at the particular CV-step.

E.2.5 Distillation of Ensembles to Single Models (Phase 2)

In the following, we present our approach to distillation which allows for utilization of unlabeled data in a semi-supervised manner. This approach is at large similar to methods sometimes known as self-training or self-distillation.

In this study, we utilize a semi-supervised adaptation of the original knowledge distillation technique, where we match the teacher on a set of unlabeled data, and employ an imbalanced smoothing of the labels - see below for details. Thus, we now refer to \mathcal{T} as the *teacher* model and initialize a new SEQSLLEEPNET model denoted by \mathcal{S} which we refer to as the *student* model following the conventions in the knowledge distillation literature. Let $\mathcal{D}_{\text{distill}} = \mathcal{D}_{\text{gt}} \cup \mathcal{D}_{\text{pseudo}}$ be the *distillation dataset*, where $\mathcal{D}_{\text{gt}} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N_{\text{gt}}}$ and $\mathcal{D}_{\text{pseudo}} = \{(\mathbf{x}_n, \mathcal{T}(\mathbf{x}_n))\}_{n=1}^{N_{\text{pseudo}}}$ are the ground-truth and pseudo-labeled data sets, respectively. We will refer to the predictions of the teacher, $\mathcal{T}(\mathbf{x})$, on the pseudo-labeled dataset, $\mathcal{D}_{\text{pseudo}}$, as *pseudo-labels*. Note, the set of input samples for \mathcal{D}_{gt} and $\mathcal{D}_{\text{pseudo}}$ need not be equal, and are often disjoint. Furthermore, $\mathcal{D}_{\text{pseudo}}$ does not require any labels allowing for a semi- or unsupervised distillation procedure. Define the distillation-loss as a weighted (by $\alpha \in [0, 1)$) sum of two terms; one for scored samples and one for pseudo-labeled samples, i.e. as

$$\mathcal{L}_{\text{distill}}(\mathcal{S}) \stackrel{\text{def}}{=} \alpha \mathcal{L}_{\text{gt}}(\mathcal{S}) + (1 - \alpha) \mathcal{L}_{\text{pseudo}}(\mathcal{S}) \quad (\text{E.1})$$

where \mathcal{L}_{gt} and $\mathcal{L}_{\text{pseudo}}$ are the ground truth and pseudo loss, respectively, defined as

$$\mathcal{L}_{\text{gt}}(\mathcal{S}) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_{\text{gt}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{gt}}} \ell_{\text{CE}}(\mathbf{y}, \mathcal{S}(\mathbf{x})), \quad \text{and} \quad (\text{E.2})$$

$$\mathcal{L}_{\text{pseudo}}(\mathcal{S}) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_{\text{pseudo}}|} \sum_{(\mathbf{x}, \mathcal{T}(\mathbf{x})) \in \mathcal{D}_{\text{pseudo}}} \ell_{\text{CE}}(\sigma(\tilde{\mathcal{T}}(\mathbf{x})/\tau), \mathcal{S}(\mathbf{x})), \quad (\text{E.3})$$

where σ is the softmax function, $\tilde{\mathcal{T}}(\mathbf{x})$ is the pre-softmax logits of $\mathcal{T}(\mathbf{x})$ (i.e. $\sigma(\tilde{\mathcal{T}}(\mathbf{x})) = \mathcal{T}(\mathbf{x})$), $\alpha \in [0, 1)$ is a weighting parameter, and τ a temperature for softening/sharpening of the teacher class-probabilities introduced in Hinton et al. (2015).⁶ Setting $\alpha = 0$ (and ensuring $\mathcal{D}_{\text{pseudo}} \neq \emptyset$) makes the distillation procedure fully unsupervised, while $\alpha \in (0, 1)$ yields a semi-supervised procedure. In this paper we consistently use $\alpha = 0.5$ and $\tau = 1$.⁷ Hence, the distillation procedure is as follows: 1) fix the teacher, \mathcal{T} , 2) compute pseudo-labels with \mathcal{T} , and 3) train the student, \mathcal{S} , on the distillation dataset, $\mathcal{D}_{\text{distill}}$, by minimizing $\mathcal{L}_{\text{distill}}(\mathcal{S})$ in (E.1). See Phase 2 in Figure E.3 for an illustration of the distillation procedure. Note, when distilling a teacher, \mathcal{T}_m , to a single baseline model we reduce the computational requirements at inference by m times which for e.g. $m = 10$ corresponds to a decrease of 90%.

Since no labels are used for $\mathcal{D}_{\text{pseudo}}$, we can use both scored (i.e. discarding the known labels) and truly unscored samples in $\mathcal{D}_{\text{pseudo}}$ as well as samples from validation and/or test subjects. When any data from the test subject is used during distillation, we refer to the student as a *personalised* student, and a *general* student otherwise.

⁶ Note, unlike classical distillation, we do not apply the temperature softening to the student logits, but only the teacher logits creating an *imbalanced* setting.

⁷ By investigation of different choices of τ , we find that the performance does not change much for τ between 0.1 and 1, and $\tau \leq 2$ yield some improvement - see Figure E.8. Thus, we consistently use $\tau = 1$. Furthermore, we leave the investigation of varying α to future work.

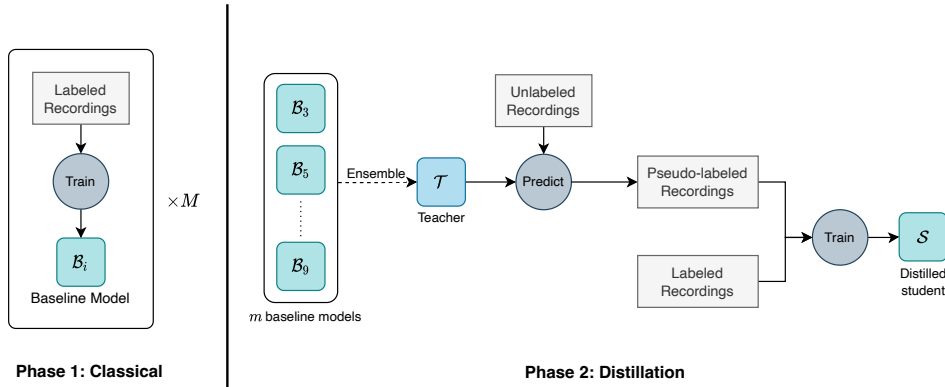


Figure E.3: Our training procedure is split into two phases: 1) classical training of baseline models and 2) distillation of ensembles of baseline models into a single student model. In Phase 1 we independently train M random initializations of the baseline model on the 15 training subjects (note, the set of training subjects depend on the initialization, but the test subject is constant.). We denote each of these trained models by B_j for $j = 1, \dots, M$. In Phase 2, we combine a subset (of size m) of baseline models into an ensemble model which we use as a teacher (denoted by \mathcal{T}). We obtain pseudo labels on a selection of unscored data (which can be from training, validation and/or test subject(s)) as predictions from \mathcal{T} and train the student model (denoted by S) on these pseudo labels as well as optionally (hard) labeled training data.

	Models	Teacher Size (m)	Personal Data	Unlabeled Data	Inference Time* (Sec.)	Long Subjects	Short Subjects	All Subjects
Baseline (ours)	1	–	–	–	10.3 (1×)	0.758 (± 0.005)	0.754 (± 0.011)	0.756 (± 0.006)
Ensemble	2	–	–	–	20.6 (2×)	0.771 (± 0.005)	0.767 (± 0.006)	0.769 (± 0.004)
Ensemble	10	–	–	–	102.8 (10×)	0.781 (± 0.000)	0.778 (± 0.000)	0.780 (± 0.000)
General Student	1	2	–	✓	10.3 (1×)	0.766 (± 0.008)	0.759 (± 0.005)	0.763 (± 0.006)
General Student	1	8	–	✓	10.3 (1×)	0.769 (± 0.003)	0.762 (± 0.003)	0.766 (± 0.002)
Personal Student	1	2	✓	✓	10.3 (1×)	0.771 (± 0.005)	0.753 (± 0.007)	0.762 (± 0.005)
Personal Student	1	8	✓	✓	10.3 (1×)	0.774 (± 0.002)	0.749 (± 0.008)	0.761 (± 0.004)
Mikkelsen et al. (2019)	1	–	–	–	–	–	–	0.73**
Mikkelsen et al. (2019)	1	–	✓	–	–	–	–	0.76**
Mikkelsen et al. (2021)	1	–	–	–	–	–	–	0.76

*Inference time is measured as the average time over 100 samples on an Apple M1 Pro CPU and extrapolated to a night of 8 hours of sleep recordings.

**Mikkelsen et al. (2019) report median performance rather than mean, and due to the left tail of the distribution, the median is larger than the mean for these reported results.

Table E.1: Here GENERAL STUDENT is trained merely on data from other subjects (see Section E.3.2), while PERSONAL STUDENT is trained on the additional 12 unlabeled subject-specific nights (see Section E.3.2). We also report a subset of ENSEMBLE models, the BASELINE introduced in this paper, as well as the state-of-the-art on the labeled data alone. The standard deviation is reported in parentheses (estimated across 4 replicated experiments for the distilled students and across all possible ensembles).

E.3 Results

We summarise selected results and baseline results in Table E.1, and have collected confusion matrices in Figure E.4. We find from the confusion matrices that the performance improvements, when going from worst performing model (baseline) to best performing (10-model ensemble), is spread across all 5 stages rather than a specific stage getting better.

Below, we have separated the analysis of our results into separate segments for specific model groups: Ensemble teachers, general students and personalised students.

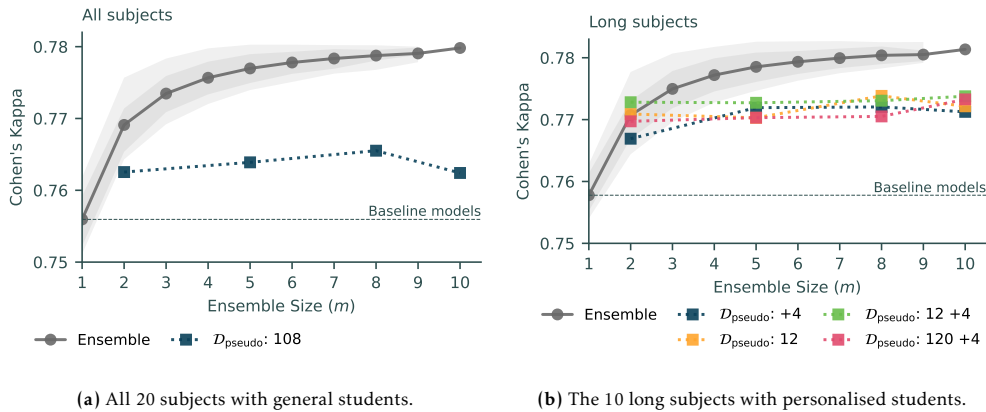


Figure E.4: Cohen’s kappa for general (a) and personalised (b) student models. The x -axis is the number of baseline models used in the teacher ensemble, but all students are merely constructed as a *single* model, and the position of students on the x -axis indicates the number of models in the teacher model. The light and dark shaded gray areas represent the 10–90% and 25–75% empirical confidence intervals.

E.3.1 Ensemble Teachers

We let $M = 10$, and in Figure E.4 we report the mean test performance of all simple ensemble models constructed of unique sets of m baseline models for $m = 1, \dots, 10$ along with the empirical 25–75% and 10–90% confidence interval for each m .⁸ In total we consider 1023 different ensemble models. We see a monotonic improvement in mean predictive performance with increasing m , where the performance increase is largest for small m , and the performance appears to saturate at ≈ 0.780 . Furthermore, there exist ensembles with $m \geq 4$ that perform equivalently to the best-performing ensemble with $m = 10$ (selecting these specific ensemble models prior to training and evaluation of all ensemble models is not possible). Compared to our baseline at 0.755, which is equivalent to the previous state-of-the-art on this dataset (Mikkelsen et al., 2021), an ensemble of merely two models improves the mean performance by 0.013, while an ensemble of 10 models improves by 0.024.

E.3.2 Distilled Students

In the following we let \mathcal{D}_{gt} be the set of all training subjects and investigate the performance of student models trained with the distillation procedure from Section E.2.5 for different choices of $\mathcal{D}_{\text{pseudo}}$. More specifically, we separately consider the case where *no* data from the test subject is used (*general students*), and the case where some data from the test subjects is used to personalize the student (*personalised students*). We investigate the impact on the student performance by the size (i.e. m) of the teacher model as well as the particular distillation dataset chosen. We repeat all experiments four times with different seeds, and report the mean performance across all four replications. See Appendix E.I for experimental details and used hyperparameters.

⁸ Note, the amount of possible models vary with m . i.e. with 10 baseline models, then for $m = 1$ there are 10 possible ensemble models, for $m = 2$ there are 45 models, for $m = 3$ there are 120 models and so on.

General Students

We now consider the case where $\mathcal{D}_{\text{pseudo}}$ is the set of 108 unscored training and validation nights.⁹ In Figure E.4a we report the mean performance for teachers of size $m = 2, 5, 8,$ and 10.

We are able to recover about 40% of the improvement obtained by the best teacher in a single student model using our distillation procedure and additional data, which yields an improvement of approx. 0.01 in predictive performance compared to the baseline. In order to verify that our distillation procedure is in fact useful, we compute a weight-space ensemble of the 10 baseline models; that is, for each layer we average the weights of the layer across all baseline models and use these averaged weights in a single SeqSlepNet model. Similar approaches to weight-space ensembles have shown great potential by Izmailov et al. (2018); Garipov et al. (2018). However, the weight-space ensemble only performs on par with the single baseline models with Cohen’s kappa of 0.759 across all 20 subjects.

Personalized Students

In the following section, we consider the case where a personalized student model is trained based on a set of unscored observations from the test subject. Thus, we only include the predictive performance on the 10 long subjects in this section. For evaluation of the models on the short subjects, we refer the reader to Figure E.10 in the appendix. Note, at no point do we use any manual scorings from the test subject. We consider the cases where we have access to either the 12 unscored nights, the 4 scored nights (without manual scores), or all 16 nights for the long test subject. In Figure E.4a we report the mean performance for teacher ensembles of size $m = 2, 5, 8,$ and 10. If we use an ensemble of size $m = 2$ personalised students perform equivalently to the teacher at a reduction of 50% in computational costs. Thus, using the distillation procedure we are able to get personalised students that improve by ≈ 0.01 in Cohen’s kappa. However, larger teacher ensembles yield only small improvements in personalised student performance. The choice of subject-specific data does not appear to be important, as long as some subject-specific data is used for distillation. This is also supported by Figure E.9 in the appendix.

In Figure E.5 we compare the performance, on a subject-level, of the baseline models, the ensemble with $m = 10$, and the personalised student based on the 12 unscored nights of the test subject. We sort the subjects by increasing baseline performance, and note that the teacher ensemble consistently outperforms the baseline models on all subjects. Furthermore, the personalised students perform at least as well as the baseline, and even surpass the teacher ensemble for some subjects, despite requiring 1/10’th of the compute at inference time (See Table E.1).

More subject-specific data is better. In Figure E.4b we observe that using the 12 unscored nights from the test subject improves the performance enough to be comparable to the ensemble teacher for some $m > 1$. In Figure E.7 we show the personalised student

⁹ For long subjects all 108 unscored training and validation nights are used, while for short subjects (for which there are 120 unscored training and validation nights) the unscored nights of one randomly chosen long subject is not included in $\mathcal{D}_{\text{pseudo}}$, which yield a total of 108 unscored nights for all subjects.

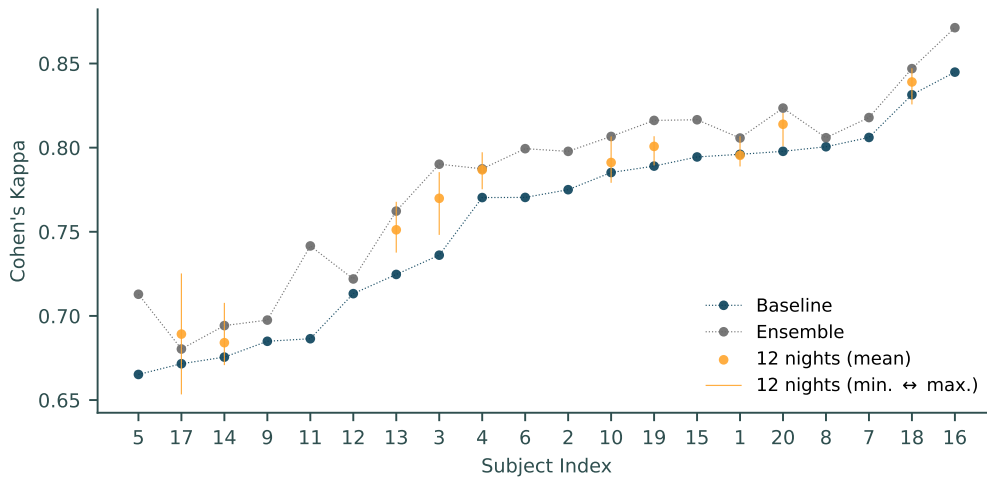


Figure E.5: Cohen's kappa on subject-level, sorted by increasing baseline performance. We report the mean of all 10 baseline models (in blue), the ensemble model of 10 baseline models (in grey), and the mean of the personalised student trained on the 12 unscored nights of the test subject (in yellow with vertical lines between min. and max. of all 8 repetitions). Note, we only report the performance of the personalised student on the long subjects.

performance when using an increasing number of unscored nights (from one to all 12 nights). We repeat the experiment 10 times with a fixed teacher of size $m = 5$, and observe a near monotonically increasing mean performance with the increase in number of nights. Thus, the more nights available to personalise the model to the test subject, the better.

Importance of adjusting pseudo-labels with a temperature. We investigate the effect of changing the temperature, τ , in (E.3) and plot the performance in Figure E.8. We observe that for $\tau \leq 2$ the performance exceeds the baseline, and more specifically we observe small differences in performance for $\tau \in \{0.1, 0.5, 1\}$ as well as hard pseudo-labels, although with a slight peak at $\tau = 0.5$. Hard pseudo-labels corresponds to letting $\tau \rightarrow 0$, and in practice we use the one-hot encoded label with one at the largest entry of the pseudo-label and zero elsewhere. For larger temperatures, the performance decreases significantly compared to the baseline.

E.4 Discussion and Conclusion

In this study we have analysed the utility of ensembles of neural networks (*neural ensembles*) for automatic sleep scoring using wearable EEG recordings. Neural ensembles appear as a likely source of improvement for a difficult problem, making the question suitable for a thorough analysis. On first approach, we find that for the size of data set available here, ensembles are a good approach, and we find a respectable improvement in Cohen's kappa when using ensembles of 10 networks, from 0.756 to 0.780, but also note that with the correct choice of baseline models, even 4-5 models is sufficient for this improvement. Crucially, we find an improvement in all individuals, meaning that, apparently, the ensemble is always better. Given that kappa values between manual

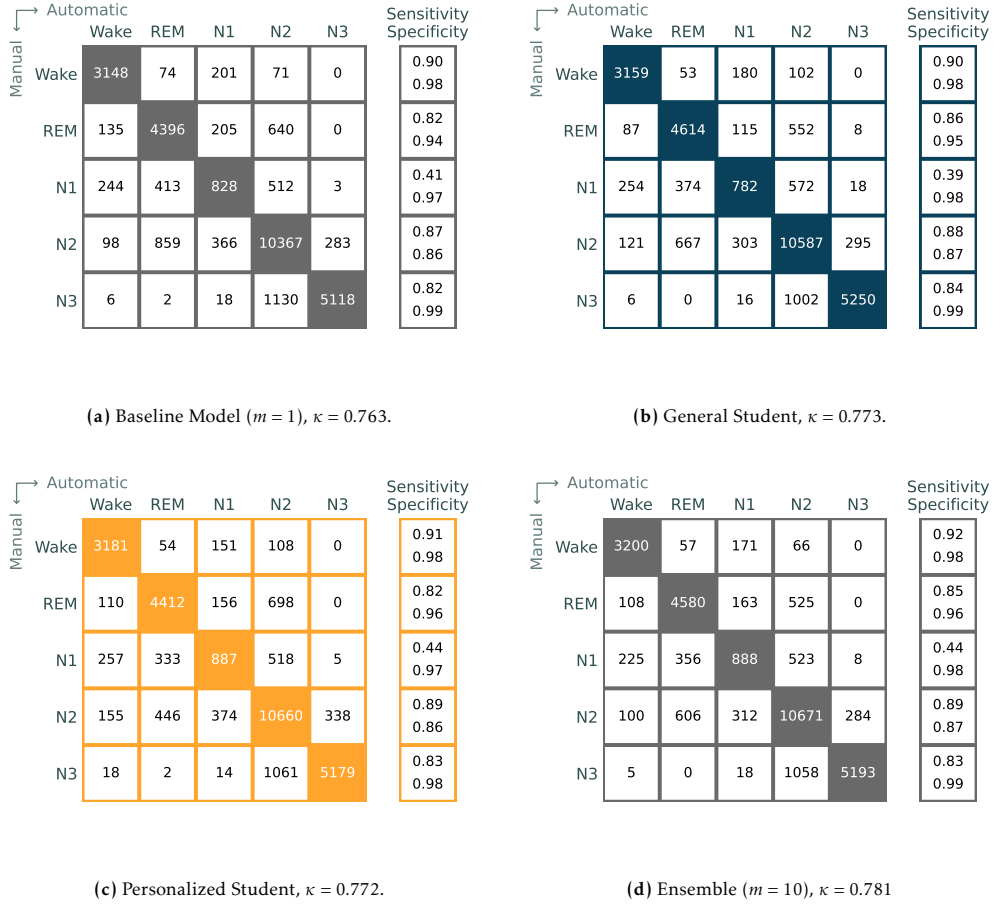


Figure E.6: Confusion matrices for comparison between manual scoring (along the y-axis) and automatic scoring (along the x-axis). The values represent the number of epochs and are computed over all long subjects for the five sleep stages. Furthermore, we include the specificity and sensitivity for each class on the right. We report the confusion matrices for (a) a single baseline model trained in a classical supervised manner, (b) a general student trained with pseudo-labels (computed by an ensemble of $m = 8$ models) on the additional 108 unscored nights of non-test subjects, (c) a personalised student trained with pseudo-labels (also computed by an ensemble of $m = 8$ models) on the additional 12 unscored nights associated with the test subject, and (d) the ensemble of $m = 10$ baseline models. Cohen’s kappa for the long subjects of the models is in the caption.

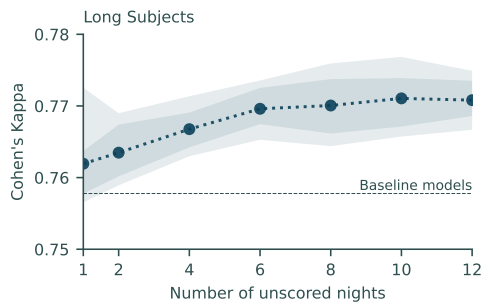


Figure E.7: Cohen's kappa for personalised students trained using the scored training nights along with a varying number of unscored nights from the test subject. We consider merely the performance on the 10 long subjects and use a teacher with $m = 5$. We report the mean along with 10 – 90% and 25 – 75% empirical confidence intervals in shaded areas.

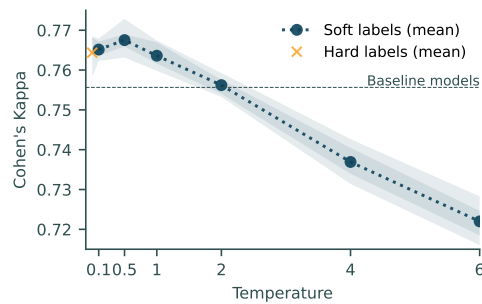


Figure E.8: Different choices of temperature, τ , when student is trained on all unscored data (120 nights) and the 15 scored training subjects. We report the performance over all 20 subjects and the points are the mean of 5 repetitions of the experiments with empirical confidence intervals in shades. Hard labels refer to one-hot encoded predictions by the teacher.

scorers are around 0.82 for this data set (Mikkelsen et al., 2021), we think that the improvement shown is close to the best-case scenario. We note that our set of baseline models is restricted to be models of identical architecture and training method but with different random initializations. On this basis, constructing ensembles using baseline models from various different network architectures, of various size, and with stacking are interesting future directions of research in order to improve the predictive performance of the ensembles even further. Furthermore, one can consider reducing the computational requirements at inference time by using cascades of models.

One benefit of mobile sleep monitoring is that the sleep analysis could conceivably be performed locally, for instance on a smartphone. In that case, it is beneficial to keep memory and computation requirements to a minimum, particularly after the model has been trained. Neural ensembles is a potentially very greedy approach which leads us to consider *knowledge distillation*, as a way to compress an ensemble into a single network. We find that the distillation is relatively successful, and a single network can inherit more than half the improvement in kappa value seen for the best teaching ensemble, but using only the same resources as the original baseline model. However, crucially, we find that this degree of improvement requires use of (unlabeled) recordings from the individual for which the model is needed. This is not necessarily a significant issue for mobile sleep monitoring (in which personal unlabeled data is easy to come by), but it is important to keep in mind. We find that the kappa value monotonically increases with the number of recorded nights from the individual, but even without any recordings from the individual, distillation is still able to recover about 40% of the improvement in kappa value, when recordings from other individuals are used.

Moving forward, we find that this is a promising approach for *ambulant sleep monitoring*, and shows an interesting way to benefit from the large amounts of unlabeled data which can easily be gathered in this way. We could imagine a process where the first n nights for a patient were uploaded to a central server in charge of performing the personalised distillation. Once the distillation was completed, the improved, personal model could be returned to the recording device, after which sleep scoring of improved quality

could be performed locally. This full process would require no manually scored labels for the patient. It will be interesting in future studies to see whether the personalised benefits could be even larger for more challenging data sets (such as elderly people), where the room for improvement is greater.

Acknowledgments

The work presented here was in part supported by the Innovation Fund Denmark, grant 7050-00007. We thank Lars N. Andersen for comments and suggestions, as well as T&W Engineering for supplied equipment.

Bibliography

- Ahn, S., Hu, S. X., Damianou, A., Lawrence, N. D., and Dai, Z. (2019). Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9155–9163. Cited on page 146.
- Anil, R., Pereyra, G., Passos, A., Ormandi, R., Dahl, G. E., and Hinton, G. E. (2018). Large scale distributed neural network training through online distillation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. Cited on page 146.
- Arnal, P. J., Thorey, V., Ballard, M. E., Hernandez, A. B., Guillot, A., Jourde, H., Harris, M., Guillard, M., Beers, P. V., Chennaoui, M., and Sauvet, F. (2019). The Dreem Headband as an Alternative to Polysomnography for EEG Signal Acquisition and Sleep Staging. *bioRxiv*, page 662734. Cited on page 146.
- Ba, J. L. and Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems*, volume 27, pages 2654–2662. Cited on pages 146.
- Berry, R. B., Brooks, R., Gamaldo, C., Harding, S. M., Lloyd, R. M., Quan, S. F., Troester, M. T., and Vaughn, B. V. (2017). AASM Scoring Manual Updates for 2017 (Version 2.4). *Journal of Clinical Sleep Medicine*, 13(5). Cited on pages 146 and 148.
- Boostani, R., Karimzadeh, F., and Nami, M. (2017). A comparative review on sleep stage classification methods in patients and healthy individuals. *Computer Methods and Programs in Biomedicine*, 140:77–91. Cited on page 146.
- Borup, K. and Andersen, L. N. (2021). Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation. In *Advances in Neural Information Processing Systems*, volume 34, pages 5316–5327. Cited on page 147.
- Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pages 535–541. Cited on pages 146.
- Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9630–9640. Cited on page 146.

- Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46. Cited on page 148.
- Fang, G., Song, J., Shen, C., Wang, X., Chen, D., and Song, M. (2019). Data-Free Adversarial Distillation. *arXiv preprint arXiv:1912.11006*. Cited on page 146.
- Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born Again Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1607–1616. PMLR. Cited on page 146.
- Gangstad, S. W., Mikkelsen, K. B., Kidmose, P., Tabar, Y. R., Weisdorf, S., Lauritzen, M. H., Hemmsen, M. C., Hansen, L. K., Kjaer, T. W., and Duun-Henriksen, J. (2019). Automatic sleep stage classification based on subcutaneous EEG in patients with epilepsy. *Biomedical engineering online*, 18(1):1–17. Cited on page 146.
- Garipov, T., Izmailov, P., Podoprikin, D., Vetrov, D., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, pages 8789–8798. Cited on page 154.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*. Cited on pages 146 and 151.
- Izmailov, P., Podoprikin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 2, pages 876–885. Cited on page 154.
- Koley, B. and Dey, D. (2012). An ensemble system for automatic sleep stage classification using single channel EEG signal. *Computers in Biology and Medicine*, 42(12):1186–1195. Cited on page 146.
- Lopes, R. G., Fenu, S., and Starner, T. (2017). Data-Free Knowledge Distillation for Deep Neural Networks. *arXiv preprint arXiv:1710.07535*. Cited on page 146.
- Micaelli, P. and Storkey, A. (2019). Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems*, volume 32, pages 9551–9561. Cited on page 146.
- Miettinen, T., Myllymaa, K., Westeren-Punnonen, S., Ahlberg, J., Hukkanen, T., Töyräs, J., Lappalainen, R., Mervaala, E., Sipilä, K., and Myllymaa, S. (2018). Success rate and technical quality of home polysomnography with self-applicable electrode set in subjects with possible sleep bruxism. *IEEE journal of biomedical and health informatics*, 22(4):1124–1132. Cited on page 146.
- Mikkelsen, K., Phan, H., Rank, M. L., Hemmsen, M. C., De Vos, M., and Kidmose, P. (2021). Sleep Monitoring Using Ear-Centered Setups: Investigating the Influence from Electrode Configurations. *IEEE Transactions on Biomedical Engineering*, 69(5):1564–1572. Cited on pages 147, 148, 152, 153, and 157.
- Mikkelsen, K. B., Kappel, S. L., Mandic, D. P., and Kidmose, P. (2015). EEG recorded from the ear: Characterizing the Ear-EEG Method. *Frontiers in Neuroscience*, 9:438. Cited on page 146.

- Mikkelsen, K. B., Tabar, Y. R., Kappel, S. L., Christensen, C. B., Toft, H. O., Hemmsen, M. C., Rank, M. L., Otto, M., and Kidmose, P. (2019). Accurate Whole-Night Sleep Monitoring with Dry-Contact Ear-EEG. *Scientific Reports*, 9(1). Cited on pages 146, 147, 148, 152, and 162.
- Mikkelsen, K. B., Tabar, Y. R., Toft, H. O., Hemmsen, M. C., Rank, M. L., and Kidmose, P. (2022). Self-applied ear-EEG for sleep monitoring at home. In *Proceedings of the 44th annual international conference of the IEEE engineering in Medicine & Biology Society (EMBC)*, pages 3135–3138. Cited on page 148.
- Mobahi, H., Farajtabar, M., and Bartlett, P. L. (2020). Self-Distillation Amplifies Regularization in Hilbert Space. In *Advances in Neural Information Processing Systems*. Cited on page 147.
- Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational Knowledge Distillation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Cited on page 146.
- Phan, H., Andreotti, F., Cooray, N., Chen, O. Y., and De Vos, M. (2019). SeqSleepNet: End-to-End Hierarchical Recurrent Neural Network for Sequence-to-Sequence Automatic Sleep Staging. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(3):400–410. Cited on pages 146, 148, and 162.
- Phan, H. and Mikkelsen, K. (2022). Automatic sleep staging of EEG signals: Recent development, challenges, and future directions. *Physiological Measurement*, 43(4). Cited on page 146.
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). FitNets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. Cited on page 146.
- Srinivas, S. and Fleuret, F. (2018). Knowledge transfer with jacobian matching. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 7515–7523. Cited on page 146.
- Stephansen, J. B., Olesen, A. N., Olsen, M., Ambati, A., Leary, E. B., Moore, H. E., Carrillo, O., Lin, L., Han, F., Yan, H., Sun, Y. L., Dauvilliers, Y., Scholz, S., Barateau, L., Hogg, B., Stefani, A., Hong, S. C., Kim, T. W., Pizza, F., Plazzi, G., Vandi, S., Antelmi, E., Perrin, D., Kuna, S. T., Schweitzer, P. K., Kushida, C., Peppard, P. E., Sorensen, H. B., Jennum, P., and Mignot, E. (2018). Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy. *Nature Communications*, 9(1):5229. Cited on page 146.
- Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020). What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*. Cited on page 146.
- van Gilst, M. M., van Dijk, J. P., Krijn, R., Hoondert, B., Fonseca, P., van Sloun, R. J. G., Arsenali, B., Vandenbussche, N., Pillen, S., Maass, H., van den Heuvel, L., Haakma, R., Leufkens, T. R., Lauwerijssen, C., Bergmans, J. W. M., Pevernagie, D., and Overeem, S. (2019). Protocol of the SOMNIA Project: An Observational Study to Create a Neurophysiological Database for Advanced Clinical Sleep Monitoring. *BMJ Open*, 9(11). Cited on page 146.

Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7130–7138. Cited on page 146.

Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Cited on page 146.

Supplementary material

E.I Experimental Details

In the following we present some details on the experimental setting and the hyperparameters used for fitting our models.

E.I.1 Hyperparameters and Training Setup

All our models are based on a PyTorch implementation of the SeqSleepNet architecture introduced in Phan et al. (2019). Our code is publicly available on GitHub: <https://github.com/Kennethborup/SeqSleepNet>. We set our sequence length to $L = 20$, and consider input epochs of length $T = 29$ with $F = 129$ frequency bins and a single $C = 1$ channel spectrogram. More specifically, we only use the LR derivation from Mikkelsen et al. (2019). We halve the learning rate when the validation loss has not improved for 50 training epochs and employ early stopping after a minimum of 700 training epochs. Training is done with the Adam optimizer, with learning rate of 10^{-3} , momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, and weight-decay of 10^{-4} . We define a training epoch as the number of gradient updates required to pass through the scored training set (15 subjects each with 4 nights) once, and when training on larger datasets (due to unlabeled data), we still consider a single training epoch to be this amount of steps, but will only pass through a random subset of the samples of the larger dataset. This way, the amount of training epochs are comparable across models, and we limit it to a maximum of 1500 training epochs. However, due to early stopping, training is often effectively stopped at less than 1000 training epochs.

E.I.2 Ablation of Temperature

When performing distillation we fix $\tau = 1$ and $\alpha = \frac{1}{2}$ for all experiments. However, in Figure E.8 we show the mean Cohen’s kappa for five repetitions of distillation with different choices of temperature τ , where the teacher model consists of 5 baseline models and is identical across all experiments. We let $\mathcal{D}_{\text{pseudo}}$ be all unscored data (120 nights) and let \mathcal{D}_{gt} be the 15 scored training subjects, irrespective of subject type (long/short). We see that for $\tau \leq 2$ we observe an increase in Cohen’s kappa, but for τ between 0.1 and 1 the differences are small. Finally, we also observe that using hard pseudo labels (i.e. one-hot encoded pseudo-labels with 1 at the entry of the largest class-probability) yields comparable performance to the best choices of τ , and can be a simple alternative to soft labels. This observation suggests that it is the utilization of additional data that is the key to the improved performance by distillation, rather than the implicit properties of the soft labels.

E.I.3 Discarding Manually Scored Labels for Distillation

During the distillation part of our training, we have the option to completely discard all manually obtained labels and merely rely on the pseudo-labels produced by the teacher model. In Figure E.9 we show the performance (across all subjects) of student models trained with pseudo-labels instead of the ground-truth labels on the nights which were

scored manually. Thus, we discard the manual labels, and effectively the distillation procedure is now fully unsupervised. However, these models slightly under-perform the models trained using the original manually produced labels. Furthermore, from Figure E.9 we also observe that without any additional data, simply using the soft labels is not sufficient to improve model predictive performance.

E.I.4 Performance on Short Subjects

In Section E.3.2 and Figure E.4b we presented the predictive performance of personalised student models on *long subjects* when trained with unlabeled data from these test subjects. In Figure E.10 we report the performance on the *short subjects* in the exact same experiments. The lack of improvement by the yellow line is due to the fact that these models are trained identically to the baseline models, and we expect them to perform equally. However, we also observe an higher relative improvement in performance than for the long subjects as long as the unlabeled data contains the 4 test nights as unlabeled training data.

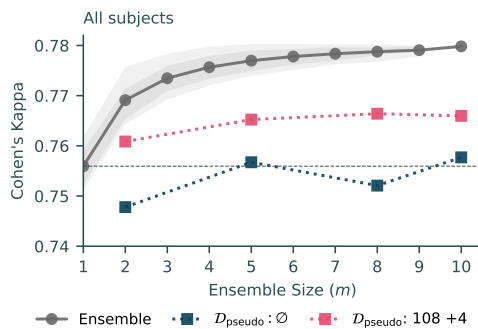


Figure E.9: Performance of students on all subjects, when the original labels of the scored nights are discarded and pseudo-labels are used on this dataset instead. We generally observe a slightly worse performance than when the original labels are used.

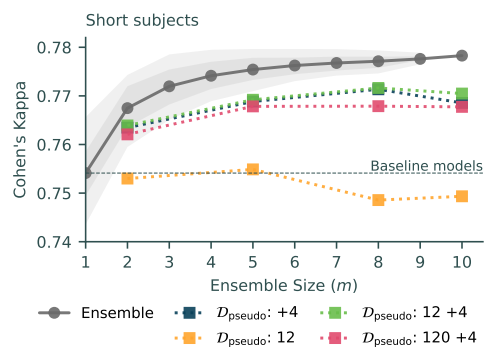


Figure E.10: Performance of personalised students on the short subjects in the same experiments as in Figure E.4b. Note that these subjects do not have the additional 12 unscored nights, and any improvement observed here must be attributed to the use of pseudo-labeled test nights.