L2AE-D: Learning to Aggregate Embeddings for Few-shot Learning with Meta-level Dropout

Heda Song[a,*], Mercedes Torres Torres[b], Ender Özcan[a], Isaac Triguero[a]

[a] *Computational Optimisation and Learning Lab, School of Computer Science, University of Nottingham, Nottingham NG8 1BB, United Kingdom*
[b] *Computer Vision Laboratory, School of Computer Science, University of Nottingham, Nottingham, UK*

Abstract

Few-shot learning focuses on learning a new visual concept with very limited labelled examples. A successful approach to tackle this problem is to compare the similarity between examples in a learned metric space based on convolutional neural networks. However, existing methods typically suffer from meta-level overfitting due to the limited amount of training tasks and do not normally consider the importance of the convolutional features of different examples within the same channel. To address these limitations, we make the following two contributions: (a) We propose a novel meta-learning approach for aggregating useful convolutional features and suppressing noisy ones based on a channel-wise attention mechanism to improve class representations. The proposed model does not require fine-tuning and can be trained in an end-to-end manner. The main novelty lies in incorporating a shared weight generation module that learns to assign different weights to the feature maps of different examples within the same channel. (b) We also introduce a simple meta-level dropout technique that reduces meta-level overfitting in several few-shot learning approaches. In our experiments, we find that this simple technique significantly improves the performance of the proposed method as well as various state-of-the-art meta-learning algorithms. Applying our method to few-shot image recognition using Omniglot and miniImageNet datasets shows that it is capable of delivering a state-of-the-art classification performance.

Keywords: Few-shot learning, Meta-learning, Metric-learning, Embedding aggregation, Attention mechanism, Meta-level dropout

1. Introduction

In recent years, deep learning techniques have dramatically been developed achieving high classification accuracy on visual recognition systems [1, 2]. These

*Corresponding author
Email address: heda.song@nottingham.ac.uk (Heda Song)

techniques usually require a large amount of labelled data to learn an appropriate model while they struggle when provided with very few data. However, in many real-world visual recognition tasks, such as images of new species or rare diseases, it is impractical to collect much labelled data. This highly restricts the successful application of deep learning. In addition, their learning style is typically not consistent with a human visual system that can generalise a new visual concept after seeing a few images based on previous experience. To address these issues, the computer vision community has raised enthusiasm for the challenge of learning from very few data, also known as few-shot learning [3, 4].

Few-shot learning typically aims to learn a new visual concept from a limited number of labelled examples. Overfitting can easily occur using conventional machine learning algorithms in such few-shot regime. To avoid this, we need a learning approach with a high generalisation ability. Inspired by the way humans are capable of quickly learning based on accumulated experience, many meta-learning approaches for few-shot learning have recently been proposed. In general, these methods learn a meta-learner to extract meta-knowledge from a distribution of few-shot learning tasks and further use it to assist unseen tasks. The extracted meta-knowledge can be represented by different algorithm components, such as a general feature extractor [5, 6, 7], a distance metric [8], promising initial model parameters [9, 10, 11, 12], optimisation strategies [13], a model parameter predictor [14, 15, 16], a example generator [17, 18, 19], scale and/or shift vectors for activation adaptation [20], or label propagation [21, 22, 23].

Although these approaches achieve a good performance, they still suffer from several issues. Some methods need to fine-tune the base model when executing target tasks [9, 10, 11, 12, 13]. Others introduce complex model architectures or external memory, which require more computing resources [22, 21, 23, 14, 16]. Generative model-based approaches learn to generate more artificial examples, but they may create some non-informative examples when provided with noisy training examples [17, 18, 19]. Metric learning based approaches are straightforward and efficient [7]. They use Convolutional Neural Networks (CNNs) to extract the embeddings of examples, which are represented by a set of feature maps, and make predictions by comparing the similarities between embeddings. However, they seldom consider outliers in a class or borrow useful features from other classes [5, 6, 7, 8]. Due to the limited amount of data in few-shot learning, as presented in 1Fig. 1, the training examples may inherently contain uncertainties, such as the outliers shown in Fig. 1(a). If we simply use the mean of each class's embeddings as the class representative, as performed in [7], the possible outliers may force the representative to deviate from the class centre in the embedding space. Therefore, it is necessary to appropriately handle the effect of outliers. However, an outlier may actually contain some useful features, which could help strengthen part of the class representative. Similarly, even the examples of different classes may share some similar features as shown in Fig. 1(b), which could be used to help them to be distinguished from other classes in multi-class classification, particularly in 1-shot cases. An concrete illustration can be found in Figure 2, which shows a 5-classes (5-way) classification task that

Figure 1: An illustration of the motivation of this study. Each embedding (rounded rectangle) consists of three feature maps (coloured squares), with outliers shown in dashed borders. (a) Binary classification with five training examples per class. We show the real class centres in the embedding space (solid circles) and the mean of each class' embeddings (hollow circle). (b) 4-class classification with one training example per class. Dashed arrows link similar feature maps in the embeddings from different classes.

targets at distinguishing 'goose', 'bird', 'bus', 'crab' and 'jellyfish'. From the given five training examples, we can see that the shape of heads of 'goose' and 'bird' are similar. Assuming that one of the channels in the last convolutional layer corresponds to the shape of head, weighted aggregating the feature maps of 'goose' and 'bird' in that channel could help 'goose' and 'bird' to be better distinguished from 'bus', 'crab' or 'jellyfish'. Overall, our goal is to reduce the impact of outliers and use as much as useful information as possible in few-shot learning.

In addition, these meta-learners may also suffer from overfitting. Although, meta-learners are trained on different few-shot learning tasks, they may consist of overlapped classes, because there are limited number of classes in the meta-training dataset and some of them are similar. For example, there are only 100 classes of objects in miniImageNet [13] and some of them are different breeds of dogs. Thus, meta-learners could be trained to perform well on meta-training tasks and not generalise well on meta-testing tasks comprised of unseen classes.

To tackle the above issues, we propose L2AE-D (Learning to Aggregate Embeddings with Meta-level Dropout), a novel meta-learning approach for few-shot learning that learns to aggregate embeddings with meta-level dropout. L2AE-D learns a CNNs based feature extractor and a channel-wise attention mechanism in an end-to-end manner. The feature extractor is used to transform the input images into discriminative embeddings. The channel-wise attention mechanism is learned to assign larger weights to useful feature maps and smaller weights to noisy ones of different embeddings within the same channel. We propose different learning strategies for one-shot and few-shot tasks aiming to effectively exploit the few training embeddings. We also introduce a meta-level dropout technique into the meta-training process to prevent meta-level overfitting. We test this technique in several representative meta-learning approaches and it significantly improves their performance. We evaluate the proposed method on

Omniglot [4] and miniImageNet [13] datasets, and it achieves either competitive or state-of-the-art performance on various few-shot learning tasks.

The remainder of this paper is organised as follow. Section 2 provides an investigation of the recent progress in few-shot learning and some related works on attention and dropout. Section 3 describes our proposed method. The experimental results are shown in Section 4. The conclusion is discussed in Section 5.

2. Related Work and Motivation

Our method falls into the research field of few-shot learning. Section 2.1 investigates the recent progress in this field, and explains our motivations. Besides, L2AE-D is based on an attention mechanism and dropout. We briefly review these techniques used in image classification in Section 2.2 and Section 2.3, respectively.

2.1. Few-shot Learning approaches

Most recent works tackle few-shot learning by meta-learning due to its high generalisation ability. In general, they learn a meta-learner to extract meta-knowledge from a number of few-shot learning tasks and use it to assist in unseen ones. Depending on the type of meta-knowledge, these methods can be broadly classified into three categories.

- **Fast parametrisation based approaches:** Approaches in this class aim to learn a fast parametrisation strategy for quickly fine-tuning the base learner to adapt to new few-shot learning tasks. The most representative method, Model-Agnostic Meta-Learning (MAML) [9], learns the model's initial parameters that can be adapted to task-specific model parameters by a few gradient descent steps based on few examples. MAML has been extended in various ways, such as introducing a first-order gradient to reduce the computational burden [10], learning model's initial parameters together with optimisation strategies (Meta-learner-LSTM [13]), to further accelerate the fine-tuning process, choosing a subset of model parameters to fine-tune in order to make the model more task-specific [12], modelling a distribution of prior model parameters to handle the inherent uncertainty of few-shot learning [11]. Rather than fine-tune the base model, some other methods learn a meta-learner to directly predict the parameters of the base model [14, 16, 15]. One of them learns to predict the parameters of the fully-connected layer from the activations (Activation2Weights) [15]. These methods can be faster while some of them use external memory, which require more resources to store the adequate historical information. Instead, our approach executes target few-shot tasks in a feed-forward manner without external memory, which is quick and does not require additional resources.

- **Generative model based approaches:** These approaches learn to generate artificial examples to compensate the lack of training data. The Neural Statistician approach learns to produce statistics of a dataset, such as mean or variance, which are used to specify a Gaussian distribution for generating data [17]. Other methods introduce generative adversarial networks to learn sharper decision boundaries (MetaGAN) [18] or model the latent distribution of novel classes [19]. These meta-learners generate fake examples to assist few-shot learning tasks. However, these examples could be non-informative when the few training examples are not representative. Conversely, our method learns to aggregate useful information and suppress noisy information, which can be more stable.

- **Metric learning approaches:** The approaches in this class learn to compare the similarity between examples in a learned metric space. Most approaches learn a general feature extractor, which is usually represented by CNNs [5, 6, 7, 8, 22, 21, 23, 24], to transform examples into embeddings and then compute the similarity between each pair of training and query embeddings based on weighted L1 distance (Siamese Nets [5]), cosine distance (Matching Nets [6]), Euclidean distance (Prototypical Networks (ProtoNets) [7]) or a learned distance metric (Relation Network (RN) [8]). Finally, the queries can be classified by a linear [5, 8], a k-nearest neighbours [7] or a weighted k-nearest neighbours [6] classifier. Some other approaches in this branch propagate label information from training examples to unlabelled query examples based on similarity [22, 21, 23]. Specifically, Transductive Propagation Network (TPN) [23] and Graph Neural Networks (GNNs) [22] learn a graph construction module and propagate labels within the graph. Another approach combines temporal convolutions and soft attention to propagate label information [21]. Improved Prototypical Networks (IPN) [25] improves ProtoNets by considering intra-class importance and adopting a distance scaling strategy. These methods also aggregate embeddings, but they treat each embedding as a whole. Instead, we aggregate feature maps in each channel, which could make use of more information, even from an outlier or an example from different classes. Some other methods hybridise the metric learning and fast parameterisation approaches [26, 27], which combines the strengths of these two approaches while also inherits the weaknesses of them.

2.2. Attention Mechanisms

An attention mechanism aims to tell a machine learner where to focus, which is inspired by the human perception system. It has been extensively studied these years and applied to various machine learning tasks, such as machine translation [28] or image caption [29]. Recently, a few works have introduced attention mechanisms to CNNs for computer vision tasks [30, 31, 2, 29]. Our method is related to the ones presented in [31, 2, 29], which learn a channel-wise attention module. Our attention module is different from theirs in four aspects. First, they target standard learning tasks, in which training and testing sets

include the same classes. Whereas, we aim to address more challenging few-shot learning problems in which there is no overlap between the classes in meta-training and meta-testing set. Second, their aim is to emphasise useful features by refining feature maps, while our goal is to handle uncertainty and fully use the few training examples by aggregating feature maps. Third, we carry out an attention mechanism along a different dimension. Specifically, they feed the whole embedding of a sample into the attention module and generate channel-wise weights for this particular sample. Instead, we feed the feature maps in a specific channel of different training samples and generate weights for that channel of different samples. Fourth, they apply multi-layer perceptrons to learn to assign weights, while we use CNNs as a meta-learner. It is noteworthy that several meta-learning approaches also introduce the attention mechanism to tackle few-shot learning problems [6, 21, 16]. However, we use it in different ways and for different purposes. The approaches in [6, 21] use attention to propagate labels based on the similarities between a query and training examples. The method in [16] use attention to generate a classifier's weights for unseen classes. In contrast, our attention mechanism is used to assign different weights to the feature maps of different examples, aiming at handling uncertainty and fully using the few training examples.

2.3. Dropout

Dropout is a simple way to prevent neural networks from overfitting [32]. The key idea is to randomly drop part of the units of neural networks during training and use the whole networks for testing, which can also be seen as a form of model averaging. Although it has been widely used in neural networks training, it is seldom applied to convolutional layers in CNNs. The reason is that the shared-filter architecture dramatically reduces the number of model parameters which reduce the model's capacity to overfit [32]. Still, the experimental results in [32] show performing dropout in convolutional layers can prevent overfitting and further improve the performance on image recognition tasks. Another method proposes a specific dropout technique called SpatialDropout for CNNs by randomly dropping the entire feature maps [33]. Different from applying dropout on common machine learning tasks, we aim at adapting dropout to the meta-learning framework to prevent the meta-learner from overperforming on some particular tasks rather than samples, so that the learned meta-knowledge could generalise well on unseen tasks during meta-testing. Since each task in meta-training includes a whole base learning process that extracts knowledge from the training examples and infers the labels of testing examples, we perform dropout on CNNs for each task at the meta-level by randomly dropping part of neurons for both training and testing examples during meta-training. The meta-learner is fixed during meta-testing.

3. Methodology

In this section, we describe the proposed learning to aggregate embeddings with meta-level dropout (L2AE-D) method. We define the problem of few-shot

learning in Section 3.1. Then, Section 3.2 describes our model consisting of an embedding, attention and distance module. The specific model architecture is discussed in Section 3.3. Finally, Section 3.4 presents how we perform meta-level dropout.

3.1. Problem Set-Up

Few-shot classification problems [3] aim to classify testing examples into one of C unique classes based on K labelled training examples for each of C class, which is called C-way K-shot classification. For each C-way K-shot classification task, the training set $D_{train} = \{(x_i, y_i)\}_{i=1}^{K \times C}$ contains $K \times C$ training examples and the testing set D_{test} contains n testing examples that share the same label space with D_{train}. In conventional machine learning, we could train a learner to predict the label for each testing example in D_{test} based on D_{train}. However, the learner cannot be trained effectively based on such few training examples.

A number of approaches including our method tackle the problem by meta-learning. Typically, we have three meta-sets, meta-training set $D_{meta-train}$, meta-validation set $D_{meta-validation}$ and meta-testing set $D_{meta-test}$. Their respective label space is disjoint from each other. The meta-training set $D_{meta-train}$ is used for training a meta-learner that generalises well across a distribution of few shot learning tasks, which is represented as $D_{meta-train} = \left\{ \left(D_{train}^{j}, D_{test}^{j} \right) \right\}_{j=1}^{N}$. The $D_{meta-validation}$ set is used to select suitable hyper-parameters of the meta-learner. We can evaluate the meta-learner on the $D_{meta-test}$ set.

Since the $D_{meta-train}$ set includes a large amount of different few shot classification tasks, it is best to train the meta-learner in an episode-based manner as proposed in [6]. In each meta-training iteration, a single few shot classification ($\left(D_{train}^{j}, D_{test}^{j} \right), j \in [1, N]$) is sampled to train the meta-learner based on its performance on D_{test}^{j}. We can also introduce the strategy of batch meta-training as done in [9]. Thus, in each meta-training iteration, we sample a batch of few shot classification to train the meta-learner.

3.2. The L2AE-D Model

L2AE-D can be divided into three modules: embedding module f_φ, attention module g_ϕ and distance module as shown in Fig. 2 and Fig. 3. The attention module is different for the 1-shot and K-shot cases. Fig. 2 shows our strategy for C-way 1-shot classification and Fig. 3 depicts our strategy for C-way K-shot classification. Pseudocode for the training process of L2AE-D is provided in Algorithm 1.

- **Embedding module:** This module aims to extract features of each input image and transform it into embeddings. For each input example x_i belonging to the c-th class, we feed it into the embedding module f_φ to

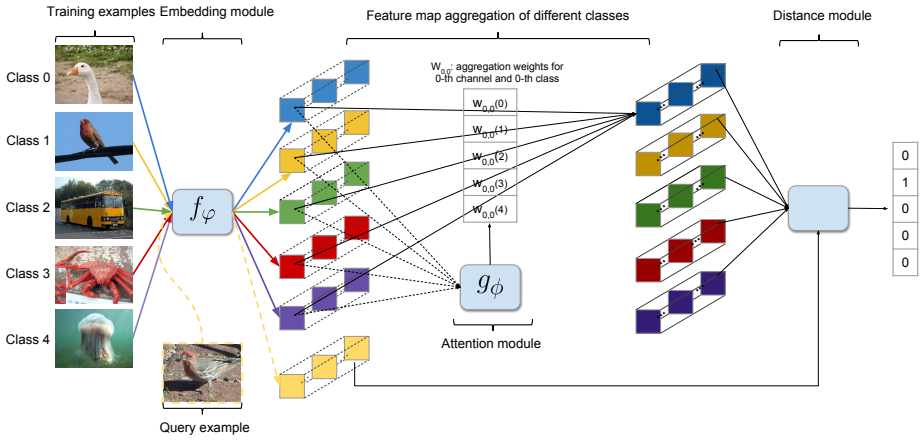

Figure 2: 5-way 1-shot classification with L2AE-D: (1) Training samples are transformed by f_φ into embeddings (set of feature maps shown in coloured squares); (2) To strengthen the first feature map for the first class, we put it in the first channel and the other feature maps in the others, then we feed the concatenated 5-channel feature maps into g_ϕ to generate aggregation weights; (3) The 5 feature maps are aggregated based on the generated weights; (4) To make predictions, we feed a query into f_φ, then compare its embedding with the aggregated training embeddings in the distance module. This outputs a one-hot vector representing the predicted label of the query.

generate an embedding $E_{i,c} = \left\{e^1_{i,c}, e^2_{i,c}, ..., e^n_{i,c}\right\}$, which comprises n feature maps $e^k_{i,c} \in R^{l\times l}$ with the size of $l \times l$. Then, the training embeddings are fed into the attention module.

- **Attention module:** This module is used for generating aggregation weights of the feature maps in a channel-wise manner as shown in Fig. 4. Also, it is shared among different channels. We use two different strategies to do aggregation for 1-shot and K-shot tasks as shown in Fig. 2 and Fig. 3, respectively. For K-shot C-way tasks, we aggregate the feature maps of K training embeddings in the same class to be the class-representative feature maps. For the k-th channel, we join the corresponding feature maps of K training embeddings in the c-th class as $F_{k,c} = \left\{e^k_{1,c}, e^k_{2,c}, ..., e^k_{K,c}\right\}$. For C-way 1-shot tasks, we aggregate the feature maps of C training embeddings from different classes, since there is only one training embedding in each class. To generate aggregation weights for the c-th class in the k-th channel, we concatenate the corresponding feature maps of C training embeddings from different classes as $F_{k,c} = \left\{e^k_{1,c}, e^k_{1,1}, ..., e^k_{1,C}\right\}$. Note that we locate $e^k_{1,c}$ in the first channel and other $C-1$ feature maps behind randomly in $F_{k,c}$.

  Next, the concatenated feature maps are inputted into CNNs based attention networks g_ϕ, which produce the aggregation weights $w_{k,c} \in R^K$

Algorithm 1 The training of L2AE-D

Require: Meta-training set $D_{meta-train}$

Require: The number of classes C and the number of training exampels K in each task, the number of the channels n in the last convolutional layer of the Embedding module, the metric-based classifier M

randomly initialize φ and ϕ
while not done **do**
  $D_{train}, D_{test} \leftarrow$ randomly sample from $D_{meta-train}$, $D_{train} = \{(x_i, y_i)\}_{i=1}^{K \times C}$, $D_{test} = \{(x_i, y_i)\}_{i=1}^{m}$
  for $i = 1$ **to** K **do**
    for $c = 1$ **to** C **do**
      $E_{i,c} = f_{\varphi}(x)$, x from D_{train}, $E_{i,c} = \left\{e_{i,c}^1, e_{i,c}^2, ..., e_{i,c}^n\right\}$
    end for
  end for
  for $c = 1$ **to** C **do**
    for $k = 1$ **to** n **do**
      if K==1 **then**
        $F_{k,c} = \left\{e_{1,c}^k, e_{1,1}^k, ..., e_{1,C}^k\right\}$
      else
        $F_{k,c} = \left\{e_{1,c}^k, e_{2,c}^k, ..., e_{K,c}^k\right\}$
      end if
      $w_{k,c} = g_{\phi}(F_{k,c})$
      $\widetilde{e}_c^k = w_{k,c} \cdot F_{k,c}$
    end for
    $\widetilde{E}_c = \left\{\widetilde{e}_c^1, \widetilde{e}_c^2, ..., \widetilde{e}_c^n\right\}$
  end for
  $\widetilde{E} = \left\{\widetilde{E}_1, \widetilde{E}_2, ..., \widetilde{E}_c\right\}$
  for $q = 1$ **to** m **do**
    $E_q = f_{\varphi}(x_q)$, x_q from D_{test}
  end for
  $\mathcal{L}_{test} = \sum_{q=1}^{m} \mathcal{L}(M(\widetilde{E}, E_q), y_q)$
  Update φ and ϕ based on $\bigtriangledown_{\varphi,\phi} \mathcal{L}_{test}$
end while

for K-shot tasks or $w_{k,c} \in R^C$ for 1-shot tasks. After that, we can aggregate the feature maps $F_{k,c}$ based on the weights $w_{k,c}$. The aggregated feature map of the c-th class in the k-th channel is represented by $\widetilde{e}_c^k = w_{k,c} \cdot F_{k,c} \in R^{l \times l}$. In the end, we concatenate the aggregated feature maps in all channels and obtain a new embedding $\widetilde{E}_c = \left\{\widetilde{e}_c^1, \widetilde{e}_c^2, ..., \widetilde{e}_c^n\right\}$ for the c-th class. The new training embedding set is then represented by $\widetilde{E}_{train} = \left\{\widetilde{E}_c\right\}_{c=1}^{C}$, in which $\widetilde{E}_c$ can be seen as a class representative.

- **Distance module:** This module is used to measure the distance between the embeddings of query examples, $E_q = f_{\varphi}(x_q)$, and the aggregated

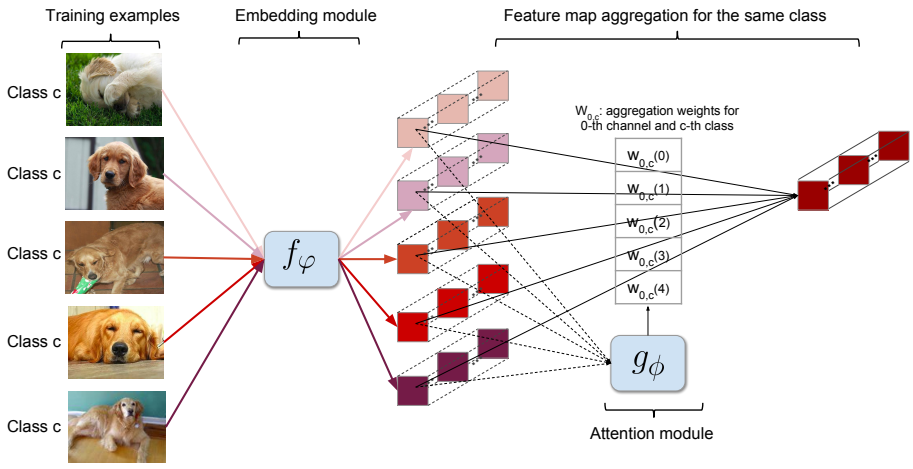

Figure 3: C-way 5-shot classification with our approach. L2AE-D aggregates embeddings for each class: (1) The training examples are transformed by f_φ into embeddings represented by a set of feature maps; (2) For each channel, we collect the feature maps and feed them into the attention module; (3) The feature maps are concatenated in depth and fed into g_ϕ to generate aggregation weights; (4) The feature maps are then aggregated based on the generated weights to represent a feature for this class.

embeddings $\widetilde{E}_{train}$. Following [7], we choose the Euclidean distance as the distance function $d() : R^M \times R^M \rightarrow [0, +\infty)$. Thus, the distance between E_q and $\widetilde{E}_c$ is computed by $d(E_q, \widetilde{E}_c)$.

- **Loss function:** We consider cross-entropy loss to train our model. First, the softmax function is applied over the negative distance between the query embeddings and aggregated training embeddings as follows:

$$p_{\varphi,\phi}(y = c \mid x_q) = \frac{exp(-d(E_q, \widetilde{E}_c))}{\sum_{c'} exp(-d(E_q, \widetilde{E}_{c'}))}$$

Then the loss function can be formulated as

$$L(\varphi, \phi) = -\sum_{q=1}^{N_q} \sum_{c=1}^{C} log\, p_{\varphi,\phi}(y = c \mid x_q)$$

where N_q is the number of query examples in each training epoch.

3.3. Model Architecture

L2AE-D follows the same architecture as the embedding module in prior approaches [6, 7], which contain 4 convolutional blocks. Each block is composed of a 3×3 convolution with 64 filters, followed by batch normalisation (BN) [34], a ReLU nonlinearity and a 2×2 max-pooling. For Omniglot, due to the small

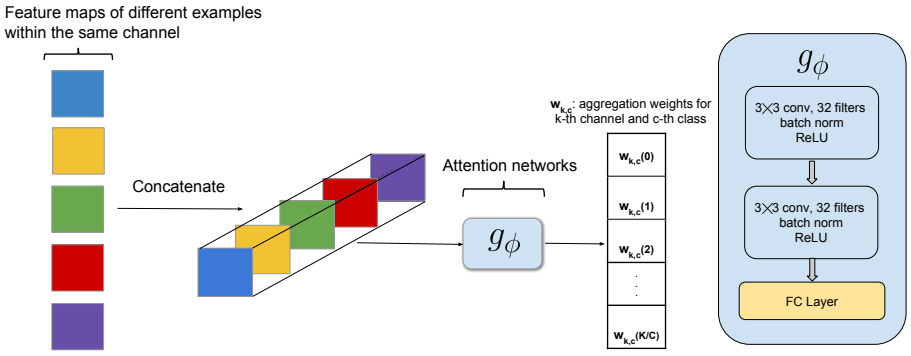

Figure 4: The architecture of the attention module.

size of the input images, we feed the embeddings into the CNNs based attention module and we remove the max-pooling layer from the last convolutional block. The architecture of our attention module showed in Fig. 4 consists of 2 convolutional blocks and a fully connected (FC) layer. Each convolutional block in this module comprises a 3×3 convolution with 32 filters, followed by batch normalisation, a ReLU nonlinearity. The FC layer results in a m-dimensional output, which represent the aggregation weights for the m feature maps. For K-shot tasks, we use the softmax function after the FC layer since we aim to assign positive weights, whose sum is 1, to the m embeddings of the same class.

3.4. Meta-level Dropout

Most meta-learning approaches use multi-layer CNNs to extract features on few-shot learning. As discussed before, we incorporate the dropout technique in the meta-level to tackle meta-level overfitting. Specifically, we randomly drop part of units of CNNs for both the training and testing examples in each few-shot learning task during meta-training. During meta-testing, we use the whole trained CNNs to extract features on both training and testing examples. Note that the dropout in the convolutional layers works in a different way from that in the fully connected layers, because the kernel weights are shared with the units at different spatial positions, so that, the weights would still be updated by back-propagation even if part of units are dropped. The actual effect of performing dropout in the convolutional layers is to scale the learning rate [33], which can also help with preventing overfitting. We find this technique can improve several meta-learning approaches significantly according to the experimental results in Section 4.4.

4. Experiments

This section evaluates our method on the widely studied datasets Omniglot [4] and miniImageNet [13]. The experimental setup is provided in Section

4.1. Section 4.2 and 4.3 analyse the results on Omniglot and miniImageNet, respectively. We also introduce a meta-level dropout technique into several promising few-shot learning approaches and we test its behaviour on miniImageNet in Section 4.4. Section 4.5 visualises the working of L2AE-D based on T-distributed Stochastic Neighbor Embedding (t-SNE) [35]. The code for L2AE-D is available online[1].

4.1. Experimental setup

This section introduces the details of the two used datasets and the configurations followed to test the behaviour of L2AE-D against the state-of-the-art.

- **Omniglot** consists of 1,623 handwritten characters collected from 50 alphabets. There are 20 examples of each character, which are drawn by different people. We augmented the datasets with rotations with multiple 90 degrees as proposed by [36] to get 6492 classes. Following [9], we randomly select 1,200 classes (4,800 classes after augmentation) for meta-training, 100 classes (400 classes after augmentation) for meta-validation, and the remaining 323 (1292 classes after augmentation) for meta-testing. All the input images are resized to 28×28 as suggested by [6] to get a suitable sized embedding.

- **miniImageNet** was proposed by [6] derived from the original ILSVRC-12 dataset [37]. It comprises 100 classes of colour images with 600 of each (60,000 in total). In our experiments, we use the widely used splits proposed by [13], which divides the 100 classes into 64 for meta-training, 16 for meta-validation and 20 for meta-testing. All the input images are resized to 84×84 as done by most few-shot learning approaches [13, 7, 9]. Note that the existing approaches use different tools to resize the images in miniImageNet. We use the library provided by OPENCV [38] following [23].

To allow for fair comparisons with the current state-of-the-art, we maintain the different experimental setups reported on Omniglot (20-way 5-shot, 20-way 1-shot, 5-way 5-shot, 5-way 1-shot) and miniImageNet (5-way 5-shot, 5-way 1-shot). All experiments are performed using TensorFlow [39] on a Titan V GPU.

- **Meta-training:** Following most existing methods [7, 9, 8], we train our model in an episode-based manner and use a meta-batch size of 4, which means in each episode we randomly sample 4 C-way K-shot classification tasks to train the model. For each few-shot task, besides the $C \times K$ training examples, we randomly sample 5 or 15 query examples per class to compute the loss for Omniglot and miniImageNet, respectively. We train our model with Adam [40] with a initial learning rate of 0.001 in an

[1] `github.com/Heda-Song/L2AE-D`

end-to-end manner [7, 8]. We cut the learning rate in half every 20,000 episodes to stabilise training and use meta-validation set to choose the best-performing model for meta-testing. It is noteworthy that existing methods conduct BN in different ways. As pointed out in [13], there would be a bad impact on performance if we use the global BN statistics accumulated from meta-training set to normalise batches of examples in meta-testing set, since there is no overlap between the classes in these two sets. Thus, we perform BN on each batch of examples following [9, 8]. Specifically, for each task during both meta-training and meta-testing, we use each batch's statistics to normalise the training or query examples, which can be seen as a transductive way.

- **Meta-testing:** To be consistent with the existing few-shot learning approaches, we evaluate our model on 1,000 or 600 randomly sampled C-way K-shot classification tasks, which consist of $C \times K$ training examples and 5 or 15 query examples per class, for Omniglot and miniImageNet, respectively. We report the average accuracy on these tasks with 95% confidence intervals. However, we find that most previous methods only use a single seed to randomly sample a batch of testing tasks and report the average accuracy. Since there are a large number of tasks in meta-testing, they may sample a large proportion of easy-to-classify or difficult-to-classify tasks using different seeds, which would lead to a result with high variance. To get a more reliable result, we use 10 different seeds to randomly sample different batches of testing tasks for 10 times and report the best, worst and average accuracy. Note that the existing methods are not strictly comparable since their experimental settings are not consistent with each other.

4.2. Analysis of the Results on Omniglot

We compare our approach against state-of-the-art methods from each family of few-shot learning approaches that provide experimental results on Omniglot. They are MAML [9] from fast-parametrisation based approaches, Neural Statistician [17] and MetaGAN [18] from generative model based approaches, and Siamese Nets [5], Matching Nets [6], ProtoNets [7], GNN [22] and RN [8] as metric learning approaches. Their reported experimental results and ours are shown in Table 1. In general, all the methods perform worse on 20-way tasks than 5-way tasks, which shows 20-way tasks are more difficult. L2AE-D achieves state-of-the-art performance on 20-way tasks even in the worst case and competitive results on 5-way tasks. Besides, our results are very stable, since the differences between the best and worst accuracies for all the tasks are no more than 0.2%. On 5-way 5-shot and 20-way 1-shot tasks, L2AE-D mostly obtains the best performance on different batches of tasks (using different seeds) since the average and best accuracy are the same. MetaGAN performs better on 5-way tasks by generating more examples to assist RN while it improves marginally upon RN.

Model	FT	5-way Acc.		20-way Acc.	
		1-shot	5-shot	1-shot	5-shot
Siamese Nets [5]	N	96.7%	98.4%	88.0%	96.5%
Matching Nets [6]	N	98.1%	98.9%	93.8%	98.5%
Neural Statistician [17]	N	98.1%	99.5%	93.2%	98.1%
ProtoNets [7]	N	98.8%	99.7%	96.0%	98.9%
GNN [22]	N	99.2 %	99.7%	97.4%	99.0 %
MAML [9]	Y	98.7±0.4%	**99.9±0.1%**	95.8±0.3%	98.9±0.2%
RN [8]	N	99.6±0.2%	99.8±0.1%	97.6±0.2%	99.1±0.1%
MetaGAN [18]+**RN** [8]	N	**99.67±0.18%**	99.86±0.11%	97.64±0.17%	**99.21±0.1%**
L2AE-D (worst)	N	99.2±0.2%	99.7±0.1%	**97.7±0.2%**	**99.2±0.1%**
L2AE-D (average)	N	99.3±0.2%	99.8±0.1%	**97.8±0.2%**	**99.2±0.1%**
L2AE-D (best)	N	99.4±0.2%	99.8±0.1%	**97.8±0.2%**	**99.3±0.1%**

Table 1: Few shot classification results on Omniglot averaged over 1,000 testing tasks. It is noteworthy we only compare our method with prior approaches that are based on the same type of model, 4-layer CNNs. The ± shows 95% confidence over tasks. FT stands for fine-tuning. The best-performing results are highlighted in bold. All the results are rounded to 1 decimal place other than MetaGAN's that are reported with 2 decimal places.

4.3. Analysis of the Results on miniImageNet

The existing few-shot learning approaches typically use two types of models to extract features, 4-layer CNNs [9, 7, 8] and deep residual networks [21, 19, 20]. Deep residual network [1] is a kind of neural network with skip connections and more hidden layers, which has a more complex architecture but better representation capability compared to 4-layer CNNs. For a fair comparison, we compare our method with prior approaches that are based on the same type of model, 4-layer CNNs. As before, we choose state-of-the-art methods from each family that provide experimental results on miniImageNet. They are Meta-learner-LSTM [13], MAML [9] and Activation2Weights [15] from fast-parameterisation based approaches, MetaGAN [18] from generative model based approaches, Matching Nets [6], ProtoNets [7], GNN [22], RN [8] and TPN [23] from metric learning approaches. Their reported experimental results and ours are shown in Table 2. L2AE-D achieves state-of-the-art performance on 5-way 5-shot classification even in the worst case. On 5-way 1-shot classification, L2AE-D (average) provides the second best result, which is slightly worse than Activations2Weights. However, the feature extractor of Activations2Weights is trained with more classes (higher ways) and more queries in each meta-training episode. In contrast, our model is trained on 5-way classification with 15 queries per episode, which is consistent with the setting of most existing approaches. Besides, TPN obtains very competitive results on 1-shot and 5-shot classification. However, TPN is a transductive method that requires unlabelled data to propagate labels and its performance is affected by the number of query examples. Even though we use query batch statistics to normalise the query examples in a transductive way, we can simply modify it into an inductive way by using training batch statistics to normalise the query data without decreasing the performance much.

4.4. Analysis of the e ect of Meta-level Dropout

Since the augmented Omniglot dataset includes much more classes (4,800) than miniImageNet (64) in the meta-training set, the meta-learners do not suffer much from meta-level overfitting on Omniglot. Therefore, we focus on miniImagent to analyse the effect of meta-level dropout through 5-way 1-shot tasks. We introduce meta-level dropout into several representative meta-learning approaches, including MAML [9], ProtoNets [7] and RN [8]. Specifically, we use their provided code and add dropout in the middle two convolutional layers before max-pooling with the keep probability of 0.5, because there is more co-

4-316(324-317l)-45ltao316()28(y17l)-433(c439ev)24Td [(17l350(induc)-45)-377163(and)-47163s.955 -356(with)(

Table 2: Few-shot classification results on miniImageNet averaged over 600 tests based on 4-layer CNNs. It is noteworthy we only compare our method with prior approaches that are based on the same type of model, 4-layer CNNs. The ± shows 95% confidence over tasks. FT stands for fine-tuning. The best-performing results are highlighted in bold.

Model	**FT**	**5-way Acc.**	
		1-shot	5-shot
Matching Nets [6]	N	43.56 ± 0.84%	55.31 ± 0.73%
Meta-Learner-LSTM [13]	N	43.44 ± 0.77%	60.60 ± 0.71%
MAML (1 query) [9]	Y	48.70 ± 1.84%	63.11 ± 0.92%
ProtoNets [7]	N	49.42 ± 0.78%	68.20 ± 0.66%
GNN [22]	N	50.33 ± 0.36%	66.41 ± 0.63%
RN [8]	N	50.44 ± 0.82%	65.32 ± 0.70%
MetaGAN [18]+**RN** [8]	N	52.71 ± 0.64%	68.63 ± 0.67%
TPN [23]	N	53.75 ± 0.86%	69.43 ± 0.67%
Activations2Weights [15]	N	**54.53 ± 0.40%**	67.87 ± 0.70%
L2AE-D (worst)	N	53.03 ± 0.84%	**69.53 ± 0.65%**
L2AE-D (average)	N	53.85 ± 0.85%	**70.16 ± 0.65%**
L2AE-D (best)	N	54.26 ± 0.87%	**70.76 ± 0.67%**

Table 3: Few shot classification results on miniImageNet with or without dropout averaged over 600 testing tasks. The ± shows 95% confidence over tasks. * denotes MAML uses 64 filters and tests on 15 queries per class.

Model	**5-way 1-shot Acc.**	
	without dropout	with dropout
MAML* [9]	47.71 ± 0.84%	**50.43 ± 0.87%**
ProtoNets [7]	49.42 ± 0.78%	**52.08 ± 0.81%**
RN [8]	50.44 ± 0.82%	**52.40 ± 0.85%**
L2AE	51.55 ± 0.82%	**53.85 ± 0.85% (L2AE-D)**

as well as ours. It can also be seen that, even without dropout, L2AE also outperforms those representative few-shot learning approaches. Since the proposed L2AE algorithm improves upon ProtoNets, Table 3 presents an ablation analysis. The only difference between ProtoNets and L2AE is that L2AE adds our proposed attention based aggregation module upon ProtoNets. Both of our results with and without dropout outperform ProtoNets by around 2%, which demonstrates that adding our channel-wise attention based aggregation module is effective for few-shot learning.

4.5. Visualisation of the working of L2AE-D

To further show how our approach works, we visualise the aggregated embeddings for the unseen few-shot classification tasks in the meta-testing set based on t-SNE [35]. t-SNE is a technique for dimensionality reduction that is particularly well suited for the visualisation of high-dimensional datasets [35]. Fig. 5(a) shows the visualisation of the aggregated embeddings for an unseen

Figure 5: t-SNE visualisation of the aggregated embeddings of unseen classes for a 5-way 1-shot classification task on Omniglot (a) and a 5-way 5-shot task on miniImagenet (b). The embeddings of training samples are shown as points. Aggregated embeddings are shown as triangles. The embeddings of regular examples are shown as crosses. The Means of training embeddings are shown as diamonds.

5-way 1-shot classification task on Omniglot. The embeddings aggregated from different classes tend to move away from their own cluster and be farther from the clusters of other classes.

Fig. 5(b) shows the visualisation of the aggregated embeddings for an unseen 5-way 5-shot classification task on miniImagenet. Compared to the embeddings of Omniglot in Fig. 5(a), we can see that the embeddings of miniImagenet are much messier and consist of more unrepresentative examples. This indicates the difficulty of few-shot learning on miniImagenet and the necessity to reduce the impact of outliers for a method. When there are unrepresentative examples in the training set, such as the outliers (the bottom red point) and the example located near the boundary of a cluster (the rightmost blue point), the mean of training embeddings [7] deviates from a good position that represents a class in the embedding space. However, our aggregated embeddings stick to a representative position in the embedding space and are much more stable regardless of unrepresentative examples, which can lead to a more robust decision boundary.

5. Conclusion

In this paper, we propose a novel meta-learning approach for aggregating useful convolutional features and suppressing noisy ones based on a channel-wise attention mechanism. We propose two different learning strategies for one-shot and few-shot tasks aiming to fully and effectively use the few training examples. Our model does not require any fine-tuning and can be trained in an end-to-end manner. In addition, we tackle the problem of meta-level overfitting by introducing a meta-level dropout technique. This technique significantly improve several well-known meta-learning approaches as well as ours. Furthermore, we achieve state-of-the-art performance over 20-way classification tasks on Omniglot and 5-way tasks on miniImageNet, which demonstrate the

effectiveness and competitiveness of our method.

Acknowledgement

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

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

[2] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

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

[4] B. Lake, R. Salakhutdinov, J. Gross, J. Tenenbaum, One shot learning of simple visual concepts, in: Proceedings of the Annual Meeting of the Cognitive Science Society, Vol. 33, 2011.

[5] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in: Proceedings of the 32th International Conference on Machine Learning, deep learning workshop, 2015.

[6] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: Advances in neural information processing systems, 2016, pp. 3630–3638.

[7] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087.

[8] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, 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, pp. 1199–1208.

[9] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1126–1135.

[10] N. Alex, A. Joshua, S. John, On first-order meta-learning algorithms, arXiv preprint arXiv:1803.02999.

[11] C. Finn, K. Xu, S. Levine, Probabilistic model-agnostic meta-learning, in: Advances in Neural Information Processing Systems, 2018, pp. 9537–9548.

[12] Y. Lee, S. Choi, Gradient-based meta-learning with learned layerwise metric and subspace, in: International Conference on Machine Learning, 2018, pp. 2933–2942.

[13] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning, in: International Conference on Learning Representations, 2017.

[14] T. Munkhdalai, H. Yu, Meta networks, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 2554–2563.

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

[16] S. Gidaris, N. Komodakis, Dynamic few-shot visual learning without forgetting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4367–4375.

[17] H. Edwards, A. Storkey, Towards a neural statistician, in: International Conference on Learning Representations, 2017.

[18] R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, Y. Song, Metagan: An adversarial approach to few-shot learning, in: Advances in Neural Information Processing Systems, 2018, pp. 2371–2380.

[19] H. Gao, Z. Shou, A. Zareian, H. Zhang, S.-F. Chang, Low-shot learning via covariance-preserving adversarial augmentation networks, in: Advances in Neural Information Processing Systems, 2018, pp. 983–993.

[20] B. Oreshkin, P. R. López, A. Lacoste, Tadam: Task dependent adaptive metric for improved few-shot learning, in: Advances in Neural Information Processing Systems, 2018, pp. 719–729.

[21] N. Mishra, M. Rohaninejad, X. Chen, P. Abbeel, A simple neural attentive meta-learner, in: International Conference on Learning Representations, 2018.

[22] V. Garcia, J. Bruna, Few-shot learning with graph neural networks, in: International Conference on Learning Representations, 2018.

[23] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, Y. Yang, Learning to propagate labels: Transductive propagation network for few-shot learning, in: International Conference on Learning Representations, 2019.

[24] Z. Ji, X. Liu, Y. Pang, X. Li, Sgap-net: Semantic-guided attentive prototypes network for few-shot human-object interaction recognition., in: Proceedings of the 34th AAAI Conference on Artificial Intelligence, 2020, pp. 11085–11092.

[25] Z. Ji, X. Chai, Y. Yu, Y. Pang, Z. Zhang, Improved prototypical networks for few-shot learning, Pattern Recognition Letters 140 (2020) 81–87.

[26] D. Wang, Y. Cheng, M. Yu, X. Guo, T. Zhang, A hybrid approach with optimization-based and metric-based meta-learner for few-shot learning, Neurocomputing 349 (2019) 202–211.

[27] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, H. Larochelle, Meta-dataset: A dataset of datasets for learning to learn from few examples, arXiv preprint arXiv:1903.03096.

[28] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: International Conference on Learning Representations, 2015.

[29] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, T.-S. Chua, Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5659–5667.

[30] S. Jetley, N. A. Lord, N. Lee, P. H. Torr, Learn to pay attention, in: International Conference on Learning Representations, 2018.

[31] S. Woo, J. Park, J.-Y. Lee, I. So Kweon, CBAM: Convolutional block attention module, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 3–19.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[33] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, C. Bregler, Efficient object localization using convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 648–656.

[34] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.

[35] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (Nov) (2008) 2579–2605.

[36] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, T. Lillicrap, Meta-learning with memory-augmented neural networks, in: International conference on machine learning, 2016, pp. 1842–1850.

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International Journal of Computer Vision 115 (3) (2015) 211–252.

[38] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools.

[39] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467.

[40] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, 2015.

[41] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in neural information processing systems, 2014, pp. 3320–3328.