Bridging Multi-Task Learning and Meta-Learning: Towards Efficient Training and Effective Adaptation

Haoxiang Wang [1] Han Zhao [1] Bo Li [1]

Abstract

Multi-task learning (MTL) aims to improve the generalization of several related tasks by learning them jointly. As a comparison, in addition to the joint training scheme, modern meta-learning allows unseen tasks with limited labels during the test phase, in the hope of fast adaptation over them. Despite the subtle difference between MTL and meta-learning in the problem formulation, both learning paradigms share the same insight that the shared structure between existing training tasks could lead to better generalization and adaptation. In this paper, we take one important step further to understand the close connection between these two learning paradigms, through both theoretical analysis and empirical investigation. Theoretically, we first demonstrate that MTL shares the same optimization formulation with a class of gradient-based meta-learning (GBML) algorithms. We then prove that for over-parameterized neural networks with sufficient depth, the learned predictive functions of MTL and GBML are close. In particular, this result implies that the predictions given by these two models are similar over the same unseen task. Empirically, we corroborate our theoretical findings by showing that, with proper implementation, MTL is competitive against state-of-the-art GBML algorithms on a set of few-shot image classification benchmarks. Since existing GBML algorithms often involve costly second-order bi-level optimization, our first-order MTL method is an order of magnitude faster on large-scale datasets such as mini-ImageNet. We believe this work could help bridge the gap between these two learning paradigms, and provide a computationally efficient alternative to GBML that also supports fast task adaptation.

[1]University of Illinois at Urbana-Champaign, Urbana, IL, USA. Correspondence to: Haoxiang Wang <hwang264@illinois.edu>, Han Zhao <hanzhao@illinois.edu>, Bo Li <lbo@illinois.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

1. Introduction

Multi-task learning has demonstrated its efficiency and effectiveness on learning shared representations with training data from multiple related tasks simultaneously (Caruana, 1997; Ruder, 2017; Zhang & Yang, 2017). Such shared representations could transfer to many real-world applications, such as object detection (Zhang et al., 2014), image segmentation (Kendall et al., 2018), multi-lingual machine translation (Dong et al., 2015), and language understanding evaluation (Wang et al., 2019). On the other hand, in addition to the joint training scheme, modern meta-learning can leverage the shared representation to fast adapt to unseen tasks with only minimum limited data during the test phase (Hospedales et al., 2020). As a result, meta-learning has drawn increasing attention and been applied to a wide range of learning tasks, including few-shot learning (Snell et al., 2017; Vinyals et al., 2016; Lee et al., 2019b), meta reinforcement learning (Finn et al., 2017), speech recognition (Hsu et al., 2020) and bioinformatics (Luo et al., 2019).

Despite their subtle differences in problem formulation and objectives, both MTL and meta-learning aim to leverage the correlation between different tasks to enable better generalization to either seen or unseen tasks. However, a rigorous exploration of this intuitive observation is severely lacking in the literature. As a result, while being effective on fast adaptation to unseen tasks, many meta-learning algorithms still suffer from expensive computational costs (Nichol et al., 2018a; Antoniou et al., 2019; Hospedales et al., 2020). On the other hand, while being efficient in training, due to its problem formulation, MTL does not allow adaptation to unseen tasks, at least in a straightforward manner. Hence, a natural question to ask is,

> *Can we combine the best of both worlds from MTL and meta-learning, i.e., fast adaptation to unseen tasks with efficient training?*

To answer this question, one needs to first understand the relationship between MTL and meta-learning in greater depth. To this end, in this paper, we take the *first* attempt with the goal to bridge these two learning paradigms. In particular, we focus on a popular class of meta-learning methods, gradient-based meta-learning (GBML), which takes a bi-level optimization formulation inspired from the Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017). From an optimization perspective, we first show that MTL and a class of GBML algorithms share the same optimization formulation. Inspired by this simple observation, we then prove that, for sufficiently wide neural networks, these two methods lead to close predictors: on any test task, the predictions given by these two methods are similar, and the gap is inversely proportional to the network depth. Our theoretical results imply that, in principle, it is possible to improve the existing MTL methods to allow fast adaptation to unseen tasks, without loss in its training efficiency, and thus provide an affirmative answer to the above question.

Empirically, to corroborate our findings, we first conduct a series of experiments on synthetic data to show the increasing closeness between the predictors given by MTL and GBML, as the network depth grows. We then perform extensive experiments to show that with proper implementation, MTL can achieve similar or even better results than the *state-of-the-art* GBML algorithms, while enjoys *significantly lower* computational costs. This indicates that MTL could be potentially applied as a powerful and efficient alternative to GBML for meta-learning applications.

Our contributions could be briefly summarized as follows:

- *Bridging MTL and GBML from the optimization perspective:* We show that MTL and a class of GBML algorithms share the same optimization formulation. In particular, GBML takes a regularized bi-level optimization while MTL adopts the simple joint training.
- *Closeness in the function space:* we prove that for over-parameterized neural nets with sufficient width, the learned predictive functions of MTL and a GBML algorithm are close in the function space, indicating their predictions are similar on unseen (test) tasks. Furthermore, we empirically validate this theoretical result on synthetic data.
- *Empirical performance and efficiency:* Motivated by our theoretical results, we implement MTL with modern deep neural nets, and show that the performance of MTL is competitive against MetaOptNet (Lee et al., 2019b), a *state-of-the-art* GBML algorithm, on few-shot image classification benchmarks. Notably, the training of MTL is *an order of magnitude faster* than that of MetaOptNet, due to its first-order optimization. The code is released at `https://github.com/AI-secure/multi-task-learning`

2. Related Work

Multi-Task Learning Multi-task learning (MTL) is a method to jointly learn shared representations from multiple training tasks (Caruana, 1997). Past research on MTL is abundant. Theoretical results on learning shared representation with MTL have shown that the joint training scheme is more sample efficient than single-task learning, at least under certain assumptions of task relatedness, linear features and model classes (Maurer et al., 2016; Tripuraneni et al., 2020). Other works on MTL include designing more efficient optimization methods to explore the task and feature relationships (Evgeniou & Pontil, 2007; Argyriou et al., 2008; Zhang & Yeung, 2010; Zhao et al., 2020).

Meta-Learning Meta-learning, or learning-to-learn, is originally proposed for few-shot learning tasks (Thrun & Pratt, 1998; Baxter, 1998), where the goal is fast adaptation to unseen tasks. Among various meta-learning methods, a line of works following MAML, termed as gradient-based meta-learning (GBML) (Finn et al., 2017; Rajeswaran et al., 2019), has been increasingly applied in many downstream application domains. Recent works on understanding GBML have shown that MAML is implicitly performing representation learning, which is the key to its empirical success (Raghu et al., 2020). In particular, Saunshi et al. (2020) compares MTL and Reptile (Nichol et al., 2018b), a first-order variant of MAML, in a toy setting of *1d* subspace learning with *2-layer linear* models, and shows the upper bounds of their sample complexity are of the same order. In contrast, our theory is compatible with *non-linear* neural nets of *any depth* and has no restriction on the input dimension, which is a more realistic and practical setting. In addition to GBML, the considered MTL implementation shares some similarities with metric-based meta-learning (i.e., metric learning) methods in few-shot learning scenarios (Snell et al., 2017; Vinyals et al., 2016), since here we also only keep the trained hidden layers for test.

3. Preliminaries

We first provide a brief discussion to the common network architectures, training algorithms, and evaluation protocols for MTL and GBML.

3.1. Neural Networks Architectures

Consider a L-layer fully-connected neural network $f_\theta : \mathbb{R}^d \mapsto \mathbb{R}^k$, which contains h_l neurons in the l-th hidden layer for $l \in [L-1]$. Denote the parameters of the first $L-1$ layers (i.e., hidden layers) as $\theta^{<L}$, and the last hidden layer output (i.e., network representation/features) as $\phi_{\theta^{<L}}^\top : \mathbb{R}^d \mapsto \mathbb{R}^{h_{L-1}}$. For simplicity, we assume the output layer (i.e., network *head*) has no bias, and denote it as $w \in \mathbb{R}^{h_{L-1}\times k}$. Thus, for any input $x \in \mathbb{R}^d$, the neural network output can be expressed as

$$f_\theta(x) = \phi_{\theta^{<L}}^\top(x)\, w\ .$$

Networks used in MTL often have a *multi-head* structure, where each head corresponds to a training task. In this case, the shared hidden layers are treated as the shared representations. Formally, denote a L-layer N-head neural network as $\hat{f}_{\hat{\theta}}: \mathbb{R}^d \times [N] \mapsto \mathbb{R}^k$, s.t. for any input $x \in \mathbb{R}^d$ and *head index* $i \in [N]$, the network output is

$$\hat{f}_{\hat{\theta}}(x, i) = \phi^\top_{\hat{\theta}<L}(x)\ \hat{w}^{(i)}, \tag{1}$$

where $\phi_{\hat{\theta}<L}(x)$ is the last hidden layer output, $\hat{\theta}^{<L}$ is the parameters of first $L-1$ layers, and w_i is the i-th head in the output layer. Note that the network parameters are the union of parameters in the hidden layers and multi-head output layer, i.e., $\hat{\theta} = \{\hat{\theta}^{<L}\} \cup \{\hat{w}^{(i)}\}_{i\in[N]}$.

3.2. Multi-Task Learning

In MTL, a multi-head neural net with N heads is trained over N training tasks each with n samples (Ruder, 2017). For $i \in [N]$, denote the data for the i-th task as (X_i, Y_i), where $X_i \in \mathbb{R}^{n\times d}$ and $Y_i \in \mathbb{R}^{n\times k}$. The training of MTL is to minimize the following objective given loss function ℓ,

$$\min_{\hat{\theta}} \mathcal{L}_{MTL}(\hat{\theta}) := \sum_{i\in[N]} \ell\left(\phi^\top_{\hat{\theta}<L}(X_i)\ \hat{w}^{(i)},\ Y_i\right) \tag{2}$$

where $\phi^\top_{\hat{\theta}<L}(X_i) := \left(\phi^\top_{\hat{\theta}}(x)\right)_{x\in[X_i]} \in \mathbb{R}^{n\times h_{L-1}}$.

3.3. Gradient-Based Meta-Learning and ANIL

Here we introduce a representative algorithm of GBML, *Almost-No-Inner-Loop* (ANIL) (Raghu et al., 2020), which is a simplification of MAML. The setup is the same as Sec. 3.2, where N training tasks each with n sample-label pairs are provided, i.e., $\{X_i, Y_i\}_{i=1}^N$. In practice, a training protocol, *query-support split* (cf. Appendix *A* for more details), is often adopted. However, recent work has shown that such a split is not necessary (Bai et al., 2021). Hence, we do not consider the query-support split through this work.

ANIL minimizes the following loss over $\theta = \{\theta^{<L}, w\}$,

$$\min_{\theta} \mathcal{L}_{ANIL}(\theta) := \sum_{i\in[N]} \ell(\phi^\top_{\theta<L}(X_i)\ w_i^*,\ Y_i) \tag{3}$$

$$\text{s.t. } w_i^* = \texttt{InnerLoop}(w, \phi^\top_{\theta<L}(X_i), Y_i, \tau, \lambda) \tag{4}$$

where (4) is the common *inner-loop* optimization of GBML, which *runs τ steps of gradient descent w.r.t. w on the loss $\ell(\phi^\top_{\theta<L}(X_i)w, Y_i)$, with learning rate λ.*

Notably, Lin et al. (2021) empirically shows that with a frozen head w across training, ANIL has no performance drop, indicating that optimizing over w in the outer loop (3) is insignificant. Thus, the corresponding training objective of this ANIL without the outer-loop optimization of w is

$$\min_{\theta<L} \mathcal{L}_{ANIL}(\theta) := \sum_{i\in[N]} \ell(\phi^\top_{\theta<L}(X_i)\ w_i^*,\ Y_i) \tag{5}$$

with w_i^* defined in the same way as (4).

3.4. Fine-Tuning for Test Task Adaptation

In the test phase of few-shot learning, an arbitrary test task $\mathcal{T}$ consists of $(X, Y, X', Y') \in \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times k} \times \mathbb{R}^{n'\times d} \times \mathbb{R}^{n'\times k}$, where (X, Y) are *query* data and (X', Y') are *support* data. Note that the original formulation of MTL with multi-head network structures does not support adaptation to unseen tasks. To compare MTL and GBML on an equal footing, in this work, we adopt the following same test protocol on both MTL and GBML. First, a randomly initialized head w_{test} is appended to the last hidden layer of networks trained under MTL or GBML. Then, the head w_{test} is *fine-tuned* on labelled *support* samples (X', Y'), and the network makes predictions on the *query* samples X. Specifically, for a trained MTL model with parameters $\hat{\theta}$, its prediction on X after fine-tuning w_{test} on (X', Y') for $\hat{\tau}$ steps is

$$F_{MTL}(X, X', Y') = \phi^\top_{\hat{\theta}<L}(X)\ w^*_{test} \tag{6}$$

$$\text{s.t. } w^*_{test} = \texttt{InnerLoop}(w_{test}, \phi^\top_{\hat{\theta}<L}(X'), Y', \hat{\tau}, \lambda) \tag{7}$$

where w^*_{test} is the fined-tuned test head after $\hat{\tau}$ steps of gradient descent on w_{test}, and $\texttt{InnerLoop}$ is defined in the same way as (4). Similarly, the prediction of a trained ANIL model with parameters θ on X is

$$F_{ANIL}(X, X', Y') = \phi^\top_{\theta<L}(X)\ w^*_{test} \tag{8}$$

$$\text{s.t. } w^*_{test} = \texttt{InnerLoop}(w_{test}, \phi^\top_{\theta<L}(X'), Y', \hat{\tau}, \lambda) \tag{9}$$

4. Theoretical Analysis

In this section, we compare MTL with a class of GBML algorithms, and show that *(i)* essentially, they optimize the same objective with different optimization approaches, *(ii)* the learned predictors from both algorithms are close under a certain norm in the function space.

4.1. A Taxonomy of Gradient-Based Meta-Learning

Various GBML methods often differ in the way on how to design the *inner-loop optimization* in Eq. (4). For example, MAML, ANIL and some other MAML variants usually take *a few* gradient descent steps (typically 1∼10 steps), which is treated as an *early stopping* type of regularization (Rajeswaran et al., 2019; Grant et al., 2018). As a comparison, another line of GBML algorithms uses the explicit ℓ_2 *regularization* in the inner loop instead (Rajeswaran et al., 2019; Lee et al., 2019b; Bertinetto et al., 2019; Zhou et al., 2019b; Goldblum et al., 2020). In addition to regularization, variants of GBML methods also differ in the exact layers to optimize in the inner loop: While MAML optimizes all network layers in the inner loop, some other GBML algorithms are able to achieve state-of-the-art performance (Lee et al., 2019b) by only optimizing the *last layer* in the inner loop.

Based on the different *regularization* strategies and *optimized layers* in the inner-loop, we provide a taxonomy of GBML algorithms in Table 1.

Table 1. A taxonomy of gradient-based meta-learning algorithms based on the algorithmic design of Eq. (4).

Inner-Loop Optimized Layers	**Early Stopping**	ℓ_2 **Regularizer**
Last Layer	ANIL (Raghu et al., 2020)	MetaOptNet (Lee et al., 2019b) R2D2 (Bertinetto et al., 2019)
All Layers	MAML (Finn et al., 2017)	iMAML (Rajeswaran et al., 2019) Meta-MinibatchProx (Zhou et al., 2019b)

Table 2. Typical instantiations of the problem (10). $\alpha \in \mathbb{R}^+$ controls the strength of the regularization.

	ANIL	MetaOptNet	R2D2
ℓ	Cross-Entropy	SVM Loss	Squared Loss
$R(w_i)$	Early Stopping	$\alpha\|w_i\|_2^2$	$\alpha\|w_i\|_2^2$

For algorithms that only optimize the last layer in the inner-loop, we formulate their training objectives in a *unified framework*:

$$\min_{\theta^{<L}} \sum_{i\in[N]} \ell\big(\phi_{\theta^{<L}}^\top(X_i)\, w_i^*,\ Y_i\big) \tag{10}$$

$$\text{s.t. } w_i^* = \arg\min_{w_i}\ \ell\big(\phi_{\theta^{<L}}^\top(X_i)\, w_i,\ Y_i\big) + R(w_i) \tag{11}$$

Certainly, there are abundant choices for the loss function ℓ and the regularization $R(w_i)$, and we summarize the typical choices used in the literature in Table 2. For algorithms that optimize all layers in the inner loop, a unified framework similar to (10) and (11) is provided in Appendix A.

4.2. Equivalence Between GBML and MTL from an Optimization Perspective

In this section, we provide a simple observation that, surprisingly, the optimization objective of MTL shares the same formulation as that of GBML algorithms, (10). Specifically, the objective function of MTL, (2), can be re-written as

$$\min_{\hat\theta^{<L},\{\hat w^{(i)}\}_{i=1}^N} \overbrace{\sum_{i\in[N]} \ell\left(\phi_{\hat\theta^{<L}}^\top(X_i)\, \hat w^{(i)},\ Y_i\right)}^{\mathcal{L}_{MTL}(\hat\theta)} \tag{12}$$

where $\{\hat w^{(i)}\}_{i=1}^N$ are heads of a multi-head neural net, and $\hat\theta = \{\hat\theta^{<L}\} \cup \{\hat w^{(i)}\}_{i=1}^N$. As a comparison, if we plug (11) into (10), the GBML objective (10) can be simplified as

$$\min_{\theta^{<L}} \overbrace{\left[\min_{\{w_i\}_{i=1}^N} \sum_{i\in[N]} \ell\Big(\phi_{\theta^{<L}}^\top(X_i)\, w_i, Y_i\Big) + R(w_i)\right]}^{\mathcal{L}_{GBML}(\theta^{<L})} \tag{13}$$

Note that different from (12), the heads $\{w_i\}_{i=1}^N$ in (13) are transient, in the sense that GBML algorithms do not explicitly save them during training. On the other hand, $\theta^{<L}$ contains all parameters to optimize in (13), and $\theta^{<L}$ is optimized over $\ell_{GBML}(\theta^{<L})$, which is obtained by plugging in the minimizer of $\{w_i\}_{i=1}^N$ on the regularized loss. In other words, (13) is a bi-level optimization problem, with outer-loop optimization on network parameters $\theta^{<L}$ and inner-loop optimization on the transient heads $\{w_i\}_{i=1}^N$.

Clearly, up to the regularization term, the optimization problems (12) and (13) share the same structure and formulation. In terms of the algorithms used to solve these two optimization problems, it is worth pointing out that GBML usually solves (13) as a *bi-level* program where for each fixed $\theta^{<L}$, the algorithm will first compute the optimal heads $\{w_i\}_{i=1}^N$ as a function of $\theta^{<L}$, whereas in MTL, (12) is solved by the simple *joint optimization* over both $\hat\theta^{<L}$ and $\{\hat w^{(i)}\}_{i=1}^N$.

From the discussions above, we conclude that the optimization formulation of GBML is equivalent to that of MTL, where the only difference lies in the optimization algorithms used to solve them. Motivated by this observation, in the next section, we explore the equivalence of these two algorithms in terms of the predictors obtained after convergence, when the networks are sufficiently wide.

4.3. Closeness Between MTL and GBML from a Functional Perspective

In this section, we theoretically analyze MTL and a representative GBML algorithm, ANIL (Raghu et al., 2020), from a *functional* perspective, and show that the learned predictors of MTL and ANIL after convergence are close under a certain norm. Due to the page limit, we defer detailed proofs to appendix, and mainly focus on discussing the implications of our theoretical results. Before we proceed, we first formally introduce the problem setup and training protocol used in the following analysis.

Problem Setup To simplify our analysis and presentation, we consider the squared loss, i.e., $\ell(\hat y, y) = \frac{1}{2}\|\hat y - y\|_2^2$. Note that the use of squared loss is standard for theoretical analyses of neural net optimization (Jacot et al., 2018; Du et al., 2019; Allen-Zhu et al., 2019). Furthermore, recently, Hui & Belkin (2021) has also empirically demonstrated the effectiveness of squared loss in classification tasks from various domains. For the activation function and initialization scheme of neural nets, we focus on networks with ReLU activation and He's initialization[1] (He et al., 2016), which is also standard in practice.

With the squared loss, the objectives of MTL and ANIL, i.e., (2) and (3), can be simplifed to

$$\mathcal{L}_{MTL}(\hat{\theta}_t) = \frac{1}{2}\sum_{i\in[N]} \left\| \mathrm{vec}\left(\phi^\top_{\hat{\theta}<L}(X_i)\,\hat{w}^{(i)} - Y_i\right)\right\|_2^2,$$

$$\mathcal{L}_{ANIL}(\theta_t) = \frac{1}{2}\sum_{i\in[N]} \left\| \mathrm{vec}\Big(\phi^\top_{\theta<L}(X_i)\,w_i^* - Y_i\Big)\right\|_2^2,$$

where $\mathrm{vec}(\cdot)$ is the vectorizaton operation and $w_i^* = \mathtt{InnerLoop}(w, \phi^\top_{\theta<L}(X_i), Y_i, \tau, \lambda)$. During the test phase, for networks trained by MTL and ANIL, the predictions on any test task are obtained by fine-tuning an output head and predicting with this fine-tuned head (cf. Sec. 3.4).

Training Dynamics We consider gradient flow (i.e., continuous-time gradient descent) for the training of both MTL and ANIL, which is a common setting used for theoretical analysis of neural nets with more than two layers (Jacot et al., 2018; Lee et al., 2019a; Arora et al., 2019). In more detail, let η be the learning rate, then the training dynamics of MTL and ANIL can be described by

$$\frac{d\hat{\theta}_t}{dt} = -\eta\nabla_{\hat{\theta}_t}\mathcal{L}_{MTL}(\hat{\theta}_t), \quad \frac{d\theta_t}{dt} = -\eta\nabla_{\theta_t}\mathcal{L}_{ANIL}(\hat{\theta}_t) \quad (14)$$

where θ_t and $\hat{\theta}_t$ are network parameters at training step t.

NTK and NNGP Kernels Our forthcoming theoretical analysis involves both the Neural Tangent Kernel (NTK) (Jacot et al., 2018; Du et al., 2019; Allen-Zhu et al., 2019) and the Neural Network Gaussian Process (NNGP) kernel (Lee et al., 2018; Novak et al., 2019), which are tools used to understand the training trajectories of neural nets by reduction to classic kernel machines. For completeness, here we provide a brief introduction to both, so as to pave the way for our following presentation. Let the kernel functions of NTK and NNGP be $\Theta(\cdot,\cdot)$ and $\mathcal{K}(\cdot,\cdot)$, respectively. Analytically, the NTK and NNGP for networks of L layers can be computed recursively layer by layer (Lee et al., 2019a; Arora et al., 2019). Numerically, both kernels can be computed by using the Neural Tangents package (Novak et al., 2020). Furthermore, without loss of generality, we assume the inputs are normalized to have unit variance, following (Xiao et al., 2020), and we adopt the NTK parameterization (Lee et al., 2019a), which is the same with the standard neural net parameterization in terms of network output and training dynamics. More details about the parametrization can be found in Appendix A.

With the above discussions clearly exposed, now we are ready to state the following main lemma, which serves as the basic for our main result in this section. In particular, by leveraging tools from Lee et al. (2019a) and Wang et al. (2020), we are able to prove that for sufficiently wide neural nets trained under gradient flow of ANIL or MTL (i.e., by (14)), their predictions on any test task are equivalent to a special class of kernel regression, with kernels that we name as *(i)* ANIL Kernel Φ_{ANIL} and *(ii)* MTL Kernel Φ_{MTL}. Notice that both Φ_{ANIL} and Φ_{MTL} are composite kernels built on the NTK Θ and NNGP kernels $\mathcal{K}$.

Lemma 1 (Test Predictions of MTL & ANIL)**.** *Consider an arbitrary test task $\mathcal{T} = (X, Y, X', Y')$, as defined in Sec. 3.4. For arbitrarily small $\delta > 0$, there exists $\eta^*, h^* \in \mathbb{R}_+$ such that for networks with width greater than h^* and trained under gradient flow with learning rate $\eta < \eta^*$, with probability at least $1-\delta$ over random initialization, the test predictions on $\mathcal{T}$ (i.e., Eq. (6) and (8)) are*

$$F_{MTL}(X, X', Y') = G_{\hat{\tau}}(X, X', Y') + \Phi'_{MTL}((X, X', \hat{\tau}), \mathcal{X})\Phi^{-1}_{MTL}(\mathcal{X},\mathcal{X})\cdot\mathcal{Y}, \quad (15)$$

$$F_{ANIL}(X, X', Y') = G_{\hat{\tau}}(X, X', Y') + \Phi'_{ANIL}((X, X', \hat{\tau}), \mathcal{X})\Phi^{-1}_{ANIL}(\mathcal{X},\mathcal{X})\big[\mathcal{Y} - G_\tau(\mathcal{X},\mathcal{X},\mathcal{Y})\big], \quad (16)$$

up to an error of $\mathcal{O}(\frac{1}{\sqrt{h^*}})$ measured in ℓ_2 norm. In above equations, we used shorthand $\mathcal{X} = (X_i)_{i=1}^N \in \mathbb{R}^{Nn\times d}$ and $\mathcal{Y} = \mathrm{vec}((Y_i)_{i=1}^N) \in \mathbb{R}^{Nnk}$. Besides, the function G, kernels Φ_{MTL} & Φ_{ANIL}, and their variants Φ'_{MTL} & Φ'_{ANIL}, are defined below.

- ***Function** G. The function G is defined as*

$$G_{\hat{\tau}}(X, X', Y') = \mathcal{K}(X, X')\mathcal{K}(X', X')^{-1}(I - e^{-\lambda\mathcal{K}(X',X')\hat{\tau}})Y'$$

and $G_\tau(\mathcal{X}, \mathcal{X}', \mathcal{Y}') = \mathrm{vec}((G_\tau(X_i, X_i, Y_i))_{i=1}^N)$.

- ***MTL Kernels.** The kernel $\Phi_{MTL}(\mathcal{X},\mathcal{X})$ is a block matrix of $N\times N$ blocks. Its (i,j)-th block for any $i,j\in[N]$ is*

$$[\Phi_{MTL}(\mathcal{X},\mathcal{X})]_{ij} = \Theta(X_i, X_j) - \mathbb{1}[i\neq j]\mathcal{K}(X_i, X_j)\,.$$

Besides, Φ'_{MTL} is variant of the kernel function Φ_{MTL}, and $\Phi'_{MTL}((X, X', \hat{\tau}), \mathcal{X})$ is also a block matrix, of $1\times N$ blocks, with the $(1,j)$-th block as

$$[\Phi'_{MTL}((X, X', \hat{\tau}), \mathcal{X})]_{1j} = \Theta(X, X_j) - \mathcal{K}(X, X_j) - \mathcal{K}(X, X')T^{\hat{\tau}}_{\mathcal{K}}(X')\Big[\Theta(X', X_j) - \mathcal{K}(X', X_j)\Big]$$

where the function T is defined as

$$T^{\hat{\tau}}_{\mathcal{K}}(X') = \mathcal{K}(X', X')^{-1}\left(I - e^{-\lambda\mathcal{K}(X')\hat{\tau}}\right) \quad (17)$$

- ***ANIL kernels.** $\Phi_{ANIL}(\mathcal{X},\mathcal{X})$ is also a block matrix of $N\times N$ blocks. Its (i,j)-th block for any $i,j\in[N]$ is*

$$[\Phi_{\mathrm{ANIL}}(\mathcal{X},\mathcal{X})]_{ij} = e^{-\lambda\mathcal{K}(X_i,X_i)\tau}\Theta(X_i, X_j)e^{-\lambda\mathcal{K}(X_j,X_j)\tau},$$

[1]This is the common and default initialization scheme in Keras and PyTorch.

while $\Phi'_{ANIL}((X, X', \hat{\tau}), \mathcal{X})$ is a block matrix of $1 \times N$ blocks, with the $(1, j)$-th block as

$$[\Phi'_{ANIL}((X, X', \hat{\tau}), \mathcal{X})]_{1j} = \Theta(X, X_j) e^{-\lambda \mathcal{K}(X_j, X_j)\tau} - \mathcal{K}(X, X') T^{\hat{\tau}}_{\mathcal{K}}(X') \Theta(X', X_j) e^{-\lambda \mathcal{K}(X_j, X_j)\tau}$$

Remark The function G is implicitly related to task adaptation. For instance, on the test task $\mathcal{T} = (X, Y, X', Y')$, $G_{\hat{\tau}}(X, X', Y')$ is equivalent to the output of a trained wide network on X, where the network is trained on data (X', Y') with learning rate λ for $\hat{\tau}$ steps from the initialization.

Proof Sketch. Lemma 1 is a key lemma used in our analysis, hence we provide a high-level sketch of its proof. The main idea is that, for over-parametrized neural nets, we could approximate the network output function by its first-order Taylor expansion with the corresponding NTKs and NNGPs (Lee et al., 2019a), provided the network parameters do not have a large displacement during training. Under this case, we can further prove the global convergence of both MTL and ANIL by leveraging tools from Wang et al. (2020). The last step is then to analytically compute the corresponding kernels, as shown in Lemma 1. ■

With Lemma 1, we proceed to derive the main result in this section. Namely, the predictions given by MTL and ANIL over any test task are close. Intuitively, from (15) and (16), we can see the test predictions of MTL and ANIL admit a similar form, even though they use different kernels. Inspired by this observation, a natural idea is to bound the difference between the MTL and ANIL kernels by analyzing their spectra, which leads to the following theorem:

Theorem 1. *Consider an arbitrary test task, $\mathcal{T} = (X, Y, X', Y') \in \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times k} \times \mathbb{R}^{n' \times d} \times \mathbb{R}^{n' \times k}$. For any $\epsilon > 0$, there exists a constant $h^* = \mathcal{O}(\epsilon^{-2})$ s.t. if the network width h is greater than h^*, for ReLU networks with He's initialization, the average difference between the predictions of ANIL and MTL on the query samples X is bounded by*

$$\|F_{ANIL}(X, X', Y') - F_{MTL}(X, X', Y')\|_2 \leq \mathcal{O}\left(\lambda\tau + \frac{1}{L}\right) + \epsilon. \qquad (18)$$

Remark The bound (18) is dominated by $\mathcal{O}(\lambda\tau + \frac{1}{L})$. Notice that λ and τ are the inner-loop learning rate and adaptation steps of ANIL. In practical implementations, $\lambda\tau \in [0.01, 0.5]$, which is small. In the state-of-the-art meta-learning models, the network depth $L \geq 12$, hence $1/L$ is also small. Since the bound holds for *any* test data, it implies that the average discrepancy between the learned predictors of MTL and ANIL is small. Notice that we only study the effect of hyperparameters of models and algorithms (e.g., L, λ, τ), and consider dataset-specific parameters (e.g., N, n, k, d) as constants.

Proof Sketch. The first step is to apply the analytic forms of F_{ANIL} and F_{MTL} in Lemma 1 to compute their difference. We then prove that the norm of the difference is bounded as

$$\|F_{ANIL}(X, X', Y') - F_{MTL}(X, X', Y')\|_2 \leq \mathcal{O}\left(L\|\Theta(\mathcal{X}, \mathcal{X})^{-1} - \Phi^{-1}_{MTL}(\mathcal{X}, \mathcal{X})\|_{op}\right) + \mathcal{O}(\lambda\tau + \frac{1}{\sqrt{h}})$$

Then, by leveraging theoretical tools from Xiao et al. (2020), we obtain an in-depth structure of the spectrum of the MTL kernel Φ_{MTL} for deep ReLU nets, in order to prove that

$$\|\Theta(\mathcal{X}, \mathcal{X})^{-1} - \Phi^{-1}_{MTL}(\mathcal{X}, \mathcal{X})\|_{\mathrm{op}} \leq \mathcal{O}(\frac{1}{L^2}),$$

with a fine-grained analysis. Finally, defining $h^* = \mathcal{O}(\varepsilon^{-2})$, we obtain the bound (18) for networks with $h > h^*$. ■

Theorem 1 could also be extended to ResNets, which have been widely adopted in modern meta-learning applications:

Corollary 1.1. *For (i) Residual ReLU networks (He et al., 2016) and (ii) Residual ReLU networks with Layer Normalization (Ba et al., 2016), Theorem 1 holds true.*

Proof Sketch. By leveraging tools from Xiao et al. (2020), we show that the residual connection only puts an extra factor e^L on the MTL kernel Φ_{MTL}. However, plugging it in to the expression for F_{MTL} derived in Lemma 1, one can find that the extra factors cancel out, since

$$e^L \Phi'_{MTL}((X, X', \hat{\tau}), \mathcal{X}) \cdot (e^L \Phi_{MTL}(\mathcal{X}, \mathcal{X}))^{-1} = \Phi'_{MTL}((X, X', \hat{\tau}), \mathcal{X}) \Phi_{MTL}(\mathcal{X}, \mathcal{X})^{-1}.$$

Similar observation also holds for Φ_{ANIL} and F_{ANIL}. Thus, Theorem 1 applies to residual ReLU networks as well.

For residual ReLU nets with LayerNorm, Φ_{MTL} and Φ_{ANIL} have identical kernel spectra and structures as the regular ReLU nets, up to a difference of a negligible order. Hence, Theorem 1 also applies to this class of networks. ■

See Appendix B for the full proof of Lemma 3, Theorem 1 and Corollary 1.1.

5. Experiments

In this section, we first provide an empirical validation of Theorem 1 on synthetic data. Then, we perform a large-scale empirical study of MTL with unseen task adaptation on few-shot image classification benchmarks to compare with state-of-the-art meta-learning algorithms. The code is released at `https://github.com/AI-secure/multi-task-learning`

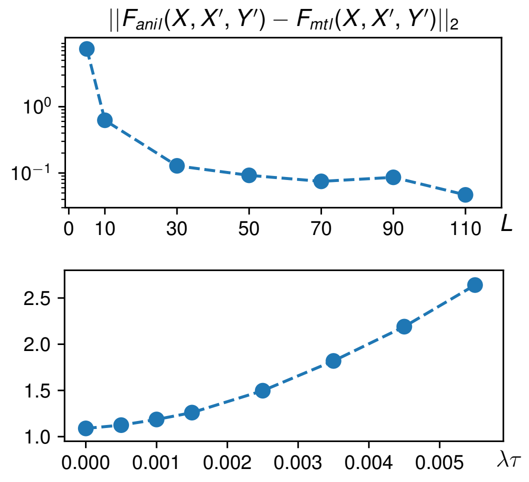

Figure 1. Empirical validation of Theorem 1 on synthetic data. We vary L in the first figure with fixed $\lambda\tau = 0$, and vary $\lambda\tau$ in the two figures with fixed $L = 10$, to observe the corresponding trends in the prediction difference $\|F_{ANIL}(X, X', Y') - F_{MTL}(X, X', Y')\|_2$.

5.1. Closeness between MTL and GBML predictions

Problem Setting We consider a few-shot regression problem to verify the theoretical claims in Theorem 1, and adopt the notation defined in Sec. 3. For each training task i with data (X_i, Y_i), it has two task-specific parameters $\mu_i \in \mathbb{R}^d$ and $\nu_i \in \mathbb{R}+$. The data points in X_i are sampled i.i.d. from $\mathcal{N}(\mu_i, \nu_i^2 I)$, and the label of each point x is generated by a quadratic function $y = \nu_i(x - \mu_i)^2$. Similarly, any test task $\mathcal{T} = (X, Y, X', Y')$ also has its task-specific parameters $(\mu_{\text{test}}, \nu_{\text{test}})$, and the points from its query and support set X, X' are drawn i.i.d. from $\mathcal{N}(\mu_{\text{test}}, \nu_{\text{test}}^2 I)$, with the label of each point x following $y = \nu_{\text{test}}(x - \mu_{\text{test}})^2$.

Dataset Synthesis We fix the input dimension $d = 10$ and generate $N = 20$ training tasks each with $n = 10$ data points. In each test task, there are 5 support and 10 query data points. For each training or test task, its task-specific parameters (μ, ν) are generated by $\mu \sim \mathcal{N}(0, I)$ and $\nu \sim \text{Unif}(1.3, 1.6)$.

Implementation Details We implement the functions F_{MTL} and F_{ANIL} in (15) and (16) by using the empirical kernel functions of NTK and NNGP provided by Neural Tangents (Novak et al., 2020). As suggested by Novak et al. (2020), we construct neural nets with width as 512 to compute kernels. Following Sec. 4.3, the networks use the ReLU activation and He's initialization (He et al., 2016).

Results We generate 20 test tasks over 5 runs for the empirical evaluation, and we vary the values of $\lambda\tau$ and L appearing in the bound (18) of Theorem 1. Figure 1 shows that as $\lambda\tau$ decreases or L increases, the norm of the prediction difference $\|F_{MTL}(X, X', Y') - F_{ANIL}(X, X', Y')\|_2$ decreases correspondingly, which is in agreement with (18). More experimental details can be found in Appendix C.

Note that Theorem 1 is built on fully connected nets, thus it is not directly applicable to modern convolutional neural nets (ConvNets) with residual connections, max pooling, BatchNorm, and Dropout, which are commonly used in meta-learning practice. Hence, we perform another empirical study on modern ConvNets in Sec. 5.2.

5.2. Few-Shot Learning Benchmarks

We conduct experiments on a set of widely used benchmarks for few-shot image classification: mini-ImageNet, tiered-ImageNet, CIFAR-FS and FC100. The first two are derivatives of ImageNet (Deng et al., 2009), while the last two are derivatives of CIFAR-100 (Krizhevsky, 2009). **Benchmarks.**

- mini-ImageNet (Vinyals et al., 2016): It contains 60,000 colored images of 84x84 pixels, with 100 classes (each with 600 images) split into 64 training classes, 16 validation classes and 20 test classes.
- tiered-ImageNet (Ren et al., 2018): It contains 779,165 colored images of 84x84 pixels, with 608 classes split into 351 training, 97 validation and 160 test classes.
- CIFAR-FS (Bertinetto et al., 2019): It contains 60,000 colored images of 32x32 pixels, with 100 classes (each with 600 images) split into 64 training classes, 16 validation classes and 20 test classes.
- FC100 (Oreshkin et al., 2018): It contains 60,000 colored images of 32x32 pixels, with 100 classes split into 60 training classes, 20 validation classes and 20 test classes.

Network Architecture Following previous meta-learning works (Lee et al., 2019b; Oreshkin et al., 2018; Tian et al., 2020), we use ResNet-12 as the backbone, which is a residual neural network with 12 layers (He et al., 2016).

Data Augmentation In training, we adopt the data augmentation used in Lee et al. (2019b) that consists of random cropping, color jittering, and random horizontal flip.

Optimization Setup We use RAdam (Liu et al., 2020), a variant of Adam (Kingma & Ba, 2015), as the optimizer for MTL. We adopt a public PyTorch implementation[2], and use the default hyper-parameters. Besides, we adopt the ReduceOnPlateau learning rate scheduler[3] with the early stopping regularization[4].

Model Selection. At the end of each training epoch, we evaluate the validation accuracy of the trained MTL model

[2]`https://github.com/jettify/pytorch-optimizer`

[3]`https://pytorch.org/docs/stable/optim.html#torch.optim.lr_scheduler.ReduceLROnPlateau`

[4]We stop the training if the validation accuracy does not increase for several epochs.

Table 3. **Comparison on four few-shot image classification benchmarks.** Average few-shot test classification accuracy (%) with 95% confidence intervals. 32-32-32-32 denotes a 4-layer convolutional neural net with 32 filters in each layer. In each column, **bold** values are the highest accuracy, or the accuracy no less than 1% compared with the highest one.

		mini-ImageNet 5-way		**tiered-ImageNet 5-way**	
Model	**Backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
MAML (Finn et al., 2017)	32-32-32-32	48.70 ± 1.84	63.11 ± 0.92	51.67 ± 1.81	70.30 ± 1.75
ANIL (Raghu et al., 2020)	32-32-32-32	48.0 ± 0.7	62.2 ± 0.5	-	-
R2D2 (Bertinetto et al., 2019)	96-192-384-512	51.2 ± 0.6	68.8 ± 0.1	-	-
TADAM (Oreshkin et al., 2018)	ResNet-12	58.50 ± 0.30	76.70 ± 0.30	-	-
MetaOptNet (Lee et al., 2019b)	ResNet-12	$\mathbf{62.64 \pm 0.61}$	$\mathbf{78.63 \pm 0.46}$	65.99 ± 0.72	81.56 ± 0.53
MTL-ours	ResNet-12	59.84 ± 0.22	$\mathbf{77.72 \pm 0.09}$	$\mathbf{67.11 \pm 0.12}$	$\mathbf{83.69 \pm 0.02}$

		CIFAR-FS 5-way		**FC100 5-way**	
Model	**Backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
MAML (Finn et al., 2017)	32-32-32-32	58.9 ± 1.9	71.5 ± 1.0	-	-
R2D2 (Bertinetto et al., 2019)	96-192-384-512	65.3 ± 0.2	79.4 ± 0.1	-	-
TADAM (Oreshkin et al., 2018)	ResNet-12	-	-	40.1 ± 0.4	56.1 ± 0.4
ProtoNet (Snell et al., 2017)	ResNet-12	$\mathbf{72.2 \pm 0.7}$	$\mathbf{83.5 \pm 0.5}$	37.5 ± 0.6	52.5 ± 0.6
MetaOptNet (Lee et al., 2019b)	ResNet-12	$\mathbf{72.6 \pm 0.7}$	$\mathbf{84.3 \pm 0.5}$	41.1 ± 0.6	55.5 ± 0.6
MTL-ours	ResNet-12	69.5 ± 0.3	$\mathbf{84.1 \pm 0.1}$	$\mathbf{42.4 \pm 0.2}$	$\mathbf{57.7 \pm 0.3}$

and save a model checkpoint. After training, we select the model checkpoint with the highest validation accuracy, and evaluate it on the test set to obtain the test accuracy.

Feature Normalization Following a previous work on few-shot image classification (Tian et al., 2020), we normalize features (i.e., last hidden layer outputs) in the meta-test and meta-validations stages. Besides, we also find the feature normalization is effective to the training of MTL on most benchmarks[5], which might be due to the effectiveness of feature normalization for representation learning (Wang & Isola, 2020).

Fine-Tuning for Task Adaptation In the meta-validation and meta-testing stages, following Sec. 3.4, we fine-tune a linear classifier on the outputs of the last hidden layer with the cross-entropy loss. We use the logistic regression classifier with ℓ_2 regularization from `scikit-learn` for the fine-tuning (Pedregosa et al., 2011). An ablation study on the ℓ_2 regularization is provided in Appendix C.2.

Implementation Details Our implementation is built on the `learn2learn`[6] package (Arnold et al., 2020), which provides data loaders and other utilities for meta-learning in PyTorch (Paszke et al., 2019). We implement MTL on a multi-head version of ResNet-12. Notice that the number of distinct training tasks is combinatorial for 5-way classification on the considered benchmarks, e.g., $\binom{64}{5} = 7.6 \times 10^6$ for mini-ImageNet and $\binom{351}{5} = 4.3 \times 10^9$ for tiered-ImageNet. Hence, due to memory constraints, we cannot construct separate heads for all tasks. Thus, we devise a memory-efficient implementation of the multi-head structure. For instance, on tiered-ImageNet with 351 training classes, we construct a 351-way linear classifier on top of the last hidden layer. Then, for each training task of 5 classes, we select the 5 corresponding row vectors in the weight matrix of the 351-way linear classifier, and merge them to obtain a 5-way linear classifier for this training task.

Empirical Results During meta-testing, we evaluate MTL over 3 runs with different random seeds, and report the mean accuracy with the 95% confidence interval in Table 3. The accuracy for each run is computed as the mean accuracy over 2000 tasks randomly sampled from the test set. The model selection is made on the validation set.

Performance Comparison In Table 3, we compare MTL with a set of popular meta-learning algorithms on the four benchmarks, in the common setting of 5-way few-shot classification. Notice that MetaOptNet is a state-of-the-art GBML algorithm, and MTL is competitive against it on these benchmarks: across the 8 columns/settings of Table 3, MTL is worse than MetaOptNet in 2 columns, comparable with MetaOptNet in 2 columns, and outperforms MetaOptNet in 4 columns. Therefore, we can conclude that MTL is competitive with the state-of-the-art of GBML algorithms on few-shot image classification benchmarks.

[5]It is effective on mini-ImageNet, tiered-ImageNet, and CIFAR-FS, while being ineffective on FC100.

[6]`http://learn2learn.net/`

Table 4. Efficiency Comparison on mini-ImageNet for 5-way 5-shot classification.

	Test Accuracy	GPU Hours
MetaOptNet	78.63%	85.6 hrs
MTL	77.72%	3.7 hrs

Training Efficiency GBML algorithms are known to be computationally expensive due to the costly second-order bi-level optimization they generally take (Hospedales et al., 2020). In contrast, MTL uses first-order optimization, and as a result, the training of MTL is significantly more efficient. To illustrate this more concretely, we compare the training cost of MTL against MetaOptNet on a AWS server with 4x Nvidia V100 GPU cards[7]. For MetaOptNet (Lee et al., 2019b), we directly run the official PyTorch code[8] with the optimal hyper-parameters[9] provided by the authors. Since the implementations of MetaOptNet and MTL are both written in PyTorch with the same network structure and similar data loaders (both adopting TorchVision dataset wrappers), we believe the efficiency comparison is fair. Note that the two ImageNet derivatives (i.e., mini-ImageNet of 7.2 GB and tiered-ImageNet of 29 GB) are much bigger than that of the two CIFAR-100 derivatives (i.e., CIFAR-FS of 336 MB and FC100 of 336 MB). It is more practically meaningful to reduce the training cost on big datasets like the ImageNet derivatives, thus we only perform the efficiency comparison on mini-ImageNet and tiered-ImageNet.

In Table 4, we present the GPU hours for the training of MetaOptNet and MTL with optimal hyper-parameters on mini-ImageNet, showing that the training of MTL is 23x times faster compared with MetaOptNet.

Figure 2 shows the *efficiency-accuracy tradeoff* of MTL vs. MetaOptNet on tiered-Imagenet. The training of MetaOptNet takes 63 GPU hours, while MTL has various training costs depending on the batch size and the number of epochs. From Figure 2 we can see that, even though MTL is only 3.6x faster when achieving the optimal test accuracy, we can train MTL with a smaller number of epochs or batch size, which reduces the training time at the cost of a small performance drop ($\leq 2.2\%$). As shown in Figure 2, while the training of MTL is 11x faster compared with MetaOptNet, its test accuracy (81.55%) can still match MetaOptNet (81.56%).

[7]The p3.8xlarge instance in AWS EC2: https://aws.amazon.com/ec2/instance-types/p3/

[8]https://github.com/kjunelee/MetaOptNet/

[9]The optimal hyperparameters of MetaOptNet for mini-ImageNet and tiered-Imagenet involve a large batch size that requires 4 GPUs.

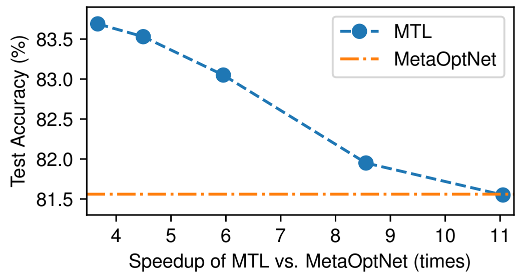

Figure 2. Efficiency comparison on tiered-ImageNet for 5-way 5-shot classification. The x-axis is the speedup of MTL compared with MetaOptNet, and the y-axis is the mean test accuracy. Notice that we only tune the batch size and number of training epochs for MTL in this comparison.

Remarks on the Empirical Results Traditionally, the training tasks and test tasks for MTL are the same. Our empirical results on few-shot learning reveal that, even as the test tasks are distinct from the training tasks, MTL can still be quite powerful. Recent theoretical studies on MTL show that the joint training of MTL over diverse tasks can learn representations useful for unseen tasks (Tripuraneni et al., 2020; Du et al., 2021), and our few-shot learning experiment supports these theories with positive empirical results. On the other hand, the MTL model we implemented is quite simple, which can be viewed as the original MTL proposal (Caruana, 1997) with a new memory-efficient trick. It is likely that more advanced variants of MTL could achieve even better performance on few-shot learning.

6. Conclusion

In this paper, we take an important step towards bridging the gap between MTL and meta-learning, both theoretically and empirically. Theoretically, we show that MTL and gradient-based meta-learning (GBML) share the same optimization formulation. We then further prove that, with sufficiently wide neural networks, the learned predictors from both algorithms give similar predictions on unseen tasks, which implies that it is possible to achieve fast adaptation and efficient training simultaneously. Inspired by our theoretical findings, empirically, we develop a variant of MTL that allows adaptation to unseen tasks, and show that it is competitive against the state-of-the-art GBML algorithms over a set of few-shot learning benchmarks while being significantly more efficient. We believe our work contributes to opening a new path towards models that simultaneously allow efficient training and fast adaptation.

Acknowledgements

Haoxiang Wang would like to thank Sébastien Arnold and Ruoyu Sun for helpful discussions. This work is partially supported by NSF grant No.1910100, NSF CNS 20-46726 CAR, and Amazon Research Award.

References

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. *International Conference on Machine Learning*, 2019.

Antoniou, A., Edwards, H., and Storkey, A. How to train your MAML. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJGven05Y7.

Argyriou, A., Evgeniou, T., and Pontil, M. Convex multi-task feature learning. *Machine learning*, 73(3):243–272, 2008.

Arnold, S. M., Mahajan, P., Datta, D., Bunner, I., and Zarkias, K. S. learn2learn: A library for meta-learning research. *arXiv preprint arXiv:2008.12284*, 2020.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. *NeurIPS*, 2019.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bai, Y., Chen, M., Zhou, P., Zhao, T., Lee, J. D., Kakade, S. M., Wang, H., and Xiong, C. How important is the train-validation split in meta-learning? In *ICML*, 2021.

Balcan, M.-F., Khodak, M., and Talwalkar, A. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pp. 424–433, 2019.

Baxter, J. Theoretical models of learning to learn. In *Learning to learn*, pp. 71–94. Springer, 1998.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxnZh0ct7.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1723–1732, 2015.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. *International Conference on Machine Learning*, 2019.

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. Few-shot learning via learning the representation, provably. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=pW2Q2xLwIMD.

Evgeniou, A. and Pontil, M. Multi-task feature learning. *Advances in neural information processing systems*, 19: 41, 2007.

Fallah, A., Mokhtari, A., and Ozdaglar, A. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pp. 1082–1092, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. Online meta-learning. In *International Conference on Machine Learning*, pp. 1920–1930, 2019.

Goldblum, M., Reich, S., Fowl, L., Ni, R., Cherepanova, V., and Goldstein, T. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*, pp. 3607–3616. PMLR, 2020.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ_UL-k0b.

Hanin, B. and Nica, M. Finite depth and width corrections to the neural tangent kernel. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgndT4KwB.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Henderson, H. V. and Searle, S. R. On deriving the inverse of a sum of matrices. *Siam Review*, 23(1):53–60, 1981.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey, 2020.

Hsu, J.-Y., Chen, Y.-J., and Lee, H.-y. Meta learning for end-to-end low-resource speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7844–7848. IEEE, 2020.

Hu, Y., Zhang, S., Chen, X., and He, N. Biased stochastic gradient descent for conditional stochastic optimization. *arXiv preprint arXiv:2002.10790*, 2020.

Hui, L. and Belkin, M. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In *ICLR*, 2021.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Ji, K., Yang, J., and Liang, Y. Multi-step model-agnostic meta-learning: Convergence and improved algorithms. *arXiv preprint arXiv:2002.07836*, 2020.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Khodak, M., Balcan, M.-F. F., and Talwalkar, A. S. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pp. 5915–5926, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1EA-M-0Z.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019a.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *CVPR*, 2019b.

Lin, Z., Zhao, Z., Zhang, Z., Baoxing, H., and Yuan, J. To learn effective features: Understanding the task-specific adaptation of {maml}, 2021. URL https://openreview.net/forum?id=FPpZrRfz6Ss.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020.

Luo, Y., Ma, J., Ideker, X. T., Zhao, J., Su, P. Y., and Liu, Y. Mitigating data scarcity in protein binding prediction using meta-learning. In *Research in Computational Molecular Biology: 23rd Annual International Conference, RECOMB 2019, Washington, DC, USA, May 5-8, 2019, Proceedings*, volume 11467, pp. 305. Springer, 2019.

Maurer, A., Pontil, M., and Romera-Paredes, B. The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(1):2853–2884, 2016.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018a.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018b.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1g30j0qF7.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

Oreshkin, B., Rodríguez López, P., and Lacoste, A. Tadam: Task dependent adaptive metric for improved few-shot learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 721–731. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/

66808e327dc79d135ba18e051673d906-Paper.pdf.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pp. 113–124, 2019.

Ren, M., Ravi, S., Triantafillou, E., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJcSzz-CZ.

Ruder, S. An overview of multi-task learning in deep neural networks, 2017.

Saunshi, N., Zhang, Y., Khodak, M., and Arora, S. A sample complexity separation between non-convex and convex meta-learning, 2020.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Thrun, S. and Pratt, L. Learning to learn: Introduction and overview. In *Learning to learn*, pp. 3–17. Springer, 1998.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. Rethinking few-shot image classification: a good embedding is all you need? *ECCV*, 2020.

Tripuraneni, N., Jordan, M., and Jin, C. On the theory of transfer learning: The importance of task diversity. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7852–7862. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/59587bffec1c7846f3e34230141556ae-Paper.pdf.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Wang, H., Sun, R., and Li, B. Global convergence and induced kernels of gradient-based meta-learning with neural nets. *arXiv preprint arXiv:2006.14606*, 2020.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Xiao, L., Pennington, J., and Schoenholz, S. S. Disentangling trainability and generalization in deep learning. *ICML*, 2020.

Xu, R., Chen, L., and Karbasi, A. Meta learning in the continuous time limit. *arXiv preprint arXiv:2006.10921*, 2020.

Zhang, Y. and Yang, Q. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 09 2017. ISSN 2095-5138. doi: 10.1093/nsr/nwx105. URL https://doi.org/10.1093/nsr/nwx105.

Zhang, Y. and Yeung, D. Y. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, UAI 2010*, pp. 733, 2010.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pp. 94–108. Springer, 2014.

Zhao, H., Stretcu, O., Smola, A. J., and Gordon, G. J. Efficient multitask feature and relationship learning. In *Uncertainty in Artificial Intelligence*, pp. 777–787. PMLR, 2020.

Zhou, P., Yuan, X., Xu, H., and Yan, S. Efficient meta learning via minibatch proximal update. *Neural Information Processing Systems*, 2019a.

Zhou, P., Yuan, X., Xu, H., Yan, S., and Feng, J. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems*, pp. 1534–1544, 2019b.