EMO: Episodic Memory Optimization for Few-Shot Meta-Learning

Anonymous authors
Paper under double-blind review

Abstract

For few-shot meta-learning, gradient descent optimization is challenging due to the limited number of training samples per task. Inspired by the human ability to recall past learning experiences from the brain's memory, we propose an episodic memory optimization for meta-learning, which we call *EMO*, that retains the gradient history of past experienced tasks in external memory. It enables few-shot learning in a memory-augmented way by leveraging the meta-learning setting and learns to retain and recall the learning process of past training tasks for gradient descent optimization. By doing so, EMO nudges the parameter updates in the right direction, even when the gradients provided by a limited number of examples are uninformative. Additionally, we prove theoretically that our algorithm converges for smooth, strongly convex objectives. EMO is generic, flexible and model-agnostic, making it a simple plug-and-play optimizer seamlessly embedded into existing optimization-based meta-learning approaches. Empirically, EMO scales well with most of the few-shot classification benchmarks, and our experiments show that the optimization-based meta-learning method enjoys accelerated convergence and improved performance with EMO.

1 Introduction

The vast majority of current few-shot learning methods fall within the general paradigm of meta-learning (Bengio et al., 1991; Schmidhuber, 1987; Thrun & Pratt, 1998). It searches for the best few-shot learning strategy as the learning experiences increase (Finn et al., 2017; Ravi & Larochelle, 2017; Andrychowicz et al., 2016). Optimization-based meta-learning (Finn et al., 2017; Ravi & Larochelle, 2017; Li et al., 2017; Raghu et al., 2019) is one of the most popular approaches, owing to its "model-agnostic" nature to incorporate different model architectures and its principled formulation that allows the application to various problems. Optimization-based meta-learning comprises inner-loop and outer-loop updates that operate on a batch of tasks per iteration. In the inner-loop, these methods learn task-specific network parameters θ by performing traditional gradient descent on a task-specific loss $\mathcal{L}(\theta;\mathcal{S})$ with the support set $\mathcal{S}$, where

$$\theta^{'} = \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta;\mathcal{S}), \tag{1}$$

and α is the learning rate which determines the step size per inner iteration. A major drawback of optimization-based meta-learning is meta-overfitting (Mishra et al., 2018; Yin et al., 2019; Rajendran et al., 2020). Gradient estimation with a small support set is inherently noisy, which causes the model to diverge or converge to a non-optimal minimum per task. Due to the small number of samples, traditional optimizers, e.g., (Allen-Zhu & Yuan, 2016; Kingma & Ba, 2015; Sutskever et al., 2013; Ruder, 2016; Duchi et al., 2011), tend to get trapped in local minima. In this paper, we propose a new inner-loop optimizer for few-shot meta-learning.

We are inspired by the human ability to quickly adapt to new tasks with limited training samples by recalling past learning experiments from episodic memory (Tulving, 1972; 1983; 2002). As a long-term memory, episodic memory has previously demonstrated its effectiveness in various machine learning tasks. In reinforcement learning, for example, recent works (Zhu et al., 2020; Gershman & Daw, 2017) use episodic memory to store past experiences to adapt to new environments and improve generalization ability quickly. In continual learning, episodic memory alleviates catastrophic forgetting (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019), while allowing beneficial knowledge transfer to previous tasks. We draw inspiration from the cognitive function of episodic memory and introduce it into meta-learning to learn to collect long-term episodic (optimization) knowledge for few-shot learning.

This paper proposes a new inner-loop optimization method for few-shot meta-learning, called *Episodic Memory Optimization* (EMO). We exploit an external memory to accrue and store the gradient history gained from past training tasks, which helps the model to update to optimal parameters more accurately and quickly when faced with new tasks. Specifically, stored in the episodic memory are the gradients of the parameters per network layer for previous tasks. When learning a new task, we retrieve a set of similar learning processes in the memory, aggregating them with the gradient of the current task in a linear or learnable way. By doing so, episodic memory could help us achieve more optimal model parameters, despite having only limited training samples available for the new task. To avoid overloading the memory storage space, we propose a memory controller which implements three different replacement strategies to replace the task in the memory. We also prove that EMO with fixed-size memory converges under strong convexity assumptions, regardless of which gradients are selected or how they are aggregated to form the update step. EMO is a general gradient descent optimization that is model-agnostic and serves as a plug-and-play module that can seamlessly be embedded into existing optimization-based meta-learning approaches. We conduct our ablations and experiments on the few-shot learning benchmarks and verify that the optimization-based meta-learning methods with EMO easily outperform the original methods in terms of both performance and convergence.

2 Method

2.1 Preliminaries

Few-shot classification The goal of few-shot classification is to construct a model using a limited number of labelled examples. In the conventional few-shot classification scenario, following (Vinyals et al., 2016), we define the N-way K-shot classification problem, which has N classes, and each class has K labelled support examples. In this scenario each task is a classification problem from a predefined task distribution $p(\mathcal{T})$. We denote the labeled support set by $\mathcal{S}=\{(x_i, y_i)\}_{i=1}^{N\times K}$; each (x_i, y_i) is a pair of an input and a label, where $y_i \in \{1, 2, \cdots, N\}$. Each task is also associated with a query set $\mathcal{Q}=\{(x_j, y_j)\}_{j=1}^{M\times K}$ to evaluate the quality of the trained model. The query set $\mathcal{Q}$ for each task is also composed of examples of the same N classes. Usually, optimizing and learning parameters for each task with a few labelled training samples is difficult. Meta-learning offers a way of learning to improve performance by leveraging knowledge from multiple tasks.

Optimization-based meta-learning In meta-learning, a sequence of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}$ are sampled from a predefined task distribution $p(\mathcal{T})$, where each one is a few-shot learning task. The core idea of meta-learning is to find a well-generalized meta-learner on the training tasks during the meta-training phase. For each task $\mathcal{T}_i$, the meta-learner $\mathcal{M}$ is applied on the base learner f_{θ_i}, and the parameter θ_i and meta-learner $\mathcal{M}$ are learned alternatively. During the meta-testing phase, the learned meta-learner is applied to tackle the new tasks composed of examples from unseen classes. Given a new few-shot learning task $\mathcal{T}_t$, the optimal meta-learner $\mathcal{M}^*$ is used to improve the effectiveness of $\mathcal{T}_t$ by solving $\min_{\theta_t} \mathcal{L}(\mathcal{M}^*(f_{\theta_t}), \mathcal{Q})$. In this way, meta-learning effectively adapts to new tasks, even when the training data for the new task is insufficient. Optimization-based meta-learning (Finn et al., 2017; Li & Malik, 2017; Raghu et al., 2019) strives to learn an optimization that is shared across tasks while being adaptable to new tasks. The most representative optimization-based meta-learning is model-agnostic meta-learning (MAML) by (Finn et al., 2017). MAML is formulated as a bilevel optimization problem with inner-loop and outer-loop optimization, where the inner-loop computes the task-specific parameters θ' (starting from θ) via a few gradient updates:

$$\theta' = \theta - \alpha \nabla_\theta \frac{1}{N \times K} \sum_{(x,y)\in\mathcal{S}} \mathcal{L}_{\mathcal{T}_i}(f_\theta(x), y) \ . \tag{2}$$

For the outer-loop, the original model parameters θ are then updated with respect to the performance after the inner-loop update, i.e.,

$$\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N_\mathcal{T}} \sum_{\mathcal{T}} \frac{1}{M \times K} \sum_{(x,y)\in\mathcal{Q}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'}(x), y) \ . \tag{3}$$

where α and β are the inner-loop and outer-loop learning rates, respectively. Training result is a model initialisation θ that can be adapted to any new task with just a few gradient steps.

2.2 Model

This paper focuses on inner-loop optimization for optimization-based meta-learning methods. We propose a new inner-loop optimization method called *Episodic Memory Optimization* (EMO). Our proposed EMO model is composed of four parts: an **encoder** that generates a representation for the incoming query data and the available support data; an external **memory store** which contains previously seen task representations and the gradients of each layer with writing managed by a **memory controller**; and an **episodic gradient memory stochastic gradient descent** that ingests the gradients from the new task and data from the memory store to generate the new gradients over the current task.

The encoder first takes in support data $\mathcal{S}=\{(x_i, y_i)\}_{i=1}^{N\times K}$ and then converts these data to the representation $\{e_i\}_{i=1}^{N\times K}$ of lower dimension. In this paper, the input is an image, and we choose a convolutional network architecture for the encoder function f_θ.

Memory store Our external memory module $\mathcal{M}=\{M_t\}_{t=1}^{T}$ contains the stored learning process of experienced tasks, where t is the memory capacity. Each of the slots corresponds to each experienced task. In our work, the memory stores a key-value pair in each row of the memory array as (Graves et al., 2014). The keys are the task representations k^t of each task, the value is the gradient representation as value V_t^l, $M_t = [\mathbf{K}_t, \mathbf{V}_t]$, where $\mathbf{V}_t = \{V_t^1, V_t^2, \cdots, V_t^l\}$, t indicates the task t and l indicates the l-th convolutional layer. The memory module is queried by finding the k-nearest neighbors between the test task representation and the task K_t in a given slot. The distance metric used to calculate proximity between the points is an available choice, and here we always use Euclidean distance.

For the task representation K_t, to allow the flexibility of variable input sizes of task representations, we use the generic Transformer architecture (Vaswani et al., 2017):

$$K_t = \texttt{Transformer}([\texttt{cls}_t, e_1, e_2, \cdots, e_n])[0], \tag{4}$$

where $\texttt{cls}_t$ is the task representation token embedding, and $e_i = \textbf{Encoder}(x_i)$ is the encoded i-th support data pair $\mathcal{S}=\{(x_i, y_i)\}_{i=1}^{N\times K}$. After the transformer, we take the position output $\texttt{cls}$ as the task embedding K_t.

For the memory value V_t^l, we first compute the gradients of task t at layer l as:

$$\mathbf{g}_t^l = \sum_{i=1}^{N\times K} \frac{\partial \mathcal{L}(\hat{y}_i^t, y_i^t)}{\partial \theta^l}, \tag{5}$$

where θ^l uses the parameters at layer l, we denote with $\mathcal{L}(\cdot)$ a loss function (such as the cross entropy loss on the labels). To avoid confusion, we omit the superscript l for the memory from now on.

Memory controller To avoid overloading the memory storage space, we propose a memory controller that decides to replace the episodic memory slot at a certain moment. The input of the memory controller consists of the gradient $\mathbf{g}_t$ of the current task and the selected memory $\hat{M}_c$ that needs to be replaced. The controller is written as:

$$M_c = \texttt{Controller}(\mathbf{g}_t, \hat{M}_c). \tag{6}$$

Inspired by the page replacement algorithm in operating systems, we propose three implementations of the memory controller: *First In First Out Episodic Memory* (`FIFO-EM`), *Least Recently Used Episodic Memory* (`LRU-EM`) and *Clock Episodic Memory* (`CLOCK-EM`). (1) `FIFO-EM` keeps track of all the memory in a queue, with the oldest memory in the front of the queue. When a memory needs to be replaced, the memory in the front of the queue is selected for removal. (2) `LRU-EM` is a content-based memory writer that writes episodic memories to either the least used or the most recently used memory location. New task information is written into rarely-used locations, preserving recently encoded data, or written to the last used location, which can function as an update of the memory with newer, possibly more relevant information. (3) In the `CLOCK-EM`, the candidate memory for removal is considered in a round robin fashion and a memory that has been accessed between consecutive considerations will be spared, similar to the CLOCK page replacement algorithm in the operating system (Janapsatya et al., 2010). When the memory $\mathcal{M}$ is not complete, we directly store the $\mathbf{g}_t$ to be added into $\mathcal{M}$, while once the memory is complete, we use the Controller to achieve memory replacement. The best-suited memory controller is specific to the underlying meta-learning method and datasets. We provide the comparison of each `Controller` in the experiments.

Episodic gradient memory stochastic gradient descent Episodic gradient memory stochastic gradient descent is the explicit integration of episodic memory gradients into SGD. To be specific, the iteration comes in the form:

$$\theta_{t+1} = \theta_t - \alpha \cdot \texttt{Aggr}(\mathbf{g}_t,\ \mathcal{V}_t), \tag{7}$$

where $\mathbf{g}_t$ are the gradients of the support set from the current task, $\mathcal{V}_t$ is the collection of episodic gradients selected from the memory based on the similarity of memory key and the current task representation, and `Aggr` denotes an aggregation function which is used to combine the episodic gradients with the current-iteration gradient. We consider three possible functions for `Aggr` including `Mean`, the average of $\mathcal{V}_t$ and all selected episodic gradients; `Sum`, the addition of $\mathcal{V}_t$ to the average of all selected episodic gradients; and `Transformer`, the learnable combination of $\mathcal{V}_t$ to all the selected episodic gradients. Mathematically, these three `Aggr` functions are defined as:

$$\texttt{Mean}(\mathbf{g}_t,\ \mathcal{V}_t) = \frac{1}{M_{\mathcal{V}_t}+1}(\mathbf{g}_t + \sum_{V_t \in \mathcal{V}_t} V_t), \tag{8}$$

$$\texttt{Sum}(\mathbf{g}_t,\ \mathcal{V}_t) = \mathbf{g}_t + \frac{1}{M_{\mathcal{V}_t}} \sum_{V_t \in \mathcal{V}_t} V_t, \tag{9}$$

$$\texttt{Transformer}(\mathbf{g}_t,\ \mathcal{V}_t) = \texttt{Transformer}([\texttt{cls}_g, \mathbf{g}_t, V_t^1, V_t^2, \cdots, V_t^{M_{\mathcal{V}_t}}])[0]. \tag{10}$$

where $\texttt{cls}_g$ is the new gradients token embedding. The best-suited aggregation function is specific to the meta-learning method into which the episodic gradients are integrated. We compare each aggregation function with different optimization-based meta-learning methods in the experiments.

2.3 Meta-training and meta-test

Following (Ravi & Larochelle, 2017; Finn et al., 2017), we perform episodic training by exposing the model to a variety of tasks from the training distribution $p(\mathcal{T})$. For a given training task $\mathcal{T}_i$, the model first computes its parameters by Eq. (7) in the inner-loop, then incurs a loss $\mathcal{L}_i$ of this task, and updates the model parameters by Eq. (3); we sum these losses and back-propagate through the sum at the end of the task. We evaluate the model using a partition of the dataset that is class-wise disjoint from the training partition. In the meta-test stage, the model first computes the gradients $\mathbf{g}_t$ and recalls the memory $\mathcal{V}_t$ based on the task representation k, which is computed by the support set $\mathcal{S}$. Then the model updates the task-specific parameters by Eq. (7) in the inner-loop. After the inner-loop, we evaluate the model on the query set.

2.4 Analysis of Convergence

The core of EMO is to explicitly integrate the current gradient with the episodic memory $\texttt{Aggr}(\mathbf{g}_t, \mathcal{V}_t)$. In practice, we observe that the proposed method has a higher convergence rate than previous optimizers. Here, we theoretically analyze the proposed EMO optimization's convergence rate of gradient descent.

To do so, we reformulate the aggregation process as a linear multi-step system (Polyak, 1964; Assran & Rabbat, 2020), leading to $\texttt{Aggr}(\mathbf{g}_t, \mathcal{V}_t) = \sum_{s=0}^{S-1} w_{t,s} g_{t-s}$. S is the number of steps. At the t-th iteration, the multi-step system involves the gradients from the past S time steps. $w_{t,s}$ is the aggregation scalar of the corresponding gradient g_{t-s} in the linear multi-step system, which is bounded by the interval $[0, 1]$. The system involves all gradients in the episodic memory, $\mathcal{V}_t \subseteq \{g_{t-s}\}_{s=0}^{S}$. For the gradient that does not appear in the memory, $\exists\ s \in \{1, 2, ..., S\}, g_{t-s} \notin \mathcal{V}_t$, the corresponding aggregation scalar $w_{t,s}$ is 0. In general, we define a model-agnostic objective as

$\min_\theta f(\theta)$. θ_t and θ^* denote model parameters of the t-th iteration and the optimal. The difference between both parameters is $\Delta\theta_t = \theta_t - \theta^*$. We assume f is continuously differentiable, μ-strongly convex and L-Lipschitz ($0 < \mu \leq L$). These assumptions imply the Hessian matrix $\nabla^2 f(\theta)$ exists and is bounded by the interval $[\mu, L]$. We consider the stochastic gradient g_t as a random vector and $\mathbb{E}[g_t] = \nabla f(\theta_t)$. ϵ_t denotes the independent gradient noise at iteration t. The gradient noise has zero mean, and its variance is bounded by a finite constant σ^2. Thus, the gradient in each iteration can be formulated as:

$$g_t = \Delta\theta_t \int_0^1 \nabla^2 f(\theta^* + u\Delta\theta_t)du + \epsilon_t, \tag{11}$$

where $\int_0^1 \nabla^2 f(\theta^* + u\Delta\theta_t)du$ is the average rate of the gradient changes from the t-th iteration to the optimal one with respect to the model parameters. Based on the assumptions of the objective, the average rate is also bounded between μ and L. We incorporate Eq. (11) into Eq. (7) with the linear multi-step system. In this case, the convergence of the system depends on the spectral properties of the system matrix (McRae et al., 2022).

Theorem 1 (Convergence rate of EMO)**.** We define a system matrix[1] for each iteration as A_t, which contains aggregation scalars and average rates of gradient changes of the past S gradients. λ_t is the square root of the largest singular value of the corresponding system matrix, and thus the spectral norm of the system matrix is not larger than λ_t. $\lambda_{\max}$ is the upper bound for all λ_t corresponding to all system matrices. Since α is chosen sufficiently small such that $\lambda_{\max} < 1$, we have that:

$$f(\theta_{t+1}) - f(\theta^*) \leq \frac{L}{2}(\lambda_{\max}^{2t} \left\|\Delta\theta_1\right\|^2 + \frac{\alpha^2\sigma^2 S}{1-\lambda_{\max}^2}). \tag{12}$$

From this theorem, the learning rate mainly depends on $\lambda_{\max}$. The lower $\lambda_{\max}$, the faster the convergence rate and the smaller the variance. When the number of step in the system is 1, Eq. (12) degenerates to the conventional stochastic gradient decent as used in the previous meta-learning methods (Finn et al., 2017; Li et al., 2017; Raghu et al., 2019). In practice, our model usually sets a large number as the number of steps. In this case, it is possible to set the learning rate and aggregation scalars to obtain a faster convergence rate than SGD. Proofs are presented in Appendix B.

3 Related Work

Episodic memory Episodic memory has shown its effectiveness in a variety of machine learning tasks. When episodic memory is used for reinforcement learning, explicit records of past events are retained and used to make decisions about the current situation. The chosen action is associated with the highest value based on the outcome of similar situations in the past. Recent works (Zhu et al., 2020; Gershman & Daw, 2017; Botvinick et al., 2019; Hu et al., 2021; Lampinen et al., 2021) use episodic memory to store past experiences to help the intelligence quickly adapt to new environments and improve its generalization ability. In continual learning, episodic memory alleviates catastrophic forgetting (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Derakhshani et al., 2021) while allowing beneficial transfer of knowledge to previous tasks. We draw inspiration from the cognitive function of episodic memory and introduce it into meta-learning to learn to collect long-term episodic (optimization) knowledge for few-shot learning.

Meta-learning Meta-learning designs models to learn new tasks or adapt to new environments quickly with only a few training examples. There are four common research lines of meta-learning: (1) metric-based meta-learning (Snell et al., 2017; Vinyals et al., 2016; Sung et al., 2018; Oreshkin et al., 2018; Yoon et al., 2019; Du et al., 2022) generally learn a shared/adaptive embedding space in which query images can be accurately matched to support images for classification; (2) optimization-based meta learning (Finn et al., 2017; 2018; Lee & Choi, 2018; Yoon et al., 2018; Grant et al., 2018; Kalais & Chatzis, 2022; Abbas et al., 2022; Flennerhag et al., 2021; Zou et al., 2021) learns an optimization algorithm that is shared across tasks, and can be adapted to new tasks, enabling learning to be conducted efficiently and effectively; (3) model-based meta-learning (Mishra et al., 2018; Gordon et al., 2019) explicitly learns a base-learner that incorporates knowledge acquired by the meta-learner and effectively solves individual tasks; (4) memory-based meta-learning (Munkhdalai &

[1]For clarity, we provide the definition of A_t in Eq. (16) of Appendix B.

Yu, 2017; Ramalho & Garnelo, 2019; Zhen et al., 2020; Santoro et al., 2016; Du et al., 2022) deploys an external memory to rapidly assimilate new data of unseen tasks, which is used for quick adaptation or to make decisions. Our method combines optimization-based meta-learning with memory-based meta-learning. To the best of our knowledge, it is the first optimization-based meta-learning method with episodic memory, intending to perform few-shot classification.

Memory-based few-shot learning (Andrychowicz et al., 2016) and (Ravi & Larochelle, 2017) propose the update rule for neural network parameters by transforming gradients via an LSTM, which outperforms fixed SGD update rules. The Meta-network (Munkhdalai & Yu, 2017) learns to transform the gradients to fast weights as memory, which are stored and retrieved via attention during testing. Conditionally shifted neurons (Munkhdalai & Trischler, 2018) modify the activation values with task-specific shifts retrieved from an external memory module, which is populated rapidly based on limited task experience. (Santoro et al., 2016) leverages the Neural Turning Machine (Graves et al., 2014) for online few-shot learning by designing efficient read and write protocols. (Ramalho & Garnelo, 2019) introduced adaptive posterior learning, which approximates probability distributions by remembering the most surprising observations it has encountered in external memory. (Babu et al., 2021) proposed a distributed memory architecture, which recast the problem of meta-learning is simply learning with memory-augmented models. These methods (Andrychowicz et al., 2016; Ravi & Larochelle, 2017) leverage an LSTM to design a new update rule for the network parameters, which can be seen as implicit memory. The other memory-based methods (Munkhdalai & Yu, 2017; Munkhdalai & Trischler, 2018; Santoro et al., 2016; Ramalho & Garnelo, 2019; Babu et al., 2021) design an external memory to transform their neuron activations for fast adaptation. Our approach differs from these methods since we propose an external memory to retain and recall past training tasks' learning process for updating network parameters.

4 Experiments

4.1 Experimental Setup

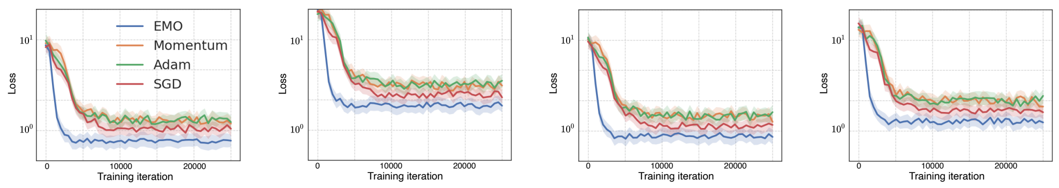
In our experiments we consider two datasets: (i) *Meta-Dataset-BTAF* (Yao et al., 2019), which contains four fine-grained image classification datasets: (a) *Bird* (Wah et al., 2011), *Texture* (Cimpoi et al., 2014), *Aircraft* (Maji et al., 2013), and *Fungi* (FUNGI, 2018). (ii) *mini*ImageNet (Vinyals et al., 2016) which consists of 100 randomly chosen classes from ILSVRC2012 (Russakovsky et al., 2015). For the *Meta-Dataset-BTAF*, each meta-training and meta-test task samples classes from one of the four datasets. This benchmark is more heterogeneous and closer to real-world image classification. Following the conventional meta-learning settings (Vinyals et al., 2016; Finn et al., 2017), all datasets are divided into meta-training, meta-validation and meta-testing classes. The N-way K-shot settings are used to split the training and test sets for each task. We report the average few-shot classification accuracy (%, top-1) along with the 95% confidence intervals across all test images and tasks. The detailed implementation is provided in Appendix A. Our code will be released.

4.2 Results

Benefit of episodic memory optimizer To show the benefit of our proposed episodic memory optimizer, we compare MAML (Finn et al., 2017), Meta-SGD (Li & Malik, 2017), and ANIL (Raghu et al., 2019) with their EMO variants. Each original meta-learning method uses SGD as the inner-loop optimizer, while each EMO variant uses EMO as the inner-loop optimizer. Table 1 shows adding EMO improves performance independent of the meta-learning method or dataset. On the challenging Texture dataset, which has the largest domain gap, Meta-SGD with EMO delivers 36.26%, surpassing Meta-SGD by 3.88%. In addition, Meta-SGD with EMO achieves the best performance compared with other meta-learning methods. This is because Meta-SGD with EMO stores not only the gradients of each layer, but also the gradients of the learning rate in the inner-loop, thus accelerating training. ANIL only stores the gradients of the last layer, causing the number of parameters and the memory size to be much smaller than in MAML and Meta-SGD. Despite the reduced accuracy, ANIL with EMO is still beneficial for applications that require compute efficiency, as ANIL is about 4.8 times faster than MAML and Meta-SGD. We attribute the improvements with EMO to our model's ability to leverage the episodic memory to adjust the model parameters, allowing the model to update the test task model using the most similar training task-like update.

Table 1: Benefit of episodic memory optimizer for few-shot fine-grained classification. All evaluated optimization-based meta-learning methods consistently achieve better performance with EMO than without. Meta-SGD with EMO achieves the best performance, especially for the 5-way 5-shot setting.

	MAML		ANIL		Meta-SGD	
Dataset	w/o EMO	w/ EMO	w/o EMO	w/ EMO	w/o EMO	w/ EMO
5-way 1-shot						
Bird	53.94 $\pm$ 1.45	56.32 $\pm$ 1.33	52.57 $\pm$ 1.44	54.78 $\pm$ 1.43	55.58 $\pm$ 1.43	**58.95** $\pm$ 1.41
Texture	31.66 $\pm$ 1.31	34.75 $\pm$ 1.41	31.45 $\pm$ 1.32	33.15 $\pm$ 1.31	32.38 $\pm$ 1.32	**36.26** $\pm$ 1.33
Aircraft	51.37 $\pm$ 1.38	53.99 $\pm$ 1.33	50.45 $\pm$ 1.34	52.79 $\pm$ 1.33	52.99 $\pm$ 1.36	**55.29** $\pm$ 1.35
Fungi	42.12 $\pm$ 1.36	43.15 $\pm$ 1.36	41.14 $\pm$ 1.34	43.75 $\pm$ 1.31	41.74 $\pm$ 1.34	**45.24** $\pm$ 1.34
5-way 5-shot						
Bird	68.52 $\pm$ 0.79	70.91 $\pm$ 0.71	67.17 $\pm$ 0.74	69.25 $\pm$ 0.73	67.87 $\pm$ 0.74	**72.74** $\pm$ 1.40
Texture	44.56 $\pm$ 0.68	47.21 $\pm$ 0.64	43.41 $\pm$ 0.68	45.78 $\pm$ 0.68	45.49 $\pm$ 0.68	**49.15** $\pm$ 0.68
Aircraft	66.18 $\pm$ 0.71	68.13 $\pm$ 0.61	65.34 $\pm$ 0.70	67.15 $\pm$ 0.71	66.84 $\pm$ 0.70	**69.73** $\pm$ 0.70
Fungi	51.85 $\pm$ 0.85	56.17 $\pm$ 0.75	52.11 $\pm$ 0.83	54.35 $\pm$ 0.83	52.51 $\pm$ 0.81	**58.21** $\pm$ 0.79



(a) Bird (b) Texture (c) Aircraft (d) Fungi

Figure 1: Comparisons for MAML with EMO and other optimizers. EMO speeds up MAML training and outperforms the other optimizers for few-shot learning.

Comparison with the other optimizers To show the benefit of our episodic memory optimizer, we first compare EMO with other commonly used optimizers in the inner-loop stage of MAML. Learning curves for MAML using different optimizers are displayed in Figure 1. We also report the final specific results by comparing them with different optimizers in the Appendix C.1. In this experiment, EMO outperforms other optimizers by a considerable margin. Momentum and Adam have somewhat degraded performance compared to SGD, which means that these traditional optimizers cannot exploit past inaccurate gradients for few-shot learning. However, EMO can speed up training and improve performance since EMO acquires the ability to adaptively choose the most relevant task update rules for the test task.

Effect of inner-loop steps We make further analysis on the effectiveness of our optimization in fast adaptation by varying the number of update steps. Specifically, we compare the performance of MAML with and without EMO in Figure 2. We find that MAML with EMO achieves about 51.34% accuracy at step 0, which is more than 19.86% higher than MAML, which means that EMO needs only episodic memory to achieve competitive performance. Also, MAML with EMO can reach convergence very quickly (step 2 vs. step 5) and with a much higher performance than MAML. Although MAML without EMO already performs fast adaptation with 5 steps, MAML with EMO is even faster and better. This again demonstrates the benefit of EMO.

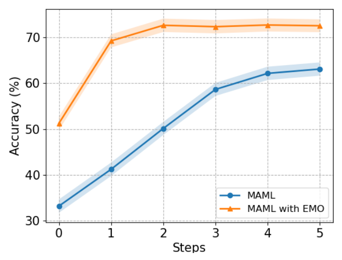


Figure 2: Effect of inner-loop steps. MAML with EMO both speeds up training and achieves better performance compared to original MAML.

Comparison with aggregation functions We also ablate the effect of EMO's aggregation function to generate the new gradients. We report the performance of MAML and Meta-SGD with EMO using different `Aggr` in Table 2, and the experiments for ANIL with EMO are proposed in Appendix C.2. The results show that the best-suited aggregation function is specific to the optimization-based meta-

Table 2: Effect of different aggregation functions on *Meta-Dataset-BTAF* under the 5-way 1-shot setting. The best-suited aggregation function for MAML is `Mean`, while the best-suited aggregation function for Meta-SGD is `Transformer`.

	MAML with EMO			Meta-SGD with EMO		
Dataset	`Sum`	`Mean`	`Transformer`	`Sum`	`Mean`	`Transformer`
Bird	54.35 ± 1.34	56.32 ± 1.33	55.91 ± 1.35	57.15 ± 1.31	57.03 ± 1.40	**58.95** ± 1.41
Texture	33.13 ± 1.45	34.75 ± 1.41	34.23 ± 1.40	34.93 ± 1.42	35.97 ± 1.41	**36.26** ± 1.33
Aircraft	52.53 ± 1.30	53.99 ± 1.33	53.15 ± 1.30	53.12 ± 1.27	54.01 ± 1.25	**55.29** ± 1.35
Fungi	44.07 ± 1.33	43.15 ± 1.36	**45.27** ± 1.35	43.49 ± 1.32	44.13 ± 1.31	45.24 ± 1.34

learning method into which episodic gradients are integrated. The best-suited aggregation function for MAML with EMO is the `Mean`, while the best-suited aggregation function for Meta-SGD with EMO is the `Transformer`. To ensure consistency of implementation on each dataset and for each model, we choose the `Mean` aggregation function for MAML with EMO and the `Transformer` aggregation function for Meta-SGD with EMO in the remaining experiments.

Comparison with memory controller To assess the effect of the memory controller, we compare three memory controllers: `FIFO-EM`, `CLOCK-EM`, `LRU-EM` on the *Meta-Dataset-BTAF* under the 5-way 1-shot setting. The experimental results for MAML with EMO are reported in Table 3, and results for Meta-SGD and ANIL with EMO are in the Appendix C.3. `FIFO-EM` achieves the worst performance compared to the other memory controllers since `FIFO-EM` may replace some crucial or commonly used memory, causing the test task to fail to find the precise memory to learn quickly. With `LRU-EM`, MAML with EMO leads to a small but consistent gain under all the datasets, as it replaces the memory that is not commonly used, and these memories can usually be seen as outliers. In Table 8, with `CLOCK-EM`, Meta-SGD with EMO achieves the better performance on the all datasets. `CLOCK-EM` is a more balanced algorithm in terms of performance and overhead, which is more suitable for methods that require large memory, such as Meta-SGD with EMO, which also requires additional storage of the gradient of the inner-loop learning rate. To ensure consistency of implementation on each dataset, we choose the `LRU-EM` function for MAML with EMO and ANIL with EMO, `CLOCK-EM` is used for Meta-SGD with EMO.

Table 3: Effect of memory controller. `LRU-EM` achieves better performance than alternatives.

	MAML with EMO		
Dataset	`FIFO-EM`	`CLOCK-EM`	`LRU-EM`
Bird	51.91 ± 1.35	54.01 ± 1.33	**56.32** ± 1.33
Texture	30.11 ± 1.40	32.14 ± 1.41	**34.75** ± 1.41
Aircraft	48.16 ± 1.40	50.91 ± 1.38	**53.99** ± 1.33
Fungi	41.17 ± 1.35	**43.97** ± 1.35	43.15 ± 1.36

Effect of task-memory size Task-memory size cannot be increased indefinitely. To study the effect of task-memory size on EMO, we conduct this ablation on *mini*ImageNet using MAML with EMO under the 5-way 1-shot and 5-way 5-shot settings. From Figure 3, we observe the performance increases along with the increase in task-memory size. This is expected since more significant memory provides more context information for building better memory. Naturally, the memory size has a greater impact in the 1-shot setting. In this case, the model updated from only one example might be insufficiently representative of the object class. Leveraging context information provided by the memory compensates for the limited number of samples. We adopt memory sizes 100 for 1-shot and 200 for 5-shot on each dataset.

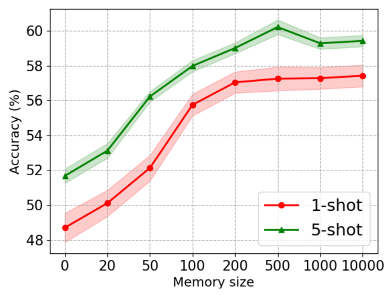


Figure 3: Effect of task-memory size.

Analysis of episodic memory In this experiment, we meta-train MAML and MAML with EMO on the Bird dataset and meta-test on *Meta-Dataset-BTAF*. Therefore the episodes saved in the memory

Table 4: Comparative results of different algorithms on *Meta-Dataset-BTAF* and *mini*ImageNet using a Conv-4 backbone under the 5-way 1-shot setting. Results under the 5-way 5-shot are provided in the Appendix C.5. The results of the *Meta-Dataset-BTAF* of other methods are provided by (Yao et al., 2019; Jiang et al., 2022). Equipping ARML with EMO results in top-performance.

Method	Bird	Texture	Aircraft	Fungi	*mini*ImageNet
ANIL (Raghu et al., 2019)	53.36 $\pm$ 1.42	31.91 $\pm$ 1.25	52.87 $\pm$ 1.34	42.30 $\pm$ 1.28	46.70 $\pm$ 0.40
MAML (Finn et al., 2017)	53.94 $\pm$ 1.45	31.66 $\pm$ 1.31	51.37 $\pm$ 1.38	42.12 $\pm$ 1.36	48.70 $\pm$ 1.84
BMG (Flennerhag et al., 2021)	54.12 $\pm$ 1.46	32.19 $\pm$ 1.21	52.09 $\pm$ 1.35	43.00 $\pm$ 1.37	52.97 $\pm$ 0.85
Meta-SGD (Li et al., 2017)	55.58 $\pm$ 1.43	32.38 $\pm$ 1.32	52.99 $\pm$ 1.36	41.74 $\pm$ 1.34	50.47 $\pm$ 1.87
MUSML (Jiang et al., 2022)	60.52 $\pm$ 0.33	**41.33** $\pm$ 1.30	54.69 $\pm$ 0.69	45.60 $\pm$ 0.43	-
HSML (Yao et al., 2019)	60.98 $\pm$ 1.50	35.01 $\pm$ 1.36	57.38 $\pm$ 1.40	44.02 $\pm$ 1.39	50.38 $\pm$ 1.85
TSA-MAML (Zhou et al., 2021)	61.37 $\pm$ 1.42	35.41 $\pm$ 1.39	58.78 $\pm$ 1.37	44.17 $\pm$ 1.25	48.44 $\pm$ 0.91
ARML (Yao et al., 2020)	62.33 $\pm$ 1.47	35.65 $\pm$ 1.40	58.56 $\pm$ 1.41	44.82 $\pm$ 1.38	50.42 $\pm$ 1.73
ARML with EMO	**64.31** $\pm$ 1.35	37.25 $\pm$ 1.43	**59.99** $\pm$ 1.35	**46.15** $\pm$ 1.38	**57.84** $\pm$ 0.93

are only from the Bird dataset. The experiments that meta-train on the other three datasets are provided in Appendix C.4. From Figure 4, there is no doubt that MAML with EMO achieves a better performance than MAML on Bird. Surprisingly, MAML with EMO also outperforms MAML on Aircraft. It might be that the two datasets have more similar shapes (wings), Bird memory can still help accelerate the Aircraft tasks' training. However, when the test task has significant distribution shifts with the training task, e.g., Texture, Fungi, the memory will not be helpful or harmful. We will explore in future work how to use episodic memory to address cross-domain few-shot challenges.

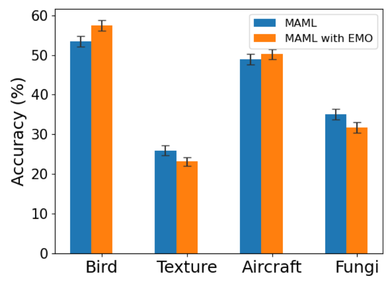


Figure 4: EMO is trained only on the Bird to show that EMO also holds semantic information. MAML with EMO achieves better performance on the same dataset (Bird) and the similar shapes dataset (Aircraft), while it is harmful on the test tasks that have significant distribution shifts (Texture and Fungi) from the training tasks.

Comparison with the state-of-the-art methods We first compare our method on *Meta-Dataset-BTAF* using a Conv-4 backbone under the 5-way 1-shot setting in Table 4. Both the results of the 5-way 5-shot and using ResNet-12 can be found in Appendix C.5. In this comparison, we apply ARML (Yao et al., 2020) with EMO to experiment since ARML is the current state-of-the-art algorithm based on optimization-based meta-learning. Our method achieves state-of-the-art performance on each dataset under the 5-way 1-shot setting. On Texture, our model surpasses the second best method, i.e., ARML (Yao et al., 2020), by 1.6%. The better performance confirms that EMO can find the most similar task to the test task and update the parameters so that it can converge faster and perform better. We also evaluate our method on traditional few-shot classification, in which the training and test datasets are from the same dataset. In this experiment, we also apply ARML (Li et al., 2017) with EMO experiment. Our EMO performs consistently better than the previous methods on the *mini*ImageNet. The results demonstrate that optimization-based meta-learning benefits from EMO for traditional few-shot learning.

5 Conclusions

In this paper, we propose episodic memory optimization (EMO) that retains the gradient history of past experienced tasks in external memory. EMO accumulates long-term, general learning processes knowledge of past tasks, allowing it to learn a new task quickly based on task similarity. EMO integrated with several optimization-based meta-learning methods accelerates the learning in all settings and datasets tested and improves their performance. We also prove that EMO with fixed-size memory converges under assumptions of strong convexity, regardless of which gradients are selected or how they are aggregated to form the update step. We conduct thorough ablation studies to demonstrate the effectiveness of the memory-augmented optimizer. Experiments on several few-shot learning datasets further substantiate the benefit of the episodic memory optimizer.

REFERENCES

Momin Abbas, Quan Xiao, Lisha Chen, Pin-Yu Chen, and Tianyi Chen. Sharp-maml: Sharpness-aware model-agnostic meta learning. In *ICML*, 2022.

Zeyuan Allen-Zhu and Yang Yuan. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *ICML*, pp. 1080–1089. PMLR, 2016.

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.

Mahmoud Assran and Michael Rabbat. On the convergence of nesterov's accelerated gradient method in stochastic settings. In *ICML*, 2020.

Sudarshan Babu, Pedro Savarese, and Michael Maire. Online meta-learning via learning with layer-distributed memory. *NeurIPS*, 34:14795–14808, 2021.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. In *IJCNN*, 1991.

Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, pp. 3606–3613, 2014.

Mohammad Mahdi Derakhshani, Xiantong Zhen, Ling Shao, and Cees Snoek. Kernel continual learning. In *ICML*, pp. 2621–2631. PMLR, 2021.

Yingjun Du, Xiantong Zhen, Ling Shao, and Cees GM Snoek. Hierarchical variational memory for few-shot learning across domains. In *ICLR*, 2022.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, pp. 9516–9527, 2018.

Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning. In *ICLR*, 2021.

FUNGI. 2018 fgcvx fungi classification challenge. https://www.kaggle.com/c/fungi-challenge-fgvc-2018, 2018.

Samuel J Gershman and Nathaniel D Daw. Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual review of psychology*, 68:101–128, 2017.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, pp. 4367–4375, 2018.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*, 2018.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *NeurIPS*, pp. 4005–4016, 2020.

Hao Hu, Jianing Ye, Guangxiang Zhu, Zhizhou Ren, and Chongjie Zhang. Generalizable episodic memory for deep reinforcement learning. In *ICML*, 2021.

Andhi Janapsatya, Aleksandar Ignjatović, Jorgen Peddersen, and Sri Parameswaran. Dueling clock: Adaptive cache replacement policy based on the clock algorithm. In *Design, Automation & Test in Europe Conference & Exhibition*, pp. 920–925, 2010.

Weisen Jiang, James Kwok, and Yu Zhang. Subspace learning for effective meta-learning. In *ICML*, pp. 10177–10194. PMLR, 2022.

Konstantinos Kalais and Sotirios Chatzis. Stochastic deep networks with linear competing units for model-agnostic meta-learning. In *ICML*, pp. 10586–10597. PMLR, 2022.

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

Andrew Lampinen, Stephanie Chan, Andrea Banino, and Felix Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *NeurIPS*, 34:28182–28195, 2021.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pp. 10657–10665, 2019.

Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, pp. 2927–2936, 2018.

Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *CVPR*, pp. 1–10, 2019.

Ke Li and Jitendra Malik. Learning to optimize. In *ICLR*, 2017.

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.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *NeurIPS*, 30, 2017.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Paul-Aymeric McRae, Prasanna Parthasarathi, Mahmoud Assran, and Sarath Chandar. Memory augmented optimizers for deep learning. In *ICLR*, 2022.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

Tsendsuren Munkhdalai and Aadam Trischler. Metalearning with hebbian fast weights. *arXiv preprint arXiv:1807.05076*, 2018.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, pp. 2554–2563, 2017.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, pp. 721–731, 2018.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

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.

Janarthanan Rajendran, Alexander Irpan, and Eric Jang. Meta-learning requires meta-augmentation. *NeurIPS*, 33:5705–5715, 2020.

Tiago Ramalho and Marta Garnelo. Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *ICLR*, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv*, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, pp. 1842–1850, 2016.

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 S Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pp. 4077–4087, 2017.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, pp. 403–412, 2019.

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 *CVPR*, pp. 1199–1208, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Sebastian Thrun and Lorien Pratt (eds.). *Learning to Learn*. Kluwer Academic Publishers, USA, 1998.

Endel Tulving. Episodic and semantic memory. *Neuropsychologia*, 1972.

Endel Tulving. Elements of episodic memory. *Neuropsychologia*, 1983.

Endel Tulving. Episodic memory: From mind to brain. *Annual Review of Psychology*, 53(1):1–25, 2002.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pp. 5998–6008, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *NeurIPS*, 29:3630–3638, 2016.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *ICML*, pp. 7045–7054. PMLR, 2019.

Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. *arXiv preprint arXiv:2001.00745*, 2020.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*, pp. 7343–7353, 2018.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *ICML*, pp. 7115–7123, 2019.

Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. Variational few-shot learning. In *ICCV*, pp. 1685–1694, 2019.

Xiantong Zhen, Yingjun Du, Huan Xiong, Qiang Qiu, Cees GM Snoek, and Ling Shao. Learning to learn variational semantic memory. In *NeurIPS*, 2020.

Pan Zhou, Yingtian Zou, Xiao-Tong Yuan, Jiashi Feng, Caiming Xiong, and Steven Hoi. Task similarity aware meta learning: Theory-inspired improvement on maml. In *UAI*, pp. 23–33. PMLR, 2021.

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. In *ICLR*, 2020.

Yingtian Zou, Fusheng Liu, and Qianxiao Li. Unraveling model-agnostic meta-learning via the adaptation learning rate. In *ICLR*, 2021.

A Implementation Details

We follow (Finn et al., 2017; Yao et al., 2019) by adopting the standard four-block convolutional layers as the feature extractor for our episodic memory optimizer and all baselines. We also conduct our experiments by ANIL (Raghu et al., 2019), which removes the inner-loop updates for the feature extractor network, and applies inner-loop adaptation only to the classifier during training and testing. For all experiments, we keep the outer-loop optimizer consistent with the traditional optimization-based meta-learning approaches, e.g., Adam (Kingma & Ba, 2015). Our code will be publicly released.

B Proof of Convergence

To analyze the convergence rate of the model, we first derive the upper bound for the expectation $\mathbb{E}\left\|\Delta\theta_{t+1}\right\|^2$ with respect to the independent random noises for all previous gradients $\{\epsilon_j\}_{j=1}^{t}$, where $\|\cdot\|$ is the spectral norm. We reformulate the aggregation process of our method as a linear multi-step system. Thus the gradient for the t-th iteration is $\texttt{aggr}(\mathbf{g}_t, \mathcal{V}_t) = \sum_{s=0}^{S-1} w_{t,s} g_{t-s}$, where S is the number of step in the system. By incorporating the aggregation process into the update rule Eq. (7) and subtracting θ^* from both sides, we obtain the recursive formulation about the difference $\Delta\theta_t$ as:

$$\Delta\theta_{t+1} = \Delta\theta_t - \alpha \sum_{s=0}^{S-1} w_{t,s} g_{t-s}. \tag{13}$$

In the paper, the gradient of each iteration is reformulated by adding its mean and the corresponding noise in Eq. (11). For clarity in the proof below, we define the average rate of the gradient changes from the t-th iteration of model parameters to the optimal as:

$$\mathcal{R}_t = \frac{\nabla f(\theta_t) - \nabla f(\theta^*)}{\Delta\theta_t} = \int_0^1 \nabla^2 f(\theta^* + u\Delta\theta_t) du. \tag{14}$$

With the assumptions about the objective function f, the average rate of gradient changes is also bounded between μ and L. By incorporating Eq. (14) into Eq. (11), we simplify the recursive formulation about the difference $\Delta\theta_t$ as:

$$\Delta\theta_{t+1} = \Delta\theta_t - \alpha \sum_{s=0}^{S-1} w_{t,s} \mathcal{R}_{t-s} \Delta\theta_{t-s} - \alpha \sum_{s=0}^{S-1} w_{t,s} \epsilon_{t-s}. \tag{15}$$

We take recursive formulations about $\{\Delta\theta_{t+1-s}\}_{s=0}^{S-1}$ together and get the matrix version of the the recursion below:

$$\begin{bmatrix} \Delta\theta_{t+1} \\ \Delta\theta_t \\ \vdots \\ \Delta\theta_{t-S} \end{bmatrix} = A_t \begin{bmatrix} \Delta\theta_t \\ \Delta\theta_{t-1} \\ \vdots \\ \Delta\theta_{t-S+1} \end{bmatrix} + \begin{bmatrix} -\alpha \sum_{s=0}^{S-1} w_{t,s}\epsilon_{t-s} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$\text{where } A_t = \begin{bmatrix} I - \alpha w_{t,0}\mathcal{R}_t & -\alpha w_{t,1}\mathcal{R}_{t-1} & \cdots & -\alpha w_{t,S-2}\mathcal{R}_{t-S+2} & -\alpha w_{t,S-1}\mathcal{R}_{t-S+1} \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}. \tag{16}$$

Note that A_t is the system matrix at the t-th iteration. By unrolling the recursion below, the upper bound of the expectation $\mathbb{E}\left\|\Delta\theta_{t+1}\right\|^2$ can be derived :

$$
\begin{aligned}
\mathbb{E}\left\|\Delta\theta_{t+1}\right\|^2 &\le \mathbb{E}_{\epsilon_t,\cdots,\epsilon_1}\left\|\begin{bmatrix}\Delta\theta_{t+1}\\ \Delta\theta_t\\ \vdots\\ \Delta\theta_{t-S}\end{bmatrix}\right\|^2\\
&= \mathbb{E}_{\epsilon_t,\cdots,\epsilon_1}\left\|A_t\begin{bmatrix}\Delta\theta_t\\ \Delta\theta_{t-1}\\ \vdots\\ \Delta\theta_{t-S+1}\end{bmatrix} + \begin{bmatrix}-\alpha\sum_{s=0}^{S-1} w_{t,s}\epsilon_{t-s}\\ 0\\ \vdots\\ 0\end{bmatrix}\right\|^2\\
&= \left\|A_t\right\|^2\mathbb{E}_{\epsilon_{t-1},\cdots,\epsilon_1}\left\|\begin{bmatrix}\Delta\theta_t\\ \Delta\theta_{t-1}\\ \vdots\\ \Delta\theta_{t-S+1}\end{bmatrix}\right\|^2 + \alpha^2\sum_{s=0}^{S} w_{t,s}^2\mathbb{E}_{\epsilon_{t-s}}\left\|\epsilon_{t-s}\right\|^2 \qquad (17)\\
&\le \left\|A_t\right\|^2\mathbb{E}_{\epsilon_{t-1},\cdots,\epsilon_1}\left\|\begin{bmatrix}\Delta\theta_t\\ \Delta\theta_{t-1}\\ \vdots\\ \Delta\theta_{t-S+1}\end{bmatrix}\right\|^2 + \alpha^2 S\sigma^2\\
&\cdots\\
&\le \prod_{j=1}^{t}\left\|A_j\right\|^2\left\|\Delta\theta_1\right\|^2 + \alpha^2 S\sigma^2\sum_{j=1}^{S}(\left\|A_t\right\|^2\cdots\left\|A_{j+1}\right\|^2).
\end{aligned}
$$

According to the definition of the spectral norm and the properties of block matrix (Polyak, 1964; Assran & Rabbat, 2020; McRae et al., 2022), we get the upper bound of the spectral norm below:

$$
\|A_t\| \le \lambda_t(\widehat{A}_t^\top\widehat{A}_t),
$$

$$
\text{where } \widehat{A}_t = \begin{bmatrix} 1-\alpha w_{t,0}\tau_t & -\alpha w_{t,1}\tau_{t-1} & \cdots & -\alpha w_{t,S-2}\tau_{t-S+2} & -\alpha w_{t,S-1}\tau_{t-S+1}\\ 1 & 0 & \cdots & 0 & 0\\ 0 & 1 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \qquad (18)
$$

Note that $\lambda_t(\widehat{A}_t^\top\widehat{A}_t)$ is the square root of the largest eigenvalue of the matrix $\widehat{A}_t^\top\widehat{A}_t$. The matrix $\widehat{A}_t \in \mathbb{R}^{S\times S}$ has bounded hyperparameters: $\tau_t \in [\mu, L]$ and $w_t \in [0,1]$. We introduce $\lambda_{\max}$ as the upper bound for all λ_t corresponding to all system matrices. Since the learning rate is chosen sufficiently small such that $\lambda_{\max} < 1$, we further simplify Eq. (17) below:

$$
\mathbb{E}\left\|\Delta\theta_{t+1}\right\|^2 \le \lambda_{\max}^{2t}\left\|\Delta\theta_1\right\|^2 + \frac{\alpha^2\sigma^2 S}{1-\lambda_{\max}^2}. \qquad (19)
$$

Recall that $f(\cdot)$ is assumed to be L-smooth, we get the convergence rate of our model as

$$
f(\theta_{t+1}) - f(\theta^*) \le \frac{L}{2}(\lambda_{\max}^{2t}\left\|\Delta\theta_1\right\|^2 + \frac{\alpha^2\sigma^2 S}{1-\lambda_{\max}^2}). \qquad (20)
$$

C More results

C.1 Comparison with other optimizers

To show the benefit of the episodic memory optimizer, we compare MAML (Finn et al., 2017), Meta-SGD (Li et al., 2017), and ANIL (Raghu et al., 2019) with their EMO variants, where each

Table 5: Comparison with other optimizers on *Meta-Dataset-BTAF* under the 5-way 1-shot setting. EMO achieves better performance compared to other optimizers on all datasets.

	MAML			
Dataset	w/ SGD	w/ Momentum	w/ Adam	w/ EMO
Bird	**53.94** ± 1.45	52.98 ± 1.42	52.55 ± 1.41	**56.32** ± 1.33
Texture	31.66 ± 1.31	31.38 ± 1.31	30.95 ± 1.34	**34.75** ± 1.41
Aircraft	**51.37** ± 1.38	51.09 ± 1.35	50.15 ± 1.33	**53.99** ± 1.33
Fungi	**42.12** ± 1.36	**41.54** ± 1.35	**41.04** ± 1.31	**43.15** ± 1.36

variant uses EMO as the inner-loop optimizer. Table 5 shows each method with EMO achieves better performance by a large margin than the original methods on four different datasets. More importantly, the most challenging, which has the largest domain gap Texture, delivers 34.75%, surpassing the Meta-SGD by 2.09%. We attribute the improvements to our model's ability to leverage the episodic memory to adjust the model parameters, allowing the model to update the test task model using the most training task-like update, and thus leading to improvements over original models.

C.2 Effect of different aggregation functions

We also give the ANIL with EMO for ablating the effect of EMO's aggregation function used to compute the new gradients. We report the performance of ANIL with EMO using different aggregation functions in Table 6. The best-suited aggregation function for ANIL with EMO is the `Transformer`. To ensure consistency of implementation on each dataset, we choose the `Transformer` aggregation function for ANIL with EMO.

Table 6: Effect of ANIL with different aggregation functions. `Mean` achieves better performance than alternatives.

	ANIL with EMO		
Dataset	`sum`	`Mean`	`Transformer`
Bird	54.91 ± 1.33	**55.18** ± 1.34	54.78 ± 1.33
Texture	32.71 ± 1.30	33.14 ± 1.40	**33.15** ± 1.41
Aircraft	**53.16** ± 1.40	52.11 ± 1.38	52.79 ± 1.33
Fungi	43.17 ± 1.34	43.07 ± 1.31	**43.75** ± 1.36

C.3 Effect of memory controller

We further assess the effect of the memory controller with ANIL with EMO and Meta-SGD with EMO in Table 7. With `CLOCK-EM`, Meta-SGD with EMO achieves better performance on all datasets, while ANIL with EMO leads to a small but consistent gain under all the datasets with `LRU-EM`. To ensure consistency of implementation on each dataset, we choose the `LRU-EM` function for ANIL with EMO, and `CLOCK-EM` is used for Meta-SGD with EMO.

C.4 Analysis of episodic memory

In this section, we further analysis of our proposed episodic memory with the other three datasets. In this experiment, we meta-train MAML and MAML with EMO on the Texture, Aircraft, and Fungi datasets, respectively, and meta-test on *Meta-Dataset-BTAF*. Therefore the episodes saved in the memory are from the Texture, Aircraft, and Fungi, respectively. The results are shown in Figure 5. Consistent with the results in the Figure 4, MAML with EMO has a significant performance improvement when the meta-training dataset is the same as the meta-test dataset. Interestingly, the memory of Aircraft can also help Bird to achieve better performance in Figure 5 (b). Similarly, when the test task has large distribution shifts with the training task, the memory will not be useful or even harmful.

Table 7: Effect of ANIL with different memory controllers. `LRU-EM` achieves better performance than alternatives.

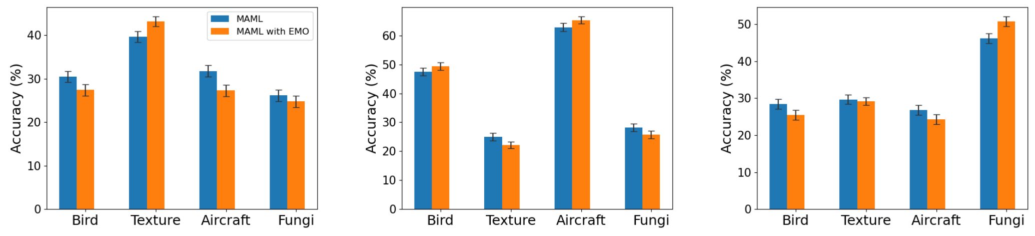
	ANIL with EMO		
Dataset	`FIFO-EM`	`CLOCK-EM`	`LRU-EM`
Bird	50.11 ± 1.31	53.91 ± 1.34	**54.78** ± 1.43
Texture	29.11 ± 1.41	32.94 ± 1.40	**33.15** ± 1.31
Aircraft	47.96 ± 1.40	**53.91** ± 1.35	52.79 ± 1.33
Fungi	40.97 ± 1.35	43.17 ± 1.35	**43.75** ± 1.31

Table 8: Effect of Meta-SGD with different memory controllers. `LRU-EM` achieves better performance than alternatives.

	Meta-SGD with EMO		
Dataset	`FIFO-EM`	`CLOCK-EM`	`LRU-EM`
Bird	53.05 ± 1.34	**58.95** ± 1.41	57.31 ± 1.34
Texture	32.13 ± 1.41	**36.26** ± 1.33	35.95 ± 1.41
Aircraft	49.16 ± 1.41	55.21 ± 1.35	**56.19** ± 1.34
Fungi	41.61 ± 1.34	**45.24** ± 1.35	44.75 ± 1.36

C.5 Comparison with the state-of-the-art on few-shot learning datasets

We further conduct experiments on the *Meta-Dataset-BTAF* and *mini*ImageNet under the 5-way 5-shot setting in Table 9. We also give the comparative results for few-shot learning on *mini*ImageNet and *tired*ImageNet using a ResNet-12 back in the Table 10. In these comparison, we apply ARML Yao et al. (2020) with EMO to do the experiment. Our method achieves state-of-the-art performance on all benchmarks under the 5-way 5-shot setting.



(a) Trained only Texture (b) Trained only Aircraft (c) Trained only Fungi

Figure 5: Analysis of episodic memory.

Table 9: Comparative results of different algorithms on the *Meta-Dataset-BTAF* using a Conv-4 backbone under the 5-way 5-shot setting. The results of other methods are provided by (Yao et al., 2019; Jiang et al., 2022). Equipping ARML with EMO makes it a consistent top-performer.

Method	Bird	Texture	Aircraft	Fungi	*mini*ImageNet
MAML (Finn et al., 2017)	68.52 ± 0.79	44.56 ± 0.68	66.18 ± 0.71	51.85 ± 0.85	63.11 ± 0.92
Meta-SGD (Li et al., 2017)	67.87 ± 0.74	45.49 ± 0.68	66.84 ± 0.70	52.51 ± 0.81	64.03 ± 0.94
HSML (Yao et al., 2019)	71.68 ± 0.73	48.08 ± 0.69	73.49 ± 0.68	56.32 ± 0.80	65.91 ± 0.95
ARML (Yao et al., 2020)	73.34 ± 0.70	49.67 ± 0.67	74.88 ± 0.64	57.55 ± 0.82	66.87 ± 0.93
TSA-MAML (Zhou et al., 2021)	72.31 ± 0.71	49.50 ± 0.68	74.01 ± 0.70	56.95 ± 0.80	65.52 ± 0.92
ANIL (Raghu et al., 2019)	70.67± 0.72	44.67 ± 0.95	66.05 ± 1.07	52.89 ± 0.30	61.50 ± 0.92
BMG (Flennerhag et al., 2021)	71.56 ± 0.76	49.44 ± 0.73	66.83 ± 0.79	52.56 ± 0.89	66.73 ± 0.91
MUSML (Jiang et al., 2022)	76.69 ± 0.72	52.41 ± 0.75	77.76 ± 0.82	57.74 ± 0.81	65.12 ± 1.48
ARML with EMO	**77.17** ± 0.65	**53.25** ± 0.68	**77.83** ± 0.63	**59.15** ± 0.79	**71.05** ± 0.91

Table 10: Comparative results for few-shot learning on *mini*Imagenet and *tiered*Imagenet using a ResNet-12 backbone. ARML with EMO can also improve performance for traditional few-shot learning.

	***mini*Imagenet 5-way**		***tiered*Imagenet 5-way**	
Method	**1-shot**	**5-shot**	**1-shot**	**5-shot**
SNAIL (Mishra et al., 2018)	55.71 ± 0.99	68.88 ± 0.92	-	-
Dynamic FS (Gidaris & Komodakis, 2018)	55.45 ± 0.89	70.13 ± 0.68	-	-
TADAM (Oreshkin et al., 2018)	58.50 ± 0.30	76.70 ± 0.30	-	-
MTL (Sun et al., 2019)	61.20 ± 1.80	75.50 ± 0.80	-	-
VariationalFSL (Zhang et al., 2019)	61.23 ± 0.26	77.69 ± 0.17	-	-
TapNet (Yoon et al., 2019)	61.65 ± 0.15	76.36 ± 0.10	63.08 ± 0.15	80.26 ± 0.12
MetaOptNet (Lee et al., 2019)	62.64 ± 0.61	78.63 ± 0.46	65.81 ± 0.74	81.75 ± 0.53
CTM (Li et al., 2019)	62.05 ± 0.55	78.63 ± 0.06	64.78 ± 0.11	81.05 ± 0.52
CAN (Hou et al., 2020)	63.85 ± 0.48	79.44 ± 0.34	69.89 ± 0.51	84.23 ± 0.37
HVM (Du et al., 2022)	67.83 ± 0.57	83.88 ± 0.51	73.67 ± 0.71	88.05 ± 0.44
ARML with EMO	**69.15** ± 0.34	**84.13** ± 0.25	**75.17** ± 0.35	**89.05** ± 0.25