Meta Module Generation for Fast Few-Shot Incremental Learning

Shudong Xie Yiqun Li Dongyun Lin Tin Lay Nwe Sheng Dong
Institute for Infocomm Research, A*STAR, Singapore
{xie_shudong,yqli,lin_dongyun,tlnma,dongs}@i2r.a-star.edu.sg

Abstract

There are two challenging problems in applying standard Deep Neural Networks (DNNs) for incremental learning with a few examples: (i) DNNs do not perform well when little training data is available; (ii) DNNs suffer from catastrophic forgetting when used for incremental class learning. To simultaneously address both problems, we propose Meta Module Generation (MetaMG), a meta-learning method that enables a module generator to rapidly generate a category module from a few examples for a scalable classification network to recognize a new category. The old categories are not forgotten after new categories are added in. Comprehensive experiments conducted on 4 datasets show that our method is promising for fast incremental learning in few-shot setting. Further experiments on the `miniImageNet` *dataset show that even it is not specially designed for the N-way K-shot learning problem, MetaMG can sitll perform relatively well especially for 20-way K-shot setting.*

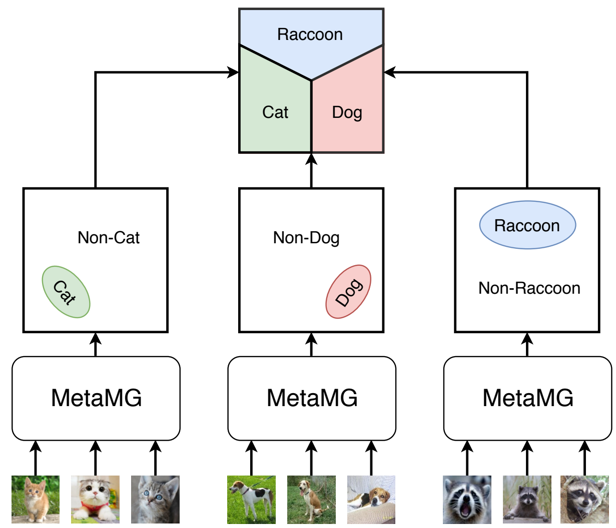

Figure 1. An illustration of few-shot incremental learning using MetaMG for three categories: cat, dog and raccoon. Given 3 examples of a new category, *e.g*., cat, our MetaMG generates a module which encloses the region belonging to cat in the feature space. By repeating the similar steps, three modules corresponding to the three categories can be generated. They are jointly adapted for the three-categorical classification problem based on the rule of selecting the module with the highest score.

1. Introduction

Deep learning has achieved great success in supervised image classification [18, 16, 12, 29]. A general pipeline to train high capacity deep neural networks is to iteratively tune the network parameters on a large amount of labelled data using gradient-based approaches. However, deep neural networks trained through this pipeline can easily break down due to overfitting when encountering the situation where objects of new categories are required to be classified with very few training samples. Intuitively, such a limitation of deep neural networks contradicts the fact that human learning is efficient and incremental. Human beings can apply the experience learned from the past to achieve fast generalization on new categories from very limited examples. Human can also accumulate new experience through learning without much forgetting. These abilities are imitated in machine learning and named as few-shot learning and incremental learning. There are a growing number of recent research interests in few-shot learning [14, 32, 30, 9, 20, 26, 31] and incremental learning [28, 21, 27, 19, 37, 35].

Few-shot learning [8, 17, 32] aims at learning to recognize visual categories using only a few labelled exemplars from each category. Specifically, an N-way K-shot learning task is framed as learning to discriminate N categories providing K training examples for each category [32]. Such a task could be treated as an extreme case of training data shortage where transfer learning [7, 2] and regularization could face big challenges due to overfitting. Currently, the existing approaches to solving few-shot image classification exploit the idea of meta-learning or “learning to learn”. Unlike the traditional supervised classification paradigm where training is conducted from a set of labelled exemplars, meta-learning is conducted based on a set of tasks, each containing a training set and a testing set. In the context of supervised image classification, meta-learning frames a learning process at two phases: meta-training and meta-testing. In the meta-training phase, a meta-learner is trained by learning from a number of tasks from an auxiliary dataset to capture transferable knowledge across the tasks. Such knowledge could be image representations where the similarity between images can be measured through defined metrics [32, 14, 31, 30] or optimizers which can provide optimization strategies tailored to address the classification problem under the few-shot setting [9, 26, 20]. After meta-training, the meta-learner can be applied to address the targeting few shot classification problem by treating it as a new task and solving it using the generalized knowledge learned from auxiliary tasks in the meta-training phase.

Motivated by the observations that human beings can efficiently learn to identify a new category by subconsciously comparing the new examples with the visual concepts kept in the memory, we propose Meta Module Generation (MetaMG), a method that leverages meta-learning to enable a module generator with the ability to rapidly generate a category module from a few examples for a scalable classification network to recognize a new category. The recognition is achieved by jointly adapting all the category modules together to partition the feature space into different regions for different categories. Figure 1 illustrates the intuition behind our incremental learning process.

Overall, the proposed method provides a clean meta-learning solution to generate new modules by feedforwarding the training images through a module generator network without further weight updates. Thereby a fast and effective few-shot incremental learning is realized. It is also noted that our MetaMG remains dynamically expanded as the discriminative knowledge of the new category is incorporated into the generated module. Therefore, the proposed method can implement fast incremental learning without forgetting the learned knowledge. Also, no retraining or data storing for the learned categories is required, which keeps our method under low computational and storage complexities.

Our contributions are twofold:

1. We propose a novel meta-learning framework for fast incremental learning using a few examples from each category starting from the first category. The framework includes a module generator which is able to generate category modules to form a scalable classification network. We have explored an LSTM-based module generator that is better in few-shot feature correlation than DeepSets, and a spherical category module that performs relatively well.

2. We proposed MetaMG, a meta-learning method that enables a module generator to rapidly generate a category module from only a few examples. The category module is then plugged into the scalable classification network to recognize a new category.

2. Related Work

Few-shot image classification has been studied using generative models and statistical inferences in some early works [8, 17]. Along with discriminative neural networks such as CNNs that produce superior performance for image classification, there has been growing interests in applying deep neural network models to address the few-shot image classification problem [32, 9]. Regarding discriminative approaches, meta-learning is commonly conducted to learn transferable knowledge from similar tasks and apply the knowledge to address the targeting few-shot classification problem. Since our method is proposed to solve few-shot incremental learning using discriminative neural network structures and meta-learning, here we briefly review several state-of-the-art deep neural network based few-shot learning methods and incremental learning methods.

2.1. Meta-Learning on Embedding and Metrics

The first group of methods that solve the few-shot image classification problem cast the problem under an image verification framework [14, 32]. These methods learn project functions for image embedding in the meta-learning phase. In the meta-testing phase, the training images and testing images are projected to the learned embedding space and classification is implemented either by image verification, *i.e.*, comparing training and testing images in pairs [14], or by nearest neighbor classification [32]. Snell *et al*. [30] further extended the idea of image verification to a prototype matching by using the class centroids in the embedding space as the templates. Sung *et al*. [31] designed a relation network as a non-linear comparator instead of fixed linear comparators [14, 32] to classify images in the embedding space. The embedding and metric learning approaches do not require further fine-tuning in the meta-testing phase and hence the performance of these methods relies on the assumption that the embedding learned across the meta-training tasks is sufficiently discriminative for the new tasks.

Compared with the embedding and metric learning methods, our method also has the similar advantage of fast training in the meta-testing phase. For module generation, the parameters of the module are generated by simply feedforwarding the training examples through the module generator network without fine-tuning. Through meta-training, we assume that the proposed module generator is capable of generating discriminative category modules using very few in-category examples. Hence, instead of learning discriminative embeddings, our method focuses on generating discriminative non-linear decision boundaries.

2.2. Meta-Learning on Optimizers and Parameters

Another group of methods apply meta-learning across the meta-training tasks to learn an optimizer which can provide optimization strategies for a deep neural network to fine-tune without severe overfitting using very few training examples within a small number of gradient-descent updates. The MAML [9] provides an effective way to learn initial conditions through meta-learning. From the observation that stochastic gradient descent rule resembles the update of the cell state in LSTM, Ravi and Larochelle [26] extended the idea of MAML by proposing a meta-learner LSTM to learn not only initial conditions, but also the learning rates and update directions of SGD. Li *et al*. [20] proposed meta-SGD which is similar to MAML but can also learn learning rates and update directions. Compared with meta-learner LSTM, meta-SGD can achieve faster learning in the meta-testing phase since only one iteration of fine-tuning is applied. For most optimizer learning methods, fine-tuning is required and therefore the computational complexity is generally higher than embedding and metric learning based approaches.

Compared with optimizer learning methods, our method also leverages meta-learning to output module generator parameters which can be used in the meta-testing phase. The existing optimizer learning approaches learn for the optimization conditions which can be used for weights update in the meta-testing phrase. However, our module generator directly learns to output the category module weights and therefore no further fine-tuning is required in meta-testing.

MetaMG is similar to the Qiao *et al*. [25] and Gidaris *et al*. [10] in the way of generating network parameters by feedforwarding images. The differences are: firstly, MetaMG is able to learn new classes quickly using a few examples while [25] aims at addressing few-shot classification without incremental learning capability; secondly, MetaMG is able to add new class from the start, whereas [10] aims at building an incremental classification system in which a batch of base categories are learned first. Then, each new class is added incrementally.

2.3. Incremental Learning

Many approaches have been proposed for inclemental learning. These include the approaches such as End-to-End [4], FearNet [13], A-GEM [6], RWalk [5] and ExStream [11] and more listed in [1]. Most of these approaches address the catastrophic forgetting problem of CNNs [22], *i.e*., training from class-incremental examples might cause the classification performance to quickly and severely deteriorate for those previously learned classes. To alleviate this problem, a group of approaches selectively store a subset of the previous training data to represent the learned classes. For example, iCaRL [28] stores a subset of previous training samples which can best represent the corresponding category and trains a class-incremental learner based on nearest neighbor classification. To alleviate the catastrophic forgetting problem, iCaRL tunes the network parameters by minimizing the cost function including a distillation term [28] to make the predictions on the learned classes invariant. Lopez-Paz *et al*. [21] proposed Gradient Episodic Memory (GEM) which can provide positive backward transfer to the learned classes, *i.e*., learning new tasks could enhance the performance on the previous knowledge. The second group of approaches handle the catastrophic forgetting problem by dynamically expanding the network with regularization and no training data from learned classes are retained [27, 19, 37, 35]. The third group of approaches such as Gidaris *et al*. [10] etc. studied various frameworks for incremental learning of novel categories after a set of base categories are learned. Compared to the above mentioned approaches, the uniqueness of our method is that we perform incremental learning using a few labeled examples for every category and start incremental learning from the first category without the learning of a set of base categories using a lot of examples. Retraining and reuse of sample data for previous categories are not required as well. Technically, our method can generate a discriminative module for each new category without retraining or storing previously learned categorical examples, which in theory, could achieve unlimited category learning, *i.e*., lifelong learning.

3. Framework

An overview of our framework is illustrated in Figure 2. It has two components, namely a classification network and a module generator. The design of the framework components is based on the following principles:

- When a pretrained feature extractor is applied on a very different dataset, the classes of embed features may not be linearly separable. Therefore, category modules should be **non-linear** in order to separate them. Moreover, the size of classification network should be endurable after a large amount of category modules are added in. Thus we also require a category module to be **lightweight**.
- The module generator is a function $G: \ R^{K\times d} \rightarrow R^p$ that maps features $\{f_1, \cdots, f_K\}$ of category examples to the weights $\boldsymbol{w}$ of a category module. Therefore, its architecture should be **strong in feature correlation** in order to produce a highly relevant category module.

The detailed design of framework components are discussed in the following sections.

3.1. Classification Network

The **classification network** is a cascade of a feature extractor and a set of lightweight category modules as shown in Figure 2. The feature extractor is a convolutional neural network which aims at producing discriminative features for image samples, whereas a category module aims at supporting samples in its corresponding category by outputting

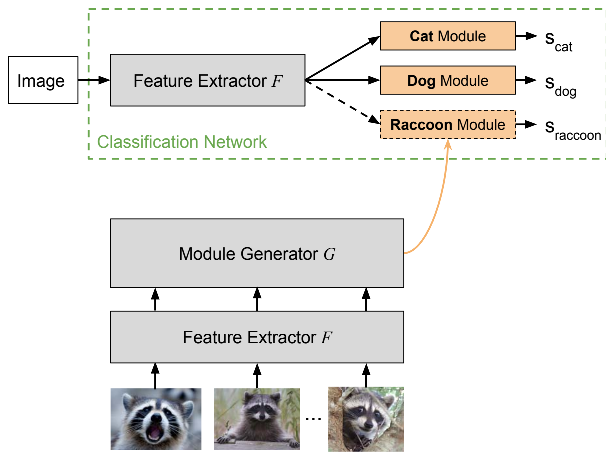

Figure 2. An illustration of the proposed method. Given a few training examples of a new category, *e.g.*, *raccoon*, the module generator generates a category module which can support the category *raccoon* by outputting higher scores for *raccoon* samples than other modules. The generated category module is plugged into the classification network and thereby the updated network can recognize the new category.

scores higher than those produced by the other modules. Given a test image being feedforwarded through the classification network, the category module that outputs the highest score indicates the predicted category.

A category module should be capable to non-linearly enclose a region in the feature space. In this paper, 3 different category modules are explored.

Spherical Category Module. A naïve choice is to use a hypersphere as the category module to approximately enclose the category region. In this way, a category module holds a centroid vector $\mathbf{m}$ and a radius r. Given a feature point $\mathbf{p}$, a spherical category module computes

$$r - \sqrt{(\mathbf{p} - \mathbf{m})^\top (\mathbf{p} - \mathbf{m})}. \tag{1}$$

Multi-Gaussian Category Module. We could design a category module under a natural assumption that feature points of a category follow a multivariate Gaussian distribution. In this way, a category module holds a mean vector $\boldsymbol{\mu}$ and a covarance matrix Σ. We adopt the Mahalanobis distance to restrict the corresponding feature points to be within 3 standard deviations from the mean. Given a feature point $\mathbf{p}$, a Multi-Gaussian category module computes

$$3 - \sqrt{(\mathbf{p} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{p} - \boldsymbol{\mu})}. \tag{2}$$

A problem to this design is that the covariance matrix has too many parameters which not only make the module heavy but also introduce difficulty to the optimization process. To alleviate this problem, we use $\Sigma = \mathrm{diag}(\sigma_1^2, \cdots, \sigma_d^2)$ to approximate the distribution.

MLP Category Module. We could also define a category module as a multi-layer perceptron (MLP) without imposing any assumption on the distribution of feature points. In this paper, the module contains a linear layer with 16 units, followed by a ReLU activation and a linear layer with 1 unit.

3.2. Module Generator

The **module generator** is designed to generate category modules which can be plugged into the classification network to recognize their corresponding categories. As illustrated in Figure 2 where two category modules for category *cat* and *dog* have been generated. Given a few training examples from a new category *raccoon*, a category module corresponding to *raccoon* is generated by feedforwarding the examples through the module generator network. The new *raccoon* category module can be plugged into the classification network and thereby the updated network can recognize the *raccoon* category.

The module generator should be capable to correlate the features of category examples. In this paper, 2 different architectures which could achieve this goal are explored.

LSTM-Based Module Generator. Walch *et al.* [34] has reported that the LSTM network is a powerful tool for feature correlation. For this reason, we specify the architecture of our module generator with an LSTM using an encoder-decoder structure as illustrated in Figure 3. The encoder part is responsible for feature correlation. It consists of a linear layer with 256 units for dimensionality reduction, followed by an LSTM network with 512 hidden units. The decoder is a single linear layer which is responsible for mapping the correlated features to the module parameters.

DeepSets-Based Module Generator. The module generator can be viewed as a function that maps a set of example features to a vector about the parameters. Thus, it is natural to adopt architectures that deal with set operations. DeepSets [38] has been proved to be capable to represent any permutation-invariant function that deals with set operations. Therefore, we also explore a DeepSets-based module generator.

The ability of the module generator to generate discriminative category modules by feeding a few training examples is learned through meta-learning on auxiliary tasks. The details of the meta-learning process is described in §4.

4. Few-Shot Incremental Learning

4.1. Notations

In this paper, we follow the notations related to meta-learning in [26] as below.

Meta-Sets. For meta-learning datasets, we exploit a meta-training set, a meta-validation set and a meta-testing set denoted as $\mathscr{D}_{\textit{meta-train}}$, $\mathscr{D}_{\textit{meta-val}}$ and $\mathscr{D}_{\textit{meta-test}}$, respec-

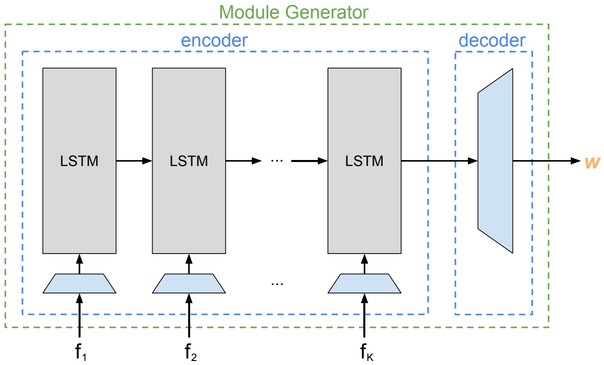

Figure 3. The LSTM-based module generator. We adopt an encoder-decoder structure in this module generator. The encoder consists of a linear layer (blue) that embeds the features $\{f_1, \cdots, f_K\}$ of a category support set to a lower dimensional space, and an LSTM network (gray) that correlates the example features together. The decoder is a single linear layer (blue) that maps the correlated features (*i.e.*, the last hidden state vector) to the parameters space.

tively. The meta-training set $\mathscr{D}_{\textit{meta-train}}$ is used to build meta-training tasks to train the module generator G. The meta-validation set $\mathscr{D}_{\textit{meta-val}}$ is used to monitor the training performance and select the best parameters of G. The meta-testing set is to evaluate the classification performance.

Task. A task τ defined over a meta-set $\mathscr{D}$ consists of a training set $D_{\textit{train}}(\tau)$ and a testing set $D_{\textit{test}}(\tau)$. The training set $D_{\textit{train}}(\tau)$ contains support sets of random chosen categories which serve as inputs for G to generate category modules. The testing set $D_{\textit{test}}(\tau)$ contains multiple randomly chosen *(sample, label)*-pairs which are used to evaluate the discriminative performance of generated modules. For a task in the meta-training phase, the loss is calculated on $D_{\textit{test}}(\tau)$ and is backpropagated to update the parameters of G. Different tasks can be built by randomly drawing samples for $D_{\textit{train}}(\tau)$ and $D_{\textit{test}}(\tau)$. All the possible tasks form the task space which distribution is defined as $p(\tau)$.

4.2. Problem Formulation

Our MetaMG aims at building a module generator with the ability to generate from a few examples a category module that could enclose the region belonging to the category in the feature space. To achieve the goal, we define the meta-training task in details as follows. A task τ corresponds to C randomly chosen categories. Its training set $D_{\textit{train}}(\tau)$ is a set of support sets for each category

$$D_{\textit{train}}(\tau) = \{S_1, \cdots, S_C\} \tag{3}$$

where each support set S_c consists of K category examples. The testing set $D_{\textit{test}}(\tau)$, on the other hand, is a set of $(sample, label)$-pairs

$$D_{\textit{test}}(\tau) = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\} \tag{4}$$

with each category exactly $T = N/C$ samples.

A category module is a function $M(\cdot\,; \boldsymbol{w})$ parameterized by the weights $\boldsymbol{w}$ which is generated by feeding the features of a support set through the module generator $G_{\boldsymbol{\theta}}$. For simplicity, we denote the generated category module of the c-th category as $M_{\boldsymbol{\theta}}^{(c)}(\cdot) = M(\cdot\,; G_{\boldsymbol{\theta}} \circ F(S_c))$.

We then define the loss function on the testing set $D_{\textit{test}}(\tau)$ as follows. Locally, for each category module, we would like it to produce high scores for samples in its category and low scores for those out of its category:

$$\begin{aligned} L_l(\tau, \boldsymbol{\theta}) = -\frac{1}{NC}\sum_{c=1}^{C}\Bigg[&\sum_{\substack{(\mathbf{x}_i, y_i)\\ y_i = c}} \log \sigma\big(M_{\boldsymbol{\theta}}^{(c)} \circ F(\mathbf{x}_i)\big) \\ &+ \sum_{\substack{(\mathbf{x}_i, y_i)\\ y_i \neq c}} \log\Big(1 - \sigma\big(M_{\boldsymbol{\theta}}^{(c)} \circ F(\mathbf{x}_i)\big)\Big)\Bigg], \end{aligned} \tag{5}$$

which could be further simplified as

$$\begin{aligned} L_l(\tau, \boldsymbol{\theta}) = -\frac{1}{NC}\sum_{(\mathbf{x}_i, y_i)}\Bigg[&\log \sigma\big(M_{\boldsymbol{\theta}}^{(y_i)} \circ F(\mathbf{x}_i)\big) \\ &+ \sum_{c \neq y_i} \log\Big(1 - \sigma\big(M_{\boldsymbol{\theta}}^{(c)} \circ F(\mathbf{x}_i)\big)\Big)\Bigg], \end{aligned} \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function.

Globally, among all the generated category modules, we would like the score of a sample produced by its corresponding category module to be higher than the scores produced by other category modules, which would provide an overview of the joint classification:

$$L_g(\tau, \boldsymbol{\theta}) = -\frac{1}{N}\sum_{(\mathbf{x}_i, y_i)} \log \frac{\exp\big(M_{\boldsymbol{\theta}}^{(y_i)} \circ F(\mathbf{x}_i)\big)}{\sum_c \exp\big(M_{\boldsymbol{\theta}}^{(c)} \circ F(\mathbf{x}_i)\big)} \tag{7}$$

Finally, our objective is to find $\boldsymbol{\theta}$ that minimizes the expectation of the combined loss over the task space:

$$\mathbb{E}_{\tau \sim p(\tau)}\big[L(\tau, \boldsymbol{\theta})\big] = \mathbb{E}_{\tau \sim p(\tau)}\big[L_l(\tau, \boldsymbol{\theta}) + \lambda L_g(\tau, \boldsymbol{\theta})\big] \tag{8}$$

Algorithm 1 summarizes the meta-training procedure of our MetaMG. Inside the algorithm, Line 3 samples a batch of tasks as defined in this section. Lines 5-6 generate C category modules from the support sets in $D_{\textit{train}}(\tau_i)$. Lines 7-12 compute the combined loss on samples in the testing set $D_{\textit{test}}(\tau_i)$. Finally, Line 13 updates the parameters $\boldsymbol{\theta}$ via gradient descent through the total loss of all the tasks.

5. Experiments

We first conducted an ablation study to find out the best settings for our MetaMG. We then evaluated the proposed model on few-shot incremental class learning on 4

Algorithm 1: Meta-Training of Module Generator

Input: task distribution $p(\tau)$ over a meta-set $\mathscr{D}_{train}$
Output: parameters $\boldsymbol{\theta}$ of G

$\boldsymbol{\theta} \leftarrow \texttt{Initialize}(\boldsymbol{\theta})$
while *not done* **do**
  $\boldsymbol{\tau} \leftarrow$ sample a batch of tasks from $p(\tau)$
  foreach $\tau_i \in \boldsymbol{\tau}$ **do**
    foreach $S_c \in D_{train}(\tau_i)$ **do**
      $M_{\boldsymbol{\theta}}^{(c)}(\cdot) \leftarrow M(\cdot\,; G_{\boldsymbol{\theta}} \circ F(S_c))$
    $L_l(\tau_i) \leftarrow 0$
    $L_g(\tau_i) \leftarrow 0$
    foreach $(\mathbf{x}_i, y_i) \in D_{test}(\tau_i)$ **do**
      $L_l(\tau_i) \leftarrow L_l(\tau_i) - \log \sigma\left(M_{\boldsymbol{\theta}}^{(y_i)} \circ F(\mathbf{x}_i)\right) - \sum_{c \neq y_i} \log\left(1 - \sigma\left(M_{\boldsymbol{\theta}}^{(c)} \circ F(\mathbf{x}_i)\right)\right)$
      $L_g(\tau_i) \leftarrow L_g(\tau_i) - \log \frac{\exp\left(M_{\boldsymbol{\theta}}^{(y_i)} \circ F(\mathbf{x}_i)\right)}{\sum_c \exp\left(M_{\boldsymbol{\theta}}^{(c)} \circ F(\mathbf{s}_i)\right)}$
    $L(\tau_i) \leftarrow \frac{1}{NC} L_l(\tau_i) + \frac{\lambda}{N} L_g(\tau_i)$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \left(\sum_{\tau_i \in \boldsymbol{\tau}} L(\tau_i)\right)$
return $\boldsymbol{\theta}$

image datasets in §5.2. Moreover, we compared the proposed method with several state-of-the-art methods on the `miniImageNet` dataset for the popular N-way K-shot classification problem in §5.3. Finally, we studied the efficiency of our MetaMG using either a CPU or a GPU device.

5.1. Ablation Study

An ablation study was conducted to explore the best settings for our MetaMG. The settings include category modules, module generators, and the number C of support sets in a meta-training task.

5.1.1 Experiment Settings

Dataset. The study was evaluated on the `CUB200-2011` [33] dataset which consists of 200 bird categories with each category around 60 images. We randomly split it into 80, 20, and 100 categories as the meta-training set, meta-validation set, and meta-testing set, respectively.

Feature Extractor. We used the ResNet101 model pretrained on `ImageNet` as the feature extractor throughout the ablation study, and fixed its weights during the meta-training process.

Meta-Training Hyperparameters. We set the number K of examples in a support set to be 1 and 5 for the 1-shot and 5-shot experiments, respectively. For the number N of samples in the testing set of a task, we fixed it to be $15C$ (*i.e.*, $T = 15$). Also, we set the number of tasks in a batch to be 32. Moreover, we set $\lambda = 1.0$ empirically in Eq. 8.

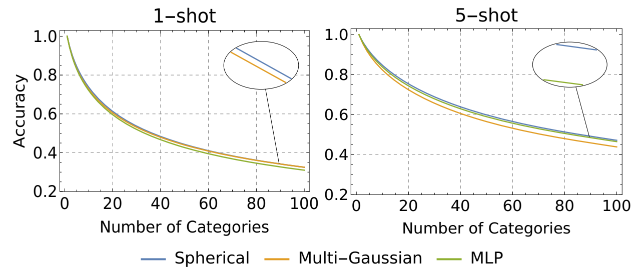

Figure 4. Comparison among 3 category modules on `CUB200-2011`. In both cases, the spherical category module performs slightly better than the other 2 modules.

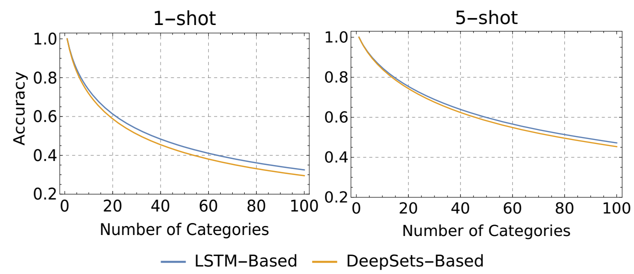

Figure 5. Comparison among 2 module generators on `CUB200-2011`. In both cases, the LSTM-based module generator performs better than the DeepSets-based one.

We trained each model 1,000 iterations, and chose the one with the best validation accuracy.

Evaluation Protocol. During the meta-testing phase, we followed the experimental protocol in [28] for incremental class learning. We generate a category module for a novel category using on its support set with K random training examples. Then, we incrementally add the generated category module to the system. For testing, we randomly selected 15 testing samples per category and measure the accuracy. After all the categories were added, we had calculated the accuracy per number of categories. To obtain stable accuracy, we conducted 20,000 iterations of incremental evaluation and calculated the average accuracy. Moreover, since the meta-training tasks are sampled randomly, even for a fixed set of parameters, different best trained models result in different accuracy during evaluation. To obtain more statistically meaningful results, for each set of parameters, we trained 10 models and averaged their evaluation accuracy to obtain our stabilized accuracy.

5.1.2 Results

Category Module. Figure 4 illustrates the accuracy with respect to the number of categories given different category modules. In the 1-shot setting, the spherical module performs slightly better than the other modules at the beginning, and yields similar accuracy as that of Multi-Gaussian module at the end. In the 5-shot setting, the spherical module performs generally better than the other two modules. We hypothesize that this is because the spherical category

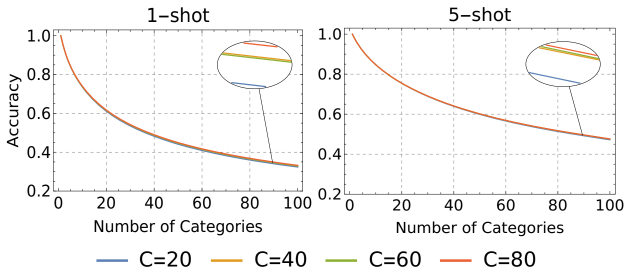

Figure 6. Comparison among different number C of support sets in a task on `CUB200-2011`. The 4 curves overlap with each other. The improvement introduced by a larger C is almost negligible.

module has fewer parameters which are easier to generate. As the spherical module performs slightly better and is lighter, we use it in the following experiments.

Module Generator. Figure 5 illustrates the accuracy with different module generators. In both cases, the LSTM-based module generator performs better than the DeepSets-based one. This suggests that the LSTM-based can better correlate the features of examples in a support set. On one hand, human learns a new concept by seeing examples one after another, and the LSTM-based module generator imitates this behaviour. On the other hand, for the LSTM-based generator, a task would become a new task by simply changing the sequence order of the feature examples in a support set, which to some degree provides more training data than the DeepSets-based one. Therefore, we choose the LSTM-based module generator in the following experiments.

Number C of Support Sets in a Task. Figure 6 illustrates the accuracy with different number C of support sets in a task. In both 1-shot and 5-shot settings, curves of different C overlap with each other. When looking closely to the curves, a larger C yields better but negligible improvement. This indicates that the choice of C has little effect on the performance of MetaMG. Since it takes a longer time for training with a larger C and the improvement is little, we use $C = 20$ in our following experiments.

5.2. Few-Shot Incremental Class Learning

In this section, we evaluated our MetaMG for few-shot incremental class learning on `CUB200-2011` as well as the following 3 image classification datasets:

- **CIFAR-100**. The `CIFAR-100` [15] dataset consists of 100 categories each with 600 images. We randomly split it for 40, 10, and 50 categories as the meta-training set, meta-validation set, and meta-testing set, respectively.
- **Flower-102**. The `Flower-102` [24] dataset consists of 102 flower categories each containing 40~258 images. We randomly split the dataset into 42, 10, and 50 categories as the meta-training set, meta-validation set, and meta-testing set, respectively.

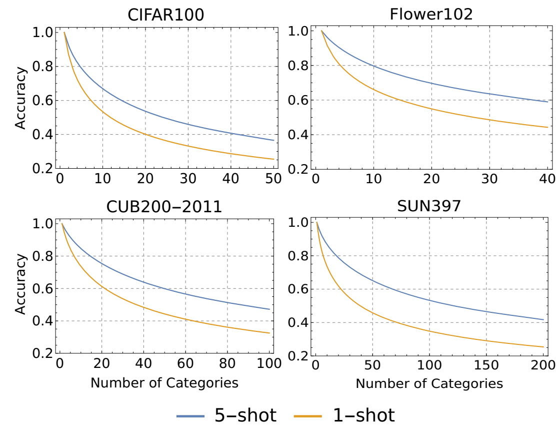

Figure 7. Incremental accuracy on several datasets.

- **SUN397**. The `SUN397` [36] dataset consists of 397 scene categories with 108,754 images. Each category contains at least 100 images. We randomly split it into 150, 47, and 200 categories as the meta-training set, meta-validation set, and meta-testing set, respectively.

We adopted the LSTM-based module generator and the spherical category module for the MetaMG architecture, and followed the experiment settings as in §5.1. Figure 7 illustrates the 1-shot and 5-shot results on the 4 datasets. It is observed from the figure that 5-shot yields significant improvement over 1-shot. Moreover, the accuracy decreases more and more slowly as the number of category increases. Given 5 examples per category, the accuracy with 100 categories on `CUB200-2011` is close to 50%, and the accuracy with 200 categories on `SUN397` is above 40%. This suggests that our MetaMG is promising for few-shot incremental learning.

Our work is the first attempt to perform incremental learning using a few examples for each category without base categories. We do not show benchmarking results with other incremental learning works as their settings are different. Those works include [4], [13], [6], [5], [11] which are not in few-shot learning setting, and [10] which the incremental learning starts from a set of base categories. Instead, as our MetaMG can also be used for few-shot learning, we compare our results with existing few-shot learning works in Section 5.3.

5.3. Few-Shot Classification

This section aims to evaluate our method on the few-shot classification given a fixed number of categories (*i.e.*, 5 or 20) which is a popular task among recent few-shot learning works. The experiments were carried out on the `miniImageNet` dataset. This dataset was collected in [32] and applied as the most popular benchmark dataset for few-shot image classification. It consists of 64, 16, and 20 different categories in the meta-training set $\mathscr{D}_{train}$, meta-validation set $\mathscr{D}_{val}$ and meta-testing set $\mathscr{D}_{test}$, respectively.

	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
Matching Networks [32]	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$	$17.31 \pm 0.22\%$	$22.69 \pm 0.20\%$
Meta-LSTM [26]	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$	$16.70 \pm 0.23\%$	$26.06 \pm 0.25\%$
MAML [9]	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$	$16.49 \pm 0.58\%$	$19.29 \pm 0.29\%$
Meta-SGD [20]	$50.47 \pm 1.87\%$	$64.03 \pm 0.94\%$	$17.56 \pm 0.64\%$	$28.92 \pm 0.35\%$
Relation Network [31]	$50.44 \pm 0.82\%$	$65.32 \pm 0.70\%$	$19.47 \pm 0.25\%$	$33.48 \pm 0.24\%$
MM-Net [3]	$53.37 \pm 0.48\%$	$66.97 \pm 0.35\%$	-	-
SNAIL [23]	$\mathbf{55.71 \pm 0.99\%}$	$\mathbf{68.88 \pm 0.92\%}$	-	-
MetaMG, Ours	$53.78 \pm 0.87\%$	$67.40 \pm 0.68\%$	$\mathbf{23.98 \pm 0.29\%}$	$\mathbf{36.57 \pm 0.29\%}$

Table 1. Benchmark of few-shot classification on `miniImageNet`. Each entry indicates the average accuracy with 95% confidence interval.

Each category contains 600 images.

Regarding the feature extractor, instead of using a pre-trained model, we learned its parameters from scratch on the meta-training set. First, a fully connected layer is appended to the feature extractor. Then we randomized the parameters of the whole model and tuned its parameters on the meta-training set $\mathscr{D}_{train}$ by solving a traditional classification problem using back-propagation. The trained network without the appended fully connected layer is used as the feature extractor. To guarantee a fair comparison with other methods, we only used the 64 training categories of `miniImageNet` to obtain the feature extractor.

For the experiment setup, we followed the same settings as in §5.1 during the meta-training phase. For the meta-testing phase, we measured the classification accuracy under the N-way K-shot settings [20]. Specifically, we randomly selected N categories among all categories in $\mathscr{D}_{test}$ with each category K random training examples and 15 random testing samples. Subsequently, N category modules were generated by feedforwarding the training examples to G and were plugged into the classification network. Finally, the N-class accuracy was evaluated on the testing samples. Such an evaluation was repeated 600 times, and the mean accuracy with 95% confidence intervals was recorded.

Table 1 shows the average classification accuracy among all the compared methods on the `miniImageNet` dataset. For the 5-way classification, the proposed MetaMG achieves near state-of-the-art accuracy, and for the 20-way classification, it achieves the highest reported accuracy among the compared methods. This suggests that even though our MetaMG is not specially designed to solve the few-shot classification problem under a fixed number of categories, it is still promising for the problem.

5.4. Efficiency of Module Generation

To show the efficiency of module generation, we measured the time spent to generate 1 category module with 5 examples on two types of devices including an NVIDIA TITAN Xp GPU and an Intel i7-6800K CPU. The measurement was conducted for 1,000 rounds, and the mean together with the standard deviation were calculated as shown in Table 2. Not surprisingly, module generation on GPU is much faster ($\sim$100x) than on CPU. Most importantly, it takes only about 1.5s to generate a category module on CPU, which means that a category module can be generated in almost real-time for practical applications using a common CPU computer. Compared to other incremental learning methods such as iCaRL [28] that require to retrain the model with plenty of samples from new and old categories, the time for adding new categories into the system using MetaMG is significantly reduced. The ability of using CPU for real-time incremental learning with only a few samples will help to solve many real-world problems. For example, when a robot is going to a new place, it may have to learn to recognize the new place quickly without collecting a lot of samples from the new place and redo the training process. For visual recognition of products in an unmanned supermarket for a grab and go kind of application, MetaMG could be a potential solution to register new products incrementally and remove obsolete products quickly and easily.

Device	GPU	CPU
Time (ms)	13.64 ± 0.76	1546.30 ± 23.97

Table 2. Time for generating 1 category module using GPU and CPU, respectively. The table entry indicates the average time with the standard deviation in milliseconds.

6. Conclusion

We have presented a meta-learning method called Meta Module Generation (MetaMG) to address the few-shot incremental learning problem. Through optimization, the module generator is capable to generate a category module from a few examples for a scalable classification network to recognize a new category. The module generation process is fast as the training examples only need to feedforward through the module generator network once. Comprehensive experiments on 4 datasets have shown that the proposed method achieves promising accuracy for incremental class learning using only a few examples from each category. Further experiments conducted on the `miniImageNet` dataset have suggested that even though our MetaMG is not specially designed for the N-way K-shot learning problem, it could still achieve the cutting edge performance.

References

[1] https://github.com/xialeiliu/awesome-incremental-learning. 3

[2] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36, 2012. 1

[3] Q. Cai, Y. Pan, T. Yao, C. Yan, and T. Mei. Memory matching networks for one-shot image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4080–4088, 2018. 8

[4] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018. 3, 7

[5] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 3, 7

[6] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 3, 7

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. A deep convolutional activation feature for generic visual recognition. arXiv preprint. *arXiv preprint arXiv:1310.1531*, 2013. 1

[8] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006. 1, 2

[9] C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*, 2017. 1, 2, 3, 8

[10] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018. 3, 7

[11] T. L. Hayes, N. D. Cahill, and C. Kanan. Memory efficient experience replay for streaming learning. *arXiv preprint arXiv:1809.05922*, 2018. 3, 7

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[13] R. Kemker and C. Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017. 3, 7

[14] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015. 1, 2

[15] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 7

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[17] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011. 1, 2

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[19] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 2017. 1, 3

[20] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017. 1, 2, 3, 8

[21] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continuum learning. In *NIPS*, 2017. 1, 3

[22] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109 – 165. Academic Press, 1989. 3

[23] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018. 8

[24] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 7

[25] S. Qiao, C. Liu, W. Shen, and A. L. Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018. 3

[26] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 3, 4, 8

[27] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *NIPS*, 2017. 1, 3

[28] S.-A. Rebuffi, A. Kolesnikov, and C. H. Lampert. iCaRL: Incremental Classifier and Representation Learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017. 1, 3, 6, 8

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[30] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 1, 2

[31] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 8

[32] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016. 1, 2, 7, 8

[33] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 6

[34] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 627–637, 2017. 4

[35] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu. Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*, 2018. 1, 3

[36] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010. 7

[37] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018. 1, 3

[38] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017. 4