Fast and Generalized Adaptation for Few-Shot Learning

Liang Song, Jinlu Liu, Yongqiang Qin
AInnovation Technology Co., Ltd.
songliang, liujinlu, qinyongqiang@ainnovation.com

Abstract

The ability of fast generalizing to novel tasks from a few examples is critical in dealing with few-shot learning problems. However, deep learning models severely suffer from overfitting in extreme low data regime. In this paper, we propose Adaptable Cosine Classifier (ACC) and Amphibian to achieve fast and generalized adaptation for few-shot learning. The ACC realizes the flexible retraining of a deep network on small data without overfitting. The Amphibian learns a good weight initialization in the parameter space where optimal solutions for the tasks of the same class cluster tightly. It enables rapid adaptation to novel tasks with few gradient updates. We conduct comprehensive experiments on four few-shot datasets and achieve state-of-the-art performance in all cases. Notably, we achieve the accuracy of 87.75% on 5-shot miniImageNet which approximately outperforms existing methods by 10%. We also conduct experiment on cross-domain few-shot tasks and provide the best results.

1. Introduction

Deep Neural Networks have shown great power in visual learning tasks such as image classification [11, 15, 29], object detection [9, 24, 10], and semantic segmentation [18, 26, 2]. However, training these models requires a large amount of labeled data. In many cases, the model performance sharply drops if the labeled data is scarce. Recent researches pay attention to rapidly learn a model in a data-efficient way. The goal is enabling a model to fast adapt to a new task without the need for hundreds of training data. In the low-data regime, fast adaptation to new tasks and avoiding the overfitting problem are challenging. These issues motivate the study of few-shot learning.

Few-shot learning [34, 6, 7] aims to efficiently learn a model that can recognize novel classes when the training examples are extremely limited. It is commonly formalized in a meta-learning way where a task is called N-way K-shot task if it consists of N classes with K labeled examples per class. In few-shot learning, we learn a model on different tasks from a large labeled dataset of base classes and aim to fast adapt the model to unseen tasks of novel classes. Most existing methods show limited performance when facing two major challenges in few-shot learning: rapid adaptation and excellent generalization. Various methods are explored to attack the challenges which can be divided into two categories: metric learning based methods [7, 34] and meta learning based methods [6, 20]. 1) Metric learning based methods target to learn to compare target examples and few labeled examples by a distance metric in the embedding space. These methods usually have weak generalization ability. Without retraining, the models trained on the base classes poorly generalize to the novel classes. While with retraining, the models are prone to overfit on the few labeled data. Even so, some metric based learning approaches have an advantage of learning discriminative features. 2) Meta learning based methods aim to learn a basic model on abundant training tasks and the model can quickly converge on new tasks with a few gradient updates or fine-tuning steps. A common solution is to find a good initialization but some methods learn a biased initialization which prevents the model from generalizing well to unseen tasks.

In this paper, our goal is to achieve fast and generalized adaptation for few-shot learning. As shown in Figure 2, our method consists of two stages: pre-training and adaptation. We propose an Adaptable Cosine Classifier for discriminative feature learning. To further improve generalization, we propose a meta learning method called Amphibian which ensures the rapid and generalized adaptation. The Adaptable Cosine Classifier consists of a feature extractor...
and a cosine classifier. It is trained in the first stage to learn a feature embedding space where examples in the same class are close to each other and far away from examples in other different classes. We train the network on the supervised classification task in the first stage. In the second stage, we just adapt the feature extractor on the novel classes. And we use the mean vectors of labeled examples to parametrize the cosine classifier since retraining the classifier on few examples takes risks of overfitting.

To better train the model, we propose a meta learning method called Amphibian which finds a good weight initialization for fast and generalized adaptation. An ideal weight initialization enables the model to learn from a task through one gradient update and achieve the best results on new tasks. To find a good weight initialization as much as possible, we aim to maximize the performance on a new task through one or few gradient updates in current task during the training process. We train the Adaptable Cosine Classifier using Amphibian strategy in the first stage. Thus, we obtain a good weight initialization from the base classes which realizes rapid adaptation and convergence on the novel classes in the second stage.

In a word, our method achieves discriminative feature learning and generalization, which takes advantages of metric learning based methods and meta learning based methods respectively. Our contributions are threefold:

1. We propose the Adaptable Cosine Classifier which has great generalization ability surpassing the existing metric learning based methods. Adaptable Cosine Classifier is flexible to be retrained and can fast adapt to novel tasks.

2. We propose a meta learning method called Amphibian to train a good weight initialization such that it enables rapid and generalized adaptation to new tasks with a small number of gradient updates.

3. Our experiments show that our method achieves the state-of-the-art and outperforms existing methods by 4%-10% on two few-shot benchmarks: miniImageNet and tieredImageNet. We also evaluate our method in cross-domain few-shot classification and produce the superior results on miniImageNet → CUB-200-2011 tasks.

2. Related Work

Many methods have been published to solve the few-shot classification problem [7, 16, 6, 20, 30, 33, 34, 28, 32]. In this section, we briefly introduce some approaches relevant to our work.

Metric learning based approach

Metric learning based approaches [34, 30, 33, 7] learn a projection function that can map examples from the image space to the feature space. And the features preserve the class neighborhood structure in the feature space so that we can recognize them easily. Instead of using a fully-connected layer, MatchingNet [34] and ProtoNet [30] use the nearest-neighbor method in the feature space with the Euclidean metric. Another commonly used distance metric is the cosine similarity [7]. There are also two commonly used methods to calculate the prototypes, using learned prototypes and using the support set to calculate prototypes in each task. Unlike the above methods, RelationNet [33] learns a deep non-linear distance metric to replace Euclidean metric or cosine similarity. By this method, it aims to learn the best metric that can adapt to different tasks. All the above methods are based on a fundamental hypothesis that the learned feature extractor can be generalized to novel classes. But there is still a significant performance gap between the base classes and the novel classes. It indicates that the mapping function can not fit in the novel classes well. So we propose the Adaptable Cosine Classifier method that can adapt the metric learning based model to novel tasks to eliminate the gap.
Meta learning based approach

Meta learning based approaches [28, 6, 20, 32] typically involve two phases: pre-training and adapting. Due to the particularity of the few-shot classification, the key is how to perform generalized adaptation in the adapting stage. Ravi et al. [28] concentrates on finding an optimizer that is better than SGD [25]. The similarity between gradient descent methods and long short-term memory [13] inspires them to propose the LSTM-based network. MAML [6] is another typical meta learning based method. It aims to learn a good weight initialization which is close to all tasks. So that with a limited number of labeled examples the model can find the task specific optimal in one gradient update step. Alex Nichol et al. propose Reptile [20] which can be treated as extended MAML with $k$ gradient update steps. Furthermore, they theoretically analyze the reason why the MAML liked method works. MTL [32] focuses on preventing meta overfitting by reducing the number of learnable parameters. They propose Scale-and-Shift parameters to adapt to novel tasks without updating all parameters. In this paper, we propose Amphibian to learn a good weight initialization from which we can perform better generalized adaptation in limited examples.

3. Methodology

In this section, we first give the problem formulation in section 3.1. Secondly, we introduce in section 3.2 the proposed approach to adapt a metric based method to novel tasks. Then we discuss how to train an initialization of the feature extractor by our proposed Amphibian that can fast adapt to all tasks in section 3.3. Finally, we provide a theoretical explanation of why Amphibian works in section 3.4.

3.1. Problem Formulation

Firstly, we introduce a generic notion of a learning task. Formally, each task $\tau = \{D_s, D_q, C_{task}, L(D_q|\phi)\}$ involves two sets: support set $D_s$ and query set $D_q$ respectively. The examples of $D_s$ and $D_q$ both belong to classes $C_{task}$. The support set $D_s$ consists of $N \times K$ examples ($N$ is the number of classes $C_{task}$, and $K$ is the number of examples per class) and corresponding labels. And the number of labeled examples $K$ is a small number like 1 or 5. The query set $D_q$ involves some unlabeled examples belong to $C_{task}$. The objective is using the limited labeled support set to learn a model $F_{\phi}$ that can recognize all the examples from the query set $D_q$. It can be formulated as minimizing the cost function $L(D_q|\phi)$. Such a task is a so-called $N$-way $K$-shot few-shot classification task.

In the few-shot classification problem, we are given two datasets: base set $D_{base}$ and novel set $D_{novel}$ respectively. The base set is composed of examples $x$ and the corresponding labels $y$, $D_{base} = \{(x_1, y_1), ..., (x_n, y_n)\}$. All the examples from the base set belong to class $C_{base}$. The novel set is similar with the base set $D_{novel} = \{(x'_1, y'_1), ..., (x'_n, y'_n)\}$, and all the examples belong to class $C_{novel}$. And $C_{novel}$ is disjoint with $C_{base}$. The base set is available for us to learn a good model. And evaluate the model on the novel set with few-shot tasks. The objective is:

$$\min_{\phi} E_{\tau}[L(D_q|A(D_s, \phi))]$$

where $A(\cdot)$ denotes the adaptation procedure. And $A(D_s, \phi)$ denotes the updated parameters on $D_s$.

3.2. Adaptable Cosine Classifier

In general, there is a feature extractor $F_{\phi}(\cdot)$ and a classifier $Z(\cdot; W)$ in a deep neural network for classification. $\phi$ is the parameter of the feature extractor $F_{\phi}$, and $W$ is the classification weight of the classifier $Z(\cdot; W)$. The feature extractor $F_{\phi}$ maps the data $x$ from the image space to the embedding space. Then the classifier $Z(\cdot; W)$ uses the feature embedding $F_{\phi}(x)$ to estimate the classification scores.

Since there are only few training examples for novel classes in few-shot classification, learning a linear classifier with good generalization ability is very difficult. In order to overcome this critical problem, the Cosine Classifier (CC) is proposed [7] which uses cosine similarity instead of dot product when computing classification scores. The cosine similarity operator has an advantage in preserving the class neighborhood structure in the embedding space. The CC can be formalized as:

$$Z^k(F_{\phi}(x); W) = \frac{e^{\gamma \cos(F_{\phi}(x), W_k)}}{\sum_{i=1}^{||C||} e^{\gamma \cos(F_{\phi}(x), W_i)}}$$

where $Z^k(\cdot)$ denotes the classification score of the $k$-th class, $W_i$ is the $i$-th classification weight, $\gamma$ denotes the scale of the softmax operator, and $||C||$ is the number of classes.

At the pre-training stage, we train the model on the examples from base classes $C_{base}$ and the objective is to minimize the negative log-likelihood loss:

$$L(x|\phi, W_b) = E_{x,y \in D_{base}} \left[-logZ^y(F_{\phi}(x); W_b)\right]$$

where $W_b$ denotes the classification weight vectors of the base categories and $y$ is the true label.

Since the feature extractor $F_{\phi}$ is trained only on the base classes, it can not extract good features of the novel classes without adaptation. However, it takes a risk of overfitting to retrain the CC on the few examples to learn classification weights of the novel classes. To overcome the difficulty,
we propose Adaptable Cosine Classifier (ACC) to adapt the feature extractor to new tasks. As discussed before, the critical challenge for few-shot classification problem is to learn the best classification weight of the novel classes without overfitting on the few training examples. So we give the definition of the best classification weight:

$$\min W_j \sum_i Distance(W_j, F_\phi(x_i))$$ (4)

where $W_j$ is the classification weight of the $j$-th class, and $x_i$ is the example belonging to class $C_j$. To be consistent with the cosine operator used at the pre-training stage, we adopt $-\cos(\cdot, \cdot)$ as the distance metric in Equation 4. The optimization of Equation 4 has the closed form solution:

$$W_j = \frac{\sum F_\phi(x_i) / K_j}{\|\sum F_\phi(x_i) / K_j\|_2}$$ (5)

where $K_j$ denotes the number of examples belonging to category $C_j$.

At the inference stage, we adapt our feature extractor to the novel classes and evaluate on the query set. Since we do not have the true distribution of novel classes, we use the support set to estimate the true distribution. The objective is to minimize the negative log-likelihood loss:

$$L(x|\phi, W_n) = \mathbb{E}_{x, y \in D_{support}} [-\log P_y(F_\phi(x); W_n)]$$ (6)

where $W_n$ is calculated directly from Equation 6.

In general, to fine-tune CC we need to re-initialize the classification weights and retrain the model from scratch. However, the model tend to overfitting with the limited training data. By our proposed ACC, we compute the best classification weights directly. It gives the model the ability of fast and generalized adaptation to novel tasks. The adapted feature extractor can extract more discriminative features which are conducive to classification. As expected, experimental results also indicate that the adaptation can bring us significant performance improvement.

### 3.3. Amphibian

To further improve the ability of adaptation, we propose a simple yet effective method named Amphibian. Like MAML [6] and Reptile [20], Amphibian learns an initialization for the parameters of a model. From this initialization, the model can perform fast and generalized adaptation to different tasks.

When adapting to a new task $\tau_s$, we are given a labeled support set $D_s$, and an unlabeled query set $D_q$. Examples of both sets are sampled from $C_{task}$. Our objective is to update the initialization $\phi$ of the model one step using the support set $D_s$ then the updated model is capable of recognize the examples from the query set $D_q$. This objective can be formalized as:

$$\min_{\phi} E_{x \in D_q} [L(x|\phi')]$$ (7)

where $\phi'$ denotes the updated parameters using one gradient descent update on task $\tau_s$, $\phi' = \phi - \beta \nabla_\phi L(\tau_s|\phi)$. $\beta$ is a hyper-parameter. Since the support set $D_s$ and the query set $D_q$ is independent. In order to minimize the expectation of the loss on $D_q$, we need to minimize the loss on all examples belonging to $C_{task}$:

$$\min_{\phi} E_{x \in D_q} [L(x|\phi')] = \min_{\phi} E_{(x,y) \in C_{task}} [L(x|\phi')]$$ (8)

Equation 8 can be further formalized as:

$$\min_{\phi} E_{x \in D_q} [L(x|\phi')] = \min_{\phi} E_{\tau_i \in T_{task}} [L(\tau_i|\phi')]$$ (9)

where $T_{task}$ denotes all the possible tasks whose examples belong to $C_{task}$, and $\tau_i$ denotes one task sampled from $T_{task}$.

Meanwhile, we aim to perform well in the current task $\tau_s$ so that the model can extract task-independent features. The integral objective can be formalized as:

$$\min_{\phi} E_{\tau_i \in T_{task}} [L(\tau_i, \phi')] + \eta L(\tau_s|\phi)$$ (10)

where $\eta$ is a hyper-parameter, controlling the weight of the above two objectives. Since we can not obtain all the tasks, we use one task $\tau_i$ to estimate $E_{\tau \in T_{task}} [L(\tau, \phi')]$. Thus, the gradient update process is:

$$\phi \leftarrow \phi - \alpha \nabla_\phi (L(\tau_i|\phi') + \eta L(\tau_s|\phi))$$ (11)

where $\alpha$ is a hyper-parameter. Note that the cost function is computed using the updated parameters $\phi'$, yet the training parameters are $\phi$. So optimizing parameters $\phi$ requires the second order gradient. Higher order gradient requires

---

**Algorithm 1 Amphibian**

**Require:** $\alpha, \beta$: step size hyper-parameters  
Initialize parameters $\phi$ of a model  
for iteration $= 1, 2, \ldots$ do  
sample $N$ categories $C_{task}$ from $C_{base}$  
Initialize $\theta_0 = \phi$  
for $i = 1, 2, \ldots, m$ do  
sample $K$ examples per category from $D_{base}$  
the $N \times K$ examples compose a task $\tau_i$  
Compute $L(\tau_i; \theta_{i-1})$  
Update $\theta_i \leftarrow \theta_{i-1} - \beta \nabla L(\tau_i; \theta_{i-1})$  
end for  
Update $\phi \leftarrow \phi + \alpha (\theta_m - \phi)$  
end for
is the update procedure of parameter $\phi$ on the $L$ loops called inner loop and outer loop respectively. The parameters belonging to the same categories as an example. In one outer step $\tau_2$, the training procedure can be formalized as:

$$\phi \leftarrow \phi - \alpha (\nabla_{\phi'} L(\tau_1|\phi') + \eta \nabla_{\phi} L(\tau_a|\phi)) \quad (12)$$

Furthermore, we can generalize the above optimization procedure to $m$ tasks. Given $m$ tasks $[\tau_1, \tau_2, ..., \tau_m]$ belonging to the same categories $C_{task}$, the training procedure can be formalized as:

$$\phi \leftarrow \phi - \alpha \sum_{i=1}^{m} \eta_i \nabla_{\theta_i} L(\tau_i|\theta_i) \quad (13)$$

$$\theta_i \leftarrow \theta_{i-1} - \beta \nabla L(\tau_i|\theta_{i-1}) \quad (14)$$

$\theta_0$ is initialized from $\phi$, and $\theta_i$ denotes the parameters after the $i$-th inner update.

The integral optimization procedure involves two nested loops called inner loop and outer loop respectively. The inner loop in Equation 14 is the update procedure of parameters $\theta$ with inner loop learning rate $\beta$, and all the tasks $\tau_i$ belong to $T_{task}$. The outer loop is composed of multiple inner loops with different classes $C_{task}$. The parameter $\phi$ is updated once in each outer step as shown in Equation 13. And $\alpha$ denotes the outer loop learning rate. In pre-training stage, we train the feature extractor in Algorithm 1 on the base set $D_{base}$.

### 3.4. Theoretical Analysis

We first use Taylor Expansion [20] to approximate the optimization process of MAML [6], Reptile [20], and Amphibian. To simplify the problem, we take two inner steps as an example. In one outer step $T_{task}$, we sample two tasks: $\tau_1$, $\tau_2$ belong to the same categories: $C_{task}$. Let $L(\tau|\phi)$ denote the cost function on task $\tau$ with parameter $\phi$, and $f_\phi$ denotes the feature extractor. We define $L_\phi(\tau|\phi') = \frac{\partial}{\partial \phi} L(\tau|\phi')$ and $L_\phi''(\tau|\phi') = \frac{\partial^2}{\partial \phi^2} L(\tau|\phi')$.

In MAML, we use the first order approximate version. The gradients of MAML and Reptile can be formalized as [20]:

$$g_{MAML} = L_\phi'(\tau_1|\phi) - \beta L_\phi''(\tau_1|\phi)L_\phi'(\tau_1|\phi) \quad (15)$$

$$g_{Reptile} = 2L_\phi'(\tau_1|\phi) - \beta L_\phi''(\tau_1|\phi)L_\phi'(\tau_1|\phi) \quad (16)$$

The derivation of our Amphibian is provided in appendix and the final formulation is:

$$g_{Amphibian} = 2L_\phi'(\tau_1|\phi) - \beta L_\phi''(\tau_2|\phi)L_\phi'(\tau_1|\phi) \quad (17)$$

The difference between Amphibian and Reptile is reflected in the second order term.

We further analyze them in the parameter space to show the difference of Reptile and Amphibian. When adapting, our objective is to update the model one step using the seen task $\tau_a$, and perform well in all the unseen tasks $\tau_a$ belonging to the same classes. The objective can be formalized as:

$$\min_{\phi} E[L(\tau_a|\phi')] \quad (18)$$

where $\phi'$ denotes the updated parameter, $\phi' = \phi - \beta L_\phi'(\tau_a)$. Equation 18 can be decomposed into two parts:

$$\min_{\phi} E[L(\tau_a|\phi')] = \min_{\phi} E[(L(\tau_a|\phi') - L(\tau_a|\phi)) + L(\tau_a|\phi)] \quad (19)$$

We can estimate the first term by Taylor Expansion:

$$L(\tau_a|\phi') = L(\tau_a|\phi) + L_\phi'(\tau_a|\phi)(\phi' - \phi) + O(\delta^2) \quad (20)$$

Substituting Equation 20 into Equation 19, we get:

$$\min_{\phi} E[L_\phi'(\tau_a|\phi)(\phi' - \phi) + L(\tau_a|\phi)] \quad (21)$$

Considering $\phi' = \phi - \beta L_\phi'(\tau_a)$, we get:

$$\min_{\phi} E[-\beta L_\phi'(\tau_a|\phi)L_\phi'(\tau_a|\phi) + L(\tau_a|\phi)] \quad (22)$$

The first term can be further simplified as:

$$-\eta \cos(L_\phi'(\tau_a|\phi), L_\phi'(\tau_a|\phi)) \quad (23)$$
Let \( \hat{\phi}_i = L^*_i(\tau_i|\phi) \), we get:
\[
\min_{\phi} E[-\eta \cos(\hat{\phi}_s, \hat{\phi}_u) + L(\tau_u|\phi)]
\] (24)
where \( \hat{\phi}_i \) denotes the relative position of the optimal solution \( \phi \) for the task \( \tau_i \) to the weight initialization. By calculate the expectation of Equation 24, the objective can be formalized as:
\[
\min_{\phi} E_{\tau_s}[E_{\tau_u}[-\eta \cos(\hat{\phi}_s, \hat{\phi}_u)] + E_{\tau_u}[L(\tau_s|\phi)]
\] (25)
where the first term is exactly the class neighborhood structure in the parameter space. And the second term denotes minimizing the empirical risk. As shown in Figure 3, we argue that preserving the class neighborhood structure in parameter space is more effective than minimizing the Euclidean distance like Reptile.

4. Experimental results

4.1. Datasets

The miniImageNet [34] consists of 100 classes of ImageNet [4] and each class has 600 images of size \( 84 \times 84 \). We follow the standard split proposed in [22]: the whole dataset is divided into three subsets which take 64, 16, and 20 classes for training, validation, and test.

The tieredImageNet [23] has 608 classes which are randomly chosen from the ImageNet [4]. As proposed in [23], the 608 classes are split into training, validation, test subsets which contain 351, 97, and 160 classes respectively. It is further split into 34 high-level semantic categories including 20, 6, 8 classes for training, validation and test. In total, there are 779,165 images with a size of \( 84 \times 84 \).

The CIFAR-FS includes 100 classes which is derived from CIFAR-100 dataset [1] and each class has 600 images of size \( 32 \times 32 \). The whole dataset is divided into three subsets: 64 training classes, 16 validation classes and 20 test classes.

The FC100 is a subset of CIFAR-100 dataset [21] which is composed of 100 classes. We follow the standard split proposed in [16]. The 100 classes are split into 60, 20, and 20 classes for training, validation and test. It is further divided into 20 higher level semantic classes including 12 training, 4 validation and 4 test classes. In total, there are 60,000 images of size \( 32 \times 32 \).

The CUB-200-2011 [35] is a fine-grained dataset of birds. It contains 200 species and 11,788 images. We followed the commonly used evaluation protocol proposed by [12]. The dataset has 100 training classes, 50 validation classes and 50 test classes. Each image is resized to \( 84 \times 84 \).

4.2. Implementation Details

Network Architectures To ensure a fair comparison with existing methods, we adopt two commonly used back-bones as our feature extractor: Conv-128 [7] and WRN-28 [36]. Conv-128 is composed of 4 convolutional modules with \( 3 \times 3 \) convolutions, each followed by a BatchNorm [14], a ReLU nonlinearity [19], and a \( 2 \times 2 \) max-pooling unit. With input images of size \( 84 \times 84 \), the output feature map has size \( 128 \times 5 \times 5 \) and then be flattened into a final 3200-dimension feature vector. WRN-28 is a 28-layer Wide Residual Network [36] with a width factor 10. It consists of 3 blocks, and each block has 4 BasicBlocks [11]. A BasicBlock is composed of 2 convolutional modules with \( 3 \times 3 \) convolutions, each followed by a BatchNorm [14], a ReLU nonlinearity [19] and a short-cut connection [11]. There is a dropout unit [31] at the end of the each block. The size of input image is \( 84 \times 84 \) and the output feature is 640-dimension after the global average pooling in the last block.

Training and Evaluation Firstly, we random sample \( m \) batches as \( m \) tasks in each outer step. Then, we apply SGD on these \( m \) tasks and get the inner loop parameter \( \theta_m \). The real parameter \( \phi \) of our model \( F_{\phi}(\cdot) \) is updated by \( \theta_m \). In our experiment, \( m \) is set to 5. We use random rotation and random crop for data augmentation at the training stage. For each novel task, we firstly adapt our feature extractor to the support set and evaluate on the query set. The query set contains 15 samples per class which is consistent with existing works. The accuracy is averaged from 600 episodes with 95% confidence interval. Detailed setups in our experiments are provided in appendix.

4.3. Comparison with State-of-the-arts

miniImageNet and tieredImageNet Table 1 summarizes the results on miniImageNet [34] and tieredImageNet [23] where our approach outperforms existing state-of-the-art methods with a significant improvement. Compared with MTL [32], our method shows good generalization ability. MTL only updates a Scale-and-Shift parameter on novel tasks to prevent overfitting while we update the whole feature extractor. It takes more risks of overfitting when updating far more parameters to adapt to novel tasks. We still outperform MTL by 12% on 5-way 5-shot miniImageNet classification. From the last two rows in Table 1, we note that our proposed Amphibian can further improve the performance which is consistent with our theoretical analysis.

CIFAR-FS and FC100 Results are reported in Table 2 which illustrates our state-of-the-art performance on CIFAR-FS and FC100. Especially in 5-shot, our method achieves the accuracy of 89.3% in CIFAR-FS and 66.9% in FC100. We outperforms MetaOptNet-SVM [16] by 5.1% and 11.1% respectively. FC100 is a harder dataset with a large gap between base and novel classes. In such a tough dataset, we still achieve state-of-the-art performance which shows our capability of fast and generalized adaptation.

miniImageNet → CUB-200-2011 To further highlight
gin achieving the state-of-the-art result. Note the results of cross-domain few-shot learning is a more tough challenge and evaluate it on the novel classes in CUB-200-2011. The adaptation ability of our method, we conduct extensive experiments in cross-domain few-shot learning: miniImageNet → CUB-200-2011. The results are shown in Table 3. We train the model on the base classes in miniImageNet and evaluate it on the novel classes in CUB-200-2011. The cross-domain few-shot learning is a more tough challenge due to the huge gap between the two datasets. Our proposed method outperforms other methods by a large margin achieving the state-of-the-art result. Note the results of other methods is evaluated with image size of 224 × 224, while we evaluate with image size of 84 × 84. Smaller input images contain less information. So we actually test our method in a more difficult setting. However, our proposed ACC+Amphibian still outperforms existing methods with a significant margin. It illustrates that the ACC+Amphibian is able to fast adapt to novel tasks even under a large domain gap.

### 4.4. Ablation Study

**Backbone Comparison** Table 4 shows the results on the 5-way miniImageNet and tiredImageNet using different backbone as the feature extractor. At test time, CC directly uses mean vectors as classification weights without adapting the feature extractor to novel classes. Our proposed ACC makes adaptation to novel classes which outperforms CC on all tasks. With deeper backbone, the effect of adaptation becomes more significant. For instance, ACC exceeds CC by 0.27% on 5-shot miniImageNet with Conv-128 and notably, it exceeds CC by 7.8% with WRN-28. As we know,
Table 4. Results on miniImageNet and tieredImageNet with different backbones. The results of CC are reported by our implementation.

| Algorithm            | Backbone | miniImageNet  | tieredImageNet |
|----------------------|----------|---------------|----------------|
|                      |          | 1-shot        | 5-shot         | 1-shot         | 5-shot         |
| CC                   | Conv-128 | 52.42         | 71.05          | 55.56          | 72.87          |
| ACC                  | Conv-128 | 52.81         | 71.32          | 56.06          | 73.98          |
| ACC + Amphibian      | Conv-128 | 53.36         | 74.75          | 56.88          | 75.08          |
| CC                   | WRN-28   | 60.32         | 77.61          | 64.59          | 81.47          |
| ACC                  | WRN-28   | 62.08         | 85.41          | 66.23          | 83.42          |
| ACC + Amphibian      | WRN-28   | 64.21         | 87.75          | 68.77          | 86.75          |

Figure 4. Accuracy comparison of adaptation and fine-tuning in different steps. The results are reported on 5-way miniImageNet classification tasks. Best viewed in color.

Table 5. Comparison with MAML and Reptile under the same experiment settings on miniImageNet. Our proposed Amphibian outperforms MAML and Reptile by a significant margin. Results are collected from [20].

| Algorithm                | 1-shot | 5-shot |
|--------------------------|--------|--------|
| MAML + Transduction     | 48.70  | 63.11  |
| FOMAML + Transduction   | 48.07  | 63.15  |
| Reptile                 | 47.07  | 62.74  |
| Reptile + Transduction  | 49.97  | 65.99  |
| Amphibian               | 47.95  | 67.58  |
| Amphibian + Transduction| 50.58  | 68.48  |

Impact of Adaptation We show the accuracy comparison of adaptation and fine-tuning in Figure 4. CC+fine-tuning refers to retraining a cosine classifier for novel classes at test time, which is depicted by the dashed lines in Figure 4. It is clear to see that with our adaptation (see solid lines), the model fast adapts to the novel classes in a few steps and achieves remarkable performance. After 2 steps, ACC achieves the highest accuracy of 62.08% in 1-shot. And after 6 steps, it achieves the accuracy of 85.41% in 5-shot. From the high performance, we highlight that our method has strong ability of fast adaptation to specific tasks without overfitting. The fine-tuning method performs poorly in few-shot scenario. It converges slowly and suffers from severe overfitting in small dataset.

Impact of Amphibian In Table 4 and Figure 4, the result comparison of ACC and ACC+Amphibian shows the improvement brought by Amphibian. It finds a better weight initialization of the feature extractor, which increases the accuracy by 3.33% in 5-shot tieredImageNet with WRN-28. Furthermore, we compare our Amphibian with MAML and Reptile in Table 5 to show the superiority of our method. For fair comparison, we use the same CNN architectures and data preprocessing as adopted in [6]. The optimizer is reset in every task to prevent information leakage when testing. As indicated in Reptile [20], [6] uses transduction setting for evaluating thus, we further provide the results under transduction setting. It shows that Amphibian surpasses MAML and Reptile in all cases. In 5-way 5-shot, Amphibian outperforms Reptile by a significant margin of 4.84%. The results verify the theoretical analysis in section 3.4 that our method is more valid than Reptile to find a better weight initialization.

5. Conclusions

We propose Adaptable Cosine Classifier and Amphibian to achieve both fast and generalized adaptation for few-shot learning. Adaptable Cosine Classifier provides the ability of fast adapting a metric based model on few data without overfitting. And Amphibian further improves the performance by learning a weight initialization that can perform great generalization to novel tasks. The theoretical explanation is provided to explain the rationale of our proposed Amphibian. It shows that Amphibian can learn a parameter space where optimal solutions for the tasks belongs to the same class are close to each other. We achieve state-of-the-art performance on four few-shot benchmarks. Furthermore, our method outperforms existing works on the fine-tuning a deep network on few data is easily to overfit and that is why MTL [32] merely fine-tunes a small scalar parameter. However, we adapt the whole feature extractor on small data without overfitting even if the backbone goes deeper. The results suggest that our proposed method can adapt deep networks in few-shot scenario without overfitting.

We propose Adaptable Cosine Classifier and Amphibian to achieve both fast and generalized adaptation for few-shot learning. Adaptable Cosine Classifier provides the ability of fast adapting a metric based model on few data without overfitting. And Amphibian further improves the performance by learning a weight initialization that can perform great generalization to novel tasks. The theoretical explanation is provided to explain the rationale of our proposed Amphibian. It shows that Amphibian can learn a parameter space where optimal solutions for the tasks belongs to the same class are close to each other. We achieve state-of-the-art performance on four few-shot benchmarks. Furthermore, our method outperforms existing works on the
cross-domain few-shot problem.

References

[1] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In International Conference on Learning Representations, 2019. 7

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4):834–848, 2017. 1

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In International Conference on Learning Representations, 2019. 7

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255. Ieee, 2009. 6

[5] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. arXiv preprint arXiv:1909.02729, 2019. 7

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning, pages 1126–1135. JMLR. org, 2017. 1, 2, 3, 4, 5, 7, 8

[7] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4367–4375, 2018. 1, 2, 3, 6, 7

[8] Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 21–30, 2019. 7

[9] Ross Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 1440–1448, 2015. 1

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 2961–2969, 2017. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016. 1, 6

[12] Nathan Hilliard, Lawrence Phillips, Scott Howland, Artém Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. arXiv preprint arXiv:1802.04376, 2018. 6

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997. 3

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, pages 448–456, 2015. 6

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998. 1

[16] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 10657–10665, 2019. 2, 6, 7

[17] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7260–7268, 2019. 7

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3431–3440, 2015. 1

[19] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the International Conference on Machine Learning, pages 807–814, 2010. 6

[20] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999, 2018. 1, 2, 3, 4, 5, 7, 8

[21] Boris Oreshkin, Pau Rodriguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In Advances in Neural Information Processing Systems, pages 721–731, 2018. 6, 7

[22] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In International Conference on Learning Representations, 2017. 6

[23] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In International Conference on Learning Representations, 2018. 6

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99, 2015. 1

[25] Herbert Robbins and Sutton Monro. A stochastic approximation method. The Annals of Mathematical Statistics, pages 400–407, 1951. 3

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015. 1

[27] Andrei Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In International Conference on Learning Representations, 2019. 7

[28] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Proceedings of
[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

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

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[32] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

[33] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[34] 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.

[35] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Advances in Water Resources*, 2011.

[36] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.
A. Theoretical Analysis

A.1. Gradient Formula

In this section, we give the derivation of Eq. (17) in the paper. We define $L_\phi’(\tau|\phi) = \frac{\partial}{\partial \phi} L(\tau|\phi)$ and $L_\phi''(\tau|\phi) = \frac{\partial^2}{\partial \phi^2} L(\tau|\phi)$ where $\tau$ is a task and $\phi$ is the learnable parameter of our feature extractor. We take two inner steps as an example to derive the formulation of our proposed Amphibian. $\phi_0$ is initialized by $\phi$: $\phi_0 = \phi$. The updated parameter after two SGD steps:

$$\phi_1 = \phi_0 - \beta L_\phi’(\tau_1|\phi_0) \tag{26}$$
$$\phi_2 = \phi_0 - \beta L_\phi’(\tau_1|\phi_0) - \beta L_\phi’(\tau_2|\phi_1) \tag{27}$$

In this manuscript, $\beta$ is the inner loop learning rate, which is consistent with the description in our paper. Then, we get the formulation of $L_\phi’(\tau_2|\phi_1)$:

$$L_\phi’(\tau_2|\phi_1) = L_\phi’(\tau_2|\phi_0) + L_\phi’(\tau_2|\phi_0)(\phi_1 - \phi_0) + O(\delta^2)$$
$$= L_\phi’(\tau_2|\phi_0) - \beta L_\phi’(\tau_2|\phi_0) L_\phi’(\tau_1|\phi_0) + O(\delta^2) \tag{28}$$

We provide the gradient formula of our Amphibian as:

$$g_{\text{Amphibian}} = L_\phi’(\tau_1|\phi_0) + L_\phi’(\tau_2|\phi_1)$$
$$= L_\phi’(\tau_1|\phi_0) + L_\phi’(\tau_2|\phi_0)$$
$$- \beta L_\phi’(\tau_2|\phi_0) L_\phi’(\tau_1|\phi_0) + O(\delta^2) \tag{29}$$

A.2. Analysis in the Parameter Space

In this section, we give more detailed analysis for clear comparison with other method. As mentioned in the paper, Reptile is an extended version of MAML. Thus we only give the objective of Reptile for illustration.

Reptile targets to update on task $\tau_0$, one step and achieves the best performance on $\tau_0$. The objective can be formulated as:

$$\min_{\phi} E[L(\tau_s|\phi')] \tag{30}$$
$$\phi' = \phi - \beta L_\phi’(\tau_s|\phi) \tag{31}$$

Eq. 30 can be decomposed into two parts:

$$\min_{\phi} E[L(\tau_s|\phi')] = \min_{\phi} E[(L(\tau_s|\phi') - L(\tau_s|\phi)) + L(\tau_s|\phi)] \tag{32}$$

We can estimate the first term by Taylor Expansion:

$$L(\tau_s|\phi') = L(\tau_s|\phi) + L_\phi’(\tau_s|\phi)(\phi' - \phi) + O(\delta^2) \tag{33}$$

Substituting Eq. 33 into Eq. 32, we get:

$$\min_{\phi} E[L_\phi’(\tau_s|\phi)(\phi' - \phi) + L(\tau_s|\phi)] \tag{34}$$

We take Eq. 31 into Eq. 34 then the objective of Reptile is:

$$\min_{\phi} E[-\beta L_\phi’(\tau_s|\phi)L_\phi’(\tau_s|\phi) + L(\tau_s|\phi)] \tag{35}$$

The first term can be further simplified as:

$$-\beta\|L_\phi’(\tau_s|\phi)\|^2_2 \tag{36}$$

The overall objective of Reptile can be formulated as:

$$\min_{\phi} E[-\beta\|L_\phi’(\tau_s|\phi)\|^2_2 + L(\tau_s|\phi)] \tag{37}$$

We give the objective of our Amphibian which is the same as the 22-th equation of the paper:

$$\min_{\phi} E[-\beta L_\phi’(\tau_s|\phi)L_\phi’(\tau_s|\phi) + L(\tau_s|\phi)] \tag{38}$$
Table 6. Hyper-parameters of ImageNet-derivatives: miniImageNet and tieredImageNet and CIFAR-derivatives: CIFAR-FS and FC100.

| Parameter                      | ImageNet-derivatives | CIFAR-derivatives |
|--------------------------------|----------------------|-------------------|
| Train                          |                      |                   |
| Epoch number                   | 40                   | 40                |
| Inner batch size               | 64                   | 256               |
| Inner iteration $m$            | 5                    | 5                 |
| Outer step size $\alpha$      | 0.5                  | 0.5               |
| Outer iterations per epoch    | 1000                 | 1000              |
| SGD learning rate $\beta$ (1-20 epoch) | $10^{-1}$         | $6 \times 10^{-3}$ |
| SGD learning rate $\beta$ (21-40 epoch) | $6 \times 10^{-3}$ | $6 \times 10^{-3}$ |
| SGD momentum                   | 0.9                  | 0.9               |
| SGD weight decay               | $5 \times 10^{-4}$   | $5 \times 10^{-4}$ |
| Dropout rate                   | 0.3                  | 0.3               |
| Evaluation                     |                      |                   |
| Adam learning rate             | $10^{-4}$            | $10^{-4}$         |

The first term is derived as:

$$-\beta L^\prime_\phi(\tau_u|\phi)L^\prime_\phi(\tau_s|\phi) =$$

$$-\beta \|L^\prime_\phi(\tau_u|\phi)\|_2 \cdot \|L^\prime_\phi(\tau_u|\phi)\|_2 \cdot \cos(L^\prime_\phi(\tau_u|\phi), L^\prime_\phi(\tau_s|\phi))$$

(39)

Thus the overall objective of our Amphibian is:

$$\min \hat{\phi} E[-\eta \cos(L^\prime_\phi(\tau_u|\phi), L^\prime_\phi(\tau_s|\phi)) + L(\tau_u|\phi)]$$

(40)

where

$$\eta = \beta \|L^\prime_\phi(\tau_u|\phi)\|_2 \cdot \|L^\prime_\phi(\tau_s|\phi)\|_2$$

(41)

From the derivation of Reptile and Amphibian, we can see that Reptile targets to find a point where the $L^2$ norm of the gradient is maximum. At this point, models can achieve fast gradient update. Notably, our Amphibian considers not only the $L^2$ norm of the gradient but also the gradient direction. It shows that our method can achieve fast and generalized adaptation in the parameter space. The formulation in the paper is simplified for illustration.

B. Setup

We display the detailed setup in Table 6. The hyper-parameters of different datasets are almost same. It demonstrates that our method can perform well on different datasets without much hyper-parameter tuning.

C. Comparison with Fine-tuning and Adaptation

The accuracy comparison displayed in the paper only gives the fine-tuning accuracy within 10 steps. For clear illustration, we give the complete accuracy curve of fine-tuning as shown in Figure 5(b). Fine-tuning is slower and takes much more steps than adaptation (see Figure 5(a)). We can see that CC+fine-tuning just achieves the 5-shot accuracy of 70% which is much lower than our ACC. ACC can be viewed as CC+adaptation which takes 6 adaptation steps to achieve the best 5-shot accuracy of 85.41%. It demonstrates the ability of fast and generalized adaptation of our method.