Meta-learning the Learning Trends Shared Across Tasks

Jathushan Rajasegaran[1]
brjathu@gmail.com
Salman Khan[2,3]
salman.khan@mbzuai.ac.ae
Munawar Hayat[4]
munawar.hayat@monash.edu
Fahad Shahbaz Khan[2,5]
fahad.khan@mbzuai.ac.ae
Mubarak Shah[6]
mshah@ucf.edu

[1]University of California, Berkeley, USA
[2]Mohamed Bin Zayed University of AI, UAE
[3]Australian National University, AU
[4]Monash Univeristy, AU
[5]CVL, Linköping University, Sweden
[6]University of Central Florida, USA

Abstract

Meta-learning stands for 'learning to learn' such that generalization to new tasks is achieved. Among these methods, Gradient-based meta-learning algorithms are a specific sub-class that excel at quick adaptation to new tasks with limited data. This demonstrates their ability to acquire transferable knowledge, a capability that is central to human learning. However, the existing meta-learning approaches only depend on the current task information during the adaptation, and do not share the meta-knowledge of how a similar task has been adapted before. To address this gap, we propose a *'Path-aware'* model-agnostic meta-learning approach. Specifically, our approach not only learns a good initialization (meta-parameters) for adaptation, it also learns an optimal way to adapt these parameters to a set of task-specific parameters, with learnable update directions, learning rates and, most importantly, the way updates evolve over different time-steps. Compared to the existing meta-learning methods, our approach offers the following benefits: (a) The ability to learn gradient-preconditioning at different time-steps of the inner-loop, thereby modeling the dynamic learning behavior shared across tasks, and (b) The capability of aggregating the learning context through the provision of direct gradient-skip connections from the old time-steps, thus avoiding overfitting and improving generalization. We report significant performance improvements on a number of datasets for few-shot learning on classification and regression tasks.

1 Introduction

Leveraging from their prior experience, humans can easily learn new concepts from a few observations. Few-shot learning aims to mimic this astounding capability, and requires quick model adaptation using only a few examples. In contrast, contemporary deep learning models are data hungry by nature, learn each task in isolation and require significant training

time. Meta-learning comes as a natural solution to this problem, due to its focus on '*learning to learn*' a generalizable model from multiple related tasks. This strategy offers a quick adaptation to new tasks.

Model-agnostic meta-learning (MAML) [5] is a popular approach that learns a generalizable representation which can be quickly adapted to a new task with only a few examples. In MAML, the meta-training operates in two nested loops. First, the task-specific parameters are learned in the inner-loop, followed by learning a shared set of parameters that acts as a good prior for all the tasks. Although MAML uses standard gradient descent for inner-loop optimization, recent works demonstrate that an improved inner-loop optimization can positively influence the performance [2, 13, 15]. However, these existing methods do not consider the learning trends (update direction, step-size and evolution through iterations) in the inner-loop across different tasks.

In this paper, we develop a new *model-agnostic* meta-learning framework called 'Path-Aware MEta-LeArning' (PAMELA). Compared to MAML, which learns a good initialization for the meta-learner, our approach learns the optimal learning trend shared across different tasks. This means that, in addition to a transferable initialization, we learn the update directions, the learning rates and, most importantly, the way updates evolve over different time-steps. As an example, our model can encode how the inner-loop training first commences with large steps and converges to shorter steps as it gets close to the local minima. Similarly, it can also learn how to optimally reuse task parameters from the past updates. In essence, our approach provides a new way to encode the prior (generalizable) knowledge in a more principled and flexible manner.

PAMELA showcases two main novelties to meta-learn the learning paths shared across tasks. *First,* we propose to learn distinct gradient preconditioning matrices at different iteration steps in the inner-loop. Pre-conditioning was first proposed in Meta-SGD [13]. However, as opposed to [13], which learns a single preconditioning matrix for the whole path, we show that learning *iteration-specific* preconditioning provides us the flexibility to model varying trends along the learning paths. *Second,* to learn better context at each time step, we propose a residual connection based gradient preconditioning that allows multiple old gradients to directly flow via the gradient-skip connections. This approach not only provides better context, but also helps avoid gradient vanishing for the task at hand.

Overall, our approach helps in providing an improved modelling of the short-term knowledge specific to a task and also the long-term trends shared between tasks. In a similar pursuit, recurrent neural network (RNN) based meta-learning methods have been explored in the literature [1, 7, 17]. Andrychowicz *et al.* [1] proposed an LSTM meta-learner that learns to mimic a gradient based optimizer, outputting updates at different time steps. Ravi and Larochelle [17] extended the LSTM meta-learner for few-shot settings, where both the base initialization and update mechanism are learned, resulting in a high complexity. In contrast to these approaches, we take a different perspective on aggregating path context. We learn unique trends in each update step while simultaneously combining past context using gradient-skip connections. This results in an easy-to-train model with faster convergence and superior performance.

2 Related Work

Meta-learning algorithms can be grouped into three main categories: **a)** Metric-based, **b)** Model-based and **c)** Gradient-based optimization methods. We outline these below and contrast our work with each line of works.

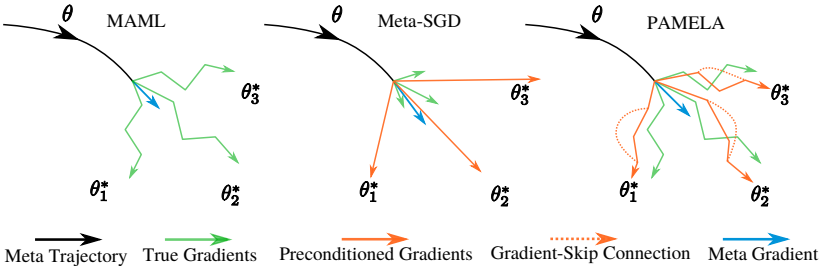


Figure 1: *Inner-loop optimization of model-agnostic gradient-based meta-learning algorithms:* MAML takes multiple steps towards the optimal parameters θ^*, and Meta-SGD takes a single step with a preconditioned direction. While PAMELA takes multiple steps towards θ^* with a unique preconditioned direction for each step, as well as shares the meta information between past and future updates via Gradient-skip connections. This results in a faster convergence closer to the optimal parameters.

Metric and Model-based Meta-learning: Metric-based methods learn an optimal distance kernel, parameterized by meta-parameters. Koch *et al.* [11] used a siamese network to compare the distance between samples. Vinyals *et al.* [21] proposed matching networks, which learn the similarity between support sets and the test samples by producing a weighted sum of the support set labels and attention kernel. Relation networks [20] use a learnable module to predict relation scores in the feature space. Further, prototypical networks [19] define a prototype for each task as an average embedding of the network output, and learn the metric space via error back-propagation. The model-based meta-learning algorithms learn to quickly adapt to the tasks by either having an internal memory or by using fast weights.

Gradient-based Meta-learning: Meta-LSTM [17] models the meta-learner as a recurrent network (LSTM), thereby learning to generate the parameters of the learner at each time step. Finn *et al.* [8] proposed a Model Agnostic Meta-Learning (MAML) algorithm. MAML functions in two loops (inner and outer) and uses the second-order gradients to update the initial parameters. This requires backpropagation through the learning steps of the inner loop. Therefore, to reduce the computational complexity of MAML, the authors also proposed a first-order approximation of MAML (FOMAML), which considers only the last gradient on the inner loop update as the meta-gradient. In addition to FOMAML, Nichol *et al.* [14] used Reptile, which is a better first-order approximation since it computes an average of all inner loop gradients for the meta-update. Reptile is more robust to sample selection and gradient noise. In addition to this, Antoniou *et al.* [2] made few changes to MAML, such as layer-wise learning rates and accumulating meta-loss in each update. Rajeswaran *et al.* [16] used a closed-form approximation to solve the MAML optimization, using a regularized loss function. Khodadadeh *et al.* [9] proposed a semi-supervised version of MAML algorithm while Im *et al.* [8] generalize MAML update with a higher-order ODE solver. However, these MAML based methods treat the inner loop as a fixed learning process and do not share any meta knowledge during the learning updates. In essence, these meta-learning methods fall into *learning-to-initialize* framework, while our work is a combination of *learning-to-initialize* and *learning-to-optimize* paradigms.

Meta-SGD [13] preconditions the inner loop gradients with meta-trainable learning rate, thus allowing the meta-information to be used in the inner loop adaptation process. Similar to Meta-SGD, Park *et al.* [15] proposed Meta Curvature, which learns the curvature information to precondition the gradients. Meta-Curvature performs better than Meta-SGD in

practice, however, it requires higher-order tensor operations to learn the curvature. Recently, Flennerhag *et al.* [6] proposed a warp gradient-based approach, which inserts warp layers in between the network layers, and updates these after a specific interval using the accumulated gradients (meta-update). However, WarpGrad [6] is not model agnostic since it requires modifications in the model architecture. Further, WarpGard preconditions the feature space in the forward pass, while our method preconditions only the gradients, thereby making it totally independent from the model architecture. In addition, different to the above mentioned approaches, PAMELA learns the optimal inner-loop update trends shared across tasks as well as the best combination of knowledge acquired at previous steps.

Meta-Curvature [15] also use gradient precondition to find an optimal point for the adaptation. However, Meta-curvature does preconditioning on the forward gradients in the inner loop, and their goal is to achieve better generalization by this preconditioning during *adaptation* and essentially uses the same MAML style algorithm to learn a good initialization of weights. Our method on the other hand focuses on both good initialization as well as adaptation. With our gradient-skip connections, optimizing the outer-loop objective becomes much easier compared to optimizing MAML objective. Therefore, PAMELA can quickly learn good set of weights for the initialization. Additionally, our choice of $\mathbf{Q}$ is simple element-wise multiplication compared to Meta-curvature's complex matrix multiplications.

MAML++ [2] has proposed various way to improve the stability of MAML training. However, our gradient-skip connection concept is different from MAML++ 's average loss function. MAML++ injects gradients into every step of the update by using a loss function which is evaluated at every step on the validation data (note MAML evaluates on the last step). While our method uses skip connections during the optimization stage ("gradient-skip" connection), thus creating a skip connection in the parameter space, not in the activation space. This will allow us to back-propagate 2nd order gradients via these gradient-skip connections.

3 Path-aware Meta-learning

In a classical learning setting, a model learns knowledge about the training set and applies it on the test set. This paradigm learns each task in isolation and demands large quantities of data and training time for each training cycle. In contrast, meta-learning seeks to learn about learning so that quick adaptation to new tasks is possible. In this pursuit, we propose a new model-agnostic meta-learner called PAMELA, which not only learns a better initialization that can generalize across tasks, but also models the learning trends that reveal how the inner-loop (task-specific update process) evolves during training. Further, it uses the acquired meta-knowledge to combine previous gradients for a stronger context, thereby converging to a better initialization.

3.1 Meta-training

Meta training of gradient-based models, involves learning a good initialization, so that during the adaptation it can converge faster. In this work, we are also interested in learning a good trajectory of the inner-loop. Therefore, during the meta-training we need to learn a set of initialization weights, as well as a set of preconditioning weights which will shape trajectory of the inner-loops. Consider a classification model f_θ, which is parameterized by θ, and $\mathbf{F}_\Phi(\theta)$, which is an inner-loop optimization function of the meta-learner parameterized by Φ. We suppose that a task $\mathcal{T}$ contains two sets of data, a training set $\mathcal{D}_{tr}$ and a validation set $\mathcal{D}_{val}$ i.e., $\{\mathcal{D}_{tr}, \mathcal{D}_{val}\} \in \mathcal{T}$. We want to optimize f_θ on each task $\mathcal{T} \sim P(\mathcal{T})$, so that after learning from multiple tasks of similar nature, $\mathbf{F}_\Phi$ can figure out how to rapidly learn a new

task sampled from the same distribution $P(\mathcal{T})$. This optimization problem is given by:

$$\underbrace{\min_{\theta,\Phi} \mathbb{E}_{\mathcal{T}\sim P(\mathcal{T})}\big[\mathcal{L}_{\mathcal{D}_{val}}(\underbrace{f_{\mathbf{F}_\Phi(\theta)}}_{\text{inner-loop}})\big]}_{\text{outer-loop}}, \tag{1}$$

where $\mathcal{L}$ is a loss function for the given task. As Eq. 1 shows, meta-training consists of two parts: an inner-loop and an outer-loop. The inner-loop learns the task-specific adaptation, while the outer-loop learns about the 'learning process of the inner-loop'. Below, we respectively describe the two nested loops.

3.1.1 Inner-loop Optimization

In PAMELA, the inner-loop optimization function $\mathbf{F}_\Phi$ is defined as an iterative update process. $\mathbf{F}_\Phi$ takes the model parameters θ and gives θ_n after n inner-loop iterations, such that $\mathcal{L}_{\mathcal{D}_{tr}}(f_\theta)$ is minimized,

$$\theta_n \leftarrow \mathbf{F}_\Phi(\theta) \approx \min_{\theta} \mathbb{E}_{\mathcal{D}_{tr}\sim\mathcal{T}}[\mathcal{L}_{\mathcal{D}_{tr}}(f_\theta)]. \tag{2}$$

The inner-loop of PAMELA differentiates it from other gradient-based meta-learning algorithms such as MAML [5] and Meta-SGD [13]. MAML [5] simply follows the gradient descent during inner-loop optimization, hence θ moves along the true gradients. The online update is formally given by: $\theta_{j+1} = \theta_j - \alpha\nabla_{\theta_j}\mathcal{L}_{\mathcal{D}_{tr}}(f_{\theta_j})$, where α is a constant scalar and $j > 0$. However, following the true gradients on a limited number of samples will not converge to a globally optimal set of parameters, which minimizes the loss on all $\mathcal{D}_{tr} \in \mathcal{T}$. In summary, MAML [5] uses no gradient pre-conditioning, thus ignoring *how other tasks update* their parameters. To improve MAML, Meta-SGD preconditions the true gradients by considering α (step size) as a meta-parameter. As a result, θ moves in a direction that can provide the optimal parameters for all tasks. Although Meta-SGD moves θ along a meta-learned direction, it takes only a single step towards the optimal parameters and therefore does not guarantee convergence. Notably, Meta-SGD cannot be simply run with multiple steps since it uses a single preconditioning matrix that cannot model the different directions required to be learned for multiple-steps in the inner-loop. As an example, early updates generally need stronger gradients, while later ones do not; thus, a single parameter cannot encompass the step-wise learning behavior. To validate this, we extend the Meta-SGD with multiple inner-loop steps and notice that it fails to converge with more iterations.

In contrast to the above methods, PAMELA preconditions the true gradients in order to learn *unique* step-wise update directions and to *share* meta-knowledge between different steps. This behavior is illustrated in Fig. 1. Consequently, our inner-loop optimization function $\mathbf{F}_\Phi$ has two sets of learnable meta-parameters, $\boldsymbol{Q}$ and $\boldsymbol{P}^w$, respectively:

$$\Phi = \{\boldsymbol{Q}, \boldsymbol{P}^w\}, s.t., \ \boldsymbol{Q} = \{Q_0, Q_1, ...Q_{n-1}\} \text{ and} \tag{3}$$

$$\boldsymbol{P}^w = \{P_0^w, P_w^w, ..., P_{(n//w-1)*w}^w\}, \tag{4}$$

where // denotes floor division, n is the total number of inner-loop updates and w is the interval size over which we aggregate context using gradient-skip connections. Each Q and P is a vector, with dimensions same as the corresponding layer parameters it represents. The parameters Q_j learn the trends for *each* inner-loop update, while P_j^w learn the correlations *between* current and past updates.

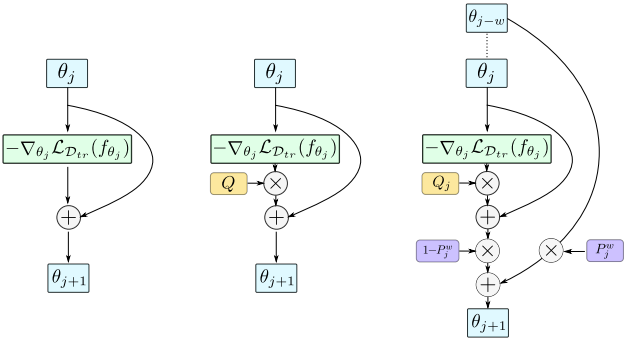


Figure 2: *Comparison between MAML, MetaSGD and our PAMELA. Left:* One step inner-loop update of MAML, which can be understood as an identity gradient-skip connection of the model parameters and learning the gradients as residual signals. However, the gradients are not pre-conditioned; therefore, the whole adaptation process is independent of the meta parameters. *Middle:* Meta-SGD uses a single matrix Q to pre-condition inner-loop gradients. However, this will lead to faster jumps at the end of multiple update steps and potentially collapse the model. *Right:* PAMELA uses a separate pre-condition matrix Q_j for each update step, as well as sharing the information from the previous learned parameters using a longer interval of gradient-skip connection with a coefficient P_j^w, thereby learning the update-trends shared across tasks.

***Q*: Modeling the Learning Directions:** Each element $Q_j \in \boldsymbol{Q}$ controls the learning trend of f_{θ_j} in each inner-loop update step, thus discretely learning and sharing the meta-knowledge between tasks at different time steps. At the j^{th} inner-loop update, we pre-condition the true gradients $\nabla_{\theta_j}\mathcal{L}_{\mathcal{D}_{tr}}(f_{\theta_j})$ by Q_j using the Hadamard product. We use the term true gradients to denote the gradients calculated from the loss function, without any preconditions or projections. Therefore, the model parameters move in a direction that jointly encompasses the true gradient directions across tasks. However, assigning a single meta-parameter for each parameter in the model will suffer from large memory requirements. For example, n inner-loop updates would require n-times more memory. Due to this, we only learn a single meta-parameter for each convolutional kernel. This choice is made because a kernel captures a single feature in the given images, therefore, all the parameters for a given kernel should adapt similarly as the model evolves in the inner-loop. For fully-connected layer, we keep a single meta-parameter for each parameter during inner-loop updates.

***P*^{w}: Modeling the Update Context and Learning Rates:** The meta-parameters P_j^w learn the correspondence between the past and the current gradient updates. For example, P_j^w will fuse the knowledge from θ_{j-w} to θ_j. Here, w (interval-size) is a hyper-parameter that controls the gap between the current and historical gradient update used to calculate the context at a given iteration. Interval size w is fixed through out the learning process and the optimal value is set using a validation set. Finally, we use the following update rule for the j^{th} inner-loop update of $\mathbf{F}_\Phi$,

$$\theta_{j+1} = \begin{cases} \theta_j - Q_j \circ \nabla_{\theta_j}\mathcal{L}_{\mathcal{D}_{tr}}(f_{\theta_j}) & \text{if } (j \bmod w) \neq 0, \\ (\mathbf{1} - P_j^w) \circ \{\theta_j - Q_j \circ \nabla_{\theta_j}\mathcal{L}_{\mathcal{D}_{tr}}(f_{\theta_j})\} + P_j^w \circ \theta_{j-w} & \text{else.} \end{cases} \tag{5}$$

Here, $\circ$ is the Hadamard product and $\mathbf{1}$ is an identity matrix. If the condition $(j \bmod w) \neq 0$

is satisfied, then the update is simple gradient descent with a preconditioning. If not, it first uses a gradient descent update for the model parameters and then combines the updated weights with previous weight from the $(j-w)^{th}$ step. The current weights and the historical weights are combined with coupling coefficients, $1-P_j^w$ and P_j^w, respectively. Similar to $\boldsymbol{Q}$, having a per-parameter model for $\boldsymbol{P}$ increases the memory complexity by the number of gradient-skip connections. Thus, we use a single meta-parameter set $\boldsymbol{P}$ for each layer. This helps to minimize the change in the parameter distribution at a layer.

This predefined weights for the inner-loop adaptation limits our ability to use more inner loop gradient steps, with preconditioning. However, for the datasets we evaluated having 5 number of gradients steps is sufficient to achieve convergence. However, we can still update the inner loop for any number of gradient steps by having $\mathbf{P} = \mathbf{Q} = \mathbf{I}$. Also, the model converges quite fast during the early stages of the inner loop (thanks to preconditioning), therefore after this adaptation the model will be already close to the optimal point.

3.1.2 Outer-loop Optimization

Meta-learning happens in the outer-loop during the optimization process. In the inner-loop, θ is optimized for a task $\mathcal{T}$, and the objective of the outer-loop is to find a new initialization for θ and Φ such that new tasks can be quickly adapted. Thus, in the outer-loop, multiple θ_n^k are available, which are optimized for their corresponding tasks $\mathcal{T}_k \sim P(\mathcal{T})$. To combine the knowledge across all the learned tasks, the outer-loop objective is set to minimize the average loss on $\mathcal{D}_{val}^k$. Therefore, we need the gradients with respect to θ and Φ on $\mathcal{L}_{\mathcal{D}_{val}}(f_{\theta_n})$,

$$\{\theta^{new}, \Phi^{new}\} = \{\theta, \Phi\} - \beta \nabla_{\{\theta,\Phi\}} \sum_{k=1}^{K} \mathcal{L}_{\mathcal{D}_{val}^k}(f_{\theta_n^k}). \tag{6}$$

Here, β is the learning rate of the outer-loop. Overall, the complete meta-training process can be combined as learning θ_n using the inner-loop function $\mathbf{F}_\Phi$, and learning θ and Φ in the outer-loop. Note that, the outer-loop optimization is similar to MAML; however, the gradients of the meta update are different from MAML and Meta-SGD. This is because of the novel *gradient-skip* connections introduced in the inner-loop function, which bypass the gradients in the backward pass and combine multiple inner-loop gradients in the forward pass[1]. It is important to note that '*gradient-skip*' connection denotes bypassing the weights from early updates to the current update, in contrast to the features as in standard residual networks e.g., ResNets.

4 Experiments

We extensively evaluate PAMELA on few-shot supervised classification and regression problems. In both cases, the training set has a large number of tasks with a very small number of samples. The fewer samples per task makes it non-trivial to learn a good representation using traditional supervised learning approaches. However, since there are many tasks available, we can learn 'what is the best way to learn a task'. Therefore, meta-learning frameworks are well suited for few-shot learning problems. In addition to evaluating on few-shot learning, we provide a detailed ablation study on PAMELA (Sec. 4.4).

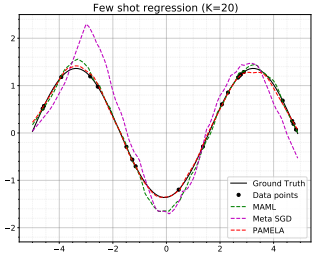
[1] Here, backward pass of gradients means the flow of the second-order gradients during the meta update and the forward pass of gradients means first-order gradients for each inner-loop update.

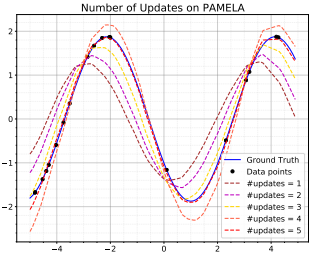
Methods	miniImageNet, 5-way 1-shot	miniImageNet, 5-way 5-shot	CIFAR-FS, 5-way 1-shot	CIFAR-FS, 5-way 5-shot	tieredImageNet, 5-way 1-shot	tieredImageNet, 5-way 5-shot
MAML [8]	48.70 ± 1.84	63.11 ± 0.92	58.9 ± 1.9	71.5 ± 1.9	51.67 ± 1.87	70.30 ± 1.75
MAML++ [2]	52.15 ± 0.26	68.32 ± 0.44	-	-		
FOMAML [8]	48.07 ± 1.75	63.15 ± 0.91	55.6 ± 0.9	70.2 ± 0.7	47.37 ± 0.80	66.12 ± 0.79
Reptile [14]	49.97 ± 0.32	65.99 ± 0.58	-	-		
Meta-LSTM [11]	43.44 ± 0.77	60.60 ± 0.71	43.4 ± 0.8	60.6 ± 0.7	-	-
Meta-SGD [13]	50.47 ± 1.87	64.03 ± 0.94	56.9 ± 0.9	70.1 ± 0.7	50.92 ± 0.93	69.28 ± 0.80
iMAML-HF [16]	49.30 ± 1.88	-	-	-		
MT-Net [11]	51.70 ± 1.84	-	-	-		
R2D2 [3]	49.50 ± 0.20	65.40 ± 0.20	62.3 ± 0.2	77.4 ± 0.2	-	-
L-MAML [1]	49.40 ± 1.83	-	-	-		
HSML [11]	50.38 ± 1.85	-	-	-		
PAMELA	**53.50 ± 0.89**	**70.51 ± 0.67**	**63.5 ± 0.9**	**79.1 ± 0.7**	**54.81 ± 0.88**	**74.39 ± 0.71**

Figure 3: Few-shot learning results on miniImageNet [17], CIFAR-FS [3] and tieredImageNet [18] datasets. All accuracies are in %.

4.1 Few-shot Classification

Settings: For an M-way N-shot setting, N samples are given for M different classes in each task. Therefore, for a k^{th} task $\mathcal{T}_k$, the training dataset $\mathcal{D}^k_{tr} \in \mathcal{T}_k$ has $N * M$ samples. For all experiments, we use 15 samples per class for validation during the meta-update, hence $\mathcal{D}^k_{val} \in \mathcal{T}_k$ has $M * 15$ samples.

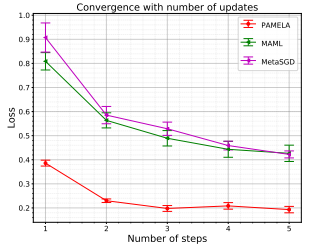


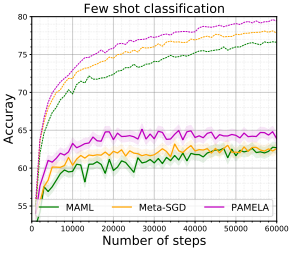


K	Method	MSE
5	MAML [8]	1.13 ± 0.18
	Meta-SGD [13]	0.90 ± 0.16
	MT-Net [11]	0.76 ± 0.09
	PAMELA	0.54 ± 0.06
10	MAML [8]	0.77 ± 0.11
	Meta-SGD [13]	0.53 ± 0.09
	MT-Net [11]	0.49 ± 0.05
	PAMELA	0.41 ± 0.04
20	MAML [8]	0.48 ± 0.08
	Meta-SGD [13]	0.31 ± 0.05
	MT-Net [11]	0.33 ± 0.04
	PAMELA	0.17 ± 0.03

Figure 4: **Left:** We compare PAMELA with MAML and Meta-SGD on the sine wave regression problem. We can see that PAMELA fits the curve better, and the curve generated by PAMELA is smoother than the curve generated by MAML and Meta-SGD. **middle:** We also show that how the function changes with the number of inner-loop updates. Note that, at 4th update, the model over shoots, however in the next update it recovers the better function. **right:** *Mean squared error on sine wave regression:* We test MAML-based algorithms on sine wave regression with $K = \{5, 10, 20\}$. PAMELA achieves the lowest error in all three cases. *(lower is better)*

Results: We compare PAMELA with several other model-agnostic gradient-based methods: MAML [8], iMAML [16], FOMAML [8], Reptile [14] and Meta-SGD [13]. On the miniImagenet 5-way 5-shot setting, with a total of 600 test tasks (out of possible 15,504 tasks), PAMELA achieves $70.51 \pm 0.67\%$, whereas for the 5-way 1-shot setting, it achieves $53.50 \pm 0.89\%$. This is an absolute gain of 2.19% and 1.35% over the current state-of-the-art method. Similarly, on the CIFAR-FS dataset, our method achieves state-of-the-art performance compared to the family of MAML algorithms. On the 5-way 5-shot setting, we achieve $79.1 \pm 0.7\%$, with a gain of 1.7%, and on 5-way 1-shot setting, we achieve





Method	Accuracy (%)
MAML	63.11 ± 0.92
MAML + Q_0 (single)	64.03 ± 0.94
MAML + Q_0 (multiple)	62.71 ± 0.69
MAML + $\boldsymbol{Q}$	69.71 ± 0.66
MAML + $\boldsymbol{P}$	68.30 ± 0.67
MAML + $\boldsymbol{Q}$ + $\boldsymbol{P}^1$	70.72 ± 0.63
MAML + $\boldsymbol{Q}$ + $\boldsymbol{P}^2$	70.17 ± 0.67
MAML + $\boldsymbol{Q}$ + $\boldsymbol{P}^3$	70.31 ± 0.69
MAML + $\boldsymbol{Q}$ + $\boldsymbol{P}^4$	70.51 ± 0.65

Figure 5: **Left:** Loss on different steps of the inner-loop updates. We can see that PAMELA converges with decreasing variation in error compared to MAML and does not overfit with the increasing number of inner-loop updates. Losses were calculated across 100 runs. **middle:** *Convergence Curves:* We plot the train accuracy (broken lines) and test accuracy (solid lines) for the training of PAMELA, MAML and Meta-SGD. The plot demonstrates that PAMELA achieves a given accuracy faster than the other compared methods. **right:** *Dissecting PAMELA*: Results are reported for miniImageNet 5-way 5-shot experiments with 600 testing samples, and $n = 5$. Means and 95% confidence intervals are reported.

$63.5 \pm 0.9\%$ with a gain of 1.2%. Further, we also surpass MAML, Meta-SGD on 5- and 1-shot learning on tieredImageNet with $54.81 \pm 0.88\%$ and $74.39 \pm 0.71\%$. The path learning behaviour of PAMELA provides valuable context and gradient preconditioning, resulting in strong improvements compared to other model-agnostic gradient-based methods.

4.2 Regression

Settings: We consider sine wave regression as our learning problem. Each task/wave is parameterized by amplitude, frequency and phase sampled uniformly in ranges $[0.1, 5]$, $[0.8, 1.2]$ and $[0, \pi]$, respectively. In addition, a task $\mathcal{T}$ contains $K + 10$ samples uniformly drawn from the true curve. The first K samples are used for the inner-loop adaptation and the remaining 10 used to compute the meta-loss.

Results: To evaluate the effectiveness of the model, we first sample 1000 random sine waves from the same distribution as the training set. Then we sample K random points on the curve to update the model. Afterwards, we sample 1000 points on the curve uniformly to find the Mean Squared Error (MSE) to evaluate the fitness of the curve. We test on MAML, Meta-SGD, MT-Net and PAMELA with $K = \{5, 10, 20\}$. In all the cases, our method obtains the lowest error, and with increasing K the loss decreases faster (see Fig. 4 right). From Fig. 4, we can see that the curves regressed by PAMELA fit the target function well, and are much smoother compared to other methods. Additionally, Fig. 5 shows how the loss changes with the number of inner-loop updates. We can see that, PAMELA has better convergence compared to MAML (standard deviation of loss decreases with inner-loop updates).

4.3 Implementation Details

For experiments on miniImageNet, CIFAR-FS and tieredImageNet, we use a four-layer convolutional neural network, each with 64 filters. During the meta update we use 15 samples per class in the validation data to compute the meta gradients and batch size of four tasks used in all the experiments. For all the experiments, the inner-loop is updated five times with two gradient-skip connections using $\boldsymbol{P}^2$. Also, the inner-loop learning rate is set to 0.01 and, in the outer-loop, Adam [10] optimizer with an initial learning rate of 0.001 is employed.

4.4 Analysis and Ablation Studies

Dissecting PAMELA: We analyze individual contributions of different components of our approach. We can breakdown PAMELA as a mixture of MAML + $\boldsymbol{Q}$ + $\boldsymbol{P}^w$. First, we see how $\boldsymbol{Q}$ affects the model learning curve. If we use only one Q for all the inner-loop updates, extending Meta-SGD, the performance degrades. This is because Q acts as a learning rate controller for each step, while having the same learning rates for all the steps, may cause the model to over-shoot or prevent it from converging. However, using different Q_j for each update helps to boost the performance by 3.07%. In addition to $\boldsymbol{Q}$, $\boldsymbol{P}^w$ helps to learn the trend across different updates. Therefore, adding $\boldsymbol{P}^w$ to the loop makes the learning more generic. As a result, the gradient-skip connection ($\boldsymbol{P}$) gives the highest boost in the performance, with a 7.40% gain over MAML. However, $\boldsymbol{P}^w$ contains P_j^w with a hyper-parameter of interval size (w), which controls the gap between each gradient-skip connection. We can see how the length of the gradient-skip connections affects the learning from Fig. 5 (right). Our results show that the influence of skip connection length is not very significant.

Number of inner-loop updates and skip connections: Our inner-loop is modeled with two hyper-parameters: the number of updates and the width of the skip connections. First, as we change the number of inner-loop updates, we notice the performance does not change significantly except for the single update scenario. This is mainly because the model's inner-loop path has an extra degree of freedom on how far the parameters have to move. While training the model, it can learn how the inner-loop path has to change for a given constraint on the total updates. All these experiments were done on miniImageNet with 5-way 5-shot problems with $\boldsymbol{P}^2$ gradient-skip connections. Secondly, the gradient-skip connection can also change the dynamics of the adaptation. Although it helps to have the gradient skip connections, the width of the connections does not have a significant impact on the performance. However, a depth of 5-10 updates may not be sufficient to prove this claim.

Method	#Parameters	#Updates	Train time	Test time
MAML	121.09×10^3	5	205.50 ms	41.33 ms
MetaSGD	242.18×10^3	1	53.75 ms	16.67 ms
PAMELA	162.39×10^3	5	215.84 ms	43.33 ms

Table 1: *Model complexity:* We compare the number of parameters, train time per task and test time per task in a 4-layer 64-channel CNN.

5 Conclusion

Model-agnostic meta-learning aims to combine across-task knowledge in order to find a better initialization for the learner. From this initialization, the model can be quickly fine-tuned on a new task with only a few examples. In this work, we propose to learn a gradient preconditioning at each inner-loop iteration to model how learning evolves over time across multiple tasks. Further, our approach utilizes historical gradients from old updates, which provides valuable context and prevents over-fitting as well as gradient vanishing. Overall, our approach achieves faster convergence and performs better classification datasets, including miniImageNet, tieredImageNet, and CIFAR-FS. We also evaluate our approach on a sine wave regression task, where it fits the curve much more accurately than the state-of-the-art methods. Further, we analyse the contribution of each respective component towards the final performance via an extensive ablation study.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

[3] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[4] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *International Conference on Learning Representations*, 2018.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

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

[7] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[8] Daniel Jiwoong Im, Yibo Jiang, and Nakul Verma. Model-agnostic meta-learning using runge-kutta methods. *arXiv preprint arXiv:1910.07368*, 2019.

[9] Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. In *Advances in Neural Information Processing Systems*, pages 10132–10142, 2019.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[12] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2927–2936, 2018.

[13] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

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

[15] Eunbyung Park and Junier B Oliva. Meta-curvature. In *Advances in Neural Information Processing Systems*, pages 3309–3319, 2019.

[16] 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.

[17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

[18] 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. *arXiv preprint arXiv:1803.00676*, 2018.

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

[20] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[21] 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.

[22] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, pages 7045–7054, 2019.