# Kernel Meta-Learning by Leveraging Natural Data Assumptions

*John Isak Texas Falk*

A dissertation submitted in partial fulfillment

of the requirements for the degree of

**Doctor of Philosophy**

of

**University College London**.

Department of Computer Science

University College London

November 28, 2023

# Abstract

Data representation is integral to meta-learning and is effectively done using kernels. Good performance requires algorithms that can learn kernels (or feature maps) from collections of tasks sampled from a meta-distribution. In this thesis we exploit natural assumptions on the meta-distribution to design meta-kernel learning algorithms, leading to two novel state-of-the-art (SOTA) meta-classification and regression algorithms. The first method, Meta-Label Learning (MeLa) [Wan+22] leverages the meta-classification assumption that each task is generated from a global base dataset by randomly sampling $C$ classes, anonymising the labels, then sampling $K$ instances from each class. Anonymity of task-labels prohibit us from pooling task-instances with the same global class. MeLa recovers, in some cases perfectly, the underlying true classes of all task-instances allowing us to form a standard dataset and train a feature map in a supervised manner. This procedure leads to SOTA performance while being faster and more robust than alternative few-shot learning algorithms. For meta-regression the notion of global classes is not well-defined. In Implicit Kernel Meta-Learning (IKML) [FCP22] we leverage the assumption that the optimal task-regressors belong to an RKHS with a kernel that is translation-invariant. We learn such a kernel from a kernel family characterized by a neural network through a pushforward model using Bochner's theorem. The model is trained by optimizing the meta-loss with random feature kernel ridge regression as the base algorithm. IKML achieves SOTA on two meta-regression benchmarks while allowing to trade accuracy for speed at test-time. We provide a bound on the excess transfer risk, allowing to specify the least number of random features necessary to achieve optimal generalization performance.

# Impact Statement

This thesis explores how to design meta-learning algorithms that leverage natural assumptions about the data generating process. As such it has immediate impact, both towards furthering the field and challenging common meta-learning assumptions, and applications in industry and government.

Academically, the algorithms contributes to pushing the state-of-the-art while being theoretically motivated. Furthermore, they address questions which have been left unanswered by the literature; for MeLa they show that pretraining is possible even without access to global labels which makes pre-training useful even when this assumption is not met (as in the initial formulation of the few-shot classification setting). For IKML, it provides a framework for learning invariances and so is applicable to many settings at once, while in particular showing improvement in the few-shot regression setting. The theoretical bound provided is of independent interest and the tools used could be useful to the few-shot and meta-learning communities at large. Finally, the kernel learning approach of IKML could be applied to other settings where learning the kernel would be of interest, such as Bayesian optimization and learning dynamical systems in the form of the dynamic mode decomposition and similar techniques.

The industrial potential benefits of this thesis are several, we enumerate them for MeLa and IKML independently. MeLa allows pre-training when local labels are anonymous which may occur in practice for example when the labelling is done by a user without a prespecified list of labels (or when the labels are consistent but wrong such as when a user swaps sweater and jumper for tagging an image of a jumper). The main impact for this would be in social platforms and personalized tagging, for example smart apps, and so has clear commercial application, especially with increasing focus on the user and the increasing prevalence of smartphones in the world. IKML on the other hand could have direct impact to multi-task and meta-learning regression problems, one extremely important example which is prediction related to energy supply and demand together with energy market forecasts, such as in spot markets.

# Acknowledgements

A doctorate is a long journey filled with ups and downs. I would not have been able to finish this journey without the help of many people to who I am indebted and have helped me along the way.

To start, I would like to thank my two supervisors, Massi and Carlo. Massi has an enthusiasm for research and science which is matched by very few people I know with a vast knowledge of the field of machine learning. Carlo has an eagle eye of the current development of the field matched with a rigorous treatment and investigation of the former, he mentored me during my first couple of years of the doctorate and made me interested in the world of kernels and influenced my taste in research greatly. Together you both pushed me when I needed it and through both of your guidance I grew as a researcher and a person. I would also like to thank Professor Guedj and Professor Álvarez for taking on the roles as examinors of my viva, the corrections greatly improved the final version of this thesis.

Secondly, thanks to everyone in our group that has been part, or just passed through, UCL or IIT. Thanks to Ricardo Grazzi, Luca Franceschi, Leonardo Cella and Arya Arkhavan for many evenings together discussing machine learning and beyond, Saverio Salzo for always having time to aid me with his deep knowledge of optimization, my collaborator Ruohan Wang for imparting me with his knowledge of deep and few-shot learning and programming, Pietro Novelli and Vladimir Kostic for discussions and partying at NeurIPS 2022, my first flat mate in Genoa Dimitri Meunier and the crew at UCL, Giulia Luise and Marco Ciccone. I owe you all many laughs and interesting ideas.

Thirdly, the do-not-circulate group, Daniel Lengyel, Michalis Lazarou, Janith Petangoda, Arinbjorn Kolbeinsson, Kate Highnam, when I was down you made my days better.

Fourthly, my two mentors during my internship at BenevolentAI, Millie Zhao and Qi Guo, and everyone else that made my short stay there great. The internship opened my eyes to the field of genetics and biology and their intersection with machine learning.

Fifthly, my dear friends Harry Booth and Liza Crompton-Prall, who let me and Berfin live with them during COVID-lockdown, you are great and I am grateful to have you in my life.

Sixthly, my family, my dad Björn, my mom Kicki, and my siblings Joel, Axel and Lisa, who always are there for me when I need them. Thank you, this would not have been possible without you.

Finally, my girlfriend Berfin Simsek. I love you. You make me happy and I know that I can count on you to support me when I need it. Thank you for always being by my side and lifting me up. For many days together and to many more.

If there are people that I have forgotten to mention, that is on me and not on you.

## Full Output of the PhD

The thesis is based on the works of [Wan+22] and [FCP22] centred around feature learning in the meta-learning setting. Other works produced during the PhD but outside the scope of the thesis are [Len+20; Gra+22; Fal+23a; Fal+23b].

# Contents

# List of Figures

# List of Tables

# Nomenclature

**Abbreviations**

| | |
|---|---|
| SOTA | State-of-the-art |
| ADAM | Adaptive Moment Estimation |
| SGD | Stochastic Gradient Descent |
| DeepEMD | Deep Earth Mover's Distance |
| FEAT | Few-Shot Embedding Adaptation |
| FRN | Feature Map Reconstruction Network |
| LEO | Latent Embedding Optimization |
| MeLa | Meta Label Learning |
| ProtoNet | Prototypical Network |
| R2D2 | Ridge Regression Differentiable Discriminator |
| RFS | Representations for Few-Shot Learning |
| TASML | Task-Adaptive Structured Meta-Learning |
| ANP | Attentive Neural Process |
| GMKL | Gaussian Multiple Kernel Learning |
| GO | Gaussian Oracle |

| | |
|---|---|
| IKML | Implicit Kernel Meta-learning |
| LSBR | Least Square Biased Regularization |
| MAML | Model-agnostic Meta-Learning |
| RF | Random Features |
| iid | Independent and identically distributed |
| RKHS | Reproducing Kernel Hilbert Space |
| KRR | Kernel Ridge Regression |
| MAE | Mean Absolute Error |
| RMSE | Root Mean Square Error |
| SMAPE | Symmetric Mean Absolute Percentage Error |
| MERM | Multitask Empirical Risk Minimization / Minimizer |
| RERM | Regularized Empirical Risk Minimization / Minimizer |
| FSL | Few-shot Learning |
| GFSL | Generalized Few-shot Learning |
| GLS | Global Label Selector |
| ITL | Independent Task Learning |

**Notation**

| | |
|---|---|
| $x, y, z$ | Input, output, data pair |
| $y_\rho^{(j)}$ | $j$'th class associated with few-shot classification task associated with $\rho$ |
| $x_i^{(j)}$ | $i$'th instance sampled conditionally on the label $y_\rho^{(j)}$ |
| $\hat{y}$ | Predicted output |

| | |
|---|---|
| $\mathcal{X}$ | Input space |
| $\mathcal{Y}$ | Output space |
| $\mathcal{Z}$ | Data space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ |
| $\mathcal{Z}^n$ | Cartesian $n$-product of set $\mathcal{Z}$ |
| $\mathcal{H}$ | Hypothesis space, RKHS |
| $\Theta$ | Meta-parameter space |
| $\Omega$ | Domain of parameters, domain of frequency, regularizer |
| $T$ | Number of tasks |
| $n$ | Number of datapoints |
| $n_{\text{tr}}, n_{\text{val}}$ | Number of train / validation data points |
| $K, k$ | Number of ways / local classes |
| $M$ | Number of random features |
| $T_{\text{iter}}$ | Number of iterations |
| $V, g_v, N_v$ | Number of clusters, centroid of cluster $v$, number of points assigned to cluster $v$ |
| $\mathcal{P}(\mathcal{A})$ | Set of distributions on set $\mathcal{A}$ |
| $\rho$ | Distribution over data points |
| $\mu$ | Meta-distribution |
| $\pi_\mu$ | Marginal distribution reconstructed from $\mu$ generating few-shot classification tasks |
| $\mathcal{N}$ | Gaussian distribution |
| $\delta_S$ | Dirac delta over set $S$ |

|  |  |
|---|---|
| $\psi\#\mathcal{N}$ | Push-forward of $\mathcal{N}$ by function $\psi$ |
| $D \sim \rho^n$ | $D$ is a set of $n$ iid samples from $\rho$ |
| $\rho_Y$ | Classes associated with few-shot learning task $\rho$ |
| $\mathcal{D}$ | Set of all datasets |
| $\mathcal{T}$ | Set of tasks |
| $D(\mathcal{T}), D_{\text{global}}$ | Dataset of flattened set of tasks $\mathcal{T}$ |
| $D^{\text{tr}}$ | Train set |
| $D^{\text{val}}$ | Validation set |
| $\tau$ | Frequency-distribution |
| $\phi_\theta$ | Feature map with parameters $\theta$ |
| $\phi_\theta^{\text{pre}}$ | Feature map with parameters $\theta$ (pretraining) |
| $\phi_\theta^{\text{sim}}$ | Feature map with parameters $\theta$ (representation learning) |
| $\phi_\theta^*$ | Feature map with parameters $\theta$ (meta-fine tuning) |
| $\phi_\theta(D)$ | Dataset of mapped inputs, $\phi_\theta(D) = (\phi_\theta(x), y)_{(x,y)\in D}$ |
| $k_\phi, m_\phi$ | Memory and computational complexity of evaluating a neural network $\phi$ |
| $\psi$ | Push-forward function |
| $\phi(x, \omega)$ | Integral feature map of IKML with instance $x$ and frequency $\omega$ |
| $\ell(y, \hat{y})$ | Supervised loss function applied to $(y, \hat{y})$ |
| $\ell_{\text{ce}}$ | Cross-entroy loss function |
| $L(\theta, D)$ | Meta-loss of meta-parameters $\theta$ applied to dataset $D$ |
| $\hat{L}$ | Estimator of meta-loss |

$\mathcal{R}(f), \mathcal{R}_\rho(f)$ Supervised learning risk of function $f$ (with respect to distribution $\rho$)

$\hat{\mathcal{R}}(f), \hat{\mathcal{R}}(f, D)$ Empirical supervised learning risk of function $f$ (with respect to dataset $D$)

$\mathcal{E}(\theta)$ Transfer risk of $\theta$

$\hat{\mathcal{E}}_T(\theta)$ Empirical transfer risk of $\theta$ for $T$ tasks

$\mathcal{E}_{\text{GLS}}$ Meta-risk when using GLS as algorithm

$L(\theta, S, D)$ Meta-loss of $\theta$ over task $D$ when using random feature sample $S$

$\tilde{L}(\theta, S, D)$ Meta-loss of $\theta$ over task $D$ when using random feature sample $S$ using $D$ for train and validation set

$\mathcal{E}_M$ Meta-risk for IKML when using random feature sample $S$

$\mathcal{E}(\theta, \mu)$ Transfer risk of $\theta$ with respect to meta-distribution $\mu$

$\theta, \theta^*, \hat{\theta}$ Meta-parameter, optimal meta-parameter, meta-parameter estimator

$A_\lambda, A, w$ Algorithm mapping from datasets to estimators (with regularization strength $\lambda$), inner algorithm

$A_{\text{KRR}}, w_{\text{KRR}}$ KRR algorithm

$\text{GLS}(W, \theta, D)$ Global Label Selector algorithm with parameters $W, \theta$ applied to dataset $D$

$W_T^{\text{GLS}}, \theta_T^{\text{GLS}}$ Weights and representation parameters of GLS minimizing the empirical transfer risk of $T$ tasks

$W_N^{\text{pre}}, \theta_N^{\text{pre}}$ Weights and representation parameters of pretraining minimizing the empirical risk of $T$ flattened tasks into $N$ samples

$W[Y], W[\rho_Y]$ Rows of $W$ corresponding to index in set $Y$ or support $\rho_Y$

| | |
|---|---|
| $\nabla_\theta f$ | Gradient of $f$ with respect to $\theta$ |
| $K$ | Kernel function |
| $K_\tau$ | Kernel parameterized by frequency-distribution $\tau$ |
| $K_{\hat{\tau}_\theta S}$ | Empirical random feature kernel constructed from random feature sample $S$ sampled from $\tau_\theta$ |
| $\|x\|$ | Norm of an element $x$ |
| $\|X\|_F$ | Frobenius norm of matrix $X$ |
| $\langle x, y\rangle$ | Scalar product between $x$ and $y$ |

Chapter 1

# Introduction

Contemporary society generates data on a scale previously unseen in the history of humanity [Gor+08]. With social media we are generating data in form of images and text, while digitalization in government and industry generates data associated with their operations and procedures. Machine learning algorithms make use of datasets in order to generate insights (unsupervised learning), take optimal decisions (reinforcement learning) or to predict the outcome for new, similar datapoints (supervised learning). In this thesis we will focus on the last part, supervised learning [Vap91], in the context of *meta-learning*.

In supervised learning we have access to a dataset of input-output pairs from a data generating process. Using prior knowledge about the data-generating process and problem at hand we select an algorithm. This algorithm takes the dataset and infers a prediction rule which takes inputs and maps them to outputs. Three key properties that a supervised learning algorithm should have is that 1) the algorithm is computationally efficient, 2) the inferred prediction rule can be evaluated efficiently, 3) the inferred prediction rule has good performance, meaning that the prediction rule generalizes to unseen data from the same data generating process.

In the real world, supervised learning is typically applied or used as follows. A practitioner wants to learn a rule which maps from inputs to outputs from a dataset. The practitioner first selects a loss which encodes how bad predicting $\hat{y}$ for some input $x$, when the true output is $y$. To find this rule the practitioner uses a supervised learning algorithm. The algorithm finds the best (approximate) fit to an objective which is a sum of losses, where each loss depends on a datapoint $z = (x, y)$ from the dataset and the prediction $\hat{y}$ gotten by applying the rule to $x$. The objective additionally incorporates a regularizer which encodes prior knowledge about what rules are reasonable for the problem at hand. The rule which minimizes the objective is then tested on holdout data or deployed in the real-world to accomplish some goal, hopefully leading to satisfactory performance.

The above workflow has been the backbone of machine learning since its conception and is in some cases optimal from the point of view of statistical learning theory. Precisely, the regularized empirical risk minimization (RERM) framework of [Vap91], which defines a family of algorithms, is many times the best you can do under standard assumptions of statistical learning theory in terms of at what rate the generalization error goes down in the size of the dataset [SB14; CD07; HK19][1]. Many algorithms such as regularized maximum likelihood estimators fall into this framework, and in particular the kernel ridge regression algorithm (KRR) [CD07].

Meta-learning is a paradigm that generalizes supervised learning; it aims to learn a supervised learning algorithm from a collection of supervised learning problems which generalizes to new supervised learning problems (the name comes from the fact that it operates on a meta-level compared to supervised learning) [VD02; Hos+22]. However, we also know from the previous paragraph that in the sense of standard assumptions on the supervised learning problem, we cannot improve upon RERM. Additionally KRR has been shown to be successful in practice and not just a theoretical construct [CD07; RCR17].

If there are supervised learning algorithms which are theoretically sound, which are successful both in theory and practice, why not just turn any meta-learning problem into a supervised learning problem by pooling all of the data and proceed by using RERM? Below we explain the issue, but also the benefits,

[1]Technically, it is minimax optimal (potentially within a log-factor).

of this approach.

Although the size and number of datasets continue to increase, the data is typically not added to pre-existing datasets. Instead, it is distributed over many similar but distinct problems (corresponding to the datasets and natural processes generating these datasets), which we call *tasks* [Car97]. Supervised learning cannot handle this situation correctly as supervised learning has no concept of multiple tasks. A naive approach would be to employ a supervised learning algorithm on each task independently, so called Independent Task Learning (ITL). If tasks are related this approach is not ideal since it does not leverage the relationship and similarity between tasks at all and thus throws away information. Learning from the tasks in a way that takes these relationships into account should lead to better performance than assuming them to be completely independent [Rud17].

A common way to tackle the problem of having multiple tasks is that of multi-task learning. Multi-task learning considers a collection of tasks, with corresponding datasets, which we want to perform well on. Compared to standard supervised learning we may use all of the datasets available to learn a rule for each task which may outperform the rules learned through independent task learning. In practice, this is often done by coupling the predictors of each task through a regularizer on the multi-task RERM problem. In this case the rules are learned by minimizing the average error over all tasks and datapoints of each task in addition to a regularizer which encodes prior belief about how the tasks are related [EP04].

While multi-task learning resembles meta-learning, there are important differences which makes them different: multi-task learning assumes that the tasks given are fixed and that we want to generalize on new data from the same tasks. In comparison, meta-learning generalizes this by assuming that the tasks are not fixed, but themselves generated from nature through a hierarchical process which ties the tasks together statistically [Bax98]. Thus meta-learning is a strictly harder problem since we do not know what kind of tasks we may see in the future (although it is sampled from the same meta-distribution) and thus have to generalize within and across tasks, while in multi-task learning the tasks are fixed and finite which means we only need to generalize within tasks, not across.

Datasets in the real world increasingly take on the form of many small datasets / tasks which are grouped together due to for example being collected in the same way. Thus, as opposed to one dataset $D$ of input-output pairs of datapoints, we get a collection of datasets $(D_t)_{t=1}^T$ where each dataset $D_t$ of input-output pairs is related but distinct to the other datasets $D_{t'}$ where $t \neq t'$. One reason for this is personalization of data due to social platforms and apps together with availability of smartphones and internet [Gre22]: a user on an online platform[2] generates data through interacting with the platform and we would like to predict some value of interest. One example would be the utility scores of potential images to show the user given the images the user has scored in the past, but in general this formulation works for any type of content. Call any image (or content) $x$ and the score $y$. To put this in the context of meta-learning, in this case, we have $T$ users and each user generates a dataset $D_t$ of content-score pairs $(x, y)$. We will use this problem as a prototype for meta-learning.

For each user $t$ we want to learn a rule $f_t$ which accurately predicts the right score for some content $x$, so that $f_t(x) \approx y$. A simple way to do this would be to just use a supervised learning algorithm such as RERM to get $T$ rules $(f_t)_{t=1}^T$, one for each user, which corresponds to ITL. If we have a lot of data for each user, this strategy may perform well. However, if the user has not scored a lot of content then the rule will typically not perform well and this strategy will fail. While each user is unique, there are often many similarities between them. The aim of meta and multi-task learning is to leverage these similarities in order to learn rules $(f'_t)_{t=1}^T$ which perform better than the independent task learning rules $(f_t)_{t=1}^T$ which do not take these

[2]For example an online marketplace or a social media platform.

similarities into account. Formally, this way of learning comes down to learning an *inductive bias* [Bax00] of an algorithm suitable for the type of problems we are considering, here scoring of content for users

Now we introduce a supervised learning algorithm $A(\theta, \cdot)$ parameterized by $\theta$ which takes as inputs datasets $D$ and outputs rules $f'_\theta$. The inductive bias in this case is specified by $\theta$, so different $\theta$'s lead to different inductive biases. In meta-learning, we can learn $\theta$ using a similar approach to RERM. We choose the inductive bias by finding the $\theta$ which minimizes the meta-objective corresponding to the average error over all available tasks and a regularization term which takes into account $\theta$ and how we think the tasks are related. If $f'_{\theta,t} = A(\theta, D_t)$, then an example of a regularizer is a term which penalizes rules $f'_{\theta,t}$ which are far from the average of the rules $\frac{1}{T}\sum_{t=1}^{T} f'_{\theta,t}$[3] [EP04]. In this case, typical users with a small amount of data benefit from other users with more data. Finally, the difference between multi-task and meta-learning comes down to what we want to generalize to. For multi-task learning, we want the learned rules to perform well on the users we have, while in meta-learning we may have a completely new user at test time with dataset $D$ for which we want to learn a rule $f'$, given the collection of tasks $(D_t)_{t=1}^{T}$. Finally, while the setting is usually called *meta-learning* [FAL17], it is also known as *learning-to-learn* [TP98].

In meta-learning, each task consist of a dataset sampled from some distribution and a loss particular to the task. In practice this loss is often fixed across tasks due to the metric of success being shared by all tasks. Due to the availability of data and interest from the computer vision community, meta-learning has most prominently been developed for and applied to the problem of few-shot image classification [Dhi+20; FAL17; RL17]. We will go into detail about the few-shot learning problem in chapter 2 but quickly outline the field here. Few-shot image classification assume that tasks come from an underlying base dataset with a large number of classes, for example the ImageNet

[3]We assume that all operations here are well-defined.

(LSVRC2013) dataset [Den+09] which is a dataset, scraped from the internet, of images belonging to 1000 classes of common objects in the real world. Each task is sampled by first sampling $C$ classes at random from this dataset and then sampling $N$ data points from each of the $C$ classes, where $C$ and $N$ are natural numbers which specify the few-shot classification setting. The goal is to do image-classification for such tasks by using meta-learning to learn a supervised learning algorithm. We use this as a starting point for observing and discussing the current state of the field.

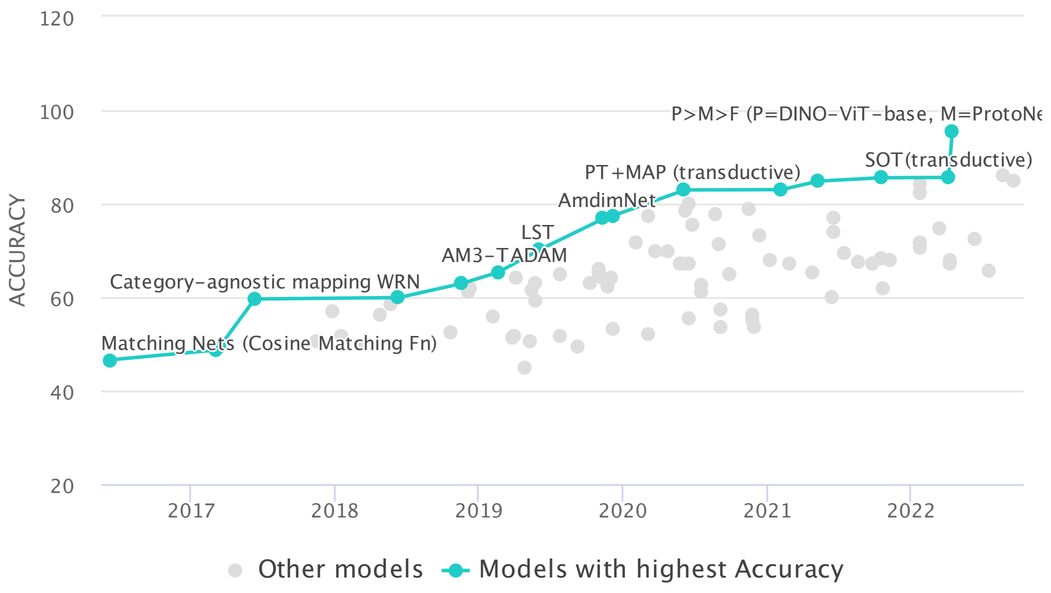


Figure 1.1: Evolution of state-of-the-art (SOTA) on 5way-1shot miniImagenet few-shot classification [pap22]. The cyan curve is the current SOTA at the point in time indicated on the $x$-axis, with each dot being the performance of an algorithm and the string the name of the algorithm. Grey points are other algorithms not achieving SOTA. See [pap22] for an exact description of the algorithms in the chart. The plot starts at July 2016 and ends at July 2022. SOTA accuracy on this benchmark has climbed from below 50% to over 80% in less than 5 years.

While the performance of meta-learning algorithms on few-shot image classification benchmarks have steadily improved over time (see Fig. 1.1 and the landmark papers [Vin+16; SSZ17; Rus+18; Lee+19] for a rough timeline) this improvement is hard to attribute to an improvement of the meta-learning algorithms themselves. This is due to different papers using slightly different evaluation schemes and rules for what is allowed in the benchmarks. In particular

- use of better feature maps for images in more recent years, often with additional pre-training (e.g. [Rus+18]),
- using external data sources to improve performance, for example in the form of pretrained models [Hu+22],
- different works using vastly different computational budgets resulting in different number of iterations, batches and hyper-hyperparameter and validation schemes when benchmarking the algorithms against each other.

Furthermore, recent papers have questioned the reliance on complicated adaptation algorithms, stressing that a simple fine-tuning algorithm together with a good pre-trained feature map achieves SOTA [Tia+20; Dum+21; Dhi+20]. Similarly, [Rag+20] shows that MAML [FAL17] primarily learns a good feature embedding and rely very little on fine-tuning on tasks to achieve good performance. Furthermore, most benchmarks are restricted to the setting of few-shot image-classification and it is unclear to what extent the improved benchmarks generalize from image classification to other domains. We can summarize the observations as follows:

- Modern meta-learning algorithms do not work well because they learn to adapt but because they learn good feature representations of the data, and
- it is unclear how much meta-learning improves over standard transfer learning when using pre-training.

The two above points are the motivating questions for chapters 3 and 4 of this thesis. For the first point, we note that while pretraining has become an integral step in basically all SOTA few-shot image classification algorithms, the original setting of few-shot learning does not permit pretraining out of the box. This is due to the setting which assumes access to tasks with *local* rather than *global* labels. To pretrain a feature map given a set of tasks $(D_t)_{t=1}^T$ it is necessary to aggregate the tasks into one dataset which is not possible since the labels of one task only make sense in the context of that task, even if the tasks share some global classes. This means that pretraining using supervised learning, while effective, is not directly possible if we follow the strict definition of the few-shot learning setting. However, since we know that pretraining is very effective, we still want to use something which mimics what is done in practice when the global labels are unavailable, such as using auxiliary labels as we do in chapter 3.

Secondly, the pretraining procedure by aggregating all of the tasks into one dataset is in practice only possible for classification. In the setting of meta-regression it is not clear how to perform pretraining. Instead we consider how to define a well-suited inductive bias for natural signals in the meta-regression setting. Taking inspiration from Fourier representation of continuous signals, in chapter 4 we create an algorithm which selects from a family of function spaces a hypothesis space, adapting the resolution of the signals or functions we use to learn automatically to the data. We do this using a kernel formulation which allows for working in the feature space implicitly and learn this feature map.

## 1.1 Organization of Thesis

The thesis is organized as follows; in chapter 2 we introduce the necessary concepts and definitions, first *supervised learning* in Sec. 2.2, then *meta-learning* in Sec. 2.3, an overview of few-shot image classification and pre-training in Sec. 2.4, and a literature review of the meta and few-shot learning field in Sec. 2.5. In chapter 3 we introduce Meta Label Learning (MeLa), based on the pre-print [Wan+22] which is an extended version of [WPC21], a clustering method for image few-shot classification which can recover the underlying global labels when only given tasks with local labels. In chapter 4 we introduce Implicit Kernel Meta-Learning (IKML), based on the paper [FCP22], a framework for meta-learning a kernel of a specific form using a generative modelling formulation. Finally, in chapter 5 we conclude with a summary of contributions, conclusions and future directions.

Chapter 2

# Background

## 2.1 Summary

In this chapter we outline the background necessary to read the thesis and understand the research context within which it exists. We build up the necessary background to present and define the setting of meta and few-shot learning; providing definitions and settings from machine learning that will be used in the meta-learning setting. Additionally we provide a literature review of the meta and few-shot learning field and tangential fields relevant to the subsequent chapters of the thesis.

## 2.2 Supervised Learning

Intuitively, supervised learning concerns itself with how to learn an input-output function $f$ from a collection of inputs and outputs $(x, y)$ such that the chosen function $f$ is approximately equal to the function $f^*$ that best describes the relationship between $x$ and $y$, so that $f(x) \approx f^*(x)$ for all $x$ that are typically seen.

Formally, given an input set $\mathcal{X}$ and output set $\mathcal{Y}$, a supervised learning problem, which we call a *task*, is characterized by a data generating distribution $\rho \in \mathcal{P}(\mathcal{Z})$, where $\mathcal{P}(\mathcal{Z})$ is the set of all distributions on $\mathcal{Z}$, on the joint set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, called the *data space*, and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ measuring prediction errors. The goal for a supervised learning problem is to find a map

$f : \mathcal{X} \to \mathcal{Y}$ minimizing the *risk*

$$\mathcal{R}_\rho(f) = \mathbb{E}_{(x,y)\sim\rho}\ell(f(x), y). \tag{2.1}$$

In practice, the data generating distribution is unknown and only a finite number $n$ of examples $D = (x_i, y_i)_{i=1}^n$ sampled iid from $\rho$ are available (denoted by $D \sim \rho^n$). Let $\mathcal{Y}^\mathcal{X} = \{f : \mathcal{X} \to \mathcal{Y}\}$ be the space of all functions from $\mathcal{X}$ to $\mathcal{Y}$ and $\mathcal{D}$ be the space of all datasets of any size on $\mathcal{Z}$, that is $\mathcal{D} = \cup_{n\geq 1}\mathcal{Z}^n$[1].

To learn $f$ we need to define a rule which codifies how we pick $f$ given a dataset $D$. We call such a rule a *learning algorithm* and we will sometimes refer to such a rule as just an *algorithm*. A learning algorithm is a function $A : \mathcal{D} \to \mathcal{Y}^\mathcal{X}$, mapping datasets $D \in \mathcal{D}$ to functions $h : \mathcal{X} \to \mathcal{Y}$ which will be used as candidate solutions to (2.1). Typically, learning algorithms are parametrized as $A(\cdot) = A(\theta, \cdot)$, by a vector of so-called *hyperparameters* $\theta \in \Theta$ for a choice of meta-parameter space $\Theta$, that allow to adapt the algorithm to the specific problem. Typical examples of hyperparameters include the regularization constant in Tikhonov regularization or the number of iterations and iteration scheme of an early-stopping procedure [see e.g. YRC07], but more complex models such as modern neural network architectures may have tens of hyperparameters to tune or more (as an example see the optional arguments of the PyTorch documentation on the MLP architecture). For a learning algorithm, we call the space of possible candidate solutions the *hypothesis space* and typically denote it by $\mathcal{H}$, and the goal of supervised learning is to design such learning algorithms that are efficient in terms of both computation and generalization.

As the true risk is typically not available, we use the *empirical risk*, where for a function $f : \mathcal{X} \to \mathcal{Y}$ and a dataset $D = (x_i, y_i)_{i=1}^n \in \mathcal{D}$ the empirical risk is defined to be

$$\hat{\mathcal{R}}(f, D) = \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i), \tag{2.2}$$

[1]For a cartesian product of sets $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_m$ the power notation $\mathcal{Z}^n$ is defined to be $\mathcal{Z}_1^n \times \cdots \times \mathcal{Z}_m^n$. In our case, $\mathcal{Z}^n = \mathcal{X}^n \times \mathcal{Y}^n$.

as a proxy for the true risk. We usually split a dataset $D$ into two disjoint datasets, $D = (D^{\text{tr}}, D^{\text{val}})$, where $D^{\text{tr}}$ is called the *train set* and $D^{\text{val}}$ is called the *validation set*. Let $n_{\text{tr}}$ and $n_{\text{val}}$ be the number of instances in $D^{\text{tr}}$ and $D^{\text{val}}$ respectively. The names derive from the fact that, given a fixed $\theta$, we use the train set to produce an estimator using the chosen algorithm, so that the estimator is $\hat{f} = A(\theta, D^{\text{tr}})$. However, there is no universal rule for how to choose $\theta$, but one popular way is to use *cross-validation* by trying to minimize the validation error

$$\hat{\mathcal{R}}(A_\lambda(\theta, D^{\text{tr}}), D^{\text{val}}) = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \ell(A_\lambda(\theta, D^{\text{tr}})(x_i), y_i) \tag{2.3}$$

where $\lambda$ is the regularization strength which we will define below.

One very successful way to formulate algorithms is the regularized empirical risk minimization framework of [Vap91; Vap00]. The RERM framework recasts the output of the algorithm $A_\lambda(\theta, \cdot)$ as one of finding the minimizer in the hypothesis space of the regularized empirical risk,

$$A_\lambda(\theta, D) = \operatorname*{argmin}_{h \in \mathcal{H}} \hat{\mathcal{R}}(h, D) + \lambda \Omega(h), \tag{2.4}$$

where $\Omega : \mathcal{H} \to \mathbb{R}_+$ is a regularization function with regularization strength $\lambda \in \mathbb{R}_+$ which may be fixed or learned, $\mathbb{R}_+$ is the positive real line, and $\mathcal{H}$ is the hypothesis space. To ease notation we drop $\lambda$ and only write $A(\theta, D)$. If $h$ is parameterized by a vector $w$ then $\Omega$ often takes the form of L2 regularization $\Omega(h_w) = \|w\|^2$. The way that $\theta$ changes the inner problem is algorithm-dependent, but a common choice is to let $\Omega(h)$ be parameterized by $\theta$, for example $\Omega(h_w; \theta) = \|\theta - w\|^2$, while another is to parameterize a feature map $\phi_\theta : \mathcal{X} \to \mathcal{H}_\phi$ which maps instances to some feature space $\mathcal{H}_\phi$ and replace all instances $x$ by $\phi_\theta(x)$. From here on we assume that all meta-algorithms $A(\theta, D)$ take the form of (2.4), i.e. as solution to an optimization problem in the form of RERM of the dataset $D$ where the objective is parameterized by $\theta$ in some way. This formulation will play a mayor part in how we formulate meta-learning strategies.

### 2.2.1 Kernel Methods and Feature Learning

Linear models use functions linear in $x$, $h_w(x) = \langle w, x \rangle$, as the space of hypotheses. They are widely used as they are interpretable and for ridge regression [HK70] (when the RERM problem (2.4) is a least squares problem, meaning that $\mathcal{Y} \subseteq \mathbb{R}, \mathcal{X} \subseteq \mathbb{R}^d$, $\ell(y, y') = (y - y')^2$ and $\Omega(h_w) = \|w\|^2$), also known as Tikhonov regularization, the solution $h_{w_{\mathrm{RR}}}$ has a closed form. Expressing a dataset $D = (x_i, y_i)_{i=1}^n$ as the design matrix $X = (x_1, \ldots, x_n)^\top$ and output vector, which we will with abuse of notation call $y$, the solution

$$w_{\mathrm{RR}} = \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (\langle w, x_i \rangle - y_i)^2 + \lambda \|w\|^2 \tag{2.5}$$

can be shown to be $w_{\mathrm{RR}} = (X^\top X + \lambda I_d)^{-1} X^\top y$ where $I_d$ is the identity matrix of size $d$. While the linearity may seem restrictive at first, the important part is that $h_w(x)$ is linear with respect to $w$ and reproducing kernels may extend the hypothesis space to functions which are highly non-linear in $x$ through a feature map $\phi$ so that $h_w(x) = \langle w, \phi(x) \rangle$ which we elaborate on below.

Reproducing kernels are a well-established tool in machine learning, at the root of most non-parametric algorithms [SS01]. They consist of positive definite functions $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that may be interpreted as a similarity between data points. A fundamental result dating back to Moore and Aronszajn [see e.g. Aro50; CS02; SS01, and references therein] establishes that a kernel is in one-to-one correspondence with a (possibly infinite dimensional) Hilbert space $\mathcal{H}_K$ of real-valued functions on $\mathcal{X}$, such that for every $x \in \mathcal{X}$ and $f \in \mathcal{H}_K$, the function $K(x, \cdot) \in \mathcal{H}_K$ and $\langle f, K(x, \cdot) \rangle_K = f(x)$, where $\langle \cdot, \cdot \rangle_K$ denotes the inner product in $\mathcal{H}_K$. A kernel is in duality with the notion of feature map: given a mapping $\phi : \mathcal{X} \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$, the inner product $K_\phi(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ is a reproducing kernel. The converse is also true, namely for any kernel $K$ there exists a feature map $\phi_K : \mathcal{X} \to \mathcal{H}_K$ such that $K(x, x') = \langle \phi_K(x), \phi_K(x') \rangle_{\mathcal{H}_K}$ [Aro50]. A key practical advantage of kernels is that they allow to learn functions parametrized as $f(x) = \langle f, \phi_K(x)\rangle$ even when $\mathcal{H}_K$ is infinite dimensional (namely the feature vector $\phi_K(x)$ has infinitely many entries). Kernels are appealing since they allow us to use linear models in some space different to the original input space $\mathcal{X}$ which also means that we can learn models which are non-linear when viewed as a function from $\mathcal{X}$ to $\mathbb{R}$.

Kernel ridge regression performs Tikhonov regularization using the least-square loss function over the space of hypotheses associated to a reproducing kernel. More precisely, assume $\mathcal{Y} \subseteq \mathbb{R}$ and $\ell$ is the L2 loss. Given a dataset $D = (x_i, y_i)_{i=1}^n$, and a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, KRR is the algorithm

$$A_{\mathrm{KRR}}(K, D) = \operatorname*{argmin}_{h \in \mathcal{H}_K} \hat{\mathcal{R}}(h, D) + \lambda \|h\|^2_{\mathcal{H}_K}, \tag{2.6}$$

with $\lambda \in \mathbb{R}_+$ a regularization parameter. Thanks to the reproducing property of the kernel, (2.6) can be solved in closed form. We have for any $x \in \mathcal{X}$, that

$$\begin{aligned} A_{\mathrm{KRR}}(K, D)(x) &= \sum_{i=1}^n \alpha_i K(x_i, x), \\ \text{with} \quad \alpha &= (G + \lambda n I_n)^{-1} y \in \mathbb{R}^{n_{\mathrm{tr}}}, \end{aligned} \tag{2.7}$$

where $G = (K(x_i, x_j))_{i,j=1}^n \in \mathbb{R}^{n \times n}$ is the kernel (Gram) matrix.

Since kernels implicitly define a feature map, they form a framework for how to learn features and thus feature learning. One particular instance of this is the multiple kernel learning framework, see [GA11], which tries to learn a combination of pre-specified kernels by optimizing some objective function. This line of work directly leads to learning a *distribution* over bivariate functions such that the expectation of this function yields a kernel [Li+19]. Feature learning of this kind has been pursued before in the setting of supervised learning [SD16] and for few-shot learning [Zha+19]. Other similar directions include for example Kernel VAE [LCd19]. Feature learning has a long history of application in supervised and multi-task learning [Car97; ZY21], for example, [AEP08; AEP07] showed that it is possible to learn sparse features in a multi-task setting by alternatingly minimizing a non-differentiable convex objective with respect to the task weights and the features map.

## 2.3 Meta-Learning

We are now ready to introduce meta-learning, building on the notation and terminology of supervised learning introduced in Sec. 2.2. While meta-learning can be defined for other settings, such as reinforcement learning [FAL17], unsupervised learning [HLF19] and generative modelling [Hew+18], in this thesis we will deliberately only focus on the supervised learning case. In the context of meta-learning, we call the supervised learning algorithm $A$ the *inner algorithm* and $\theta \in \Theta$ parameterizing $A$ the *meta-parameter*.

The meta-learning paradigm lifts the notion of cross-validation to the level of multiple tasks; assuming that we have access to many supervised learning problems (or tasks as we call them) sharing some form of similarity, meta-learning aims to find a single meta-parameter vector $\theta$ that works well across all tasks. More formally, we assume that the tasks are sampled from a *meta-distribution* $\mu$, and we assume that we use the same loss $\ell$ throughout the tasks. From each $\rho \in \mathcal{P}(\mathcal{Z})$ sampled from $\mu$, we then sample a pair of datasets $D = (D^{\mathrm{tr}}, D^{\mathrm{val}}) \sim \rho^n$ with $n = n_{\mathrm{tr}} + n_{\mathrm{val}}$ (even though in our case we assume $n_{\mathrm{tr}}$ and $n_{\mathrm{val}}$ to be fixed for simplicity, the discussion can be extended to more general settings). To work on this higher level in accordance with how we formulated supervised learning, we let the *meta-loss* of a train and validation set $D = (D^{\mathrm{tr}}, D^{\mathrm{val}})$ be the cross-validation error

$$L(\theta, D) = \hat{\mathcal{R}}(A(\theta, D^{\mathrm{tr}}), D^{\mathrm{val}}). \tag{2.8}$$

Then, meta-learning is formulated as the problem of finding the meta-parameters $\theta \in \Theta$ for a learning algorithm $A(\theta, \cdot)$ minimizing the *transfer risk* or *meta-risk* [Den+18]

$$\mathcal{E}(\theta) = \mathbb{E}_{\rho \sim \mu} \mathbb{E}_{D \sim \rho^n} \; L(\theta, D). \tag{2.9}$$

What sets meta-learning apart from supervised learning is that the data generating process is assumed to take on a hierarchical nature where there is a meta-distribution on distributions, and each distribution defines a supervised learning problem. Each distribution, or task, sampled from the meta-distribution gives rise to a dataset.

To make this concrete we will consider the meta-learning problem where each task is a linear regression problem such that the marginal distribution is $\rho_{\mathcal{X}} = \mathcal{N}(0, 1)$ and the conditional distribution of the output $Y$ given some input $x$ is defined by a coefficient vector $\beta$ such that $Y|x = \beta^\top x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$. The choice of $\mathcal{N}(0, 1)$ for the marginal distribution $\rho_{\mathcal{X}}$ and the noise distribution is chosen for simplicity, the example works by replacing these by more general distributions. In this case the meta-distribution $\mu$ defines a distribution on $\beta$. Assuming that we know the functional form $f(x) = w^\top x$, we can use ridge regression as our algorithm to learn a $w$ from a dataset $D$. Now, the benefit of using meta-learning depends on the form of $\mu$. Assume that we have $T$ tasks with $n$ datapoints in each task dataset where we have pooled the task train and validation set into one big dataset for each task. One very simple meta-learning algorithm is to simply aggregate all of the available data into one dataset and use this together with a supervised learning algorithm to learn a global model which we use on each task we want to predict on. We use this meta-learning strategy below to showcase when meta-learning may be beneficial and when it may be worse than using ITL (so called negative transfer).

In the degenerate case that $\mu = \delta_{\beta^*}$, we could pool all tasks and increase the size of each dataset of each task from $n$ to $Tn$ which in general will lead to a great improvement over ITL. However, if the meta-distribution is very flat, for example the uniform distribution over the sphere, meta-learning should hardly improve over ITL. The distribution $\mu$ dictates if meta-learning is beneficial in this case. If we do not know the functional form, so that we use a form of $f(x)$ which is not linear, then we may even have negative transfer where ITL is strictly better than meta-learning. The takeaway of this example is that the benefit and dangers of meta-learning depends on the distribution over tasks in addition to the misspecification of the model and the algorithm we use to learn on each task.

For each dataset we want to learn a rule that generalizes to unseen data from the same task. As shown in the example above, one way to do this is to pool all of the available task datasets and learn a global model, reducing it to a supervised learning problem. This is unsatisfactory due to several reasons 1) a global model will focus on the majority of the data and may not perform well on tasks which are uncommon 2) having access to just a few data points from a task may tell us a lot about how this task differs from other tasks and enable much better performance when fine-tuning than using a global model 3) the above procedure would not work when tasks become available sequentially (such as in online learning).

In practice the meta-risk $\mathcal{E}$ is replaced by the empirical meta-risk. For an iid sample of tasks $(\rho_t)_{t=1}^T \sim \mu^T$ with corresponding datasets $(D_t)_{t=1}^T$, such that $D_t = (D_t^{\text{tr}}, D_t^{\text{val}})$, and a choice of supervised learning algorithm $A$, the bi-level formulation of meta-learning is the optimization problem [Fra21, Sec. 5.2.2]

$$\min_{\theta \in \Theta} \frac{1}{T} \sum_{t=1}^{T} \hat{\mathcal{R}}(h_t(\theta), D_t^{\text{val}}) \tag{2.10}$$

$$\text{subject to } \; h_t(\theta) = A(\theta, D_t^{\text{tr}}) \; \text{ for all } \; t = 1, \dots, T. \tag{2.11}$$

As most algorithms $A(\theta, \cdot)$ performs RERM, the constraint in (2.11) requires us to solve an inner optimization problem if we want to optimize the objective with respect to $\theta$.

In specific cases such as (kernel) ridge regression, $A(\theta, D_t^{\text{tr}})$ solves (2.4) in closed form, but in general it can only be solved approximately using optimization techniques. Sometimes (2.4) can be solved efficiently up to arbitrary precision, notably for linear models with convex loss and regularizer. However, when using neural networks, we can only hope to find a local minimium.

**Algorithm 1** Meta Learning (SGD)

```
**Input:**
    meta-dataset $(D_t)_{t=1}^T$
    total number of iterations $N$.
    initial meta-parameters $\theta_0$,
    step-sizes $(\gamma(n))_{n=1}^N$,
**For** $n = 1, \dots, N$
    Sample a task $D_n = (D_n^{\text{tr}}, D_n^{\text{val}})$ from $(D_t)_{t=1}^T$
    Form approximate meta-loss $\hat{L}(\theta_n, D_n)$
    Get $\nabla_\theta \hat{L}(\theta_n, D_n)$ using for example autodiff [Bay+18]
    Update $\theta_{n+1} \leftarrow \theta_n - \gamma(n) \nabla L(\theta_n, D_n)$
**Return** $\theta_N$
```

The commonly used approach for optimizing an objective when using neural networks is to perform stochastic gradient descent (SGD), or other commonly used iterative gradient based stochastic optimization functions such as ADAM [KB15]. For simplicity we show how stochastic gradient descent works on the meta-objective (2.10) in Algorithm 1.

For an initial point $\theta_0$ and a learning rate schedule $\gamma : \mathbb{N} \to \mathbb{R}_+$, where $\mathbb{N}$ is the set of natural numbers, for each $n \in \mathbb{N}$ we perform the one-step stochastic gradient descent update by first sampling a task $D_n$ uniformly at random from $(D_t)_{t=1}^T$ and then perform the update step $\theta_{n+1} = \theta_n - \gamma(n) \nabla_\theta L(\theta_n, D_n)$ [BCN18]. Note, however, in the general case we can only solve the inner problem (2.4) approximately meaning that we only have access to an approximate minimizer. In this case we get an approximation of the meta-loss $L(\theta_n, D_n)$ which we denote $\hat{L}(\theta_n, D_n)$ and we use this as a proxy for the true meta-loss. In practice, this means that we perform the update steps $\theta_{n+1} = \theta_n - \gamma(n) \nabla_n \hat{L}(\theta_n, D_n)$. For example, in regression settings, a common choice of inner algorithm is ridge regression [HK70] and the meta-parameter parameterize a representation or embedding shared across the tasks that we wish to meta-learn [Ber+19] while another famous meta-learning algorithm is MAML [FAL17] which uses a hypothesis space of neural networks parameterized by $\theta$. For MAML, the meta-algorithm $A(\theta, D^{\text{tr}})$ performs a few steps of gradient descent on $\hat{\mathcal{R}}(h, D^{\text{tr}})$ with respect to $h$ starting from $\theta$. The initial point $\theta$ is optimized in the outer level with respect to the meta-loss.

## 2.4 Few-Shot Learning using Meta-Learning

Few-shot learning (FSL) [FFP06] considers a meta-training set of tasks $\mathcal{T} = \{(D_t^{\text{tr}}, D_t^{\text{val}})\}_{t=1}^T$, with *support set* $D_t^{\text{tr}} = \{(x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$ and *query set* $D_t^{\text{val}} = \{(x_i, y_i)\}_{i=1}^{n_{\text{val}}}$ sampled from the same distribution. This corresponds to the standard meta-learning setup with slightly different notation and terminology, where the support and query set corresponds to the train and validation set respectively. As indicated by the name, few-shot learning focuses on the case when $D_t^{\text{tr}}$ and $D_t^{\text{val}}$ contain a small number of samples $n_{\text{tr}}$ and $n_{\text{val}}$ respectively. Similarly to the meta-learning setting, we denote by $\mathcal{D}$ the space of datasets. Typically few-shot learning considers classification which we will also do from here on in this section.

A peculiarity of this setting is that often it is assumed that $D^{\text{tr}} \cap D^{\text{val}} = \emptyset$ to make sure that the validation set of the task does not contain any of the instances of the train set of the task. In practice, this is extremely rarely violated since each task is created by sampling from a base dataset which has a large number of instances per class, and if we work with a distribution with a density this will not happen almost surely. Since we require the iid sampling of datasets of each task for the concentration inequalities in this thesis to work and since the constraint is almost always satisfied as is, we will not assume this intersection property unless noted otherwise.

The meta-learning formulation for FSL is the same as that outlined in Sec. 2.3, which we restate here with the terminology of few-shot learning. We want to find the best *base learner* $A(\theta, \cdot) : \mathcal{D} \to \mathcal{H}$ that takes as input support sets $D^{\text{tr}}$, and outputs predictors $h = A(\theta, D^{\text{tr}})$, such that predictions $y = h(x)$ generalize well on the corresponding query sets $D^{\text{val}}$. The base learner is meta-parametrized by $\theta \in \Theta$. Formally, the meta-learning objective for FSL is

$$\mathbb{E}_{(D^{\text{tr}}, D^{\text{val}}) \in \mathcal{T}} \hat{\mathcal{R}}\big(A(\theta, D^{\text{tr}}), D^{\text{val}}\big), \tag{2.12}$$

where $\mathbb{E}_{(D^{\text{tr}},D^{\text{val}})\in\mathcal{T}} := \frac{1}{|\mathcal{T}|}\sum_{(S,Q)\in\mathcal{T}}$ is the empirical risk over the meta-training set $\mathcal{T}$. The *task loss* $\hat{\mathcal{R}} : \mathcal{H} \times \mathcal{D} \to \mathbb{R}$ is the empirical risk of the learner $h$ over query sets, based on some *inner loss* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{Y}$ is the space of labels,

$$\hat{\mathcal{R}}(f, D) = \mathbb{E}_{(x,y)\in D}\ell(h(x), y). \tag{2.13}$$

The equation (2.12) is sufficiently general to describe most existing methods. For instance, model-agnostic meta-learning (MAML) [FAL17] parameterizes a model $h_\theta : \mathcal{X} \to \mathcal{Y}$ as a neural network, and $A(\theta, D)$ performs one (or more) steps of gradient descent minimizing the empirical risk of $h_\theta$ on $D$. Formally, given a step-size $\eta > 0$,

$$h_{\theta'} = A(\theta, D) \quad \text{with} \quad \theta' = \theta - \eta\nabla_\theta\hat{\mathcal{R}}(h_\theta, D). \tag{2.14}$$

Clearly, base learners $A(\theta, \cdot)$ are key to model performance and various strategies have been explored. A successful paradigm for few-shot learning using meta-learning is that of meta-representation learning [Rag+20; Lee+19; Ber+19; Fra+18]. The meta-representation learning paradigm defines the meta-learning algorithm as being composed to two parts, a feature map and a supervised learning estimator working on the inputs mapped through this feature map. The feature map is typically a neural network suited for the data modality (such as CNNs or ResNets for image data) while the supervised learning estimator tends to be closed form or the solution to a convex problem (such as KRR or logistic regression) which removes the inner optimization error due to having to satisfy the constraints of the bi-level formulation of meta-learning (2.10). By mapping all inputs of a task through the feature map and solving the inner problem using the estimator on the train set of the task we get a predictor. This predictor is then applied to the validation set of the task which allows us to get the cross-validation error. The feature-map is updated using the gradients of the validation error with respect to the parameters of the feature map.

This can be formalized by parameterizing the base learner as

$$A(\theta, D) = w(\phi_\theta(D))\phi_\theta(\cdot), \tag{2.15}$$

where $\phi_\theta(D) = (\phi_\theta(x_i), y_i)_{i=1}^n$, separating it into parts of a global feature extractor $\phi_\theta : \mathcal{X} \to \mathbb{R}^m$ and a task-adaptive classifier $w : \mathcal{D} \to \{f : \mathbb{R}^m \to \mathcal{Y}\}$, where here $\mathcal{D}$ is the set of all datasets with inputs living in the output space of $\phi_\theta$. This specializes (2.12) by learning a feature extractor $\phi_\theta$ shared (and fixed) among tasks. Only the classifier returned by $w(\cdot)$ adapts to the current task, in contrast to having the entire model $h_\theta : \mathcal{X} \to \mathcal{Y}$ adapted (e.g. (2.14) for MAML). While this may appear to restrict model adaptability, [Rag+20] has demonstrated that meta-representation learning matches MAML's performance. Moreover, they showed that feature reuse is the dominant contributor to the generalization performance rather than adapting the representation to the task at hand.

The task-adaptive classifier $w(\cdot)$ may take various forms, including nearest neighbor [SSZ17], ridge regression classifier [Ber+19], embedding adaptation with transformer models [Ye+20], and Wasserstein distance metric [Zha+20]. In particular, the ridge regression estimator

$$w_{\mathrm{RR}}(D) = \operatorname*{argmin}_{W \in \mathbb{R}^{l \times d}} \mathbb{E}_{(x,y)\in D} \|Wx - y\|^2 + \lambda_1 \|W\|_F^2, \tag{2.16}$$

where $\|\cdot\|_F$ is the Frobenius norm, $l$ is the number of output dimensions and $x \in \mathbb{R}^d$, admits a differentiable closed-form solution and is computationally efficient for optimizing (2.12) when using (2.15) as the category of meta-learning algorithms.

## 2.4.1 Conditional Meta-Learning

Conditional formulations of meta-learning [DPC20; WDC20] extends (2.12) by considering base learners of the form $A(\tau(Z), D^{\mathrm{tr}})$, where the meta-parameters $\theta = \tau(Z)$ is conditioned on some "contextual" information $Z \in \mathcal{Z}$ about the task. Note that $\mathcal{Z}$ here is an arbitrary set, although we will see that it will most often coincide with the data space defined previously. Assuming each task in the meta-training set $\mathcal{T}$ to be equipped with such contextual information, (2.12) can be re-expressed as

$$\min_{\tau:\mathcal{Z}\to\Theta} \mathbb{E}_{(D^{\text{tr}},D^{\text{val}},Z)\in\mathcal{T}} \hat{\mathcal{R}}\big(A(\tau(Z),D^{\text{tr}}),D^{\text{val}}\big), \tag{2.17}$$

namely the problem of learning a function $\tau : \mathcal{Z} \to \Theta$ which, given some contextual information $Z$ in a suitable space $\mathcal{Z}$, returns a good task-specific meta-parameter $\theta = \tau(Z)$. While the contextual information could encode virtually any information available on individual tasks (e.g. a textual meta-description of the task/dataset), most recent work on the topic focus on the case where it corresponds to the task's support set itself, namely $Z = D^{\text{tr}}$, since this is always available by construction.

The conditional formulation seeks to capture complex (e.g. multi-modal) distributions of meta-training tasks, and uses a unique base learner tailored to each one. In particular, [Vuo+19; Yao+19; Rus+18] directly learn data-driven mappings from target tasks to meta-parameters, and [Jia+18] conditionally transforms feature representations based on a metric space trained to capture inter-class dependencies. Alternatively, [Jer+19] considers a mixture of hierarchical Bayesian models over the parameters of meta-learning models in order to condition on target tasks. In [WDC20], Wang *et al.* showed that conditional meta-learning can be interpreted as a structured prediction problem and proposed a method leveraging recent advances in the latter field. From a more theoretical perspective, [DPC20] proved that conditional meta-learning is theoretically advantageous compared to unconditional approaches by incurring smaller excess risk and being less prone to negative transfer. As we will discuss in Sec. 3.3, conditional meta-learning is closely related to our theoretical analysis on feature pre-training.

### 2.4.2 Feature Pre-Training

Feature pre-training has been widely adopted in the recent meta and few-shot learning literature [e.g. Man+20; OLL18; Che+18; WDC20; WTH21; YLX21; Ye+20; Bat+20; Req+19; Zha+20], and is arguably one of the key contributors to performance of state-of-the-art models. Instead of directly learning the feature extractor $\phi_\theta$ by optimizing the empirical meta-risk, pre-training first learns a feature extractor via standard supervised learning.

Formally, the meta-training set $\mathcal{T}$ is "flattened" into $D_{\text{global}}$ by merging all tasks:

$$D_{\text{global}} = D(\mathcal{T}) = \{(x_i, y_i)\}_{i=1}^{N} = \bigcup_{(D^{\text{tr}}, D^{\text{val}}) \in \mathcal{T}} (D^{\text{tr}} \cup D^{\text{val}}). \tag{2.18}$$

Pre-training then learns the embedding function $\phi_\theta$ on $D_{\text{global}}$ using the standard cross-entropy loss $\ell_{\text{ce}}$ for multi-class classification:

$$(W_N^{\text{pre}}, \theta_N^{\text{pre}}) = \underset{\theta, W \in \mathbb{R}^{C \times m}}{\operatorname{argmin}} \hat{\mathcal{R}}(W\phi_\theta, D_{\text{global}}), \tag{2.19}$$

where $W$ is the linear classifier over all classes with cardinality $C$. After pre-training, the feature extractor is either fixed [e.g. Rus+18; Ye+20; WDC20; Zha+20; Tia+20] or further adapted [e.g Bat+20; Bat+22; Gol+20; Req+19] via meta-learning.

There is limited theoretical understanding and consensus on the effect of pre-training in FSL. In [Rus+18; Bat+20; YLX21], the pre-training is only considered a standard pre-processing step for encoding the raw input and model performance is predominantly attributed to the proposed meta-learning algorithm. In [Gol+20] the authors similarly argued that meta-trained features are fundamentally better than pre-trained ones, observing that adapting the pre-trained features with several base learners resulted in worse performance compared to the meta-learned features. In contrast, however, several works also empirically demonstrated that pre-training contributes significantly towards performance. [Tia+20] first showed that combining the pre-trained features with suitable base learners already outperforms various meta-learning methods, while [El +22] observed that pre-training dominates top entries for the 2021 Meta-learning Challenge.

## 2.5 Literature Review of Meta-Learning and Few-Shot Learning

The field of meta-learning or learning-to-learn [Bax00; TP98] can trace its origins to works such as [Bax00; Sch87] and the field of multitask learning [Car97]. Well-developed theory exists in the batch case [MJ05; MPR16] and lately similar results have been developed in the online setting [BKT19; Den+19].

Meta-learning operates on top of an inner algorithm, tuning it to perform better on new tasks. The meta-algorithm acts at an outer level, relying on the inner algorithm to compute a meta-loss and corresponding meta-gradient, based on which a meta-parameter associated to the inner algorithm is updated [see e.g. Fra+18]. For example, in regression settings, a common choice of inner algorithm is ridge regression and the meta-parameter is a representation or embedding shared across the tasks that we wish to meta-learn [Ber+19].

In the past few years increasing interest has lead to many new challenging benchmark datasets such as Omniglot [LST15; LST19] and derivations of ImageNet [Den+09] in particular *mini* [Vin+16] and *tiered*-ImageNet [Ren+18], together with competitive meta-learning algorithms [FAL17]. This was made possible by formulating the meta-learning problem as that of trying to solve a specific bi-level optimization problem [RL17; Fra+18] which reduces to trying to find hyperparameters $\theta$ with low meta-risk (2.9). Ideally, for a pre-specified algorithm $A$ and a meta-dataset $(D_t)_{t=1}^T$, we want the meta-algorithm to output $\hat{\theta}((D_t)_{t=1}^T) \approx \theta^* \in \operatorname{argmin}_{\theta \in \Theta} \mathcal{E}(\theta)$.

This view has lead to a multitude of algorithms [FAL17; KZS15; Li+17; Ren+18; Rus+18; SSZ17; Vin+16], heuristically grouped into the categories of metric-based [SSZ17], optimization-based [FAL17] and black-box-based [Vin+16] meta-learning. In practice most deep meta-learning algorithms use aspects from all three categories by considering the feature map as part of the hyperparameters and optimizing this jointly with the rest of the hyperparameters with respect to the meta-loss [FAL17]. In this case learning the hyperparameters can be seen as learning a good inductive bias for tasks we are likely to see. A common theme is to learn a shared representation which lead to faster adaptation of a base learning algorithm to new tasks. Often the representation is modeled by a neural network. Recently [Tia+20; Rag+20] observed that the representation is the most important part of meta-learning algorithms. A common trend in few-shot and meta-learning is the view that few-shot learning works primarily by learning an algorithm which can adapt fast, many works have started to question this [Zin+19; FAL17], noting that this is not the full story and the feature map is very important for getting good performance. See also [Col+22] for a recent theoretical investigation for this in the context of gradient based meta-learning methods.

Chapter 3

# MeLa

## 3.1 Summary

In this chapter, based on the pre-print (under review) [Wan+22] which is an extended version of the paper [WPC21], we answer an important question in few-shot learning; why is pre-training so effective as a strategy for few-shot learning (FSL) for image classification? Few-shot learning is a central problem in meta-learning, where learners must efficiently learn from few labeled examples. Within FSL, pre-training has been shown to be integral to reaching good performance but the way that this interacts with FSL and bolsters performance is poorly understood, and requires assuming that global labels are available which may not be true in practice.

We address the above issues by first showing the connection between pre-training and meta-learning. We discuss why pre-training yields more robust meta-representation and connect the theoretical analysis to existing works and empirical results. Secondly, we introduce Meta Label Learning (MeLa), a novel meta-learning algorithm that learns task relations by inferring global labels across tasks. This allows us to exploit pre-training for FSL even when global labels are unavailable or ill-defined. Lastly, we introduce an augmented pre-training procedure that further improves the learned meta-representation. Empirically, MeLa outperforms existing methods across a diverse range of benchmarks, in particular under a more challenging setting where the number of training tasks is limited and labels are task-specific. We also provide an extensive ablation study to highlight its key properties.

### 3.1.1 Contribution

The contributions of this chapter are several. Firstly, we show that pre-training directly relates to meta-learning by minimizing an upper bound on the meta-learning loss. In particular, we show that pre-training achieves a smaller expected error and enjoys a better convergence rate compared to its meta-learning counterpart. More broadly, we connect pre-training to conditional meta-learning [DPC20; WDC20], which has favorable theoretical properties including tighter bounds. Our result provides a principled justification of why pre-training yields a robust meta-representation for FSL, and the associated performance improvement.

Secondly, motivated by the above result, we propose an augmentation procedure for pre-training in order to improve representation learning for FSL. The augmentation procedure quadruples the number of training classes by considering rotations as novel classes and classifying them jointly. This significantly increases the size of training data and leads to robust representations. We empirically demonstrate that the augmentation procedure consistently performs better across different benchmarks.

Thirdly, we introduce a way to tackle independent task learning, where for each task we only have access to local labels. We call the resulting clustering algorithm **Me**ta **La**bel Learning (MeLa), a novel algorithm that automatically infers a notion of *latent* global labels consistent with local task constraints. Inferring labels allows to perform pre-training even when the global labels are not available or not even defined, leading to improved performance. Empirically, we demonstrate that MeLa is competitive with training on oracle labels.

Fourthly, for experiments, we introduce a new generalized FSL (GFSL) setting. In addition to independent task annotation, we also adopt a fixed-size meta-training set and enforce no repetition of samples across tasks. This challenging setting evaluates how effectively meta-learning algorithms generalize from limited number of tasks, and prevents the algorithms from trivially uncover task relations by implicitly matching identical samples across tasks. We empirically show that MeLa performs robustly in both standard and GFSL settings, and clearly outperforms state-of-the-art models in the latter.

A code repository for creating / downloading the datasets and run the pipeline of MeLa on the different benchmarks has been made available at `https://github.com/IsakFalk/mela`.

### 3.1.2 Organization

In Sec. 3.2 we introduce the setting of the work and put it in context with prior work. In Sec. 3.3 we show theoretically the relationship between pre-training and meta-learning through an upper bound and based on this, in Sec. 3.4 we introduce three different algorithms (augmented pre-training, MeLa, meta fine-tuning) for use in the few-shot learning pipeline which improves performance and allows for pre-training when global labels are missing. In Sec. 3.5 we benchmark MeLa against other competitive few-shot learning algorithms and perform ablation studies.

## 3.2 Introduction

Deep neural networks have facilitated transformative advances in machine learning in various areas [e.g. Sil+17; Goo+14; He+16; Bro+20; KSH12; Mni+15]. However, state-of-the-art models typically require labeled datasets of extremely large scale, which are prohibitively expensive to curate. When training data is scarce, neural networks often overfits which degrades performance significantly. Few-shot learning aims to address this loss in performance by developing algorithms and architectures capable of learning from few labeled samples.

Meta-learning [Hos+22; Van19] is a popular class of algorithms created for tackling FSL. Broadly, meta-learning seeks to learn transferable knowledge over many FSL tasks, and to apply such knowledge to novel ones. Existing meta-learning methods for tackling FSL may be loosely classified into three categories; optimization [e.g. FAL17; Ber+19; WTH21], metric learning [e.g. Vin+16; SSZ17; Sun+18], and model-based methods [e.g. HDL17; QBL18]. The diversity of existing strategies poses a natural question: can we derive any "meta-insights" from them to facilitate the design of future methods?

Among the existing methods, several trends have emerged for designing robust few-shot meta-learners. Chen *et al.* observed that data augmentation and deeper networks significantly improves generalization performance [Che+18]. The observations have since been widely adopted [e.g. Tia+20; Bat+20; Lee+19]. On the other hand, *network pre-training* has also become ubiquitous [e.g. El +22; WDC20; Rod+20; WTH21], and dominates state-of-the-art models. Sidestepping the task structure and episodic training of meta-learning, pre-training learns (initial) model parameters by merging all FSL tasks into one "flat" dataset of labeled samples followed by standard multi-class classification. The model parameters may be further fine-tuned to improve performance.

Despite its popularity, the limited theoretical understanding of pre-training leads to diverging interpretations of existing methods. Most works consider pre-training as nothing but a standard pre-processing step, and attribute the observed performance almost exclusively to their respective algorithmic and network design choices [e.g. Ye+20; Rus+18; YLX21]. However, extensive empirical evidence suggests that pre-training is crucial for model performance [WTH21; WPC21]. Tian *et al.* demonstrated that simply learning task-specific linear classifiers over the pre-trained representation outperforms a number of various meta-learning strategies [Tia+20]. Wertheimer *et al.* further showed that earlier FSL methods may also benefit from pre-training, resulting in improved performance [WTH21].

### 3.2.1 Related Work

This chapter is based on a the pre-print [Wan+22] which is an extended version of [WPC21] with the following contributions in addition to those of the original work: 1) a deeper theoretical insight into the role of pre-training from the perspective of the risk (rather than the empirical risk as in [WPC21]), and quantifying its benefit in terms of sample complexity, 2) an augmented training procedure for FSL, 3) the Generalized Few-Shot Learning (GFSL) experimental setting, 4) significantly more empirical evidence to support the proposed algorithm.

Our proposed method is most closely related to meta-representation learning [Rag+20; Lee+19; Ber+19; Fra+18], see Sec. 2.4 and 2.4.2, which parametrizes the base learner as $A(\theta, D) = w(\phi_\theta(D))\phi_\theta(\cdot)$, separating it into parts of a global feature extractor $\phi_\theta : \mathcal{X} \to \mathbb{R}^m$ and a task-adaptive classifier $w : \mathcal{D} \to \{f : \mathbb{R}^m \to \mathcal{Y}\}$, so that for a dataset $D \in \mathcal{D}$, where we recollect that $\mathcal{D}$ is the set of all possible datasets, $w(D) = f$ is a function mapping from inputs $x \in \mathbb{R}^m$ to the output space $\mathcal{Y}$, and where finally $\phi_\theta(D) = (\phi_\theta(x), y)_{(x,y)\in D}$.

## 3.3 Pre-Training as Meta-Learning

In this section, we characterize how feature pre-training relates to meta-learning as a loss upper bound. More precisely, we show that pre-training induces a special base learner with its corresponding meta-learning loss upper bounded by the cross-entropy loss $\ell_{\text{ce}}$. Consequently, pre-training already produces a meta-representation suitable for FSL, matching the empirical results from [Tia+20; Ye+20]. In addition, we show that pre-training incurs a smaller risk compared to its meta-learning counterpart, and more generally induces a conditional formulation that exploits contextual information for more robust learning.

### 3.3.1 Notation

We consider a few-shot classification setting with a total of $C$ classes (global labels). Denote by $\mu$ the meta-distribution for task sampling and distribution $\rho$ for sampling support and query sets $(D^{\text{tr}}, D^{\text{val}})$ for each task. Task distributions $\rho$ are associated with $k \leq C$ class labels $y_\rho^{(1)}, \dots, y_\rho^{(k)} \in \{1, \dots, C\}$. Denote by $\rho_Y$ the subset of $\{1, \dots, C\}$ of $k$ task labels. Given a matrix $W \in \mathbb{R}^{C\times m}$ and a vector $Y \in \{1, \dots, C\}^n$ of indices, we denote by $W[Y] = W[\rho_Y] \in \mathbb{R}^{k\times m}$

the submatrix of $W$ obtained by selecting the rows corresponding to the unique indices $\rho_Y$ in $Y$. Lastly, given a dataset $D = (x_i, y_i)_{i=1}^n$, let $D_Y \in \{1, \dots, C\}^n$ denote the vector with entries corresponding to the labels $(y_i)_{i=1}^n$ and for a feature map $\phi_\theta$ and we recall the definition $\phi_\theta(D) = (\phi_\theta(x_i), y_i)_{i=1}^n$.

### 3.3.2 Meta-Learning Expected Error

The meta-risk for our specific case takes the form

$$\mathcal{E}(\theta, \mu) = \mathbb{E}_{\rho\sim\mu}\mathbb{E}_{(D^{\text{tr}},D^{\text{val}})\sim\rho}\hat{\mathcal{R}}(w(\phi_\theta(D^{\text{tr}})), \phi_\theta(D^{\text{val}})), \tag{3.1}$$

where we have made the dependency on $\mu$ explicit.

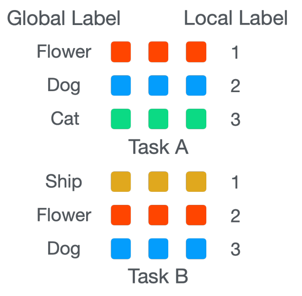
This is the meta-learning risk incurred by a meta-parameter $\theta$, namely the error incurred by training the classifier via $w(\phi_\theta(D^{\text{tr}}))$ (e.g. (2.16)) and testing it on the query set $\phi_\theta(D^{\text{val}})$, averaged over $(D^{\text{tr}}, D^{\text{val}})$ pairs sampled from tasks $\rho$, which in turn are sampled from meta-distribution $\mu$. The risk is the ideal error we wish to minimize.

### 3.3.3 Global Label Selection

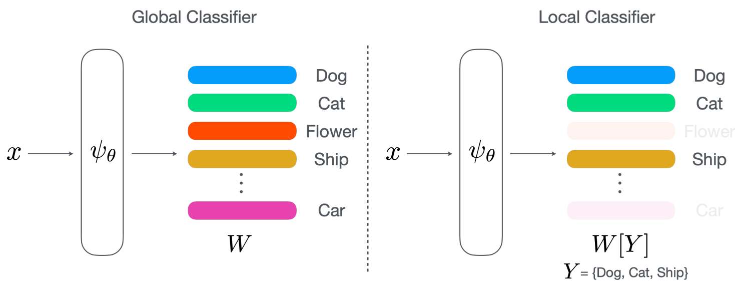
Let us consider a special FSL scenario where global labels are available to the model (in contrast to the standard setting where only local labels are available, see Sec. 2.4). Since we have access to global labels, we can design a new algorithm that learns a single global multi-class linear classifier $W$ at the meta-level (i.e. shared across all tasks), and simply select the required rows $W[D_Y^{\text{tr}}]$ when tackling a task. More formally, we can define a special base learner called *global label selector (GLS)* such that

$$A((W, \theta), D^{\text{tr}}) = \text{GLS}(W, \theta, D^{\text{tr}}) = W[D_Y^{\text{tr}}]\phi_\theta(\cdot). \tag{3.2}$$

Illustrated in Fig. 3.1b, this “algorithm” does not solve an optimization problem on the dataset $D^{\text{tr}}$, but only selects the subset of rows of $W$ corresponding to the classes present in $D^{\text{tr}}$ as the task-specific classifier. Similar to



(a) **Global vs. Local labels**



(b) **Global to Local Classifier**

Figure 3.1: **(a)** Colored squares represent samples. Tasks A and B can be "merged" meaningfully using global labels, but not local ones. **(b)** A global classifier can be used as local classifiers given the indices $Y$ of the intended classes to predict.

(3.1), we define the meta-risk for GLS as

$$\mathcal{E}_{\text{GLS}}(W,\theta,\mu) = \mathbb{E}_{\rho\sim\mu}\mathbb{E}_{(D^{\text{tr}},D^{\text{val}})\sim\rho}\hat{\mathcal{R}}(W[D_Y^{\text{tr}}]\phi_\theta(\cdot),D^{\text{val}}), \tag{3.3}$$

the error incurred by using the meta-representation $\phi_\theta$ and the global linear classifier $W$ to tackle the meta-learning problem associated with $\mu$.

Analogous to standard meta-learning (where we only learn $\theta$), since $W$ and $\theta$ are now both shared across all tasks, we may learn them jointly by solving the following minimization problem

$$\min_{W,\theta} \frac{1}{T}\sum_{t=1}^{T}\hat{\mathcal{R}}(W[D_{t,Y}^{\text{tr}}]\phi_\theta(\cdot),D_t^{\text{val}}), \tag{3.4}$$

where the sum is taken over all tasks in the meta-train set $\mathcal{T}$. This strategy, to which we refer as *meta-GLS*, learns both the representation and linear classifier at the meta-level, with the sole task-specific adaptation process being the selection of columns of $W$ using the global labels.

### 3.3.4 GLS Finds a Good Meta-Representation

As the intuition suggests, learning a global $W$ shared among multiple tasks (rather than each classifier $w(\phi_\theta(D^{\text{tr}}))$ accessing exclusively the tasks' training data), can be very advantageous for generalization. This is evident when the (global) classes are separable for a meta-representation $\phi_\theta$. In this case, due to the separability assumption, for any inner algorithm $w(\cdot)$, we have that

$$\min_W \mathcal{E}_{\mathrm{GLS}}(W, \theta, \mu) \leq \mathcal{E}(\theta, \mu), \tag{3.5}$$

namely, that given the same representation, finding a global classifier is more favorable than solving each task in isolation. This tells us that in the ideal scenario of separability, GLS provides a family of algorithms (indexed by $W$ and $\theta$) which dominates that of meta-representation learning (indexed by $w$ and $\theta$) in the sense of (3.5).

### 3.3.5 Pre-training and GLS

Existing works such as [Tia+20; El +22] demonstrates that pre-training offers a robust alternative to learn the meta-representation $\phi_\theta(\cdot)$. We will show that GLS is related to pre-training, under some mild assumptions.

**Assumption A.** *The meta-distribution $\mu$ samples tasks $\rho$. Sampling from each $\rho$ is performed as follows:*

1. *For each class $y_\rho^{(j)}$ in $\rho_Y$, with $j = 1, \ldots, k$, we sample $n$ examples $x_1^{(j)}, \ldots, x_n^{(j)}$ i.i.d. according to a conditional distribution $\pi(x|y = y_\rho^{(j)})$ shared across all tasks. All generated pairs are collected in the support set $D^{\mathrm{tr}} = (x_i^{(j)}, y_\rho^{(j)})_{i,j=1}^{n,k}$.*
2. *The query set $D^{\mathrm{val}}$ is generated by sampling $m$ points i.i.d. from $\pi_\rho(x, y)$, namely $D^{\mathrm{val}} \sim \pi_\rho^m$ with*

$$\pi_\rho(x, y) = \pi(x|y)\mathrm{Unif}_\rho(y) \tag{3.6}$$

*and $\mathrm{Unif}_\rho$ the uniform distribution over the labels in $\rho_Y$.*

In essence, the assumption characterizes the standard process of constructing meta-training tasks for FSL. In particular, let $\pi_\mu(x, y)$ be the marginal probability of observing $(x, y)$ in the meta-training tasks, i.e. firstly sampling a task

$\rho$ from $\mu$, followed by sampling a class $y$ uniformly by $\mathrm{Unif}_\rho(\cdot)$ and finally $x$ by $\pi(\cdot|y)$. It then follows that sampling a dataset $D_{\text{global}}$ from $\pi_\mu$ is equivalent to sample a meta-training set $\mathcal{T}$ from $\mu$ and flatten it into $D(\mathcal{T})$ according to the pre-training procedure described in (2.18).

We can therefore introduce the *(global) multi-class classification* risk associated to $\pi_\mu$

$$\mathcal{R}(W\phi_\theta(\cdot), \pi_\mu) = \mathbb{E}_{(x,y)\sim\pi_\mu}\ \ell_{\text{ce}}(W\phi_\theta(x), y). \tag{3.7}$$

We notice that pre-training amounts to minimizing $\hat{\mathcal{R}}(W\phi_\theta, D_{\text{global}})$ over the dataset $D_{\text{global}}$ obtained by sampling iid from $\pi_\mu$, therefore the above risk can be seen as the ideal objective of the pre-training estimator. Under the above assumptions, the following result relates the two problems to each other, the proof may be found in Appendix A.

**Theorem 1.** *Under Assumption A, let $\pi_\mu(x, y)$ be marginal distribution of observing $(x, y)$ in the meta-training set. Then, for any (global) classifier $W$,*

$$\mathcal{E}_{\text{GLS}}(W, \theta) \leq \mathcal{R}(W\phi_\theta, \pi_\mu). \tag{3.8}$$

*Moreover, if the global classes are separable,*

$$\min_{W,\theta} \mathcal{E}_{\text{GLS}}(W, \theta) = \min_{W,\theta} \mathcal{R}(W\phi_\theta, \pi_\mu). \tag{3.9}$$

The result shows that the GLS error is upper bounded by the global multi-class classification error. Hence, minimizing the global multi-class classification error also indirectly minimizes the meta-learning risk. In practice, this shows that *pre-training is explicitly linked to meta-learning by learning a meta-representation suitable for FSL through this upper bound, if we use the GLS algorithm.*

### 3.3.6 Generalization Properties

Thm. 1 shows that under the class-separability assumption, pre-training is equivalent to performing meta-GLS. We now study which of the two approaches is more sample-efficient from a generalization perspective.

Let $(W_T^{\text{GLS}}, \theta_T^{\text{GLS}})$ denote the meta-parameters learned by an algorithm minimizing (3.4) over a dataset $\mathcal{T}$ comprising of $T$ separate tasks. Applying standard results from statistical learning theory, we can obtain excess risk bounds characterizing the quality of $\theta_T^{\text{GLS}}$'s predictions in terms of the number $T$ of tasks the algorithm has observed in training. We perform the decomposition

$$\begin{aligned}\mathcal{E}_{\text{GLS}}(W_T^{\text{GLS}}, \theta_T^{\text{GLS}}, \mu) - \mathcal{E}_{\text{GLS}}(W^*, \theta^*, \mu) = {} & \mathcal{E}_{\text{GLS}}(W_T^{\text{GLS}}, \theta_T^{\text{GLS}}, \mu) \\ & \pm \mathcal{E}_{\text{GLS}}(W_T^{\text{GLS}}, \theta_T^{\text{GLS}}, D(\mathcal{T})) \\ & \pm \mathcal{E}_{\text{GLS}}(W^*, \theta^*, D(\mathcal{T})) \\ & - \mathcal{E}_{\text{GLS}}(W^*, \theta^*, \mu) \\ \leq {} & (A) + (B) + (C)\end{aligned}$$

where $(W^*, \theta^*) = \operatorname{argmin}_{W,\theta\in\Omega} \mathcal{E}_{\text{GLS}}(W, \theta, \mu)$, which is assumed to exist, and $\Omega$ is the domain on $W, \theta$ which will be explained below, and the final terms are

$$\begin{aligned}(A) &= \left|\mathcal{E}_{\text{GLS}}(W_T^{\text{GLS}}, \theta_T^{\text{GLS}}, \mu) - \mathcal{E}_{\text{GLS}}(W_T^{\text{GLS}}, \theta_T^{\text{GLS}}, D(\mathcal{T}))\right| \\ (B) &= \mathcal{E}_{\text{GLS}}(W_T^{\text{GLS}}, \theta_T^{\text{GLS}}, D(\mathcal{T})) - \mathcal{E}_{\text{GLS}}(W^*, \theta^*, D(\mathcal{T})) \\ (C) &= |\mathcal{E}_{\text{GLS}}(W^*, \theta^*, \mu) - \mathcal{E}_{\text{GLS}}(W^*, \theta^*, D(\mathcal{T}))|.\end{aligned}$$

Since $W_T^{\text{GLS}}, \theta_T^{\text{GLS}}$ minimize $\mathcal{E}_{\text{GLS}}(W, \theta, D(\mathcal{T}))$, the term $(B)$ is negative, so bounding $(A) + (C)$ is enough to upper bound the excess risk $\mathcal{E}_{\text{GLS}}(W_T^{\text{GLS}}, \theta_T^{\text{GLS}}, \mu) - \mathcal{E}_{\text{GLS}}(W^*, \theta^*, \mu)$, which may be done using Rademacher Complexities.

For instance, following [SB14, Chapter 26] we have that in expectation with respect to the sampling of $\mathcal{T}$

$$\begin{aligned}\mathbb{E}_{\mathcal{T}}\mathcal{E}_{\mathrm{GLS}}(W_T^{\mathrm{GLS}},\theta_T^{\mathrm{GLS}},\mu) &\le \min_{(W,\theta)\in\Omega}\ \mathcal{E}_{\mathrm{GLS}}(W,\theta,\mu) + 2L_{\mathrm{GLS}}\mathfrak{R}_T(\Omega)\\ &\le \min_{(W,\theta)\in\Omega}\ \mathcal{E}_{\mathrm{GLS}}(W,\theta,\mu) + \frac{2L_{\mathrm{GLS}}C_\Omega}{\sqrt{T}},\end{aligned}$$

where $L_{\mathrm{GLS}}$ denotes the Lipschitz constant of $\mathcal{E}_{\mathrm{GLS}}$ as a function of $(W,\theta)$, while $\Omega \subseteq \mathbb{R}^{m\times C}\times\Theta$ is bounded, where $\Omega$ is the space of hypotheses for the multi-class classifier $Wg_\theta(\cdot)$, the full assumptions can be found in Appendix A.1. The assumptions includes that the feature map has $\|\phi_\theta\| = 1$ and $\|W\|_{2,1}\le B_\lambda$ where $B_\lambda$ is the radius of a ball which increase as $\lambda\to 0$ and the $2,1$–norm is defined as $\|W\|_{2,1} = \sum_{z\le C}\|W_z\|_2$. Here, $\mathfrak{R}_T(\Omega)$ is the Rademacher complexity of $\Omega$ [SB14], which measures the overall potential expressivity of an estimator that can be trained over them. For neural networks, [GRS18] showed that $\mathfrak{R}_T(C_\Omega)$ may be further bounded by $\mathfrak{R}_T(\Omega)\le C_\Omega/\sqrt{T}$, where $C_\Omega$ is constant depending on the specific neural architecture, with deeper networks having a larger constant. The bound indicates that the risk incurred by GLS becomes closer to that of the ideal meta-parameters as the number of observed tasks $T$ grows.

We can apply the same Rademacher-based bounds to (3.7) and the pre-training estimator from (2.19), obtaining

$$\mathbb{E}_{\mathcal{T}}\mathcal{R}(W_N^{\mathrm{pre}},\theta_N^{\mathrm{pre}},\pi_\mu) \le \min_{(W,\theta)\in\Omega}\ \mathcal{R}(W,\theta,\pi_\mu) + \frac{2L_{\mathrm{pre}}C_\Omega}{\sqrt{N}}, \tag{3.10}$$

where $N$ is the number of samples in $D_{\mathrm{global}}$ and $L_{\mathrm{pre}}$ is the Lipschitz constant of the global multi-class classification risk. By combining the above bound with the result from Thm. 1 we conclude that

$$\mathbb{E}_{\mathcal{T}}\mathcal{E}_{\mathrm{GLS}}(W_N^{\mathrm{pre}},\theta_N^{\mathrm{pre}},\mu) \le \min_{W,\theta}\ \mathcal{E}_{\mathrm{GLS}}(W,\theta,\mu) + \frac{2L_{\mathrm{pre}}C_\Omega}{\sqrt{N}}, \tag{3.11}$$

which is an excess risk bound analogous to that obtained for meta-GLS. The key difference is that the bound above depends on the number $N$ of total samples in $D_{\text{global}}$, rather that the total number $T$ of tasks.

Comparing the rates of meta-GLS and the pre-training estimator, we observe that typically $N \gg T$ (for instance $N = nT$ when each task has the same number of $n$ samples). Additionally, since $L_{\text{pre}}$ is comparable or smaller than $L_{\text{GLS}}$ (see Appendix A.1), we conclude that *given exactly the same data ($\mathcal{T}$ for meta-GLS and $D_{\text{global}} = D(\mathcal{T})$ for pre-training), pre-training achieves a smaller error than meta-GLS and the estimator approach this error faster than the meta-GLS estimator.* For instance, in the case of a 5-way-5-shot FSL problem, pre-training improves upon the meta-GLS bound on excess risk by a factor of $\sqrt{N/T} = \sqrt{n} = \sqrt{100} = 10$. Given the relation between GLS and standard meta-learning that we highlighted in Sec. 3.3.3, our analysis provides a strong theoretical argument in favor of adopting pre-training in meta-learning settings.

Note that the above analysis is done for any meta-distribution $\mu$ which satisfy the assumptions of the results. The fact that the bound does not take into account $\mu$ except for implicitly in the meta-risk, as the Rademacher upper bound is independent of it, means that the bound fails to tell us anything about transfer (positive or negative) or the actual quality of the optimal GLS estimator $W\phi_\theta$. Thus it cannot answer questions of the kind related to when and how to perform meta-learning and how to design algorithms to avoid negative transfer. This would require a different type of analysis of the one we carried out here.

### 3.3.7 Connection to Conditional Meta-Learning

More generally, we observe that GLS is also an instance of conditional meta-learning: the global labels of the task provide additional contextual information about the task to facilitate model learning. Global labels directly reveal how tasks relate to one another and in particular if any classes to be learned are shared across tasks. GLS thus simply map global labels of tasks to task classifiers via $W[D_Y^{\text{tr}}]$. In contrast, unconditional approaches (e.g. R2D2 [Ber+19], ProtoNet [NAS18]) learn classifiers by minimizing some loss over support sets, losing out on the access to the contextual information provided by global labels. As discussed in Sec. 2.4.2, the benefits offered by the global labels has been extensively validated empirically.

In addition to our result, Denevi *et al.* [DPC20] also proved that conditional meta-learning is advantageous over the unconditional formulation by incurring a smaller excess risk, especially when the meta-distribution of tasks is organized into distant clusters. We refer readers to the original paper for a detailed discussion. In practice, global labels provide clustering of task samples for free and improve regularization by enforcing each cluster (denoted by global label $y_\rho^j$) to share classifier parameters $W[y_\rho^j]$ across all tasks. This provides further explanation to why pre-training yields a robust meta-representation with strong generalization performance.

### 3.3.8 Leveraging Pre-Training in Practice

The discussion above suggests that pre-training should be sufficient when meta-training and meta-test data are sampled from the same distribution. However, practical FSL scenarios assume that the meta-testing set shares no class labels with the meta-training set, since the goal of meta-learning is precisely to generalize to novel classes unseen during training. For these practical scenarios, extensive evidences indicate that the pre-trained representation is also robust for learning novel classes by simply replacing the GLS selector with regular classifier $w(\phi_\theta(D^{\text{tr}}))$ [TKI20; El +22]. More formally, the connection between meta-training and meta-testing classes may also be captured by the assumption that that they share a common representation, the theoretical setting analyzed in [Du+20].

Moreover, while pre-training might offer a powerful initial representation $\theta$, it may be advisable to further improve $\theta$ by directly optimizing the empirical meta-risk using the desired classifier to tackle novel classes. The general strategy of pre-training followed by what we call *meta fine-tuning* is extensively used in state-of-the-art methods [e.g. Ye+20; Rod+20; Zha+20]. Empirical results suggest that careful meta fine-tuning outperforms standalone pre-training. We investigate this aspect empirically in Sec. 3.4.3 and Sec. 3.5.

## 3.4 Methods

In this section, we propose three practical algorithms motivated by our theoretical analysis. In Sec. 3.4.1, we introduce an augmentation procedure for pre-training to further improve representation learning in image-based tasks. In Sec. 3.4.2, we tackle the scenario where global labels are absent by automatically inferring a notion of global labels. Lastly, we introduce a meta fine-tuning procedure in Sec. 3.4.3 to investigate how much meta-learning could improve the pre-trained representation.

### 3.4.1 Augmented Pre-Training for Image-Based Tasks

In general, pre-training is a standard process with well-studied techniques for improving the final learned representation. Many of these techniques, including data augmentation for image-based tasks [Che+18], auxiliary losses [Man+20] and model distillation [Tia+20], are also effective for FSL (i.e. the learned representation is suitable for novel classes during meta-testing). In particular, we may interpret data augmentation techniques as increasing $N$ in Sec. 3.3.6, thus improving the error incurred by pre-training and consequently the learned representation $\phi_\theta$.

Beyond standard augmentations (e.g. random cropping and color jittering) investigated in [Che+18], we further propose an augmented procedure for pre-training via image rotation. For every class $y \in \mathcal{Y}$ in the original dataset, we create three additional classes by rotating all images of class $y$ by $r \in \{90°, 180°, 270°\}$ respectively. All rotations are multiples of 90° such that they can be implemented by basic operations efficiently (e.g. flip and transpose) and prevent pre-training from learning any trivial features from visual artifacts produced by arbitrary rotations [GSK18]. Pre-training is then performed normally on the augmented dataset. Additionally, standard augmentations are also applied on the augmented dataset.

The augmented dataset quadruples the number of samples and classes

**Algorithm 2** MeLa

```
Input: meta-training set $\mathcal{T} = (D_t^{\text{tr}}, D_t^{\text{val}})_{t=1}^{T}$
    $\phi_\theta^{\text{sim}} = \text{argmin}_{\phi_\theta} \mathbb{E}_{(D^{\text{tr}}, D^{\text{val}}) \in \mathcal{T}} \hat{\mathcal{R}}(w(\phi_\theta(D^{\text{tr}})), \phi_\theta(D^{\text{val}}))$
    Global clusters $G = \text{LearnLabeler}(\mathcal{T}, \phi_\theta^{\text{sim}})$
    $\phi_\theta^{\text{pre}} = \text{Pretrain}(D(\mathcal{T}), G)$
    $\phi_\theta^{*} = \text{MetaLearn}(G, \mathcal{T}, \phi_\theta^{\text{pre}})$
Return $\phi_\theta^{*}$
```

compared to the original dataset. According to our analysis from Sec. 3.3.6, pre-training on the augmented dataset may yield a more robust representation. Further, we also hypothesize that the quality of the representation also depends on the number of classes available in the pre-training dataset, since classifying more classes requires learning increasingly discriminating representations. Our experiments show that 1) augmented pre-training consistently outperforms the standard one, and 2) quality of the learned representation depends on both the dataset size and the number of classes available for training.

### 3.4.2 Meta Label Learning

The ability to exploit pre-training crucially depends on access to global labels. However, it is problematic to assume easy access to global labels. As discussed in Sec. 3.2, *global labels may be unavailable or inaccessible in practical applications*, when meta-training tasks are collected and annotated independently. As we will illustrate with the experiment in Sec. 3.5.4, different tasks may present conflicting labels over the same set of data based on different task requirements. This leads to ill-defined global labels and makes pre-training not directly applicable. Therefore, we consider the more general setting where only local labels from each task are known. This setting was also adopted by most of earlier meta-learning methods [e,g FAL17; SSZ17; Vin+16; Ber+19; Lee+19]. In the local label setting, we propose Meta Label Learning in order to automatically infer a notion of latent global labels across tasks. Naturally, the inferred labels enables pre-training and bridges the gap between the experiment settings with and without global labels.

Alg. 2 outlines the general strategy for learning a few-shot model using

MeLa. We first meta-learn an initial representation $\phi_\theta^{\text{sim}}$. Secondly, we cluster all task samples using $\phi_\theta^{\text{sim}}$ as a feature map while enforcing local task constraints. The learned clusters are returned as inferred global labels. Using the inferred labels, we can apply pre-training to obtain $\phi_\theta^{\text{pre}}$, which may be further fine-tuned using meta-learning objectives to derive the final few-shot model $\phi_\theta^*$. We present in Sec. 3.4.3 a simple yet effective meta fine-tuning procedure. The similarity metric used in this algorithm is simply the Euclidean metric of the feature mapped inputs, so for two inputs $x$, $x'$ and a feature map $\phi_\theta$, the similarity metric is $d(x, x') = \|\phi_\theta(x) - \phi_\theta(x')\|_2$. All feature maps are ResNet models and the feature space output dimension $m$ is 640, see Appendix A.2 for specific details.

For learning $\phi_\theta^{\text{sim}}$, we directly optimize the empirical meta-risk using ridge regression (2.16) as the base learner. We use ridge regression for its computational efficiency and good performance. Using $\phi_\theta^{\text{sim}}$ as a base for a similarity measure, the labeling algorithm takes as input the meta-training set and outputs a set of clusters as global labels. The algorithm consists of a clustering routine for sample assignment and centroid updates, and a pruning routine for merging small clusters.

#### 3.4.2.1 Clustering

The clustering routine leverages local labels for assigning task samples to appropriate global clusters and enforcing task constraints. We observe that for any task, the local labels describe two constraints: 1) samples sharing a local label must be assigned to the same global cluster, while 2) samples with different local labels must not share the same global cluster. To meet constraint 1, we assign all samples $\{x_i^{(j)}\}_{i=1}^n$ of class $y_\rho^{(j)}$ to a single global cluster by

$$v^* = \operatorname*{argmin}_{v \in \{1, \dots, V\}} \left\| \frac{1}{n} \sum_{i=1}^{n} \phi_\theta^{\text{sim}}(x_i^{(j)}) - g_v \right\|^2. \tag{3.12}$$

with $V$ being the current number of centroids.

We apply (3.12) to all classes $y_\rho^{(1)}, \dots, y_\rho^{(k)}$ and all instances within a task.

If multiple local classes map to the same global label, we simply discard the task to meet constraint 2. Otherwise, for each task class $j$ and the corresponding $v^*$ centroid index, we proceed to update the centroids $g_{v^*}$ and sample count $N_{v^*}$ using

$$g_{v^*} \leftarrow \frac{N_{v^*} g_{v^*} + \sum_{i=1}^{n} \phi_\theta^{\text{sim}}(x_i^{(j)})}{N_{v^*} + n} \quad \text{and} \quad N_{v^*} \leftarrow N_{v^*} + n. \tag{3.13}$$

### 3.4.2.2 Pruning

We also introduce a strategy for pruning small clusters. We model the sample count of each cluster as a binomial distribution $N_v \sim B(Tn, p)$, due to having $T$ total number of tasks and each task having $n$ classes. Assuming no collisions within each task, and that each class of each task is assigned to cluster independently we have that $N_v \sim B(Tn, p)$. We set $p = \frac{1}{V}$, assuming that each cluster is equal likely to be matched by a local class of samples. Any cluster with sample count below the following threshold is discarded,

$$N_v < \bar{N_v} - q\sqrt{\text{Var}(N_v)}, \tag{3.14}$$

where $\bar{N_v}$ is the expectation of $N_v$, $\text{Var}(N_v)$ the variance, and $q$ a hyperparameter controlling how aggressive the pruning is. This procedure can be seen as a way to remove outliers since it removes clusters which have improbably few classes assigned to them. In general, a lower value of $q$ prunes the number of cluster more aggressively which may potentially lead to zero clusters which is a failure mode of the algorithm.

Alg. 3 outlines the full labeling algorithm. We first initialize a large number of clusters by setting their centroids with mean class embeddings from random classes in $\mathcal{T}$. For $V$ initial clusters, $\lceil \frac{V}{k} \rceil$ tasks are needed since each task contains $k$ classes and could initialize as many clusters. The algorithm then alternates between clustering and pruning to refine the clusters and estimate the number of clusters jointly. The algorithm terminates and returns the current clusters $G$ when the number of clusters does not change from the

**Algorithm 3** LearnLabeler

```
Input: embedding model $\phi_\theta^{\text{sim}}$, meta-training set $\mathcal{T} = (D_t^{\text{tr}}, D_t^{\text{val}})_{t=1}^{T}$, number
    of classes in a task $k$
Initialization: sample tasks from $\mathcal{T}$ to initialize clusters $G = \{g_v\}_{v=1}^{V}$,
  While $|G|$ has not converged:
    $N_v = 1$ for each $g_v \in G$
    For $(D^{\text{tr}}, D^{\text{val}}) \in \mathcal{T}$:
      Match $D^{\text{tr}} \cup D^{\text{val}}$ to its centroids $M = \{g_q\}_{q=1}^{V}$ using (3.12)
      If $M$ has $k$ unique clusters
        Update centroid $g_q$ for each $g_q \in M$ via (3.13)
    $G \leftarrow \{g_v \mid g_v \in G, N_v \geq \text{threshold in (3.14)}\}$
Return $G$
```

previous iteration. Using clusters $G$, local classes from the meta-training set can be assigned global labels with nearest neighbor matching using (3.12). For tasks that fail to map to $k$ unique global labels, we simply exclude them from the pre-training process.

The key difference between Alg. 3 and the classical $K$-means algorithm [Llo82] is that the proposed clustering algorithm exploits local information to guide the clustering process, while $K$-means algorithm is fully unsupervised. We will show in the experiments that enforcing local constraints is necessary for learning robust meta-representation.

Alg. 3 also indirectly highlights how global labels, if available, offers valuable information about meta-training set. In addition to revealing precisely how input samples relate to one another across tasks, global labels provides an overview of meta-training set, including the desired number of clusters and their sizes. In contrast, Alg. 3 needs to estimate both properties when only local labels are given.

### 3.4.3 Meta Fine-Tuning

As discussed in Sec. 3.3, while pre-training already yields a robust metra-representation for FSL, GLS, the base learner associated with pre-training is inapplicable for meta-testing when novel classes are presented. It is thus desirable to adapt the pre-trained representation by directly optimizing the empirical meta-risk, such that the new meta-representation better matches the base learner intended for novel classes. We call this additional training meta fine-tuning, which is adopted by several state-of-the-art FSL models [Zha+20; Ye+20; WDC20; LLB21].

For meta fine-tuning, existing works suggest that model performance depends crucially on preserving the pre-trained representation. In particular, [Rus+18; Ye+20; WDC20; LLB21] all keep the pre-trained representation fixed, and only learn a relatively simple transformation on top for the new base learners. Additionally, [Gol+20] showed that meta fine-tuning the entire representation model using MetaOptNet [Lee+20] or R2D2 [Ber+19] lead to worse performance compared to standard meta-learning, negating the advantages of pre-training completely.

Given the observations above, we present a simple residual architecture that preserves the pre-trained embeddings and allows adaptation for the new base learner. Formally, we consider the following parameterization for a fine-tuned meta-learned embedding $\phi^*_\theta$,

$$\phi^*_\theta(x) = \phi^{\text{pre}}_\theta(x) + h(\phi^{\text{pre}}_\theta(x)), \tag{3.15}$$

where $\phi^{\text{pre}}_\theta$ is the pre-trained representation and $h$ a learnable function (e.g. a small fully connected network). We again use (2.16) as the base learner and optimize the empirical meta-risk directly. Our experiments show that the proposed fine-tuning process achieves results competitive with more sophisticated base learners, indicating that the pre-trained representation is the predominant contributor to good test performance.

## 3.5 Experiments

We evaluate MeLa on various benchmark datasets and compare it with existing methods. The experiments are designed to address the following questions:

- How does MeLa compare to existing methods for generalization performance? Additionally, we also introduce the more challenging GFSL setting in Sec. 3.5.2.

- How do different model components (e.g. pre-training, meta fine-tuning) contribute to generalization performance?
- What is the additional cost in terms of time and memory of using MeLa?
- Does MeLa learn meaningful clusters? Can MeLa handle conflicting task labels?
- Given the importance of pre-training, how can we improve the quality of the pre-trained representation?
- How robust is MeLa to hyper-parameter choices?

### 3.5.1 Benchmark Datasets

**Mini/tiered-ImageNet.** [Vin+16; Ren+18] has become one of the default benchmark for FSL. Both datasets are subsets of ImageNet [Rus+15] with *mini*ImageNet having 60K images over 100 classes, and *tiered*ImageNet having 779K images over 608 classes. Following previous works, we report performance on 1 and 5-shot settings, using 5-way classification tasks.

#### 3.5.1.1 Variants of mini/tiered-ImageNet

We introduce several variants of mini/tiered-ImageNet to better understand MeLa and more broadly the impacts of dataset configuration on pre-training. Specifically, we create mini-60 that consists of 640 classes and 60 samples per class. The base dataset of mini-60 contains the same number of samples as the base dataset of *mini*ImageNet, though with more classes and fewer samples per class. Mini-60 is deliberately constructed so that the classes of the meta-train, validation and test sets of *mini*ImageNet are contained in the classes of the meta-train, validation and test set respectively, of mini-60, enabling a fair comparison of test performance of model trained on each dataset in turn. We designed mini-60 to investigate the behavior of MeLa when encountering a dataset with a high number of base classes and low number of samples per base class. We also use mini-60 to explore how data diversity present in the training data affects the learned representation. Analogous to mini-60, we also introduce tiered-780 as a variant to *tiered*IMAGENET, where we take the total number of samples in *tiered*IMAGENET and calculate the number of samples over the full 1000 ImageNet classes while avoiding meta-test set overlap between the two datasets.

#### 3.5.1.2 Meta-Dataset

[Tri+20] is a meta-learning classification benchmark combining 10 widely used datasets: ILSVRC-2012 (ImageNet) [Rus+15], Omniglot [LST15], Aircraft [Maj+13], CUB200 [Wel+10], Describable Textures (DTD) [Cim+14], QuickDraw [Jon+16], Fungi [SC18], VGG Flower (Flower) [NZ08], Traffic Signs [Hou+13] and MSCOCO [Lin+14]. We use Meta-Dataset to construct several challenging experiment scenarios, including learning a unified model for multiple domains and learning from tasks with conflicting labels.

### 3.5.2 Experiment Settings

The standard FSL setting [FAL17; SSZ17; Ye+20; WTH21; Bat+22] assumes that a meta-distribution of tasks is available for training. This translates to meta-learners having access to an exponential number of tasks synthetically generated from the underlying dataset, a scenario unrealistic for practical applications. Recent works additionally assume access to global labels in order to leverage pre-training, in contrast with earlier methods that assume access to only local labels. We will highlight such differences when comparing different methods.

#### 3.5.2.1 GFSL Setting

This is a more challenging and realistic FSL setting. Specifically, we only allow access to local labels, since global labels may be inaccessible or ill-defined. In addition, we employ a *no-replacement* sampling scheme when synthetically generating tasks from the underlying dataset[1]. This sampling protocol limits

[1]For instance, *mini*IMAGENET (38400 training samples) will be randomly split into around 380 tasks of 100 samples

the meta-training set to a fixed-size, which is a standard assumption for most machine learning problems. The fixed size also enables us to evaluates the sample efficiency of different methods. Secondly, no-replacement sampling prevents MeLa and other meta-learners from trivially learning task relations, a key objective of meta-learning, by matching same samples across tasks. For instance, an identical sample appearing in multiple tasks would allow MeLa to trivially cluster local classes. Lastly, the sampling process reflects any class imbalance in the underlying dataset, which might present a more challenging problem.

### 3.5.3 Performance Comparison in Standard Setting

We compare MeLa to a diverse group of existing methods on mini- and *tiered*IMAGENET in Table 3.1. We separate the methods into those requiring global labels and those that do not. We note that the two groups of methods are not directly comparable since global labels provides a significant advantage to meta-learners as discussed previously. The method groupings are intended to demonstrate the effect of pre-training on generalization performance.

Table 3.1 clearly shows that "global-labels" methods leveraging pre-training generally outperform "local-labels" methods except MeLa. We highlight that the re-implementation of ProtoNet in [WTH21] benefits greatly from pre-training, outperforming the original by over 10% across the two datasets. Similarly, while RFS and R2D2 both learn a fixed representation and only adapt the classifier based on each task, RFS's pre-trained representation clearly outperforms R2D2's meta-learned representation. We further note that state-of-the-art methods such as DeepEMD and FEAT are heavily reliant on pre-training and performs drastically worse in GFSL setting, as we will discuss in Sec. 3.5.4.

In the local-labels category, MeLa outperforms existing methods thanks to its ability to still exploit pre-training using the inferred labels. MeLa achieves about 4% improvement over the next best method in all settings. Across both categories, MeLa obtains performance competitive to state-of-the-art methods

Table 3.1: Test accuracy of meta-learning models on *mini*ImageNet and *tiered*ImageNet.

| | *mini*ImageNet | | *tiered*ImageNet | |
|---|---|---|---|---|
| | 1-shot | 5-shot | 1-shot | 5-shot |
| Global Labels | | | | |
| Simple CNAPS [Bat+22] | 53.2 ± – | 70.8 ± – | 63.0 ± – | 80.0 ± – |
| LEO [Rus+18] | 61.7 ± 0.7 | 77.6 ± 0.4 | 66.3 ± 0.7 | 81.4 ± 0.6 |
| TASML [WDC20] | 62.0 ± 0.5 | 78.2 ± 0.5 | 66.4 ± 0.4 | 82.6 ± 0.3 |
| RFS [Tia+20] | 62.0 ± 0.4 | 79.6 ± 0.3 | 69.4 ± 0.5 | 84.4 ± 0.3 |
| ProtoNet (with pre-train) [WTH21] | 62.4 ± 0.2 | 80.5 ± 0.1 | 68.2 ± 0.2 | 84.0 ± 0.3 |
| FEAT [Ye+20] | **66.7 ± 0.2** | 82.0 ± 0.1 | 70.8 ± 0.2 | 84.8 ± 0.2 |
| FRN [WTH21] | 66.4 ± 0.2 | **82.8 ± 0.1** | 71.2 ± 0.2 | **86.0 ± 0.2** |
| DeepEMD [Zha+20] | 65.9 ± 0.8 | 82.4 ± 0.6 | **71.2 ± 0.9** | **86.0 ± 0.6** |
| Local Labels | | | | |
| MAML [FAL17] | 48.7 ± 1.8 | 63.1 ± 0.9 | 51.7 ± 1.8 | 70.3 ± 0.8 |
| ProtoNet [SSZ17] | 49.4 ± 0.8 | 68.2 ± 0.7 | 53.3 ± 0.9 | 72.7 ± 0.7 |
| R2D2 [Ber+19] | 51.9 ± 0.2 | 68.7 ± 0.2 | 65.5 ± 0.6 | 80.2 ± 0.4 |
| MetaOptNet [Lee+19] | 62.6 ± 0.6 | 78.6 ± 0.5 | 66.0 ± 0.7 | 81.5 ± 0.6 |
| Shot-free [RBS19] | 59.0 ± n/a | 77.6 ± n/a | 63.5 ± n/a | 82.6 ± n/a |
| MeLa (pre-train only) | 64.5 ± 0.4 | 81.5 ± 0.3 | 69.5 ± 0.5 | 84.3 ± 0.3 |
| MeLa | **65.8 ± 0.4** | **82.8 ± 0.3** | **70.5 ± 0.5** | **85.9 ± 0.3** |

Table 3.2: Test Accuracy on Aircraft, CUB and VGG Flower (Mixed dataset). A single model is trained for each method over all tasks.

| | Aircraft | | CUB | | VGG Flower | | Average | |
|---|---|---|---|---|---|---|---|---|
| | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot |
| ProtoNet | $35.1 \pm 0.4$ | $51.0 \pm 0.5$ | $32.7 \pm 0.4$ | $46.4 \pm 0.5$ | $56.7 \pm 0.5$ | $73.8 \pm 0.4$ | $41.7 \pm 0.7$ | $57.5 \pm 0.7$ |
| MatchNet | $31.4 \pm 0.4$ | $39.4 \pm 0.5$ | $42.7 \pm 0.5$ | $54.1 \pm 0.5$ | $62.5 \pm 0.5$ | $70.0 \pm 0.1$ | $45.7 \pm 0.1$ | $54.5 \pm 0.1$ |
| R2D2 | $67.7 \pm 0.6$ | $82.8 \pm 0.4$ | $53.8 \pm 0.5$ | $69.2 \pm 0.5$ | $65.4 \pm 0.5$ | $83.3 \pm 0.3$ | $61.9 \pm 0.5$ | $78.6 \pm 0.4$ |
| DeepEMD | $34.7 \pm 0.7$ | $47.8 \pm 1.4$ | $39.3 \pm 0.7$ | $52.1 \pm 1.4$ | $61.3 \pm 0.9$ | $74.5 \pm 1.4$ | $45.1 \pm 0.7$ | $58.1 \pm 1.2$ |
| FEAT | $61.7 \pm 0.6$ | $75.8 \pm 0.5$ | $59.6 \pm 0.6$ | $73.1 \pm 0.5$ | $62.9 \pm 0.6$ | $76.0 \pm 0.4$ | $60.9 \pm 0.7$ | $75.0 \pm 0.5$ |
| FRN | $60.7 \pm 0.7$ | $77.6 \pm 0.5$ | $61.9 \pm 0.7$ | $77.7 \pm 0.5$ | $65.2 \pm 0.6$ | $81.2 \pm 0.5$ | $63.1 \pm 0.7$ | $79.7 \pm 0.5$ |
| MeLa | $\mathbf{78.2 \pm 0.5}$ | $\mathbf{89.5 \pm 0.3}$ | $\mathbf{73.8 \pm 0.6}$ | $\mathbf{88.7 \pm 0.3}$ | $\mathbf{76.6 \pm 0.4}$ | $\mathbf{91.5 \pm 0.2}$ | $\mathbf{76.2 \pm 0.3}$ | $\mathbf{89.9 \pm 0.2}$ |

such as FRN, FEAT and DeepEMD despite having no access to global labels. This indicates that MeLa is able to infer meaningful clusters to substitute global labels and obtains performance similar to methods having access to global labels. We will provide further quantitative results on the clustering algorithm in Sec. 3.5.7.

### 3.5.4 Performance Comparison in Generalized Setting

We evaluate a representative set of few-shot learners under GFSL. For this setting, we also introduce two new experimental scenario using Meta-Dataset to simulate task heterogeneity.

In the first scenario, we construct the meta-training set from Aircraft, CUB and VGG flower, which we simply denote by "Mixed". Tasks are sampled independently from one of the three datasets. For meta-testing, we sample 1500 tasks from each dataset and report the average accuracy. The chosen datasets are intended for fine-grained classification in aircraft models, bird species and flower species respectively. Thus the meta-training tasks share the broad objective of fine-grained classification, but are sampled from three distinct domains. A key challenge of this scenario is to learn a unified model across multiple domains, without any explicit knowledge about them or the global labels within each domain.

The results show that MeLa outperforms all baselines under GFSL setting. In particular, MeLa achieves a large margin of 10% improvement over the baselines, including state-of-the-art models FEAT, FRN and DeepEMD, the methods equal to MeLa in Table 3.1. In particular, FEAT and DeepEMD performed noticeably worse, indicating the methods' reliance on pre-trained representation and the difficulty of meta-learning robust representations from scratch with complex base learners. FRN is designed to also work without pre-training, and outperforms FEAT and DeepEMD as expected.

In the second scenario, we consider meta-training tasks with heterogeneous objectives, leading to conflicting task-labels and consequently ill-defined global labels. For the Aircraft dataset, each sample from the base dataset has

Table 3.3: Test accuracy on H-Aircraft in the generalized setting.

| | 1-shot | 5-shot |
|---|---|---|
| ProtoNet | $47.8 \pm 0.5$ | $66.8 \pm 0.5$ |
| MatchNet | $65.6 \pm 0.2$ | $78.7 \pm 0.2$ |
| R2D2 | $75.1 \pm 0.3$ | $86.4 \pm 0.2$ |
| DeepEMD | $51.3 \pm 0.5$ | $65.6 \pm 0.8$ |
| FEAT | $77.6 \pm 0.6$ | $87.3 \pm 0.4$ |
| FRN | $81.9 \pm 0.4$ | $91.0 \pm 0.2$ |
| MeLa | $\mathbf{84.8 \pm 0.3}$ | $\mathbf{92.9 \pm 0.2}$ |
| Oracle | $84.4 \pm 0.3$ | $93.1 \pm 0.2$ |

three labels associated with it, including variant, model and manufacturer[2] that form a hierarchy. We sample tasks based on each of the three labels and creates a meta-training set containing three different task objectives: classifying fine-grained differences between model variants, classifying different airplanes, and classifying different airplane manufacturers. To differentiate from the original dataset, we refer to this meta-training set as H-Aircraft. The training data is particularly challenging given the competing goals across different tasks: a learner is required to recognize fine-grained differences between airplane variants, while being able to identify general similarities within the same manufacturer. The training data also exhibits class imbalance. For instance, the dataset is dominated by samples from Boeing and Airbus and the meta-training set reflects that in GFSL setting.

Table 3.3 shows that MeLa outperforms all baselines for H-Aircraft. To approximate the oracle performance when ground truth labels were given, we optimize a supervised semantic softmax loss [Rid+21] over the hierarchical labels. Specifically, we train the (approximate) oracle to minimize a multi-task objective combining individual cross entropy losses over the three labels. MeLa performs competitively against the oracle, indicating the robustness of the proposed labeling algorithm in handling ill-defined labels and class imbalance.

[2] E.g. “Boeing 737-300” indicates manufacturer, model, and variant

The experimental results suggest that MeLa performs robustly in both the standard and GFSL settings. In contrast, baseline methods perform noticeably worse in the latter, due to the absence of pre-training and limited training data.

#### 3.5.4.1 Connection to Theoretical Results

We comment on the empirical results so far in relation to our theoretical analysis. The empirical results strongly indicate that pre-training produces robust meta-representations for FSL by exploiting contextual information from global labels. This is consistent with our observation that pre-training would achieve a smaller error than its meta-learning counterpart. On the other hand, the results also validate our hypothesis that the pre-trained representation can be further improved, since the pre-trained representation is not explicitly optimized for handling novel classes. In particular, FEAT, FRN, DeepEMD and MeLa all outperform the pre-trained representation from [Tia+20] by further adapting it.

### 3.5.5 Timings and Complexity of MeLa

We evaluate the time it takes to run the MeLa pipeline specified in Alg. 2 both in terms of actual time running on a computer (wall-time) in the table of Table 3.5 and computational and memory complexity derived in Appendix A.3.

In terms of wall-time, the table Table 3.5 shows that across all of the datasets used (Mixed, *mini*ImageNet and *tiered*ImageNet), the proportion of time the algorithm spend in each part is approximately the same. About 20% of the time is spent in the RepLearn (representation learning), $0-5\%$ is spent in the LearnLabeler (inferring pseudo-labels through the clustering and labelling algorithm), around 77% spent in the PreTrain step (pretraining with the inferred labels) and about $0.5-2\%$ spent in the MetaLearn step (meta-learning step after pre-training). In all, this shows that the additional steps of RepLearn and LearnLabeler increase the training time by about 20% over the standard procedure which seems like a small price to pay considering that pretraining in the GFSL setting is not possible at all without performing RepLearn and LearnLabeler (or some other potential labelling procedure). Finally, the reason for the comparably smaller cost of RepLearn versus PreTrain is that RepLearn does not require the class-augmentation which increase the size of the dataset by 4, and for the *tiered*ImageNet dataset, requires a smaller residual network than that of the pretraining step (ResNet12 vs ResNet18).

Additionally, we tabulate the time and memory complexity below for one update step with batch size 1 for the different parts of the training pipeline Alg. 2. For a representation network $\phi_\theta : \mathcal{X} \to \mathbb{R}^{d_\phi}$, we let the computational and memory complexity of mapping an instance $x$ to $\phi_\theta(x)$ be given by $O(k_\phi)$ and $O(m_\phi)$ respectively. As $w$ is KRR, it has a computational and memory complexity of $O(d_\phi^2 n_{\text{tr}} + d_\phi^3 + d_\phi n_{\text{tr}} N_{\text{way}})$ and $O(d_\phi N_{\text{way}})$ respectively which depend on the train set size $n_{\text{tr}}$ and input dimension $d_\phi$ and the number of ways $N_{\text{way}}$. The validation set size is $n_{\text{val}}$ and we let $n = n_{\text{tr}} + n_{\text{val}}$.

### 3.5.6 Ablations on Pre-Training

Given the significance of pre-training on final performance, we investigate how the rotation data augmentation and data configuration impact the performance of the pre-trained representation. For dataset configuration, we focus on the effects of dataset sizes and the number of classes present in the dataset.

#### 3.5.6.1 Rotation-Augmented Pre-Training

In Sec. 3.4.1, we proposed to increase both the size and the number of classes in a dataset via input rotation. By rotating the input images by the multiples of 90°, we quadruple both the size and the number of classes in a dataset. In Table 3.4, we compare the performance of standard pre-training against the rotation-augmented one, for multiple datasets. We use the inferred labels from MeLa for pre-training.

Table 3.5: Wall-time in seconds for all stages of Alg. 2 using the hyper-parameters in Table A.1.

| Dataset | RepLearn | LearnLabeler | PreTrain | MetaLearn | Total |
|---|---|---|---|---|---|
| Mixed | | | | | |
| Time (s) | 1408 | 28 | 5031 | 72 | 6539 |
| Percentage | 21.5 | 0.4 | 77 | 1.1 | – |
| *mini*ImageNet | | | | | |
| Time (s) | 1668 | 56 | 6108 | 144 | 7976 |
| Percentage | 20.9 | 0.7 | 76.6 | 1.8 | – |
| *tiered*ImageNet | | | | | |
| Time (s) | 15012 | 4002 | 63384 | 384 | 82782 |
| Percentage | 18.1 | 4.8 | 76.6 | 0.5 | – |

Table 3.6: Computational and memory complexity of one update for the different steps in the train pipeline Alg. 2. $C_{\text{global}}$ is the number of global classes.

| Step | Computational | Memory |
|---|---|---|
| RepLearn | $O(nk_{\phi^{\text{sim}}} + d^2_{\phi^{\text{sim}}} n_{\text{tr}} + d^3_{\phi^{\text{sim}}} + d_{\phi^{\text{sim}}} n N_{\text{way}})$ | $O(m_{\phi^{\text{sim}}} + d_{\phi^{\text{sim}}}(n + N_{\text{way}}))$ |
| LearnLabeler | $O((n_{\text{shots}} + V) N_{\text{way}} d_{\phi^{\text{sim}}})$ | $O(N_{\text{way}} d_{\phi^{\text{sim}}})$ |
| PreTrain | $O(k_\phi)$ | $O(m_\phi + C)$ |
| MetaLearn | $O(n(k_{\phi^{\text{pre}}} + k_{\phi^*}) + d^2_{\phi^*} n_{\text{tr}} + d^3_{\phi^*} + d_{\phi^*} n N_{\text{way}})$ | $O(m_{\phi^{\text{pre}}} + m_{\phi^*} + d_{\phi^*}(n + N_{\text{way}}))$ |

Table 3.4: Test accuracy comparison between pre-trained representations: standard vs. rotation-augmented.

| | 1-shot | | 5-shot | |
|---|---|---|---|---|
| | standard | rotation | standard | rotation |
| *mini*IMAGENET | $62.0 \pm 0.4$ | $64.5 \pm 0.4$ | $79.6 \pm 0.3$ | $81.5 \pm 0.3$ |
| mini-60 | $63.9 \pm 0.7$ | $67.7 \pm 0.5$ | $81.5 \pm 0.5$ | $84.1 \pm 0.5$ |
| *tiered*IMAGENET | $69.1 \pm 0.5$ | $69.5 \pm 0.6$ | $83.9 \pm 0.3$ | $84.3 \pm 0.4$ |
| tiered-780 | $78.0 \pm 0.6$ | $78.2 \pm 0.6$ | $89.9 \pm 0.4$ | $90.1 \pm 0.4$ |
| H-Aircraft | $79.2 \pm 0.5$ | $84.8 \pm 0.3$ | $89.4 \pm 0.3$ | $92.9 \pm 0.2$ |

The results suggest that rotation-augmented pre-training consistently improves the quality of the learned representation. It achieves over 3% improvements in both *mini*IMAGENET and H-aircraft, while obtaining about 0.5% in *tiered*IMAGENET. It is clear that rotation augmentation works the best with smaller datasets with fewer classes. As the dataset increases in size and diversity, the additional augmentation has less impact on the learned representation.

#### 3.5.6.2 Effects of Class Count

We further evaluate the effects of increasing number of classes in a dataset while maintaining the dataset size fixed. For this, we compare the performance of *mini*IMAGENET and *tiered*IMAGENET with their respective variants mini-60 and tiered-780.

Table 3.4 suggests that given a fixed size dataset, having more classes improves the quality of the learned representation compared to having more samples per class. We hypothesize that classifying more classes lead to more discriminative and robust features, while standard $\ell_2$-regularization applied during pre-training prevents overfitting despite having fewer samples per class.

Overall, the experiments suggest that pre-training is a highly scalable process where increasing either data diversity or dataset size will lead to more ro-

Table 3.7: The effects of no-replacement sampling on the clustering algorithm.

| Dataset | *mini*ImageNet | | *tiered*ImageNet | | mini-60 | |
|---|---|---|---|---|---|---|
| Replacement | Yes | No | Yes | No | Yes | No |
| Tasks Clustered (%) | 100 | 98.6 | 99.9 | 89.5 | 98.7 | 97.8 |
| Clustering Acc (%) | 100 | 99.5 | 96.4 | 96.4 | 72.8 | 70.1 |
| 1-shot Acc (%) | $65.8 \pm 0.4$ | $65.8 \pm 0.5$ | $70.5 \pm 0.5$ | $70.5 \pm 0.5$ | $68.4 \pm 0.7$ | $68.4 \pm 0.5$ |
| 5-shot Acc (%) | $82.8 \pm 0.3$ | $82.8 \pm 0.4$ | $85.9 \pm 0.3$ | $85.9 \pm 0.3$ | $84.0 \pm 0.5$ | $84.0 \pm 0.5$ |

bust representation for FSL. In particular, the number of classes in the dataset appears to play a more significant role than the dataset size.

### 3.5.7 Ablations on the Clustering Algorithm

The crucial component of MeLa is Alg. 3, which infers a notion of global labels and allows pre-training to be exploited in GFSL setting. We perform several ablation studies to better understand the proposed clustering algorithm.

#### 3.5.7.1 The Effects of No-Replacement Sampling

We study the effects of no-replacement sampling, since it affects both the quality of the similarity measure through $\phi_\theta^{\text{sim}}$ and the number of tasks available for inferring global clusters. The results are shown in Table 3.7.

In Table 3.7, clustering accuracy is computed by assigning the most frequent ground truth label in each cluster as the desired target. Percentage of tasks clustered refers to the tasks that map to $k$ unique clusters by Alg. 3. The clustered tasks satisfy both constraints imposed by local labels and are used for pre-training

For both sampling processes, MeLa achieves comparable performances across all three datasets. This indicates the robustness of Alg. 3 in inferring suitable labels for pre-training, even when task samples do not repeat across tasks. This shows that Alg. 3 is not trivially matching identical samples across tasks, but relying on $\phi_\theta^{\text{sim}}$ for estimating sample similarity. We note that mini-60 is particularly challenging under no-replacement sampling, with only 384 tasks in the meta-training set over 640 ground truth classes.

#### 3.5.7.2 Effects of Pruning Threshold

In Alg. 3, the pruning threshold is controlled by the hyper-parameter $q$. We investigate how different $q$ values affect the number of clusters estimated by the labeling algorithm and the corresponding test accuracy.

The results in Table 3.8 suggest that MeLa is robust to a wide range of $q$ and obtains representations similar to that produced by the ground truth labels. While it is possible to replace $q$ with directly guessing the number of clusters in Alg. 3, we note that tuning for $q$ is more convenient since appropriate $q$ values appear to empirically concentrate within a much narrower range, compared to the possible to numbers of global clusters present in a dataset. In practice we tuned $q$ by hand in order to get a reasonable number of clusters and to avoid the degenerate case of pruning all clusters until no clusters remained.

Table 3.8: Test accuracy (pre-train only) and cluster count for various pruning thresholds, 5-shot setting.

| *mini*ImageNet (64 classes) Oracle Pre-train: 81.5% | | | *tiered*ImageNet (351 classes) Oracle Pre-train: 84.5% | | | mini-60 (640 classes) Oracle Pre-train: 85.2% | | |
|---|---|---|---|---|---|---|---|---|
| $q$ | No. Clusters | MeLa | $q$ | No. Clusters | MeLa | $q$ | No. Clusters | MeLa |
| 3 | 64 | $81.5 \pm 0.4$ | 3.5 | 351 | $84.3 \pm 0.4$ | 4.5 | 463 | $84.0 \pm 0.4$ |
| 4 | 75 | $81.1 \pm 0.4$ | 4.5 | 373 | $84.1 \pm 0.3$ | 5.5 | 462 | $83.8 \pm 0.4$ |
| 5 | 93 | $80.9 \pm 0.4$ | 5.5 | 444 | $84.0 \pm 0.5$ | 6.5 | 472 | $84.0 \pm 0.4$ |

#### 3.5.7.3 Inferred Labels vs. Oracle Labels

From Tables 3.7 and 3.8, we observe that it may be unnecessary to fully recover the oracle labels (when they exists). For mini-60, MeLa inferred 463 clusters over 640 classes, which implies mixing of the oracle classes. However, the inferred labels still perform competitively against the oracle labels, suggesting the robustness of the proposed method. The results also suggest that we may improve the recovery of the ground truth labels by sampling more tasks from the meta-distribution.

#### 3.5.7.4 The Importance of Local Constraints

The clustering process enforces consistent assignment of task samples given their local labels. To understand the importance of enforcing these constraints, we consider an ablation study where Alg. 3 is replaced with the standard $K$-means algorithm. The $K$-means algorithm is fully unsupervised and ignores any local constraints. We initialize the $K$-means algorithm with 64 clusters for *mini*ImageNet and 351 clusters for *tiered*ImageNet, the true numbers of classes in respective datasets.

Table 3.9: Test Accuracy (Pre-train only) using Alg. 3 vs. $K$-mean Clustering.

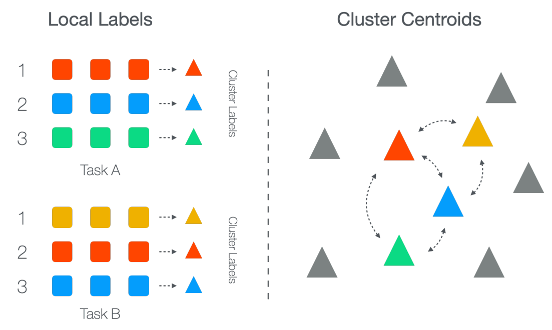
| | *mini*ImageNet | | | *tiered*ImageNet | | |
|---|---|---|---|---|---|---|
| Cluster Alg. | Cluster Acc | 1-shot | 5-shot | Cluster Acc | 1-shot | 5-shot |
| Alg. 3 (MeLa) | 100 | $64.5 \pm 0.4$ | $81.5 \pm 0.3$ | 96.4 | $69.5 \pm 0.5$ | $84.3 \pm 0.3$ |
| $K$-mean | 84.9 | $60.7 \pm 0.5$ | $76.9 \pm 0.3$ | 28.2 | $64.8 \pm 0.6$ | $78.8 \pm 0.5$ |

Table 3.9 indicates that enforcing local constraints is critical for generalization performance during meta-testing. In particular, test accuracy drops by over 5% for *tiered*ImageNet, when the $K$-means algorithm ignores local task constraints. Among the two constraints, we note that (3.12) appears to be the more important one since nearly all tasks automatically match $K$ unique clusters in our experiments (see tasks clustered in Table 3.7).

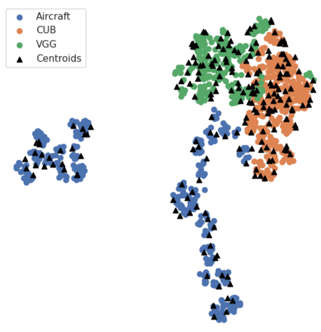
#### 3.5.7.5 Domain Inference for multi-domain tasks

In addition to inferring global labels, we may further augment Alg. 3 to infer the different domains present in a meta-training set, if we assume that all samples within a task belongs to a single domain. Given the assumption, two global clusters are connected if they both contain samples from the same task. This is illustrated in Fig. 3.2a. Consequently, inferred clusters form an undirected graph with multiple connected components, with each representing a domain. We apply the above algorithm to the multi-domain scenario from Sec. 3.5.4, where the meta-training set consists of Aircraft, CUB and VGG datasets.

Fig. 3.2b visualizes the inferred domains on the multi-domain scenario.



(a) Domain Inference via Connected Components



(b) UMAP visualization of different domains in the multi-domain dataset

Figure 3.2: **(a)** The coloured clusters (red, green, blue and yellow) are connected since they both contains samples from the same task. Domains can be inferred by computing the connected components of the inferred clusters. **(b)** UMAP visualization of the three inferred domains from the 5-shot Mixed dataset containing Aircraft, CUB and VGG. Circles are the means (using the pretrained features) of the instances in each task averaged per local class while triangles are the learned centroids, all vectors are embedded using UMAP. The three domains are recovered perfectly.

For each inferred cluster, we project its centroid into a 2-dimensional point using UMAP [McI+18]. Each connected component is assigned a different color. Despite some mis-clustering within each domain, we note that Alg. 3 clearly separates the three domains present in the meta-training set and recovers them perfectly.

Domain inference is important for multi-domain scenario as it enables domain-specific pre-training. Recent works [e.g. Liu+21; LLB21; DSM20] on Meta-Dataset have shown that combining domain-specific representation into a universal representation is empirically more advantageous than training on all domains together. Lastly, we remark that multi-domain meta-learning is also crucial for obtaining robust representation suitable for wider range of novel tasks, including cross-domain transfer.

### 3.5.8 Timings and Complexity Analysis

Chapter 4

# Implicit Kernel Meta-Learning

## 4.1 Summary

In this chapter, based on the paper [FCP22], we device a learning framework for learning the probability distribution representing a random feature kernel that we wish to use within KRR. We introduce two instances of this meta-learning framework, learning a neural network pushforward for a translation-invariant kernel and an affine pushforward for a neural network random feature kernel, both mapping from a Gaussian latent distribution. We learn the parameters of the pushforward by minimizing a meta-loss associated to the KRR objective. Since the resulting kernel does not admit an analytical form, we adopt a random feature sampling approach to approximate it.

We call the resulting method Implicit Kernel Meta-Learning (IKML). We derive a meta-learning bound for IKML, which shows the role played by the number of tasks $T$, the task sample size $n$, and the number of random features $M$. In particular the bound implies that $M$ can be the chosen independently of $T$ and only mildly dependent on $n$. We introduce one synthetic and two real-world meta-learning regression benchmark datasets. Experiments on these datasets show that IKML performs best or close to best when compared against competitive meta-learning methods.

### 4.1.1 Contribution

The principal contribution of this chapter is a method for meta-learning regression together with a bound on the excess risk which highlights how problem-specific quantities impact the number of random features needed to generalize. In particular, our method can be used to learn within a family of translation invariant kernels that is well-suited when using kernel ridge regression as the class of base learning algorithms. According to Bochner's Theorem [see e.g. RR+07], these kernels are parameterized by a distribution in the frequency space. In line with [Li+19], we parametrize this distribution as a neural network pushforward. The weights of the network are learned from a sequence of datasets within a meta-learning setting. Although we focus on distributions in the context of Bochner's theorem, our framework extends directly to radial kernels using Schoenberg's theorem [Sch38]. Additionally we experiment with using a neural network random feature kernel, an extension of R2D2 [Ber+19], and show competitive performance.

Finally, we introduce three novel meta-learning regression benchmark datasets, one synthetic and two real-world and show that our algorithm ranks at the top or close to competing meta-learning regression algorithms. We believe these results, including the theoretical guarantees together with the flexibility and ease of our method, make it a competitive candidate to be used as a plug-in meta-learning algorithm in general contexts.

A code repository for creating / downloading the datasets and run the implementation of the algorithm and benchmarks has been made available at `https://github.com/IsakFalk/IKML`.

### 4.1.2 Organization

Sec. 4.3 introduces the meta-learning representation setting for regression. We describe our proposed method in Sec. 4.4, analyze it in Sec. 4.5 and benchmark it in Sec. 4.6. discuss our findings in Sec. 5.2.2.

## 4.2 Introduction

There has been considerable interest in meta-learning for few-shot image classification [Vin+16; FAL17; SSZ17; Rus+18; Ren+18; Li+17; KZS15] but less attention has been given to designing meta-learning algorithms for regression. Most few-shot regression benchmarks fall under that of interpolating sinusoidals or a variety thereof [FAL17; OLL18; FXL18] which lacks many aspects of real-world regression problems such as being multivariate and noisy. This highlights the importance of more realistic meta-learning regression datasets and how to design meta-learning algorithms in this setting. The focus here is to close this gap.

Meta-learning algorithms employ a variety of different kinds of base algorithms, ranging from metric based to optimization based, and to black-box ones. A common theme is to learn a shared representation which lead to faster adaptation of a base learning algorithm to new tasks, indeed this feature map is integral for good performance in practice. Often the representation is modeled by a neural network. Recently [Tia+20; Rag+20] observed that the representation is the most important part of meta-learning algorithms.

In this section we build upon the above empirical observation and extend this thinking further. Explicitly, we device a meta-learning algorithm that learns a feature map implicitly in an RKHS with desirable properties by implicitly learning the representation via a kernel function from a large class of kernels defined by a random feature form. This kernel is in turn implicitly parametrized by a neural network pushforward which is learned by a meta-algorithm. When using the random feature family of translation-invariant kernels this has two main advantages: since kernel algorithms can be expressed in terms of inner products of features which are simple to compute we do not have to work with this high-dimensional feature space directly. We show that modeling the kernel directly leads to improved performance in the meta-learning regression case. A second advantage is that translation invariant kernels might be used as "plug-in" representations. We also experiment with using a neural network random feature representation, effectively combining ensembling with with random features.

### 4.2.1 Related Work

Recent advances in the image few-shot classification setting [FFP06; Lak+11] starting with the work of [FAL17; SSZ17; Vin+16] has lead to renewed interest in meta-learning, notably from the deep learning community by formulating it as an optimization problem [RL17]. While classification has received a lot of interest, regression has been given less attention. Some examples are given by [Tos+19; Tit+21; Pat+20] who apply Gaussian processes [WR06] together with deep kernel learning [Wil+16] to regression while in [Álv+10] they learn using an inducing function (a generalization of inducing points), similar to our frequency distribution, for multi-task Gaussian processes. From the ridge regression point of view; [Kon+20] investigate theoretically the meta-mixed linear regression setting while [NCL21] applied KRR to meta-learn dataset compression.

The ideas in this section can be traced directly from [Zhe+20; SD16; Li+19] which leverage the characterization provided by Bochner's theorem for kernel learning [OSW05; Cri+06]. In [SD16] they fine-tune a convex combination of sampled kernels in a supervised learning setting using kernel target alignment [Cri+06]. We also mention the work [Zhe+20] which apply variational inference to optimize a latent variable model for few-shot learning, and [Li+19] where they learn an implicit kernel using a pushforward in the case that the learning objective is linear in the kernel evaluations.

## 4.3 Meta-Representation Learning

We review the setting outlined in Sec. 2.3. The problem in meta-learning is to find the minimizer of the transfer risk $\mathcal{E}(\theta)$ when we only have access to a meta-train set $(D_t)_{t=1}^T$. The standard approach is to use the empirical meta-risk

$$\hat{\mathcal{E}}(\theta) = \frac{1}{T}\sum_{i=1}^{T} \hat{\mathcal{R}}(A(\theta, D_t^{\text{tr}}), D_t^{\text{val}})$$

instead of $\mathcal{E}(\theta)$ and try to solve the bilevel optimization problem of (2.10). If the meta-loss $L(\cdot, D)$ is (sub)differentiable, we can adopt standard stochastic first order method (e.g. SGD or Adam [KB15]) to approximate the optimal meta-parameters $\hat{\theta}_{\mathrm{opt}} = \operatorname{argmin} \hat{\mathcal{E}}(\theta)$.

In practice, the above approach might pose computational challenges since, by the chain rule, differentiating $L$ requires computing $\nabla_\theta A(\theta, D)$. Depending on the inner algorithm $A$, its gradient with respect to the meta-parameters $\theta$ might be hard to compute or not even exist. In the literature, a wide range of meta-learning strategies have been proposed, considering different choices of inner algorithm $A$ and meta-parameters $\theta$. In practice it is common to simply fix $A$ to be a learning procedure that operates on datasets, where we first map all of the inputs in the dataset through some feature map $\phi_\theta : \mathcal{X} \to \mathbb{R}^m$ (e.g. a neural network). This way of splitting the meta-learning algorithm into a feature map and a learning algorithm $A$ which only depend on $\theta$ through the feature mapped instances is called *meta-representation learning*, see Sec. 2.4, as the meta-learning problem is recast as one of finding a good feature map $\phi_\theta$ that works well when used with $A$ across tasks sampled from $\mu$.

For example, [Ber+19] considered the case that $A$ performs ridge regression and $\theta$ parameterizes the weights of a neural network. Leveraging the closed-form solution of the ridge regression estimator, this allows us to efficiently compute the gradient $\nabla_\theta A(\theta, D)$. In settings where $A$ is minimizing the empirical risk but with a loss function that does not admit a closed form, the bi-level optimization perspective [Fra+18] is often adopted which amounts to interpreting $A$ as returning the $T$-th iteration of an iterative optimization algorithm. This allows to access $\nabla_\theta A(\theta, D)$ by recursively differentiating along the iterates and use this to update the feature map in the outer loop.

## 4.4 Implicit Kernel Meta-Learning

We now introduce the proposed meta-learning strategy. While most previous work focused on learning a shared data representation or feature map [FAL17; Ber+19; Fra+18] across tasks, here we propose the dual approach of learning a shared kernel function.

### 4.4.1 Learning Translation Invariant Kernels

The definition of positive definite function underlying the notion of reproducing kernel is very general. Therefore, to formulate the problem of meta-learning a kernel, we need first to identify a suitable family. In [RR17] they introduce a "recipe" for random feature kernels defined by a random feature map $\varphi : \mathcal{X} \times \Omega \to \mathbb{R}^o$, where $o$ is the dimension of the space where $\varphi$ maps to (see (4.1) for how $\varphi$ is used with the kernel) and a distribution $\tau$ so that any kernel in this family has the form

$$K(x, x') = \int_{\Omega} \varphi(x, \omega)^{\top} \varphi(x', \omega) \, \mathrm{d}\tau(\omega). \tag{4.1}$$

Given the focus here towards regression settings, we first consider the class of *translation invariant* kernels, which are particularly suited to deal with such settings and are *interpretable* (see e.g. Fig. B.2). Let $\mathcal{X} = \mathbb{R}^d$. A kernel $K$ is called *translation invariant* if $K(x, x') = g(x - x')$ for some function $g : \mathbb{R}^d \to \mathbb{R}$; a well-known example is the Gaussian kernel $K(x, x') = e^{-\|x-x'\|^2/\sigma^2}$ with $\sigma > 0$. A famous theorem by Bochner [see e.g. RR+07; Rud62; SS15], adapted here to real-valued kernels, establishes that any properly re-scaled continuous bounded translation invariant function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel if and only if there exists a probability measure $\tau \in \mathcal{P}(\mathbb{R}^d)$ such that

$$K(x, x') = K_{\tau}(x, x') \equiv \int \cos(\langle \omega, x - x' \rangle) \, \mathrm{d}\tau(\omega), \tag{4.2}$$

which can be written in the form of (4.1) by expanding the cosine using the trigonometric identity $\cos(x - y) = \cos(x)\cos(y) + \sin(x)\sin(y)$. We call any kernel that can be written in the form of (4.1) a *Bochner kernel*. Eq. (4.2) implies that we can represent the class of translation invariant kernels as $\mathcal{T} = \{K_\tau \mid \tau \in \mathcal{P}(\mathbb{R}^d)\}$. Thus we can translate the problem of learning a kernel to that of learning a probability distribution. This perspective is in line with the implicit kernel learning approach devised in [Li+19] for generative modeling and single task settings. The second type of kernel is inspired by the success of using neural network to extract features and is given by letting $\varphi(x, \omega)$ be a neural network with $\omega$ the weights and $\tau$ a distribution over $\omega$.

#### 4.4.1.1 Pushforward Models

To learn the underlying distribution $\tau$ we consider a parametrization in terms of a pushforward model. More formally, let $\mathcal{N}$ be the unit Gaussian distribution over a latent space $\mathcal{Z}$ and let $\psi_\theta : \mathcal{Z} \to \mathbb{R}^d$ be a vector-valued function parameterized by a vector $\theta \in \Theta$ (e.g. a neural network with weights $\theta$). We denote by $\tau_\theta = \psi_\theta \# \mathcal{N}$ the probability distribution such that, the process of sampling $\omega \sim \tau_\theta$ is equivalent to first sampling $z \sim \mathcal{N}$ and then taking $\psi_\theta(z) = \omega$.[1] This is the strategy adopted to model the generator distribution in generative adversarial networks (GAN) settings and in the implicit kernel learning approach of [Li+19]. Several alternatives for the latent distribution $\mathcal{N}$ are possible (e.g. uniform). Under the notation above, we adopt as inner algorithm,

$$A(\theta, D) = A_{\mathrm{KRR}}(K_{\tau_\theta}, D), \tag{4.3}$$

namely KRR trained with a translational invariant kernel $K_{\tau_\theta}$ meta-parametrized by the pushforward map $\tau_\theta$. Below we give an example where a parametrization of $\tau_\theta$ yields an analytic form for the corresponding $K_{\tau_\theta}$; see Appendix B.1 for a derivation.

**Example 1** (Affine Pushforward Maps)**.** *Let $\theta = (Q, b)$ with $Q \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$ and consider the affine pushforward map $\psi_{(Q,b)}(s) = Qs + b$. In these*

[1] Formally, for any $B \subseteq \mathbb{R}^d$, $\tau_\theta(B) = \mathcal{N}(\{z \mid \psi_\theta(z) \in B\})$.

*settings, the kernel $K_{\tau_{(Q,b)}}$ can be expressed analytically as*

$$K_{\tau_{(Q,b)}}(x,x') = \cos(\langle b, x - x'\rangle) e^{-\left\|Q^\top (x-x')\right\|^2/2}. \tag{4.4}$$

The example above identifies a relevant family of kernels that are particularly amenable for meta-learning. Thanks to the analytic form of affine pushforward kernels, we can easily compute meta-gradients and thus directly minimize the transfer risk $\hat{\mathcal{E}}(\theta)$. On the other hand, if we consider more expressive maps $\psi_\theta$, we will hardly be able to obtain $K_{\tau_\theta}$ in analytic form. Still, this may be well worth the effort: while for large training sets the difference between (4.4) and a more sophisticate kernel may be less severe since any universal kernel is optimal [CD07], in the few-shot learning setting (where we have small training sets) the inductive bias plays an important role and being able to modify the kernel in a flexible way is key.

## 4.4.2 Stochastic Meta-Learning

The discussion above highlighted that except for a few special cases (see e.g. Example 1), given a distribution $\tau_\theta$ it is not possible to compute the kernel $K_{\tau_\theta}$ (and its gradient with respect to $\theta$) analytically. In principle, this might prevent us from applying meta-learning algorithms of the form in (4.3). To circumvent this issue, we consider a strategy based on random features [RR+07; RR17]. Rather than evaluating $K_{\tau_\theta}$, we sample a set $S = (s_j)_{j=1}^M$ from $\mathcal{N}$ and then approximate the ideal Bochner kernel by the *random features kernel*

$$K_{\hat{\tau}_{\theta S}}(x,x') = \frac{1}{M}\sum_{j=1}^{M} \cos(\langle \psi_\theta(s_j), x - x'\rangle), \tag{4.5}$$

where $\hat{\tau}_{\theta S} = \frac{1}{M}\sum_{j=1}^M \delta_{\psi_\theta(s_j)}$ is an empirical distribution associated to $\tau_\theta$ and $\delta_\omega$ denotes a Dirac's delta centered in $\omega \in \mathbb{R}^d$ which we call *frequency*. Thanks to the characterization of $K_{\tau_\theta}$ as an expectation in (4.2), we have that

$$K_{\tau_\theta}(x,x') = \mathbb{E}_{S\sim\mathcal{N}^M} K_{\hat{\tau}_{\theta S}}(x,x'), \tag{4.6}$$

namely $K_{\hat{\tau}_{\theta S}}$ is an unbiased estimator of $K_{\tau_\theta}$. Note that the notation $S \sim \mathcal{N}^M$ means that $S$ is a set of $M$ elements, each one sampled iid from $\mathcal{N}$ with replacement and $\mathbb{E}_{S\sim\mathcal{N}^M} f(S)$ means the expectation of $f(S)$ with respect to $S$ being sampled in this way. In addition to unbiasedness it is also possible to prove non-asymptotic results bounding the distance between the two kernels in sup norm [RR+07].

#### 4.4.2.1 Stochastic Meta-Learning

We now introduce a stochastic variant to the meta-learning approach from Sec. 2.3, by defining the meta-loss associated to a set of random features

$$L(\theta, S, D) = \hat{\mathcal{R}}(A_{\mathrm{KRR}}(K_{\hat{\tau}_{\theta S}}, D^{\mathrm{tr}}), D^{\mathrm{val}}), \tag{4.7}$$

and the corresponding transfer risk

$$\mathcal{E}_M(\theta) = \mathbb{E}_{\rho\sim\mu}\mathbb{E}_{D\sim\rho^n}\mathbb{E}_{S\sim\mathcal{N}^M} L(\theta, S, D), \tag{4.8}$$

which we will also denote $\mathcal{E}(\theta, S)$ when wanting to highlight the dependence on $S$ explicitly.

We propose to address the stochastic meta-learning problem

$$\min_{\theta\in\Theta} \mathcal{E}_M(\theta). \tag{4.9}$$

Alg. 4 (which is a version of Alg. 1) provides the pseudocode for a (meta) stochastic gradient descent algorithm applied to this problem. At each iteration $t' = 1, \ldots, T_{\mathrm{iter}}$, we sample a new task $\rho_{t'}$ and datasets $D_{t'} = (D^{\mathrm{tr}}_{t'}, D^{\mathrm{val}}_{t'}) \sim \rho^n_{t'}$ and a set of random features $S_{t'} \sim \mathcal{N}^M$ and then perform a gradient descent step in the direction of $\nabla_\theta L(\theta_{t'}, S_{t'}, D_{t'})$, but note that many other strategies to update the parameters using gradients are available, such as Adam [KB15]. The gradient can be computed by means of automatic differentiation (AutoGrad) [see e.g. Bay+18]. While using a large number of random features may seem expensive, both training and prediction times are linear in $M$,

**Algorithm 4** Implicit Kernel Meta-Learning

```
**Input:**
    meta-dataset $(D_t)_{t=1}^T$
    total number of iterations $T_{\text{iter}}$
    step-sizes $(\gamma_{t'})_{t'=1}^{T_{\text{iter}}}$,
    number of random features $M$,
    initial meta-parameters $\theta_0$,
**For** $t' = 1, \ldots, T_{\text{iter}}$
    Sample a task $D_{t'} = (D_{t'}^{\text{tr}}, D_{t'}^{\text{val}})$ from $(D_t)_{t=1}^T$
    Sample $M$ random features $S$ from $\mathcal{N}$
    Form $K_{\hat{\tau}_{\theta_{t'}S}}$ and compute $L(S, \theta_{t'}, D)$ as in (4.7)
    Get $\nabla_\theta L(\theta_{t'}, S, D) = \textsc{AutoGrad}(L(\cdot, S, D), \theta_{t'})$
    Update $\theta_{t'+1} \leftarrow \theta_{t'} - \gamma_{t'} \nabla L(\theta_{t'}, S, D)$
**Return** $\theta_{T_{\text{iter}}}$
```

see Appendix B.5 for a detailed analysis of the computational complexity and wall-time estimates. We refer to this method as *Implicit Kernel Meta-Learning (IKML)*.

## 4.5 Generalization Bound

We now study the generalization ability of the proposed meta-learning method. In particular, our goal is to study the effect of the number of random features on the performance of the meta algorithm. To present our observations we focus for simplicity on the case that the meta loss uses the task dataset for both training and validation, that is we use the empirical risk

$$\tilde{L}(\theta, S, D) = \hat{\mathcal{R}}(A_{\text{KRR}}(K_{\hat{\tau}_{\theta S}}, D), D), \tag{4.10}$$

which is the empirical error of KRR with kernel (4.5) on the dataset $D$ instead of (4.7). For a collection of datasets $(D_t)_{t=1}^T$ and a sample $S = (s_j)_{j=1}^M$ from $\mathcal{N}$, define the multitask empirical risk

$$\hat{\mathcal{E}}_T(\theta, S) = \frac{1}{T} \sum_{t=1}^{T} \tilde{L}(\theta, S, D_t). \tag{4.11}$$

We aim to bound the excess transfer risk

$$\mathcal{E}_M(\hat{\theta}) - \mathcal{E}(\theta^*), \tag{4.12}$$

where $\theta^* \in \Theta$ is such that $\mathcal{E}(\theta^*) = \min_\theta \mathcal{E}(\theta)$ and $\hat{\theta}$ is the minimizer of the multitask empirical risk, which we call the multitask empirical risk minimizer (MERM) which in practice we approximate by the solution returned by Alg. 4.

**Theorem 2.** *Assume that $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^d \times [0,1]$, $\mu$ is a meta-distribution on $\mathcal{Z}$, the loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$ and kernel family $\mathcal{K} = \{K_{\tau_\theta} \mid \theta \in \Theta\}$ is a family of Bochner kernels parameterized by some latent distribution $\mathcal{N}$ with support on $\mathbb{R}^l$ and a family of measurable functions $\{\psi_\theta : \mathbb{R}^l \to \mathbb{R}^d \mid \theta \in \Theta\}$. For any $n, M, T \in \mathbb{N}$ let the training task datasets $D_1, \dots, D_T$ be given by iteratively sampling a task $\rho_t \sim \mu$ and $D_t \sim \rho_t^n$ and $S \sim \mathcal{N}^M$, the family of inner algorithms being KRR with kernels $K_{\tau_\theta} \in \mathcal{K}$ and fixed regularization parameter $\lambda > 0$ and $\hat{\theta}$ being the MERM over the task datasets and random features. Then, for $\delta \in (0,1)$, with probability at least $1 - \delta$ over the datasets and random features*

$$\mathcal{E}_M(\hat{\theta}) - \mathcal{E}(\theta^*) \leq O\left(\frac{\sqrt{M} R_{n,M,T}}{T\lambda\sqrt{n}} + \sqrt{\frac{\log\frac{1}{\delta}}{T}}\right) + \tag{4.13}$$

$$O\left(\frac{1}{\lambda\sqrt{n}}\right) + \tag{4.14}$$

$$O\left(\frac{1}{\sqrt{M\lambda^3}}\left(1 + \sqrt{\frac{G_n^* \log n}{\lambda^2 n}}\right)\right) \tag{4.15}$$

*where*

$$R_{n,M,T} = \mathbb{E}_{(D_t)_{t=1}^T \sim \hat{\mu}^T} \mathbb{E}_{S,\epsilon} \sup_{\theta \in \Theta} \sum_{i,j,t}^{n,M,T} \epsilon_{i,j,t} \langle \psi_\theta(s_j), x_i^t \rangle, \tag{4.16}$$

*the random variables $\epsilon_{i,j,t}$ being i.i.d Rademacher and $D \sim \hat{\mu}$ means first sampling $\rho \sim \mu$ and then $D \sim \rho^n$, and $G_n^* = \mathbb{E}_{\rho\sim\mu}\mathbb{E}_{D\sim\rho^n}\left\|(K_{\theta^*}(x_i, x_j))_{i,j=1}^n\right\|_\infty$.*

*Proof Sketch.* We discuss the key elements of the proof and present the full details in Appendix B.2. We write $\mathcal{E}_M(\hat{\theta}) - \mathcal{E}(\theta^*) = \mathbb{E}_S[\mathcal{E}_M(\hat{\theta}, S) - \mathcal{E}(\theta^*)]$ and decompose the term inside the expectation as

$$\underbrace{\mathcal{E}(\hat{\theta},S)-\hat{\mathcal{E}}(\hat{\theta},S)}_{(A)}+\underbrace{\hat{\mathcal{E}}(\hat{\theta},S)-\hat{\mathcal{E}}_T(\hat{\theta},S)}_{(B)}+\underbrace{\hat{\mathcal{E}}_T(\hat{\theta},S)-\hat{\mathcal{E}}_T(\theta^*,S)}_{(C)}$$
$$+\underbrace{\hat{\mathcal{E}}_T(\theta^*,S)-\hat{\mathcal{E}}(\theta^*,S)}_{(D)}+\underbrace{\hat{\mathcal{E}}(\theta^*,S)-\mathcal{E}(\theta^*,S)}_{(E)}+\underbrace{\mathcal{E}(\theta^*,S)-\mathcal{E}(\theta^*)}_{(F)}$$

where $\hat{\mathcal{E}}(\theta,S)$ and $\hat{\mathcal{E}}_T(\theta,S)$ are the average empirical error and the multitask empirical error, for the meta-parameter $\theta$ and random features $S$ – see the section on the bound in the appendix. Bounding the terms (A) and (E) leads to (4.13) while bounding the terms (B) and (D) leads to (4.14). The term (C) is the optimization error and is negative if we can minimize the empirical risk objective. Finally the term (F) is bounded using [Tro19, Theorem 2.1] and auxiliary results presented in the appendix. □

There are several implications of the above theorem which we now comment upon. The first term in the r.h.s. of (4.13) contains the unnormalized Rademacher complexity $R_{n,M,T}$ of the set $\{(\langle\psi_\theta(s_j),x_i^t\rangle)_{i,j,t=1}^{n,M,T} : \theta \in \Theta\} \subseteq \mathbb{R}^{n\times M\times T}$. This is a measure of the capacity of the RKHS's we consider as part of using the kernel family $\mathcal{K}$ and quantifies the kernel families ability to fit random noise. While this quantity requires a case by case analysis it is often of order $\sqrt{T}$. Since in meta-learning the number of tasks is very large this term is negligible in many practical scenarios. For example, following the reasoning in [One+20] we obtain that $R_{n,M,T} = O(\sqrt{nMT})$. The number of random features should then be chosen so that the quantity (4.15) is smaller than (4.14). $G_n^*$ represents the size of best RKHS needed to explain the data averaged over the possible datasets sampled from the environment. In some sense it represents the degrees of freedom of the best model $\theta^*$ given the meta-distribution. A direct computation gives the condition

$$M = \Omega\left(\frac{n}{\lambda} + \frac{G_n^* \log n}{\lambda^3}\right).$$

Since $G_n^* \in [1, n]$, we conclude that the number of random features needed by the algorithm in order to be competitive with meta-learning without random feature approximation is *independent of the number of tasks* and only mildly dependent on $n$. For example, assuming $\lambda = 1/\sqrt{n}$ we obtain that $M = \Omega(n^{\frac{3}{2}} \log n)$ or $M = \Omega(n^{\frac{5}{2}} \log n)$ when $G_n^* = 1$ or $G_n^* = n$, respectively. The case that $G_n^* = O(n)$ requiring more random features corresponds to a low rank Gram matrix, meaning that the tasks are strongly related. This is however worth the effort since in this case the optimal risk $\mathcal{E}(\theta^*)$ we compare to will be very small, because the optimal low rank kernel makes learning very easy. Finally we note that $\lambda$ being in the denominator of all terms is an artifact due to comparing to the best KRR algorithm $\theta^*$ instead of the quantity $\mathcal{E}^* = \mathbb{E}_{\rho \sim \mu} \mathcal{R}_\rho(f_\rho)$ where $f_\rho = \mathbb{E}[y|\cdot]$ is the optimal predictor for the distribution $\rho$ [see Den+19, for a discussion].

While the above bound shows that IKML in the idealized setting of being able to find the MERM $\hat{\theta}$, which is not possible in practice and to account for this would incur an additional optimization error term in the bound, has a risk which tends towards that of the ideal minimizer $\theta^*$, there are still some things which it fails to explain and has some parts which are unrealistic. Firstly, it does not take into account more detailed conditions on $\mu$ which would allow for a more detailed understanding of learning in the meta-learning setting such as questions relating to negative transfer and how much IKML would improve over ITL using KRR. Secondly, we have not analyzed the approximation error which would occur if the optimal kernel would fall outside that of the pre-specified collection of kernels $\mathcal{K}$. Still, we believe the bound is encouraging as it shows that IKML learns and provides insight into how to choose the number of random features for learning to occur which is practically useful.

## 4.6 Experimental Results

We evaluate the performance of the proposed meta-learning strategy on both synthetic and real experiments against several baselines.

## 4.6.1 Synthetic Multivariate Regression

For IKML to be effective in realistic meta-learning regression scenarios it is important that it can approximate non-trivial functions defined on $\mathbb{R}^d$ where $d \gg 1$. To investigate this we create two synthetic high-dimensional meta-learning regression settings where each task is sampled from an RKHS $\mathcal{H}$ with a "complicated" kernel $K^o$. In particular, we choose $K^o$ to be the kernel given by Bochner's theorem and a pushforward, where we consider two settings, one of no misspecification using a 3-layers Multi-Layer Perceptron (MLP) with 32 hidden units per layer, ReLU activation functions and a 16-dimensional latent Gaussian distribution, and one of severe misspecification where we use a 5-layers MLP with 128 hidden units per layer, ReLU activation functions and a 32-dimensional latent Gaussian distribution. The network was initialized with weights given by the PyTorch [Pas+19] default initialization scaled by 100 and 200 respectively for the no misspecification and the severe misspecification setting. Since this kernel lacks an analytic form, we sample 10000 frequencies and use the random features kernel from (4.5) in its place. The tasks are generated from a distribution on $f \in \mathcal{H}$ and a marginal distribution on inputs fixed across all tasks. For each task we sample $n = n_{\text{tr}} + n_{\text{val}} = 50 + 50$ inputs $(x_i)_{i=1}^n$, a function $f$ and create the task $(x_i, f(x_i))_{i=1}^n$, for more details see Appendix B.4.3.

We compare the following meta-learning algorithms:

**IKML.** Alg. 4 parameterizing the pushforward $\psi_\theta$ for the measure $\tau_\theta$ with a three-layer MLP with hidden dimension set to 32 and the dimension of the latent space $\mathcal{Z} = \mathbb{R}^{16}$. The number of random features is set to $M = 10^4$ for the no misspecification settings and $M = 5000$ for the severe misspecification setting.

**Gaussian MKL meta-KRR (GMKL).** Multiple Kernel Learning (MKL) with KRR as inner algorithm. The meta-algorithm consists in learning the weights of a kernel $K = \sum_{j=1}^k \lambda_j K_j$ that is a convex combinations of Gaussian kernels $K_j(x, x') = \exp(-\frac{1}{2\sigma_j^2}\|x - x'\|^2)$ with lengthscale $\sigma_j$ taken from an logequidistant grid from $10^{-3}$ to $10^{3}$. The meta-learning algorithms learns the weights $\lambda$ parameterized in terms of the vector $z \in \mathbb{R}^k$ as $\lambda_j = \frac{\exp(z_j)}{\sum_{i=1}^k \exp(z_i)}$.

**MAML** [FAL17]**.** Optimizing through inner gradient descent with MLP to learn a good initalization in the outer loop. We use a three-layer MLP with 32 hidden units and ReLU activation functions.

**R2D2** [Ber+19]**.** Ridge regression as inner algorithm, learning a shared feature map in the outer loop. We use a three-layer MLP with 32 hidden units and ReLU activation functions.

**Oracle.** Running a separate instance of KRR on each task, with the same kernel $K^o$ used to generate the tasks, and finding $\lambda$ by cross validation on the test set.

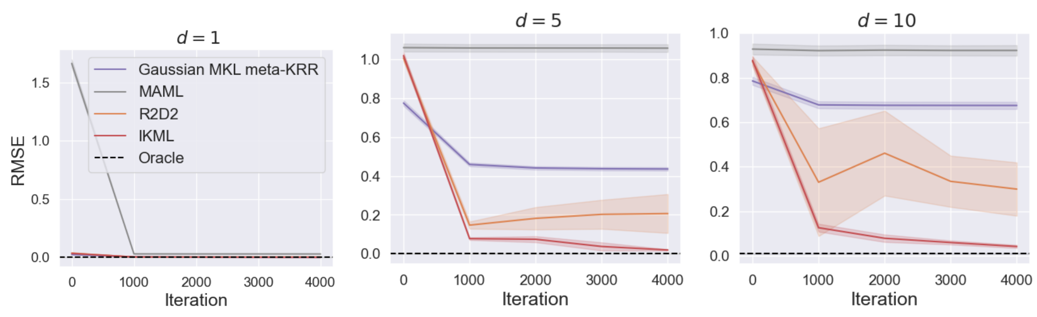


Figure 4.1: Learning curves of meta-test RMSE over three runs (mean $\pm$ 1 std) of Gaussian MKL meta-KRR, MAML, R2D2 and IKML together with the KRR Oracle on the synthetic no misspecification meta-learning problem introduced in Sec. 4.6.1 for $d = 1, 5, 10$. We generate $K^o$ once for each experiment and resample tasks for each run. Note that for low dimensions, MKL and R2D2 performs comparably to IKML. For $d = 20$ the algorithms fail to learn (left out). We stipulate that this is due to the number of train and validation points in the task not being enough to learn in this relatively higher dimensional setting.

As can be seen from Fig. 4.1 and 4.2 IKML performs best in all settings (correctly and misspecified setting) for all dimensions, highlighting that it manages to learn well even when the kernel is misspecified. R2D2 is the closest competitor except for low dimensions where Gaussian MKL manages to learn something in addition to IKML and R2D2. MAML struggles in all cases except for the 1-dimensional case. We ran experiments for $d = 20$ but none of

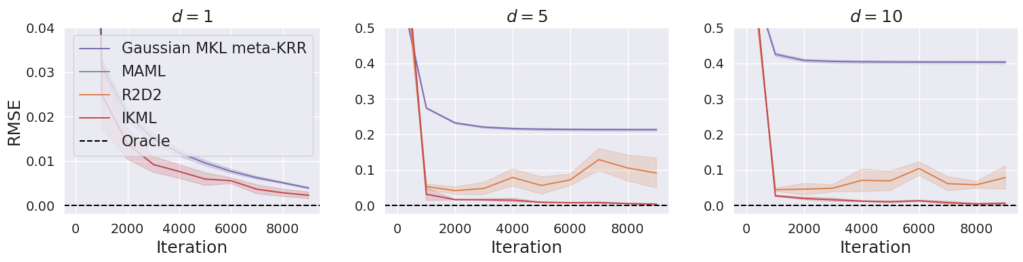


Figure 4.2: Learning curves of meta-test RMSE over three runs (mean $\pm$ 1 std) of Gaussian MKL meta-KRR, MAML, R2D2 and IKML together with the KRR Oracle on the synthetic meta-learning misspecification problem introduced in Sec. 4.6.1 for $d = 1, 5, 10$. We generate $K^o$ once for each experiment and resample tasks for each run. IKML performs better for all settings while R2D2 initially learns for $d = 5$ and 10. For $d = 20$ the algorithms fail to learn (left out). We stipulate that this is due to the number of train and validation points in the task not being enough to learn in this relatively higher dimensional setting.

the algorithms learn well, probably due to the size of the train and validation sets of a task being too small.

## 4.6.2 Real-World Data Experiments

We evaluate the proposed approach on two new real world meta-learning regression datasets adapted to the meta-learning setting from the UCI repository [DG17]. Apart from IKML and Gaussian MKL meta-KRR, we used the following algorithms in our experiments: *LS Biased Regularization [Den+19] (LSBR).* Running linear ridge regression with biased regularization $\lambda\|f - \theta\|^2$ in the inner algorithm, learning the bias $\theta$ in the outer loop.

**ANP [Kim+19].** Learns to map datasets to stochastic processes over functions using neural networks to do meta-learning. Predictor is the conditional mean of the stochastic process.

**Gaussian Oracle KRR (GO).** Gaussian KRR addressing each task as a separate learning problem but cross-validating the kernel bandwidth and regularization parameters $\sigma^2$ and $\lambda$ on the average validation error directly on the meta-test set.

We chose the baselines from landmark papers in the few-shot learning (MAML, R2D2, ANP, LSBR) and multiple-kernel learning (GMKL, GO) literature applicable to regression. We think these are natural baselines to compare against.

For both meta-learning datasets, we run the algorithms above in an online fashion where we use a meta-batch of 4 tasks per iteration sampled from the meta-train set. For IKML we fix the number of random features to 20000 which is on the order of $\Omega(n^{5/2}\log(n))$ if we would have pooled the train and validation set of 25 datapoints to one train set of size 50. Note however that further experiments show that in practice we can get away with as little as 2500 random features while mainting performance. Every 250 steps we sample 1000 tasks from the meta-validation set and evaluated the average meta-loss for each algorithm and save the model parameters. After training we sample 3000 tasks from the meta-test set. For the meta-test evaluation, for all algorithms, we use the meta-parameters with the lowest meta-validation error and get the test performance for all algorithms. We measure performance in terms of the root mean square error (RMSE). This procedure was run 5 times over different random seeds in order to get learning curves and results on the meta-test set. Below we describe the datasets and comment on the empirical evidence.

#### 4.6.2.1 Air Quality

The Beijing Air Quality dataset [Zha+17] is a time-series dataset measuring air-quality and meterological factors at 12 air-quality monitoring sites. The meterological data for each site is matched with the closest of available weather stations. The data was collected hourly and from the period March 1st, 2013 to February 28th, 2017. For further details we refer to Appendix B.3.1.

We generate a task of train and validation size $n_{\mathrm{tr}}, n_{\mathrm{val}}$ by randomly picking a station and picking a contiguous subsequence of size $n = n_{\mathrm{tr}} + n_{\mathrm{val}}$ at random from the split. We append the feature "$t$" which is the local order of data points and then randomly assign $n_{\mathrm{tr}}$ of the $n$ points to the train set and the rest to the validation set. This can be seen as a reconstruction problem: given data from sensor of which some have failed, we want to infer the output given an input at some points in time. We choose to use $n_{\mathrm{tr}} = n_{\mathrm{val}} = 25$.

After experimenting we use the following configuration of the algorithms; For Gaussian MKL meta-KRR we use 20 Gaussian kernels with lengthscale sampled geometrically from 1 to $10^{12}$ and learn the coefficients and regularisation parameter using the same parameterization as in the synthetic experiment with Adam and a meta-learning rate of 0.001. For LS Biased Regularization we learn the bias and regularisation parameter using Adam with meta-learning rate 0.01. We parameterized MAML with a 2-layer MLP with 64 hidden dimensions and with inner learning rate $10^{-7}$ and one adaptation step, learning the initialization using Adam with meta-learning rate of 0.001. We found that using a very small inner learning rate and few steps was important to get MAML to converge. For R2D2, IKML and ANP we cross-validated to find the best set of hyparparameters, see Appendix B.4.5 and Table B.1 in the appendix for more information. For Gaussian meta-KRR we learn the lengthscale and regularisation parameter using Adam with a meta-learning rate of 0.001. We benchmark a neural network IKML, called IKML-MLP in where we use a 4-layer MLP with 64 hidden units, 8 output features and 500 random features trained using Adam with learning rate of $3 \cdot 10^{-4}$, see Appendix B.4.2 for details.

From the Air Quality column of Table 4.1 we can see that IKML performs best with R2D2 and IKML-MLP close seconds. In general we can see that on this dataset the algorithms which learn a feature map linked with a closed form estimator performs best with the exception of GMKL.

#### 4.6.2.2 Gas Sensor

The Gas Sensor Modulation dataset [BJM18] is a collection of multivariate timeseries collected in a controlled environment using MOX sensors for CO detection sampled at 3.5 Hz. Each task corresponds to a subsampled timeseries from an experiment. As noted in [BJM18] the regression tasks are hard due to being heteroscedastic, non-normal and non-linear as a function of time but with tasks sharing a lot of structure, making it suitable as a meta-learning regression dataset. For further details we refer to Appendix B.3.2.

Table 4.1: Test RMSE on Beijing Air Quality and Gas Sensor. Best results in **bold**.

| Model | Air Quality RMSE | Gas Sensor RMSE |
|---|---|---|
| GMKL | 23.27 ± 0.16 | 9.61 ± 0.07 |
| LSBR | 21.68 ± 0.29 | 12.44 ± 0.14 |
| MAML | 34.96 ± 3.58 | 2.81 ± 0.12 |
| R2D2 | 20.23 ± 0.55 | **1.95 ± 0.06** |
| Gaussian meta-KRR | 25.08 ± 0.48 | 9.80 ± 0.09 |
| GO | 25.94 ± 0.91 | 12.78 ± 0.10 |
| IKML | **19.14 ± 0.93** | 2.80 ± 0.10 |
| IKML-MLP | 20.77 ± 0.57 | 2.06 ± 0.09 |
| ANP | 33.77 ± 0.70 | 2.12 ± 0.09 |

Table 4.2: Test RMSE / MAE / SMAPE on Beijing Air Quality and Gas Sensor datasets for R2D2, IKML and ANP.

| | Air Quality | | |
|---|---|---|---|
| Model | RMSE | MAE | SMAPE |
| R2D2 | 20.23 ± 0.55 | 11.67 ± 0.40 | 0.24 ± 0.01 |
| IKML | **19.14 ± 0.93** | **10.62 ± 0.19** | **0.22 ± 0.00** |
| ANP | 33.77 ± 0.70 | 21.08 ± 0.40 | 0.35 ± 0.01 |
| | Gas Sensor | | |
| Model | RMSE | MAE | SMAPE |
| R2D2 | **1.95 ± 0.06** | **0.94 ± 0.09** | 0.18 ± 0.05 |
| IKML | 2.80 ± 0.10 | 1.61 ± 0.26 | 0.24 ± 0.03 |
| ANP | 2.12 ± 0.09 | 1.06 ± 0.06 | **0.09 ± 0.01** |

We benchmark the algorithms for $n_{\text{tr}} = n_{\text{val}} = 20$. After experimenting we use the following configuration of the algorithms; For Gaussian MKL meta-KRR we use 20 Gaussian kernels with lengthscale chosen geometrically from 1 to $10^8$ and learn the coefficients and regularisation parameter using the same parameterization as in the synthetic experiment with Adam and a meta-learning rate of 0.001. For LS Biased Regularization we learn the bias and regularisation parameter using Adam with meta-learning rate 0.01.

We parameterized MAML with a 4-layer MLP with 64 hidden units with inner learning rate $10^{-4}$ and one adaptation step, learning the initialization using Adam with meta-learning rate of $10^{-4}$. For R2D2, IKML and ANP we cross-validated to find the best set of hyparparameters, see Appendix B.4.5 and Table B.1 in the appendix for more information. For Gaussian meta-KRR we learn the lengthscale and regularization parameter using Adam with a meta-learning rate of 0.001. IKML-MLP is as for Gas Sensor, but with 2 layers and 100 random features.

From the Gas Sensor column of Table 4.1 we can see that IKML and MAML gets the lowest meta-test error after R2D2 and IKML-MLP, in contrast with the Air Quality results where MAML performed poorly. This is probably due to Gas Sensor having less noise and thus making MAML easier to optimize (MAML is well-known to be hard to train [AES19]).

#### 4.6.2.3 Additional Metrics

RMSE is a well-established regression metric due to its grounding in statistical modelling, where maximum likelihood estimation of conditional output models with Gaussian noise can be shown to be equivalent to minimizing the (R)MSE of this model. However, it is not the only metric of interest and solely relying on it as a metric of success may be misleading due to RMSE being sensitive to outliers and not taking into account the magnitude of the true value of the output $y$ and the prediction $\hat{y}$. Given a collection of magnitudes of differences (we call such a value a deviance here) between the true outputs and the predictions of a model $h$, $(|y_i - h(x_i)|)_{i=1}^n$, for some dataset $D = (x_i, y_i)_{i=1}^n$, the RMSE is defined to be $\sqrt{\frac{1}{n}\sum_{i=1}^n |y_i - h(x_i)|^2}$. RMSE is sensitive to outliers which means that a few large deviances $|y - h(x)|$ will contribute most of the RMSE. In many cases this is not the behaviour we want from a metric hence the need for additional ones.

To test the abilities of IKML more holistically we evaluate it and two other strong baselines (R2D2 [Ber+19] and ANP [Kim+19]) on two additional metrics: mean absolute error (MAE) and symmetric mean absolute scaled error (SMAPE)[2] [CWJ21]. MAE is less sensitive to outliers since it does not square each deviance, being defined as $\frac{1}{n}\sum_{i=1}^n |y - h(x)|$. In the case when

[2]Note that we present this as a ratio instead of as a percentage.

the practical bad outcome of a large and very large deviance are similar this is a better metric to use than the RMSE. SMAPE is a metric which takes into account not only the deviance but also the magnitude of the prediction and outputs, being defined as $\frac{1}{2n}\sum_{i=1}^{n}\frac{|y_i-h(x_i)|}{|y_i|+|h(x_i)|}$. The impact of this is that SMAPE is not only a function of the deviance but also the raw values of the output and the prediction, unlike RMSE and MAE which are defined solely in terms of deviances. When the output space has a meaningful zero point this means that SMAPE takes into account how well the model predicts outputs of small magnitude. This is not true for RMSE and MAE; consider a model $h(x) = 0$, the deviance when using this model at an output $y$ is simply $|y|$ which can be arbitrarily small since it only depends on the magnitude of $y$, while for SMAPE each contributing term in the sum in this case is 1.

For the algorithms R2D2, IKML and ANP, with the same setting and training strategies as outlined for Air Quality and Gas Sensor datasets using RMSE, MAE and SMAPE as metrics we tabulate the results in Table 4.2. From Table 4.2 we see that IKML performs the best on all metrics on the Air Quality dataset with ANP performing poorly and the metrics of the algorithms correlating well with each other. For Gas Sensor R2D2 performs best except for SMAPE where ANP performs much better. The reason for this is due to Gas Sensor having many values close to zero which means that a low RMSE or MAE does not necessarily translate into a lower SMAPE metric, which probably means that ANP predicts outputs with small output magnitude better than R2D2 and IKML, but does not predict better than R2D2 and IKML on outputs of typical magnitude.

#### 4.6.2.4 Impact of Number of Random Features on RMSE and Time

In this section we investigate how the number of random features during train time impacts wall-time taken to train and number of random features during train and test time impacts performance for IKML. Using the Beijing Air Quality and Gas Sensor datasets we run IKML using the same hyperparameters as in Sec. 4.6.2 except for the number of random features which we let be in the set of $\{500, 1000, 5000, 10000, 20000\}$ during training and testing. We first choose a number of random features for training. After we finish training, we use the parameters with the lowest validation error during training (using the same number of random features we chose for training) and test the model with these parameters, varying the number of random features from 500 to 20000 in the same set.

Table 4.3: Test RMSE on Gas Sensor dataset for IKML as a function of number of random features during train and test time.

| Random Features (test)<br>Random Features (train) | 500 | 1000 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| 500 | 3.03 | 3.02 | 3.02 | 3.02 | 3.02 |
| 1000 | 3.09 | 3.09 | 3.09 | 3.09 | 3.09 |
| 5000 | 3.08 | 3.08 | 3.07 | 3.07 | 3.07 |
| 10000 | 3.55 | 3.52 | 3.51 | 3.51 | 3.52 |
| 20000 | 2.97 | 2.96 | 2.96 | 2.95 | 2.95 |

Table 4.4: Timings (seconds) of training for the Gas Sensor dataset for IKML as a function of number of random features during training.

| Random Features (train) | 500 | 1000 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| | 739 | 760 | 781 | 779 | 836 |

In terms of performance, Tables 4.3 and 4.5 show that over the range of random features defined in the set above, performance is similar with a slight decrease in test RMSE as the number of random features increase. An exception is row 4 in Table 4.3, with value 10000 which is slightly higher than the other rows of the table, which is due to an outlier run during training. For Table 4.5 we see a decrease as we go from 1000 to 5000 random features during training after which the performance is essentially the same for 5000 to 20000 random features. In terms of timings, as can be seen Tables 4.4 and 4.6, the wall-time increases very slightly with the number of random features used during train time, but the increase is extremely small.

Table 4.5: Test RMSE on Beijing Air Quality dataset for IKML as a function of random features number of during train and test time.

| Random Features (test)<br>Random Features (train) | 500 | 1000 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| 500 | 22.12 | 22.06 | 22.01 | 22.01 | 22.02 |
| 1000 | 22.01 | 21.98 | 21.89 | 21.90 | 21.91 |
| 5000 | 19.63 | 19.65 | 19.55 | 19.55 | 19.54 |
| 10000 | 19.32 | 19.27 | 19.21 | 19.16 | 19.19 |
| 20000 | 19.42 | 19.30 | 19.27 | 19.26 | 19.24 |

Table 4.6: Timings (seconds) of training for the Beijing Air Quality dataset for IKML as a function of random features during training.

| Random Features (train) | 500 | 1000 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| | 482 | 497 | 487 | 489 | 515 |

Chapter 5

# Conclusion and Future Work

In this section we summarize the contributions of this thesis to the field of meta-learning, few-shot learning and machine learning, and add concluding remarks and future work related to chapters 3 and 4.

We end with a discussion of future work in the direction of currently important topics in machine learning; conditional meta-learning, incorporating further invariances into meta-learning and time-series. We outline how MeLa and IKML could be adapted to work in these settings and potential advantages of these methods in these contexts. Meta-learning as a field is still developing, but it is my belief that the tools we have developed in the, maybe restrictive, setting of few-shot supervised learning can be adopted to many other settings of great importance for academia and industry.

## 5.1 MeLa

### 5.1.1 Summary of Contributions

We answer several important questions in few-shot classification and introduce an effective clustering algorithm to recover the base classes in this setting. We provide both theoretical and experimental justification for why pre-training is an effective and ubiquitous strategy for few-shot image classification. This justification is done by explicitly linking pre-training with conditional meta-learning and showing that pre-training minimizes an upper bound on the meta-learning risk. Inspired by this we introduce a data-augmentation technique for few-shot learning which introduces virtual classes, en-largening the meta-dataset, and a clustering algorithm, MeLa, that can recover base classes from a meta-dataset of tasks with only local classes. MeLa allows us to use pre-training even when base classes are not available to us, or they are ill-defined, and achieves SOTA performance against both settings where base classes are assumed and not assumed to be available. Finally, we introduce a more realistic few-shot classification setting where instances are not allowed to be shared between tasks and the number of tasks is fixed. We empirically show that our algorithm performs robustly in both standard and our introduced few-shot setting and clearly outperforms SOTA models in the latter.

### 5.1.2 Conclusion and Future Work

In this work, we focused on the role played by pre-training in meta-learning applications, with particular attention to few-shot learning problems. Our analysis was motivated by the recent popularity of pre-training as a key stage in most state-of-the-art FSL pipelines. We first investigated the benefits of pre-training from a theoretical perspective. We showed that in some settings this strategy enjoys significantly better sample complexity than pure meta-learning approaches, hence offering a justification for its empirical performance and wide adoption in practice.

We then proceeded to observe that pre-training requires access to global labels of the classes underlying the FSL problem. This might not always be possible, due to phenomena like heterogeneous labeling (i.e. multiple labelers having different labeling strategies) or contextual restrictions like privacy constraints. We proposed Meta-Label Learning as a strategy to address this concern. We compared MeLa with state-of-the-art methods on a number of tasks including well-established standard benchmarks as well as new datasets we designed to capture the above limitations on task labels. We observed that MeLa is always comparable or better than previous approaches and very robust to lack of global labels or the presence of conflicting labels.

More broadly, our work provides a solid foundation for understanding existing FSL methods, in particular the vital contribution of pre-training towards generalization performance. We also demonstrated that pre-training scales well with the size of datasets and data diversity, which in turn leads to more robust few-shot models. Future research may focus on further theoretical understanding of pre-training and better pre-training processes.

Some open questions remaining are the following; we would want to extend the clustering algorithm to properly incorporate multi-domain meta-datasets, where we have many base datasets. In this case we would need to model this structure and go beyond the binomial construction used for the pruning. We would like to apply MeLa to bigger and more realistic datasets such as meta-dataset [Tri+20] which would require require some engineering due to the large-scale nature of this dataset.

Finally we would like to continue with a more detailed analysis of pre-training and meta-learning to properly work out how they are related and not just look at rates, in addition to understanding the conditions necessary on the meta-distribution to derive bounds which give a more detailed insight into when FSL may or may not suffer compared to standard meta-learning or independent task learning, being able to take into account positive and negative transfer. A downside of the current bound is that it does not take into account the meta-distribution in a very comprehensive sense, similarly to the bound for IKML. In this way it does not add any knowledge in terms of under what conditions on the meta-distribution Me

## 5.2 IKML

### 5.2.1 Summary of Contributions

We introduce IKML, a framework for meta-learning an RKHS (or equivalently; an infinite-dimensional feature map) from a family of kernels having certain inductive biases, where we focus on the case of translation-invariant kernels, which performs well for meta-regression when the inner algorithm is KRR. As the kernel is encoded implicitly through a neural-network pushforward of a simple distribution, we use random features to get an approximation which is approximately equal to the true kernel with high probability. We derive a bound on the excess risk, assuming that we can find the true empirical risk minimizer, showing how problem-specific quantities impact the number of random features needed to generalize at test time. We show that the number of random features required for good performance is small and can be used to trade off performance against speed at test-time. Additionally, we introduce a heuristic kernel family given by a neural network random feature kernel which extends R2D2. For both kernels we show competitive performance together with interpretability. Finally, we introduce three novel meta-learning regression benchmark datasets and make them available to the community.

### 5.2.2 Conclusion and Future Work

We introduced a framework for implicit kernel meta-learning, IKML, in context of translation-invariant and deep random kernel families. Our approach focuses on problems where data does not present a clear input structure (in contrast e.g. to image classification settings) and using a plug-in translation invariant kernel might be a safer strategy. Our approach leverages the characterization of random feature kernels, in particular the translation invariant kernels granted by Bochner's theorem and ideas from the random features literature to learn it in practice. We derive a novel bound on the excess transfer risk shedding light on how to choose the number of random features. To validate our method we introduced two real-world meta-learning regression datasets.

IKML achieve best or close-to-best performance on all of the datasets against state-of-the-art methods designed for few-shot image classification. We hypothesize that when the data does not have enough structure (e.g. in most regression settings), learning a deep representation – as done by state-of-the-art methods such as MAML or R2D2 – may be less effective. We leave further investigation of this question to future work.

Some problems remain, mainly with the derived bound. The bound would be more complete if we took into account the optimization error which is the error due to not being able to finding the minimum $\hat{\theta}$. Additionally, the bound does not take into account a more fine-grained analysis of the types of meta-distributions we are likely to encounter in practice which could lead to faster rates and more insight into when using IKML is better than doing standard KRR with a fixed kernel, so called negative transfer.

We close by mentioning a couple of relevant directions for future research:

**Conditional Meta-Learning:** Is it possible to extend the framework to conditional meta-learning? One way would be to use KTA similar to [SD16] and adjusting the initial starting kernel similar to MAML.

**Theoretical Guarantees:** Can we show that IKML converges to a stationary point for benign settings? This would require understanding the bias-variance decomposition of the gradient.

**Alternative Kernel Classes:** Can we extend IKML to other kernel families? An example is dot-product kernels [KK12].

**Vector-Valued Kernels and Multi-Output Regression:** We have for simplicity focused on the case where $\mathcal{Y} \subseteq \mathbb{R}$ but in principle it should be possible to extend IKML to multi-output or vector-valued regression [ÁRL12; MP05] where $\mathcal{Y} \subseteq \mathbb{R}^l$ and allows for leveraging the correlation between the different output dimensions and correlations between inputs and outputs, encoded as cross-covariance matrices or operators. Using random fourier features for vector-valued kernels [BHB16] and modeling the regularizer using one of the methods in [ÁRL12] would be a fruitful future direction to pursuit.

### 5.2.3 Artifacts

Code and data necessary to reproduce the results of chapters 3 and 4 (including results in the appendix Appendices A and B) can be found at

**MeLa**

`https://github.com/IsakFalk/mela`,

**IKML**

`https://github.com/IsakFalk/IKML`.

## 5.3 Concluding Remarks and Future Work

Machine learning is currently grappling with growing pains as it is developing from a field which has been mainly academic to a tool being deployed in the real world. This development, while very exciting, has also shown that problems such as non-independence of samples in the form of dataset shift during deployment and data-distributions shifting over time (for example, time-series data) is very real when going from benchmarks to deployment. If machine learning is to be used in industry without potentially catastrophic failures and to additionally become trustworthy so that it can be used in deployment and reach wide-scale use, we need to solve the above problems. To use the definition of a *technology* in [Lem06, p. 5][1]:

> Machine learning is not yet a technology; it requires expert knowledge in order to train and deploy, and furthermore it has a propensity to be brittle and deteriorate under dataset shift.

These types of questions are naturally answered in the setting of meta-learning as it *assumes that the data is heterogeneous* and further in few-shot learning *the meta-train, validation and test sets are assumed disjoint* which by itself is a type of dataset shift. By extending meta-learning in the direction of meta-learning algorithms which are robust to dataset shift, takes into account natural data invariances and work on time-series data, either on the task or dataset level [see e.g. Den+19, for steps in this direction], we move closer to machine learning that is deployable in the real world and is reliable, that is, can be considered a technology. Below we propose some potential directions for future work in the vein of the above.

### 5.3.1 Conditional Meta-Learning

Recently it has been shown that conditional meta-learning has beneficial properties when compared to standard meta-learning, as outlined in Sec. 2.4.1.

[1]The authors propose linear programming as an example of a mature technology as it is reliable and can be used by practitioners without advanced knowledge of the underlying theory and tricks.

While we have investigated how MeLa relates to conditional meta-learning theoretically, another direction for IKML would be to investigate how to turn this algorithm conditional. While there may be many ways to do this, one principled way would be to use kernel target alignment similar to [SD16] and adjusting the initial starting kernel similar to MAML [FAL17]. Kernel target alignment comes with theoretical backing [CMR12] and it may be possible to derive better generalization bounds when compared to standard, unconditional IKML, similarly to what is done in [DPC20].

### 5.3.2 Incorporating Invariances

Assumptions on the data are very common in machine learning, leading to notions such as sparsity and low-rank models (e.g. LASSO [Tib96]). Recently a different type of data assumption have become popular based on invariances, mainly due to the efforts of the geometric deep learning community and machine learning applied to physical problems [Bro+21; Bro+17]. IKML is linked to this development through the form of the kernel family. Many kernel families with certain properties such as translation-invariance and rotation invariance can be shown to have a dual formulation using some integral transform. Understanding this link better would allow to specify conditions when IKML generalizes by incorporating invariances, preferably quantified within an upper bound on the meta-risk. Work in this direction has already been done in the KRR case [EZ21; Ele21] and it would be interesting to build upon this work and extend it to the meta-learning setting of IKML when the data is assumed to posses these invariant properties.

### 5.3.3 Meta-learning for Time-Series

Time-series are ubiquitous in many contexts where machine learning is applied, ranging from economics, social sciences to meterology. Many phenomenon in both the natural and social sciences can be modelled as time-series, including climate indicators, GDP and financial instruments. The time-series setting allows us to weaken the sometimes restrictive assumption of iid samples and replace this with some notion of correlation that vary with some time variable $t$. While this makes the setting more general, it also makes it much harder since generalizing from a train to a test set is not as simple as applying standard concentration inequality tools to get a notion of generalization performance. In the worst case, the generalization performance depend on the timestamp that you want to predict on, leading to a family of risks indexed by future timestamps different from the ones observed in the train set.

Many real-world problems can be cast into the multi-task or meta-learning setting where each task is a time-series prediction problem. This is clear from the pressing issue of how to transition from an energy system based on fossil fuels to one based on renewables, as renewable energy requires accurate forecasting of supply and demand [Swe+20; Wan+19]. The energy grid can readily be cast as an instance of multi-task time-series regression. Recently there has been some work on meta-learning algorithms for time series, most notably [Ore+20] which achieved good performance on the well-known M4 and M3 time-series benchmarks [MSA18; MH00].

In the context of the methods proposed in this thesis, it is still unclear how to learn a feature map efficiently for meta-time-series regression. Some aspects of time-series modelling which makes this hard is the fact that time series are potentially unbounded and may not be stationary (as the underlying distribution changes with time) and the fact that many of the losses used are highly non-convex as they depend not only on the outputs and predictions but also on their magnitudes in non-trivial ways.

For MeLa it would be interesting to see if can we extend the assumption on the meta-distribution in few-shot classification of being sampled from some base dataset to the time-series setting. Assuming that we can recast this assumption successfully, a further question would be if we can extend the theory linking pre-training to meta-learning to the case of meta-time-series regression. The works of [Ler+23; Ler+22] could be used as a basis for this. For IKML, the translation-invariance of the kernel is related to the Fourier transform which features extensively in ARMA theory of time-series modelling [BDF91]. Here it would be interesting to see if we can apply IKML to time-series forecasting and relate the kernel learned to ARMA models, with theoretical guarantees similar to the ones given in this thesis.

# Appendix A

# MeLa

We first restate the notations and assumptions necessary to prove Thm. 1. From Assumption A, we define the joint distribution $\pi_\mu(x, y)$ as the probability of observing an input-output pair $(x, y)$ when first sampling a task $\rho$ from $\mu$ and then sampling $(x, y)$ according to the "query" distribution $\pi_\rho$ in (3.6). In other words, $\pi_\mu(x, y) = \pi(x|y)\text{Unif}_\rho(y)\mu(\rho)$ is the marginal distribution of $(x, y)$ with respect to $\rho$.

We observe that for any $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$,

$$\mathbb{E}_{(x,y)\sim\pi_\mu} g(x, y) = \mathbb{E}_{\rho\sim\mu}\mathbb{E}_{(x,y)\sim\pi_\rho} g(x, y). \tag{A.1}$$

Given Assumption A, $D^{\text{tr}}$ and $D^{\text{val}}$ are sampled independently by the task $\rho$. In particular the marginal $\rho_{D^{\text{val}}}$ of $\rho$ with respect to $D^{\text{tr}}$ corresponds to $\pi_\rho$. Similarly, we denote $\rho_{D^{\text{tr}}}$ the distribution over support sets obtained by marginalizing out the query set. We report one remark following the assumption above.

**Remark A.0.1.** *For any task $\rho$ and any algorithm $D \mapsto f_D$ returning func-*

*tions $f_{D^{\text{tr}}} : \mathcal{X} \to \mathbb{R}^k$, we have*

$$\begin{aligned}
\mathbb{E}_{(D^{\text{tr}}, D^{\text{val}}) \sim \rho} \hat{\mathcal{R}}(f_{D^{\text{tr}}}, D^{\text{val}}) &= \mathbb{E}_{D^{\text{tr}} \sim \rho_{D^{\text{tr}}}} \mathbb{E}_{D^{\text{val}} \sim \pi_\rho^m} \hat{\mathcal{R}}(f_{D^{\text{tr}}}, D^{\text{val}}) && \text{(A.2)} \\
&= \mathbb{E}_{D^{\text{tr}} \sim \rho_{D^{\text{tr}}}} \mathbb{E}_{D^{\text{val}} \sim \pi_\rho^m} \frac{1}{m} \sum_{(x,y) \in D^{\text{val}}} \ell(f_{D^{\text{tr}}}(x), y) && \text{(A.3)} \\
&= \mathbb{E}_{D^{\text{tr}} \sim \rho_{D^{\text{tr}}}} \mathbb{E}_{(x,y) \sim \pi_\rho} \ell(f_{D^{\text{tr}}}(x), y) && \text{(A.4)} \\
&= \mathbb{E}_{D^{\text{tr}} \sim \rho_{D^{\text{tr}}}} \mathcal{R}(f_{D^{\text{tr}}}, \pi_\rho), && \text{(A.5)}
\end{aligned}$$

*where for any $f : \mathcal{X} \to \mathbb{R}^k$ we have denoted by*

$$\mathcal{R}(f, \pi_\rho) = \mathbb{E}_{(x,y) \sim \pi_\rho} \ell(f_{D^{\text{tr}}}(x), y), \tag{A.6}$$

*the expected risk of a function $f : \mathcal{X} \to \mathbb{R}^k$ with respect to the loss $\ell$ and the distribution $\pi_\rho$.*

Using the above remark, we shows how the GLS risk for the meta-distribution $\mu$ is related to multi-class classification risk for the corresponding multi-class distribution $\pi_\mu$.

**Theorem 1.** *Under Assumption A, let $\pi_\mu(x, y)$ be marginal distribution of observing $(x, y)$ in the meta-training set. Then, for any (global) classifier $W$,*

$$\mathcal{E}_{\text{GLS}}(W, \theta) \leq \mathcal{R}(W\phi_\theta, \pi_\mu). \tag{3.8}$$

*Moreover, if the global classes are separable,*

$$\min_{W,\theta} \mathcal{E}_{\text{GLS}}(W, \theta) = \min_{W,\theta} \mathcal{R}(W\phi_\theta, \pi_\mu). \tag{3.9}$$

*Proof.* We start by studying the GLS risk for a pair $(W, \theta)$ of meta-parameters.

By expanding the GLS objective explicitly with Remark A.0.1, we have

$$\begin{aligned}\mathcal{E}_{\text{GLS}}(W,\theta,\mu) &= \mathbb{E}_{\rho\sim\mu}\mathcal{R}_{\text{ce}}(\text{GLS}(W,\theta,\rho),\pi_\rho) &\text{(A.7)}\\ &= \mathbb{E}_{\rho\sim\mu}\mathbb{E}_{(x,y)\sim\pi_\rho}\ell_{\text{ce}}(\text{GLS}(W,\theta,\rho)(x),y) &\text{(A.8)}\\ &= \mathbb{E}_{\rho\sim\mu}\mathbb{E}_{(x,y)\sim\pi_\rho}\ell_{\text{ce}}(W[\rho_Y]\phi_\theta(x),y). &\text{(A.9)}\end{aligned}$$

Since $\rho_Y$ is a subset of $\{1,\dots,C\}$, by definition of cross-entropy we have

$$\begin{aligned}\ell_{\text{ce}}(W[\rho_Y]\phi_\theta(x),y) &= -\log\frac{\exp(W[y]\phi_\theta(x))}{\sum_{y'\in\rho_Y}\exp(W[y']\phi_\theta(x))} &\text{(A.10)}\\ &\le -\log\frac{\exp(W[y]\phi_\theta(x))}{\sum_{y'\in\{1,\dots,C\}}\exp(W[y']\phi_\theta(x))} &\text{(A.11)}\\ &= \ell_{\text{ce}}(W\phi_\theta(x),y). &\text{(A.12)}\end{aligned}$$

We can now apply (A.1) where we take $g(x,y)=\ell_{\text{ce}}(W\phi_\theta(x),y)$. Then,

$$\mathbb{E}_{\rho\sim\mu}\mathbb{E}_{(x,y)\sim\pi_\rho}\ell_{\text{ce}}(W\phi_\theta(x),y) = \mathbb{E}_{(x,y)\sim\pi_\mu}\ell_{\text{ce}}(W\phi_\theta(x),y) = \mathcal{R}_{\text{ce}}(W,\theta,\pi_\mu), \tag{A.13}$$

which concludes the proof for (3.8).

Now, if the global classes are linearly separable, we have

$$\min_{W,\theta}\mathcal{R}(W\phi_\theta,\pi_\mu)=0. \tag{A.14}$$

Since $\ell_{ce}(\cdot)$ is non-negative, combining (A.14) and (3.8) yields

$$0\le\min_{W,\theta}\mathcal{E}_{\text{GLS}}(W,\theta)\le\min_{W,\theta}\mathcal{R}(W\phi_\theta,\pi_\mu)=0. \tag{A.15}$$

We thus conclude that

$$\min_{W,\theta}\mathcal{E}_{\text{GLS}}(W,\theta)=\min_{W,\theta}\mathcal{R}(W\phi_\theta,\pi_\mu). \tag{A.16}$$

□

## A.1 Comparison between the Lipschitz Constants of $\mathcal{E}_{\mathrm{GLS}}$ and $\mathcal{R}$

In this section, we compare the Lipschitz constants associated to the meta-GLS risk and the pre-training objective function discussed in Sec. 3.3.6, showing that $L_{\mathrm{pre}}$ is comparable or smaller than $L_{\mathrm{GLS}}$.

We recall that the Lipschitz constant of an objective functional of the form $\mathbb{E}_{\xi\sim\rho}\ell(\omega,\xi)$ is $\sup_\xi L_\xi$, where $L_\xi$ is the Lipschitz constant of $\ell(\cdot,\xi)$ for any $\xi$ in the support of the probability $\rho$. Additionally, we recall that for a smooth function $\ell(\cdot,\xi)$ defined on a set $\Omega$, the Lipschitz constant can be characterized as

$$\sup_{\substack{\omega\in\Omega\\ \xi\in\mathrm{supp}\rho}} \|\nabla_\omega \ell(\omega,\xi)\|, \tag{A.17}$$

where $\mathrm{supp}\rho$ denotes the support of the probability distribution $\rho$.

In the following we will assume $\Omega = B^{2,1}_\lambda(0)\times\Theta$, where $B_\lambda(0)$ is a ball of radius $\lambda$ with respect to the $2,1$–norm centred at 0, namely that $\|W\|_{2,1}\leq\lambda$. The $2,1$–norm is defined as $\|W\|_{2,1} = \sum_{z\leq C}\|W_z\|_2$ and helps to simplify the following analysis. $\Theta$ is the space of parameters for $\phi_\theta$ and we assume the representation model to be normalized, namely $\|\phi_\theta(x)\| = 1$ for all $\theta\in\Theta$ and $x\in\mathcal{X}$. The normalized representational model could easily generalized to any bounded representation, namely $\sup_{\theta,x}\|\phi_\theta(x)\| < +\infty$.

We now compare the Lipschitz constants of $\mathcal{E}_{\mathrm{GLS}}$, $L_{\mathrm{GLS}}$ and the pre-training risk from $\mathcal{R}$, $L_{\mathrm{pre}}$. The key difference between the two objective functions is that meta-GLS applies the cross-entropy loss over subsets of $\{1,\dots,C\}$ classes at each time (the classes appearing in a given task), while global multi-class classification applies the cross-entropy to the entire set of $C$ classes. Therefore, to highlight the dependency on the number of classes in the following we denote $\ell^k_{\mathrm{ce}}$ the cross entropy evaluated among $k$ classes for $k$ an integer. Then, under the notation introduced in this section and in Sec. 3.3.6, we need to compare

$$L_{\text{GLS}} = \sup_{\substack{(W,\theta)\in\Omega \\ (D^{\text{tr}},D^{\text{val}})\in \text{supp}\tilde{\mu}}} \left\| \nabla_{W,\theta}\ \ell(W[D_Y^{\text{tr}}]g_\theta(\cdot), D^{\text{val}}) \right\| = \sup_{\substack{(W,\theta)\in\Omega \\ (D^{\text{tr}},D^{\text{val}})\in \text{supp}\tilde{\mu} \\ (x,y)\in D^{\text{val}}}} \left\| \nabla_{W,\theta}\ \ell_{\text{ce}}^k(W[D_Y^{\text{tr}}]g_\theta(x), y) \right\|,$$

with the Lipschitz constant of the global multi-class classifier loss, corresponding to

$$L_{\text{pre}} = \sup_{\substack{(W,\theta)\in\Omega \\ (x,y)\in \text{supp}\pi_\mu}} \left\| \nabla_{W,\theta}\ \ell_{\text{ce}}^C(W g_\theta(x), y) \right\|,$$

where $\tilde{\mu}$ denotes the probability of sampling a pair $(D^{\text{tr}}, D^{\text{val}})$ by first sampling $\rho \sim \mu$ and then $(D^{\text{tr}}, D^{\text{val}}) \sim \rho$.

We first observe that if $\text{supp}\pi_\mu = \text{supp}\tilde{\mu}$ then $L_{\text{GLS}} = L_{\text{pre}}$. This happens *if the few-shot learning distribution can sample datasets containing at least one example per class.* We note however that the two quantities are in general very close to each other when the FSL datasets contain at most $k < C$ classes each, as shown in the result below.

**Theorem 3.** *With the notation and assumptions introduced above*

$$L_{\text{pre}} \leq L_{GLS} + O(e^{-\lambda})$$

*Proof.* We proceed by studying the norm of the gradients of $\ell_{\text{ce}}^k$ with respect to $W$ and $\theta$ separately.

**Gradient with respect to** $W$ Given a pair of $(x, y)$ and parameters $\theta$ and $W \in \mathbb{R}^{k\times m}$, the derivative with respect to $W_y$ the $y$-th column of $W$ is

$$\nabla_{W_y}\ \ell_{\text{ce}}^k(W\phi_\theta(x), y) = \left( \frac{e^{W_y^\top \phi_\theta(x)}}{\sum_{z\leq k} e^{W_z^\top \phi_\theta(x)}} - 1 \right) \phi_\theta(x).$$

The gradient with respect to a column $W_z$ for $z \neq y$ is

$$\nabla_{W_z} \ \ell^k_{\text{ce}}(W\phi_\theta(x), y) = \frac{e^{W_z^\top \phi_\theta(x)}}{\sum_{z \le k} e^{W_z^\top \phi_\theta(x)}} \ \phi_\theta(x).$$

We conclude that the norm of the gradient with respect to the full $W$ is

$$\begin{aligned} \left\| \nabla_W \ \ell^k_{\text{ce}}(W\phi_\theta(x), y) \right\| &= \sqrt{\left(1 - \frac{e^{W_y^\top \phi_\theta(x)}}{\sum_{z \le k} e^{W_z^\top \phi_\theta(x)}}\right)^2 + \sum_{z \neq y} \left(\frac{e^{W_z^\top \phi_\theta(x)}}{\sum_{z \le k} e^{W_z^\top \phi_\theta(x)}}\right)^2} \ \|g(x)\| \\ &= \frac{1}{\sum_z e^{W_z \phi_\theta(x)}} \sqrt{\left(\sum_{z \neq y} e^{W_z \phi_\theta(x)}\right)^2 + \sum_{z \neq y} e^{2W_z \phi_\theta(x)}}, \end{aligned}$$

where we have used the fact that representations are normalized $\|\phi_\theta(x)\| = 1$. We note that for any $W$, the norm above is upper bounded by $\sqrt{2}$. This follows by observing that

$$1 - \frac{e^{W_y^\top \phi_\theta(x)}}{\sum_{z \le k} e^{W_z^\top \phi_\theta(x)}} \le 1$$

and the term

$$\sum_{z \neq y} \left(\frac{e^{W_z^\top \phi_\theta(x)}}{\sum_{z \le k} e^{W_z^\top \phi_\theta(x)}}\right)^2 \le \sum_{z \le k} \left(\frac{e^{W_z^\top \phi_\theta(x)}}{\sum_{z \le k} e^{W_z^\top \phi_\theta(x)}}\right)^2 \le 1$$

corresponds to the (squared) norm of a vector in the simplex (minus the component associated to $W_y$). Hence it's $\ell_2$ norm is no larger than 1. Hence

$$\sup_{(W,\theta) \in \Omega} \left\| \nabla_{W_z} \ \ell^k_{\text{ce}}(W\phi_\theta(x), y) \right\| \le \sqrt{2}$$

Now, we provide a lower bound to the gradient norm. Let $\bar{z} \neq y$ and evaluate the gradient at a $W$ such that $W_z = 0$ for any $z \neq \bar{z}$ ($y$ included) and

$W_{\bar{z}} = \lambda\phi_\theta(x)$ for $\lambda > 0$, we have

$$\left\|\nabla_W \ \ell_{\text{ce}}^k(W\phi_\theta(x), y)\right\| = \frac{1}{e^\lambda + k - 1}\sqrt{(e^\lambda + k - 2)^2 + e^{2\lambda} + k - 2}$$
$$= \frac{1}{1 + \frac{k-1}{e^\lambda}}\sqrt{\left(1 + \frac{k-2}{e^{2\lambda}}\right)^2 + 1 + \frac{k-2}{e^\lambda}}$$

We note that as $\lambda \to +\infty$, the above equation converges to $\sqrt{2}$ as fast as $O(e^{-\lambda})$. In particular, this implies that for any $\theta$

$$\left|\sup_{(W,\theta)\in\Omega} \left\|\nabla_W \ \ell_{\text{ce}}^k(W\phi_\theta(x), y)\right\| - \sqrt{2}\right| \leq O(e^{-\lambda})$$

The gradient norm converges to $\sqrt{2}$ exponentially fast with respect to the upper bound on norm of $W$. In particular we have that for any number of classes $k$,

$$\sup_{(W,\theta)\in\Omega} \left\|\nabla_{W_z} \ \ell_{\text{ce}}^k(W\phi_\theta(x), y)\right\| \geq \sqrt{2} - O(e^{-\lambda}).$$

Since $\sqrt{2}$ is an upper bound, we have that for any $k \leq C$ and any $\theta \in \Theta$

$$\sup_{(W,\theta)\in\Omega} \left\|\nabla_{W_z} \ \ell_{\text{ce}}^C(W\phi_\theta(x), y)\right\| - \sup_{(W,\theta)\in\Omega} \left\|\nabla_{W_z} \ \ell_{\text{ce}}^k(W\phi_\theta(x), y)\right\|$$
$$\leq \sqrt{2} - (\sqrt{2} - O(e^{-\lambda})) = O(e^{-\lambda}),$$

which implies that the component of the norm of the gradient of $\ell_{\text{ce}}^C$ is at most larger than the same component but for $\ell_{\text{ce}}^k$ of a quantity that decreases exponentially fast with respect to $\lambda$.

**Gradient with respect to $\theta$** We now consider the gradient with respect to $\theta$. Given an input-output pair $(x, y)$ and a linear classifier $W$, we have

$$\nabla_\theta \ell_{\text{ce}}^k(W\phi_\theta(x), y) = \nabla_\theta \phi_\theta(x)^\top \left[\sum_{z \neq y} \frac{e^{W_z^\top \phi_\theta(x)}}{\sum_{z \leq k} e^{W_z^\top \phi_\theta(x)}}(W_y - W_z)\right].$$

To estimate the maximum of the norm of the gradient above with respect to the parameters $W$, we will show below that the maximum is achieved by choosing a class $\bar{z} \neq y$ such that $W_z = 0$ for all $z \neq \bar{z}$ and $W_{\bar{z}}$ is the only non-zero column.

Let $W_y = \lambda_y \, v$ for some vector $v \in \mathbb{R}^m$ of norm one $\|v\| = 1$ and $\lambda_y > 0$. We start by observing that the maximum length vector for the gradient above is obtained by summing vectors that are all aligned, and therefore choosing $W_z = -\lambda_z v$ for any $z$, with an appropriate scaling $\lambda_z \in \mathbb{R}$ is optimal. The gradient above becomes

$$\nabla_\theta \ell_{\text{ce}}^k(W\phi_\theta(x), y) = \nabla_\theta \phi_\theta(x)^\top v \ \left[\sum_{z \neq y} \frac{(\lambda_y + \lambda_z)e^{-\lambda_z v^\top \phi_\theta(x)}}{\sum_{z \leq k} e^{-\lambda_z v^\top \phi_\theta(x)}}\right],$$

and its sup with respect to $W$ is

$$\max_{\|W\|_{2,1} \leq \lambda} \left\|\nabla_\theta \ell_{\text{ce}}^k(W\phi_\theta(x), y)\right\| = \max_{\|v\|=1} \left[\left\|\nabla_\theta \phi_\theta(x)^\top v\right\| \left|\max_{\sum_{z \leq k} |\lambda_z| \leq \lambda} \sum_{z \neq y} \frac{(\lambda_y + \lambda_z)e^{\lambda_z v^\top \phi_\theta(x)}}{\sum_{z \leq k} e^{-\lambda_z v^\top \phi_\theta(x)}}\right|\right],$$

where we have used the fact that the constraint $\|W\|_{2,1} \leq \lambda$ on the linear parameters $W$ corresponds to the constraint $\sum_{z \leq k} |\lambda_z| \leq \lambda$.

We first note that to achieve the maximum, $\lambda_z \geq 0$ for all $z \neq y$, otherwise the term $\lambda_y + \lambda_z$ would be smaller than $\lambda_y + |\lambda_z|$ (note that the term in the exponential is not affected by the sign of $\lambda_z$, since we can choose $v$ such that $v^\top \phi_\theta(x)$ has either sign and $\left\|\nabla_\theta \phi_\theta(x)^\top v\right\| = \left\|-\nabla_\theta \phi_\theta(x)^\top v\right\|$. This implies that we do not need the absolute value on the term

$$\max_{\sum_{z \leq k} |\lambda_z| \leq \lambda} \sum_{z \neq y} \frac{(\lambda_y + \lambda_z)e^{\lambda_z v^\top \phi_\theta(x)}}{\sum_{z \leq k} e^{-\lambda_z v^\top \phi_\theta(x)}}.$$

Additionally, we note that the maximum above is achieved for any set of $(\lambda_z)_{z\leq k}$ such that $\lambda_{\bar{z}} > 0$ for some $\bar{z} \neq y$ and $\lambda_z = 0$ for all $z \neq \bar{z}$. This follows by noting that

$$\max_{\sum_{z\leq k}\lambda_z\leq\lambda} \sum_{z\neq y} \frac{(\lambda_y+\lambda_z)e^{-\lambda_z v^\top \phi_\theta(x)}}{\sum_{z\leq k} e^{-\lambda_z v^\top \phi_\theta(x)}} \leq \max_{\sum_{z\leq k}\lambda_z\leq\lambda} \sum_{z\neq y} \frac{e^{-\lambda_z v^\top \phi_\theta(x)}}{\sum_{z\leq k} e^{-\lambda_z v^\top \phi_\theta(x)}} \, (\lambda_y + \max_{z\neq y}\lambda_z).$$

Now, both $\max_{z\neq y}\lambda_z$ and $\sum_{z\neq y} \frac{e^{-\lambda_z v^\top \phi_\theta(x)}}{\sum_{z\leq k} e^{-\lambda_z v^\top \phi_\theta(x)}}$ are maximized by choosing all $\lambda_z$ to be equal to zero except for one. In particular the inequality above becomes an equality if $\lambda_{\bar{z}} = \lambda$ for some $\bar{z} \neq y$ and $\lambda_z = 0$ for any $z \neq \bar{z}$ (including $y$). Plugging this estimation of the maximum in the gradient norm derived above, we have

$$\max_{\|W\|_{2,1}\leq\lambda} \left\|\nabla_\theta \ell^k_{\text{ce}}(W\phi_\theta(x), y)\right\| = \max_{\|v\|=1} \left\|\nabla\phi_\theta(x)^\top v\right\| \frac{\lambda e^{-\lambda v^\top \phi_\theta(x)}}{e^{-\lambda v^\top \phi_\theta(x)} + k - 1}$$

We note now that, for any $\lambda$ and $v \in \mathbb{R}^m$, the quantity that we are maximizing in the equation above is strictly decreasing with respect to $k$. This means in particular that

$$\max_{(W,\theta)\in\Omega} \left\|\nabla_\theta \ell^k_{\text{ce}}(W\phi_\theta(x), y)\right\| \leq \max_{(W,\theta)\in\Omega} \left\|\nabla_\theta \ell^C_{\text{ce}}(W\phi_\theta(x), y)\right\|$$

**Putting the two together** Let us consider the norm squared of $\ell^C_{\text{ce}}$. We have

$$\begin{aligned} L_{\text{pre}} &= \sup_{(W,\theta)\in\Omega} \left\|\nabla_{W,\theta}\ \ell^C_{\text{ce}}(W\phi_\theta(x), y)\right\|^2 \\ &= \sup_{(W,\theta)\in\Omega} \left\|\nabla_W\ \ell^C_{\text{ce}}(W\phi_\theta(x), y)\right\|^2 + \left\|\nabla_\theta\ \ell^C_{\text{ce}}(W\phi_\theta(x), y)\right\|^2 \\ &\leq \sup_{(W,\theta)\in\Omega} \left\|\nabla_W\ \ell^k_{\text{ce}}(W\phi_\theta(x), y)\right\|^2 + O(e^{-2\lambda}) + \left\|\nabla_\theta\ \ell^k_{\text{ce}}(W\phi_\theta(x), y)\right\|^2 \\ &= \sup_{(W,\theta)\in\Omega} \left\|\nabla_{W,\theta}\ \ell^k_{\text{ce}}(W\phi_\theta(x), y)\right\|^2 + O(e^{-2\lambda}) \\ &= L^2_{\text{GLS}} + O(e^{-2\lambda}), \end{aligned}$$

where for the inequality we have used the fact that, for both gradients (with respect to $W$ or $\theta$), the maximum with respect to $W$, for a fixed $\theta$ is achieved by selecting a $W$ that is zero along all directions but one. Since $\sqrt{a+b} \leq \sqrt{a}+\sqrt{b}$ for any $a, b \geq 0$, we conclude

$$L_{\text{pre}} \leq \sqrt{L_{\text{GLS}} + O(e^{-2\lambda})} \leq L_{\text{GLS}} + O(e^{-\lambda}),$$

as required. $\square$

The theorem above shows that the Lipschitz constant for the (global) multi-class classification risk is at most larger than that for GLS by an amount that decreases exponentially fast as $\lambda$ increases. From regularization theory we know that $\lambda$ grows proportionally to the number of samples observed, hence we can expect that for all practical purposes $L_{\text{pre}}$ is smaller than $L_{\text{GLS}}$.

## A.2 Experimental Details

In this section we specify the datasets and hyperparameter settings for the experiments in the main body. The training follow the same steps regardless of dataset, we specify the available hyperparameters for each step below. We denote the steps (in sequence as they appear) of Alg. 2 by RepLearn, LearnLabeler, PreTrain, MetaLearn, and evaluation by Eval. The code repository with exact implementation details can be found at `https://github.com/isakfalk/mela`. In the code base we use PyTorch [Pas+19], scikit-learn [Ped+11], numpy [Har+20], matplotlib [Hun07] and umap [McI+18].

### A.2.1 Datasets

#### A.2.1.1 *mini*/*tiered*ImageNet

We use the standard *mini*ImageNet and *tiered*ImageNet dataset.

#### A.2.1.2 *mini*-60 and *tiered*-780

We provide code for generating the datasets of *mini*-60 and *tiered*-780 using the ILSVRC2012 ImageNet dataset and global labels provided (`https://www.image-net.org/challenges/LSVRC/2012/index.php`). Code and instructions for creating these datasets can be found in `https://github.com/isakfalk/mela`.

#### A.2.1.3 Meta-Dataset (mixed and H-Aircraft)

We use three component datasets of Meta-Dataset [Tri+20], those of `cu_birds`, `aircraft` and `vgg_flower` in the dataset summary table of the Meta-Dataset repository and pre-process them according to the instructions provided.

### A.2.2 Model Details

The hyperparameters across all experiments are specified below.

#### A.2.2.1 Optimization

We use two instances of optimization algorithms, relying on the optimization library of PyTorch

- SGD: SGD with an initial learning rate of 0.05, weight decay factor of 0.0005 and momentum of 0.9
- AdamW: AdamW [LH19] with learning rate of 0.0001, weight decay factor of $10^{-6}$.

For each optimization algorithm we use the torch multi-step learning rate scheduler which anneals the learning rate of the optimization algorithm by $\gamma = 0.1$ at selected epochs in `lr_decay_epochs`.

- MultiStepLR: Learning rate annealing scheduler, which multiplies the learning rate by $\gamma$ at the beginning of epochs in the list `lr_decay_epochs`.

#### A.2.2.2 Augmentation

We use two instances of augmentation (and one option of no augmentation)

- DATAAUG: Data augmentation where we use a pipeline of

  1. Random cropping using a shape of $84 \times 84$ with padding of 8
  2. Color jittering with the PyTorch arguments of (`brightness=0.4, contrast=0.4, saturation=0.4`)
  3. Randomly flip the image horizontally.
  4. Normalization: (channel-wise) using ImageNet sample channel mean and standard deviation for ImageNet type datasets, min-max scaling to $[-1, 1]$ for Mixed and H-aircraft datasets

- ROTATEAUG: Rotation-class augmentation as laid out in Sec. 3.4.1 together with DATAAUG

- NONE: No augmentation.

#### A.2.2.3 Backbone Architecture

We use ResNets [He+16] for the backbone throughout the experiments

- RESNET12: ResNet with block sequence $[1, 1, 1, 1]$, using adaptive average pooling, drop-blocks for the final 2 ResNet layers and a drop rate of 0.1, with output dimension being 640

- RESNET18: ResNet with block sequence $[1, 1, 2, 2]$, using adaptive average pooling, drop-blocks for the final 2 ResNet layers and a drop rate of 0.1, with output dimension being 640

#### A.2.2.4 Residual Adapters for Meta Fine-Tuning

In Sec. 3.4.3, we introduced a residual adapter for meta fine-tuning. The learnable network $h$ is a three-layer MLP with RESNET{12/18}+RESFC: MLP with residual connection and layer-normalization applied to the output. Both the input and output dimensions are the same as the feature representation from either RESNET12 or RESNET18 backbone.

#### A.2.2.5 Learning the Similarity Measure (RepLearn)

For training embedding $\phi_\theta^{\text{sim}}$, when given a task $D = (D^{\text{tr}} \cup D^{\text{val}})$ we one-hot encode the outputs and scale them using $f(y) = 2y - 1$. We get the classifier $w(\phi_\theta^{\text{sim}}(D^{\text{tr}}))$ using (2.16) on the embedded support set $\phi_\theta^{\text{sim}}(D^{\text{tr}})$ (we add a column of ones to the embeddings for a bias term) with a regularization strength of $\lambda_{\text{MetaLS}} = 0.001$. As an inner loss we use $\ell = \ell_{\text{FS}}$ where $\ell_{\text{FS}}$ is the few-shot loss using mean-squared error inner loss (2.16). We train for a fixed number of epochs, where each epoch is a full sweep over the meta-train set in the GFSL setting or some predefined number of tasks $T_{\text{tasks}}$ in the standard setting. Number of tasks in each batch is set to 1. We use meta-validation set for early stopping and model selection.

#### A.2.2.6 Global Label Inference (LearnLabeler)

Given a trained backbone $\phi_\theta^{\text{sim}}$ we use the clustering algorithm of Sec. 3.4.2.1 with the hyperparameters $q$ and $K_{\text{init}}$ where $q$ is the pruning aggression parameter and $K_{\text{init}}$ is the initial number of centroids on the same meta-train few-shot dataset on which $\phi_\theta^{\text{sim}}$ was trained, which gives rise to centroids $G$ and in extension the inferred global labels for standard multi-class classification.

#### A.2.2.7 Pre-Training via Multi-Class Classification (PreTrain)

Given a flat supervised dataset with inferred labels, we train a backbone $\phi_\theta^{\text{pre}}$ using the cross-entropy loss as in (2.19) with SGD and one of the data augmentation strategies outlined above. We train for a set number of epochs, where each epoch is a full sweep over the *flattened* meta-train set in the GFSL setting or some predefined number of tasks $T_{\text{tasks}}$ (inferred labels, flattened) in the standard setting. For the Oracles we use the same procedure on the full flat supervised dataset with the ground-truth labels. For the H-Aircraft dataset with multiple ground-truth labels, we use the semantic softmax [Rid+21] with augmentation DataAug. We use meta-validation set for early stopping and model selection.

#### A.2.2.8 Meta Fine-Tuning (MetaLearn)

We adapt the pre-trained representation $\phi_\theta^{\text{pre}}$ towards $\phi_\theta^*$ by combining the original backbone with the residual adapter. We use augmentation DataAug without rotation and optimize the empirical meta-risk using AdamW. We train for a set number of epochs, where each epoch is a sweep over the meta-train sets. Number of tasks per batch i set to 1. We use meta-validation set for early stopping and model selection.

#### A.2.2.9 Model Evaluation

During model evaluation (either using meta-validation or meta-testing set), we obtain the embedding of each task sample using the trained feature extractor. Each sample embedding is normalized to unit-length before being passed to the classifier $w(\cdot)$. We follow [Tia+20] and uses logistic regression from Scikit-learn as the classifier. We set the regularization strength as 1.0 for pre-trained feature extractor $\phi_\theta^{\text{pre}}$ and 0.001 for the fine-tuned feature extractor $\phi_\theta^*$.

#### A.2.2.10 Hyper-Parameters

The hyper-parameters and the values used in the experiments are listed in Table A.1.

## A.3 Complexity Analysis

In this section we specify the computational and memory complexity of all of the steps of the MeLa algorithm (Alg. 2). We specify the complexity of each of the four steps: pretraining the representation (RepLearn), global label inference (LearnLabeler), pretraining (PreTrain) and meta fine-tuning (MetaLearn).

For a representation network $\phi_\theta : \mathcal{X} \to \mathbb{R}^{d_\phi}$, we let the computational and memory complexity of mapping an instance $x$ to $\phi_\theta(x)$ be given by $O(k_\phi)$ and $O(m_\phi)$ respectively. As $w$ is KRR, it has a computational and memory complexity of $O(d_\phi^2 n_{\text{tr}} + d_\phi^3 + d_\phi n_{\text{tr}} N_{\text{way}})$ and $O(d_\phi N_{\text{way}})$ respectively which

Table A.1: Hyperparameters for all datasets.

| Pipeline Step | *mini*ImageNet | *tiered*ImageNet | mini-60 | tiered-780 | Mixed | H-aircraft |
|---|---|---|---|---|---|---|
| RepLearn | | | | | | |
| Architecture | ResNet12 | ResNet12 | ResNet12 | ResNet12 | ResNet12 | ResNet12 |
| Augmentation | None | None | None | None | None | None |
| Epochs | 30 | 40 | 30 | 40 | 50 | 50 |
| `lr_decay_epochs` | [20, 25] | [28, 36] | [20, 25] | [28, 36] | [30, 42] | [30, 42] |
| $T_{\text{tasks}}$ (if not GFSL) | 2800 | 2800 | 2800 | 2800 | 2800 | 2800 |
| LearnLabeler | | | | | | |
| Architecture | ResNet12 | ResNet12 | ResNet12 | ResNet12 | ResNet12 | ResNet12 |
| Augmentation | None | None | None | None | None | None |
| q | 3 | 3.5 | 4.5 | 4.5 | 3.5 | 3.5 |
| $K_{\text{init}}$ | 600 | 1200 | 700 | 1100 | 400 | 400 |
| Pretrain | | | | | | |
| Architecture | ResNet12 | ResNet18 | ResNet12 | ResNet12 | ResNet12 | ResNet12 |
| Augmentation | RotateAug | RotateAug | RotateAug | RotateAug | RotateAug | RotateAug |
| MetaLearn | | | | | | |
| Epochs | 3 | 3 | 3 | 3 | 3 | 3 |

depend on the train set size $n_{\mathrm{tr}}$ and input dimension $d_\phi$ and the number of ways $N_{\mathrm{way}}$. The validation set size is $n_{\mathrm{val}}$ and we let $n = n_{\mathrm{tr}} + n_{\mathrm{val}}$.

For simplicity, we assume that the batch size is 1, changing this is trivial, and we give the complexity in terms of per-gradient update.

### A.3.1 RepLearn

First mapping all of the instances using $\phi_\theta$ yields a computational complexity of $O(nk_\phi)$ and a memory complexity of $O(m_\phi + nd_\phi)$. Secondly, calculating the weights of KRR has a computational complexity of $O(d_\phi^2 n_{\mathrm{tr}} + d_\phi^3 + d_\phi n_{\mathrm{tr}} N_{\mathrm{way}})$ and a memory complexity of $O(d_\phi N_{\mathrm{way}})$. Next, predicting on the query set has a computational complexity of $O(d_\phi n_{\mathrm{val}} N_{\mathrm{way}})$ and a memory complexity of $O(1)$. Finally, calculating the empirical loss over the query set when using cross-entropy has computational and memory complexity which is negligible compared to previous terms. Finally, taking the gradient of this can be done in the same computational and memory complexity as getting the loss due to automatic differentiation. In total, this means we have a computational complexity of

$$O(nk_\phi + d_\phi^2 n_{\mathrm{tr}} + d_\phi^3 + d_\phi n N_{\mathrm{way}})$$

and a memory complexity of

$$O(m_\phi + d_\phi(n + N_{\mathrm{way}}))$$

per iteration of gradient descent.

### A.3.2 LearnLabeler

We provide computational and memory complexity for one step of this algorithm. Let the number of clusters of a step be $V$. We first need to calculate the centroids which can be done in $O(n_{\mathrm{shots}} d_\phi)$ computational complexity and $O(d_\phi)$ memory complexity, where $n_{\mathrm{shots}}$ is the number of shots per class in the train set. This is done for the $N_{\mathrm{way}}$ classes in the train set which gives $O(nd_\phi)$ and $O(N_{\mathrm{way}} d_\phi)$ in computational and memory complexity respectively.

For each of the $N_{\text{way}}$'s class centroids we need to find the closest of the current $V$ centroids. To do this we need to calculate the distance of each class centroid to all of the $V$ centroids which has computational complexity $O(Vd_\phi)$ and memory complexity $O(1)$. Together this has a computational complexity of $O(N_{\text{way}}Vd_\phi)$ and memory complexity of $O(1)$. As each class index in the task, and since there are $N_{\text{way}}$'s of them, are mapped to one of the $V$ centroids (assuming no ties, else we can choose one at random) we need to update $N_{\text{way}}$ number of centroids. Updating each centroid has a computational complexity of $O(d_\phi)$ and a memory complexity of $O(1)$, which leads to a total per-iteration update computational complexity of $O(N_{\text{way}}d_\phi)$ and memory complexity of $O(1)$.

In total, we have that each iteration update for the clustering step has a computational complexity of

$$O((n + N_{\text{way}}V)d_\phi) = O((n_{\text{shots}} + V)N_{\text{way}}d_\phi),$$

and a memory complexity of

$$O(N_{\text{way}}d_\phi).$$

### A.3.3 PreTrain

The pre-iteration pre-training is similar to that of RepLearn but simpler. First mapping all of the instances in a batch of size $B$ using $\phi_\theta$ yields a computational complexity of $O(Bk_\phi)$ and a memory complexity of $O(m_\phi + Bd_\phi)$. Calculating the loss is negligible compared to this and taking the gradients has the same complexity as the forward pass due to automatic differentiation. Thus the computational complexity is

$$O(Bk_\phi)$$

and the memory complexity is

$$O(m_\phi + Bd_\phi),$$

where here $d_\phi$ is the number of global classes.

### A.3.4 MetaLearn

This is essentially the same as RepLearn except that all of the inputs $x$ are replaced by $\phi_\theta^{\text{pre}}(x)$ where the $\phi_\theta^{\text{pre}}$ is the representation and then using a new feature map $\phi_\theta^*$. The only difference is that we have to map $x$ first through $\phi_\theta^{\text{pre}}$ to the output space of size $d_{\phi^{\text{pre}}}$ which yields a computational complexity of $O(nk_{\phi^{\text{pre}}})$ and a memory complexity of $O(m_{\phi^{\text{pre}}} + nd_{\phi^{\text{pre}}})$. Secondly, we have to do the same using the network $O(\phi_\theta^*)$ which yields a computational complexity of $O(nk_{\phi^*})$ and a memory complexity of $O(m_{\phi^*} + nd_{\phi^*})$. The rest of the analysis is exactly the same as that of RepLearn except that we do not take the gradient with respect to $\phi^{\text{pre}}$ only with respect to $\phi^*$. In total we have a computational complexity of complexity of

$$O(n(k_{\phi^{\text{pre}}} + k_{\phi^*}) + d_{\phi^*}^2 n_{\text{tr}} + d_{\phi^*}^3 + d_{\phi^*} n N_{\text{way}})$$

and a memory complexity of

$$O(m_{\phi^{\text{pre}}} + m_{\phi^*} + d_{\phi^*}(n + N_{\text{way}}))$$

per iteration of gradient descent.

# Appendix B

# IKML

The supplementary material is organized as follows. In Appendix B.1 we derive the closed form of the stochastic kernel of the affine pushforward kernel. In Appendix B.2 we derive the detailed bounds presented in Theorem 1. In Appendix B.3 we elaborate on the creation of the Air Quality (B.3.1) and the Gas Sensor (B.3.2) datasets. In Appendix B.4 we include the information on the numerical experiment presented in the main body. Finally, in Appendix B.5 we comment on the computational complexity of IKML and compare it against that of R2D2 since they both rely on KRR as the inner algorithm.

## B.1 Kernel for Affine Pushforward and Gaussian Latent

In this section we give the closed form of the kernel when the distribution $\tau$ is the affine pushforward of a standard Gaussian.

We use the following trick to find the closed form kernel. We can rewrite the kernel in Bochner's theorem as

$$K(x, x') = \int \cos(\langle \omega, x - x' \rangle) \, \mathrm{d}\tau(\omega) \tag{B.1}$$

$$= \int \Re(\cos(\langle \omega, x - x' \rangle) + i \sin(\langle \omega, x - x' \rangle)) \, \mathrm{d}\tau(\omega) \tag{B.2}$$

$$= \int \Re \exp(i\langle \omega, x - x' \rangle) \, \mathrm{d}\tau(\omega) \tag{B.3}$$

$$= \Re \int \exp(i\langle \omega, x - x' \rangle) \, \mathrm{d}\tau(\omega) \tag{B.4}$$

so finding the kernel is the same as finding the real part of the characteristic function (CF) of $\tau$. For a Gaussian the CF is well-known and we give it below.

**Lemma B.1.1.** *Let $\omega \sim \tau = \mathcal{N}(\mu, \Sigma)$ where $\Sigma$ is pd, then for any $\Delta \in \mathbb{R}^d$*

$$\int_{\mathbb{R}^d} \exp(i\omega^\top \Delta) \, \mathrm{d}\tau(\omega) = \exp(i\mu^\top \Delta - \frac{1}{2}\Delta^\top \Sigma \Delta). \tag{B.5}$$

*Proof.* The pdf of $\omega$ is $f(\omega) = (2\pi)^{-d/2} |\det(\Sigma)|^{-1/2} \exp(-\frac{1}{2}(\omega - \mu)\Sigma^{-1}(\omega - \mu))$. We make the change of variable $\phi = \Sigma^{-1/2}(\omega - \mu)$ so $\omega = \Sigma^{1/2}\phi + \mu$ where $\Sigma^{1/2}$ and $\Sigma^{-1/2}$ exist due to $\Sigma$ being pd. This means that $\mathrm{d}\omega = |\det(\Sigma)|^{1/2} \, \mathrm{d}\phi$ so that we have

$$\begin{aligned}
\int_{\mathbb{R}^d} \exp(i\omega^\top \Delta) \, \mathrm{d}\tau(\omega) &= \int_{\mathbb{R}^d} \exp(i\omega^\top \Delta) f(\omega) \, \mathrm{d}\omega \\
&= (2\pi)^{-d/2} \int_{\mathbb{R}^d} \exp(i(\Sigma^{1/2}\phi + \mu)^\top \Delta) \exp(-\frac{1}{2}\phi^\top \phi) \, \mathrm{d}\phi \\
&= (2\pi)^{-d/2} \exp(i\mu^\top \Delta) \int_{\mathbb{R}^d} \exp(i\phi^\top \Sigma^{1/2} \Delta) \exp(-\frac{1}{2}\phi^\top \phi) \, \mathrm{d}\phi \\
&= (2\pi)^{-d/2} (2\pi)^{d/2} \exp(i\mu^\top \Delta - \frac{1}{2}\Delta^\top \Sigma \Delta) \\
&= \exp(i\mu^\top \Delta - \frac{1}{2}\Delta^\top \Sigma \Delta).
\end{aligned} \tag{B.6}$$

□

Now we parameterize $\tau$ using $S \sim \mathcal{N}$ and $\theta = (Q, b)$ with $Q \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$ so that $\tau = \psi_{(Q,b)} \# \mathcal{N}$. An affine transformation of a Gaussian random variable is again Gaussian, and in this particular case it's easy to show that

$\tau \sim \mathcal{N}(b, QQ^\top)$. Combining (B.4) and Lemma B.1.1 we have

$$
\begin{aligned}
K(x, x') &= \Re \int \exp(i\langle \omega, x - x'\rangle) \,\mathrm{d}\tau(\omega) && \text{(B.7)}\\
&= \Re \exp(ib^\top(x - x') - \frac{1}{2}(x - x')^\top QQ^\top (x - x')) && \text{(B.8)}\\
&= \cos(b^\top(x - x')) \exp(-\frac{1}{2}(x - x')^\top QQ^\top (x - x')) && \text{(B.9)}\\
&= \cos(b^\top(x - x')) \exp(-\frac{1}{2}\|Q^\top(x - x')\|^2). && \text{(B.10)}
\end{aligned}
$$

## B.2 Error Decomposition

### B.2.1 Setup

We follow the notation of [Mau09] with some modifications and note that this differs at places from the notation used in the main body of the paper. We recall the meta-learning setting. There is some meta-distribution $\rho$ which generates tasks $\mu$, from $\mu$ we are given a train set $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \sim \mu^n$, where $(x, y) \in \mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^d \times [0, 1]$. Given the kernel ridge regression (KRR) algorithm with a fixed regularization parameter $\lambda > 0$ and an RKHS and corresponding kernel indexed by $\theta \in \Theta$, where $\Theta \subseteq \mathbb{R}^D$ is compact. We write this family as $\mathcal{H}_\Theta$ and the family of kernels as $\mathcal{K}_\Theta$. For a kernel $K_\theta$ let $\phi_\theta(x) = K_\theta(x, \cdot)$ be the canonical feature map and $\mathcal{H}_\theta$ the corresponding RKHS. The KRR solution is

$$
\omega_\theta(\boldsymbol{z}) = \operatorname*{argmin}_{w \in \mathcal{H}_\theta} \left( \frac{1}{n} \sum_{i=1}^{n} (\langle w, \phi_\theta(x_i)\rangle_\theta - y_i)^2 + \lambda \|w\|_\theta^2 \right), \tag{B.11}
$$

where we use $\langle \cdot, \cdot \rangle_\theta$ and $\|\cdot\|_\theta$ to denote the inner product and norm in RKHS $\mathcal{H}_\theta$. We will drop $\theta$ when it's clear what RKHS we are referring to. Given a weight vector $w \in \mathcal{H}_\theta$, a prediction on a new datapoint $x$ is given by $\langle w, \phi_\theta(x)\rangle$.

The transfer risk of the algorithm $\omega_\theta$ and a loss $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is defined to be

$$
\mathcal{E}(\theta) = \mathbb{E}_{\mu \sim \rho} \mathbb{E}_{\boldsymbol{z} \sim \mu^n} \mathbb{E}_{(x,y) \sim \mu} \ell(\langle \omega_\theta(\boldsymbol{z}), \phi_\theta(x)\rangle, y). \tag{B.12}
$$

We have access to $T$ datasets from tasks by sampling $(\mu_t)_{t=1}^T \sim \rho^T$ which gives rise to datasets $\boldsymbol{z}^t = (\boldsymbol{x}^t, \boldsymbol{y}^t) \sim \mu_t^n$. For the meta-dataset $\boldsymbol{Z} = (\boldsymbol{z}^t)_{t=1}^T$ sampled by first sampling $(\mu_t)_{t=1}^T \sim \rho^T$ and then $\boldsymbol{z}^t \sim \mu^n$, we denote this sampling process by $\boldsymbol{Z} \sim \hat{\rho}^T$. Using the KRR algorithm $\omega_\theta$ we let

$$\hat{\ell}_\theta(\boldsymbol{z}^t) = \frac{1}{n}\sum_{i=1}^n \ell(\langle \omega_\theta(\boldsymbol{z}^t), \phi_\theta(x_i^t)\rangle, y_i^t), \tag{B.13}$$

which is the training error of task $t$ using $\omega_\theta$. For a sample of latent variables $S = (s_k)_{k=1}^M \sim \mathcal{N}^M$ so that the random features $\psi_\theta(s_k) \sim \tau_\theta$ (that is $\tau_\theta = \psi_\theta \# \mathcal{N}$), in which case we define $K_\theta = K_{\tau_\theta}$ and $K_{\theta,S} = K_{\hat{\tau}_\theta S}$, we let

$$\hat{\ell}_\theta(\boldsymbol{z}^t, S) = \frac{1}{n}\sum_{i=1}^n \ell(\langle \omega_{\theta,S}(\boldsymbol{z}^t), \phi_{\theta,S}(x_i^t)\rangle, y_i^t), \tag{B.14}$$

where $\omega_{\theta,S}$ is the same as (B.11) where we replace the kernel $K_\theta$ by the random feature kernel $K_{\theta,S}$ and the corresponding RKHS, see Appendix B.2.2. When the algorithm $\omega$ is clear from context we simply write $\hat{\ell}(\boldsymbol{z})$ and $\hat{\ell}(\boldsymbol{z}, S)$. We opt to select $\theta$ using ERM, letting

$$\hat{\theta} = \operatorname*{argmin}_{\theta \in \Theta} \left\{ \hat{\mathcal{E}}_T(\theta) = \frac{1}{T}\sum_{t=1}^T \hat{\ell}_\theta(\boldsymbol{z}^t) \right\}. \tag{B.15}$$

As the problem of (B.15) is non-convex we cannot solve it in general. We let $\tilde{\theta}$ be the output of an optimization procedure $\tilde{\theta} = \mathrm{Alg}(\hat{\mathcal{E}}_T)$ and encode this optimization discrepancy through the term $\hat{\mathcal{E}}_T(\tilde{\theta}) - \hat{\mathcal{E}}_T(\hat{\theta})$.

### B.2.2 Kernel Family

Let $\mathcal{H}$ be an RKHS defined by Bochner's theorem through the kernel defined by any probability measure $\tau \in M_1(\mathcal{X})$,

$$K(x, x') = \int \xi(x; v)\overline{\xi}(x'; v)\, \mathrm{d}\tau(v). \tag{B.16}$$

We will assume that $\xi(x; v) = \exp(iv^\top x)$, but the analysis should generalize to the more general setting. For a real-valued kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, it can be shown that any such kernel satisfying (B.16) can be rewritten as

$$K(x, x') = \int_{(-\pi/2,\pi/2]^d} \cos(\langle v, x - x'\rangle)\, \mathrm{d}\tau'(v), \tag{B.17}$$

for some measure $\tau'$ with support on $(-\pi/2, \pi]^d$. For IKML we parameterise a class of measures by $\psi_\theta\#\mathcal{N}$ where $\psi_\theta$ is an MLP with weights $\theta$ and we denote the kernel and RKHS by $K_\theta$ and $\mathcal{H}_\theta$.

Given a dataset of inputs $\boldsymbol{x} = (x_i)_{i=1}^n$, denote the kernel matrix $\boldsymbol{G}_\theta(\boldsymbol{x})$ so that $\boldsymbol{G}_\theta(\boldsymbol{x})_{ij} = K_\theta(x_i, x_j)$ and let $\boldsymbol{G}_{\theta,\lambda}(\boldsymbol{x}) = \boldsymbol{G}_\theta(\boldsymbol{x}) + n\lambda I$. Similarly for a set of latents $S \sim \mathcal{N}^M$ we denote the respective matrices $\boldsymbol{G}_\theta(\boldsymbol{x}, S)$ and $\boldsymbol{G}_{\theta,\lambda}(\boldsymbol{x}, S)$ were we replace every instance of

$$K_\theta(x, x') = \int \cos(\langle \psi_\theta(s), x - x'\rangle) \tag{B.18}$$

by the empirical mean

$$K_{\theta,S}(x, x') = \frac{1}{M}\sum_{j=1}^{M} \cos(\langle \psi_\theta(s_j), x - x'\rangle) = \phi_{\theta,S}(x)^\top \phi_{\theta,S}(x'), \tag{B.19}$$

where

$$\phi_{\theta,S}(x) = \frac{1}{\sqrt{M}}((\sin(\psi_\theta(s_i)^\top x), \cos(\psi_\theta(s_i)^\top x))^\top)_{i=1}^M \in \mathbb{R}^{2M}. \tag{B.20}$$

We will omit $\theta$ and $\boldsymbol{x}$ from $\boldsymbol{G}_\theta(\boldsymbol{x})$ when clear from context. Similarly we let $\hat{\ell}_\theta(\boldsymbol{x}, \boldsymbol{y}, S)$ be the train loss when trained on $\boldsymbol{x}, \boldsymbol{y}$ with random features induced by $S$ and we omit $\theta$ when clear from context.

### B.2.3 Auxiliary Results

Let $\|\cdot\|_\infty$ be the operator norm and $\|\cdot\|_F$ the Frobenius norm. For an algorithm $\omega$ and a dataset $\boldsymbol{z}$, let $\hat{\ell}(\boldsymbol{z})$ be the training error of $\omega$ on $\boldsymbol{z}$ using loss $\ell$.

**Definition B.2.1.** *Given any $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$ or two input sets $\boldsymbol{x}_1, \boldsymbol{x}_2$ of size $n$, where $x \in \mathcal{X}$ and $y \in [0, 1]$, relative to a fixed loss function $\ell$, an algorithm $\omega$*

*taking outputs in an RKHS $\mathcal{H}$ is said to be*

- *$\beta$-bounded if $\|\omega(\boldsymbol{z})\| \leq \beta$ and $\hat{\ell}(\boldsymbol{z}) \leq \beta$.*
- *have kernel stability $L$ if*

$$\hat{\ell}(\boldsymbol{x}_1, \boldsymbol{y}) - \hat{\ell}(\boldsymbol{x}_2, \boldsymbol{y}) \leq \frac{L}{n}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_F \tag{B.21}$$

- *have random feature stability $L$ if*

$$\hat{\ell}(\boldsymbol{x}, \boldsymbol{y}, S) - \hat{\ell}(\boldsymbol{x}, \boldsymbol{y}) \leq \frac{L}{n}\|\boldsymbol{G}(\boldsymbol{x}, S) - \boldsymbol{G}(\boldsymbol{x})\|_F. \tag{B.22}$$

**Lemma B.2.1** ([Mau09], Lemma 3)**.** *Let $G_1$ and $G_2$ be positive semidefinite operators on any Hilbert space and $\lambda > 0$, then*

1. *$G_i + \lambda I$ is invertible,*
2. *$\|(G_i + \lambda I)^{-1}\|_\infty \leq \frac{1}{\lambda}$ and*
3. *we have*

$$\left\|(G_1 + \lambda I)^{-1} - (G_2 + \lambda)^{-1}\right\|_\infty \leq \frac{1}{\lambda^2}\|G_1 - G_2\|_\infty. \tag{B.23}$$

4. *Let $\phi_1, \phi_2$ satisfy $(G_i + \lambda I)\phi_i = y$. Then*

$$\left|\|\phi_1\|^2 - \|\phi_2\|^2\right| \leq 2\lambda^{-3}\|G_1 - G_2\|_\infty \|y\|^2 \tag{B.24}$$

For any dataset $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$ of size $n$, kernel $K$ with RKHS $\mathcal{H}$ and feature map $\phi$, and corresponding KRR algorithm $\omega$, we define the following quantities, following [Mau09],

$$\omega(\boldsymbol{z}) = \operatorname*{argmin}_{w\in\mathcal{H}} \left( \frac{1}{n}\sum_{i=1}^{n}(\langle w, \phi(x_i)\rangle - y_i)^2 + \lambda\|w\|^2 \right), \tag{B.25}$$

$$\hat{\ell}_\omega(\boldsymbol{z}) = \frac{1}{n}\sum_{i=1}^{n}(\langle \omega(\boldsymbol{z}), \phi(x_i)\rangle - y_i)^2, \tag{B.26}$$

$$\xi_\omega(\boldsymbol{z}) = \min_{w\in\mathcal{H}} \left( \frac{1}{n}\sum_{i=1}^{n}(\langle w, \phi(x_i)\rangle - y_i)^2 + \lambda\|w\|^2 \right) = \hat{\ell}_\omega(\boldsymbol{z}) + \lambda\|\omega(\boldsymbol{z})\|^2 \tag{B.27}$$

**Proposition B.2.1.** *For any kernel $K$ of the form (B.17), for any dataset $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$ or two input sets $\boldsymbol{x}_1, \boldsymbol{x}_2$, where $x \in \mathcal{X}, y \in [0,1]$, of size $n$ and a sample of random features $S \sim \mathcal{N}^M$ we have that*

1. $\hat{\ell}_\omega(\boldsymbol{z}) \leq 1$, $\|\omega(\boldsymbol{z})\| \leq \lambda^{-1/2}$, $\xi_\omega(\boldsymbol{z}) \leq 1$,
2. $\left|\hat{\ell}_\omega(\boldsymbol{x}_1, \boldsymbol{y}) - \hat{\ell}_\omega(\boldsymbol{x}_2, \boldsymbol{y})\right| \leq \frac{2\lambda^{-1}}{n}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_F$,
3. $\left|\hat{\ell}_\omega(\boldsymbol{x}, \boldsymbol{y}) - \hat{\ell}_{\omega,S}(\boldsymbol{x}, \boldsymbol{y})\right| \leq \frac{2\lambda^{-1}}{n}\|\boldsymbol{G}(\boldsymbol{x}) - \boldsymbol{G}(\boldsymbol{x}, S)\|_F$,
4. $|\xi_\omega(\boldsymbol{x}_1, \boldsymbol{y}) - \xi_\omega(\boldsymbol{x}_2, \boldsymbol{y})| \leq \frac{\lambda^{-1}}{n}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_F$,
5. $|\xi_\omega(\boldsymbol{x}, \boldsymbol{y}) - \xi_{\omega,S}(\boldsymbol{x}, \boldsymbol{y})| \leq \frac{\lambda^{-1}}{n}\|\boldsymbol{G}(\boldsymbol{x}) - \boldsymbol{G}(\boldsymbol{x}, S)\|_F$

*where $\boldsymbol{G}(\boldsymbol{x}, S)$ is the kernel matrix of $\boldsymbol{x}$ using random features induced by $S$.*

*Proof.* We simply note that

$$\begin{aligned}
\hat{\ell}_\omega(\boldsymbol{z}) + \lambda\|\omega(\boldsymbol{z})\|^2 &= \xi_\omega(\boldsymbol{z}) && \text{(B.28)}\\
&= \min_{w\in\mathcal{H}} \left( \frac{1}{n}\sum_{i=1}^{n}(\langle w, \phi(x_i)\rangle - y_i)^2 + \lambda\|w\|^2 \right) && \text{(B.29)}\\
&\leq \frac{1}{n}\sum_{i=1}^{n}(\langle 0, \phi(x_i)\rangle - y_i)^2 + \lambda\|0\|^2 && \text{(B.30)}\\
&\leq 1.
\end{aligned}$$

Since both $\hat{\ell}_\omega(\boldsymbol{z})$ and $\lambda\|\omega(\boldsymbol{z})\|^2$ are positive and the sum is less than 1, we have that $\hat{\ell}_\omega(\boldsymbol{z}) \leq 1$ and $\lambda\|\omega(\boldsymbol{z})\|^2 \leq 1$ which implies that $\|\omega(\boldsymbol{z})\| \leq \lambda^{-1/2}$.

For the second point, using the dual formulation $\boldsymbol{G}_\lambda(\boldsymbol{x})\alpha = \boldsymbol{y}$ and $\langle \omega(\boldsymbol{z}), \phi(x_i)\rangle = (\boldsymbol{G}(\boldsymbol{x})\alpha)_i$,

$$\begin{aligned}\hat{\ell}_\omega(\boldsymbol{z}) &= \frac{1}{n}\|\boldsymbol{G}(\boldsymbol{x})\alpha - \boldsymbol{y}\|^2 = \frac{1}{n}\|\boldsymbol{G}_\lambda(\boldsymbol{x})\alpha - \boldsymbol{y} - \lambda n\alpha\|^2 \qquad \text{(B.31)}\\ &= \frac{1}{n}\|\lambda n\alpha\|^2 = \lambda^2 n\|\alpha\|^2. \qquad \text{(B.32)}\end{aligned}$$

Using this and the fact that $\|\omega(\boldsymbol{z})\|^2 = \alpha^\top \boldsymbol{G}(\boldsymbol{x})\alpha$ in $\xi_\omega$,

$$\begin{aligned}\xi_\omega(\boldsymbol{z}) &= \hat{\ell}_\omega(\boldsymbol{z}) + \lambda\|\omega(\boldsymbol{z})\|^2 \qquad \text{(B.33)}\\ &= \lambda^2 n\|\alpha\|^2 + \lambda\alpha^\top \boldsymbol{G}(\boldsymbol{x})\alpha \qquad \text{(B.34)}\\ &= \lambda(\lambda n\alpha^\top\alpha + \alpha^\top\boldsymbol{G}(\boldsymbol{x})\alpha) = \lambda(\alpha^\top\boldsymbol{G}_\lambda(\boldsymbol{x})\alpha) = \lambda(\boldsymbol{y}^\top\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}\boldsymbol{y}). \qquad \text{(B.35)}\end{aligned}$$

Thus

$$\begin{aligned}\left|\hat{\ell}_\omega(\boldsymbol{x}_1,\boldsymbol{y}) - \hat{\ell}_\omega(\boldsymbol{x}_2,\boldsymbol{y})\right| &= \lambda^2 n\left|\left\|\boldsymbol{G}_\lambda(\boldsymbol{x}_1)^{-1}\boldsymbol{y}\right\|^2 - \left\|\boldsymbol{G}_\lambda(\boldsymbol{x}_2)^{-1}\boldsymbol{y}\right\|^2\right| \qquad \text{(B.36)}\\ &\le (\lambda^2 n)2(\lambda n)^{-3}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_\infty\|\boldsymbol{y}\|^2 \qquad \text{(B.37)}\\ &\le 2\lambda^{-1}n^{-2}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_\infty\|\boldsymbol{y}\|^2 \qquad \text{(B.38)}\\ &\le \frac{2\lambda^{-1}}{n}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_F, \qquad \text{(B.39)}\end{aligned}$$

where we have used point 4 in Lemma B.2.1 and the fact that $\|\boldsymbol{y}\|^2 = \sum_{i=1}^n y_i^2 \le n$ as $y_i \in [0,1]$ for any $i \in [n]$. Then

$$\begin{aligned}|\xi_\omega(\boldsymbol{x}_1,\boldsymbol{y}) - \xi_\omega(\boldsymbol{x}_2,\boldsymbol{y})| &\le \lambda\left|\boldsymbol{y}^\top\boldsymbol{G}_\lambda(\boldsymbol{x}_1)^{-1}\boldsymbol{y} - \boldsymbol{y}^\top\boldsymbol{G}_\lambda(\boldsymbol{x}_2)^{-1}\boldsymbol{y}\right| \qquad \text{(B.40)}\\ &\le \lambda\left|\boldsymbol{y}^\top(\boldsymbol{G}_\lambda(\boldsymbol{x}_1)^{-1} - \boldsymbol{G}_\lambda(\boldsymbol{x}_2)^{-1})\boldsymbol{y}\right| \qquad \text{(B.41)}\\ &\le \lambda(\lambda n)^{-2}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_\infty\|\boldsymbol{y}\|^2 \qquad \text{(B.42)}\\ &\le \lambda n(\lambda n)^{-2}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_F \qquad \text{(B.43)}\\ &\le \frac{\lambda^{-1}}{n}\|\boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{G}(\boldsymbol{x}_2)\|_F. \qquad \text{(B.44)}\end{aligned}$$

For the third point and fifth point, the proof is the same as above, replacing $\boldsymbol{G}_\lambda(\boldsymbol{x_1})$ and $\boldsymbol{G}_\lambda(\boldsymbol{x_2})$ with $\boldsymbol{G}_\lambda(\boldsymbol{x})$ and $\boldsymbol{G}_\lambda(\boldsymbol{x}, S)$ respectively. Thus all of the results follows. $\square$

**Definition B.2.2** (Complexities)**.** *Let $(\sigma_i)_{i=1}^k$ denote a sequence of independent Rademacher variables (Uniform distribution on $\{-1,1\}$) independent of each other. For a set $A \subseteq \mathbb{R}^k$, the Rademacher and Gaussian complexities are defined to be*

$$\mathcal{R}(A) = \mathbb{E}_\sigma \sup_{\boldsymbol{x}\in A} \frac{2}{k}\sum_{i=1}^{k} \sigma_i x_i. \tag{B.45}$$

*If $\mathcal{F}$ is a class of real functions on a space $\mathcal{X}$ and $\boldsymbol{x} \in \mathcal{X}^k$, we write*

$$\mathcal{F}(\boldsymbol{x}) = \{(f(x_1), \dots, f(x_k)) \,:\, f \in \mathcal{F}\} \subseteq \mathbb{R}^k. \tag{B.46}$$

*The empirical Rademacher complexities of $\mathcal{F}$ on $\boldsymbol{x}$ is $\mathcal{R}(\mathcal{F}(\boldsymbol{x}))$. If $\mu \in M_1(\mathcal{X})$ is a probability measure on $\mathcal{X}$ then the corresponding expected complexity is $\mathbb{E}_{\boldsymbol{x}\sim\mu^k}\mathcal{R}(\mathcal{F}(\boldsymbol{x}))$.*

**Theorem 4** ([Mau09], Thm. 4)**.** *Let $\mathcal{F}$ be a real-valued function class on a space $\mathcal{X}$ and $\mu \in M_1(\mathcal{X})$. For $\boldsymbol{x} = (x_1, \dots, x_k) \in \mathcal{X}^k$ define*

$$\Phi(\boldsymbol{x}) = \sup_{f\in\mathcal{F}} \left( \mathbb{E}_{x\sim\mu} f(x) - \frac{1}{k}\sum_{i=1}^{k} f(x_i) \right). \tag{B.47}$$

*Then*

1. *$\mathbb{E}_{\boldsymbol{x}\sim\mu^k}\Phi(\boldsymbol{x}) \le \mathbb{E}_{\boldsymbol{x}\sim\mu^k}\mathcal{R}(\mathcal{F}(\boldsymbol{x}))$,*
2. *if $\mathcal{F}$ is $[0,1]$-valued, then for any $\delta > 0$ we have with probability greater than $1-\delta$ in $\boldsymbol{x} \sim \mu^k$ that*

$$\Phi(\boldsymbol{x}) \le \mathbb{E}_{\boldsymbol{x}\sim\mu^k}\mathcal{R}(\mathcal{F}(\boldsymbol{x})) + \sqrt{\frac{\log(1/\delta)}{2k}}. \tag{B.48}$$

**Corollary B.2.1** ([Mau09], Corollary 1)**.** *Let $A \subseteq \mathbb{R}^k$ and $\phi_1, \dots, \phi_k$ be real functions, each with Lipschitz constant $L$. Denote $\phi \circ A = \{(\phi_1(x_1), \dots, \phi_k(x_k)) \,:\, (x_1, \dots, x_k) \in A\}$. Then $\mathcal{R}(\phi \circ A) \le L\mathcal{R}(A)$.*

### B.2.4 Decomposition

We want to control the excess meta-risk $\mathbb{E}_{S\sim\mathcal{N}^M}[\mathcal{E}(\hat{\theta}, S) - \mathcal{E}(\theta^*)]$, where $\theta^* = \operatorname{argmin}_{\theta\in\Theta}\mathcal{E}(\theta)$. We introduce the following terms

$$\hat{\mathcal{E}}(\theta) = \mathbb{E}_{\boldsymbol{Z}\sim\rho^T}\hat{\mathcal{E}}_T(\theta) \tag{B.49}$$

and the corresponding term $\hat{\mathcal{E}}(\theta, S)$ where we replace the kernel $K_\theta$ by $K_{\theta,S}$. We decompose the excess meta-risk as follows

$$\begin{aligned}&\mathbb{E}_{S\sim\mathcal{N}^M}[\mathcal{E}(\hat{\theta}, S) - \mathcal{E}(\theta^*)] = \\ &\mathbb{E}_{S\sim\mathcal{N}^M}[\underbrace{\mathcal{E}(\hat{\theta}, S) - \hat{\mathcal{E}}(\hat{\theta}, S)}_{(A)} + \underbrace{\hat{\mathcal{E}}(\hat{\theta}, S) - \hat{\mathcal{E}}_T(\hat{\theta}, S)}_{(B)} + \underbrace{\hat{\mathcal{E}}_T(\hat{\theta}, S) - \hat{\mathcal{E}}_T(\theta^*, S)}_{(C)} \\ &\quad + \underbrace{\hat{\mathcal{E}}_T(\theta^*, S) - \hat{\mathcal{E}}(\theta^*, S)}_{(D)} + \underbrace{\hat{\mathcal{E}}(\theta^*, S) - \mathcal{E}(\theta^*, S)}_{(E)} + \underbrace{\mathcal{E}(\theta^*, S) - \mathcal{E}(\theta^*)}_{(F)}]\end{aligned} \tag{B.50}$$

We bound each of the terms.

### B.2.5 Bounding the Estimation Error for the Future Task

This follows [Mau09, Sec. 4.1], but we present the results in the order that they are needed. This argument bounds both $(A)$ and $(E)$.

**Theorem 5** (Upper bound of estimation error for future task)**.** *For any $\theta \in \Theta$, any loss $\ell$ such that for all $y \in [0, 1]$ $\ell(\cdot, y) : [-L, L] \to \mathbb{R}_+$ has Lipschitz constant* Lip$(L)$*, with $\omega$ being KRR with regularization parameter $\lambda > 0$ and RKHS induced by $K_\theta$*

$$\mathbb{E}_{S\sim\mathcal{N}^M}[\mathcal{E}(\theta, S) - \hat{\mathcal{E}}(\theta, S)] \leq \mathrm{Lip}(\lambda^{-1/2})\mathbb{E}_{S\sim\mathcal{N}^M}\mathbb{E}_{\boldsymbol{z}\sim\hat{\rho}}\mathcal{R}(\mathcal{G}(\boldsymbol{z})), \tag{B.51}$$

*where $\mathcal{G} = \{z = (x, y) \mapsto \lambda^{-1/2}\langle v, \phi_{\theta,S}(x)\rangle_{\theta,S} \ : \ \|v\|_{\theta,S} \leq 1\}$. Furthermore, we also have the upper bound*

$$\mathbb{E}_{S\sim\mathcal{N}^M}[\mathcal{E}(\theta, S) - \hat{\mathcal{E}}(\theta, S)] \leq \frac{2\lambda^{-1/2}\mathrm{Lip}(\lambda^{-1/2})}{\sqrt{n}}. \tag{B.52}$$

*Proof.* We may rewrite term $\mathbb{E}_{S\sim\mathcal{N}^M}[\mathcal{E}(\theta,S)-\hat{\mathcal{E}}(\theta,S)]$ as

$$\mathbb{E}_{S\sim\mathcal{N}^K}\mathbb{E}_{\mu\sim\rho}\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mu^n}\left(\mathbb{E}_{(x,y)\sim\mu}\ell(\langle\omega_{\theta,S}(\boldsymbol{x},\boldsymbol{y}),\phi_{\theta,S}(x)\rangle,y)-\hat{\ell}_\theta(\boldsymbol{x},\boldsymbol{y},S)\right). \tag{B.53}$$

This is bounded in [Mau09, Thm. 6], and we follow similarly. For a fixed $\theta\in\Theta$ and any sample $S$, let $\mathcal{W}=\{w:\|w\|_{\theta,S}\le\lambda^{-1/2}\}$. By proposition B.2.1, we have that for any dataset $\boldsymbol{z}$ of size $n$ generated according to our assumptions, for any $\theta\in\Theta$, $\|\omega_{\theta,S}(\boldsymbol{z})\|_{\theta,S}\le\lambda^{-1/2}$. Thus, for any $\mu\in M_1(\mathcal{X}\times[0,1])$,

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mu^n}\left(\mathbb{E}_{(x,y)\sim\mu}\ell(\langle\omega_{\theta,S}(\boldsymbol{x},\boldsymbol{y}),\phi_{\theta,S}(x)\rangle,y)-\hat{\ell}_\theta(\boldsymbol{x},\boldsymbol{y},S)\right) \tag{B.54}$$
$$\le\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mu^n}\sup_{w\in\mathcal{W}}\left(\mathbb{E}_{(x,y)\sim\mu}\ell(\langle w,\phi_{\theta,S}(x)\rangle,y)-\hat{\ell}_\theta(\boldsymbol{x},\boldsymbol{y},S)\right) \tag{B.55}$$
$$=\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mu^n}\sup_{f\in\mathcal{F}}\left(\mathbb{E}_{(x,y)\sim\mu}f(z)-\frac{1}{n}\sum_{i=1}^n f(z_i)\right) \tag{B.56}$$

where we have the family of functions

$$\mathcal{F}=\{z=(x,y)\mapsto\ell(\lambda^{-1/2}\langle v,\phi_{\theta,S}(x)\rangle,y)\;:\;\|v\|_{\theta,S}\le 1\}. \tag{B.57}$$

By Thm. 4 we can upper bound this by the Rademacher complexity, getting the upper bound

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mu^n}\sup_{f\in\mathcal{F}}\left(\mathbb{E}_{(x,y)\sim\mu}f(z)-\frac{1}{n}\sum_{i=1}^n f(z_i)\right)\le\mathbb{E}_{\boldsymbol{z}\sim\mu^n}\mathcal{R}(\mathcal{F}(\boldsymbol{z})) \tag{B.58}$$

Furthermore, by assumption $y\in[0,1]$ and $\ell(\cdot,y)$ has a Lipschitz constant upper bounded by $\mathrm{Lip}(L)$ when we consider $\mathrm{dom}(\ell(\cdot,y))=[-L,L]$. By Cauchy-Schwartz and $\|v\|_{\theta,S}\le 1$, we have that $\lambda^{-1/2}\langle v,\phi_{\theta,S}(x)\rangle_{\theta,S}\in[-\lambda^{-1/2},\lambda^{1/2}]$ and so $L=\lambda^{-1/2}$. Letting $\phi_i:t\mapsto\ell(t,y_i)$ with domain $[-\lambda^{-1/2},\lambda^{-1/2}]$ then the Lipschitz constant is $\mathrm{Lip}(\lambda^{-1/2})$. We let $\mathcal{G}=\{z=(x,y)\mapsto\lambda^{-1/2}\langle v,\phi_{\theta,S}(x)\rangle_{\theta,S}\;:\;\|v\|_{\theta,S}\le 1\}$.

Since $\mathcal{F} = \phi \circ \mathcal{G}$ we have by Cor. B.2.1 that

$$\begin{aligned}
&\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mu^n}\left(\mathbb{E}_{(x,y)\sim\mu}\ell(\langle\omega_{\theta,S}(\boldsymbol{x},\boldsymbol{y}),\phi_{\theta,S}(x)\rangle,y) - \hat{\ell}_{\omega_{\theta,S}}(\boldsymbol{x},\boldsymbol{y})\right) && \text{(B.59)}\\
&\leq \mathbb{E}_{\boldsymbol{z}\sim\mu^n}\mathcal{R}(\phi\circ\mathcal{G}(\boldsymbol{z})) && \text{(B.60)}\\
&\leq \mathrm{Lip}(\lambda^{-1/2})\mathbb{E}_{\boldsymbol{z}\sim\mu^n}\mathcal{R}(\mathcal{G}(\boldsymbol{z})). && \text{(B.61)}
\end{aligned}$$

We can further bound $\mathbb{E}_{\boldsymbol{z}\sim\mu^n}\mathcal{R}(\mathcal{G}(\boldsymbol{z}))$ using a standard RKHS rademacher complexity argument.

By standard arguments of Rademacher complexity of kernels such that $K(x,x) = 1$ we have the bound

$$\mathbb{E}_{\boldsymbol{z}\sim\mu^n}\mathcal{R}(\mathcal{G}(\boldsymbol{z})) \leq \frac{2\lambda^{-1/2}}{\sqrt{n}} \tag{B.62}$$

Substituting the upper bounds $\mathrm{Lip}(\lambda^{-1/2})\mathbb{E}_{\boldsymbol{z}\sim\mu^n}\mathcal{R}(\mathcal{G}(\boldsymbol{z}))$ or $\frac{2\lambda^{-1/2}\mathrm{Lip}(\lambda^{-1/2})}{\sqrt{n}}$ of (B.54) and combining everything we have

$$\begin{aligned}
&\mathbb{E}_{S\sim\mathcal{N}^k}[\mathcal{E}(\hat{\theta},S) - \hat{\mathcal{E}}(\hat{\theta},S)] \leq \mathrm{Lip}(\lambda^{-1/2})\mathbb{E}_{S\sim\mathcal{N}^M}\mathbb{E}_{\boldsymbol{z}\sim\hat{\rho}}\mathcal{R}(\mathcal{G}(\boldsymbol{z})) && \text{(B.63)}\\
&\mathbb{E}_{S\sim\mathcal{N}^k}[\mathcal{E}(\hat{\theta},S) - \hat{\mathcal{E}}(\hat{\theta},S)] \leq \frac{2\lambda^{-1/2}\mathrm{Lip}(\lambda^{-1/2})}{\sqrt{n}} && \text{(B.64)}
\end{aligned}$$

□

We note the following about the bound above. The bound $\mathbb{E}_{\boldsymbol{z}\sim\hat{\rho}}\mathcal{R}(\mathcal{G}(\boldsymbol{z})) \leq \frac{2\lambda^{-1/2}}{\sqrt{n}}$ is standard and applies to all kernels such that $K(x,x) = 1$. However, in the benign case that $\mathbb{E}_{S\sim\mathcal{N}^M}\mathcal{R}(\mathcal{G}(\boldsymbol{z})) \ll \frac{2\lambda^{-1/2}\mathrm{Lip}(\lambda^{-1/2})}{\sqrt{n}}$, using IKML would lead to a term much smaller than fixing a kernel and learning the tasks independently.

### B.2.6 Predicting the Empirical Error for the Future Task

In this section we focus on the terms $(B), (D)$, each of the form $\mathbb{E}_{S\sim\mathcal{N}^M}\hat{\mathcal{E}}_T(\theta,S) - \hat{\mathcal{E}}(\theta,S)$ where $\theta\in\Theta$. To control this term we use [Mau16, Sec. 4.2]. We want to control the term $\hat{\mathcal{E}}_T(\theta, S) - \hat{\mathcal{E}}(\theta, S)$ using a uniform bound of the form

$$\hat{\mathcal{E}}_T(\theta, S) - \hat{\mathcal{E}}(\theta, S) \leq \sup_{\theta \in \Theta} \left( \frac{1}{T} \sum_{t=1}^{T} \hat{\ell}_\theta(\boldsymbol{z}^t, S) - \mathbb{E}_{\boldsymbol{z} \sim \hat{\rho}} \hat{\ell}_\theta(\boldsymbol{z}^t, S) \right), \tag{B.65}$$

where $\boldsymbol{z}^t = (\boldsymbol{x}^t, \boldsymbol{y}^t) \sim \mu_t^n$ and we let $\boldsymbol{Z} = (\boldsymbol{z}^t)_{t=1}^T \sim \hat{\rho}^T$. This enables us to control both $(B)$ involving the ERM parameter $\hat{\theta}$ and $(D)$ involving $\theta^*$. We switch the order of the terms to get the standard form $\sup_{f \in \mathcal{F}} \mathbb{E} f(x) - \frac{1}{n} \sum_{i=1}^n f(x_i)$. We define the loss class $\mathcal{F} = \{f : \mathcal{Z}^n \to \mathbb{R}_{\geq 0},\ f(\boldsymbol{z}) = \hat{\ell}_\theta(\boldsymbol{z}, S),\ \forall \theta \in \Theta\}$ and note that it implicitly depends on the random feature sample $S$.

**Theorem 6.** *For any $\theta \in \Theta$, squared error $\ell(y, \hat{y}) = (y - \hat{y})^2$ such that for all $y \in [0, 1]$, $\ell(\cdot, y) : [-L, L] \to \mathbb{R}_+$ has Lipschitz constant* $\mathrm{Lip}(L)$*, with $\omega$ being KRR with regularization parameter $\lambda > 0$ and RKHS induced by $K_\theta$, with probability greater than $1 - \delta$ over the choice of meta-train set $\overline{\boldsymbol{Z}} \in (\mathcal{Z}^n)^T$ we have*

$$\mathbb{E}_{S \sim \mathcal{N}^M}[\hat{\mathcal{E}}_T(\theta, S) - \hat{\mathcal{E}}(\theta, S)] \leq \mathbb{E}_{S \sim \mathcal{N}^M} \mathbb{E}_{\boldsymbol{Z} \sim \hat{\rho}^T} \mathcal{R}(\mathcal{F}(\boldsymbol{Z})) + \sqrt{\frac{\log(1/\delta)}{2T}}. \tag{B.66}$$

*Proof.* Relating our setting to [Mau16], for some $S$ we let the loss function class be

$$\mathcal{F} = \{f : \mathcal{Z}^n \to \mathbb{R}_{\geq 0},\ f(\boldsymbol{z}) = \hat{\ell}_\theta(\boldsymbol{z}, S),\ \forall \theta \in \Theta\}, \tag{B.67}$$

and for $\overline{\boldsymbol{Z}} \in (\mathcal{Z}^n)^T$, we denote $\Phi(\overline{\boldsymbol{Z}}) = \sup_{f \in \mathcal{F}} \mathbb{E}_{\boldsymbol{Z} \sim \hat{\rho}^T} f(\boldsymbol{Z}) - \frac{1}{T} \sum_{t=1}^T f(\overline{\boldsymbol{Z}}_t)$. By 2. of Thm. 4, since $\hat{\ell}_\theta(\boldsymbol{z}, S) \in [0, 1]$, for any $\delta > 0$ we have with probability greater than $1 - \delta$ over $\overline{\boldsymbol{Z}} \sim \hat{\rho}^T$ that

$$\Phi(\overline{\boldsymbol{Z}}) \leq \mathbb{E}_{\boldsymbol{Z} \sim \hat{\rho}^T} \mathcal{R}(\mathcal{F}(\boldsymbol{Z})) + \sqrt{\frac{\log(1/\delta)}{2T}}, \tag{B.68}$$

so we focus on controlling the Rademacher complexity $\mathbb{E}_{\boldsymbol{Z} \sim \hat{\rho}^T} \mathcal{R}(\mathcal{F}(\boldsymbol{Z}))$. Following [Mau16, Sec. 3.3], note that our notation differs, the translation is as follows ([Mau16, Sec. 3. 3] $\to$ this work) $\mathcal{H} \to \{\psi_{\theta,S}(\cdot),\ \forall \theta \in \Theta\}$,

$\psi_t(\cdot) \to \hat{\ell}_{(\cdot),S}(\boldsymbol{z}^t)$, $\mathbb{E}_{t\sim\rho}[\psi_t(h)] \to \hat{\mathcal{E}}(\theta)$, $\frac{2}{n}R(\mathcal{H}, \bar{\boldsymbol{x}}) \to \mathcal{R}(\mathcal{F}(\overline{\boldsymbol{Z}}))$, $\phi_t(h) \to \Phi_{\theta,S}$, we will specify $\Phi_{\theta,S}$ below. To apply [Mau16, Thm. 2] we need find a "Lipschitz" constant $L$ such that

$$\hat{\ell}_{\theta,S}(\boldsymbol{z}^t) - \hat{\ell}_{\theta',S}(\boldsymbol{z}^t) \leq \frac{L}{\sqrt{n}}\|\Phi_{\theta,S} - \Phi_{\theta',S}\|_F, \tag{B.69}$$

where $(\Phi_{\theta,S})_{:,j} = \phi_{\theta,S}(x_j)$ and $\Phi_{\theta,S} \in \mathbb{R}^{2M\times n}$, e.g. $\Phi_{\theta,S}$ is the feature matrix of the kernel $\boldsymbol{G}_\theta(\boldsymbol{x}, S)$ so that $\boldsymbol{G}_\theta(\boldsymbol{x}, S) = \Phi_{\theta,S}^\top\Phi_{\theta,S}$. We proceed as follows; assume that for loss $\ell(\cdot,\cdot)$ we have kernel stability

$$\hat{\ell}_{\theta,S}(\boldsymbol{z}) - \hat{\ell}_{\theta',S}(\boldsymbol{z}) \leq \frac{L_\ell}{n}\|\boldsymbol{G}_\theta(\boldsymbol{x}, S) - \boldsymbol{G}_{\theta'}(\boldsymbol{x}, S)\|_F, \tag{B.70}$$

this holds true for the least squares loss with $L_\ell = 2\lambda^{-1}$ following similarly from the proof of proposition B.2.1. Since $\Phi_{\theta,S}^\top\Phi_{\theta,S} = \boldsymbol{G}_\theta(\boldsymbol{x}, S)$ for $\theta$ and similarly for $\theta'$ we can write

$$\|\boldsymbol{G}_\theta(\boldsymbol{x}, S) - \boldsymbol{G}_{\theta'}(\boldsymbol{x}, S)\|_F = \left\|\Phi_{\theta,S}^\top\Phi_{\theta,S} - \Phi_{\theta',S}^\top\Phi_{\theta',S}\right\|_F \tag{B.71}$$

and letting $M_{\max} = \max(\|\Phi_{\theta,S}\|_F, \|\Phi_{\theta',S}\|_F)$, using the matrix identity $A^\top A - B^\top B = A^\top(A - B) + (A - B)^\top B$, we can upper bound it

$$\left\|\Phi_{\theta,S}^\top\Phi_{\theta,S} - \Phi_{\theta',S}^\top\Phi_{\theta',S}\right\|_F \leq \|\Phi_{\theta,S}\|_F\|\Phi_{\theta,S} - \Phi_{\theta',S}\|_F + \|\Phi_{\theta',S}\|_F\|\Phi_{\theta',S} - \Phi_{\theta,S}\|_F \tag{B.72}$$

$$\leq 2M_{\max}\|\Phi_{\theta',S} - \Phi_{\theta,S}\|_F. \tag{B.73}$$

Now

$$\|\Phi_{\theta,S}\|_F^2 = \frac{1}{M}\sum_{k=1}^{M}\sum_{j=1}^{n}(\sin(\psi_\theta(s_k)^\top x_j)^2 + \cos(\psi_\theta(s_k)^\top x_j)^2) = \frac{1}{M}\sum_{k=1}^{M}\sum_{j=1}^{n} 1 = n, \tag{B.74}$$

which means that $M_{\max} = \sqrt{n}$ and thus we have the sought Lipschitz property

$$\hat{\ell}_{\theta,S}(\boldsymbol{z}^t) - \hat{\ell}_{\theta',S}(\boldsymbol{z}^t) \leq \frac{L_\ell}{\sqrt{n}} \|\Phi_{\theta',S} - \Phi_{\theta,S}\|_F. \tag{B.75}$$

From this we have that

$$\mathcal{R}(\mathcal{F}(\overline{\boldsymbol{Z}})) \leq \frac{2L_\ell}{T\sqrt{nM}} \left( \mathbb{E}_\epsilon \sup_{\theta \in \Theta} \sum_{t,k,i}^{T,M,n} \epsilon_{tki}(\sin(\psi_\theta(s_k)^\top x_i^t) + \cos(\psi_\theta(s_k)^\top x_i^t)) \right). \tag{B.76}$$

□

## B.2.7 Random Feature Error

In this section we show how to control the term $(F)$ in the bound, the term

$$\mathbb{E}_{S \sim \mathcal{N}^M}[\mathcal{E}(\theta^*, S) - \mathcal{E}(\theta^*)]. \tag{B.77}$$

Remember that for a dataset $\boldsymbol{x} \in \mathcal{X}^n$, random features $S$ and $\theta \in \Theta$, we let $\boldsymbol{g}(x) = (K_\theta(x_i, x))_{i=1}^n$ and similarly $\boldsymbol{g}(x, S) = (K_{\theta,S}(x_i, x))_{i=1}^n$.

We first need some additional results

**Theorem 7** ([Tro19], Theorem 2.1)**.** *Let $A$ be a self-adjoint matrix of size $n$. Given a iid samples $(R_k)_{k=1}^M$ of self-adjoint matrices such that $\mathbb{E}R_1 = A$ and $\|R_1\|_\infty \leq B$. Let $m_2(R_1) = \|\mathbb{E}R_1^2\|_\infty$ and $\bar{R}_M = \frac{1}{M}\sum_{k=1}^M R_M$, then*

$$\mathbb{E}\|\bar{R}_M - A\| \leq \sqrt{\frac{2m_2(R_1)\log(2n)}{M}} + \frac{2B\log(2n)}{3M}. \tag{B.78}$$

**Lemma B.2.2.** *For any $\boldsymbol{x} \in \mathcal{X}^n$, any $\theta \in \Theta$, $\mathbb{E}_{S \sim \mathcal{N}^M}\|\boldsymbol{g}(x) - \boldsymbol{g}(x, S)\|_2 \leq 2n^{1/2}M^{-1/2}$*

*Proof.* We have that

$$\mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{g}(x)-\boldsymbol{g}(x,S)\|_2 \leq \sqrt{\mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{g}(x)-\boldsymbol{g}(x,S)\|_2^2} \tag{B.79}$$

$$= \sqrt{\sum_{i=1}^n \mathbb{E}_{S\sim\mathcal{N}^M}(\boldsymbol{g}(x)_i - \frac{1}{M}\sum_{k=1}^M \boldsymbol{g}(x,s_k)_i)^2}. \tag{B.80}$$

Define $T_{i,k} = K_\theta(x,x_i) - K_{\theta,s_k}(x,x_i)$, then $\mathbb{E}(T_{i,k}) = 0$, for any $i \in [n]$, $(T_{i,k})_{k=1}^M$ is an iid sample of random variables and finally $|T_{i,k}| \leq |K_\theta(x,x_i)| + |K_{\theta,s_k}(x,x_i)| \leq 2$. We can then express

$$\sum_{i=1}^n \mathbb{E}_{S\sim\mathcal{N}^M}(\boldsymbol{g}(x)_i - \frac{1}{M}\sum_{k=1}^M \boldsymbol{g}(x,s_k)_i)^2 = \sum_{i=1}^n \mathbb{E}_{S\sim\mathcal{N}^M}\left(\frac{1}{M}\sum_{k=1}^M T_{i,k}\right)^2 \tag{B.81}$$

$$= M^{-2}\sum_{i=1}^n \mathbb{E}_{S\sim\mathcal{N}^M}\left(\sum_{k=1}^M T_{i,k}\right)^2. \tag{B.82}$$

For arbitrary $i \in [n]$, we see that

$$\mathbb{E}_{S\sim\mathcal{N}^M}\left(\sum_{k=1}^M T_{i,k}\right)^2 \leq \sum_{k=1}^M \mathbb{E}_{s_k\sim\mathcal{N}} T_{i,k}^2 + 2\sum_{k<l}\mathbb{E}_{s_k,s_l\sim\mathcal{N}} T_{i,k}T_{i,l} \tag{B.83}$$

$$= \sum_{k=1}^M \mathbb{E}_{s_k\sim\mathcal{N}} T_{i,k}^2 + 2\sum_{k<l}\mathbb{E}_{s_k\sim\mathcal{N}} T_{i,k}\mathbb{E}_{s_l\sim\mathcal{N}} T_{i,l} = \sum_{k=1}^M \mathbb{E}_{s_k\sim\mathcal{N}} T_{i,k}^2, \tag{B.84}$$

where we used the fact that $T_{i,k}$ is zero-mean and for fixed $i \in [n]$, $T_{i,k}, T_{i,l}$ are independent. Since $|T_{i,k}| \leq 2$ we see that $|T_{i,k}|^2 \leq 4$, hence

$$\mathbb{E}_{S\sim\mathcal{N}^M}\left(\sum_{k=1}^M T_{i,k}\right)^2 \leq 4M. \tag{B.85}$$

Thus we see that

$$\mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{g}(x)-\boldsymbol{g}(x,S)\|_2 \leq \sqrt{\mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{g}(x)-\boldsymbol{g}(x,S)\|_2^2} \leq 2n^{1/2}M^{-1/2}. \tag{B.86}$$

□

Following [Tro19, Section 2.2] we have the following result

**Lemma B.2.3.** *For any $\boldsymbol{x} \in \mathcal{X}^n$, any $\theta \in \Theta$, $\mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{G}(\boldsymbol{x}) - \boldsymbol{G}(\boldsymbol{x}, S)\|_\infty \leq \sqrt{\frac{4\|\boldsymbol{G}(\boldsymbol{x})\|_\infty n\log(2n)}{M}} + \frac{2n\log(2n)}{4M}$*

*Proof.* Note that due to the identity $\cos(x-y) = \sin(x)\sin(y) + \cos(x)\cos(y)$ we have $K_\theta(x,y) = \mathbb{E}_{s\sim\mathcal{N}}\cos(\langle x, \psi_\theta(s)\rangle - \langle y, \psi_\theta(s)\rangle) = \mathbb{E}_{s\sim\mathcal{N}}\phi_{\theta,s}(x)^\top\phi_{\theta,s}(y)$. For the sample $\boldsymbol{x} = (x_i)_{i=1}^n$ let $Z_{k,i,:} = \phi_{\theta,s_k}(x_i)$ be the matrix of features corresponding to sample $s_k$, and let $Z_k^l$ be the $l$'th column of $Z_k$. Thus we have that $\mathbb{E}Z_kZ_k^\top = \boldsymbol{G}(\boldsymbol{x})$ and $\boldsymbol{G}(\boldsymbol{x}, S) = \frac{1}{M}\sum_{k=1}^M Z_kZ_k^\top$.

To put this in the notation of Thm. 7 we let $\bar{R}_M = \boldsymbol{G}(\boldsymbol{x}, S)$ and $A = \boldsymbol{G}(\boldsymbol{x})$. To invoke Thm. 7 need to upper bound the quantities $\|Z_kZ_k\|_\infty$ and $m_2(Z_kZ_k^\top)$. For the first term

$$\begin{aligned}\|Z_kZ_k^\top\|_\infty &\leq \|Z_k\|_\infty\|Z_k\|_\infty \\ &\leq \|Z_k\|_F^2 = \sum_{i=1}^n \cos(\langle x_i, \psi_\theta(s_k)\rangle)^2 + \sin(\langle x_i, \psi_\theta(s_k)\rangle)^2 = n.\end{aligned} \tag{B.87}$$

For the second term, consider $\mathbb{E}(Z_kZ_k^\top)^2$. We can rewrite this in the form $\mathbb{E}Z_kCZ_k^\top$ where $C = Z_k^\top Z_k$ hence symmetric and psd. We can write this as a sum

$$Z_kCZ_k^\top = C_{11}Z_k^1(Z_k^1)^\top + C_{22}Z_k^2(Z_k^2)^\top + C_{12}(Z_k^1(Z_k^2)^\top + Z_k^2(Z_k^1)^\top). \tag{B.88}$$

We can bound $0 \leq C_{11} = \|Z_k^1\|_2^2 \leq n, 0 \leq C_{22} = \|Z_k^2\|_2^2 \leq n$ and $|C_{12}| = |\langle Z_k^1, Z_k^2\rangle| \leq n$. Using the identity $ab^\top + ba^\top = \frac{1}{2}((a+b)(a+b)^\top - (a-b)(a-b)^\top)$ we can then express $Z_kZ_k^\top$ as a sum of four psd matrices (with possibly negative coefficients)

$$\begin{aligned}Z_kZ_k^\top &= C_{11}Z_k^1(Z_k^1)^\top + C_{22}Z_k^2(Z_k^2)^\top \\ &\quad + \frac{C_{12}}{2}(Z_k^1 + Z_k^2)(Z_k^1 + Z_k^2)^\top - \frac{C_{12}}{2}(Z_k^1 - Z_k^2)(Z_k^1 - Z_k^2)^\top,\end{aligned} \tag{B.89}$$

then we see that we can majorize $Z_kZ_k$ by the matrix $nZ_k^1Z_k^1 + nZ_k^2Z_k^2 + \frac{n}{2}(Z_k^1 +$

$Z_k^2)(Z_k^1 + Z_k^2)^\top + \frac{n}{2}(Z_k^1 - Z_k^2)(Z_k^1 - Z_k^2)^\top$ in the Loewner order and expanding this majorant we see that $(Z_k Z_k^\top)^2 \preceq 2n Z_k Z_k^\top$ where we let $\preceq$ be the Loewner order on psd matrices. It follows that $m_2(Z_k Z_k^\top) = \left\|\mathbb{E}(Z_k Z_k^\top)^2\right\|_\infty \leq 2n\left\|\mathbb{E} Z_k Z_k^\top\right\|_\infty = 2n\|\boldsymbol{G}(\boldsymbol{x})\|_\infty$. □

We are now ready to state the theorem of the random feature error

**Theorem 8** (Random feature error)**.** *For any $\theta \in \Theta$, any loss $\ell$ such that for all $y \in [0,1]$, $\ell(\cdot, y) : [-L, L] \to \mathbb{R}_+$ has Lipschitz constant* $\mathrm{Lip}(L)$*, with $\omega$ being KRR with regularization parameter $\lambda > 0$ and RKHS induced by $K_\theta$ we have that*

$$\begin{aligned}
\mathbb{E}_{S\sim\mathcal{N}^M}[\mathcal{E}(\theta, S) - \mathcal{E}(\theta)] &\leq 2\mathrm{Lip}(\lambda^{-1/2})\lambda^{-1}M^{-1/2} && \text{(B.90)}\\
&+ 2\mathrm{Lip}(\lambda^{-1/2})\lambda^{-2}M^{-1/2}n^{-1}\sqrt{\log(2n)\mathbb{E}_{\mu\sim\rho}\mathbb{E}_{\boldsymbol{x}\sim\mu^n}\|\boldsymbol{G}(\boldsymbol{x})\|_\infty} && \text{(B.91)}\\
&+ \frac{1}{2}\mathrm{Lip}(\lambda^{-1/2})\lambda^{-2}M^{-1}n^{-1}\log(2n) && \text{(B.92)}
\end{aligned}$$

*Proof.* For any $\theta$ we can bound

$$\begin{aligned}
&\mathbb{E}_{S\sim\mathcal{N}^M}\mathcal{E}(\theta, S) - \mathcal{E}(\theta)\\
&= \mathbb{E}_{\mu\sim\rho,(\boldsymbol{z},z)\sim\mu^{n+1},S\sim\mathcal{N}^M}(\ell(\langle\omega_{\theta,S}(\boldsymbol{x},\boldsymbol{y}), \phi_{\theta,S}(x)\rangle, y) - \ell(\langle\omega_\theta(\boldsymbol{x},\boldsymbol{y}), \phi_\theta(x)\rangle, y)) \quad \text{(B.93)}\\
&\leq \mathrm{Lip}(\lambda^{-1/2})\mathbb{E}_{\mu\sim\rho,(\boldsymbol{z},z)\sim\mu^{n+1},S\sim\mathcal{N}^M}|\langle\omega_{\theta,S}(\boldsymbol{x},\boldsymbol{y}), \phi_{\theta,S}(x)\rangle - \langle\omega_\theta(\boldsymbol{x},\boldsymbol{y}), \phi_\theta(x)\rangle|.
\end{aligned}$$

Then

$$\begin{aligned}
&\mathbb{E}_{S\sim\mathcal{N}^M}|\langle\omega_{\theta,S}(\boldsymbol{x},\boldsymbol{y}), \phi_{\theta,S}(x)\rangle - \langle\omega_\theta(\boldsymbol{x},\boldsymbol{y}), \phi_\theta(x)\rangle| && \text{(B.94)}\\
&= \mathbb{E}_{S\sim\mathcal{N}^M}\left|\boldsymbol{y}^\top(\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}\boldsymbol{g}(x) - \boldsymbol{G}_\lambda(\boldsymbol{x}, S)^{-1}\boldsymbol{g}(x, S))\right| && \text{(B.95)}\\
&\leq \mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{y}\|\left\|\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}\boldsymbol{g}(x) - \boldsymbol{G}_\lambda(\boldsymbol{x}, S)^{-1}\boldsymbol{g}(x, S)\right\| && \text{(B.96)}\\
&\leq \sqrt{n}\mathbb{E}_{S\sim\mathcal{N}^M}\left\|\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}\boldsymbol{g}(x) - \boldsymbol{G}_\lambda(\boldsymbol{x}, S)^{-1}\boldsymbol{g}(x, S)\right\|. && \text{(B.97)}
\end{aligned}$$

Using the matrix identity $AB - CD = A(B - D) + (A - C)D$, the triangle inequality, together with Lemma B.2.1 we get

$$\begin{aligned}
&\left\|\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}\boldsymbol{g}(x)-\boldsymbol{G}_\lambda(\boldsymbol{x},S)^{-1}\boldsymbol{g}(x,S)\right\| \\
&\leq \left\|\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}\right\|_\infty\|\boldsymbol{g}(x)-\boldsymbol{g}(x,S)\|_2 + \left\|\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}-\boldsymbol{G}_\lambda(\boldsymbol{x},S)^{-1}\right\|_\infty\|\boldsymbol{g}(x,S)\|_2 \qquad \text{(B.98)} \\
&\leq (n\lambda)^{-1}\|\boldsymbol{g}(x)-\boldsymbol{g}(x,S)\|_2 + (n\lambda)^{-2}\|\boldsymbol{G}(\boldsymbol{x})-\boldsymbol{G}(\boldsymbol{x},S)\|_\infty\sqrt{n} \qquad \text{(B.99)}
\end{aligned}$$

and so we consider the terms

$$\mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{g}(x)-\boldsymbol{g}(x,S)\|_2$$

and

$$\mathbb{E}_{S\sim\mathcal{N}^M}\|\boldsymbol{G}(\boldsymbol{x})-\boldsymbol{G}(\boldsymbol{x},S)\|_\infty. \qquad \text{(B.100)}$$

Using Lemma B.2.2 and Lemma B.2.3 we can upper bound (B.99) (together with factor $\sqrt{n}$) as

$$\begin{aligned}
&\sqrt{n}\left\|\boldsymbol{G}_\lambda(\boldsymbol{x})^{-1}\boldsymbol{g}(x)-\boldsymbol{G}_\lambda(\boldsymbol{x},S)^{-1}\boldsymbol{g}(x,S)\right\| \qquad \text{(B.101)} \\
&\leq 2\lambda^{-1}M^{-1/2} + 2\lambda^{-2}M^{-1/2}n^{-1/2}\sqrt{\log(2n)\|\boldsymbol{G}(\boldsymbol{x})\|_\infty} \qquad \text{(B.102)} \\
&+\frac{1}{2}\lambda^{-2}M^{-1}n^{-1/2}\log(2n) \qquad \text{(B.103)}
\end{aligned}$$

The final bound follows by pulling $\mathbb{E}_{\mu\sim\rho}\mathbb{E}_{\boldsymbol{z}\sim\mu^n}$ into the square root using Jensen and multiplying by $\mathrm{Lip}(\lambda^{-1/2})$.

□

Combining the above we have that

**Theorem 9** (IKML Excess risk bound)**.** *Assume that $\mathcal{X}\times\mathcal{Y}\subseteq\mathbb{R}^d\times[0,1]$ and $\ell(y,\hat{y})=(y-\hat{y})^2$. Let $\mathcal{G}_{future}=\{z=(x,y)\mapsto\lambda^{-1/2}\langle v,\phi_{\theta,S}(x)\rangle_{\theta,S} : \|v\|_{\theta,S}\leq 1\}$ and $\mathcal{F}=\{f:\mathcal{Z}^n\to\mathbb{R}_{\geq 0},\ f(\boldsymbol{z})=\hat{\ell}_\theta(\boldsymbol{z},S),\ \forall\theta\in\Theta\}$, and let $\overline{\boldsymbol{Z}}=(\boldsymbol{x}^t,\boldsymbol{y}^t)_{t=1}^T\sim\hat{\rho}^T$ then with probability greater than $1-\delta$ over the sampling* *of* $\overline{\boldsymbol{Z}}$

$$\begin{aligned}
&\mathbb{E}_{S\sim\mathcal{N}^M}[\mathcal{E}(\hat{\theta}, S) - \mathcal{E}(\theta^*)] && \text{(B.104)}\\
&\qquad \leq 2\mathrm{Lip}(\lambda^{-1/2})\mathbb{E}_{S\sim\mathcal{N}^M}\mathbb{E}_{\boldsymbol{z}\sim\hat{\rho}}\mathcal{R}(\mathcal{G}_{future}(\boldsymbol{z})) && \text{(B.105)}\\
&\qquad + \mathbb{E}_{S\sim\mathcal{N}^M}\mathbb{E}_{\boldsymbol{Z}\sim\hat{\rho}^T}\mathcal{R}(\mathcal{F}(\boldsymbol{Z})) + \sqrt{\frac{\log(1/\delta)}{2T}} && \text{(B.106)}\\
&\qquad + \mathbb{E}_{S\sim\mathcal{N}^M}(\hat{\mathcal{E}}_T(\hat{\theta}, S) - \hat{\mathcal{E}}_T(\theta^*, S)) && \text{(B.107)}\\
&\qquad + 2\mathrm{Lip}(\lambda^{-1/2})\lambda^{-1}M^{-1/2} && \text{(B.108)}\\
&\qquad + 2\mathrm{Lip}(\lambda^{-1/2})\lambda^{-2}M^{-1/2}n^{-1/2}\sqrt{\log(2n)\mathbb{E}_{\mu\sim\rho}\mathbb{E}_{\boldsymbol{x}\sim\mu^n}\|\boldsymbol{G}(\boldsymbol{x})\|_\infty} && \text{(B.109)}\\
&\qquad + \frac{1}{2}\mathrm{Lip}(\lambda^{-1/2})\lambda^{-2}M^{-1}n^{-1/2}\log(2n) && \text{(B.110)}
\end{aligned}$$

We note the following

- (B.105) can be replaced by the strictly greater bound

$$2\mathrm{Lip}(\lambda^{-1/2})\lambda^{-1/2}n^{-1/2},$$

- we hypothesise that the term $\mathbb{E}_{S\sim\mathcal{N}^M}\mathbb{E}_{\boldsymbol{Z}\sim\hat{\rho}^T}\mathcal{R}(\mathcal{F}(\boldsymbol{Z}))$ in (B.106) is $O(1/\sqrt{T})$ due to standard form of rademacher complexities of bounded balls in RKHS,
- (B.107) is the optimization error and we assume that this is negligible,
- since we are using the squared loss $\mathrm{Lip}(L) = 2(L+1)$ and thus all terms $\mathrm{Lip}(\lambda^{-1/2}) = 2(\lambda^{-1/2}, +1) = O(\lambda^{-1/2})$.

## B.3 Datasets

### B.3.1 Beijing Air Quality

The Beijing Air Quality dataset [Zha+17] is a time-series dataset measuring air-quality and meterological factors at 12 air-quality monitoring sites. The meterological data for each site is matched with the closest of available weather stations. The data was collected hourly and from the period March 1st, 2013 to February 28th, 2017.

Each datum consists of a timestamp, the site name and various features. We use the feature of interest, “PM2.5” for the fine particulate matter (PM) concentration and remove the features “PM10”, “wd”, “WSPM” since the first one correlates heavily with “PM2.5” and “wd”, “WSPM” since “wd” is the direction of the wind and thus categorical and “WSPM” since this is the wind speed of the direction. This leaves us with 9 features and one output feature.

The dataset was created as follows:

1. Remove any rows with missing entries.
2. For each station, split the time-series into 3 sub-series of 64/16/20% starting at the beginning forming the meta-train, validation and test sets.

Tasks are sampled as follows:

1. Sample a station uniformly at random.
2. Given a train and validation size $n = n_{\text{tr}} + n_{\text{val}}$ sample a contiguous sequence of size $n$ at random from the available starting points. We add a temporal feature $t$ which is just an index from 1 to $n$ to encode temporal dependency local to the task.
3. From this contiguous sequence randomly assign $n_{\text{tr}}$ datapoints to the train set and $n_{\text{val}}$ datapoints to the validation set.

This leaves us with the columns ’SO2’, ’NO2’, ’CO’, ’O3’, ’TEMP’, ’PRES’, ’DEWP’ and ’RAIN’ as features.

### B.3.2 Gas Sensor

The Gas Sensor Modulation dataset [BJM18] is a collection of multivariate timeseries collected in a controlled environment using MOX sensors for CO detection. The sensors output voltage recordings sampled at a frequency of 3.5 Hz.

Each timeseries can be chunked up into contiguous subseries corresponding to experiments by looking at the heating cycle, the end of a cycle which marks a new experiment. We let each subsequence correspond to one task distribution from which we sample $n = n_{\text{tr}} + n_{\text{val}}$ datapoints and permute the indices to make the task into a supervised regression task. The output was chosen to be the 2nd feature 3 timesteps into the future as this seen to vary over the tasks and not directly inferable by one of the other features. In total there are 13 csv files with experiment. Each such file has a set number of experiments after preprocessing, we split each files experiment into 64/16/20% meta-train, validation and test splits.

The dataset was created as follows:

1. All subsequences were extracted by locating the start and end of a heating cycle.
2. All extra features which were not gas sensors were dropped.
3. Output feature isolated and lagged.

Tasks are sampled as follows:

1. For a new task we first sample one of the csv files uniformly at random.
2. From the experiments in this csv file we sample a subsequence uniformly at random which is the task-distribution.
3. Add “$t$” to the features.
4. From this subsequence we sample $n = n_{\text{tr}} + n_{\text{val}}$ points at random which forms out train and validation set.

## B.4 Experimental Results

### B.4.1 Hardware

All of the experiments were run on a single computer with specifications

**CPU**

AMD Ryzen 7 3700X 8-Core Processor

**GPU**

NVIDIA GeForce RTX 2060 SUPER

**RAM**

2x16GB DDR4 Vengeance

### B.4.2 Algorithms

In this section we elaborate on the algorithm used.

MAML [FAL17] parameterize a set of functions $f_\theta : \mathcal{X} \to \mathcal{Y}$, typically a family of neural networks. For a new task $D$ it optimizes the objective $\mathrm{argmin}_\theta \hat{\mathcal{E}}(f_\theta, D^{\mathrm{tr}})$ using gradient descent starting from the hyperparameter $\theta_0$ so that the fine-tuned weight vector is a function of $\theta_0$, $\theta(\theta_0)$. In the outer loop it optimizes the objective $\mathrm{argmin}_{\theta_0} \hat{\mathcal{E}}(f_{\theta(\theta_0)}, D^{\mathrm{val}})$ using gradient descent.

R2D2 [Ber+19] parameterize a feature map $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$ which give rise to a kernel $M_\theta(x, x') = \langle \phi_\theta(x), \phi_\theta(x') \rangle$. The inner algorithm is KRR with $K_\theta$. For a task $D$ it first does KRR in the inner loop giving the KRR estimator $f_\theta = A_{\mathrm{KRR}}(K_\theta, D^{\mathrm{tr}})$ and in the outer loop it optimizes $\mathrm{argmin}_\theta \hat{\mathcal{E}}(f_\theta, D^{\mathrm{val}})$ using gradient descent.

LS Biased Regularization [Den+19] performs biased ridge regression where the functions are given by $f_\theta(x) = \langle \theta, x \rangle$. For the inner algorithm it solves the biased ridge regression problem $\mathrm{argmin}_w \frac{1}{n}\|X\theta - y\|^2 + \lambda\|\theta - \theta_0\|^2$ which has a closed form, see [Den+19]. For a task $D$ the algorithm first finds $\theta(\theta_0)$ using $D^{\mathrm{tr}}$ using biased RR and in the outer loop it optimizes $\mathrm{argmin}_{\theta_0} \hat{\mathcal{E}}(f_{\theta(\theta_0)}, D^{\mathrm{val}})$ using gradient descent.

IKML-MLP is the same as IKML but uses the general random features representation of the kernel $K(x, x') = \int_\Omega \varphi(x, \omega)^\top \varphi(x, \omega) \, \mathrm{d}\tau(\omega)$. In this case, we let $\varphi(\cdot, \omega) : \mathbb{R}^d \to \mathbb{R}^o$ be an MLP with ReLU activation functions, some fixed hidden dimension $h$ and an output dimension $o$. Let $D$ be the size of $\omega$. In this case the feature map is complicated, so we opt for a simpler pushforward to make it easier to train. In particular, we let the pushforward take on a "variational form" by letting the pushforward $\psi_\theta(s) = \psi_{\mu,\sigma}(s) = \mu + \sigma \odot s$ where $s, \mu, \sigma \in \mathbb{R}^D$ and for two matrices $A, B$, $(A \odot B)_{ij} = A_{ij}B_{ij}$ is the Hadamard product. We train using the same procedure as in Alg. 1. Of note is that this can be seen as an ensemble method over R2D2 where instead of ensembling over the learned functions we ensemble over kernel functions.

### B.4.3 Toy Regression

#### B.4.3.1 Setup

We create a synthetic high-dimensional meta-learning regression setting where each task is sampled from an RKHS $\mathcal{H}$ with a "complicated" kernel $K^o$. In particular, we choose $K^o$ to be the kernel given by Bochner's theorem and a pushforward of a 3-layers Multi-Layer Perceptron (MLP) with 32 hidden units per layer, ReLU activation functions and a 16-dimensional latent Gaussian distribution. The network was initialized with weights given by the PyTorch [Pas+19] default initialization scaled by 100. Since this kernel lacks an analytic form, we sample 10000 frequencies and use the random features kernel in its place.

The tasks are generated in $\mathcal{H}$ by independently sampling $R$ points $(x_r)_{r=1}^R$ with $x_r \sim U_{[0,0.2]^d}$ and $R$ coefficients $(\alpha_r)_{r=1}^R$ with $\alpha_j \sim U_{[0,1]}$. Together they model the target regressor as $f(x) = \sum_{r=1}^R \alpha_r K(x, x_r)$. We set $R = 3$. The task datasets is created by independently sampling $n = n_{\text{tr}} + n_{\text{val}} = 50 + 50$ datapoints $(x_i, y_i)_{i=1}^n$ with $x_i \sim U_{[0,0.2]^d}$ and $y_i = f(x_i)$.

#### B.4.3.2 Initial and Learned Kernel

For the same setup of the environment as in the synthetic experiments we look at how the initial and learned kernels differ from the true kernel. We do this for the algorithms *IKML* and *Gaussian MKL meta-KRR*. These algorithms where chosen since they define translation invariant kernels and are easy to visualize. We let all of the algorithms (R2D2, MAML, IKML) be parameterized by a 3-layer MLP but with varying the dimensionality of $x$. We also tried using 1 and

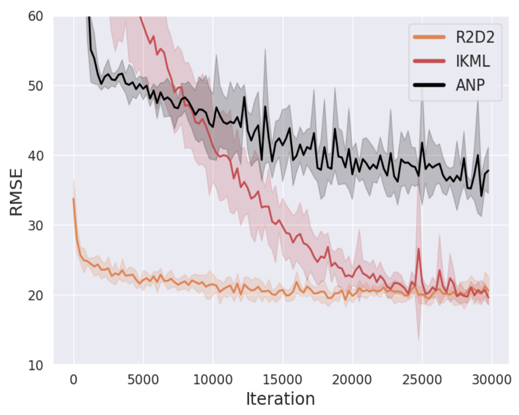
2-layer MLPs for the parameterization but the results were almost identical.

For an experiment with dimension $d$ we visualize the kernels of Gaussian MKL meta-KRR and IKML by sampling 5 directions $(v_i)_{i=1}^5$ from the unit ball in $\mathbb{R}^d$. For a direction $v_i$ we plot the value of the kernel on the line with direction $v_i$ where on the $x$-axis we have $t$ from $-0.4$ to $0.4$ and on the $y$-axis the value of $K(0, t \cdot v_i)$. We plot the result of the first run for each experiment, other runs look similar.

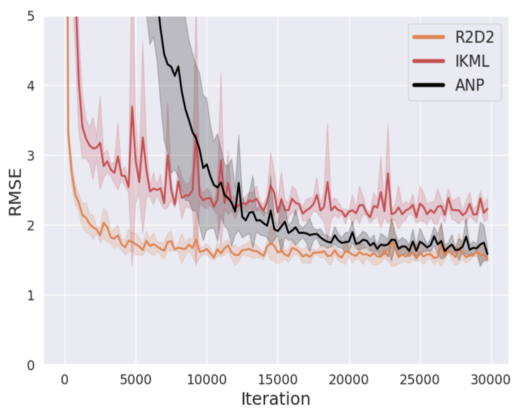
We plot the learning curves and kernel for each dimension 1, 5, 10, 20. For each row in Fig. B.2 corresponding to a dimension $d$ the $i$'th column plots kernels in the direction of $v_i$ with the first row of the subplot corresponding to the kernels at initialization and the second row the kernels after training. The sample $(v_i)_{i=1}^5$ is resampled for each dimension. For IKML we sample 10000 frequencies once and fix them before plotting.

## B.4.4 Learning Curves

We show the behaviour of the optimization trajectory of the algorithms R2D2, IKML and ANP. See Fig. B.1a and Fig. B.1b.



(a) (a)



(b) (b)

Figure B.1: Learning curves of meta-validation RMSE for the algorithms IKML, R2D2 and ANP for (a) Beijing Air Quality (25-shot), (b) Gas Sensor dataset (20-shot) over 5 runs (mean ± 1 std). R2D2 and ANP were chosen due to their recency and performance as few-shot learning algorithms compared to all other algorithms evaluated.

### B.4.5 Cross-Validation for Real-World Datasets

We cross-validated R2D2, IKML and ANP on the 25-shot Air Quality and 20-shot Gas Sensor dataset where we do a grid search over the number of hidden layers in an MLP with ReLU activation function and the meta-learning rate. For IKML and R2D2 the number of hidden layers are in $\{1, 2, 3, 4, 6, 8, 10, 15, 20\}$ while for ANP we use the same architecture for encoder and decoder and use $\{1, 2, 3, 4\}$ layers, the hidden dimension is fixed to 64, the meta-learning rate are in $\{10^{-4}, 10^{-5}\}$. The training setup is the same as in the main body and the metric is the RMSE on a holdout-set sampled from the meta-validation split of the best model from the snapshots during the 30000 iterations. The results can be seen in Table B.1.

Table B.1: Validation results for meta-hyperparameter configurations for IKML, R2D2 [Ber+19] and ANP [Kim+19] on 25-shot Air Quality dataset and 20-shot Gas Sensor. Best set of parameters in **bold**. We run the algorithms for 30000 iterations and evaluate it on the validation set at 250 intervals. We get the validation RMSE on a holdout set (3000 tasks) using the model with the lowest evaluation validation error.

| | | 25-shot Air Quality | | | 20-shot Gas Sensor | | |
|---|---|---|---|---|---|---|---|
| Meta-lr | Layers | IKML | R2D2 | ANP | IKML | R2D2 | ANP |
| $10^{-4}$ | 1 | 101.65 | 8861.31 | 1390.14 | 2.16 | 2.64 | 2.38 |
| | 2 | 98.37 | 13761.01 | 38.61 | 2.14 | 1.85 | 1.72 |
| | 3 | 98.07 | 205.06 | **38.37** | 2.14 | 1.65 | **1.53** |
| | 4 | 21.45 | 508.55 | 36.32 | 2.11 | 1.49 | 1.57 |
| | 6 | 24.24 | 21.57 | – | 2.13 | **1.46** | – |
| | 8 | 23.88 | 21.96 | – | **2.06** | 1.53 | – |
| | 10 | 27.30 | **21.32** | – | 2.06 | 1.49 | – |
| | 15 | 27.57 | 22.85 | – | 2.12 | 1.48 | – |
| | 20 | 40.57 | 25.01 | – | 7.20 | 1.50 | – |
| $10^{-5}$ | 1 | 125.75 | 3237.35 | 76.50 | 19.53 | 6.45 | 7.83 |
| | 2 | 110.01 | 1233.41 | 41.75 | 2.70 | 3.20 | 8.43 |
| | 3 | 76.58 | 431.61 | 47.24 | 2.50 | 2.34 | 7.42 |
| | 4 | **19.05** | 57.37 | 43.71 | 2.41 | 1.86 | 6.35 |
| | 6 | 20.52 | 22.68 | – | 2.43 | 1.59 | – |
| | 8 | 23.86 | 21.98 | – | 2.35 | 1.55 | – |
| | 10 | 134.89 | 22.44 | – | 2.45 | 1.53 | – |
| | 15 | 28.40 | 24.80 | – | 2.46 | 1.56 | – |
| | 20 | 135.18 | 26.62 | – | 2.45 | 1.55 | – |

### B.4.6 More Shots

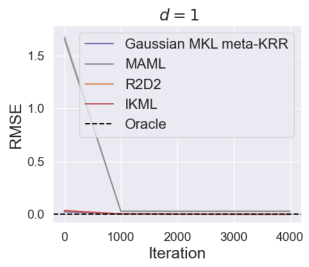
Further test-RMSE for various numbers of shots for Air Quality Table B.2 and Gas Sensor Table B.3. We benchmark LS Biased Regularization, IKML, R2D2, ANP for both Air Quality and Gas Sensor and additionally IKML-MLP for Gas Sensor. We reuse the cross-validated models for IKML, R2D2 and ANP and the hyperparameters used for the other models. We get 5 test-RMSE scores for Air Quality and 2 for Gas Sensor and report the mean and standard deviation for Air Quality and mean for Gas Sensor. The low number of shots in Gas Sensor is due to many of the underlying time series from which each task is generated having as few as 40 points.

Table B.2: Test-RMSE (mean ± 1 std) for IKML, R2D2, ANP and LSBR over 5 independent runs on Air Quality for various shots. Same tasks for all algorithms over each run. Best result for each shot in **bold**.

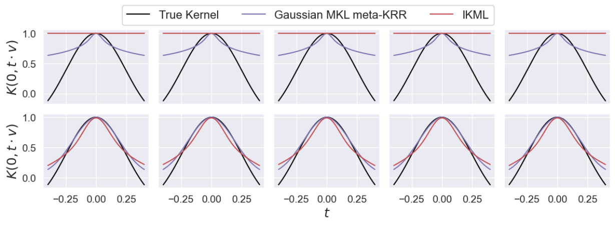
| | Air Quality (shots) | | | |
|---|---|---|---|---|
| Model | 10 | 25 | 50 | 100 |
| IKML | $24.32 \pm 5.21$ | $\mathbf{19.14 \pm 0.93}$ | $\mathbf{19.36 \pm 1.02}$ | $\mathbf{18.88 \pm 0.51}$ |
| R2D2 | $\mathbf{21.21 \pm 0.28}$ | $20.23 \pm 0.55$ | $23.42 \pm 3.44$ | $20.75 \pm 0.79$ |
| ANP | $31.05 \pm 0.90$ | $33.77 \pm 0.70$ | $37.30 \pm 0.94$ | $41.08 \pm 1.07$ |
| LSBR | $21.49 \pm 0.40$ | $21.68 \pm 0.29$ | $23.69 \pm 0.47$ | $27.32 \pm 0.16$ |

Table B.3: Test-RMSE (mean, standard deviation left out due to low number of runs) for IKML, R2D2, ANP, LSBR and IKML-MLP over 2 independent runs on Gas Sensor for various shots. Same tasks for all algorithms over each run. Best result for each shot in **bold**.

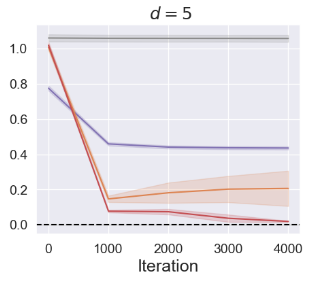
| | Gas Sensor (shots) | | | |
|---|---|---|---|---|
| Model | 5 | 10 | 15 | 20 |
| IKML | 10.04 | 4.57 | 3.42 | 2.80 |
| R2D2 | 6.00 | **2.44** | 2.12 | 1.95 |
| ANP | **2.57** | **2.44** | **2.10** | 2.12 |
| LSBR | 13.97 | 12.21 | 11.12 | 12.44 |
| IKML-MLP | 4.03 | 2.64 | 2.23 | **1.94** |



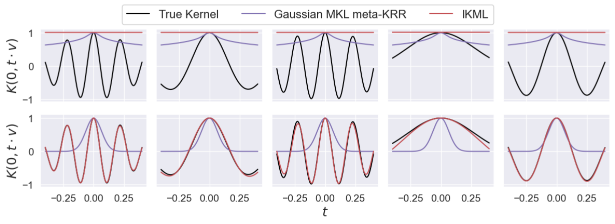
(a) Learning curves for $d = 1$



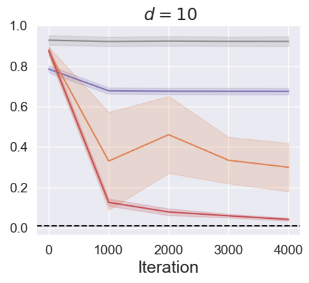
(b) Initial and learned kernels for $d = 1$



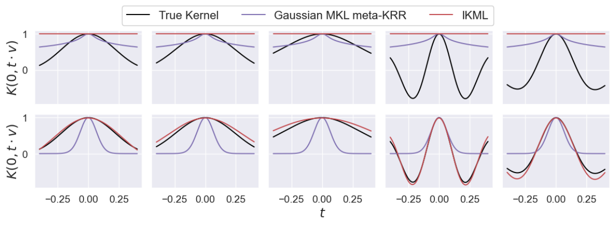
(c) Learning curves for $d = 5$



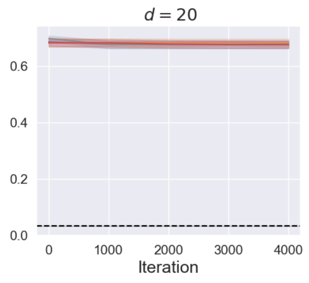
(d) Initial and learned kernels for $d = 5$



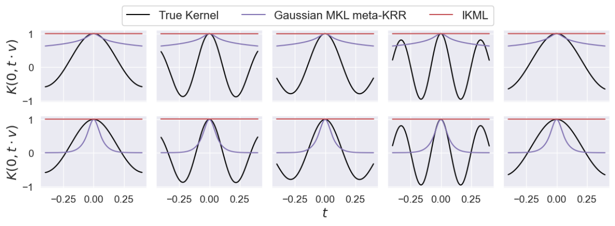
(e) Learning curves for $d = 10$



(f) Initial and learned kernels for $d = 10$



(g) Learning curves for $d = 20$



(h) Initial and learned kernels for $d = 20$

Figure B.2: Parameterization using a 3-layer MLP: Learning curves and initial vs learned kernels for different input dimension on synthetic dataset (all algorithms using a 3-layer MLP). **Column 1**: learning curves (meta-test RMSE) over 3 runs. **Column 2**: **Sub-row 1** kernel before training, **Sub-row 2** kernel at test time. Each column plots the kernel in a random direction drawn from the unit ball.

### B.4.7 Sensitivity of R2D2 and IKML-MLP

We highlight the qualitative difference between R2D2 and IKML-MLP. We compare the learning curves and holdout meta-valid and test RMSE. Note that the test-split is used to asses out-of-sample few-shot performance since we train and choose best model using train and validation set respectively. In this case we let the feature map $\varphi(x, \omega)$ be an MLP with ReLU activation functions, with number of layers and number of hidden units varied. We perform this analysis on the Air Quality and Gas Sensor datasets with the settings as given in the main body unless specified and compare the results.

#### B.4.7.1 Air Quality

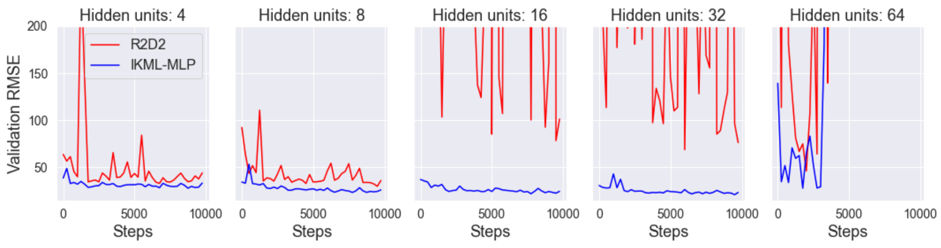
We run IKML-MLP and R2D2 for 10000 iterations using Adam with meta-learning rate $3 \cdot 10^{-4}$ and vary the number of layers and and the number of hidden units in isolation. Both of the algorithms share the same base feature map $\varphi(\cdot, \omega)$ but IKML-MLP calculates the kernel $K(x, x') = \int_\Omega \varphi(x, \omega)^\top \varphi(x, \omega) \, \mathrm{d}\tau(\omega)$ by sampling while R2D2 has a fixed feature map yielding the kernel $K(x, x') = \varphi(x, w)^\top \varphi(x', w)$ for a fixed weight $w$. We use a feature dimension of 8. The only difference to the setup in the main body is that we use 10000 iterations insted of 30000.

When we fix the number of layers to be 2, we can see from Fig. B.3 that IKML-MLP dominates R2D2 in terms of performance both on the validation and test set. In contrast, when we fix the number of hidden units to be 64 and vary the number of layers, we can see from Fig. B.4 that R2D2 performs equally well as IKML-MLP. As the network becomes deeper we noticed, for this dataset, that IKML-MLP requires more random features to train well (in contrast to the Gas Sensor case). We hypothesis that for noisy tasks, like in the Air Quality dataset, the number of random features required to get accurate gradients to be able to train deeper networks increase quickly with depth. However, on this dataset we see that the number of layers is not required to be very deep to reach good performance, so in this case it does not pose a problem.

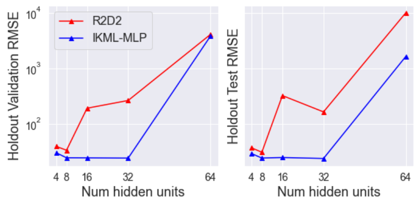
#### B.4.7.2 Gas Sensor

We run IKML-MLP and R2D2 for 10000 iterations using Adam with meta-learning rate $3 \cdot 10^{-4}$ and vary the number of layers and and the number of hidden units in isolation. Both of the algorithms share the same base feature map $\varphi(\cdot, \omega)$ but IKML-MLP calculates the kernel $K(x, x') = \int_\Omega \varphi(x, \omega)^\top \varphi(x, \omega) \, \mathrm{d}\tau(\omega)$ by sampling while R2D2 has a fixed feature map yielding the kernel $K(x, x') = \varphi(x, w)^\top \varphi(x', w)$ for a fixed weight $w$. We use a feature dimension of 4. The only difference to the setup in the main body is that we use 10000 iterations insted of 30000.

Compared to the Air Quality figures Fig. B.3 and B.4 training is much more stable due to the dataset being much less noisy than that of the Air Quality dataset. R2D2 and IKML-MLP both train well and have good performance, although IKML-MLP overfits less to the meta-split as can be seen on the holdout performance plots in Fig. B.5 and B.6. In this case 100 random features were enough for the training to be successful for IKML-MLP which supports the hypothesis given in the previous section on Air Quality.

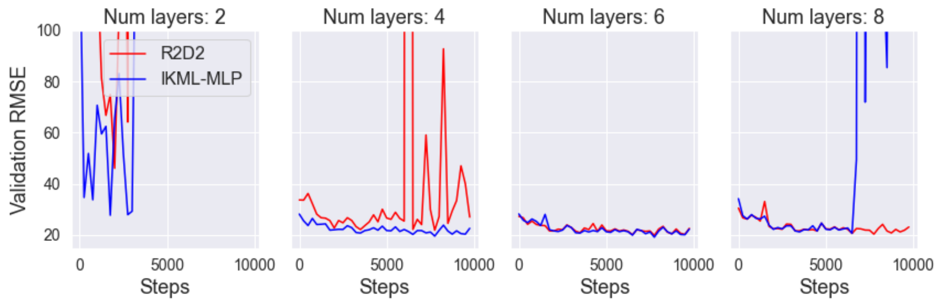


(a) Learning curves

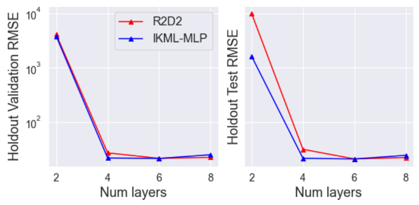


(b) Holdout Performance

Figure B.3: Learning curves (above) and performance plots (below) of R2D2 vs IKML-MLP on the Air Quality dataset when varying the number of hidden units. IKML-MLP is more robust to hyperparameter settings than R2D2. We fix the number of hidden layers to 2 and feature dimension to be 8. For IKML-MLP we fix the number of random features to be 100. Note that performance plot is log-scaled due to large range of reported numbers.

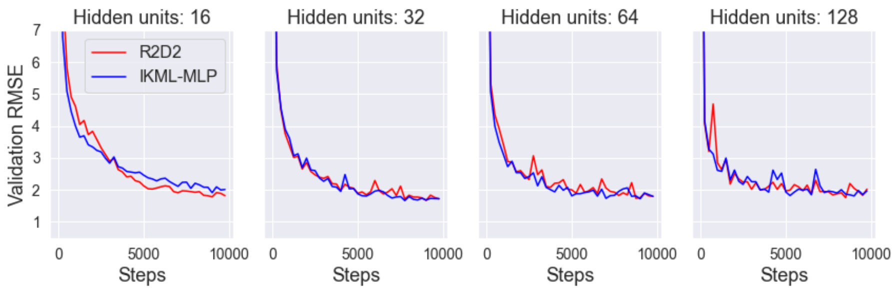


(a) Learning curves: we optimize the models using the train split

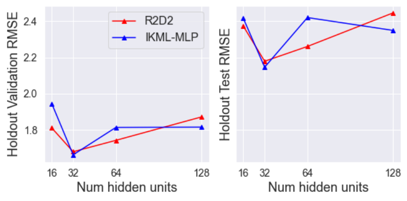


(b) Holdout Performance

Figure B.4: Learning curves (above) and performance plots (below) of R2D2 vs IKML-MLP on the Air Quality dataset when varying the number of layers. IKML-MLP stabilize training up to a point but for deeper networks we found that IKML-MLP requires more random features. We fix the number of hidden units to 64 and feature dimension to be 8. For IKML-MLP we fix the number of random features to be 400.

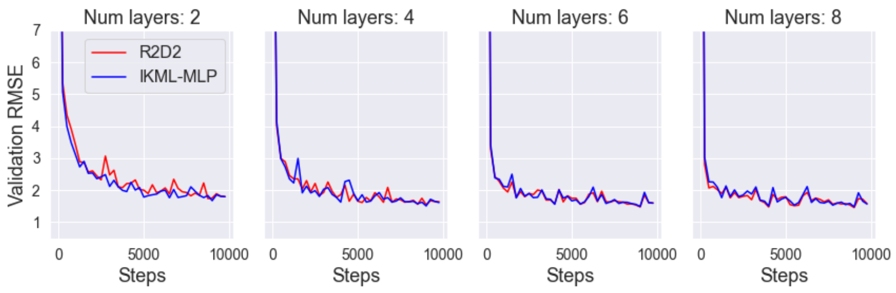


(a) Learning curves

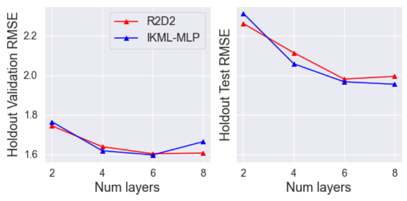


(b) Holdout Performance

Figure B.5: Learning curves (above) and performance plots (below) of R2D2 vs IKML-MLP on the Gas Sensor dataset when varying the number of hidden units. IKML-MLP is more robust to hyperparameter settings than R2D2. We fix the number of hidden layers to 2 and feature dimension to be 8. For IKML-MLP we fix the number of random features to be 100.



(a) Learning curves: we optimize the models using the train split



(b) Holdout Performance

Figure B.6: Learning curves (above) and performance plots (below) of R2D2 vs IKML-MLP on the Gas Sensor dataset when varying the number of layers. IKML-MLP stabilize training up to a point but for deeper networks we found that IKML-MLP requires more random features. We fix the number of hidden units to 64 and feature dimension to be 8. For IKML-MLP we fix the number of random features to be 100.

## B.5 Computational Complexity and Walltime Table

In this section we show the computational complexity using big-$O$ notation of IKML / IKML-MLP and compare it against that of R2D2 since they both rely on KRR as the inner algorithm. In addition we measure the performance in practice through wall-time table of IKML, IKML-MLP, MAML and R2D2.

We first recall the complexity of training and validation of KRR in the dual form when we have a train set $D^{\text{tr}} = (x_i, y_i)_{i=1}^{n_{\text{tr}}}$ and a validation set $D^{\text{val}} = (x_j, y_j)_{j=1}^{n_{\text{val}}}$. We focus on the dual formulation since generally data set sizes are small in meta-learning while the feature space dimension is large, which means that the dual form is more efficient than the primal form.

Assume that we have a kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that can be evaluated in $O(\kappa)$. For training we need to calculate the dual coefficients $\alpha = (G + n_{\text{tr}}\lambda I)^{-1}y$ where $y$ is the output vector. This means we first need to calculate the regularized kernel matrix of the train set, $G + n_{\text{tr}}\lambda I \in \mathbb{R}^{n_{\text{tr}} \times n_{\text{tr}}}$, which can be calculated in $O(\kappa n_{\text{tr}}^2 + n_{\text{tr}})$ since $n_{\text{tr}}\lambda I$ is a diagonal matrix, then invert this matrix which can be calculated in $O(n_{\text{tr}}^3)$ and finally perform the matrix-vector multiplication which is $O(n_{\text{tr}}^2)$. Summing all of the steps gives a final complexity of $O(\kappa n_{\text{tr}}^2 + n_{\text{tr}}^3)$. Prediction on the validation set $D^{\text{val}}$ means first calculating the matrix $(M_{ls})_{l,s=1}^{n_{\text{val}}, n_{\text{tr}}} = (K(x_l, x_s))_{l,s=1}^{n_{\text{val}}, n_{\text{tr}}}$ between the validation and train set which is done in $O(\kappa n_{\text{tr}} n_{\text{val}})$. After calculating $M$, calculating $\hat{y} = M\alpha$ can be done in $O(n_{\text{tr}} n_{\text{val}})$ which means that the total number of operations is $O((\kappa + 1)n_{\text{tr}} n_{\text{val}})$.

The complexity for both training and validation when using KRR depends implicitly on $\kappa$ which will depend on the meta-learning algorithm. For IKML with Bochner kernel using $M$ random features we first need to sample $M$ features. This can be done in $O(Cm)$ where $C$ is the time it takes to evaluate the pushforward neural network. Note that this is a one-time cost before training and validation. In practice we use batches so that for $B$ tasks we sample $M$ features once, which reduces this cost further by a factor of the

Table B.4: *Time (seconds) to solve one batch of tasks for IKML, IKML-MLP, R2D2 and MAML.* We measure the time required for solving one batch of tasks: training, calculating the meta-loss, and updating the hyperparameters. We use the Air Quality ($d = 10$) dataset with 25 train and 25 validation points per task, meta-batch size of 4. All algorithms use a 4-layer MLP with 64 hidden units. For IKML we let $M = 2 \cdot 10^4$, while for IKML-MLP we let $M = 100$. We run each algorithm for 5000 steps and normalize the total time by dividing with 5000. We repeat this 3 times and report the mean and standard deviation.

| Algorithm | Seconds for one batch (mean $\pm$ 1 std) |
|---|---|
| IKML | $0.017 \pm 0.00004$ |
| IKML-MLP | $0.075 \pm 0.002$ |
| R2D2 | $0.031 \pm 0.001$ |
| MAML | $0.022 \pm 0.001$ |

number of tasks in a batch. Letting $W \in \mathbb{R}^{M \times d}$ be the matrix of random features stacked horizontally then the feature map $\phi : \mathcal{X} \to \mathbb{R}^M$ is $\phi(x) = \cos(Wx + b)$ where $b$ is a vector of iid entries sampled uniformly from $[0, 2\pi]$, sampled at the same time as $W$. Evaluating $\phi$ once is done in $O((d+1)m)$. For one task, we first calculate the $M$ features in $O((d+1)m)$ and training This means that training and prediction for IKML costs $O(dmn_{\mathrm{tr}}^2 + n_{\mathrm{tr}}^3)$ and $O(dmn_{\mathrm{tr}}n_{\mathrm{val}})$ respectively, both which are linear in $M$.

For R2D2 the feature map $\phi : \mathcal{X} \to \mathbb{R}^h$ where $h$ is the dimension of the feature space is a neural network. Assuming that $\phi$ takes the form of an $L$-layer MLP with weights and biases $(W_i, b_i)_{i=1}^T$ such that $W_1 \in \mathbb{R}^{h_1 \times d}, b_1 \in \mathbb{R}^{h_1}$, and for $1 < l < T$, $W_l \in \mathbb{R}^{h_l \times h_{l-1}}, b_l$ and finally $W_L \in \mathbb{R}^{h \times h_l}, b_l \in \mathbb{R}^h$ with nonlinearity $\sigma$ which can be evaluated in constant time $A$, then evaluating $\phi(x)$ is done in $O(\prod_{l=1}^L h_l h_{l-1} + \sum_{l=1}^{L-1}(1+A)h_l + h_L) = O(\prod_{l=1}^L h_l h_{l-1})$. Running IKML-MLP, if $h_l = h$ for any $l$ we get $O(dh^{2L-1})$. Except for the extra factor of $h^{2L-1}$ the same conclusion as for Bochner holds in this case.

# Bibliography

[Wan+22] Ruohan Wang, Isak Falk, Massimiliano Pontil, and Carlo Ciliberto. *Robust Meta-Representation Learning via Global Label Inference and Classification.* Dec. 2022. DOI: 10.48550/arXiv.2212.11702.

[FCP22] John Isak Texas Falk, Carlo Cilibert, and Massimiliano Pontil. "Implicit Kernel Meta-Learning Using Kernel Integral Forms". In: *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence.* PMLR, Aug. 2022, pp. 652–662.

[Len+20] Daniel Lengyel, Janith Petangoda, Isak Falk, Kate Highnam, Michalis Lazarou, Arinbjörn Kolbeinsson, Marc Peter Deisenroth, and Nicholas R Jennings. "Genni: Visualising the geometry of equivalences for neural network identifiability". In: *arXiv preprint arXiv:2011.07407* (2020).

[Gra+22] Riccardo Grazzi, Arya Akhavan, Isak John Falk, Leonardo Cella, and Massimiliano Pontil. "Group Meritocratic Fairness in Linear Contextual Bandits". In: *Advances in Neural Information Processing Systems.* Oct. 2022.

[Fal+23a] Isak Falk, Millie Zhao, Juba Nait Saada, and Qi Guo. "Learning the Kernel for Rare Variant Genetic Association Test". In: *Frontiers in Genetics* 14 (2023), p. 1245238.

[Fal+23b] John Isak Texas Falk, Luigi Bonati, Pietro Novelli, Michele Parrinello, and massimiliano Pontil. "Transfer learning for atomistic simulations using GNNs and kernel mean embeddings". In: *Thirty-seventh Conference on Neural Information Processing Systems.* 2023.

[Ber+19] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. "Meta-Learning with Differentiable Closed-Form Solvers". In: *International Conference on Learning Representations.* 2019.

[Kim+19] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. "Attentive Neural Processes". In: *International Conference on Learning Representations.* 2019.

[Gor+08] Ian Gorton, Paul Greenfield, Alex Szalay, and Roy Williams. "Data-Intensive Computing in the 21st Century". In: *Computer* 41.4 (Apr. 2008), pp. 30–32. ISSN: 1558-0814. DOI: 10.1109/MC.2008.122.

[Vap91] V. Vapnik. "Principles of Risk Minimization for Learning Theory". In: *Advances in Neural Information Processing Systems.* Vol. 4. Morgan-Kaufmann, 1991.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge: Cambridge University Press, 2014. ISBN: 978-1-107-05713-5. DOI: 10.1017/CBO9781107298019.

[CD07] A. Caponnetto and E. De Vito. "Optimal Rates for the Regularized Least-Squares Algorithm". In: *Foundations of Computational Mathematics* 7.3 (July 2007), pp. 331–368. ISSN: 1615-3383. DOI: 10.1007/s10208-006-0196-8.

[HK19] Steve Hanneke and Aryeh Kontorovich. "Optimality of SVM: novel proofs and tighter bounds". In: *Theoretical Computer Science* 796 (2019), pp. 99–113.

[VD02] Ricardo Vilalta and Youssef Drissi. “A Perspective View and Survey of Meta-Learning”. In: *Artificial Intelligence Review* 18.2 (June 2002), pp. 77–95. ISSN: 1573-7462. DOI: 10.1023/A:1019956318069.

[Hos+22] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. “Meta-Learning in Neural Networks: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9 (Sept. 2022), pp. 5149–5169. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3079209.

[RCR17] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. “FALKON: An Optimal Large Scale Kernel Method”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[Car97] Rich Caruana. “Multitask Learning”. In: *Machine Learning* 28.1 (July 1997), pp. 41–75. ISSN: 1573-0565. DOI: 10.1023/A:1007379606734.

[Rud17] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. June 2017. DOI: 10.48550/arXiv.1706.05098.

[EP04] Theodoros Evgeniou and Massimiliano Pontil. “Regularized Multi–Task Learning”. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’04. New York, NY, USA: Association for Computing Machinery, Aug. 2004, pp. 109–117. ISBN: 978-1-58113-888-7. DOI: 10.1145/1014052.1014067.

[Bax98] Jonathan Baxter. “Theoretical Models of Learning to Learn”. In: *Learning to Learn*. Ed. by Sebastian Thrun and Lorien Pratt. Boston, MA: Springer US, 1998, pp. 71–94. ISBN: 978-1-4615-5529-2. DOI: 10.1007/978-1-4615-5529-2_4.

[Gre22] Shannon Greenwood. *3. Internet, Smartphone and Social Media Use*. Dec. 2022.

[Bax00] J. Baxter. “A Model of Inductive Bias Learning”. In: *Journal of Artificial Intelligence Research* 12 (Mar. 2000), pp. 149–198. ISSN: 1076-9757. DOI: 10.1613/jair.731.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, July 2017, pp. 1126–1135.

[TP98] Sebastian Thrun and Lorien Pratt. “Learning to Learn: Introduction and Overview”. In: *Learning to Learn*. Ed. by Sebastian Thrun and Lorien Pratt. Boston, MA: Springer US, 1998, pp. 3–17. ISBN: 978-1-4615-5529-2. DOI: 10.1007/978-1-4615-5529-2_1.

[Dhi+20] Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. “A Baseline for Few-Shot Image Classification”. In: *International Conference on Learning Representations*. Jan. 2020.

[RL17] Sachin Ravi and Hugo Larochelle. “Optimization as a Model for Few-Shot Learning”. In: *International Conference on Learning Representations*. July 2017.

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[pap22] paperswithcode. *5way-1shot mini-ImageNet state-of-the-art evolution*. https://paperswithcode.com/sota/few-shot-image-classification-on-mini-2. [Online; accessed November 17, 2022]. 2022.

[Vin+16] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. “Matching Networks for One Shot Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[SSZ17] Jake Snell, Kevin Swersky, and Richard Zemel. “Prototypical Networks for Few-shot Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[Rus+18] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. “Meta-Learning with Latent Embedding Optimization”. In: *International Conference on Learning Representations*. Jan. 2018.

[Lee+19] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. “Meta-Learning With Differentiable Convex Optimization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10657–10665.

[Hu+22] Shell Xu Hu, Da Li, Jan Stühmer, Minyoung Kim, and Timothy M Hospedales. “Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9068–9077.

[Tia+20] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. “Rethinking Few-Shot Image Classification: A Good Embedding Is All You Need?” In: *Computer Vision ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 266–282. ISBN: 978-3-030-58568-6. DOI: 10.1007/978-3-030-58568-6_16.

[Dum+21] Vincent Dumoulin, Neil Houlsby, Utku Evci, Xiaohua Zhai, Ross Goroshin, Sylvain Gelly, and Hugo Larochelle. “A Unified Few-

Shot Classification Benchmark to Compare Transfer and Meta Learning Approaches". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* 1 (Dec. 2021).

[Rag+20] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML". In: *International Conference on Learning Representations*. Sept. 2020.

[WPC21] Ruohan Wang, Massimiliano Pontil, and Carlo Ciliberto. "The Role of Global Labels in Few-Shot Classification and How to Infer Them". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 27160–27170.

[YRC07] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. "On Early Stopping in Gradient Descent Learning". In: *Constructive Approximation* 26.2 (Aug. 2007), pp. 289–315. ISSN: 1432-0940. DOI: 10.1007/s00365-006-0663-2.

[Vap00] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. New York, NY: Springer, 2000. DOI: 10.1007/978-1-4757-3264-1.

[HK70] Arthur E. Hoerl and Robert W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems". In: *Technometrics* 12.1 (Feb. 1970), pp. 55–67. ISSN: 0040-1706. DOI: 10.1080/00401706.1970.10488634.

[SS01] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[Aro50] Nachman Aronszajn. "Theory of Reproducing Kernels". In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.

[CS02] Felipe Cucker and Steve Smale. “On the Mathematical Foundations of Learning”. In: *Bulletin of the American Mathematical Society* 39 (2002), pp. 1–49.

[GA11] Mehmet Gönen and Ethem Alpaydn. “Multiple Kernel Learning Algorithms”. In: *The Journal of Machine Learning Research* 12 (2011), pp. 2211–2268.

[Li+19] Chun-Liang Li, Wei-Cheng Chang, Youssef Mroueh, Yiming Yang, and Barnabas Poczos. “Implicit Kernel Learning”. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2019, pp. 2007–2016.

[SD16] Aman Sinha and John C Duchi. “Learning kernels with random features”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1298–1306.

[Zha+19] Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. “Variational Few-Shot Learning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1685–1694.

[LCd19] Pierre Laforgue, Stephan Clémençon, and Florence d’Alché-Buc. “Autoencoding any data through kernel autoencoders”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 1061–1069.

[ZY21] Yu Zhang and Qiang Yang. “A survey on multi-task learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2021), pp. 5586–5609.

[AEP08] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. “Convex Multi-Task Feature Learning”. In: *Machine learning* 73.3 (2008), pp. 243–272.

[AEP07] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. "Multi-task feature learning". In: *Advances in neural information processing systems*. 2007, pp. 41–48.

[HLF19] Kyle Hsu, Sergey Levine, and Chelsea Finn. "Unsupervised Learning via Meta-Learning". In: *International Conference on Learning Representations*. Jan. 2019.

[Hew+18] Luke B Hewitt, Maxwell I Nye, Andreea Gane, Tommi Jaakkola, and Joshua B Tenenbaum. "The Variational Homoencoder: Learning to Learn High Capacity Generative Models from Few Examples". In: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*. 2018.

[Den+18] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. "Learning To Learn Around A Common Mean". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[Fra21] Luca Franceschi. "A Unified Framework for Gradient-based Hyperparameter Optimization and Meta-learning". Doctoral. UCL (University College London), June 2021. DOI: 10/1/Thesis_Luca_Franceschi_UCL.pdf.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[Bay+18] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic differentiation in machine learning: a survey". In: *Journal of Machine Learning Research* 18 (2018).

[BCN18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. “Optimization Methods for Large-Scale Machine Learning”. In: *SIAM Review* 60.2 (Jan. 2018), pp. 223–311. ISSN: 0036-1445. DOI: 10.1137/16M1080173.

[FFP06] Li Fei-Fei, Rob Fergus, and Pietro Perona. “One-shot learning of object categories”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.4 (2006), pp. 594–611.

[Fra+18] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. “Bilevel Programming for Hyperparameter Optimization and Meta-Learning”. In: *International Conference on Machine Learning*. 2018, pp. 1568–1577.

[Ye+20] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. “Few-shot learning via embedding adaptation with set-to-set functions”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8808–8817.

[Zha+20] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. “DeepEMD: Few-Shot Image Classification With Differentiable Earth Mover’s Distance and Structured Classifiers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12203–12213.

[DPC20] Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. “The Advantage of Conditional Meta-Learning for Biased Regularization and Fine-Tuning”. In: *Advances in Neural Information Processing Systems*. 2020.

[WDC20] Ruohan Wang, Yiannis Demiris, and Carlo Ciliberto. “Structured Prediction for Conditional Meta-Learning”. In: *Advances in Neural Information Processing Systems* (2020).

[Vuo+19] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. "Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation". In: *Advances in Neural Information Processing Systems*. 2019, pp. 1–12.

[Yao+19] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. "Hierarchically Structured Meta-learning". In: *International Conference on Machine Learning*. 2019, pp. 7045–7054.

[Jia+18] Xiang Jiang, Mohammad Havaei, Farshid Varno, Gabriel Chartrand, Nicolas Chapados, and Stan Matwin. "Learning to learn with conditional class dependencies". In: *International Conference on Learning Representations*. 2018.

[Jer+19] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. "Reconciling meta-learning and continual learning with online mixtures of tasks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 9119–9130.

[Man+20] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. "Charting the right manifold: Manifold mixup for few-shot learning". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 2218–2227.

[OLL18] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. "Tadam: Task dependent adaptive metric for improved few-shot learning". In: *Advances in Neural Information Processing Systems*. 2018, pp. 721–731.

[Che+18] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. "A Closer Look at Few-shot Classification". In: *International Conference on Learning Representations*. 2018.

[WTH21] Davis Wertheimer, Luming Tang, and Bharath Hariharan. “Few-Shot Classification With Feature Map Reconstruction Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021, pp. 8012–8021.

[YLX21] Shuo Yang, Lu Liu, and Min Xu. “Free Lunch for Few-shot Learning: Distribution Calibration”. In: *International Conference on Learning Representations.* 2021.

[Bat+20] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. “Improved few-shot visual classification”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 14493–14502.

[Req+19] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. “Fast and flexible multi-task classification using conditional neural adaptive processes”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[Bat+22] Peyman Bateni, Jarred Barber, Raghav Goyal, Vaden Masrani, Jan-Willem van de Meent, Leonid Sigal, and Frank Wood. “Beyond Simple Meta-Learning: Multi-Purpose Models for Multi-Domain, Active and Continual Few-Shot Learning”. In: *arXiv preprint arXiv:2201.05151* (2022).

[Gol+20] Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. “Unraveling meta-learning: Understanding feature representations for few-shot tasks”. In: *International Conference on Machine Learning.* PMLR. 2020, pp. 3607–3616.

[El +22] Adrian El Baz, Ihsan Ullah, Edesio Alcobaça, André CPLF Carvalho, Hong Chen, Fabio Ferreira, Henry Gouk, Chaoyu Guan, Isabelle Guyon, Timothy Hospedales, et al. “Lessons learned from the NeurIPS 2021 MetaDL challenge: Backbone fine-tuning without episodic meta-learning dominates for few-shot learning image classification”. In: *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR. 2022, pp. 80–96.

[Sch87] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. 1987.

[MJ05] Andreas Maurer and Tommi Jaakkola. “Algorithmic Stability and Meta-Learning.” In: *Journal of Machine Learning Research* 6.6 (2005).

[MPR16] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. “The Benefit of Multitask Representation Learning”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2853–2884.

[BKT19] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. “Provable guarantees for gradient-based meta-learning”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 424–433.

[Den+19] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. “Learning-to-Learn Stochastic Gradient Descent with Biased Regularization”. In: *International Conference on Machine Learning*. 2019, pp. 1566–1575.

[LST15] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. “Human-level concept learning through probabilistic program induction”. In: *Science* 350.6266 (2015), pp. 1332–1338.

[LST19] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. “The Omniglot Challenge: a 3-year Progress Report”. In: *Current Opinion in Behavioral Sciences* 29 (2019), pp. 97–104.

[Ren+18] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. "Meta-learning for semi-supervised few-shot classification". In: *International conference on learning representations* (2018).

[KZS15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition". In: *ICML deep learning workshop*. Vol. 2. Lille. 2015.

[Li+17] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. "Meta-Sgd: Learning To Learn Quickly for Few-Shot Learning". In: *arXiv preprint arXiv:1707.09835* (2017).

[Zin+19] Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. "Fast context adaptation via meta-learning". In: *Proceedings of the 36th International Conference on Machine Learning-Volume 70*. JMLR. org. 2019.

[Col+22] Liam Collins, Aryan Mokhtari, Sewoong Oh, and Sanjay Shakkottai. "MAML and ANIL Provably Learn Representations". In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, June 2022, pp. 4238–4310.

[Sil+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778.

[Bro+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[Mni+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[Van19] Joaquin Vanschoren. "Meta-learning". In: *Automated Machine Learning.* Springer, Cham, 2019, pp. 35–61.

[Sun+18] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. "Learning to compare: Relation network for few-shot learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2018, pp. 1199–1208.

[HDL17] David Ha, Andrew M Dai, and Quoc V Le. "HyperNetworks". In: *International Conference on Learning Representations.* 2017.

[QBL18] Hang Qi, Matthew Brown, and David G Lowe. "Low-shot learning with imprinted weights". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2018, pp. 5822–5830.

[Rod+20] Pau Rodríguez, Issam Laradji, Alexandre Drouin, and Alexandre Lacoste. “Embedding propagation: Smoother manifold for few-shot classification”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 121–138.

[GRS18] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. “Size-independent sample complexity of neural networks”. In: *Conference On Learning Theory*. PMLR. 2018, pp. 297–299.

[NAS18] Alex Nichol, Joshua Achiam, and John Schulman. “On first-order meta-learning algorithms”. In: *arXiv preprint arXiv:1803.02999* (2018).

[TKI20] Yonglong Tian, Dilip Krishnan, and Phillip Isola. “Contrastive Representation Distillation”. In: *International Conference on Learning Representations*. 2020.

[Du+20] Simon Shaolei Du, Wei Hu, Sham M Kakade, Jason D Lee, and Qi Lei. “Few-Shot Learning via Learning the Representation, Provably”. In: *International Conference on Learning Representations*. 2020.

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. “Unsupervised representation learning by predicting image rotations”. In: *International Conference on Learning Representations* (2018).

[Llo82] Stuart Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[LLB21] Wei-Hong Li, Xialei Liu, and Hakan Bilen. “Universal representation learning from multiple domains for few-shot classification”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9526–9535.

[Lee+20] Hae Beom Lee, Hayeon Lee, Donghyun Na, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. “Learning to Balance: Bayesian Meta-Learning for Imbalanced and Out-of-distribution Tasks". In: *International Conference on Learning Representations* (2020).

[Rus+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[Tri+20] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. "Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples". In: *International Conference on Learning Representations.* Apr. 2020.

[Maj+13] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. "Fine-Grained Visual Classification of Aircraft". In: *CoRR* abs/1306.5151 (2013).

[Wel+10] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. *Caltech-UCSD birds 200.* Tech. rep. California Institute of Technology, 2010.

[Cim+14] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. "Describing textures in the wild". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2014, pp. 3606–3613.

[Jon+16] Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. "The quick, draw!-ai experiment (2016)". In: *URL http://quickdraw. withgoogle. com* 4 (2016).

[SC18] B. Schroeder and Y. Cui. *Fgvcx fungi classification challenge.* 2018.

[NZ08] Maria-Elena Nilsback and Andrew Zisserman. “Automated flower classification over a large number of classes”. In: *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing.* IEEE. 2008, pp. 722–729.

[Hou+13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. “Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark”. In: *The 2013 international joint conference on neural networks (IJCNN).* Ieee. 2013, pp. 1–8.

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. “Microsoft coco: Common objects in context”. In: *European conference on computer vision.* Springer. 2014, pp. 740–755.

[RBS19] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. “Few-shot learning with embedded class models and shot-free meta training”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 331–339.

[Rid+21] Tal Ridnik, Emanuel Ben Baruch, Asaf Noy, and Lihi Zelnik. “ImageNet-21K Pretraining for the Masses”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.* Ed. by Joaquin Vanschoren and Sai-Kit Yeung. 2021.

[McI+18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas GroSSberger. “UMAP: Uniform Manifold Approximation and Projection”. In: *Journal of Open Source Software* 3.29 (2018), p. 861.

[Liu+21] Lu Liu, William L Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle. “A Universal Representation Transformer Layer for Few-Shot Image Classification". In: *International Conference on Learning Representations.* 2021.

[DSM20] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. "Selecting relevant features from a multi-domain representation for few-shot classification". In: *European Conference on Computer Vision.* Springer. 2020, pp. 769–786.

[RR+07] Ali Rahimi, Benjamin Recht, et al. "Random Features for Large-Scale Kernel Machines." In: *NIPS.* Vol. 3. 2007, p. 5.

[Sch38] Isaac J Schoenberg. "Metric Spaces and Completely Monotone Functions". In: *Annals of Mathematics* (1938), pp. 811–841.

[FXL18] Chelsea Finn, Kelvin Xu, and Sergey Levine. "Probabilistic model-agnostic meta-learning". In: *Advances in Neural Information Processing Systems.* 2018, pp. 9516–9527.

[Lak+11] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. "One shot learning of simple visual concepts". In: *Proc. of the Annual Meeting of the Cognitive Science society.* Vol. 33. 2011.

[Tos+19] Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. "Adaptive Deep Kernel Learning". In: *arXiv preprint arXiv:1905.12131* (2019).

[Tit+21] Michalis K Titsias, Francisco JR Ruiz, Sotirios Nikoloutsopoulos, and Alexandre Galashov. "Information theoretic meta learning with gaussian processes". In: *Uncertainty in Artificial Intelligence.* PMLR. 2021, pp. 1597–1606.

[Pat+20] Massimiliano Patacchiola, Jack Turner, Elliot J. Crowley, and Amos Storkey. "Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels". In: *Advances in Neural Information Processing Systems.* 2020.

[WR06] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA, 2006.

[Wil+16] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. “Deep kernel learning”. In: *Artificial intelligence and statistics*. PMLR. 2016, pp. 370–378.

[Álv+10] Mauricio Álvarez, David Luengo, Michalis Titsias, and Neil D. Lawrence. “Efficient Multioutput Gaussian Processes through Variational Inducing Kernels”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, Mar. 2010, pp. 25–32.

[Kon+20] Weihao Kong, Raghav Somani, Sham Kakade, and Sewoong Oh. “Robust meta-learning for mixed linear regression with small batches”. In: *Advances in neural information processing systems* 33 (2020), pp. 4683–4696.

[NCL21] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. “Dataset Meta-Learning from Kernel Ridge-Regression”. In: *International Conference on Learning Representations*. 2021.

[Zhe+20] Xiantong Zhen, Haoliang Sun, Yingjun Du, Jun Xu, Yilong Yin, Ling Shao, and Cees Snoek. “Learning to learn kernels with variational random features”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11409–11419.

[OSW05] Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. “Learning the Kernel with Hyperkernels”. In: *Journal of Machine Learning Research* 6 (Dec. 2005), pp. 1043–1071.

[Cri+06] Nello Cristianini, Jaz Kandola, Andre Elisseeff, and John Shawe-Taylor. “On kernel target alignment”. In: *Innovations in Machine Learning*. Springer, 2006, pp. 205–256.

[RR17] Alessandro Rudi and Lorenzo Rosasco. "Generalization Properties of Learning with Random Features." In: *NIPS*. 2017, pp. 3215–3225.

[Rud62] Walter Rudin. *Fourier analysis on groups*. Vol. 121967. Wiley Online Library, 1962.

[SS15] Bharath Sriperumbudur and Zoltan Szabo. "Optimal Rates for Random Fourier Features". In: *Advances in Neural Information Processing Systems*. Vol. 28. 2015, pp. 1144–1152.

[Tro19] Joel A Tropp. *Matrix Concentration & Computational Linear Algebra*. July 2019.

[One+20] Luca Oneto, Michele Donini, Giulia Luise, Carlo Ciliberto, Andreas Maurer, and Massimiliano Pontil. "Exploiting MMD and Sinkhorn divergences for fair and transferable representation learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15360–15370.

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems*. 2019, pp. 8026–8037.

[DG17] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017.

[Zha+17] Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen. "Cautionary tales on air-quality improvement in Beijing". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2205 (2017), p. 20170457.

[BJM18] Javier Burgués, Juan Manuel Jiménez-Soto, and Santiago Marco. "Estimation of the Limit of Detection in Semiconductor Gas Sensors Through Linearized Calibration Models”. In: *Analytica chimica acta* 1013 (2018), pp. 13–25.

[AES19] Antreas Antoniou, Harrison Edwards, and Amos Storkey. “How to train your MAML”. In: *International conference on learning representations* (2019).

[CWJ21] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation”. In: *PeerJ Computer Science* 7 (2021), e623.

[KK12] Purushottam Kar and Harish Karnick. “Random feature maps for dot product kernels”. In: *Artificial intelligence and statistics*. PMLR. 2012, pp. 583–591.

[ÁRL12] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. “Kernels for Vector-Valued Functions: A Review”. In: *Foundations and Trends in Machine Learning* 4.3 (June 2012), pp. 195–266. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000036.

[MP05] Charles A. Micchelli and Massimiliano Pontil. “On Learning Vector-Valued Functions”. In: *Neural Computation* 17.1 (Jan. 2005), pp. 177–204. ISSN: 0899-7667. DOI: 10.1162/0899766052530802.

[BHB16] Romain Brault, Markus Heinonen, and Florence Buc. “Random Fourier Features For Operator-Valued Kernels”. In: *Proceedings of The 8th Asian Conference on Machine Learning*. PMLR, Nov. 2016, pp. 110–125.

[Lem06] Claude Lemaréchal. “S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004 Hardback, ISBN 0 521 83378 7”. In: *Eur. J. Oper. Res.* 170.1 (2006), pp. 326–327. DOI: 10.1016/j.ejor.2005.02.002.

[CMR12] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. “Algorithms for Learning Kernels Based on Centered Alignment”. In: *The Journal of Machine Learning Research* 13.1 (Mar. 2012), pp. 795–828. ISSN: 1532-4435.

[Tib96] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[Bro+21] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velikovi. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. May 2021. DOI: 10.48550/arXiv.2104.13478.

[Bro+17] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric Deep Learning: Going beyond Euclidean Data”. In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42. ISSN: 1558-0792. DOI: 10.1109/MSP.2017.2693418.

[EZ21] Bryn Elesedy and Sheheryar Zaidi. “Provably Strict Generalisation Benefit for Equivariant Models”. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, July 2021, pp. 2959–2969.

[Ele21] Bryn Elesedy. “Provably Strict Generalisation Benefit for Invariance in Kernel Methods”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17273–17283.

[Swe+20] Conor Sweeney, Ricardo J Bessa, Jethro Browell, and Pierre Pinson. “The future of forecasting for renewable energy”. In: *Wiley Interdisciplinary Reviews: Energy and Environment* 9.2 (2020), e365.

[Wan+19] Huaizhi Wang, Zhenxing Lei, Xian Zhang, Bin Zhou, and Jianchun Peng. “A review of deep learning for renewable energy forecasting”. In: *Energy Conversion and Management* 198 (2019), p. 111799.

[Ore+20] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. *N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting.* Feb. 2020. DOI: 10.48550/arXiv.1905.10437.

[MSA18] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. “The M4 Competition: Results, Findings, Conclusion and Way Forward”. In: *International Journal of Forecasting* 34.4 (Oct. 2018), pp. 802–808. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2018.06.001.

[MH00] Spyros Makridakis and Michèle Hibon. “The M3-Competition: Results, Conclusions and Implications”. In: *International Journal of Forecasting.* The M3- Competition 16.4 (Oct. 2000), pp. 451–476. ISSN: 0169-2070. DOI: 10.1016/S0169-2070(00)00057-1.

[Ler+23] Arthur Leroy, Pierre Latouche, Benjamin Guedj, and Servane Gey. “Cluster-Specific Predictions with Multi-Task Gaussian Processes”. In: *Journal of Machine Learning Research* 24.5 (2023), pp. 1–49. ISSN: 1533-7928.

[Ler+22] Arthur Leroy, Pierre Latouche, Benjamin Guedj, and Servane Gey. “MAGMA: Inference and Prediction Using Multi-Task Gaussian Processes with Common Mean”. In: *Machine Learning* 111.5 (May 2022), pp. 1821–1849. ISSN: 1573-0565. DOI: 10.1007/s10994-022-06172-1.

[BDF91] Peter J Brockwell, Richard A Davis, and Stephen E Fienberg. *Time series: theory and methods.* Springer Science & Business Media, 1991.

[Ped+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[Har+20] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362.

[Hun07] John D Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.03 (2007), pp. 90–95.

[LH19] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2019.

[Mau09] Andreas Maurer. "Transfer Bounds for Linear Feature Learning". In: *Machine Learning* 75.3 (2009), pp. 327–350.

[Mau16] Andreas Maurer. "A vector-contraction inequality for rademacher complexities". In: *International Conference on Algorithmic Learning Theory*. Springer. 2016, pp. 3–17.