Global Convergence of MAML and Theory-Inspired Neural Architecture Search for Few-Shot Learning

Haoxiang Wang* Yite Wang* Ruoyu Sun Bo Li
University of Illinois Urbana-Champaign
{hwang264,yitew2,ruoyus,lbo}@illinois.edu

Abstract

Model-agnostic meta-learning (MAML) and its variants have become popular approaches for few-shot learning. However, due to the non-convexity of deep neural nets (DNNs) and the bi-level formulation of MAML, the theoretical properties of MAML with DNNs remain largely unknown. In this paper, we first prove that MAML with over-parameterized DNNs is guaranteed to converge to global optima at a linear rate. Our convergence analysis indicates that MAML with over-parameterized DNNs is equivalent to kernel regression with a novel class of kernels, which we name as Meta Neural Tangent Kernels (MetaNTK). Then, we propose MetaNTK-NAS, a new training-free neural architecture search (NAS) method for few-shot learning that uses MetaNTK to rank and select architectures. Empirically, we compare our MetaNTK-NAS with previous NAS methods on two popular few-shot learning benchmarks, miniImageNet, and tieredImageNet. We show that the performance of MetaNTK-NAS is comparable or better than the state-of-the-art NAS method designed for few-shot learning while enjoying more than 100x speedup. We believe the efficiency of MetaNTK-NAS makes itself more practical for many real-world tasks. Our code is released at `github.com/YiteWang/MetaNTK-NAS`.

1. Introduction

Meta-learning, or learning-to-learn (LTL) [59], has received much attention due to its applicability in few-shot image classification [23, 69], meta reinforcement learning [18, 23, 62], and other domains such as natural language processing [8, 78] and computational biology [43]. The primary motivation for meta-learning is to fast learn a new task from a small amount of data, with prior experience on similar but different tasks. There are a few meta learning approaches designed for few shot image classification, such as metric-based [57, 63], model-based [45, 56], optimizer-based [39, 52]. Model-agnostic meta-learning (MAML) is a popular gradient-based meta-learning approach, due to its simplicity and good performance in many meta-learning tasks [18, 19]. MAML formulates a bi-level optimization problem, where the inner-level objective represents the adaption to a given task, and the outer-level objective is the meta-training loss. There are many variants of MAML, [20, 21, 31, 46, 51], and they are almost always applied together with deep neural networks (DNNs) in practice.

Even though MAML with DNNs is empirically successful, this approach still lacks a thorough theoretical understanding. For example, the most common practice is to use *gradient descent* approach (e.g., SGD or Adam [34]) to train MAML with DNNs, and the optimization can usually obtain almost zero training loss and 100% training accuracy (i.e., global convergence) with suitable hyper-parameters [3, 19]. However, prior theoretical works could not account for the global convergence of MAML trained with gradient descent on non-linear neural nets of more than two layers. Hence, a crucial question that remains unknown for MAML optimization is:

Can MAML with DNNs converge to global minima?

This question motivates us to analyze the optimization properties of MAML with DNNs, and we provide a positive answer with rigorous theoretical analysis. Briefly, for over-parameterized DNNs, we analyze the optimization trajectory of MAML with square loss and prove that the training loss is guaranteed to converge to zero at a linear rate. Additionally, in this convergence analysis, we find the DNN trained by MAML can be described by a kernel regression, with a novel class of kernels that we name as ***Meta Neural Tangent Kernels*** (MetaNTK).

One may wonder whether our theory has any practical implications. Intuitively, our theory reveals that MetaNTK is closely related to the performance of MAML. To demonstrate the practical value of our theory, we provide a concrete use case of MetaNTK: MetaNTK can help us efficiently find neural net architecture for few-shot learning.

Most meta-learning algorithms adopt standard network structures such as ConvNets [36], ResNets [22] and Wide ResNets [79] for few-shot image classification, the most

*equal contribution

popular task to benchmark meta-learning. However, these network structures were developed on supervised learning benchmarks such as CIFAR [35], and ImageNet [12], and recently, it has been shown that these popular structures actually *overfit* to the supervised learning task on these datasets [54]. This indicates that the popular network structures may not be optimal for tasks other than supervised learning, such as few-shot learning. Thus, one may naturally consider neural architecture search (NAS) [15, 42, 81] to automatically search for neural net architectures that are suitable for few-shot learning. To this end, prior works [16, 32, 40] designed NAS methods specific for few-shot learning, but they require substantial computational cost (e.g., the search cost of [32] and [16] on mini-ImageNet is 100 and 7 GPU days, respectively; the training of [40] takes 150 GPU days on miniImageNet), which makes them impractical for many real-world tasks and not environmental-friendly [13, 74]. Hence, a natural question is:

Can we accelerate NAS for few-shot learning to have much lower or even negligible search cost (compared to training cost)?

We provide an efficient solution to this quest, ***MetaNTK-NAS***, which is inspired by the MetaNTK we derive in our global convergence analysis of MAML. Briefly, we use the condition number of MetaNAK as an indicator for the trainability of networks under MAML. Since MetaNTK is directly computed at initialization, no training is needed in the search stage, leading to a surprisingly small search cost (e.g., less than 0.07 GPU day on mini-ImageNet).

Our main contributions are summarized below:

- **Global Convergence and Induced Kernels of MAML**[1]: We prove that with over-parameterized DNNs (i.e., DNNs with a large number of neurons in each layer), MAML is guaranteed to *converge to global optima with zero training loss* at a linear rate. The key to our proof is to develop bounds on the gradient of the MAML objective, and then analyze the optimization trajectory of DNN parameters trained under MAML. Furthermore, we show that in the over-parameterization regime, the output of MAML-trained networks becomes equivalent to the output of a special kernel regression with a new class of kernels, Meta Neural Tangent Kernels (MetaNTK).
- **Theory-Inspired Efficient NAS for Few-Shot Learning:** We propose MetaNTK-NAS, a new NAS method for few-shot learning that takes advantage of MetaNTK. Briefly, it uses the condition number of the MetaNTK of each network as an indicator for its trainability under meta-learning. Empirically, our MetaNTK-NAS is comparable or better than MetaNAS [16], the state-of-the-art NAS method for few-shot learning, on both miniImageNet and tieredImageNet, while consuming 100x less cost in the search process.

[1]This part was also presented in a prior tech report of ours [65].

2. Related Works

Meta-Learning Optimization. The MAML family (i.e., MAML [19] and its variants) is a popular approach for meta-learning. Several recent works theoretically analyze MAML or its variants in the case of *convex* objectives [5, 7, 20, 24, 31, 75]. However, neural nets are *non-convex*, so these works do not account for common practices of MAML with neural nets. On the other hand, [17, 28, 51, 80] consider the non-convex setting, but they only provide convergence to *stationary points*. Since *stationary points* can have high training/test error, the convergence to them is not very satisfactory. In particular, [67] proves the global convergence of MAML for a special class of *two-layer* networks that has frozen last layers with binary weight values. The unrealistic setting of [67] makes its results much weaker than our work, where our analysis is compatible with any depth and has no restriction on layer weight values. Notably, the MAML family shares similarities with multi-task learning from an optimization perspective [66].

Neural Tangent Kernels. Recently, there is a line of works studying the optimization of neural networks in the setting of supervised learning, e.g., [2, 14, 26, 30, 47], and many of them are restricted to two-layer networks. Notably, [26] proves that gradient flow on infinitely wide neural nets of any depth is guaranteed to converge to global optima, while its training dynamics can be described by kernel regression with neural tangent kernels (NTK). Further, [37] relaxes some assumptions of [26], and proves that gradient descent on finitely wide neural nets of any depth also enjoys global convergence as long as the width is large enough and the learning rate is sufficiently low. Notice that these works are tailored for supervised learning, thus it is unknown if global convergence is also guaranteed in other problems using neural networks. In this work, we leverage the tools of NTK from [26, 37] to analyze MAML in the few-shot learning setting, and our analysis can be easily generalized to other variants of MAML such as [9, 50, 51].

Neural Architecture Search. Neural Architecture Search (NAS) is proposed to automate neural architecture discovery to replace cumbersome manual designs for various deep learning tasks. Early works successfully utilize reinforcement learning [6, 81] and evolutionary algorithm [53] to find high-performance architectures. However, most of these methods are computationally expensive. To enable efficient architecture search, DARTS [42] proposed continuous relaxation of the architecture representation to allow search via gradient descent. Unfortunately, DARTS is hard to optimize and may suffer from performance degradation due to its weight-sharing strategy [68, 77]. To further accelerate architecture search, [44] proposed training-free NAS to evaluate randomly initialized architectures, thus fully eliminating neural network training during the search. Some following training-free methods propose to

search with NTK [11, 72], linear regions [11] and pruning-related criterion [1]. On the other hand, there are a few works applying NAS to few-shot learning using some meta-learning algorithms. [32] apply progressive neural architecture search to few-shot learning and [16,40] adopt DARTS-variants. But these approaches are very costly (e.g., [32,40] take more than 100 GPU days and [16] takes over 1 GPU week), thus it remains unexplored how to *efficiently* apply NAS to few-shot learning.

3. Preliminaries

Few-Shot Learning Consider a few-shot learning problem with a set of *training tasks* that contains N *supervised-learning tasks* $\{\mathcal{T}_i\}_{i=1}^N$. Each task is represented as

$$\mathcal{T}_i = (X_i, Y_i, X_i', Y_i') \in \mathbb{R}^{n\times d} \times \mathbb{R}^{nk} \times \mathbb{R}^{m\times d} \times \mathbb{R}^{mk},$$

where (X_i, Y_i) represents n *query* samples (i.e. test samples of $\mathcal{T}_i$) with corresponding labels, while (X_i', Y_i') represents m *support* samples (i.e. training samples of $\mathcal{T}_i$) with labels. Then, we denote

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

In few-shot learning, $\{\mathcal{T}_i\}_{i=1}^N$ are training tasks for meta-learners to train on (i.e., for meta-training). In the evaluation stage, an arbitrary test task $\mathcal{T} = (X, Y, X', Y')$ is picked, and the labeled support samples (X', Y') are given to the trained meta-learner as input, then the meta-learner is asked to predict the labels of the query samples X from $\mathcal{T}$. Notice that this few-shot learning problem above can also be called a n-shot k-way learning problem.

Neural Net Setup Consider a neural network f_θ with L hidden layers, where parameters $\theta \in \mathbb{R}^D$. For $i \in [L]$, we use l_i to denote the width of the i-th hidden layer. Without loss of generality, we consider all hidden layers have the same width[2], i.e., $l_1 = l_2 = \cdots = l_L = l$. We consider the parameters θ are Gaussian initialized[3], with details shown in Appendix A.

MAML The algorithm of MAML is shown in Algorithm 1. For simplicity, it shows MAML with one inner-loop update step, while our theory is compatible with arbitrary steps of inner-loop update (e.g., Line 5 of Algorithm 1 can be modified to have multiple gradient descent steps). For convenience, we define a *meta-output* function F as the output of the model f with adapted parameters. For MAML with one inner-loop step, the meta-output on arbitrary task $\mathcal{T} = (X, Y, X', Y')$ is

$$F_\theta(X, X', Y') = f_{\theta'}(X), \text{ s.t. } \theta' = \theta - \nabla_\theta \ell(f_\theta(X'), Y') \tag{1}$$

[2]This same-width assumption is not a necessary condition. One can also define $l = \min\{l_i\}_{i=1}^L$ instead and all theoretical results in this paper still hold true.

[3]Kaiming initialization [22]is also a kind of Gaussian initialization.

Algorithm 1 MAML for Few-Shot Learning (version of one inner-loop step)

Require: $\{\mathcal{T}_i\}_{i=1}^N$: Training Tasks
Require: η, λ: Learning rate hyperparameters
1: Randomly initialize θ
2: **while** not done **do**
3: **for all** $\mathcal{T}_i$ **do**
4: Evaluate the loss of f_θ on support samples of $\mathcal{T}_i$: $\ell(f_\theta(X_i'), Y_i')$
5: Compute adapted parameters θ_i' with gradient descent: $\theta_i' = \theta - \lambda\nabla_\theta \ell(f_\theta(X_i'), Y_i')$
6: Evaluate the loss of $f_{\theta_i'}$ on query samples of $\mathcal{T}_i$: $\ell(f_{\theta_i'}(X_i), Y_i)$
7: **end for**
8: Update parameters with gradient descent: $\theta \leftarrow \theta - \eta\nabla_\theta \sum_{i=1}^N \ell(f_{\theta_i'}(X_i), Y_i)$
9: **end while**

where $f_{\theta'}(X) = (f_{\theta'}(x))_{x\in X}$ is the concatenation of model outputs on all samples in X.

In this paper, we consider the square loss function $\ell(\hat{y}, y) = \frac{1}{2}\|\hat{y} - y\|_2^2$ for convenience. Then the MAML training loss is

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{2}\sum_{i\in[N]} \|F_\theta(X_i, X_i', Y_i') - Y_i\|_2^2 \\ &= \frac{1}{2}\|F_\theta(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y}\|_2^2 \end{aligned} \tag{2}$$

where $F_\theta(\mathcal{X}, \mathcal{X}', \mathcal{Y}') = (F_\theta(X_i, X_i', Y_i'))_{i=1}^N$ is the concatenation of meta-outputs.

Second-Order Gradient (Hessian) in MAML It is well known that gradient descent on the MAML objective (2) induces Hessian terms. For example, for MAML with one inner-loop step such as (1), the gradient of objective (2) is

$$\begin{aligned}\nabla_\theta \mathcal{L}(\theta) &= \sum_{i\in[N]} \nabla_\theta F_\theta(X_i, X_i', Y_i') \left(F_\theta(X_i, X_i', Y_i') - Y_i\right) \\ &= \sum_{i\in[N]} \left(I - \nabla_\theta^2 \ell(f_\theta(X'), Y')\right) \nabla_{\theta'} f_{\theta'}(X_i) \\ &\qquad \cdot \left(F_\theta(X_i, X_i', Y_i') - Y_i\right) \end{aligned} \tag{3}$$

where $\nabla_\theta^2 \ell(f_\theta(X'), Y') \in \mathbb{R}^{D\times D}$ is a Hessian term. Modern neural nets typically have millions of parameters, e.g., $D > 10^7$ in ResNet-12 [22] that is commonly used in recent few-shot learning works [38, 60]. Thus the Hessian terms usually have huge computation and memory cost.

4. Theoretical Results

Notation. For notational convenience, we denote the Jacobian of the meta-output on training data as $J(\theta) =$

$\nabla_\theta F_\theta(\mathcal{X}, \mathcal{X}', \mathcal{Y}')$, and define a kernel function,

$$\hat{\Phi}_\theta(\cdot, *) \coloneqq \frac{1}{l} \nabla_\theta F_\theta(\cdot) \nabla_\theta F_\theta(*)^\top , \tag{4}$$

which we name as *Meta Neural Tangent Kernel* function (MetaNTK). As the width l approaches infinity, for Gaussian randomly initialized parameters θ_0, the kernel function $\hat{\Phi}_{\theta_0}(\cdot, *)$ becomes a deterministic function independent of θ_0 (proved by Lemma 3 in Appendix B), denoted as

$$\Phi(\cdot, *) \coloneqq \lim_{l \to \infty} \hat{\Phi}_{\theta_0}(\cdot, *) \tag{5}$$

For convenience, we denote $F_t(\cdot) \triangleq F_{\theta_t}(\cdot)$, $f_t(\cdot) \triangleq f_{\theta_t}(\cdot)$ and $\hat{\Phi}_t(\cdot, *) \triangleq \hat{\Phi}_{\theta_t}(\cdot, *)$. For any diagonalizable matrix M, we use $\sigma_{\min}(M)$ and $\sigma_{\max}(M)$ to denote its least and largest eigenvalues.

4.1. Global Convergence of MAML

Suppose the neural network is sufficiently over-parameterized, i.e., the width of hidden layers, l, is large enough. Then, we prove that gradient descent on the MAML objective (2) is guaranteed to converge to global optima with zero training loss at a linear rate. The detailed setup, assumptions, and proof can be found in Appendix B. We provide a simplified theorem below.

Theorem 1 (Global Convergence). *Define* $\Phi = \lim_{l \to \infty} \frac{1}{l} J(\theta_0) J(\theta_0)^T$ *and* $\eta_0 = \frac{2}{\sigma_{max}(\Phi) + \sigma_{min}(\Phi)}$. *For arbitrarily small* $\delta > 0$, *and there exist* $R > 0$, $l^* \in \mathbb{N}$, *and* $\lambda_0 > 0$, *such that: for width* $l \geq l^*$, *running gradient descent with learning rates* $\eta < \frac{\eta_0}{l}$ *and* $\lambda < \frac{\lambda_0}{l}$ *over random initialization, the following upper bound on the training loss holds true with probability at least* $1 - \delta$*:*

$$\begin{aligned} \mathcal{L}(\theta_t) &= \frac{1}{2} \| F_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y} \|_2^2 \\ &\leq \left(1 - \frac{\eta_0 \sigma_{min}(\Phi)}{3} \right)^{2t} R . \end{aligned} \tag{6}$$

Main Proof Ideas Here, we depict the big picture of our global convergence proof for MAML with DNNs. To prove the global convergence of MAML with DNNs, we first obtain Lemma 1 (shown in Appendix B), which indicates that the Jacobian of the meta-output, J, changes locally in a small region under perturbations on initial network parameters. Then, we analyze the training dynamics of MAML and show that the parameter movement during training is confined in a small region. Hence, the Jacobian is stable across training, indicating the loss landscape is almost quadratic in the local neighborhood of initialization. Importantly, we find that for sufficiently large width, there definitely exists a global minimum with zero training loss inside the neighborhood of any parameter initialization θ_0. As a result, the almost quadratic loss landscape guarantees that gradient descent with a sufficiently small learning rate will reach this global minimum in the neighborhood at a linear rate.

Challenges Even though the big picture of our global convergence proof for MAML may look intuitive and simple, there exist several severe challenges in the analysis that do not appear in the supervised learning setting. Here we give one example of the challenges we deal with in the analysis: MAML is a bi-level optimization problem, and each (outer-loop) gradient descent step on its training objective (2) consists of (multiple) inner-loop gradient descent steps. Hence, the Jacobian (i.e., the gradient of the meta-output) consists of Hessian (i.e., second-order gradient) terms that do not exist in supervised learning models, and these Hessian terms appear in the form of matrix exponentials. In order to prove the local stability of the Jacobian (i.e., Lemma 1), we utilize non-trivial theoretical tools on matrix exponentials [29, 61] that can tackle this challenge.

4.2. Equivalence between MAML and Kernels

Analytical Expression of Meta-Output. In the setting of Theorem 1, the training dynamics of the MAML can be described by a differential equation

$$\frac{dF_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}')}{dt} = -\eta \hat{\Phi}_0 (F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y}) \tag{7}$$

where we denote $\hat{\Phi}_0 \equiv \hat{\Phi}_{\theta_0}((\mathcal{X}, \mathcal{X}', \mathcal{Y}'), (\mathcal{X}, \mathcal{X}', \mathcal{Y}'))$ and $F_t \equiv F_{\theta_t}$ for convenience. Notice that (7) is a first-order ODE. The solution of this ODE gives rise to the analytical expression of F_t on arbitrary task $\mathcal{T} = (X, Y, X', Y')$,

$$\begin{aligned} F_t(X, X', Y') = F_0(X, X', Y') \\ + \hat{\Phi}_0(X, X', Y') T^\eta_{\hat{\Phi}_0}(t) \left(\mathcal{Y} - F_0(\mathcal{X}, \mathcal{X}', \mathcal{Y}') \right) \end{aligned} \tag{8}$$

where $\hat{\Phi}_0(\cdot) \equiv \hat{\Phi}_{\theta_0}(\cdot, (\mathcal{X}, \mathcal{X}', \mathcal{Y}'))$ and $T^\eta_{\hat{\Phi}_0}(t) = \hat{\Phi}_0^{-1} \left(I - e^{-\eta \hat{\Phi}_0 t} \right)$ are shorthand notations.

Here, we provide a formal definition of NTK [26, 37], which will be used shortly.

Definition 1 (NTK). *For any neural net function* $f : \mathbb{R}^k \mapsto \mathbb{R}$ *with randomly initialized parameters* θ_0, *its neural tangent kernel function is defined as* $\hat{\Theta}_0(\cdot, *) \equiv \hat{\Theta}_{\theta_0}(\cdot, *) \coloneqq \nabla_{\theta_0} f_{\theta_0}(\cdot) \nabla_{\theta_0} f_{\theta_0}(*)^\top$, *and it converges to a deterministic kernel as the width* l *approaches infinity [26, 37],*

$$\Theta(\cdot, *) = \lim_{l \to \infty} \hat{\Theta}_0 \tag{9}$$

Below, we provide an analytical expression of Meta Neural Tangent Kernel (MetaNTK) for MAML in the infinite width limit, indicating that the kernel function Φ defined in (5) can be equivalently viewed as a composite kernel function built upon NTK function $\Theta(\cdot, *)$. The derivation of this expression can be found in Lemma 3 in Appendix B.

Definition 2 (MetaNTK in the Infinite Width Limit)**.** *As the width l approaches infinity, MetaNTK can be expressed as $\Phi \equiv \Phi((\mathcal{X}, \mathcal{X}'), (\mathcal{X}, \mathcal{X}')) \in \mathbb{R}^{knN \times knN}$, which is a block matrix that consists of $N \times N$ blocks of size $kn \times kn$. For $i, j \in [N]$, its (i, j)-th block is*

$$[\Phi]_{ij} = \phi((X_i, X_i'), (X_j, X_j')) \in \mathbb{R}^{kn \times kn}, \tag{10}$$

where $\phi : (\mathbb{R}^{n \times k} \times \mathbb{R}^{m \times k}) \times (\mathbb{R}^{n \times k} \times \mathbb{R}^{m \times k}) \to \mathbb{R}^{nk \times nk}$ is a kernel function defined as

$$\begin{aligned} &\phi((\cdot, *), (\bullet, \star)) \\ &= \Theta(\cdot, \bullet) + \Theta(\cdot, *)\widetilde{T}_\Theta^\lambda(*, \tau)\Theta(*, \star)\widetilde{T}_\Theta^\lambda(\star, \tau)^\top \Theta(\star, \bullet) \\ &- \Theta(\cdot, *)\widetilde{T}_\Theta^\lambda(*, \tau)\Theta(*, \bullet) - \Theta(\cdot, \star)\widetilde{T}_\Theta^\lambda(\star, \tau)^\top \Theta(\star, \bullet). \end{aligned} \tag{11}$$

The following theorem shows that as the width of neural nets approaches infinity, MAML becomes equivalent to a special kernel regression with the MetaNTK of Definition 2. The proof is provided in Appendix D.

Theorem 2 (MAML as Kernel Regression)**.** *Suppose learning rates η and λ are infinitesimal. As the network width l approaches infinity, with high probability over random initialization of the neural net, the MAML output, (8), converges to a special kernel regression,*

$$\begin{aligned} F_t(X, X', Y') &= G_\Theta^\tau(X, X', Y') \\ &+ \Phi((X, X'), (\mathcal{X}, \mathcal{X}'))T_\Phi^\eta(t)\left(\mathcal{Y} - G_\Theta^\tau(\mathcal{X}, \mathcal{X}', \mathcal{Y}')\right) \end{aligned} \tag{12}$$

where G is a function defined below.

$$G_\Theta^\tau(X, X', Y') = \Theta(X, X')\widetilde{T}_\Theta^\lambda(X', \tau)Y'. \tag{13}$$

with $\widetilde{T}_\Theta^\lambda(\cdot, \tau) \triangleq \Theta(\cdot, \cdot)^{-1}(I - e^{-\lambda\Theta(\cdot,\cdot)\tau})$. Besides, $G_\Theta^\tau(\mathcal{X}, \mathcal{X}', \mathcal{Y}') = (G_\Theta^\tau(X_i, X_i', Y_i'))_{i=1}^N$.

The $\Phi((X, X'), (\mathcal{X}, \mathcal{X}')) \in \mathbb{R}^{kn \times knN}$ in (12) is also a block matrix, which consists of $1 \times N$ blocks of size $kn \times kn$, with the $(1, j)$-th block as follows for $j \in [N]$,

$$[\Phi((X, X'), (\mathcal{X}, \mathcal{X}'))]_{1,j} = \phi((X, X'), (X_j, X_j')).$$

Remarks. The kernel Φ is what $\hat{\Phi}_0$ converges to as the network width approaches infinity. However, $\hat{\Phi}_0 \equiv \hat{\Phi}_0((\mathcal{X}, \mathcal{X}', \mathcal{Y}'), (\mathcal{X}, \mathcal{X}', \mathcal{Y}'))$ depends on $\mathcal{Y}$ and $\mathcal{Y}'$, while $\Phi \equiv \Phi((\mathcal{X}, \mathcal{X}'), (\mathcal{X}, \mathcal{X}'))$ does not, since the terms in Φ depending on $\mathcal{Y}$ or $\mathcal{Y}'$ all vanish as the width approaches infinity. Although (12) is a sum of two kernel regression terms, it can still be viewed as a single special kernel regression relying on MetaNTK Φ. Notably, Φ can be seen as a *composite* kernel built upon the *base* kernel function Θ.

5. MetaNTK-NAS

Recently, there has been a series of *training-free NAS* works that reduce the search cost of NAS by directly searching over untrained candidate structures [11, 41, 44, 73]. The key of these works is to measure certain properties or metrics of untrained networks that are correlated with the final training/test accuracy. Among them, TE-NAS [11] utilizes two theory-motivated metrics of untrained networks to achieve training-free NAS: (*i*) the condition number of the neural tangent kernel (NTK), which can be seen as an indicator for the trainability of the network under supervised learning; (*ii*) the number of linear regions in the input space, which implies the expressivity (i.e., representation power) of the network. Empirical studies of [11] confirm that these two metrics of untrained networks are correlated with the test accuracy of trained networks with the same structures.

Since few-shot learning is quite different from supervised learning, metrics specific to supervised learning may not be suitable for few-shot learning. For example, NTK is derived in the supervised learning setting, thus the use of condition number of NTK in TE-NAS [11] might be ineffective in few-shot learning. A natural candidate for few-shot learning is the MetaNTK we derive in Theorem 2, which can be viewed as the counterpart of NTK in meta-learning, thus it is a natural idea to replace NTK with MetaNTK in TE-NAS to accommodate the few-shot learning task, which is the core of our proposed MetaNTK-NAS.

Briefly speaking, the main idea of our MetaNTK-NAS is to compute an importance score for each untrained can-

Algorithm 1: MetaNTK-NAS: Training-Free NAS for Few-Shot Learning.

Input: Search step $k = 0$. A initial supernet $\mathcal{N}_0$ of stacked cells. Each cell has E edges. Each edge has $|\mathcal{O}|$ operators.
while $\mathcal{N}_k$ *is not a single-path network* **do**
  for *each operator* o_j *in* $\mathcal{N}_k$ **do**
    $\Delta C_{k,o_j} = C_{\mathcal{N}_k} - C_{\mathcal{N}_k \backslash o_j}$
    (The higher $\Delta C_{t,o_j}$, the more likely we will prune o_j)
    $\Delta R_{t,o_j} = R_{\mathcal{N}_k} - R_{\mathcal{N}_k \backslash o_j}$
    (The lower $\Delta R_{t,o_j}$, the more likely we will prune o_j)
  Get importance score of $C_{\mathcal{N}}$: $s_C(o_j) =$ index of o_j in descendingly sorted list $[\Delta C_{t,o_1}, ..., \Delta C_{t,o_{|\mathcal{N}_k|}}]$
  Get importance score of $R_{\mathcal{N}}$: $s_R(o_j) =$ index of o_j in ascendingly sorted list $[\Delta R_{t,o_1}, ..., \Delta R_{t,o_{|\mathcal{N}_k|}}]$
  Get total importance score $s(o_j) = s_C(o_j) + s_R(o_j)$
  $\mathcal{N}_{k+1} \leftarrow \mathcal{N}_k$
  for *each edge* e_i, $i = 1, ..., E$ **do**
    $j^* = \text{argmin}_j\{s(o_j) : o_j \in e_i\}$
    (Find the operator with the least importance score on each edge.)
    $\mathcal{N}_{k+1} = \mathcal{N}_{k+1} \backslash o_{j^*}$
  $k \leftarrow k + 1$
return *Single-path network* $\mathcal{N}_k$.

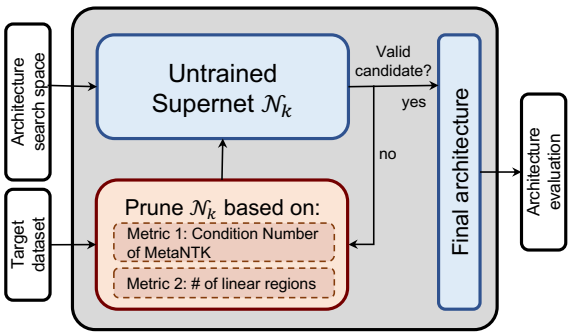


Figure 1. Illustration of our MetaNTK-NAS (Algorithm 1).

didate structure by the condition number of MetaNTK and the number of linear regions in the input space, then search for a structure that maximizes this score.

Specifically, for each candidate structure $\mathcal{N}$ with parameters θ_0 randomly initialized by Kaiming Initialization [22], we sample a batch of training tasks $\{(X_i, Y_i, X_i', Y_i')\}$ from the training set, where $X_i \in \mathbb{R}^{n\times k}, X_i' \in \mathbb{R}^{m\times k}$ are n query samples and m support samples, respectively. Then, we compute the MetaNTK of the network based on the analytical expression derived in Eqs. (10) and (11). Notice that Eq. (11) demonstrates the MetaNTK Φ is a composite kernel built upon the NTK kernel function in the infinite width limit. We assume this holds approximately true for finitely wide networks, and use the formula (11) to build MetaNTK $\hat{\Phi}_{\theta_0}$ from NTK kernel function $\hat{\Theta}_{\theta_0}$. This approach has a computational benefit: if we compute MetaNTK following the definition in (4), that will involve second-order gradients; however, building MetaNTK from NTK following (11) could get rid of this burden since $\hat{\Theta}_{\theta_0}$ is computed purely from first-order gradients. Also, $\hat{\Phi}_{\theta_0}$ is positive definite[4], thus its eigenvalues are all positive, and we can define its condition number to be

$$C_{\mathcal{N}} := \frac{\sigma_{\max}(\Phi_{\theta_0})}{\sigma_{\min}(\Phi_{\theta_0})} \tag{14}$$

We compute another metric following Definition 1 of [11],

$$R_{\mathcal{N}} := \text{number of linear regions} \tag{15}$$

The condition number $C_{\mathcal{N}}$ indicates the trainability[5] of the network structure (the lower, the better) under meta-learning, and $R_{\mathcal{N}}$ stands for the representation power of the structure (the higher, the better). Our Algorithm 1 is designed to minimize $C_{\mathcal{N}}$ and maximize $R_{\mathcal{N}}$ over candidate structures in the search space, following the algorithm design of TE-NAS [11]. An illustration of Algorithm 1 is provided in Fig. 1.

[4]Since NTK is positive definite [26] and we build MetaNTK from NTK in a way that preserves the positive definiteness.

[5]The connection between the condition number of NTK and trainability is discussed in Sec. 3.1.1 of [11]

6. Experiments

6.1. Experiment Setup

We conduct experiments on two popular few-shot image classification datasets, mini-ImageNet and tiered-ImageNet, which both are subsets of ImageNet [12]. And our experiments consist of three stages: search, train and evaluate. Here we give an overview of the datasets.

- *mini-ImageNet* [64]: It contains 60,000 RGB images of 84x84 pixels extracted from ImageNet [12]. It includes 100 classes (each with 600 images) that are split into 64 training classes, 16 validation classes and 20 test classes.
- *tiered-ImageNet* [55]: This dataset contains 779,165 RGB images of 84x84 pixels extracted from ImageNet [12]. It includes 608 classes that are split into 351 training, 97 validation and 160 test classes.

Search Space Following the NAS literature [11, 16, 42], we search for a normal cell and a reduction cell as the building block of the final architecture. Both cells have three intermediate nodes. MetaNAS [16] uses a modified version of standard DARTS search space. We use the same search space as MetaNAS for a fair comparison. Specifically, the set of candidate operations include *MaxPool3x3, AvgPool3x3, SkipConnect, Conv1x5-5x1, Conv3x3, SepConv3x3* and *DilatedConv3x3*.

Implementation Details The neural network we use is obtained by stacking 5 or 8 searched cells together. Cells located at the 1/3 and 2/3 of the total depth of the network[6] are reduction cells, where we decrease the spatial resolution and double the number of channels[7]. We set the initial number of channels to 48[8]. To search for the candidate cells, we start with a supernetwork $\mathcal{N}_0$ composed of all possible edges and operations. We follow TE-NAS [11] to prune one operator on each edge by its importance per round. The importance of each operator is measured by the change of condition number of MetaNTK $C_{\mathcal{N}}$ and the number of linear regions $R_{\mathcal{N}}$ before and after being pruned. We repeat the process until the supernetwork $\mathcal{N}_k$ becomes a single-path network. We summarize our algorithm in Algorithm 1.

To evaluate the architecture searched by MetaNTK-NAS, we train the network on the same dataset where the search is conducted. For the training, we keep the same number of cells and the initial number of channels used in the search stage. As for the training recipe, we use RFS [60] (without their additional knowledge distillation trick), an FSL method in the fashion of pre-

[6]The network has an output layer in addition to cells, e.g., a 5-cells network has $5 + 1 = 6$ layers in total.

[7]This is different from the original setup of MetaNAS [16], where they fix number of channels throughout the whole network.

[8]We use the same number of initial channels and number of cells during search and evaluation when computing MetaNTK, which is different from TE-NAS [11]. TE-NAS uses a small proxy network to compute NTK.

Model	Arch.	#Cells	Train	mini-ImageNet 5-way				tiered-ImageNet 5-way			
				#Param.	Search Cost	1-shot	5-shot	#Param.	Search Cost	1-shot	5-shot
MAML [19]	Conv4	-	MAML	30k	-	48.70±1.84	63.11±0.92	30k	-	51.67±1.81	70.30±1.75
ANIL [50]	Conv4	-	ANIL	30k	-	48.0±0.7	62.2±0.5	-	-	-	-
MetaOptNet [38]	ResNet-12	-	MetaOptNet	12.5M	-	62.64±0.61	78.63±0.46	12.7M	-	65.99±0.72	81.56±0.53
RFS [60]	ResNet-12	-	RFS	12.5M	-	62.02±0.63	79.64±0.44	12.7M	-	69.74±0.72	84.41±0.55
AutoMeta [32]	Cells	-	Reptile	100k	2688 hr	57.6±0.2	74.7±0.2	-	-	-	-
T-NAS++ [40]	Cells	2	FOMAML	27k	48 hr	54.11±1.35	69.59±0.85	-	-	-	-
MetaNAS [16]	Cells	5	Reptile	1.1M	168 hr	63.1±0.3	79.5±0.2	-	-	-	-
MetaNAS(retrained)†	Cells	5	RFS	2.01M	168 hr	**64.24±0.11**	79.75±0.13	2.17M	168 hr	70.16±0.09	84.99±0.22
MetaNTK-NAS	Cells	5	RFS	1.77M	1.54 hr	63.88±0.81	**80.07±0.45**	2.22M	2.20 hr	**71.12±0.49**	**85.71±0.22**
MetaNAS(retrained)†	Cells	8	RFS	3.53M	168 hr	63.88±0.23	79.88±0.14	3.70M	168 hr	**72.32±0.02**	**86.48±0.06**
MetaNTK-NAS	Cells	8	RFS	3.21M	1.92 hr	**64.26±0.14**	**80.35±0.12**	4.78M	2.73 hr	**72.37±0.79**	**86.43±0.53**

Table 1. **Comparison on few-shot image classification benchmarks.** Average few-shot test classification accuracy (%) ± standard deviation. The first 4 rows are few-shot learning algorithms on standard networks, and the following 3 rows (AutoMeta, T-NAS++, MetaNAS) are prior NAS methods designed for few-shot learning. The last 4 rows are from our experiments: we stack cells presented by MetaNAS [16] and searched by our MetaNTK-NAS in the same way (stacks of 5 cells or 8 cells), and train both of them with RFS [60]. † (*i*) The authors of MetaNAS [16] only present one cell structure that they manually select over multiple structures in the search process on miniImageNet, and train it for 3 runs. Thus this structure can be seen as the *best* structure they obtain. In contrast, we run the search-train-evaluate pipeline of MetaNTK-NAS for 3 independent runs and take average of the test accuracy. (*ii*) [16] does not run on tieredImageNet, thus we stack the cells they searched on miniImageNet and train the structure on tieredImageNet using RFS. We believe MetaNAS cells work relatively well on tiered-ImageNet because mini- and tiered-ImageNet are both subsets of ImageNet, sharing lots of similarities.

training+finetuning, since it is an efficient first-order optimization method. RFS greatly reduces the training time, and GPU memory compared to higher-order optimization methods such as MAML [19]. To compare our searched cells with MetaNAS, we evaluate cells found by MetaNAS[9] with the same RFS training-evaluate pipeline and present as MetaNTK(retrained) in Table 1.

Hyper-parameters During the search phase, for computing NTK, MetaNTK, and the number of linear regions, we adopt data augmentation used in TE-NAS [11]. In addition to the MAML-induced MetaNTK (MAML-kernel) we derive in Theorem 2, we also implement another MetaNTK induced by ANIL (ANIL-kernel) [50], a simplified version of MAML. We use MAML-kernel and ANIL-kernel for the 5-cells and 8-cells experiments, respectively. More details about hyperparameters such as batch size, dropout rate [58] and normalization [25, 70] can be found in Appendix E.

Optimization Setup Following [60], we adopt SGD optimizer with a momentum of 0.9 and a weight decay of 0.0005. We train all models for 100 epochs and 60 epochs on miniImageNet and tieredImagenet, respectively. For miniImageNet, the learning rate is 0.02 initially, and it is decayed by 10x at epoch 60 and 80. For tieredImageNet, the learning rate is 0.01 initially, and it is decayed by 10x at epochs 30, 40, 50.

Model Selection. We always take the model checkpoints at the end of training for evaluation.

Evaluation on Test Tasks Following [60], for any test task, we remain hidden layers intact and finetune the linear output layer of each network on labeled support samples, and then evaluate its prediction accuracy on the query samples as the test accuracy. Similar to [60], we mostly use ℓ_2 regularized cross-entropy loss to finetune the linear layer, while we also use ℓ_2 regularized hinge loss in some 1-shot cases. More details regarding the setup and hyperparameters of the evaluation stage can be found in Appendix E.

Code Our code is written in PyTorch [48]. For the search stage of NAS, we build our code upon the released codebase of [11]. Opacus [76] is used to compute per-sample-gradients efficiently to further construct MetaNTK. For the training and evaluation stages, we adopt the code of [60].

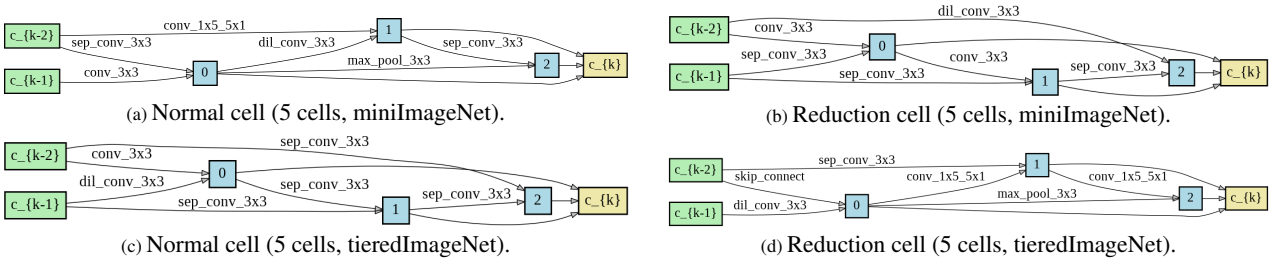
Hardware Most of our experiments were run on NVIDIA V100s, and the rest were run on NVIDIA RTX 3090s. Each experiment is run on a single GPU at a time. The search cost of MetaNTK-NAS is benchmarked on V100s.

6.2. Experiment Results

Empirical Results Each experiment of MetaNTK-NAS and MetaNAS(retrained) is repeated over at least 3 runs of different random seeds. We evaluate each trained model on 1000 test tasks randomly sampled from the test set, and report the mean and standard deviation of the test accuracy.

Performance Comparison In Table 1, we compare MetaNTK-NAS with multiple meta-learning algorithms on miniImageNet and tieredImageNet under 5-way few-shot classification setting. Among these algorithms, MetaOptNet is the state-of-the-art gradient-based meta-learning algorithm, and RFS is an efficient few-shot learning algorithm, while AutoMeta, T-NAS++, and MetaNAS are existing NAS algorithms specifically designed for few-shot learning. Compared to MetaOptNet and RFS, which use

[9] We evaluate the cells used in their large-scale experiments.



(a) Normal cell (5 cells, miniImageNet).

(b) Reduction cell (5 cells, miniImageNet).

(c) Normal cell (5 cells, tieredImageNet).

(d) Reduction cell (5 cells, tieredImageNet).

Figure 2. Examples of normal and reduction cells found by MetaNTK-NAS that are used for the evaluation. Green, blue and yellow boxes denote outputs of the previous cells, intermediate nodes and output of the current cell, respectively. See [42] for more details.

Model	MetaNTK	NTK	Linear Region	1-shot 5 Cells	1-shot 8 Cells	5-shot 5 Cells	5-shot 8 Cells
MetaNTK-NAS	✓		✓	**63.88 ± 0.81**	**64.26 ± 0.14**	**80.07 ± 0.45**	**80.35 ± 0.12**
TE-NAS [11]		✓	✓	62.51 ± 0.42	63.51 ± 0.16	79.02 ± 0.35	79.86 ± 0.42
Linear Region			✓	62.96 ± 1.05	63.99 ± 0.76	79.13 ± 0.96	79.88 ± 0.51
Random Cell				62.55 ± 1.15	62.76 ± 0.83	78.90 ± 0.58	79.18 ± 0.72

Table 2. **Ablation study on mini-ImageNet.** The row "Linear Region" stands for NAS with only the number of linear regions. The row "Random Cell" evaluates stacks of randomly sampled candidate cells from the search space.

ResNet-12 as backbone architecture, our method outperforms them with a much fewer number of parameters (2.5x∼6x fewer). Besides, our MetaNTK-NAS achieves comparable or better performance than MetaNAS, the state-of-the-art NAS algorithm for few-shot learning across various settings on the two datasets (cf. the last 4 rows of Tab. 1). Examples of searched cells are also shown in Fig. 2.

Efficiency Comparison While achieving competitive performance, our method enjoys *100x* faster search speed (1.54 GPU hours search cost compared to 168 GPU hours search cost of MetaNAS on miniImageNet for 5-cells structures). Since MetaNAS cells are obtained by training supernetwork in the small-scale regime of 100k parameters on miniImageNet, we believe that comparing our search cost of 5-cells setup on miniImageNet to theirs is fair. We also believe MetaNTK-NAS is likely to have an even larger improvement in search speed compared to MetaNAS on tieredImageNet[10] or larger datasets. Therefore, we can conclude that *MetaNTK-NAS is comparable to or better than the state-of-the-art of NAS methods for few-shot learning with 100x faster search speed.*

6.3. Ablation Experiments

In this section, we conduct ablation studies to analyze the effects of different ingredients used in MetaNTK-NAS. We compare the following setups with different combinations of components: (a) we conduct a search with TE-NAS [11], which searches with NTK and number of linear regions[11]; (b) we conduct a search using only the number of linear regions (no NTK or meta-NTK is used); (c) we conduct a random search, where we randomly sample candidate cells from the search space.

Tab. 2 shows the results of our ablation study on mini-ImageNet. The improvement of MetaNTK-NAS over the rest methods indicates the usefulness of the MetaNTK condition number in NAS for few-shot learning. Notice that the NTK condition number seems to have adverse effects on the performance. It indicates that NTK, which is derived in supervised learning, does not fit few-shot learning well.

Conclusion

In this paper, we first focus on the *optimization* properties of *model-agnostic meta-learning* (MAML) equipped with deep neural networks (DNNs), and prove the *global convergence* of MAML with over-parameterized deep neural nets. Based on the convergence analysis, we prove that in the infinite width limit of DNNs, MAML converges to a kernel regression with a new class of kernels, which we name as *Meta Neural Tangent Kernel* (MetaNTK). Inspired by recent works that apply Neural Tangent Kernel (NTK) to NAS for supervised learning [11, 72], we propose MetaNTK-NAS, a new NAS method for few-shot learning based on our derived MetaNTK. Empirically, we compare MetaNTK-NAS with prior works, and observe that the performance of MetaNTK-NAS is comparable or better than the state-of-the-art NAS methods for few-shot learning on miniImageNet and tieredImageNet, while enjoying 100x less search cost.

Acknowledgement

This work is partially supported by NSF grant No.1910100 and NSF CNS 20-46726 CAR. This work utilizes resources supported by NSF Major Research Instrumentation program, grant No.1725729 [33].

[10] The experiments of the original MetaNAS paper [16] is limited to miniImageNet and Omniglot (a much smaller dataset).

[11] We use the same number of training samples to compute MetaNTK and NTK for a fair comparison.

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021. 3

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *International Conference on Machine Learning*, 2019. 2

[3] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml, 2018. 1

[4] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *NeurIPS*, 2019. 14, 17, 22, 23, 26, 27

[5] Yu Bai, Minshuo Chen, Pan Zhou, Tuo Zhao, Jason Lee, Sham Kakade, Huan Wang, and Caiming Xiong. How important is the train-validation split in meta-learning? In *International Conference on Machine Learning*, pages 543–553. PMLR, 2021. 2

[6] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 2

[7] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433, 2019. 2

[8] Trapit Bansal, Rishikesh Jha, and Andrew McCallum. Learning to few-shot learn across diverse natural language classification tasks, 2019. 1

[9] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. 2

[10] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pages 10835–10845, 2019. 17, 21

[11] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four {gpu} hours: A theoretically inspired perspective. In *International Conference on Learning Representations*, 2021. 3, 5, 6, 7, 8

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 2, 6

[13] Payal Dhar. The carbon impact of artificial intelligence. *Nature Machine Intelligence*, 2(8):423–425, 2020. 2

[14] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *International Conference on Machine Learning*, 2019. 2

[15] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019. 2

[16] Thomas Elsken, Benedikt Staffler, Jan Hendrik Metzen, and Frank Hutter. Meta-learning of neural architectures for few-shot learning. In *CVPR*, 2020. 2, 3, 6, 7, 8

[17] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092, 2020. 2

[18] Chelsea Finn. *Learning to Learn with Gradients*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2018. 1

[19] 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-Volume 70*, pages 1126–1135. JMLR. org, 2017. 1, 2, 7

[20] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930, 2019. 1, 2

[21] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018. 1

[22] 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*, pages 770–778, 2016. 1, 3, 6

[23] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020. 1

[24] Yifan Hu, Siqi Zhang, Xin Chen, and Niao He. Biased stochastic gradient descent for conditional stochastic optimization. *arXiv preprint arXiv:2002.10790*, 2020. 2

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 7, 28

[26] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018. 2, 4, 6, 17, 22, 23, 26, 27

[27] Arthur Jacot, Franck Gabriel, and Clement Hongler. The asymptotic spectrum of the hessian of dnn throughout training. In *International Conference on Learning Representations*, 2020. 16

[28] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Multi-step model-agnostic meta-learning: Convergence and improved algorithms. *arXiv preprint arXiv:2002.07836*, 2020. 2

[29] Bo Kågström. Bounds and perturbation bounds for the matrix exponential. *BIT Numerical Mathematics*, 17(1):39–57, 1977. 4, 21

[30] Kenji Kawaguchi and Leslie Pack Kaelbling. Elimination of all bad local minima in deep learning. *arXiv preprint arXiv:1901.00279*, 2019. 2

[31] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pages 5915–5926, 2019. 1, 2

[32] Jaehong Kim, Sangyeul Lee, Sungwan Kim, Moonsu Cha, Jung Kwon Lee, Youngduck Choi, Yongseok Choi, Dong-Yeon Cho, and Jiwon Kim. Auto-meta: Automated gradient based meta learner search. *arXiv preprint arXiv:1806.06927*, 2018. 2, 3, 7

[33] Volodymyr Kindratenko, Dawei Mu, Yan Zhan, John Maloney, Sayed Hadi Hashemi, Benjamin Rabe, Ke Xu, Roy Campbell, Jian Peng, and William Gropp. *HAL: Computer System for Scalable Deep Learning*. Association for Computing Machinery, New York, NY, USA, 2020. 8

[34] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[35] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 2

[36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015. 1

[37] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019. 2, 4, 14, 16, 17, 18, 19, 20, 21, 22, 23, 26, 27

[38] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019. 3, 7, 27

[39] Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016. 1

[40] Dongze Lian, Yin Zheng, Yintao Xu, Yanxiong Lu, Leyu Lin, Peilin Zhao, Junzhou Huang, and Shenghua Gao. Towards fast adaptation of neural architectures with meta learning. In *International Conference on Learning Representations*, 2020. 2, 3, 7

[41] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance deep image recognition. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, 2021. 5

[42] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 2, 6, 8, 28

[43] Yunan Luo, Jianzhu Ma, Xiaoming Trey Ideker, Jian Zhao, Peng1 YufengB Su, and Yang Liu. 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, page 305. Springer, 2019. 1

[44] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021. 2, 5

[45] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017. 1

[46] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 1

[47] Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks, 2019. 2

[48] 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. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019. 7

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 28

[50] 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*, 2020. 2, 7, 27

[51] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019. 1, 2

[52] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016. 1

[53] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Aging evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, volume 2, 2019. 2

[54] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019. 2

[55] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018. 6

[56] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016. 1

[57] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 1

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 7

[59] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998. 1

[60] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *ECCV*, 2020. 3, 6, 7, 27, 28

[61] Charles Van Loan. The sensitivity of the matrix exponential. *SIAM Journal on Numerical Analysis*, 14(6):971–981, 1977. 4, 21

[62] Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018. 1

[63] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016. 1

[64] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016. 6

[65] Haoxiang Wang, Ruoyu Sun, and Bo Li. Global convergence and generalization bound of gradient-based meta-learning with deep neural nets. *arXiv preprint arXiv:2006.14606*, 2020. 2

[66] Haoxiang Wang, Han Zhao, and Bo Li. Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation. In *International Conference on Machine Learning*. PMLR, 2021. 2

[67] Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. On the global optimality of model-agnostic meta-learning. In *International Conference on Machine Learning*, pages 101–110, 2020. 2

[68] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representations*, 2020. 2

[69] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 2019. 1

[70] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 7, 28

[71] Lechao Xiao, Jeffrey Pennington, and Samuel S Schoenholz. Disentangling trainability and generalization in deep learning. *ICML*, 2020. 21

[72] Jingjing Xu, Liang Zhao, Junyang Lin, Rundong Gao, Xu Sun, and Hongxia Yang. Knas: green neural architecture search. In *International Conference on Machine Learning*, pages 11613–11625. PMLR, 2021. 3, 8

[73] Jingjing Xu, Liang Zhao, Junyang Lin, Rundong Gao, Xu Sun, and Hongxia Yang. Knas: green neural architecture search. In *International Conference on Machine Learning*, pages 11613–11625. PMLR, 2021. 5

[74] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. A survey on green deep learning, 2021. 2

[75] Ruitu Xu, Lin Chen, and Amin Karbasi. Meta learning in the continuous time limit. *arXiv preprint arXiv:2006.10921*, 2020. 2

[76] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in pytorch, 2021. 7, 28

[77] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2019. 2

[78] Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 1

[79] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

[80] Pan Zhou, Xiaotong Yuan, Huan Xu, and Shuicheng Yan. Efficient meta learning via minibatch proximal update. *Neural Information Processing Systems*, 2019. 2

[81] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2017. 2