

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,

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

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

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

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

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

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 $f(\cdot|\theta^*)$ 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

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

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:

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

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

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

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

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

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 $\varphi_c : \mathcal{X} \rightarrow \mathcal{H}$ which satisfies $f_c(\cdot|\theta_c) = \langle \theta_c, \varphi_c(\cdot) \rangle$ for every $x \in \mathcal{X}$ where $f(\cdot|\theta) = [f_1(\cdot|\theta_1), \dots, f_N(\cdot|\theta_N)]^T$.

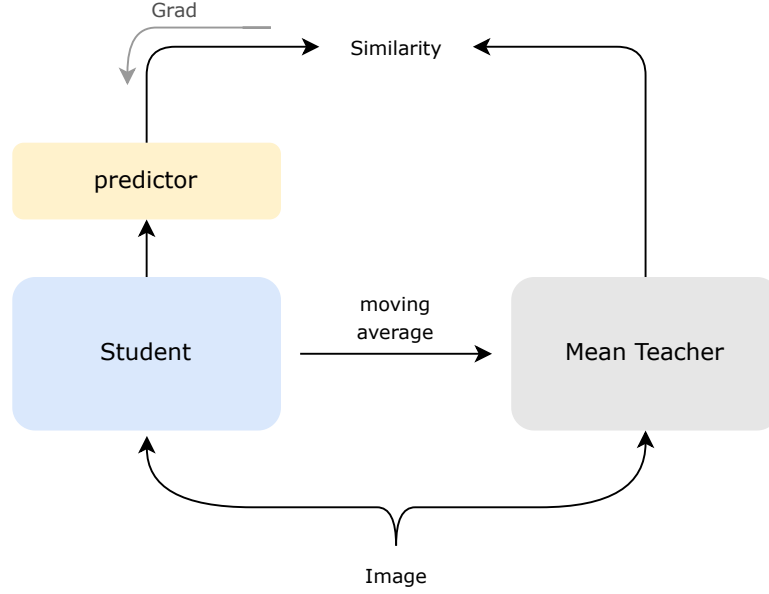


Figure 5: Overall view of pretraining scheme, which has same architecture to Grill et al. (2020) on abstract level to prevent collapse.

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

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

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

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

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 $i, j \in [1, \dots, N]$.

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

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

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

E PRETRAINING A GBML WITH LIL LOSS

When we see Figure 3, the phase loss reduces the distance between $f(x|\theta^*)$ and $f(x|\hat{\theta}^*)$ 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, $f(x|\theta^*)$ and $f(x|\hat{\theta}^*)$ 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 ($f(x|\hat{\theta}^*)$) and for the teacher network, we infer with 3-optimization steps ($f(x|\theta^*)$). For fair comparison, we did not use any augmentation both in inner and outer loops. Note that computation cost of LIL in pretraining 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 $f(x|\hat{\theta}^*)$. So, there is no need to compute multiplication between Hessians, *i.e.*, $\prod_{i=0}^{k-1} \frac{df_{\theta^{i+1}}}{df_{\theta^i}}$.

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 $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$. 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:

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

We can solve this problem with SGD:

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

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

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

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

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

This type of SGD has been proven to be optimal by the Representer 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.