ENFORCING ZERO-HESSIAN IN META-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient-Based Meta Learning (GBML) enables us to get task-specific parameters with few-labeled datapoints in an inner loop. However, it has not yet been discussed how GBML can adapt to a new task within a few optimization steps with a huge learning rate in the inner loop. We find that the gradient does not change from the beginning to the end of the inner loop, meaning that it behaves like a linear model. In this paper, we argue that this characteristic is an essential key to understanding convergence in inner loops with huge learning rates. Also, we show that gradient-based meta-learning can be interpreted as metric-based meta-learning when we adopt our hypothesis that linearity in the inner loop is the key to operating GBML. To empirically prove and exploit our hypothesis, we propose a regularization-based algorithm called enforcing Linearity in the Inner Loop (LIL) which exploits our observation and can be applied to any baselines that have the form of GBML. LIL proves its potential by showing its boosted performance not only on top of general baselines in various architectures but also on adverse or Hessian-free baselines. Qualitative experiments are also conducted to explain the performance of LIL.

1 INTRODUCTION

With the advent of *deep learning revolution* in the 2010s, machine learning has widened its application to many areas and outperformed humans in some areas. Nevertheless, machines are still far behind humans in the ability to learn by itself, in that humans can learn concepts of new data with only a few samples, while machines cannot. Meta-Learning deals with this problem of *learning to learn.* An approach to address this issue is the metric-based methods (Chen & He, 2021; Snell et al., 2017; Bromley et al., 1993; Grill et al., 2020; Chen et al., 2020; Sung et al., 2018; Radford et al., 2021), which aims to learn 'good' kernels in order to project given data into a well-defined feature space. Another popular line of research is optimization-based methods (Finn et al., 2017; Ravi & Larochelle, 2017; Rusu et al., 2018a), so-called Gradient-Based Meta-Learning (GBML), which aims to achieve the goal of meta-learning by gradient descent.

As the most representative GBML methods applicable to any model trained with the gradient descent process, model-agnostic meta-learning (MAML) (Finn et al., 2017) and its variations (Raghu et al., 2019; Oh et al., 2020; Rusu et al., 2018b) exploit nested loops to find a good meta-initialization point from which new tasks can be rapidly optimized. To do so, it divides the problem into two loops: the inner loop (task-specific loop) and the outer loop (meta loop). The former tests the meta-learning property of fast learning, and the latter moves the meta-initialization point by observing the dynamics of the inner loop. The general training is performed by alternating the two loops. In this paper, we focus on the gradient-based meta-learning methods.

To properly evaluate the meta-learning property in the inner loop, GBML samples a new problem for each inner loop. Fig. 1 shows how the problem is formulated. For every inner loop, the task is sampled from the task distribution. The classes constituting each task are randomly sampled, and the configuration order (class order) is also random. This means that the model tackles an entirely new task at the beginning of each inner loop. Since the task is unknown and the model has to solve the problem within a few gradient steps (some extreme algorithms (Raghu et al., 2019; Oh et al., 2020) solve the problem in one gradient step), the model usually exploits large learning rates to adapt rapidly. For example, MAML (Finn et al., 2017) uses a learning rate of 0.4 for the Omnigot dataset, and Raghu et al. (2019) exploits 0.1, which is hundreds times larger compared to the learning rate for the outer loop, which is in the order of 1e-3. Furthermore, this learning rate is much larger compared

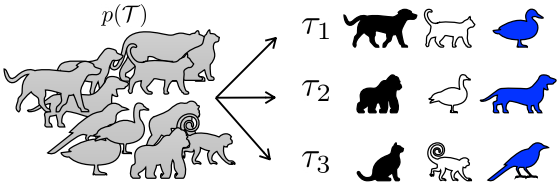

Figure 1: For the i-th inner loop in meta-learning, the task, τ_i, is sampled from the task distribution: N (e.g. 3) classes constituting each task are randomly sampled with a random configuration order, meaning that the model does not have any information before it retrieves an unseen task.

to modern deep learning techniques, for example, ViT (Dosovitskiy et al., 2020) exploits the learning rate in the scale of 1e-6, which means that GBML exploits millions times larger learning rate compared to this. Although exploiting a large learning rate can move parameters quickly in SGD, it causes instability due to the high-order terms such as Hessian (LeCun et al., 2015). This problem gets worse at the initial training phase and some deep-learning methodologies use a heuristic that exploits a smaller learning rate at the beginning to alleviate this problem (Dosovitskiy et al., 2020; He et al., 2018). Thus, we expect the problem caused by high-order terms may be severe with a large learning rate in each inner loop since GBML initially has no knowledge about the given task. However, it seems that GBML does not suffer from those problems regarding its performance.

In this paper, we aim to answer the corresponding phenomenon by analysing the inner loop of GBML. We observed that the gradient of a GBML model is almost constant within an inner loop. This means the model becomes linear after the whole process of few-shot learning. Also, we explain intuitively that this kind of phenomenon is the key property to solving the implicit risk of a large learning rate in the inner loop. Then, we show that the corresponding result makes GBML a variant of metric-based meta learning. Although there exist some algorithms exploiting the same observation as ours implicitly (Nichol et al., 2018; Finn et al., 2017), to the best of our knowledge, we are the first to explicitly quantify linearity of the inner loop and investigate the connection between the GBML model with linearity in the inner loop and the metric-based meta learning.

Our hypothesis that GBML implicitly drives the inner loop's loss surface linear hints a desiderata of a GBML algorithm. Thus, if we can inject this linearity explicitly into the model, we expect that it will converge faster and get better performance. So we propose an algorithm called *Linearity in the Inner Loop* (LIL) which enforces the model to have zero Hessian in the inner loop. Our LIL exploits two regularization losses which measure non-linearity in the inner loop.

Since LIL is a regularization method, it can be applied to any GBML baselines. We empirically validate LIL in standard few-shot learning tasks (miniImagenet, tieredImagenet, Cars, CUB (Vinyals et al., 2016; Ren et al., 2018; Krause et al., 2013; Welinder et al., 2010)) and also test its cross-domain ability with not only general baselines but also adverse or Hessian-free baselines to show that LIL can boost GBML in an algorithm- and architecture-independent manner. Finally, we show that LIL behaves as we expected by analyzing its dynamics.

2 Related Works

Meta Learning is a methodology which aims to learn to learn by itself (Thrun & Pratt, 2012; Schmidhuber, 1987). In the meta-learning community, two main lines of researches are actively studied: *i.e.*, metric-based meta-learning (MBML) and gradient-based meta-learning (GBML).

The core concept of MBML is similar to classic machine learning algorithms such as nearest-neighbor classifiers or support vector machines in the sense that it tries to learn 'good' feature embeddings which can be easily separated by simple algorithms or shallow networks (Snell et al., 2017; Grill et al., 2020). Thus, MBML can be interpreted as a methodology for learning a 'kernel' since its goal is to learn a good encoder. To achieve this property, one tries to cluster few-shot images of the same class by making the corresponding prototype and train the encoder to make the encoded samples cluster well around the prototype (Snell et al., 2017; Liu et al., 2020). Another line of research is self-supervised learning (Grill et al., 2020; Chen et al., 2020; Chen & He, 2021; Bromley et al., 1993), which scores state-of-the-art performance on few-shot learning nowadays. They want an encoder to have the ability to extract general knowledge from the perspective of mutual information. Since their goal is to learn a 'good' encoder, they usually attach an additional module to the encoder to construct an appropriate decision boundary for a given task.

GBML, or optimization-based meta-learning is a type of methodology which wants to mimic people's rapid learning skill. Thus, a model should learn with a small number of samples and gradient steps. To achieve this property, Ravi & Larochelle (2017) tried to save meta-data to recurrent networks. Recently, MAML-based methods (Finn et al., 2017; Rajeswaran et al., 2019; Rusu et al., 2018b; Baik et al., 2020; Bernacchia, 2021) are typically considered as a synonym of GBML consisting of nested loops: outer loop (meta-loop) and inner loop (task-specific loop). They sample few-shot classification tasks to test the meta-learning property in the inner loop and update the meta-initialization parameters in the outer loop with SGD. To learn rapidly, they exploit a more-than-thousand times larger learning rate in the inner loop compared to modern deep learning settings such as ViT (Dosovitskiy et al., 2020) and CLIP (Radford et al., 2021). Although some researches have proved their convergence property (Fallah et al., 2020; Wang et al., 2020), they set infinitesimal size of learning rate to prove convergence. Fallah et al. (2020) set the learning rate considering the Hessian *i.e.*, setting the learning rate small enough so that the higher-order terms do not affect the convergence and Wang et al. (2020) proved the convergence with the assumption that the learning rate goes to zero. Although some lines of research have shown algorithms dealing with learning rates (Baik et al., 2020; Bernacchia, 2021), they did not explain the success of a large learning rate. Baik et al. (2020) used an additional model to predict a proper task-specific learning rate. Raghu et al. (2019); Oh et al. (2020) tried to explain GBML with the dynamics of the encoder's output features. Although both used the same feature-level perspective, they argued in exactly different ways. Raghu et al. (2019) argued feature reuse is an essential component while Oh et al. (2020) said that feature adaption is crucial. Both algorithms can be considered as a variant of preconditioning meta-learning since they freeze most layers in the inner loop, thereby frozen layers can be considered as warp-parameters (Flennerhag et al., 2019).

In this paper, we attempt to connect between metric-based meta-learning and gradient-based meta-learning based on the observation that GBML favors a linear loss surface. With our hypothesis, we propose new loss terms which inject our linearity hypothesis as prior into the model. The proposed *Linearity in the Inner Loop* (LIL) can be applied in an architecture- and algorithm-independent way. Since our proposed hypothesis is valid for any algorithm that takes advantage of a large learning rate in the inner loop, LIL has a large potential of application.

3 PRELIMINARIES: GBML

We consider the few-shot classification task of N-way K-shot, which requires learning K samples per class to distinguish N classes in each inner loop. In few-shot GBML, we can apply gradient descent for only a limited number of steps for fast adaptation.

Let $f(x|\theta) \in \mathbb{R}^N$ be the output of the classifier parameterized by θ for a given input x. In this setting, f can be considered as the logit before the softmax operation. The model parameters θ are updated by the loss, $L(x, y|\theta) = D(s(f(x|\theta)), y)$, where x and y are the input and the corresponding label, $s(\cdot)$ is the softmax operation and $D(\cdot, \cdot)$ is some distance metric.

As shown in Fig. 1, at each inner loop, we sample a task $\tau \sim p(\mathcal{T})$, an N-way classification problem consisting of two sets of labeled data: $\mathcal{D}^\tau_{support}$ (support set) and $\mathcal{D}^\tau_{target}$ (target set). In the inner loop, the goal is to find the task-specific parameter $\theta^\star_\tau$ which minimizes the loss L for the support set $\mathcal{D}^\tau_{support}$ given the initial parameter θ_0 as follows:

$$\text{Inner Loop:} \quad \theta^\star_\tau = \arg\min_\theta \sum_{(x,y)\in\mathcal{D}^\tau_{support}} L(x, y|\theta; \theta_0). \tag{1}$$

Then, for the outer loop, we optimize the initialization parameters θ_0 which improve the performance of $\theta^\star_\tau$ on $\mathcal{D}_{target}$ as follows:

$$\text{Outer Loop:} \quad \theta^\star_0 = \arg\min_{\theta_0} \sum_\tau \sum_{(x,y)\in\mathcal{D}^\tau_{target}} L(x, y|\theta^\star_\tau; \theta_0). \tag{2}$$

Although GBML gained its success with this setting, this success is quite curious: since the N classes constituting each task are randomly sampled and the configuration (class) order is also random, 'What characteristics of meta-initialization θ_0 make GBML work?' remains quite uncertain.

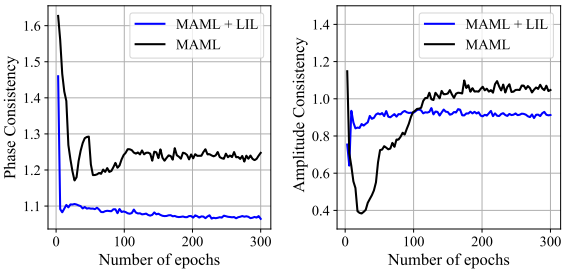

Figure 2: Phase Consistency (PC) and Amplitude Consistency (AC) measure the linearity in the whole trajectory in an inner loop. A model behaves like a linear model when both metrics go to '1'. We observed that MAML behaves almost like a linear model in that both PC and AC settle near 1 as (outer) iterations go on and propose to further 'enforce' linearity in the inner loop (LIL).

4 LINEARITY IS ESSENTIAL IN THE INNER LOOP

In this section, we will show that a GBML model behaves like a linear model in the inner loop as training continues. *i.e.*, the orientation and the magnitude of the gradient do not vary much in different iterations in an inner loop. This means that a loss surface is almost linear along the trajectory of the inner loop. We argue that this phenomenon explains how GBML achieves convergence although it exploits an extremely large learning rate with only a few gradient steps. To support our hypothesis, we intuitively show that linearity in the inner loop helps convergence. Although our hypothesis is implicitly assumed in previous papers (Finn et al., 2017), to the best our knowledge, we are the first to explicitly report this phenomenon.

4.1 OUR OBSERVATION: GBML FAVORS LINEAR LOSS SURFACE

To measure linearity, we have to firstly decide how linearity can be defined and which metric to use. A model is linear in the inner loop if and only if the gradient does not change along the trajectory in the inner loop. *i.e.*, $\nabla_\theta f(x|\theta^k)$ is constant in $k \in [0, S)$, where θ^k is the parameters at iteration $(k+1)$ updated by the SGD rule and S is the number of iterations in the inner loop. If we write a gradient in the phasor form, it means that the orientation (phase) and the magnitude of the gradient (amplitude) are constant. Therefore, we define the following two metrics to measure linearity:

Phase Consistency is defined as

$$\text{PC} = \frac{\sum_{k=0}^{S-1} \|\theta^{k+1} - \theta^k\|}{\|\theta^S - \theta^0\|}. \tag{3}$$

The metric is a comparison of the total variation of the inner loop trajectory and the linear distance between the meta-initialization parameter θ^0 and the final task-specific parameter θ^S. PC is always greater than or equal to 1 and if a gradient is phase-consistent throughout the trajectory, it will be 1.

Amplitude Consistency is defined as:

$$\text{AC} = \frac{1}{S} \sum_{k=1}^{S} \frac{\|\nabla_\theta f(x|\theta^k)\|}{\|\nabla_\theta f(x|\theta^0)\|}. \tag{4}$$

The metric is to compare how the gradient norm at the inner loop differs from the gradient norm at the meta-initialization point. If the amplitude is consistent, the corresponding value will be 1.

We can consider the model as linear if both PC and AC go to 1. The black line of Fig. 2 shows that in the inner loop of GBML, both AC and PC approach 1, which means that the model behaves like a linear model while solving each task τ in the inner loop. Note that we measure linearity *within an inner loop trajectory*, which means the gradient does not change much from the beginning to the end of the inner loop in GBML. This characteristic is unique and cannot be understood at once considering that meta-initialization itself has no meaning in the beginning of an inner loop dealing with an unseen task because it has no difference from randomly initialized parameters from the performance perspective. But once a GBML model starts adapting, the outer loop moves the parameters to a locally linear region and the inner loop learns the unseen task within a few steps using a large learning rate.

Table 1: Comparison of test accuracies (%) of the 4-conv network between the random initialization and the linearity-enforced initialization (LIL) without task-adaptation, *i.e.*, performance of θ^0. Note that LIL does not receive any label information from the outer loop. The numbers in parentheses are the numbers of shots.

meta-train	miniImageNet			Cars		
meta-test	miniImageNet	tieredImageNet	Cars	Cars	CUB	miniImageNet
Random-init (1)	21.36 ± 0.01	21.20 ± 0.01	21.10 ± 0.93	21.20 ± 0.93	21.62 ± 0.02	21.36 ± 0.01
LIL-init (1)	**24.62** ± 0.04	**23.80** ± 0.06	**24.63** ± 0.04	**25.84** ± 0.06	**25.86** ± 0.03	**26.48** ± 0.04
Random-init (5)	21.29 ± 0.09	21.72 ± 0.19	21.58 ± 0.37	21.58 ± 0.04	22.31 ± 0.06	21.28 ± 0.09
LIL-init (5)	**27.32** ± 0.59	**31.28** ± 1.68	**29.10** ± 1.81	**28.28** ± 0.79	**30.34** ± 2.69	**26.61** ± 1.05

4.2 Benefit of linearity in the inner loop

If so, why does this kind of 'strange' phenomenon happens? In this paper, we hypothesize that this phenomenon is actually a key to making GBML work. First, we explain this phenomenon by interpreting the higher-order terms such as Hessian as noise, then we give an intuitive experimental result that shows the benefit of linearity in the inner loop.

Hessian is 'noise' in SGD. Consider the loss L and the learning parameters θ^k for the k-th iteration. In the gradient descent (GD) setting, L decreases if and only if

$$\int_0^1 \nabla L(\theta(t)) \cdot \nabla L(\theta^k) dt \approx \int_0^1 \|\nabla L(\theta^k)\|_2^2 - \alpha t \nabla L(\theta^k)^T H(\theta^k) \nabla L(\theta^k) dt > 0. \tag{5}$$

where α is the learning rate and H is the Hessian. Note that SGD does not take Hessian into account and H can be viewed as a noise term. Normal deep learning fields do not suffer from this noise term because they exploit an extremely small learning rate. However, GBML uses a large learning rate in the inner loop, so it is very vulnerable to this noise term. Thus, in order to resolve this problem, the Hessian along the trajectory has to be reduced. A proof of (5) is in Appendix B

Linearity helps GBML work. Table 1 gives an intuitive experimental result showing that linearity is helpful for GBML. Here, we have compared the randomly initialized model with our linearly initialized model. The latter is trained to make the inner loop more linear, the details of which will be described in Sec. 5. Note that it does not receive any information about the label in the outer loop. We can see that our meta-initialization method that enforces linearity in the inner loop (LIL) performs significantly better than a random initialization. For a randomly initialized meta-initialization point, there is actually no effect of learning in the inner loop since its performance is not better than a random selection with 20% accuracy. This is exactly what was discussed above: because the effect of the high-order terms is large at initiation, the 'noise' term becomes dominant, hindering the loss from decreasing. This implies that we can improve the performance of GBML simply by enforcing Hessian to zero, which can be obtained by enforcing linearity in the inner loop.

4.3 Consequences of linearity in the inner loop

Connection to Kernel SGD Suppose we are to solve the general problem with the feature mapping $\varphi : \mathcal{X} \to \mathcal{H}$. *i.e.*, transform the given datapoint x to $\varphi(x)$ with a feature mapping φ. Then we can think of the linear model in $\mathcal{H}$ with loss l. When we apply SGD to this linear model in the feature space, we call this 'kernel SGD' as it can be computed with a kernel trick if $\mathcal{H}$ is an RKHS (reproducing kernel Hilbert space) (Hofmann et al., 2008) and the parameter update rule is as

$$\theta^{k+1} = \theta^k - \alpha_{k+1} \nabla_{\theta^k} l(h_\theta(\varphi(x)); y). \tag{6}$$

Note that h_θ is a linear model in $\mathcal{H}$ parameterized by θ. Normally, θ^0 is defined as 0 (Hofmann et al., 2008). Also, with respect to the hypothesis of linearity in the inner loop discussed above, we can treat f introduced in Sec. 3 as linear in θ in the inner loop. More specifically, there exists an equivalent feature map $\varphi_f : \mathcal{X} \to \mathcal{H}$ which satisfies $f(\cdot|\theta) = \langle \theta, \varphi_f(\cdot) \rangle$ for every $x \in \mathcal{X}$. Regarding

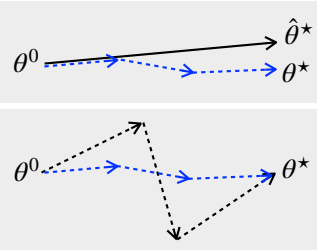

Figure 3: Illustration of losses for LIL (Sec. 5). θ^0 and $\theta^\star$ are the parameters initialized (meta-initialization) and optimized in the inner loop by (7), respectively. **Top: Phase Loss** reduces $KL(s(f(x|\theta^\star)), s(f(x|\hat{\theta}^\star)))$ where s is the softmax function. It draws $\theta^\star$ towards $\hat{\theta}^\star$, thus enforces linearity. **Bottom: Pseudo-Amplitude Loss** reduces the total length of the trajectory in the inner loop favoring the blue trajectory over the black.

meta-initialization parameter θ_0 is not related to the given task, due to its characteristics of meta-learning and its limited number of update steps, we can think of the inner loop as a finite sum of kernel SGD with $\theta^0 = 0$. More details are discussed on Appendix F.

Then GBML's learning schemes can be interpreted as 1) test current kernel's performance by adapting to the new task with kernel SGD in the inner loop, then 2) update the parameters to get a better kernel in the outer loop. From this perspective, we can connect gradient-based meta learning and metric-based meta learning since they share the same goal, *learning a good kernel to adapt to unseen tasks*. Furthermore, we can more directly relate GBML to MBML by interpreting the inner loop as constructing abstract prototype vectors, which is one of the most popular metric-based method for few-shot learning (Snell et al., 2017). We discuss the latter perspective in Appendix D.

Our hypothesis that linearity in the inner loop is a key to the success of GBML gives a new perspective to analyze existing algorithms. For example, there exists a paradox between ANIL (Raghu et al., 2019) and BOIL (Oh et al., 2020): the former argues feature reuse is the key to GBML while the latter says feature adaptation is essential. Both arguments are persuasive considering their good performances. With our linearity hypothesis, we can interpret their good performances originate from linearity. Because both BOIL and ANIL restrict the number of layers which can be updated in the inner loop, it makes the function simpler, thus enhancing linearity. More details are discussed in Appendix A.

Connection to 'Conventional' Deep learning. Also, when we assume linearity in the inner loop, we can analyze its dynamics in continuous-time domain since it meets the condition of gradient flow, which models neural network's learning process as an ODE $\dot{x}(t) = \nabla_x f(x, t)$. This is because gradient flow is the limit case of SGD ($\Delta\theta = \alpha\nabla f(x, \theta)$) where learning rate goes to zero. Since this ODE enables to analyze evolution of a neural network in the continuous domain, the condition of infinitesimal learning rate which makes gradient flow work is widely assumed in many theoretical analyses of deep learning, such as NTK (Jacot et al., 2018) and Mean-Field theory (Chizat & Bach, 2018). However, GBML exploits a large learning rate. So we cannot approximate GBML's setup with gradient flow normally. However, since we can treat the model as almost linear in the inner loop, the dynamics becomes an exact ODE of gradient flow. So we can exploit rich result of existing theorem on deep-learning-related gradient flow.

5 Enforcing Linearity in the Inner Loop (LIL)

Our analysis starts from the observation that GBML results in linearity of the inner loop. From this observation, we boldly assumed that 'linearity is a key factor to enable GBML to learn'. However, this is an imperfect and strong assumption in the sense that correlation does not imply causation. Perhaps there could have been other key factors that have resulted in linearity. How do we verify that linearity is a key factor in GBML? We need additional evidences. In this section, we are to introduce our algorithm: Linearity in the inner loop (LIL) which is based on our observation. It not only adds an evidence of our hypothesis but also generally enhances the performances without requiring any architectural modifications to the current GBML model.

Our hypothesis is that linearity is *essential* in the inner loop and if we can enforce a GBML model to be 'more' linear, it will better-adapt to new tasks *i.e.*, converge faster showing better performances.

As we discussed in Sec. 4.1, we can obtain linearity in the loop by fixing both the phase (orientation; $\frac{v}{\|v\|}$), and the amplitude (norm; $\|v\|$) of the gradient in the inner loop. In this context, we propose two regularization terms to enforce linearity. The first is the regularization term that directly restricts

Table 2: Test accuracy % of 4-conv network on benchmark data sets. The values in parentheses are the number of shots. The better accuracy between the baseline and LIL is bold-faced.

Domain	General(Coarse-grained)		Specific (Fine-grained)	
Dataset	miniImageNet	tieredImageNet	Cars	Cub
MAML (1)	47.88 ± 0.55	46.93 ± 1.07	47.78 ± 0.99	57.04 ± 1.42
MAML + LIL (1)	**49.07** ± 0.14	**49.29** ± 0.24	**49.53** ± 0.89	**58.06** ± 0.72
FOMAML (1)	46.29 ± 0.94	46.96 ± 0.81	47.91 ± 0.56	57.46 ± 0.21
FOMAML + LIL (1)	**46.81** ± 0.87	**47.86** ± 0.81	**48.45** ± 0.56	**58.01** ± 1.00
ANIL (1)	48.53 ± 0.76	50.09 ± 0.27	49.49 ± 0.71	**58.54** ± 3.76
ANIL + LIL (1)	**49.45** ± 0.31	**50.33** ± 0.28	**50.62** ± 1.06	57.87 ± 0.13
BOIL (1)	**50.44** ± 1.00	**50.56** ± 0.54	56.30 ± 0.76	**63.14** ± 0.38
BOIL + LIL (1)	49.91 ± 0.34	50.48 ± 0.43	**56.51** ± 0.85	62.93 ± 0.03
MAML (5)	64.81 ± 1.63	66.12 ± 1.10	62.24 ± 2.01	72.48 ± 0.86
MAML + LIL (5)	**66.34** ± 0.35	**67.85** ± 0.67	**69.94** ± 0.24	**74.37** ± 0.49
FOMAML (5)	62.91 ± 1.85	65.56 ± 0.89	62.75 ± 1.55	71.44 ± 0.34
FOMAML + LIL (5)	**65.05** ± 1.25	**66.37** ± 0.27	**65.69** ± 3.52	**72.52** ± 0.22
ANIL (5)	63.74 ± 1.39	66.12 ± 1.11	63.03 ± 0.95	71.75 ± 0.75
ANIL + LIL (5)	**66.55** ± 2.09	**66.75** ± 0.72	**63.74** ± 1.11	**73.68** ± 0.63
BOIL (5)	66.34 ± 0.35	69.50 ± 0.57	75.54 ± 0.22	**77.60** ± 0.12
BOIL + LIL (5)	**67.09** ± 0.59	**69.81** ± 0.17	**76.39** ± 0.02	77.19 ± 0.21

our gradient's phase fixed. And the second one behaves as a proximal term to help the dynamics, implicitly inducing linearity. Note that both losses are applied in the outer loop after adaptation of each inner loop.

Phase Loss What we want is to put the linearity assumption as a prior, and to implement it, the following term is added to the loss:

$$KL(s(f(x|\theta^S)), s(f(x|\hat{\theta}))) \text{ where } \hat{\theta} = \theta^0 - \delta\nabla_\theta f(x|\theta^0), \text{ and } x \in \mathcal{D}_{target}, \tag{7}$$

where KL is the Kullback-Leibler distance, $s(\cdot)$ is the softmax function, S is the number of optimization steps in the inner loop and δ is a scalar such that:

$$\sum_{k=0}^{S-1}\|\theta^{k+1} - \theta^k\| = \delta\|\nabla_\theta f(x|\theta^0)\|, \;\; \textit{i.e.}, \;\; \delta = \frac{\sum_{k=0}^{S-1}\|\theta^{k+1} - \theta^k\|}{\|\nabla_\theta f(x|\theta^0)\|}. \tag{8}$$

This is due to the following assumption: if the gradient's phase is constant, it will be sufficient to estimate the parameter after S steps by only the gradient measured at the initialization of the inner loop. To this end, the orientation of the first gradient is multiplied by the length of the trajectory while going by S steps. The role of this term is illustrated at the top of Fig. 3.

Pseudo-Amplitude Loss To further enhance stability of the learning, we add the following term:

$$\sum_{k=0}^{S-1}\|\theta^{k+1} - \theta^k\|. \tag{9}$$

There are two main reasons for including the loss. First, since this term reduces step size directly, it can be viewed as a simple proximal term, so we can expect improvement of dynamics such as in Rajeswaran et al. (2019). Second, it implicitly reduces the variance of gradients. *i.e.*, Amplitude. If you look at the bottom of Fig. 3, you can see a clear example. For the loss that reduces the distance between the existing meta-initialization parameter and the task-specific parameter, both the black case and the blue case show the same loss value. On the other hand, in the case of our trajectory loss considering the entire trajectory, the loss of the blue one is lower. Therefore, it can be seen that it implicitly reduces the variance of the gradient, *i.e.*, enforces AC to 1. Note that although there is no explicit requirements for amplitude, Fig. 2 shows that the amplitude condition is naturally satisfied by adjusting the phase loss and the pseudo-amplitude loss.

5.1 Characteristics of LIL

Since LIL ultimately aims to linearize the learning dynamics in the inner loop, it is natural to compare our algorithm with other existing linearization algorithms. Our algorithm differs from previous studies in two main aspects. First, LIL has much more relaxed conditions compared to the existing linearization algorithms. In SAM optimizer, which seeks to find flat minima (Foret et al., 2020), the

Table 3: Test accuracy % of 4-conv network on cross-domain adaptation. The values in parentheses are the number of shots. The better accuracy between the baseline and LIL is bold-faced.

adaptation	General to General		General to Specific		Specific to General		Specific to Specific	
meta-train	tieredImageNet	miniImageNet	miniImageNet	miniImageNet	Cars	Cars	CUB	Cars
meta-test	miniImageNet	tieredImageNet	Cars	CUB	miniImageNet	tieredImageNet	Cars	CUB
MAML (1)	47.52 ± 1.66	51.84 ± 0.24	34.41 ± 0.47	**40.91** ± 0.57	28.67 ± 1.17	30.79 ± 1.17	32.74 ± 1.12	30.95 ± 1.41
MAML + LIL (1)	**50.09** ± 0.39	**53.03** ± 0.14	**34.54** ± 0.31	40.48 ± 0.40	**28.73** ± 0.62	**30.87** ± 0.38	**32.98** ± 0.60	**31.11** ± 0.70
FOMAML (1)	47.30 ± 0.95	49.62 ± 1.54	34.35 ± 0.30	40.86 ± 0.83	28.49 ± 0.76	31.06 ± 0.22	**33.57** ± 0.84	**31.52** ± 0.34
FOMAML + LIL (1)	**48.81** ± 0.76	**50.73** ± 1.15	**34.73** ± 0.12	**41.48** ± 0.37	**29.47** ± 0.27	**31.26** ± 0.22	33.53 ± 0.40	30.98 ± 1.19
ANIL (1)	51.04 ± 0.11	52.76 ± 0.28	**34.10** ± 0.50	**40.37** ± 1.23	28.26 ± 0.65	29.86 ± 0.74	32.87 ± 2.70	30.68 ± 1.86
ANIL + LIL (1)	**51.19** ± 0.62	**53.08** ± 0.45	34.07 ± 0.20	40.08 ± 0.40	**29.73** ± 0.73	**30.62** ± 1.04	**32.48** ± 0.18	**30.48** ± 1.20
BOIL(1)	51.50 ± 0.81	53.99 ± 0.20	36.78 ± 0.37	44.36 ± 0.38	33.94 ± 0.75	**33.82** ± 0.38	35.46 ± 0.21	34.37 ± 0.37
BOIL + LIL (1)	**52.05** ± 0.10	**54.13** ± 0.34	**36.78** ± 0.44	**44.56** ± 0.77	**34.40** ± 0.25	33.79 ± 0.50	**35.60** ± 0.33	**34.53** ± 0.23
MAML (5)	66.86 ± 1.58	67.96 ± 1.22	46.57 ± 0.53	56.32 ± 1.17	37.23 ± 1.95	41.00 ± 1.86	44.02 ± 2.29	41.84 ± 1.25
MAML + LIL (5)	**69.25** ± 0.50	**69.47** ± 0.35	**48.86** ± 0.71	**58.58** ± 1.08	**43.09** ± 1.24	**46.01** ± 1.41	**44.26** ± 1.44	**46.06** ± 2.10
FOMAML (5)	66.04 ± 0.89	66.39 ± 1.05	45.48 ± 2.78	57.03 ±0.84	37.48 ± 1.36	41.11 ± 1.93	43.32 ± 2.62	40.53 ± 3.58
FOMAML + LIL (5)	**66.31** ± 0.27	**67.96** ± 0.87	**47.80** ± 0.30	**57.10** ± 0.44	**39.59** ± 0.75	**43.28** ± 0.59	**44.80** ± 0.22	**43.50** ± 1.65
ANIL (5)	67.08 ± 0.19	67.08 ± 1.38	42.38 ± 1.57	**56.60** ± 2.60	38.09 ± 1.50	41.29 ± 3.04	41.27 ± 1.41	42.00 ± 1.28
ANIL + LIL (5)	**68.05** ± 2.32	**67.23** ± 1.69	**45.38** ± 2.49	56.31 ± 3.98	**38.47** ± 1.32	**41.39** ± 0.55	**43.40** ± 0.52	**42.52** ± 2.29
BOIL (5)	70.12 ± 0.67	**70.00** ± 0.12	**53.57** ± 1.46	**62.15** ± 0.52	45.40 ± 0.25	45.69 ± 0.10	**48.52** ± 0.19	**48.45** ± 0.19
BOIL + LIL (5)	**70.83** ± 0.10	69.83 ± 0.33	53.45 ± 0.20	62.03 ± 1.58	**45.68** ± 0.34	**45.80** ± 0.43	48.35 ± 0.77	48.35 ± 0.92

condition that the model is linear within some ϵ-Ball, $B(x, \epsilon)$, is a much stronger condition than ours as LIL only considers linearity along the trajectory of the inner loop. Since it is a much more relaxed condition than the existing one, the probability of the existence of the solution we want in the function set (neural nets) is relatively greater. The second difference is that LIL considers the whole trajectory, not the end point only. Unlike other methodologies (Foret et al., 2020; Qin et al., 2019), our GBML framework evaluates and updates the previous loop at the end of each inner loop, allowing us to project the entire trajectory into the functional space we want. Existing methodologies just cannot do this because they only consider the current state.

6 EXPERIMENTS

Experiment setup To show the effectiveness of our algorithm prioritizing linearity in the inner loop, we conducted experiments on various benchmark sets for two general datasets, miniImageNet (Vinyals et al., 2016) and tieredImageNet (Ren et al., 2018), and two specific datasets, Cars (Krause et al., 2013) and CUB (Welinder et al., 2010) using two backbone networks, 4-Conv network with 64 channels from Vinyals et al. (2016) and ResNet-12 consisting of four blocks of which the first block starts with 64 channels from Oreshkin et al. (2018). Following Oh et al. (2020), we trained 4-Conv network for 30k iterations and ResNet-12 for 10k iterations. We experimented on 5-way 5-shot and 5-way 1-shot settings with 3 optimization steps in each inner loop and constructed one batch with 4 tasks. We reproduced all the results for 3 times and reported the mean and standard deviation.

Table 4: Test accuracy % of ResNet-12. The values in parentheses are the number of shots. The better accuracy between the baseline and LIL is bold-faced.

meta-train	miniImageNet		
meta-test	miniImageNet	tieredImageNet	Cars
MAML (1)	50.63 ± 1.70	54.33 ± 1.02	35.41 ± 1.01
MAML + LIL (1)	**52.72** ± 0.15	**54.66** ± 0.58	**35.75** ± 0.29
MAML (5)	71.32 ± 1.55	74.01 ± 1.69	45.75 ± 0.89
MAML + LIL (1)	**72.50** ± 0.94	**75.13** ± 1.35	**48.21** ± 0.49

Baseline Since our method can be applied on top of any algorithm utilizing nested loops, we applied our losses to 4 algorithms: MAML (Finn et al., 2017) and its variations ANIL (Raghu et al., 2019) and BOIL (Oh et al., 2020), and the Hessian-free algorithm, FOMAML (Finn et al., 2017). The reason for choosing these models is as follows: MAML is the most general GBML algorithm which has nested loops. So we can expect that if LIL works for MAML, it will also work for other variants with high probability. ANIL and BOIL were selected as they have overlapping characteristics to our loss, LIL, in that they implicitly enforce linearity because by freezing the body (encoder) and the head (classifier) respectively in the inner loop, they can be thought to have a implicit linearity prior since a model can optimize only limited layers in the inner loop. See Appendix A for details. Finally, we selected FOMAML because we wanted to show LIL also works for Hessian-free algorithm. For FOMAML, unlike other baselines, we used LIL as a pre-training scheme inspired by the result of Table 1. This is because FOMAML is Hessian-free while LIL is not. When LIL loss is used only, it has the possibility of collapsing into a trivial solution. So we used the self-supervised learning scheme (SSL) (Grill et al., 2020) to obtain a good meta-initialization point. The full details are discussed in the Appendix E.

Table 5: Ablation of our Loss term. PC denotes Phase Loss in LIL, and AC denotes Pseudo-Amplitude Loss in LIL. The values in parentheses are the number of shots.

Meta-train	miniImageNet			Cars		
Meta-test	miniImageNet	tieredImageNet	Cars	Cars	miniImageNet	CUB
MAML (1)	47.88 ± 0.55	51.84 ± 0.24	34.19 ± 0.66	47.77 ± 0.99	28.66 ± 1.17	30.95 ± 1.41
MAML + PC (1)	47.13 ± 0.30	50.95 ± 0.65	34.55 ± 0.82	49.04 ± 1.54	28.76 ± 0.17	31.02 ± 0.16
MAML + AC (1)	48.62 ± 0.29	52.53 ± 0.47	34.34 ± 0.35	49.53 ± 0.07	28.83 ± 0.12	30.82 ± 0.49
MAML + LIL (1)	49.07 ± 0.14	53.03 ± 0.14	34.54 ± 0.31	49.53 ± 0.89	28.73 ± 0.62	31.11 ± 0.70
MAML (5)	64.81 ± 1.63	67.96 ± 1.22	46.67 ± 0.53	62.24 ± 2.01	37.23 ± 1.95	41.84 ± 1.25
MAML + PC (5)	65.48 ± 0.30	68.63 ± 0.34	48.09 ± 0.82	65.07 ± 0.85	39.37 ± 0.88	42.79 ± 0.85
MAML + AC (5)	65.98 ± 0.50	69.01 ± 0.06	46.37 ± 1.15	67.27 ± 0.92	37.66 ± 2.29	39.01 ± 2.43
MAML + LIL (5)	66.34 ± 0.35	69.47 ± 0.35	48.86 ± 0.71	69.94 ± 0.24	43.09 ± 1.23	46.06 ± 2.10

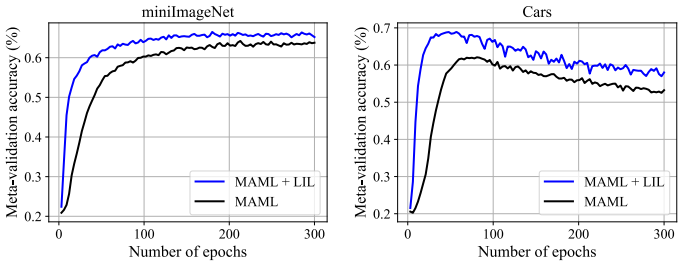

Figure 4: 5-way 5-shot meta-validation performance as a function of epoch. **Left:** validation accuracy trained on mini-Imagenet dataset. **Right:** validation accuracy trained on Cars dataset.

Hyperparameter setting We have fine-tuned the learning rate in the inner loop. Note that it performs better than the original setting in Finn et al. (2017). We multiplied 0.1 for the Phase loss in all the experiments. Pseudo-Amplitude loss was multiplied by 0.01 and 0.1 for 1-shot and 5-shot respectively for MAML baseline, and we did not apply Pseudo-Amplitude loss for other baselines. For reproducing ANIL and BOIL, we just restricted the parameter space to the head or the body in the inner loop with other settings being untouched. More details can be found in Appendix C.

Performance of LIL Table 2 and 3 show the results of LIL on various baselines, and Table 4 shows the result of LIL on other architecture. It can be seen that performance gain is larger in MAML and FOMAML compared to ANIL and BOIL. This is because ANIL and BOIL already enforced linearity itself by freezing layers, thereby the effect adding explicit regularization term reduced. We can also confirm that LIL also works on different architectures. In most cases, the LIL-augmented baseline performs better than the vanilla.

Dynamics of LIL Fig. 2 and 4 imply that linearity in the inner loop is the essential prior of GBML. In Fig. 4, LIL shows much faster convergence compared to the baseline method (MAML). And in Fig. 2, LIL satisfies linearity condition much faster than the baseline although both meets the condition at the end. This leads to the interpretation that LIL guides GBML to move the parameters faster to a well-conditioned region with adequate prior. Table 5 studies the effectiveness of each loss term. We can verify that both terms are effective by itself.

7 Conclusion

In this paper, we observed that GBML works as a linear model in the inner loop. With that observation, we hypothesized that linearity is an essential prior for the success of GBML. With the linearity hypothesis in the inner loop, we could connect gradient-based meta learning to metric-based meta learning by showing the equivalence between SGD in the inner loop and Kernel SGD. To prove and exploit our hypothesis, we proposed a regularization-based algorithm enforcing Linearity in the Inner Loop (LIL) which encourages the model to behave more like a linear model in the inner loop. To prove the potential of LIL, we applied LIL to various baseline GBML algorithms including a Hessian-free baseline and tested performances on various datasets. Both quantitative and qualitative analyses were conducted showing that LIL converges faster with better performance by meeting the linearity condition faster.

REFERENCES

Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. Advances in Neural Information Processing Systems, 33:20755–20765, 2020.

Alberto Bernacchia. Meta-learning with negative learning rates. arXiv preprint arXiv:2102.00940, 2021.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. Advances in neural information processing systems, 6, 1993.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pp. 1597–1607. PMLR, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15750–15758, 2021.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. Advances in neural information processing systems, 31, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. CoRR, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In Silvia Chiappa and Roberto Calandra (eds.), Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, pp. 1082–1092. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/fallah20a.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In International conference on machine learning, pp. 1126–1135. PMLR, 2017.

Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. arXiv preprint arXiv:1909.00025, 2019.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. CoRR, abs/2010.01412, 2020. URL https://arxiv.org/abs/2010.01412.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems, 33:21271–21284, 2020.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks, 2018. URL https://arxiv.org/abs/1812.01187.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. The annals of statistics, 36(3):1171–1220, 2008.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. Advances in neural information processing systems, 31, 2018.

George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. Journal of mathematical analysis and applications, 33(1):82–95, 1971.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In Proceedings of the IEEE international conference on computer vision workshops, 2013.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436–444, 2015.

Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. In European Conference on Computer Vision, pp. 741–756. Springer, 2020.

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

Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. Boil: Towards representation change for few-shot learning. arXiv preprint arXiv:2008.08882, 2020.

Boris N. Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. CoRR, abs/1805.10123, 2018. URL http://arxiv.org/abs/1805.10123.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. Advances in Neural Information Processing Systems, 32, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. CoRR, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. arXiv preprint arXiv:1909.09157, 2019.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. Advances in neural information processing systems, 32, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In International Conference on Learning Representations, 2017. URL https://openreview.net/forum?id=rJY0-Kcll.

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

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. CoRR, abs/1807.05960, 2018a. URL http://arxiv.org/abs/1807.05960.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. arXiv preprint arXiv:1807.05960, 2018b.

Jürgen Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook. PhD thesis, Technische Universität München, 1987.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. Advances in neural information processing systems, 30, 2017.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1199–1208, 2018.

Sebastian Thrun and Lorien Pratt. Learning to learn. Springer Science & Business Media, 2012.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. Matching networks for one shot learning. Advances in neural information processing systems, 29, 2016.

Haoxiang Wang, Ruoyu Sun, and Bo Li. Global convergence and induced kernels of gradient-based meta-learning with neural nets. CoRR, abs/2006.14606, 2020. URL https://arxiv.org/abs/2006.14606.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

A LINEARITY OF ANIL (RAGHU ET AL., 2019) AND BOIL (OH ET AL., 2020)

In this part, we analyze ANIL (Raghu et al., 2019) and BOIL (Oh et al., 2020) from the perspective of linearity. As Raghu et al. (2019) discussed, GBML tends to exploit the head part most at the inner loop. As empirically proved in Figure 14 of Oh et al. (2020), we can identify that the size of gradient norm is predominant in head layers. From this observation, Raghu et al. (2019) proposed an Almost No Inner Loop (ANIL) algorithm which only exploits head layers in the inner loop. Since ANIL could get a good performance only by optimizing the head by reusing the encoder, they argued that representation reuse is the key of GBML.

On the other hand, unlike Raghu et al. (2019), Oh et al. (2020) argued that feature adaptation is more essential to GBML and proposed an algorithm called Body Only update in the Inner Loop (BOIL). This algorithm only freezes head, thereby it updates only the encoder (body) in the inner loop. This means that it freezes the decision boundary for all tasks. Since both ANIL and BOIL showed non-negligible performance improvements over the baseline, both arguments – feature reuse vs. feature adaptation – look persuasive despite they argue exactly in the opposite ways. However, our perspective of linearity can incorporate both arguments.

For ANIL, we can explain its success from the perspective of parameter restriction which enforces linearity. It reduced the number of non-linear components which lie in the activation function between layers. Hence, we can think it as enforcing linearity in the inner loop. As a result, this algorithm is much more powerful at 1-step optimization. For example, ANIL scored better than MAML in 1-step optimization in Oh et al. (2020). Also, ANIL algorithm can be thought as a variant of the preconditioning method (Flennerhag et al., 2019) since the encoder is shared across all tasks and only the head is adjusted task-specifically.

Also, we can explain BOIL's success originates from the fact that it enforces linearity in 'good' layers. Although our argument that *linearity is the key to GBML* seems unconvincing, since BOIL exploits far more non-linearity compared to ANIL, in Figure 14 of Oh et al. (2020), we can see that gradient norm is predominant only in the penultimate layer. So their algorithm can be interpreted as a variant of ANIL, which updates the penultimate layer only. We can empirically check this argument from Table 16 of the same paper. We can see that they actually gain a boosted performance when they freeze all but one layer, which has much stronger linearity since it has no non-linearity inside.

B PROOF OF (5)

Due to fundamental theorem of calculus,

$$L(\theta^{k+1}) - L(\theta^k) = \int_C \nabla L(\theta) \cdot d\theta = \int_0^1 \nabla L(\theta(t)) \cdot v(t) dt, \tag{10}$$

where C is a trajectory whose start and end points are θ^k and θ^{k+1}. In GD setting, because $\theta^{k+1} = \theta^k - \alpha \nabla L(\theta^k)$ for some learning rate $\alpha > 0$, we can think of the straight line trajectory joining θ^k and θ^{k+1}. In this case, the velocity vector becomes $v(t) = -\alpha \nabla L(\theta^k)$ and

$$L(\theta^{k+1}) - L(\theta^k) = -\alpha \int_0^1 \nabla L(\theta(t)) \cdot \nabla L(\theta^k) dt \tag{11}$$

where $\theta(0) = \theta^k$, $\theta(1) = \theta^{k+1}$ and $\theta(t) = (1-t)\theta(0) + t\theta(1)$.

By Taylor series expansion it becomes

$$\nabla L(\theta(t)) \approx \nabla L(\theta^k) + H(\theta^k)(\theta(t) - \theta^k) = (I - \alpha t H(\theta^k)) \nabla L(\theta^k) \tag{12}$$

and combining Eq.(11) and Eq.(12) proves Eq.(5).

C IMPLEMENTATION DETAILS

Hyperparameter Details For the inner loop, we set the learning rate as 0.5 and 0.3 for 4-Conv network and ResNet-12, respectively. Also, we set the meta-learning rate, which is the learning rate

of outer loops as 0.001 and 0.0006 for 4-Conv network and 0.3 for ResNet-12. For fair comparison, we pretrained FOMAML with LIL loss for 15k iterations and then trained with FOMAML without LIL loss for the remaining 15k iterations.

Fine-Tuning of learning rate in the inner loop We have fine-tuned the learning rate in the inner loop from Finn et al. (2017). Our fine-tuned settings work better even though it has harsher conditions. For example, in miniImageNet, Finn et al. (2017) exploited 5 gradient steps in the inner loop for training, and 10 gradient steps for inference; it is contradictory to their report that 3 gradient steps are enough for the inner loop. They also used additional gradient steps at the test time which is advantageous to performance by degrading its inference time twice. While ours exploits only 3 gradient steps in both training and inference, we achieved higher performance compared to the original ones: Finn et al. (2017) reported 63.11%p and ours reported 64.81%p.

Usage of Pseudo-Amplitude loss For BOIL and ANIL, we only used the phase loss without pseudo-amplitude loss because both of them have more stable dynamics due to their innate linearity prior (See Appendix A) and pre-training scheme (See Appendix E) was used.

D GBML IS A VARIANT OF PROTOTYPE VECTOR METHOD

Suppose there exists a meta-learning model that satisfies the linearity assumption in the inner loop, then classifying a new classification task with a task-specific function $f(\cdot|\theta^\star)$ after an inner loop is equivalent to creating a prototype vector for each class on a specific feature map and classifying the input as the class of the most similar prototype vector.

The proof starts by defining the prototype vector at first.

Prototype Vector We define a prototype vector V_c for class c in an N-way K-shot classification task formally as

$$V_c = \sum_{i=1}^{N} \sum_{j=1}^{K} \beta_{ij} \varphi_c(X_{ij}), \quad c \in \{1, \cdots, N\}, \tag{13}$$

where X_{ij} is the j-th input sample for the i-th class, $\varphi_c(\cdot) \in \mathcal{H}$ is a class-specific feature map and β_{ij} indicates the importance of X_{ij} for constituting the prototype vector V_c.

In other words, there exists a feature map φ_c for each class c, and the support set is mapped to the corresponding feature map and then weighted-averaged to constitute the prototype vector of the corresponding class. At inference time, the classification of a given query X is done by taking the class of the most similar prototype vector as follows:

$$\hat{c} = \arg\max_c \langle V_c, \varphi(X) \rangle. \tag{14}$$

Here, $\varphi : \mathcal{X} \to H$ is a non-class-specific mapping. We can also rewrite the prototype vector using φ and by defining a projection $P_c : \mathcal{X} \to \mathcal{X}$ as

$$P_c(X) = \begin{cases} X & \text{if } y(X) = c, \\ \nu \in \mathcal{N}(\varphi), & \text{if } y(X) \neq c \end{cases} \tag{15}$$

where $y(X)$ is the ground truth class of X and $\mathcal{N}$ is the null space of φ *i.e.*, $\varphi(\nu) = 0$.

Then by defining $\varphi_c \triangleq \varphi \circ P_c$ and $\beta_{ij} \triangleq \frac{1}{K}$, it becomes

$$V_c = \frac{1}{K} \sum_{j=1}^{K} \varphi(X_{cj}). \tag{16}$$

SGD in the inner loop If GBML satisfies the hypothesis of linearity in the inner loop, f is locally linear in θ in an inner loop. More specifically, there exists an equivalent feature map $\varphi_c : \mathcal{X} \to \mathcal{H}$ which satisfies $f_c(\cdot|\theta_c) = \langle \theta_c, \varphi_c(\cdot) \rangle$ for every $x \in \mathcal{X}$ where $f(\cdot|\theta) = [f_1(\cdot|\theta_1), \cdots, f_N(\cdot|\theta_N)]^T$.

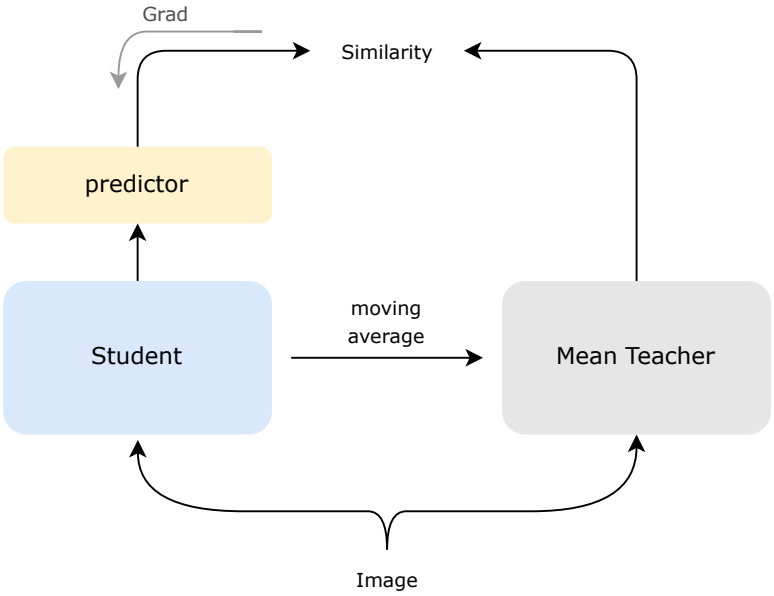

Figure 5: Overall view of pretraining scheme, which has same architecture to Grill et al. (2020) on abstract level to prevent collapse.

With the loss function $L(x, y|\theta) = D(s(f(x|\theta)), y)$ for some distance measure D such as cross entropy, we can formulate the inner loop of N-way K-shot meta learning by SGD as

$$\theta_c^{k+1} = \theta_c^k - \alpha \sum_{i=1}^{N} \sum_{j=1}^{K} \frac{\partial L(X_{ij}, y(X_{ij})|\theta)}{\partial \theta_c} = \theta_c^k - \alpha \sum_{i=1}^{N} \sum_{j=1}^{K} \frac{\partial D}{\partial f_c} \varphi_c(X_{ij}), \tag{17}$$

since all samples in the support set are inputted in a batch of an inner loop.

Because the model is linear in the inner loop, the batch gradient does not change. Let $\beta_{ij} = -\frac{\partial D}{\partial f_c}|_{\theta_c^0, X_{ij}}$. Then after t steps, by (13), the model becomes

$$\theta_c^t = \theta_c^0 + \alpha t \sum_{i=1}^{N} \sum_{j=1}^{K} \beta_{ij} \varphi_c(X_{ij}) = \theta_c^0 + \alpha t V_c. \tag{18}$$

At the initialization step of an inner loop, there is no information about the class, even the configuration order of the class, because the task is randomly sampled. If so, the problem is solved in the inner loop. For example, if a class *Dog* is allocated to a specific index such as *Class 3*. There is no guarantee that it will have the identical index the next time the class *Dog* comes in. Thus, at a meta-initialization point θ^0, the scores for different classes would not be much different, *i.e.*, $f_i(x|\theta_i^0) \simeq f_j(x|\theta_j^0)$ for $i, j \in [1, \cdots N]$.

Considering the goal of classification is achieved through relative values between $f_i(X)$'s, the value at the initialization point does not need to be considered significantly. Therefore

$$\arg\max_c f_c(X) = \arg\max_c \langle \theta_c^t, \varphi(X) \rangle = \arg\max_c \langle \theta_c^0 + \alpha t V_c, \varphi(X) \rangle \sim \arg\max_c \langle V_c, \varphi(X) \rangle \tag{19}$$

So inner loop in GBML can be interpreted as making proptotype vector with given support set. □

E PRETRAINING A GBML WITH LIL LOSS

When we see Figure 3, the phase loss reduces the distance between $f(x|\theta^\star))$ and $f(x|\hat{\theta}^\star)$ which can be viewed as different views of the same input x. So it can be interpreted as reducing the

mutual information between the two representations, which is a typical objective in conventional self-supervised learning (SSL) methods such as Chen et al. (2020); Grill et al. (2020); Chen & He (2021). These SSL methods can also be categorized as metric-based meta-learning methods because they learn a good encoder to produce a good prototype for each sample. Considering SSL methods are typically used for pretraining a model, we might expect the possibility of LIL as a pretraining scheme. Since our phase loss only reduces the distance between positive pair, it has the same problem as the SSL algorithms (Chen & He, 2021; Grill et al., 2020), *i.e.*, it can collapse to zero embedding. For example, if f falls into space where gradient is zero everywhere, $f(x|\theta^\star))$ and $f(x|\hat{\theta}^\star)$ will be exactly the same. This kind of problem occurs similarly to SSL which exploits only positive pair such as Grill et al. (2020); Chen & He (2021). To prevent this phenomenon, we adapted similar methodology from BYOL (Grill et al., 2020) as described below.

Fig. 5 shows the overall training scheme. For pretraining, we exploit only the phase loss which corresponds to reducing the mutual information. In the student network, we infer one-step approximation ($f(x|\hat{\theta}^\star)$) and for the teacher network, we infer with 3-optimization steps ($f(x|\theta^\star)$). For fair comparison, we did not use any augmentation both in inner and outer loops. Note that computation cost of LIL in pretaining is similar to other GBML methodologies which exploits 1-step optimization in the inner loop. This is because it only calculates the student's gradient. Which produces the one-step approximation $f(x|\hat{\theta}^\star)$. So, there is no need to compute multiplication between Hessians, *i.e.*, $\Pi_{i=0}^{k-1} \frac{df_{\theta^{i+1}}}{df_{\theta^i}}$.

We set the momentum parameter as 0.999 and multiplied 0.1 to the phase loss.

F KERNEL TRICK AND KERNEL SGD

This part is just for completeness of the paper and most materials are from Hofmann et al. (2008)

For shallow learning, it is often beneficial to map data with feature map $\phi : \mathcal{X} \to \mathbb{R}^d$. Then we can apply SGD in that feature space.

Hilbert space SGD. Let $\mathcal{H}$ be a Hilbert space and $\mathcal{X}$ be input space, $\mathcal{Y} \subset \mathbb{R}$, $\theta \in \mathcal{H}$ and $h_\theta(x) = \langle \theta, \phi(x) \rangle_{\mathcal{H}}$. Consider the following problem:

$$\min_{\theta \in \mathcal{H}} \mathbb{E}_{(X,Y)\sim\mathcal{H}}[l(h_\theta(X); Y)]. \tag{20}$$

We can solve this problem with SGD:

$$\theta^{k+1} = \theta^k - \beta_{k+1}\phi(X_{k+1}), \tag{21}$$

where $\theta^0 = 0$ and $\beta_{k+1} = \alpha_{k+1}\, l'(h_{\theta^k}(X_{k+1}); Y_{k+1})\phi(X_{k+1})$. Here, $l' \triangleq \frac{\partial l}{\partial h_\theta}$ and α_k is the learning rate.

Although it is feasible if $\dim\mathcal{H} < \infty$, we cannot solve this with one-pass SGD if $\mathcal{H}$ has infinite dimension and if $\mathcal{H}$ is an RKHS on $\mathcal{X}$ with RK K, This becomes

$$f^{k+1} = f^k - \beta_k K(X_{k+1}, \cdot) \tag{22}$$

for $f^0 = 0$. However, we normally access data multiple times during training. So one should think about the situation of random indices, $i(k)$, such that $X_{i(k)} \in \mathcal{X}$. Then SGD becomes

$$f^{k+1} = f^k - \beta_k K(X_{i(k+1)}, \cdot). \tag{23}$$

This type of SGD has been proven to be optimal by the Representor theorem (Kimeldorf & Wahba, 1971). and this problem usually focuses on how to find a 'good' RK that produces its RKHS that fits the data well.