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
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 environmentally-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**
  - 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].
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 $\{T_i\}_{i=1}^N$. Each task is represented as $T_i = (X_i, Y_i, X'_i, Y'_i) \in \mathbb{R}^{n \times d} \times \mathbb{R}^{m \times k}$, $X_i$ represents $n$ query samples (i.e. test samples of $T_i$) with corresponding labels, while $(X'_i, Y'_i)$ represents $m$ support samples (i.e. training samples of $T_i$) with labels. Then, we denote $X = (X_i)_{i=1}^N, Y = (Y_i)_{i=1}^N, X' = (X'_i)_{i=1}^N, Y' = (Y'_i)_{i=1}^N$. In few-shot learning, $\{T_i\}_{i=1}^N$ are training tasks for meta-learners to train on (i.e., for meta-training). In the evaluation stage, an arbitrary test task $T = (X, Y, X', Y')$ 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 $T$. 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 $f_\theta$ 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$, i.e., l_1 = l_2 = \cdots = l_L = l$. We consider the parameters $\theta$ are Gaussian initialized, 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 $T = (X, Y, X', Y')$ is $F_\theta(X, X', Y') = f_{\theta'}(X)$, s.t. $\theta' = \theta - \nabla_\theta \ell(f_{\theta'}(X'), Y')$

\begin{equation}
F_\theta(X, X', Y') = f_{\theta'}(X), \text{ s.t. } \theta' = \theta - \nabla_\theta \ell(f_{\theta'}(X'), Y')
\end{equation}

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

Require: $\{T_i\}_{i=1}^N$: Training Tasks
Require: $\eta, \lambda$: Learning rate hyperparameters
1: Randomly initialize $\theta$
2: while not done do
3:     for all $T_i$ do
4:         Evaluate the loss of $f_\theta$ on support samples of $T_i$: $\ell(f_\theta(X_i), Y_i)$
5:         Compute adapted parameters $\theta'_i$ 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 $T_i$: $\ell(f_{\theta'_i}(X_i), Y_i)$
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

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

$\nabla_\theta \mathcal{L}(\theta) = \sum_{i \in [N]} \nabla_\theta F_\theta(X_i, X'_i, Y'_i) (F_\theta(X_i, X'_i, Y'_i) - Y_i)$

\begin{equation}
\sum_{i \in [N]} \nabla_\theta \mathcal{L}(\theta) = \sum_{i \in [N]} (I - \nabla_\theta^2 \ell(f_\theta(X'), Y')) \nabla_\theta f_{\theta_i}(X_i)
\end{equation}

\begin{equation}
\cdot (F_\theta(X_i, X'_i, Y'_i) - Y_i)
\end{equation}

where $\nabla_\theta \mathcal{L}(\theta)$ 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 $J(\theta) =$

\begin{equation}
J(\theta) = \frac{\partial f_{\theta'}(X)}{\partial \theta}
\end{equation}
\[ \nabla_{\theta} F_{\theta}(\mathcal{X}, \mathcal{X}', \mathcal{Y}'), \] and define a kernel function,
\[ \hat{\Phi}_{\theta}(\cdot, \cdot) := \frac{1}{l} \nabla_{\theta} F_{\theta}(\cdot) \nabla_{\theta} F_{\theta}(\cdot)^T, \] (4)
which we name as Meta Neural Tangent Kernel function (MetaNTK). As the width \( l \) approaches infinity, for Gaussian randomly initialized parameters \( \theta_0 \), the kernel function \( \hat{\Phi}_{\theta_0}(\cdot, \cdot) \) becomes a deterministic function independent of \( \theta_0 \) (proved by Lemma 3 in Appendix B), denoted as
\[ \hat{\Phi}(\cdot, \cdot) := \lim_{l \to \infty} \hat{\Phi}_{\theta_0}(\cdot, \cdot) \] (5)
For convenience, we denote \( F_t(\cdot) \triangleq F_{\hat{\theta}_t}(\cdot), f_t(\cdot) \triangleq f_{\hat{\theta}_t}(\cdot) \) and \( \hat{\Phi}_t(\cdot, \cdot) \triangleq \hat{\Phi}_{\hat{\theta}_t}(\cdot, \cdot) \). 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 \( R > 0 \), \( \lambda > 0 \), such that: for width \( l \geq l^* \), running gradient descent with learning rates \( \eta < \frac{\lambda}{2l} \) and \( \lambda < \frac{\lambda_0}{2} \) over random initialization, the following upper bound on the training loss holds true with probability at least \( 1 - \delta \):
\[
\mathcal{L}(\theta_t) = \frac{1}{2} \| F_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y} \|^2_2 \\
\leq \left( 1 - \frac{\eta_0 \sigma_{\min}(\Phi)}{3} \right)^{2t} R.
\] (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 \( \theta_0 \). 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. Analytical Expression 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
\[
\frac{dF_{t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}')} {dt} = -\eta \hat{\Phi}_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y} \] (7)
where we denote \( \Phi \equiv \hat{\Phi}_{\theta_0}(\mathcal{X}, \mathcal{X}', \mathcal{Y}'), (\mathcal{X}, \mathcal{X}', \mathcal{Y}')) \) 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 \( T = (X, \mathcal{X}', Y, Y') \).

\[
F_t(\mathcal{X}, \mathcal{X}', Y') = F_0(\mathcal{X}, \mathcal{X}', Y') + \hat{\Phi}_{\theta_0}(\mathcal{X}, \mathcal{X}', \mathcal{Y}') T_{\Phi_{\theta_0}}^\eta (t) \] (8)
where \( \hat{\Phi}_{\theta_0}(\cdot, \cdot) \equiv \hat{\Phi}_{\theta_0}(\cdot, \cdot, \mathcal{Y}') \) and \( T_{\Phi_{\theta_0}}^\eta (t) = \hat{\Phi}_{\theta_0}^{-1} (1 - e^{-\eta \hat{\Phi}_{\theta_0}}) \) 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 \( f : \mathbb{R}^k \to \mathbb{R} \) with randomly initialized parameters \( \theta_0 \), its neural tangent kernel function is defined as \( \Theta_0(\cdot, \cdot) \equiv \Theta_{\theta_0}(\cdot, \cdot, \cdot) \equiv \nabla_{\theta_0} \nabla_{\theta_0} f_{\theta_0}(\cdot, \cdot)^T \), and it converges to a deterministic kernel as the width \( l \) approaches infinity [26, 37],
\[
\Theta(\cdot, \cdot) = \lim_{l \to \infty} \Theta_0
\] (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 \( \Phi \) defined in (5) can be equivalently viewed as a composite kernel function built upon NTK function \( \Theta(\cdot, \cdot) \). 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 \( \Phi \equiv \Phi((X, X'), (X', X'')) \in \mathbb{R}^{kN \times kN} \), 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

\[
[\Phi]_{ij} = \phi((X_i, X'_i), (X_j, X'_j)) \in \mathbb{R}^{kn \times kn},
\]

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

\[
\phi((\cdot, \cdot), (\cdot, \cdot)) = \Theta((\cdot, \cdot) + \Theta((\cdot, \cdot)^T \Theta((\cdot, \cdot) - \Theta((\cdot, \cdot)^T \Theta((\cdot, \cdot).
\]

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 \( \eta \) and \( \lambda \) are infinitesimal. As the network width \( l \) approaches infinity, with high probability random initialization of the neural net, the MAML output, \( (8) \), converges to a special kernel regression,

\[
F_\ell(X, X', Y') = G_{\phi}^\ell(X, X', Y') + \Phi((X, X'), (X, X')) \Theta^{\ell}(t) (Y' - G_{\phi}^\ell(X, X', Y'))
\]

where \( G \) is a function defined below.

\[
G_{\phi}^\ell(X, X', Y') = \Theta(X, X') \Theta^{\ell}(t) Y'.
\]

The \( \Phi((X, X'), (X, X')) \in \mathbb{R}^{kN \times kN} \) 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 \( j \in [N] \).

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

Remarks. The kernel \( \Phi \) is what \( \Phi_0 \) converges to as the network width approaches infinity. However, \( \Phi_0 \equiv \Phi_0((X, X', Y'), (X, X', Y')) \) depends on \( Y \) and \( Y' \), while \( \Phi \equiv \Phi((X, X'), (X, X')) \) does not, since the terms in \( \Phi \) depending on \( Y \) or \( Y' \) 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 \( \Phi \). Notably, \( \Phi \) can be seen as a composite kernel built upon the base kernel function \( \Theta \).

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-
Untrained Supernet $\mathcal{N}$

Final architecture

Architecture search space

Valid candidate? yes no

Metric 1: Condition Number of MetaNTK

Metric 2: # of linear regions

Prune $\mathcal{N}$ based on:

Target dataset

Architecture evaluation

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 $\theta_0$ 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 $X_i, Y_i \in \mathbb{R}^{n \times k}, X_i' \in \mathbb{R}^{m \times k}$ are $n$ query samples and $m$ support samples, respectively. Then, we compute the MetaNTK of the network following the analytical expression derived in Eqs. (10) and (11). Notice that Eq. (11) demonstrates the MetaNTK $\hat{\Phi}$ 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 $\Theta_{\theta_0}$. 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 $\Theta_{\theta_0}$ is computed purely from first-order gradients. Also, $\hat{\Phi}_{\theta_0}$ is positive definite, thus its eigenvalues are all positive, and we can define its condition number to be

$$C_{\mathcal{N}} := \frac{\sigma_{\text{max}}(\hat{\Phi}_{\theta_0})}{\sigma_{\text{min}}(\hat{\Phi}_{\theta_0})}$$

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

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

The condition number $C_{\mathcal{N}}$ indicates the trainability 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.

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 $\text{MaxPool3x3}$, $\text{AvgPool3x3}$, $\text{SkipConnect}$, $\text{Conv1x5-5x1}$, $\text{Conv3x3}$, $\text{SepConv3x3}$ and $\text{DilatedConv3x3}$.

**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 are reduction cells, where we decrease the spatial resolution and double the number of channels. We set the initial number of channels to 48. 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-

---

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

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

---

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

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

8We 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.
During the search phase, for computing hyper-parameters MetaNTK with the same RFS training-evaluate pipeline and present as methods such as MAML [19]. To compare our searched and GPU memory compared to higher-order optimization method. RFS greatly reduces the training time, work relatively well on tiered-ImageNet because mini- and tiered-ImageNet are both subsets of ImageNet, sharing lots of similarities. thus we stack the cells they searched on miniImageNet and train the structure on tieredImageNet using RFS. We believe MetaNAS cells best † 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].

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 retrained over at least 3 runs of MetaNAS(retrained) is repeated over at least 3 runs of the test accuracy. Similar to [60], we mostly use the evaluation 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.

### Table 1. Comparison on few-shot image classification benchmarks.

| Model          | Arch.  | #Cells | Train | #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++ [60]   | Cells  | 2      | FOMAML| 27k                 | 54.11±0.35 | 69.59±0.85 | -                   | -      | -      |
| MetaNAS [16]   | Cells  | 5      | Reptile | 1.1M       | 65.1±0.3  | 79.5±0.2  | -                   | -      | -      |
| MetaNAS(retrained) | Cells | 5      | RFS    | 2.01M       | 64.24±0.11 | 79.75±0.13 | 2.17M       | 70.16±0.09 | 84.99±0.22 |
| MetaNTK-NAS    | Cells  | 5      | RFS    | 1.77M       | 63.88±0.81 | 80.07±0.45 | 2.22M       | 71.12±0.49 | 85.71±0.22 |
| MetaNAS(retrained) | Cells | 8      | RFS    | 3.53M       | 63.88±0.23 | 79.88±0.14 | 3.70M       | 72.32±0.02 | 86.48±0.06 |
| MetaNTK-NAS    | Cells  | 8      | RFS    | 3.21M       | 64.26±0.14 | 80.35±0.12 | 4.78M       | 72.37±0.79 | 86.43±0.53 |

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.

### 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 $\ell_2$ regularized cross-entropy loss to finetune the linear layer, while we also use $\ell_2$ 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 code-base 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
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.

### Table 2. Ablation study on mini-ImageNet

| Model               | MetaNTK | NTK | Linear Region | 1-shot 5 Cells | 8 Cells | 5-shot 5 Cells | 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  |

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 miniImageNet. 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].
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] Trapti Bansal, Rishikesh Jha, and Andrew McCallum. Learning to few-shot learn across diverse natural language classification tasks, 2019. 2

[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 Stauffer, 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.

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

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

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

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

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

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

[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.

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

[35] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

[37] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019.

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

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

[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.

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

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

[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.

[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.

[45] 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.
[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, Salonl 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
Overview of the Appendix

The appendix is divided into the following sections,

• Sec. A: Describes the neural network setup and parameterization.

• Sec. B: Presents theoretical results and proof regarding the global convergence of gradient-based meta-learning.

• Sec. C: Derives the expression of MAML output.

• Sec. D: Derives the equivalence between MAML and kernel regressions.

• Sec. E: Presents more details of experiments in Sec. 6.

A. Neural Network Setup

In this paper, we consider a fully-connected feed-forward network with \( L \) hidden layers. Each hidden layer has width \( l_i \), for \( i = 1, \ldots, L \). The readout layer (i.e. output layer) has width \( l_{L+1} = k \). At each layer, for arbitrary input \( x \in \mathbb{R}^d \), we denote the pre-activation and post-activation functions by \( h_i(x) \), \( z_i(x) \in \mathbb{R}^{l_i} \). The relations between layers in this network are

\[
\begin{align*}
    h^{i+1} &= z^i W^{i+1} + b^{i+1} \\
    z^{i+1} &= \sigma(h^{i+1})
\end{align*}
\]

and

\[
\begin{align*}
    W_{\mu,\nu}^i &= \omega_{\mu,\nu}^i \sim \mathcal{N}(0, \sigma^2_{\omega} l_i^{\frac{1}{2}}) \\
    b_{\nu}^i &= \beta_{\nu}^i \sim \mathcal{N}(0, \sigma_b^2),
\end{align*}
\]

where \( W_{\mu,\nu}^i \in \mathbb{R}^{l_i \times l_{i+1}} \) and \( b_{\nu}^i \in \mathbb{R}^{l_{i+1}} \) are the weight and bias of the layer, \( \omega_{\mu,\nu}^i \) and \( \beta_{\nu}^i \) are trainable variables drawn i.i.d. from zero-mean Gaussian distributions at initialization (i.e., \( \sigma^2_{\omega} \) and \( \sigma^2_b \) are variances for weight and bias, and \( \sigma \) is a point-wise activation function.

B. Proof of Global Convergence for Gradient-Based Meta-Learning with Deep Neural Networks

In this section, we will prove the global convergence for gradient-based meta-learning with over-parameterized neural nets. To prove the global convergence theorem, we introduce several key lemmas first, i.e., Lemma 1, 2, 3. Specifically, the subsections of this section are formulated as follows.

• Sec. B.1: Present several helper lemmas with proof.

• Sec. B.2: Provides the proof of Lemma 1.

• Sec. B.3: Provides the proof of Lemma 2.

• Sec. B.4: Provides the proof of Lemma 3.

• Sec. B.5: Proves the global convergence theorem for MAML, i.e., Theorem 3 (restated version of Theorem 1).

Notice that in this section, we consider the standard parameterization scheme of neural networks shown in (16).

The global convergence theorem, Theorem 1, depends on several assumptions and lemmas. The assumptions are listed below. After that, we present the lemmas and the global convergence theorem, with proofs in Appendix B.1,B.3,B.4 and B.5. For Corollary 3.1, we append its proof to Appendix C.

Assumption 1 (Bounded Input Norm). \( \forall X \in \mathcal{X}, \) for any sample \( x \in X \), \( \|x\|_2 \leq 1 \). Similarly, \( \forall X' \in \mathcal{X}' \), for any sample \( x' \in X' \), \( \|x'\|_2 \leq 1 \). (This is equivalent to a input normalization operation, which is common in data preprocessing.)

Assumption 2 (Non-Degeneracy). The meta-training set \( (\mathcal{X}, \mathcal{Y}) \) and the meta-test set \( (\mathcal{X}', \mathcal{Y}') \) are both contained in some compact set. Also, \( \mathcal{X} \) and \( \mathcal{X}' \) are both non-degenerate, i.e. \( \forall X, \tilde{X} \in \mathcal{X}, X \neq \tilde{X}, \) and \( \forall X', \tilde{X}' \in \mathcal{X}', X' \neq \tilde{X}' \).

Assumption 3 (Same Width for Hidden Layers). All hidden layers share the same width, \( l \), i.e., \( l_1 = l_2 = \cdots = l_L = l \).

Assumption 4 (Full-Rank). The kernel \( \Phi \) defined in Lemma 3 is full-rank.

13
These assumptions are common, and one can find similar counterparts of them in the literature for supervised learning [4,37]. In particular, notice that Assumption 3 is just for simplicity purpose without loss generality. In fact, one can directly set \( l = \min_{i \in [l]} l_i \) as the minimum width across hidden layers, and all theoretical results in this paper still hold true [37].

As defined in the main text, \( \theta \) is used to represent the neural net parameters. For convenience, we define some short-hand notations:

\[
\begin{align*}
    f_k(\cdot) & = f_{\theta}(\cdot) \quad (17) \\
    F_k(\cdot) & = F_{\theta}(\cdot) \quad (18) \\
    f(\theta) & = f_{\theta}(\mathcal{X}) = ((f_{\theta}(X_i))_{i=1}^N) \quad (19) \\
    F(\theta) & = F_{\theta}(\mathcal{X}', \mathcal{Y}') = ((F_{\theta}(X_i, X_i', Y_i'))_{i=1}^N) \quad (20) \\
    g(\theta) & = F_{\theta}(\mathcal{X}', \mathcal{Y}') - \mathcal{Y'} \quad (21) \\
    J(\theta) & = \nabla_\theta F(\theta) = \nabla_\theta F_{\theta}(\mathcal{X}', \mathcal{Y}') \quad (22)
\end{align*}
\]

and

\[
\begin{align*}
    \mathcal{L}(\theta_t) & = \ell(F(\theta_t), \mathcal{Y}) = \frac{1}{2} \|g(\theta_t)\|_2^2 \quad (23) \\
    \hat{\Phi}_t & = \frac{1}{t} \nabla F_{\theta_t}(\mathcal{X}', \mathcal{Y}') \nabla F_{\theta_t}(\mathcal{X}', \mathcal{Y}') = \frac{1}{t} J(\theta) J(\theta)^\top \quad (24)
\end{align*}
\]

where we use the \( \ell_2 \) loss function \( \ell(\hat{y}, y) = \frac{1}{2} \|\hat{y} - y\|_2^2 \) in the definition of training loss \( \mathcal{L}(\theta_t) \) in (23), and the \( \hat{\Phi}_t \) in (24) is based on the definition of \( \Phi_{\theta_t}(\cdot, \cdot) \) in Sec. 4.1.

Below, Lemma 1 proves the Jacobian \( J \) is locally Lipschitz, Lemma 2 proves the training loss at initialization is bounded, and Lemma 3 proves \( \Phi_0 \) converges in probability to a deterministic kernel matrix with bounded positive eigenvalues. Finally, with these lemmas, we can prove the global convergence of MAML in Theorem 3.

**Lemma 1 (Local Lipschitzness of Jacobian).** For arbitrarily small \( \delta > 0 \), then there exists \( K > 0 \) and \( l^* > 0 \) such that: \( \forall \ C > 0 \) and \( l > l^* \), the following inequalities hold true with probability at least \( 1 - \delta \) over random initialization,

\[
\begin{align*}
    \forall \theta, \tilde{\theta} \in B(\theta_0, C l^{-\frac{1}{2}}), \quad & \left\{ \begin{array}{ll}
        \frac{1}{\sqrt{l}} \|J(\theta) - J(\tilde{\theta})\|_F & \leq K \|\theta - \tilde{\theta}\|_2 \\
        \frac{1}{\sqrt{l}} \|J(\theta)\|_F & \leq K
    \end{array} \right. \quad (25)
\end{align*}
\]

where \( B \) is a neighborhood defined as

\[
    B(\theta_0, R) := \{ \theta : \|\theta - \theta_0\|_2 < R \}.
\]

**Proof.** See Appendix B.1.

**Lemma 2 (Bounded Initial Loss).** For arbitrarily small \( \delta_0 > 0 \), there are constants \( R_0 > 0 \) and \( l^* > 0 \) such that as long as the width \( l > l^* \), with probability at least \( 1 - \delta_0 \) over random initialization,

\[
    \|g(\theta_0)\|_2 = \|F_{\theta_0}(\mathcal{X}', \mathcal{Y}') - \mathcal{Y}'\|_2 \leq R_0,
\]

which is also equivalent to

\[
    \mathcal{L}(\theta_0) = \frac{1}{2} \|g(\theta_0)\|_2^2 \leq \frac{1}{2} R_0^2.
\]

**Proof.** See Appendix B.3.

**Lemma 3 (Kernel Convergence).** Suppose the learning rates \( \eta \) and \( \lambda \) sufficiently small. As the network width \( l \) approaches infinity, \( \Phi_0 = J(\theta_0) J(\theta_0)^\top \) converges in probability to a deterministic kernel matrix \( \Phi \) (i.e., \( \Phi = \lim_{l \to \infty} \Phi_0 \)), which is independent of \( \theta_0 \) and can be analytically calculated. Furthermore, the eigenvalues of \( \Phi \) is bounded as, \( 0 < \sigma_{\min}(\Phi) \leq \sigma_{\max}(\Phi) < \infty \).

\[\text{[12]}\text{There is a typo in the definition of } \Phi_{\theta}(\cdot, \cdot) \text{ in Sec. 4.1: a missing factor } \frac{1}{t}. \text{ The correct definition should be } \Phi_{\theta}(\cdot, \cdot) = \frac{1}{t} \nabla_\theta F_{\theta}(\cdot) \nabla_\theta F_{\theta}(\cdot)^\top. \text{ Similarly, the definition of } \Phi \text{ in Theorem 1 also misses this factor: the correct version is } \Phi = \frac{1}{t} \lim_{l \to \infty} J(\theta_0) J(\theta_0)^\top.\]
Proof. See Appendix B.4.

Note the update rule of gradient descent on \( \theta_t \) with learning rate \( \eta \) can be expressed as

\[
\theta_{t+1} = \theta_t - \eta J(\theta_t) \nabla g(\theta_t).
\] (28)

The following theorem proves the global convergence of MAML under the update rule of gradient descent.

**Theorem 3 (Global Convergence) (Theorem 1 restated).** Denote \( \sigma_{\min} = \sigma_{\min}(\Phi) \) and \( \sigma_{\max} = \sigma_{\max}(\Phi) \). For any \( \delta_0 > 0 \) and \( \eta_0 < 2 \sigma_{\max} / \sigma_{\min} \); there exist \( R_0 > 0 \), \( \Lambda \in \mathbb{N} \), \( K > 1 \), and \( \lambda_0 > 0 \), such that: for width \( l \geq \Lambda \), running gradient descent with learning rates \( \eta = \frac{\eta_0}{l} \) and \( \lambda = \frac{\lambda_0}{l} \) over random initialization, the following inequalities hold true with probability at least \((1 - \delta_0)\):

\[
\sum_{j=1}^{l} \|\theta_j - \theta_{j-1}\|_2 \leq \frac{3K R_0}{\sigma_{\min}} l^{-\frac{1}{2}}
\] (29)

\[
\sup_t \|\Phi_0 - \Phi_t\|_F \leq \frac{6K^3 R_0}{\sigma_{\min}} l^{-\frac{1}{2}}
\] (30)

and

\[
g(\theta_t) = \|F(\theta_t) - \mathcal{Y}\|_2 \leq \left(1 - \frac{\eta_0 \sigma_{\min}}{3}\right)^t R_0,
\] (31)

which leads to

\[
\mathcal{L}(\theta_t) = \frac{1}{2} \|F(\theta_t) - \mathcal{Y}\|_2^2 \leq \left(1 - \frac{\eta_0 \sigma_{\min}}{3}\right)^{2t} \frac{R_0^2}{2}
\] (32)

indicating the training loss converges to zero at a linear rate.

Proof. See Appendix B.5.

In the results of Theorem 3 above, (29) considers the optimization trajectory of network parameters, and show the parameters move locally during training. (30) indicates the kernel matrix \( \Phi_t \) changes slowly. Finally, (32) demonstrates that the training loss of MAML decays exponentially to zero as the training time evolves, indicating convergence to global optima at a linear rate.

**B.1. Helper Lemmas**

**Lemma 4.** As the width \( l \to \infty \), for any vector \( a \in \mathbb{R}^{m \times 1} \) that \( \|a\|_F \leq C \) with some constant \( C > 0 \), we have

\[
\|\nabla_{\theta} \hat{\Theta}_\theta(x, X') : a\|_F \to 0
\] (33)

where \( \theta \) is randomly initialized parameters.

Proof. Notice that

\[
\hat{\Theta}_\theta(x, X') = \frac{1}{l} \sum_{e \in \mathbb{R}^d} \sum_{\theta \in \mathbb{R}^{m \times d}} \nabla_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \in \mathbb{R}^{1 \times m}
\] (34)

with gradient as

\[
\nabla_{\theta} \hat{\Theta}_\theta(x, X') = \frac{1}{l} \sum_{e \in \mathbb{R}^d \times D} \nabla_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top + \frac{1}{l} \sum_{e \in \mathbb{R}^{D \times m}} \nabla_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \in \mathbb{R}^{1 \times m \times D}
\] (35)

where we apply a dot product in the first two dimensions of 3-tensors and matrices to obtain matrices.

Then, it is obvious that our goal is to bound the Frobenius Norm of

\[
\nabla_{\theta} \hat{\Theta}_\theta(x, X') : a = \left(\frac{1}{l} \sum_{e \in \mathbb{R}^d \times D} \nabla_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \right) : a + \left(\frac{1}{l} \sum_{e \in \mathbb{R}^{D \times m}} \nabla_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \right) : a
\] (36)

Below, we prove that as the width \( l \to \infty \), the first and second terms of (36) both have vanishing Frobenius norms, which finally leads to the proof of (33).
• First Term of (36). Obviously, reshaping $\nabla^2_{\theta} f_\theta(x) \in \mathbb{R}^{1 \times D \times D}$ as a $\mathbb{R}^{D \times D}$ matrix does not change the Frobenius norm $\frac{1}{l} \nabla^2_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \|_F$ (in other words, $\frac{1}{l} \nabla^2_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \|_F = \frac{1}{l} \nabla^2_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \|_F$).

By combining the following three facts,

1. $\frac{1}{\sqrt{l}} \nabla^2_{\theta} f_\theta(x) \|_{op} \to 0$ indicated by [27],

2. the matrix algebraic fact $\|HB\|_F \leq \|H\|_{op}\|B\|_F$.

3. the bound $\frac{1}{\sqrt{l}} \nabla_{\theta} f_\theta(\cdot) \|_F < \text{constant}$ from [37],

one can easily show that the first term of (35) has vanishing Frobenius norm, i.e.,

$$\frac{1}{l} \nabla^2_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \|_F \to 0 \quad (37)$$

Then, obviously,

$$\| \left( \frac{1}{l} \nabla^2_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \right) \cdot a \|_F \leq \frac{1}{l} \nabla^2_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \|_F \|a\|_F \to 0 \quad (38)$$

• Second Term of (36). From [27], we know that

$$\| \frac{1}{\sqrt{l}} \nabla^2_{\theta} f_\theta(X')^\top \cdot a \|_{op} \to 0 \quad (39)$$

Then, similar to the derivation of (37), we have

$$\| \left( \frac{1}{l} \nabla_{\theta} f_\theta(x) \cdot \nabla^2_{\theta} f_\theta(X')^\top \right) \cdot a \|_F \leq \frac{1}{\sqrt{l}} \nabla_{\theta} f_\theta(x) \|_F \| \nabla^2_{\theta} f_\theta(X')^\top \cdot a \|_{op} \to 0 \quad (40)$$

• Finally, combining (38) and (40), we obtain (33) by

$$\| \nabla_{\theta} \hat{\Theta}_\theta(x, X') \cdot a \|_F \leq \frac{1}{l} \nabla^2_{\theta} f_\theta(x) \cdot \nabla_{\theta} f_\theta(X')^\top \cdot a \|_F \quad (41)$$

Lemma 5. Given any task $T = (X, Y, X', Y')$ and randomly initialized parameters $\theta$, as the width $l \to \infty$, for any $x \in X$, where $x \in \mathbb{R}^d$ and $X \in \mathbb{R}^{n \times d}$, we have

$$\| \nabla_{\theta} \left( \hat{\Theta}_\theta(x, X') \hat{\Theta}_\theta^{-1}(I - e^{-\lambda \hat{\Theta}_\theta^\top}) \right) (f_\theta(X') - Y') \|_F \to 0 \quad (42)$$

and furthermore,

$$\| \nabla_{\theta} \left( \hat{\Theta}_\theta(x, X') \hat{\Theta}_\theta^{-1}(I - e^{-\lambda \hat{\Theta}_\theta^\top}) \right) (f_\theta(X') - Y') \|_F \to 0 \quad (43)$$
Proof of Lemma 5.

Overview. In this proof, we consider the expression

\[ \nabla_{\theta} \left( \hat{\Theta}_{\theta}(x, X') \hat{\Theta}_{\theta}^{-1}(I - e^{-\hat{\Theta}_{\theta} \tau}) \right) (f_{\theta}(X') - Y') \]

(44)

and we prove the terms of (45), (46) and (47) all have vanishing Frobenius norm. Thus, (44) also has vanishing Frobenius norm in the infinite width limit, which is exactly the statement of (42). This indicates that (43) also has a vanishing Frobenius norm, since \( \Theta_{\theta}(X, X') \) can be seen as a stack of \( n \) copies of \( \Theta_{\theta}(x, X') \), where \( n \) is a finite constant.

Step I. Each factor of (44) has bounded Frobenius norm.

- \( \| \hat{\Theta}_{\theta}(x, X') \|_{F} \). It has been shown that \( \| \frac{1}{\sqrt{d}} \nabla_{\theta} f(\cdot) \|_{F} \leq \text{constant} \) in \([37]\), thus we have \( \| \hat{\Theta}_{\theta}(x, X') \|_{F} = \| \frac{1}{\sqrt{d}} \nabla_{\theta} f(x) \|_{F} \leq \| \frac{1}{\sqrt{d}} \nabla_{\theta} f(X') \|_{F} \leq \text{constant} \).

- \( \| \hat{\Theta}_{\theta}^{-1} \|_{F} \). It has been shown that \( \hat{\Theta}_{\theta} \) is positive definite with positive least eigenvalue \([4, 26]\), thus \( \| \hat{\Theta}_{\theta}^{-1} \|_{F} \leq \text{constant} \).

- \( \| I - e^{-\hat{\Theta}_{\theta} \tau} \|_{F} \). \([10]\) shows that largest eigenvalues of \( \hat{\Theta}_{\theta} \) are of \( O(L) \), and we know \( \hat{\Theta}_{\theta} \) is positive definite \([4, 26]\), thus it is obvious the eigenvalues of \( I - e^{-\hat{\Theta}_{\theta} \tau} \) fall in the set \( \{ z \mid 0 < z < 1 \} \). Therefore, certainly we have \( \| I - e^{-\hat{\Theta}_{\theta} \tau} \|_{F} \leq \text{constant} \).

- \( \| f_{\theta}(X') - Y' \|_{F} \). \([37]\) shows that \( \| f_{\theta}(X') - Y' \|_{2} \leq \text{constant} \), which indicates that \( \| f_{\theta}(X') - Y' \|_{F} \leq \text{constant} \).

In conclusion, we have shown

\[ \| \hat{\Theta}_{\theta}(x, X') \|_{F}, \| \hat{\Theta}_{\theta}^{-1} \|_{F}, \| I - e^{-\hat{\Theta}_{\theta} \tau} \|_{F}, \| f_{\theta}(X') - Y' \|_{F} \leq \text{constant} \]  

(48)

Step II. Bound (45).

Without loss of generality, let us consider the neural net output dimension \( k = 1 \) in this proof, i.e., \( f_{\theta} : \mathbb{R}^{d} \to \mathbb{R} \). (Note: with \( k > 1 \), the only difference is that \( \nabla_{\theta} f(X') \in \mathbb{R}^{mk \times D} \), which has no impact on the proof). Then, we have

\[ \hat{\Theta}_{\theta}(x, X') = \frac{1}{\lambda} \nabla_{\theta} f_{\theta}(x) \cdot \nabla_{\theta} f_{\theta}(X')^{\top} \in \mathbb{R}^{1 \times m} \]  

(49)

with gradient as

\[ \nabla_{\theta} \hat{\Theta}_{\theta}(x, X') = \frac{1}{\lambda} \nabla_{\theta}^{2} f_{\theta}(x) \cdot \nabla_{\theta} f_{\theta}(X')^{\top} + \frac{1}{\lambda} \nabla_{\theta} f_{\theta}(x) \cdot \nabla_{\theta}^{2} f_{\theta}(X')^{\top} \in \mathbb{R}^{1 \times m \times D} \]  

(50)

where we apply a dot product in the first two dimensions of 3-tensors and matrices to obtain matrices.

Based on (48), we know that

\[ \| \hat{\Theta}_{\theta}^{-1}(I - e^{-\hat{\Theta}_{\theta} \tau})(f_{\theta}(X') - Y') \|_{F} \leq \| \hat{\Theta}_{\theta}^{-1} \|_{F} \| I - e^{-\hat{\Theta}_{\theta} \tau} \|_{F} \| f_{\theta}(X') - Y' \|_{F} \leq \text{constant} . \]

Then, applying (33), we have

\[ \| \nabla_{\theta} \left( \hat{\Theta}_{\theta}(x, X') \right) : \hat{\Theta}_{\theta}^{-1}(I - e^{-\hat{\Theta}_{\theta} \tau})(f_{\theta}(X') - Y') \|_{F} \to 0 \]  

(51)
Step III. Bound (46) and (47)

• Bound (46): \( \hat{\Theta}_{\theta}(x, X') \left( \nabla_{\theta} \hat{\Theta}_{\theta}^{-1} \right) (I - e^{-\lambda \hat{\Theta}_{\theta}^T})(f_{\theta}(X') - Y'). \)

Clearly, \( \nabla_{\theta} \hat{\Theta}_{\theta}^{-1} = - \hat{\Theta}_{\theta}^{-1} \cdot (\nabla_{\theta} \hat{\Theta}_{\theta}) \cdot \hat{\Theta}_{\theta}^{-1}, \) where we apply a dot product in the first two dimensions of the \( 3 \)-tensor and matrices.

Note that \( \nabla_{\theta} \hat{\Theta}_{\theta} = \sqrt{T \nabla_{\theta}^2 f_{\theta}(X') \cdot \nabla_{\theta} f_{\theta}(X') \cdot \nabla_{\theta}^2 f_{\theta}(X')^T}. \) Obviously, by (39) and (48), we can easily prove that

\[
\| \nabla_{\theta} \hat{\Theta}_{\theta}(x, X') \left( \nabla_{\theta} \hat{\Theta}_{\theta}^{-1} \right) (I - e^{-\lambda \hat{\Theta}_{\theta}^T})(f_{\theta}(X') - Y') \|_F \rightarrow 0
\]  

(52)

• Bound (47): \( \hat{\Theta}_{\theta}(x, X') \hat{\Theta}_{\theta}^{-1} \left( \nabla_{\theta}(I - e^{-\lambda \hat{\Theta}_{\theta}^T}) \right) (f_{\theta}(X') - Y') \)

Since \( \nabla_{\theta}(I - e^{-\lambda \hat{\Theta}_{\theta}^T}) = \lambda \cdot (\nabla_{\theta} \hat{\Theta}_{\theta}) \cdot (\nabla_{\theta} \hat{\Theta}_{\theta}) \), we can easily obtain the following result by (39) and (48),

\[
\| \hat{\Theta}_{\theta}(x, X') \hat{\Theta}_{\theta}^{-1} \left( \nabla_{\theta}(I - e^{-\lambda \hat{\Theta}_{\theta}^T}) \right) (f_{\theta}(X') - Y') \|_F \rightarrow 0
\]

(53)

Step IV. Final result: prove (44) and (43).

Combining (51), (52) and (53), we can prove (44)

\[
\| \nabla_{\theta} \left( \hat{\Theta}_{\theta}(x, X') \hat{\Theta}_{\theta}^{-1}(I - e^{-\lambda \hat{\Theta}_{\theta}^T}) \right) (f_{\theta}(X') - Y') \|_F \leq \| \nabla_{\theta} \left( \hat{\Theta}_{\theta}(x, X') \right) \hat{\Theta}_{\theta}^{-1}(I - e^{-\lambda \hat{\Theta}_{\theta}^T})(f_{\theta}(X') - Y') \|_F \\
+ \| \hat{\Theta}_{\theta}(x, X') \left( \nabla_{\theta} \hat{\Theta}_{\theta}^{-1} \right) (I - e^{-\lambda \hat{\Theta}_{\theta}^T})(f_{\theta}(X') - Y') \|_F \\
+ \| \hat{\Theta}_{\theta}(x, X') \hat{\Theta}_{\theta}^{-1} \left( \nabla_{\theta}(I - e^{-\lambda \hat{\Theta}_{\theta}^T}) \right) (f_{\theta}(X') - Y') \|_F \\
\rightarrow 0
\]

(55)

Then, since \( \hat{\Theta}_{\theta}(X, X') \) can be seen as a stack of \( n \) copies of \( \hat{\Theta}_{\theta}(x, X') \), where \( n \) is a finite constant, we can easily prove (43) by

\[
\| \nabla_{\theta} \left( \hat{\Theta}_{\theta}(X, X') \hat{\Theta}_{\theta}^{-1}(I - e^{-\lambda \hat{\Theta}_{\theta}^T}) \right) (f_{\theta}(X') - Y') \|_F \\
\leq \sum_{i \in [n]} \| \nabla_{\theta} \left( \hat{\Theta}_{\theta}(x_i, X') \hat{\Theta}_{\theta}^{-1}(I - e^{-\lambda \hat{\Theta}_{\theta}^T}) \right) (f_{\theta}(X') - Y') \|_F \\
\rightarrow 0
\]

(57)

where we denote \( X = (x_i)_{i=1}^n \).

B.2. Proof of Lemma 1

Proof of Lemma 1. Consider an arbitrary task \( \mathcal{T} = (X, Y, X', Y') \). Given sufficiently large width \( l \), for any parameters in the neighborhood of the initialization, i.e., \( \theta \in B(\theta_0, Cl^{-1/2}) \), based on [37], we know the meta-output can be decomposed into a terms of \( f_{\theta} \),

\[
F_{\theta}(X, X', Y') = f_{\theta}(X) - \hat{\Theta}_{\theta}(X, X') \hat{\Theta}_{\theta}^{-1}(I - e^{-\lambda \hat{\Theta}_{\theta}^T})(f_{\theta}(X') - Y'),
\]

(58)

where \( \hat{\Theta}_{\theta}(X, X') = \frac{1}{l} \nabla_{\theta} f_{\theta}(X) \nabla_{\theta} f_{\theta}(X')^T \), and \( \hat{\Theta}_{\theta} \equiv \hat{\Theta}_{\theta}(X', X') \) for convenience.
Therefore, it is easy to get that for arbitrary $K$

By Lemma 5, we know the last term of (59) has a vanishing Frobenius norm as the width increases to infinity. Thus, for any $\varepsilon > 0$ and $0 < \delta < 1$, there exists $l^* > 0$ s.t. for width $l > l^*$, with probability at least $1 - \delta$, the last term of (59) is of $O(\varepsilon)$, i.e.,

$$
\nabla_\theta F_\theta(X, X', Y') = \nabla_\theta f_\theta(X) - \hat{\Theta}_\theta(X, X') \hat{\Theta}_\theta^{-1}(I - e^{-\lambda H_\theta \tau}) \nabla_\theta f_\theta(X')
$$

For any $\varepsilon > 0$, we can easily see that the eigenvalues of $H_\theta$ are all in the range $[0, 1]$ for arbitrary $\tau > 0$. Therefore, it is easy to get that for arbitrary $\tau > 0$,

$$
||\nabla_\theta F((X, X', Y'), \theta)||_F = ||\nabla_\theta f_\theta(X) e^{-\lambda H_\theta \tau}||_F \leq ||\nabla_\theta f_\theta(X)||_F
$$

By Lemma 1 of [37], we know that there exists a $K_0 > 0$ such that for any $X$ and $\theta$,

$$
||\frac{1}{\sqrt{l}} \nabla_\theta f_\theta(X)||_F \leq K_0.
$$

Combining (63) and (64), we have

$$
||\frac{1}{\sqrt{l}} \nabla_\theta F((X, X', Y'), \theta)||_F \leq ||\frac{1}{\sqrt{l}} \nabla_\theta f_\theta(X)||_F \leq K_0,
$$
which is equivalent to
\[ \frac{1}{\sqrt{l}} \| J(\theta) \|_F \leq K_0 \] (66)

Now, let us study the other term of interest, \( \| J(\theta) - J(\bar{\theta}) \|_F = \frac{1}{\sqrt{l}} \nabla_{\theta} F((X, X', \gamma), \theta) - \frac{1}{\sqrt{l}} \nabla_{\theta} F((X, X', \gamma), \bar{\theta}) \|_F \), where \( \theta, \bar{\theta} \in B(\theta_0, C_l^{-1/2}) \).

To bound \( \| J(\theta) - J(\bar{\theta}) \|_F \), let us consider
\[
\| \nabla_{\theta} F((X, X', \gamma), \theta) - \nabla_{\theta} F((X, X', \gamma), \bar{\theta}) \|_{op}
\]
(67)
\[
= \| \nabla_{\theta} f_\theta(X) e^{-\lambda H_{\theta} \gamma} - \nabla_{\theta} f_\bar{\theta}(X) e^{-\lambda H_{\bar{\theta}} \gamma} \|_{op}
\]
(68)
\[
= \frac{1}{2} \| (\nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X)) (e^{-\lambda H_{\theta} \gamma} + e^{-\lambda H_{\bar{\theta}} \gamma})
+ (\nabla_{\theta} f_\theta(X) + \nabla_{\theta} f_\bar{\theta}(X)) (e^{-\lambda H_{\theta} \gamma} - e^{-\lambda H_{\bar{\theta}} \gamma}) \|_{op}
\]
(69)
\[
\leq \frac{1}{2} \| \nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X) \|_{op} \| e^{-\lambda H_{\theta} \gamma} + e^{-\lambda H_{\bar{\theta}} \gamma} \|_{op}
+ \frac{1}{2} \| \nabla_{\theta} f_\theta(X) + \nabla_{\theta} f_\bar{\theta}(X) \|_{op} \| e^{-\lambda H_{\theta} \gamma} - e^{-\lambda H_{\bar{\theta}} \gamma} \|_{op}
\]
(70)
\[
\leq \frac{1}{2} \| \nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X) \|_{op} \| e^{-\lambda H_{\theta} \gamma} \|_{op} + \| e^{-\lambda H_{\theta} \gamma} \|_{op}
+ \frac{1}{2} \| \nabla_{\theta} f_\theta(X) \|_{op} + \| \nabla_{\theta} f_\bar{\theta}(X) \|_{op} \| e^{-\lambda H_{\theta} \gamma} - e^{-\lambda H_{\bar{\theta}} \gamma} \|_{op}
\]
(71)

It is obvious that \( \| e^{-\lambda H_{\theta} \gamma} \|_{op}, \| e^{-\lambda H_{\theta} \gamma} \|_{op} \leq 1 \). Also, by the relation between the operator norm and the Frobenius norm, we have
\[
\| \nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X) \|_{op} \leq \| \nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X) \|_F
\]
(72)

Besides, Lemma 1 of [37] indicates that there exists a \( K_1 > 0 \) such that for any \( X \) and \( \theta, \bar{\theta} \in B(\theta_0, C_l^{-1/2}) \),
\[
\frac{1}{\sqrt{l}} \| \nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X) \|_F \leq K_1 \| \theta - \bar{\theta} \|_2
\]
(73)

Therefore, (72) gives
\[
\| \nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X) \|_{op} \leq K_1 \sqrt{l} \| \theta - \bar{\theta} \|_2
\]
(74)

and then (70) is bounded as
\[
\frac{1}{2} \| \nabla_{\theta} f_\theta(X) - \nabla_{\theta} f_\bar{\theta}(X) \|_{op} (\| e^{-\lambda H_{\theta} \gamma} \|_{op} + \| e^{-\lambda H_{\theta} \gamma} \|_{op}) \leq K_1 \sqrt{l} \| \theta - \bar{\theta} \|_2.
\]
(75)

As for (71), notice that \( \| \cdot \|_{op} \leq \| \cdot \|_F \) and (64) give us
\[
\frac{1}{2} (\| \nabla_{\theta} f_\theta(X) \|_{op} + \| \nabla_{\theta} f_\bar{\theta}(X) \|_{op}) \leq \sqrt{l} K_0.
\]
(76)

Then, to bound \( \| e^{-\lambda H_{\theta} \gamma} - e^{-\lambda H_{\theta} \gamma} \|_{op} \) in (71), let us bound the following first
\[
\| H_{\theta} - H_{\bar{\theta}} \|_F = \| \frac{1}{l} \nabla_{\theta} f_\theta(X')^T \nabla_{\theta} f_\theta(X') - \frac{1}{l} \nabla_{\theta} f_\bar{\theta}(X')^T \nabla_{\theta} f_\bar{\theta}(X') \|_F
\]
(77)
Then, with the results above and a perturbation bound\(^\text{13}\) on matrix exponentials from [29], we have
\[
\|e^{-\lambda H_\theta \tau} - e^{-\lambda H_\bar{\theta} \tau}\|_{\text{op}} \leq \|H_\theta - H_\bar{\theta}\|_{\text{op}} \cdot \left(\lambda \tau e^{-\lambda \tau (\|H_\theta\|_{\text{op}} - \|H_\theta - H_\bar{\theta}\|_{\text{op}})}\right)
\]
\[
\leq \frac{\|H_\theta - H_\bar{\theta}\|_{\text{op}}^2}{\|H_\theta\|_{\text{op}} - \|H_\theta - H_\bar{\theta}\|_{\text{op}}}
\]
\[
\leq \mathcal{O}(\|H_\theta - H_\bar{\theta}\|_{\text{op}})
\]
\[
\leq 2K_0K_1K_2\|\theta - \bar{\theta}\|_2
\]
(78)
where we used the facts \(\|H_\theta\|_{\text{op}} = \|\hat{\Theta}_\theta\|_{\text{op}} \geq \mathcal{O}(1)\) [10, 71] and \(\|H_\theta - H_\bar{\theta}\|_{\text{op}} \leq \mathcal{O}(\|\theta - \bar{\theta}\|_2) \leq \mathcal{O}(\frac{1}{\sqrt{n}})\).
Hence, by (76) and (78), we can bound (71) as
\[
\frac{1}{2} (\|\nabla_\theta f_\theta(X)\|_{\text{op}} + \|\nabla_\bar{\theta} f_\bar{\theta}(X)\|_{\text{op}}) \|e^{-\lambda H_\theta \tau} - e^{-\lambda H_\bar{\theta} \tau}\|_{\text{op}} \leq 2\sqrt{K_0^2K_1K_2}\|\theta - \bar{\theta}\|_2
\]
(79)
Finally, with (75) and (79), we can bound (67) as
\[
\|\nabla_\theta F((X, X', Y'), \theta) - \nabla_\theta F((X, X', Y'), \bar{\theta})\|_{\text{op}} \leq (K_1 + 2K_0^2K_1K_2)\sqrt{n}\|\theta - \bar{\theta}\|_2
\]
Finally, combining these bounds on (70) and (71), we know that
\[
\|J(\theta) - J(\bar{\theta})\|_F = \left\|\frac{1}{\sqrt{n}} \nabla_\theta F((X, X', Y'), \theta) - \frac{1}{\sqrt{1}} \nabla_\theta F((X, X', Y'), \bar{\theta})\right\|_F
\]
\[
\leq \sqrt{\frac{k}{n^2}} \|\nabla_\theta F((X, X', Y'), \theta) - \nabla_\theta F((X, X', Y'), \bar{\theta})\|_{\text{op}}
\]
\[
\leq \sqrt{\frac{k}{n}}(K_1 + 2K_0^2K_1K_2)\|\theta - \bar{\theta}\|_2
\]
(80)
Define \(K_3 = \sqrt{k/n}(K_1 + 2K_0^2K_1K_2)\), we have
\[
\|J(\theta) - J(\bar{\theta})\|_F \leq K_3\|\theta - \bar{\theta}\|_2
\]
(81)
Taking \(K = \max\{K_0, K_3\}\) completes the proof. □

B.3. Proof of Lemma 2

Proof of Lemma 2. It is known that \(f_\theta(\cdot)\) converges in distribution to a mean zero Gaussian with the covariance \(\mathcal{K}\) determined by the parameter initialization [37]. As a result, for arbitrary \(\delta_1 \in (0, 1)\) there exist constants \(l_1 > 0\) and \(R_1 > 0\), such that: \(\forall l \geq l_1\), over random initialization, the following inequality holds true with probability at least \((1 - \delta_1)\),
\[
\|f_{\theta_0}(X) - Y\|_2, \|f_{\theta_0}(X') - Y'\|_2 \leq R_1
\]
(82)
We know that \(\forall T = (X, Y, X', Y') \in D\),
\[
F_{\theta_0}(X, X', Y') = f_{\theta_0}(X)
\]
where \(\theta_0\) is the parameters after \(\tau\)-step update on \(\theta_0\) over the meta-test task \((X', Y')\):
\[
\theta_\tau = \theta', \theta_0 = \theta,
\]
\[
\theta_{i+1} = \theta_i - \lambda \nabla_{\theta_i} \ell(f_{\theta_i}(X'), Y') \quad \forall i = 0, \ldots, \tau - 1,
\]
(83)
Suppose the learning rate \(\lambda\) is sufficiently small, then similar to (58), we have
\[
F_{\theta_0}(X, X', Y') = f_{\theta_0}(X) + \hat{\Theta}_0(X, X') \hat{\Theta}_0^{-1}(I - e^{-\lambda \hat{\theta}_0 \tau})(f_{\theta_0}(X') - Y').
\]
(84)
where \(\hat{\Theta}_0(\cdot, \cdot) = \nabla_{\theta_0} f_{\theta_0}(\cdot) \nabla_{\theta_0} f_{\theta_0}(\cdot)^\top\) and we use a shorthand \(\hat{\Theta}_0(\bar{T}) \equiv \hat{\Theta}_0(X', X')\).

\(^{13}\)This bound is also derived in [61].
[26] proves that for sufficiently large width, \( \hat{\Theta}_0 \) is positive definite and converges to \( \Theta \), the Neural Tangent Kernel, a full-rank kernel matrix with bounded positive eigenvalues. Let \( \sigma_{\min}(\Theta) \) and \( \sigma_{\max}(\Theta) \) denote the least and largest eigenvalue of \( \Theta \), respectively. Then, it is obvious that for a sufficiently over-parameterized neural network, the operator norm of \( \hat{\Theta}(X, X')^{-1}(I - e^{-\lambda \hat{\Theta}}) \) can be bounded based on \( \sigma_{\min}(\Theta) \) and \( \sigma_{\max}(\Theta) \). Besides, [4, 37] demonstrate that the neural net output at initialization, \( f_{\theta_0}(\cdot) \), is a zero-mean Gaussian with small-scale covariance. Combining these results and (82), we know there exists \( R(\bar{R}_1, N, \sigma_{\min}(\Theta), \sigma_{\max}(\Theta)) \) such that

\[
\|F_{\theta_0}(X, X', Y') - Y\|_2 \leq R(\bar{R}_1, N, \sigma_{\min}(\Theta), \sigma_{\max}(\Theta)) \quad (85)
\]

By taking an supremum over \( R(\bar{R}_1, N, \sigma_{\min}, \sigma_{\max}) \) for each training task in \( \{T_i = (X_i, Y_i, X'_i, Y'_i)\}_{i\in[N]} \), we can get \( R_2 \) such that \( \forall i \in [N] \)

\[
\|F_{\theta_0}(X_i, X'_i, Y'_i) - Y_i\|_2 \leq R_2 \quad (86)
\]

and for \( R_0 = \sqrt{NR_2} \), define \( \delta_0 \) as some appropriate scaling of \( \delta_1 \), then the following holds true with probability \( (1 - \delta) \) over random initialization,

\[
g(\theta_0)\|_2 = \sqrt{\sum_{X, Y, X', Y' \in D} \|F((X, X', Y'), \theta_0) - y\|_2^2} \leq R_0 \quad (87)
\]

\[\square\]

B.4. Proof of Lemma 3

Proof of Lemma 3. The learning rate for meta-adaption, \( \lambda \), is sufficiently small, so the inner-loop optimization becomes continuous-time gradient descent. Based on [37], for any task \( T = (X, Y, X', Y') \),

\[
F_0(X, X', Y') = f_0(X) + \hat{\Theta}_0(X, X')\tilde{T}_\theta^\lambda(X', \tau) (Y' - f_0(X')) \quad (88)
\]

where \( \hat{\Theta}_0(\cdot, \cdot) = \frac{1}{T} \nabla \theta_0 f_0(\cdot) \nabla \theta_0 f_0(\cdot)^T \), and \( \tilde{T}_\theta^\lambda(\cdot, \cdot) := \hat{\Theta}_0(\cdot, \cdot)^{-1}(I - e^{-\lambda \hat{\Theta}_0(\cdot, \cdot)\tau}) \).

Then, we consider \( \nabla \theta_0 F_0(X, X', Y') \), the gradient of \( F_0(X, X', Y') \) in (88). By Lemma 5, we know that for sufficiently wide networks, the gradient of \( F_0(X, X', Y') \) becomes

\[
\nabla \theta_0 F_0(X, X', Y') = \nabla \theta_0 f_0(X) - \hat{\Theta}_0(X, X') T_{\theta_0}^\lambda(X', \tau) \nabla \theta_0 f_0(X') \quad (89)
\]

Since \( \hat{\Phi}_0 := \hat{\Phi}_0((X, X', Y'), (X', X', Y')) = \frac{1}{T} \nabla \theta_0 f_0(X, X', Y') \nabla \theta_0 f_0(X, X', Y')^T \) and \( F_0(X, X', Y') = (F_0(X_i, X'_i, Y'_i))_{i=1}^N \in \mathbb{R}^{knN} \), we know \( \hat{\Phi}_0 \) is a block matrix with \( N \times N \) blocks of size \( kn \times kn \). For \( i, j \in [N] \), the \((i, j)\)-th block can be denoted as \( [\hat{\Phi}_0]_{ij} \) such that

\[
[\hat{\Phi}_0]_{ij} = \frac{1}{T} \nabla \theta_0 f_0(X_i, X'_i, Y'_i) \nabla \theta_0 f_0(X_j, X'_j, Y'_j)^T
\]

\[
\begin{align*}
&= \frac{1}{T} \nabla \theta_0 f_0(X_i) \nabla \theta_0 f_0(X'_i)^T + \frac{1}{T} \hat{\Theta}_0(X_i, X'_i) \tilde{T}_\theta^\lambda(X'_i, \tau) \nabla \theta_0 f_0(X'_i) \nabla \theta_0 f_0(X'_j)^T \tilde{T}_\theta^\lambda(X'_j, \tau)^T \hat{\Theta}_0(X'_j, X_j) \\
&- \frac{1}{T} \nabla \theta_0 f_0(X_i) \nabla \theta_0 f_0(X'_j)^T \tilde{T}_\theta^\lambda(X'_j, \tau)^T \hat{\Theta}_0(X'_j, X_j) \\
&- \frac{1}{T} \hat{\Theta}_0(X_i, X'_i) \tilde{T}_\theta^\lambda(X'_i, \tau) \nabla \theta_0 f_0(X'_i) \nabla \theta_0 f_0(X'_j)^T \\
&= \hat{\Theta}_0(X_i, X_j)
\end{align*}
\]

\[
\begin{align*}
&= \hat{\Theta}_0(X_i, X'_i) \tilde{T}_\theta^\lambda(X'_i, \tau) \hat{\Theta}_0(X'_i, X'_j) \tilde{T}_\theta^\lambda(X'_j, \tau)^T \hat{\Theta}_0(X'_j, X_j) \\
&- \hat{\Theta}_0(X_i, X'_i) \tilde{T}_\theta^\lambda(X'_i, \tau)^T \hat{\Theta}_0(X'_j, X_j) \\
&- \hat{\Theta}_0(X_i, X'_i) \tilde{T}_\theta^\lambda(X'_i, \tau) \hat{\Theta}_0(X'_i, X_j)
\end{align*}
\]

(90)
where we used the equivalences $\hat{\Theta}_0(\cdot, *) = \hat{\Theta}_0(\cdot, \cdot)^T$ and $\frac{1}{k} \nabla_{\theta_0} f_0(\cdot) \nabla_{\theta_0} f_0(\cdot) = \hat{\Theta}_0(\cdot, \cdot)$.

By Algebraic Limit Theorem for Functional Limits, we have

$$\lim_{l \to \infty} \hat{\Phi}_{ij} = \lim_{l \to \infty} \Theta_0(X_i, X_j) \cdot \hat{\Phi}_{ij}$$

where $\Theta_0(\cdot, \cdot)$ is a deterministic kernel function, the Neural Tangent Kernel function (NTK) from the literature on supervised learning [4, 26, 37]. Specifically, $\hat{\Theta}_0(\cdot, \cdot)$ converges to $\Theta_0(\cdot, \cdot)$ in probability as the width $l$ approaches infinity.

Hence, for any $i, j \in [N]$, as the width $l$ approaches infinity, $\hat{\Phi}_{ij}$ converges in probability to a deterministic matrix $\lim_{l \to \infty} \hat{\Phi}_{ij}$, as shown by (91). Thus, the whole block matrix $\hat{\Phi}_0$ converges in probability to a deterministic matrix in the infinite width limit. Denote $\Phi = \lim_{l \to \infty} \hat{\Phi}_0$, then we know $\Phi$ is a deterministic matrix with each block expressed as (91).

Since $\hat{\Phi}_0 \equiv \hat{\Phi}_0((X, X', Y'), (X, X', Y')) = \frac{1}{k} \nabla_{\theta_0} f_0(X, X', Y') \nabla_{\theta_0} f_0(X, X', Y')^T$, it is a symmetric square matrix. Hence all eigenvalues of $\hat{\Phi}_0$ are greater or equal to 0, which also holds true for $\Phi$. In addition, because of Assumption 4, $\Phi$ is positive definite, indicating $\sigma_{\text{max}}(\Phi) > 0$. On the other hand, from [4], we know diagonal entries and eigenvalues of $\Theta_0(\cdot, \cdot)$ are positive real numbers upper bounded by $O(L)$, as a direct result, it is easy to verify that the diagonal entries of the matrix $\Phi$ are also upper bounded, indicating $\sigma_{\text{max}}(\Phi) < \infty$. Hence, we have $0 < \sigma_{\text{min}}(\Phi) < \sigma_{\text{max}}(\Phi) < \infty$.

**Extension.** It is easy to extend (91), the expression for $\Phi \equiv \lim_{l \to \infty} \hat{\Phi}_0((X, X', Y'), (X, X', Y'))$, to more general cases. Specifically, we can express $\Phi(\cdot, \cdot)$ analytically for arbitrary inputs. To achieve this, let us define a kernel function, $\phi: \mathbb{R}^{n \times k} \times \mathbb{R}^{m \times k} \times \mathbb{R}^{n \times k} \times \mathbb{R}^{m \times k} \mapsto \mathbb{R}^{kn \times kn}$ such that

$$\phi((\cdot, \cdot), (\cdot, \cdot)) = \Theta(\cdot, \cdot) + \Theta(\cdot, \cdot) \tilde{T}_{\Theta}(\cdot, \cdot) \Theta(\cdot, \cdot) \tilde{T}_{\Theta}(\cdot, \cdot)^T \Theta(\cdot, \cdot)$$

$$- \Theta(\cdot, \cdot) \tilde{T}_{\Theta}(\cdot, \cdot) \Theta(\cdot, \cdot) - \Theta(\cdot, \cdot) \tilde{T}_{\Theta}(\cdot, \cdot)^T \Theta(\cdot, \cdot).$$

Then, it is obvious that for $i, j \in [N]$, the $(i, j)$-th block of $\Phi$ can be expressed as $[\Phi]_{ij} = \phi((X_i, X'_i), (X_j, X'_j))$.

For cases such as $\Phi((X, X'), (X, X')) \in \mathbb{R}^{kn \times kn}$, it is also obvious that $\Phi((X, X'), (X, X'))$ is a block matrix that consists of $1 \times N$ blocks of size $kn \times kn$, with the $(1, j)$-th block as follows for $j \in [N]$, $\Phi((X, X'), (X, X'))_{1,j} = \phi((X, X'), (X_j, X_j))$.

**B.5. Proof of Theorem 3**

**Proof of Theorem 3.** Based on these lemmas presented above, we can prove Theorem 3.

Lemma 2 indicates that there exist $R_0$ and $l^*$ such that for any width $l \geq l^*$, the following holds true over random initialization with probability at least $(1 - \delta_0/10)$,

$$\|g(\theta_0)\|_2 \leq R_0.$$

Consider $C = \frac{3K}{\sigma_0}$ in Lemma 1.
First, we start with proving (29) and (32) by induction. Select $\tilde{l} > l^*$ such that (93) and (25) hold with probability at least $1 - \frac{\delta_0}{5}$ over random initialization for every $l \geq \tilde{l}$. As $t = 0$, by (28) and (25), we can easily verify that (29) and (32) hold true

\[
\begin{aligned}
\|g(\theta_0)\|_2 &\leq R_0 \\
\|g(\theta_0)\|_2 &\leq \frac{K\eta_0}{\sqrt{l}} R_0.
\end{aligned}
\]

Assume (29) and (32) hold true for any number of training step $j$ such that $j < t$. Then, by (25) and (32), we have

\[
\|g(\theta_{t+1} - \theta_t)\|_2 \leq \eta\|J(\theta_t)\|_2 \|g(\theta_t)\|_2 \leq \frac{K\eta_0}{\sqrt{l}} R_0.
\]

Beside, with the mean value theorem and (28), we have the following

\[
\|g(\theta_{t+1})\|_2 = \|g(\theta_{t+1} - g(\theta_t) + g(\theta_t))\|_2
\leq \|I - \eta J(\theta_t)J(\theta_t)\|_2 \|g(\theta_t)\|_2
\leq \|I - \eta J(\theta_t)J(\theta_t)\|_2 \|g(\theta_t)\|_2
\leq \|I - \eta J(\theta_t)J(\theta_t)\|_2 \leq \left(1 - \frac{\eta_0\sigma_{\min}}{3}\right)^t R_0.
\]

where we define $\theta_t^\mu$ as a linear interpolation between $\theta_t$ and $\theta_{t+1}$ such that $\theta_t^\mu := \mu\theta_t + (1 - \mu)\theta_{t+1}$ for some $0 < \mu < 1$

Now, we will show that with probability $1 - \frac{\delta_0}{2}$,

\[
\|I - \eta J(\theta_t^\mu)J(\theta_t^\mu)\|_2 \leq 1 - \frac{\eta_0\sigma_{\min}}{3}.
\]

Recall that $\hat{\Phi}_0 \rightarrow \Phi$ in probability, proved by Lemma 3. Then, there exists $\bar{l}$ such that the following holds with probability at least $1 - \frac{\delta_0}{5}$ for any width $l > \bar{l}$,

\[
\|\Phi - \hat{\Phi}_0\|_F \leq \frac{\eta_0\sigma_{\min}}{3}.
\]

Our assumption $\eta_0 < \frac{2}{\sigma_{\max} + \sigma_{\min}}$ makes sure that

\[
\|I - \eta_0 \Phi\|_2 \leq 1 - \eta_0\sigma_{\min}.
\]

Therefore, as $l \geq (\frac{18K^3R_0}{\sigma_{\min}})^2$, with probability at least $1 - \frac{\delta_0}{2}$ the following holds,

\[
\begin{aligned}
\|I - \eta J(\theta_t^\mu)J(\theta_t^\mu)\|_2 &= \|I - \eta_0 \Phi + \eta_0 \Phi - \hat{\Phi}_0 + \eta (J(\theta_0)J(\theta_0)^\top - J(\theta_t^\mu)J(\theta_t^\mu)^\top)\|_2 \\
&\leq \|I - \eta_0 \Phi\|_2 + \eta_0\|\Phi - \hat{\Phi}_0\|_2 + \eta\|J(\theta_0)J(\theta_0)^\top - J(\theta_t^\mu)J(\theta_t^\mu)^\top\|_2 \\
&\leq 1 - \eta_0\sigma_{\min} + \frac{\eta_0\sigma_{\min}}{3} + \eta_0 K^2 \|\theta_t - \theta_0\|_2 + \|\theta_t^\mu - \theta_0\|_2 \\
&\leq 1 - \frac{\eta_0\sigma_{\min}}{3} + \frac{\eta_0\sigma_{\min}}{3} + \frac{6\eta_0 K^3R_0}{\sigma_{\min} \sqrt{l}} \\
&\leq 1 - \frac{\eta_0\sigma_{\min}}{3}
\end{aligned}
\]

where we used the equality $\frac{1}{2}J(\theta_0)J(\theta_0)^\top = \hat{\Phi}_0$.

Hence, as we choose $\Lambda = \max\{l^*, \bar{l}, \frac{18K^3R_0}{\sigma_{\min}}\}$, the following holds for any width $l > \Lambda$ with probability at least $1 - \frac{\delta_0}{5}$ over random initialization

\[
\|g(\theta_{t+1})\|_2 \leq \|I - \eta J(\theta_t^\mu)J(\theta_t^\mu)\|_2 \leq \left(1 - \frac{\eta_0\sigma_{\min}}{3}\right)^t R_0 \leq \left(1 - \frac{\eta_0\sigma_{\min}}{3}\right)^{t+1} R_0,
\]

(94)
which finishes the proof (32).

Finally, we prove (30) by

\[
\|\hat{\Phi}_0 - \hat{\Phi}_t\|_F = \frac{1}{T} \|J(\theta_0)J(\theta_0)^\top - J(\theta_t)J(\theta_t)^\top\|_F \\
\leq \frac{1}{T} \|J(\theta_0)\|_{op} \|J(\theta_0)^\top - J(\theta_t)^\top\|_F + \frac{1}{T} \|J(\theta_t) - J(\theta_0)\|_{op} \|J(\theta_t)^\top\|_F \\
\leq 2K^2\|\theta_0 - \theta_t\|_2 \\
\leq \frac{6K^3R_0}{\sigma_{\min}\sqrt{T}}
\]

where we used (29) and Lemma 1.

C. Analytical Expression of MAML Output

In this section, we will present Corollary 3.1. Briefly speaking, with the help of Theorem 3, we first show the training dynamics of MAML with over-parameterized DNNs can be described by a differential equation, which is analytically solvable. By solving this differential equation, we obtain the expression for MAML output on any training or test task.

Remarks. This corollary implies for a sufficiently over-parameterized neural network, the training of MAML is determined by the parameter initialization, \(\theta_0\). Given access to \(\theta_0\), we can compute the functions \(\Phi_0\) and \(F_0\), and then the trained MAML output can be obtained by simple calculations, without the need for running gradient descent on \(\theta_0\). This nice property enables us to perform a deeper analysis on MAML with DNNs.

Corollary 3.1 (MAML Output (Corollary 3.1 Restated)). In the setting of Theorem 1, the training dynamics of the MAML can be described by a differential equation

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

where we denote \(F_t \equiv F_0\), and \(\hat{\Phi}_0 \equiv \hat{\Phi}_{\theta_0}((\mathcal{X}, \mathcal{X}', \mathcal{Y}'), (\mathcal{X}, \mathcal{X}', \mathcal{Y}'))\) for convenience.

Solving this differential equation, we obtain the meta-output of MAML on training tasks at any training time as

\[
F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}') = (I - e^{-\eta\hat{\Phi}_0t})\mathcal{Y} + e^{-\eta\hat{\Phi}_0t}F_0(\mathcal{X}, \mathcal{X}', \mathcal{Y}')
\]

Similarly, on arbitrary test task \(\mathcal{T} = (\mathcal{X}, \mathcal{Y}, \mathcal{X}', \mathcal{Y}')\), the meta-output of MAML is

\[
F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}') = F_0(\mathcal{X}, \mathcal{X}', \mathcal{Y}') + \hat{\Phi}_0(\mathcal{X}, \mathcal{X}', \mathcal{Y}')T_{\hat{\Phi}_0}^0(t) (\mathcal{Y} - F_0(\mathcal{X}, \mathcal{X}', \mathcal{Y}'))
\]

where \(\hat{\Phi}_0(\cdot) \equiv \hat{\Phi}_{\theta_0}(\cdot, (\mathcal{X}, \mathcal{X}', \mathcal{Y}'))\) and \(T_{\hat{\Phi}_0}^0(t) \equiv \hat{\Phi}_0^{-1}(I - e^{-\eta\hat{\Phi}_0t})\) are shorthand notations.

Proof. For the optimization of MAML, the gradient descent on \(\theta_t\) with learning rate \(\eta\) can be expressed as

\[
\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L(\theta_t) \\
= \theta_t - \frac{1}{2} \eta \nabla_{\theta_t} \|F_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y}\|^2_2 \\
= \theta_t - \eta \nabla_{\theta_t} F_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}')^\top (F_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y})
\]

Since the learning rate \(\eta\) is sufficiently small, the discrete-time gradient descent above can be re-written in the form of continuous-time gradient descent (i.e., gradient flow),

\[
\frac{d\theta_t}{dt} = -\eta \nabla_{\theta_t} F_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}')^\top (F_{\theta_t}(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y})
\]

Then, the training dynamics of the meta-output \(F_t(\cdot) \equiv F_{\theta_t}(\cdot)\) can be described by the following differential equation,

\[
\frac{dF_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}')}{dt} = \nabla_{\theta_t} F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}') \frac{d\theta_t}{dt} \\
= -\eta \nabla_{\theta_t} F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}') \nabla_{\theta_t} F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}')^\top (F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y}) \\
= -\eta \hat{\Phi}_t (F_t(\mathcal{X}, \mathcal{X}', \mathcal{Y}') - \mathcal{Y})
\]
where \( \hat{\Phi}_t = \hat{\Phi}_t((X, X'), (X, X'), (X, X')) = \nabla_{\theta_t} F_t(X, X', Y') \nabla_{\theta_t} F_t(X, X', Y')^\top \).

On the other hand, Theorem 3 gives the following bound in (30),

\[
\sup_t \|\hat{\Phi}_0 - \hat{\Phi}_t\|_F \leq \frac{6K^3 R_0}{\sigma_{\min}} t^{-\frac{1}{2}},
\]

indicating \( \hat{\Phi}_t \) stays almost constant during training for sufficiently over-parameterized neural networks (i.e., large enough width \( l \)). Therefore, similar to [37], we can replace \( \hat{\Phi}_t \) by \( \hat{\Phi}_0 \) in (98), and get

\[
\frac{dF_t(X, X', Y')}{dt} = -\eta \hat{\Phi}_0 \left( F_t(X, X', Y') - Y' \right),
\]

which is an ordinary differential equation (ODE) for the meta-output \( F_t(X, X', Y') \) w.r.t. the training time \( t \).

This ODE is analytically solvable with a unique solution. Solving it, we obtain the meta-output on training tasks at any training time \( t \) as,

\[
F_t(X, X', Y') = (I - e^{-\eta \Phi_0 t}) Y + e^{-\eta \Phi_0 t} F_0(X, X', Y').
\]

The solution can be easily extended to any test task \( T = (X, Y, X', Y') \), and the meta-output on the test task at any training time is

\[
F_t(X, X', Y') = F_0(X, X', Y') + \hat{\Phi}_0(X, X', Y') T_{\Phi_0}^\eta(t) (Y - F_0(X, X', Y')),
\]

where \( \hat{\Phi}_0(\cdot) = \hat{\Phi}_{\theta_0}(\cdot, (X, X', Y')) \) and \( T_{\Phi_0}^\eta(t) = \hat{\Phi}_0^{-1}(I - e^{-\eta \Phi_0 t}) \) are shorthand notations.

\[\square\]

**D. Gradient-Based Meta-Learning as Kernel Regression**

In this section, we first make an assumption on the scale of parameter initialization, then we restate Theorem 2. After that, we provide the proof for Theorem 2.

[37] shows the output of a neural network randomly initialized following (16) is a zero-mean Gaussian with covariance determined by \( \sigma_w \) and \( \sigma_b \), the variances corresponding to the initialization of weights and biases. Hence, small values of \( \sigma_w \) and \( \sigma_b \) can make the outputs of randomly initialized neural networks approximately zero. We adopt the following assumption from [4] to simplify the expression of the kernel regression in Theorem 2.

**Assumption 5** (Small Scale of Parameter Initialization). The scale of parameter initialization is sufficiently small, i.e., \( \sigma_w, \sigma_b \) in (16) are small enough, so that \( f_0(\cdot) \approx 0 \).

Note the goal of this assumption is to make the output of the randomly initialized neural network negligible. The assumption is quite common and mild, since, in general, the outputs of randomly initialized neural networks are of small scale compared with the outputs of trained networks [37].

**Theorem 4** (MAML as Kernel Regression (Theorem 2 Restated)). Suppose learning rates \( \eta \) and \( \lambda \) 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,

\[
F_t(X, X', Y') = G_0^\Theta(X, X', Y') + \Phi((X, X'), (X, X')) T_\Phi^\eta(t) (Y - G_0^\Theta(X, X', Y'))
\]

(103)

where \( G \) is a function defined below, \( \Theta \) is the neural tangent kernel (NTK) function from [26] that can be analytically calculated without constructing any neural net, and \( \Phi \) is a new kernel, which name as Meta Neural Kernel (MNK). The expression for \( G \) is

\[
G_0^\Theta(X, X', Y') = \Theta(X, X') T_\Theta(X', \tau) Y'.
\]

(104)

where \( T_\Theta(\cdot, \tau) := \Theta(\cdot)^{-1}(I - e^{-\lambda \Theta(\cdot, \tau)}) \). Besides, \( G_0^\Theta(X, X', Y') = (G_0^\Theta(X_i, X'_i, Y'_i))_{i=1}^N \).

The MNK is defined as \( \Phi \equiv \Phi((X, X'), (X, X')) \in \mathbb{R}^{knN \times knN} \), which is a block matrix that consists of \( N \times N \) blocks of size \( kn \times kn \). For \( i, j \in [N] \), the \((i, j)\)-th block of \( \Phi \) is

\[
[\Phi]_{ij} = \phi((X_i, X'_i), (X_j, X'_j)) \in \mathbb{R}^{kn \times kn},
\]

(105)
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
\[
\phi((\cdot,\cdot), (\cdot,\cdot)) = \Theta((\cdot,\cdot) + \Theta((\cdot,\cdot)\mathcal{T}_\Theta^\lambda(\cdot,\tau))\Theta((\cdot,\cdot)\mathcal{T}_\Theta^\lambda(\cdot,\tau)^\top \Theta((\cdot,\cdot) - \Theta((\cdot,\cdot)\mathcal{T}_\Theta^\lambda(\cdot,\tau))\Theta((\cdot,\cdot)\mathcal{T}_\Theta^\lambda(\cdot,\tau)^\top \Theta((\cdot,\cdot)).
\]
(106)

Here \(\Phi((X, X'), (X', X')) \in \mathbb{R}^{kn \times kn}\) 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 \(j \in [N]\),
\[
[\Phi((X, X'), (X', X'))]_{1,j} = \phi((X, X'), (X_j, X'_j)).
\]
(107)

Proof. First, (8) shows that the output of MAML on any test task \(T = (X, Y, X', Y')\) can be expressed as
\[
F_t(X, X', Y') = F_0(X, X', Y') + \hat{\Theta}_0(X, X', Y')T_{\hat{\theta}_0}^n(t)(Y - F_0(X, X', Y'))
\]
(108)

Note (88) in Appendix B.4 shows that
\[
F_0(X, X', Y') = f_0(X) + \hat{\Theta}_0(X, X')\mathcal{T}_\hat{\Theta}^\lambda(X', \tau)(Y' - f_0(X'))
\]
(109)

With Assumption 5, we can drop the terms \(f_0(X)\) and \(f_0(X')\) in (109). Besides, from \([4, 26, 37]\), we know \(\lim_{\tau \to \infty} \hat{\Theta}_0(\cdot, \cdot) = \Theta(\cdot, \cdot)\), the Neural Tangent Kernel (NTK) function, a deterministic kernel function. Therefore, \(F_0(X, X', Y')\) can be described by the following function as the width approaches infinity.
\[
\lim_{\tau \to \infty} F_0(X, X', Y') = G_\Theta^+ (X, X', Y') = \Theta(X, X')\mathcal{T}_\Theta^\lambda(X', \tau)Y'.
\]
(110)

where \(\mathcal{T}_\Theta^\lambda(\cdot, \tau) := \Theta(\cdot, \cdot)^{-1}(I - e^{-\lambda \Theta(\cdot, \cdot) \tau})\). Besides, \(G_\Theta^+(X, X', Y') = (G_\Theta^+ (X_i, X'_i, Y'_i))_{i=1}^N\).

In addition, from Lemma 3, we know \(\lim_{\tau \to \infty} \phi_0(\cdot, \cdot) = \Phi(\cdot, \cdot)\). Combined this with (110), we can express (108) in the infinite width limit as
\[
F_t(X, X', Y') = G_\Theta^+(X, X', Y') + \Phi((X, X'), (X', X'))T_{\hat{\theta}_0}^n(t)(Y - G_\Theta^+(X, X', Y'))
\]
(111)

where \(\Phi(\cdot, \cdot)\) is a kernel function that we name as Meta Neural Kernel function. The derivation of its expression shown in (105)-(107) can be found in Appendix B.4.

\[\Box\]

ANIL Kernel The above theorem derives the analytical expression of the kernel induced by MAML. Certainly that variants of MAML will induce kernels with (slightly) different expressions. A recent impactful variant of MAML is Almost-No-Inner-Loop (ANIL) \([50]\). ANIL is a simplification of MAML that retains the performance of MAML while enjoying a significant training speedup. The key idea of ANIL is to remove the inner-loop updates on the hidden layers; in other words, ANIL only update the last linear layer in the inner loop, resulting in a much smaller computation and memory cost compared with MAML. Following procedures in Appendix C and D, one can straightforwardly derive the expression of the kernel induced by ANIL, which just replaces Eq. (106) (kernel function induced by MAML) by
\[
\phi((\cdot, \cdot), (\cdot, \cdot)) = \Theta((\cdot, \cdot) + \mathcal{K}(\cdot, \cdot)\mathcal{T}_\lambda^\lambda(\cdot, \tau)\Theta((\cdot, \cdot)\mathcal{T}_\lambda^\lambda(\cdot, \tau)^\top \mathcal{K}(\cdot, \cdot) - \mathcal{K}(\cdot, \cdot)\mathcal{T}_\lambda^\lambda(\cdot, \tau)^\top \mathcal{K}(\cdot, \cdot)\mathcal{T}_\lambda^\lambda(\cdot, \tau))\Theta((\cdot, \cdot)\mathcal{T}_\lambda^\lambda(\cdot, \tau)^\top \mathcal{K}(\cdot, \cdot)\mathcal{T}_\lambda^\lambda(\cdot, \tau)).
\]
(112)

where \(\mathcal{K}\) is the neural tangent kernel function corresponds to neural networks with frozen hidden layers (i.e., only the last linear layer is optimized by gradient descent). The appearance of \(\mathcal{K}\) directly results from the special inner-loop update rule of ANIL (i.e., only updates the last linear layer in the inner loop).

E. More Details on Experiments

Training Data Augmentation Following previous few-shot learning works \([38, 60]\), in the training stage, we adopt data augmentation operations, including random cropping, color jittering, and random horizontal flip.

Training Batch Size For all 5-cells experiments, a batch size of 64 is used. For 8-cells experiments, we set the batch size to 40 for miniImageNet and 56 for tieredImageNet to accommodate the GPU memory of a single GPU card.
**Dropout Rate**  We use dropout rate of 0.2 and 0.1 for miniImageNet and tieredImageNet, respectively. Following DARTS [42], we gradually increase the dropout rate during the training.

**Normalization Layers**  To enable efficient computation of per-sample-gradients with Opacus [76] (it does not support BatchNorm so far), we first convert all the BatchNorm [25] layers to GroupNorm [70] layers with 16 number of groups in the search stage. After obtaining the cells, we train and evaluate the selected architectures with BatchNorm layers.

**Hyper-parameters for Computing MetaNTK**  MAML kernels (defined in Definition 2)) and ANIL kernels (defined in Eq. (112)) are used for 5-cells and 8-cells experiments, respectively. To write more concisely, We denote the product of inner loop learning rate and training time as $\lambda \tau$. An $\lambda \tau = \infty$ and a regularization coefficient of 0.001 are used for all 5-cells experiments. For 8-cells experiments, an $\lambda \tau = 1$ and a kernel regularization coefficient of $10^{-5}$ are used for miniImageNet experiments while an $\lambda \tau = \infty$ and a kernel regularization coefficient of 0.001 are used for tieredImageNet experiments.

**Hyper-parameters for Evaluation**  In the evaluation stage, we fine-tune the last layer of the learned neural net on the labelled support samples of each test task, and then evaluate its prediction accuracy on the query samples. Following the evaluation strategies of RFS [60], (i) we normalize the last hidden layer output of each sample to unit norm before passing to the last layer during the evaluation; (ii) we enlarge the set of support samples by applying data augmentation (used in the training stage) to create 5x augmented support samples for fine-tuning. We use cross-entropy loss and hinge loss for the fine-tuning, both with $\ell_2$ regularization. For cross-entropy fine-tuning, we use the Logistic Regression (LR) solver provided in scikit-learn [49]; for the hinge loss fine-tuning, we adopt the C-Support Vector Classification (SVC) with linear kernel provided in scikit-learn [49]. Notice that these the $\ell_2$ regularization in scikit-learn solvers is controlled by a regularization parameter $C = \frac{1}{\ell_2 \text{penalty}}$. On mini-ImageNet: (i) in the 5-cells case, we use SVC with $C = 0.2$ for 1-shot and LR with $C = 0.6$ for the 5-shot experiments; (ii) in the 8-cells case, we use SVC with $C = 0.35$ for 1-shot and LR with $C = 0.4$ for the 5-shot experiments. On tiered-ImageNet: (i) in the 5-cells case, we use SVC with $C = 0.75$ for 1-shot and LR with $C = 0.8$ for the 5-shot experiments; (ii) in the 8-cells case, we use LR with $C = 0.95$ for 1-shot and LR with $C = 0.5$ for the 5-shot experiments.