Few-shot Learning with Meta Metric Learners

Yu Cheng[1] **Mo Yu**[2] **Xiaoxiao Guo**[2] **Bowen Zhou**[3]
[1] Microsoft AI & Research, [2]IBM Research AI, [3]JD AI Research
yu.cheng@microsoft.com {yum,xiaoxiao.guo}@ibm.com bowen.zhou@jd.com

Abstract

Few-shot Learning aims to learn classifiers for new classes with only a few training examples per class. Existing meta-learning or metric-learning based few-shot learning approaches are limited in handling diverse domains with various number of labels. The meta-learning approaches train a meta learner to predict weights of homogeneous-structured task-specific networks, requiring a uniform number of classes across tasks. The metric-learning approaches learn one task-invariant metric for all the tasks, and they fail if the tasks diverge. We propose to deal with these limitations with meta metric learning. Our meta metric learning approach consists of task-specific learners, that exploit metric learning to handle flexible labels, and a meta learner, that discovers good parameters and gradient decent to specify the metrics in task-specific learners. Thus the proposed model is able to handle unbalanced classes as well as to generate task-specific metrics. We test our approach in the 'k-shot N-way' few-shot learning setting used in previous work and new realistic few-shot setting with diverse multi-domain tasks and flexible label numbers. Experiments show that our approach attains superior performances in both settings.

1 Introduction

Supervised deep learning methods have been successfully applied to many applications such as computer vision, speech recognition and natural language processing. In practice, those methods usually require large amount of labeled data for model training, in order to make the learned model generalize well. However, collecting sufficient amount of training data for each task needs a lot of human-labeling work and the process is time-consuming.

Few-shot learning [1] was proposed to learn classifiers for new classes with only a few training examples per class. Two key ideas of few-shot learning are data aggregation and knowledge sharing. First, though each few-shot learning task may lack sufficient training data, the union of all the tasks will provide significant amount of labeled data for model training. Therefore the model training and prediction on a new coming few-shot could benefit from all the learned tasks. Secondly, the experiences of learning model parameters for a large number of tasks in the past will assist the learning process of the incoming new task. The few-shot learning idea has recently been combined with the deep learning techniques in two main lines of works: (1) learning metric/similarity from multiple few-shot learning tasks with deep networks [2, 3]; and (2) learning a meta-model on multiple few-shot learning tasks, which could be then used to predict model weights given a new few-shot learning task [4–6].

The aforementioned deep few-shot learning models usually are applied to the so called "k-shot, N-way" scenario, in which each few-shot learning task has the same N number of class labels and each label has k training instances. However, such "N-way" simplification is not realistic in real-world few-shot learning applications, because different tasks usually do not have the same number of labels. Existing meta-learning approaches build on the "N-way" simplification to let the meta-learner predict weights of homogeneous-structured task-specific networks. If we allow

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

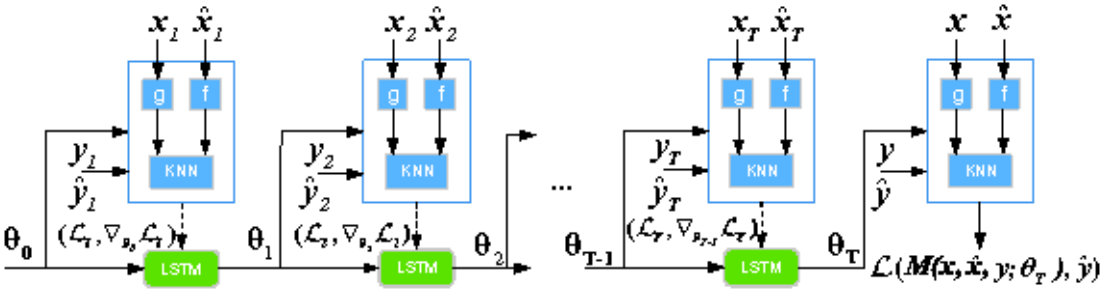

Figure 1: Computational graph of the forward pass of meta metric learner. Each (x_i, y_i) is the i^{th} batch sampled from D_{train} and (x, y) are all the samples of D_{train}. Analogously, each $(\hat{x}_i, \hat{y}_i)$ is the i^{th} batch sampled from D_{test} and $(\hat{x}, \hat{y})$ are all the samples of D_{test}. The dashed arrows indicate that the gradient is not back-propagated though that step when training the meta metric learner. We refer to the matching network learner as M with two encoders named g and f, and $M(x, \hat{x}, y; \theta_T)$ is the output of learner M.

different tasks work with different number of labels, the task-specific networks will be heterogeneous. Heterogeneous-structured task-specific networks complicate the weight prediction of the meta-learner. To the best of our knowledge, none of the existing meta-learning based few-shot learning approaches could resolve this issue. Although the metric-learning approaches could alleviate the variations on class labels, they suffer from the limitation of model expressiveness: these methods usually learn a task-invariant metric for all the few-shot learning tasks. However, because of the variety of tasks, the optimal metric will also vary across tasks. The learned task-invariant metric would fail if the tasks diverge.

Moreover, in real world applications, the few-shot learning tasks usually come from different domains or different resources. For example, for sentiment classification of product reviews, we could have products from different product departments on an e-commerce platform like Amazon. For an machine learning cloud service, there may be different clients submitting training data for their own business tasks. The tasks from different clients may deal with different problems, such as spam detection or sentiment classification. In such few-shot learning scenario, the two aforementioned issues of existing few-shot learning approaches will become more serious: the numbers and meanings of class labels may vary a lot among different tasks, so it will be hard for a meta-learner to learn how to predict weights for heterogeneous neural networks given the few-shot labeled data; and different tasks are not guaranteed to be even closely related to each other, so there will unlikely exits a uniform metric suitable for all the tasks from different domains or resources.

We propose to deal with the limitations of previous work with meta metric learning. The model consists of two main learning modules (Figure 1). The meta learner that operates across tasks uses an LSTM-based architecture to discover good parameters and gradient decent in task-specific base learners. The base learners exploit Matching Networks [3] and parameterize the task metrics using the weight prediction from the meta-learner. Thus the proposed model is now able to handle unbalanced classes in meta-train and meta-test sets as the usage of Matching Network as well as to generate task-specific metrics leveraging the weight prediction of the meta-learner given task instances.

We make the following contributions: (1) we improve the existing few-shot learning work to handle various class labels; (2) we further enable the model to learn task specific metrics via training a meta learner and (3) we are the first to investigate few-shot deep learning methods in the text domains. The contributions (1) and (2) make our approach suit better to the real few-shot learning scenarios where different tasks have various numbers of class labels and could come from different domains. We test our approach in the classic "k-shot N-way" few-shot learning setting following previous work and a new but more realistic few-shot setting with diverse multi-domain tasks and flexible label numbers. Experiments show that our approach attains superior performance on both settings.

2 Backgrounds on Few-Shot Learning

Few-Shot Learning (FSL) [1, 7] aims to learn classifiers for new classes with only a few training examples per class. Bayesian Program Induction [8] represents concepts as simple programs that best explain observed examples under a Bayesian criterion. Siamese neural networks rank similarity between inputs [2]. Matching Networks [3] map a small labeled support set and an unlabeled example to its label, obviating the need for fine-tuning to adapt to new class types. Siamese neural networks and Machine Networks essentially learn one metric for all the tasks, but to use only one metric is sub-optimal when the tasks are diverse. Recently, an LSTM-based meta-learner [5] learns the exact optimization algorithm used to train another learner neural network classifier for the few-shot learning problems. However, it requires a uniform number of classes across tasks. Our FSL approach could handle the challenges of diverse tasks with flexible labels.

2.1 Matching Network

Matching network [3] is a recent model developed for few-shot learning in computer vision applications. We adapt it to handle flexible class numbers in our Meta Metric Learner for both single-task and multi-task few-shot learning problems. Matching networks consist of two neural network (with shared parameters) as embedding functions and an augmented memory. The embedding functions, $f()$ and $g()$, map a review paragraph $x \in X$ to a N-length vector, i.e., $f, g : X \rightarrow \mathbb{R}^N$. The augmented memory stores a support set $S = \{(x_i, y_i)\}_{i=1}^k$, where x_i is the supporting instance and y_i is its corresponding label in a one-hot format. The matching networks explicitly define a classifier c_S conditioned on the supporting set. For any new data $\hat{x}$, the matching network predicts its label via a similarity function $\alpha(.,.)$ between the instance and the support set:

$$y = P(.|\hat{x}, S) = \sum_{i=1}^{k} \alpha(\hat{x}, x_i; \theta) y_i. \tag{1}$$

Specifically, we define the similarity function to be a softmax distribution given the inter-product between the test instance $\hat{x}$ and the supporting instance x_i, *i*.e., $\alpha(\hat{x}, x_i; \theta) = \exp(f(\hat{x}) \cdot g(x_i)) / \sum_{j=1}^{k} \exp(f(\hat{x}) \cdot g(x_j))$, where θ are the parameters of the embedding functions f and g. Thus, y is a distribution over the supporting set's labels $\{y_i\}_{i=1}^k$. We choose f to be convolutional neural networks following [9, 10].

Training Objective The original training objective of Matching network is specialized to match the test condition to the training condition for few-shot learning settings. We first sample a few-shot dataset D from all available datasets T, $D \sim T$. For each class in the sampled dataset D, we sample k random instances in that class to construct a support set S, and sample a batch of training instances B as training examples, i.e., $B, S \sim D$. The objective function to optimize the embedding parameters is to minimize the prediction error of the training samples given the supporting set as the follow:

$$\mathop{\mathbb{E}}_{D \sim T} \Big[\mathop{\mathbb{E}}_{B,S \sim D} \big[\sum_{(x,y) \in B} \log(P(y|x, S; \theta)) \big] \Big]. \tag{2}$$

The parameters of the embedding function, θ, are optimized via stochastic gradient descent methods.

2.2 Meta Learning

Meta-learning (also known as Learning to learn) has a long history [11, 12]. The key idea is framing the learning problem at two levels: the first is the quick acquisition of knowledge from each separate task presented and the second is accumulating these knowledge to learn the similarities and difference across all tasks. A recent approach to meta-learning [13] casts the hand-designed optimization algorithm as a learning problem, and trains an LSTM-based meta-learner to predict model parameters. An LSTM-based meta-learner [5] was applied to few-shot learning tasks.

The procedure of training a learner with parameters θ can be expressed as the problem of optimizing some loss function $\mathcal{L}(\theta)$ over some domain. The standard optimization algorithms are some variant of gradient descend:

$$\theta_{t+1} = \theta_t - \alpha_{t+1} \nabla \mathcal{L}(\theta_t) \tag{3}$$

where θ_t are the learner parameters after t update steps, α_{t+1} is the learning rate at time step $t+1$, $\nabla\mathcal{L}(\theta_t)$ is the gradient of the loss function with respect to parameters θ_t, and θ_{t+1} are the updated parameters of the learner. The LSTM-based meta-learner leverages that the update of learner parameters resembles the update of the cell state in an LSTM [14]:

$$c_{t+1} = f_{t+1} \odot c_t + i_{t+1} \odot \widetilde{c}_{t+1} \tag{4}$$

If we set the cell state of LSTM to be the parameters of the learner, i.e. $c_t = \theta_t$, the candidate cell state $\widetilde{c}_{t+1} = -\nabla\mathcal{L}(\theta_t)$, the output of forget gate $f_{t+1} = 1$, and the output of input gate $i_{t+1} = \alpha_{t+1}$, these two update procedures are completely the same. i_{t+1} and f_{t+1} determine how the meta-learner updates the parameters of learner. Thus, an LSTM can be trained as a meta-learner to learn an update rule for training a learner (such as a neural network).

3 Meta Metric Learner for Few-Shot Learning

In this section, we first describe the meta metric-learner in a single-task setting. After that, we show it is easy to generalize the model in a multi-task learning setting, which relates to retrieve auxiliary sets from other sources/tasks.

3.1 Meta Metric Learner

Let's consider the meta-learning in few-shot setting, on a data resource R with meta set $\mathcal{D}$, where $\mathcal{D}$ consists of three parts: $\mathscr{D}_{meta-train}$, $\mathscr{D}_{meta-validate}$ and $\mathscr{D}_{meta-test}$. Generally, in real applications, the number of classes in $\mathscr{D}_{meta-train}$ is different from that in $\mathscr{D}_{meta-test}$. Although the CNN base learner used in [5] is powerful to model image and text, it lacks an ability to handle unbalanced classes in train and test datasets in a straightforward way. Matching networks, on the other hand, as a trainable kNN and non-parametric algorithm by nature, can generalize easily to any new datasets, even if the number of classes are different. Hence, we apply the LSTM Meta-learner in [5] for few-shot learning tasks, but replace the CNN with Matching Networks as the base learner, so that it can tackle class-unbalanced few-shot learning problems. Matching networks is a kind of metric learning algorithm, and we train it using an LSTM-based meta-learner, so we call our method Meta Metric Learner.

Suppose we have a meta metric learner with parameters θ from the base learner M and Θ from meta learner R. We can use the meta-train set $\mathscr{D}_{meta-train}$ to train the LSTM-based meta-learner. When using the trained meta-learner and meta-test set $\mathscr{D}_{meta-test}$ to update the parameters of base learner, i.e. a matching networks, it takes as input the loss and its gradient w.r.t parameters of matching networks. Thus we need labels of both train data and test data to compute them, which means we need more than one labeled data from meta-test set. This is fine in few-shot learning problems (using a subset of $\mathscr{D}_{meta-test}$), but violates the assumption of one-shot learning. For class-unbalanced one-shot learning problems, we propose to exploit resource from other tasks to update the parameters of matching networks. The auxiliary set is named $\mathscr{D}_{aux}$. How to choose appropriate auxiliary dataset is described in the next section.

The meta metric learning training algorithm is shown in Algorithm 1 and the framework is also shown in Figure 1. It is notable that even for few shot learning ($k \neq 1$) with multi-tasks resource, we do not necessarily to use the subset $\mathscr{D}_{sub}$. The auxiliary set $\mathscr{D}_{aux}$ can be used to support the training of base learner. In such situation, the learning algorithm could save resource as well as having a good generalization power.

3.2 Auxiliary Task Retrieval via Task-Level Matching Networks

In this section, we discuss how to choose the auxiliary set $\mathscr{D}_{aux}$. The intuition is, in many real world applications, we could get few-shot learning data from multiple tasks (sources), such as different text classification tasks on dialogues. Such resources from multiple tasks could significantly increase the training data for our few-shot learning method as well as the previous few-shot learning methods as described in Section 2. However there is rarely a guarantee that those different tasks are related to each other. When the tasks are from unrelated resources, it will be difficult for the existing few-shot learning methods to learn a good metric or a good meta-learner. In this case when adding more significantly unrelated training resources, the performance may decrease. To overcome this difficulty,

Algorithm 1 Meta Metric Learner Training

Input:

Meta-train set $\mathscr{D}_{meta-train}$, Meta-test set $\mathscr{D}_{meta-test}$, auxiliary set $\mathscr{D}_{aux}$ or meta-test subset $\mathscr{D}_{sub}$, Matching networks learner M with parameters θ, Meta-Learner R with parameters Θ

****Meta-training****
$\Theta_0 \leftarrow$ random initialization
for $d = 1, n$ **do**
  $D^d_{train}, D^d_{test} \leftarrow$ random sampled dataset from $\mathscr{D}_{meta-train}$
  $\theta_0 \leftarrow c_0$
  for $t = 1, T$ **do**
    $x_t, y_t \leftarrow$ random batch sampled from D^d_{train}
    $\hat{x}_t, \hat{y}_t \leftarrow$ random batch sampled from D^d_{test}
    $\mathcal{L}_t \leftarrow \mathcal{L}(M(x_t, \hat{x}_t, y_t; \theta_{t-1}), \hat{y}_t)$
    $c_t \leftarrow R((\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t)); \Theta_{d-1})$
    $\theta_t \leftarrow c_t$
  end for
  $x, y \leftarrow$ all samples from D^d_{train}
  $\hat{x}, \hat{y} \leftarrow$ all samples from D^d_{test}
  $\mathcal{L}_{test} \leftarrow \mathcal{L}(M(x, \hat{x}, y; \theta_T), \hat{y})$
  Updating Θ_d using $\nabla_{\Theta_{d-1}} \mathcal{L}_{test}$
end for

****Updating Learner Parameters****
$D_{train}, D_{test} \leftarrow$ random sampled dataset from $\mathscr{D}_{aux}$ or $\mathscr{D}_{sub}$
$\theta_0 \leftarrow c_0$
for $t = 1, T$ **do**
  $x_t, y_t \leftarrow$ random batch sampled from D_{train}
  $\hat{x}_t, \hat{y}_t \leftarrow$ random batch sampled from D_{test}
  $\mathcal{L}_t \leftarrow \mathcal{L}(M(x_t, \hat{x}_t, y_t; \theta_{t-1}), \hat{y}_t)$
  $c_t \leftarrow R((\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t)); \Theta_n)$
  $\theta_t \leftarrow c_t$
end for

given a target task for few-shot learning, we propose the following approach to select related tasks following [15].

Specifically, consider a list of n tasks (such as a list of domains in sentiment analysis, or a list of alphabets in hand-writing recognition) $\mathcal{T} = \{T^1, T^2, ..., T^n\}$. From each task T^i we could sample few-shot learning data sets D^i_js. Because the D^i_js are usually too small to reflect any statistical relatedness among them, our approach deal with the problem at the task-level with the following steps: (1) For each data resource T^i we merge all the data sets D^i_j together and train a matching network M^i on it[1]. (2) For the target task T^{target}, on its combined task we apply each model M^i to get the accuracy $acc_{i \to k}$. Note that the accuracy scores are usually low but their relative magnitudes could reflect the relatedness between different tasks to T^{target}. (3) Finally we select the top s tasks T^i with the highest scores $acc_{i \to k}$ as the auxiliary set $\mathscr{D}_{aux}$ for the target task T^{target}.

4 Experimental Results

In this section, we execute experiment with k-shot learning in multi-task setting. The experiments are ran on three data sets: two text classification sets and one image classification set, comparing meta metric-learner model against several strong baselines. We first describe the datasets, experimental settings and baseline models.

[1] On each T^i, this step works the same as the standard matching network approach under the single-task setting.

Datasets We introduce three datasets, which can be used in multi-task setting: 1) **Sentence Classification Service (SCS)**: The data set is from an on-line service which trains and serves text classification models for different clients for their business purposes. The number of total clients is 12, and the number of classes per clients ranges from 10 to 28. 2) **Omniglot**: the data comes with a standard split of 30 training alphabets with 964 classes and 20 evaluation alphabets with 659 classes. Each of these was hand drawn by 20 different people. The large number of classes (characters) with 20 data per class; 3) **Amazon Reviews**: this corresponds a multi-domain sentiment classification data set from [16]. The dataset consists of Amazon product reviews for 25 product types. Each review's label is a rating based on 1-5 stars[2].

Baseline Models The first baseline we used is Matching Network. We implemented our own version of both the basic and the fully-conditional embedding (FCE) versions. The Second baseline is Meta-Learner LSTM. We implemented our own version according to [5], which takes similar structures, dropping out and batch normalization.

CNN architectures We exploit two different CNN architecture used in all the methods: 1) for text, we used a simple yet powerful CNN [9] as the embedding function, which consists of a convolution layer and a max-pooling operation over the entire sentence for each feature map. The model uses multiple filters with varying window sizes $h(h = 3, 4, 5)$. Word embedding are initialized with 100-dimensional Glove embeddings trained on 6B corpus from [17]; 2) for image, the 2D CNN architecture in [3] is used, which consists of a stack of modules, a 3×3 convolution with 64 filters followed by batch normalization, a Relu non-linearity and 2×2 max-pooling.

Hyper-paramters There are several hyper-parameters required for meta-learner, matching network and CNNs. All of them are tuned in the validation set.

4.1 Multi-tasks/domains Setting

Sentence Classification Service (SCS) On SCS, the data of each clients can be viewed as a task. For each set, we randomly sample 50% data into a meta-training set, 20% into a meta-validation set, and assign the rest data into the meta-test set. For each sample, we did stop-word removal/tokenization with the CMU NLP tool [18] for the preprocessing. Since there are large number of clients on the service, random sampling auxiliary tasks has low chance to find related tasks. As a result, we use the method in Section 3.2 to retrieve top 10 related tasks from the 175-task pool. Then we train all the few-shot learning methods on the sampled tasks together with the meta-training set. The baseline, matching network is trained iteratively with one sampled task data each time. Obvious, meta-learner LSTM can not take the additional data and is only trained with the meta-training set.

After that, all the approaches are evaluated on the meta-test set with 5 samples per class. The validation set is used to adjust the hyper-parameter of the model. Results comparing the baselines to our model on SCS are shown in Table 1, for both 1-shot and 5-shot setting. Meta Metric-learners achieve the best accuracies over all the methods. Even with the help of additional sources, the performance of matching network is not better than meta-learner LSTM.

Table 1: Comparisons between the models and their baselines on the real-world sentence classification tasks from multiple resources (SCS).

Model	Matching Fn	Additional Data	Average Acc	
			1-shot	5-shot
Matching Network	Basic	Y	53.59%	68.73%
Matching Network	FCE	Y	54.24%	70.28%
Meta-learner LSTM	-	N	56.98%	72.54%
Meta Metric-learner	Basic	Y	57.62%	73.83%
Meta Metric-learner	FCE	Y	58.13%	74.54%

[2]Data from http://www.cs.jhu.edu/ mdredze/datasets/sentiment/. The 3-star samples were removed due to their ambiguous nature [16].

Omniglot For Omniglot, each task corresponds to an alphabet, and the total is number of 50. We randomly choose 20 tasks in this experiment. Considering multi-task setting for Omniglot has a clear motivation, as cross-alphabet knowledge sharing is likely to be useful. We use a similar strategy to find related top-s (s is different according to different tasks) tasks and train the metric learner. The following setting is used: splitting each task with 5:2:3 as meta-training, meta-validation and meta-testing. The validation set is used to tune the hyper-parameters. We use 10 examples per class for evaluation in each test set. Results comparing the baselines to our model on Omniglot are shown in Table 2. For 1-shot and 5-shot, our model can achieve better classification accuracies than others. With the help of more resources, matching network can beat meta-learner LSTM around 2-3%.

Table 2: Comparisons between the models and their baselines on Omniglot.

Model	Matching Fn	Additional Data	Average Acc	
			1-shot	5-shot
Matching Network	Basic	Y	94.62%	98.37%
Matching Network	FCE	Y	95.84%	98.65%
Meta-learner LSTM	-	N	93.54%	97.22%
Meta Metric-learner	Basic	Y	95.32%	98.56%
Meta Metric-learner	FCE	Y	95.79%	98.83%

Amazon Reviews We treat each product category as a task and the goal is classify each sample into one of four categories. We select one domain as the target few-shot learning task. For the rest tasks, we select 5 tasks from them to train the metric-learner and 1 task used as the meta-validation set to search over hyper-parameters. At each iteration, data from one task is sampled to train the models. Under such condition, Meta-learner LSTM can take the benefit of using additional samples from other tasks. We use 5 sentence per class for evaluation in each test set. The results are shown in Table 3. The meta-learner attains result that are better than the baselines discussed. For 5-shot, we are able to improved matching network more than 5%, whereas for 1-shot, the results are still competitive and outperform the second-best around 2%. It is a little surprising that LSTM meta-learners can not work well on this dataset, with/without additional data, showing that using sample of different sources to jointly train meta-learner is not help.

Table 3: Comparisons between the models and their baselines on Amazon Reviews.

Model	Matching Fn	Additional Data	Average Acc	
			1-shot	5-shot
Matching Network	Basic	Y	44.25%	51.92%
Matching Network	FCE	Y	47.18%	54.64%
Meta-learner LSTM	-	N	42.69%	51.36%
Meta-learner LSTM	-	Y	43.47%	52.05%
Meta Metric-learner	Basic	Y	48.25%	58.44%
Meta Metric-learner	FCE	Y	49.38%	60.82%

4.2 Single Task Setting

To demonstrate the effectiveness of our meta metric learner, we also execute experiments for single task/resource, on Sentence Classification Service and Omniglot datasets, in which no auxiliary set $\mathscr{D}_{aux}$ is available from other tasks. Thus we need to perform $k*2$-shot learning ($k=1,2$), i.e., for each class, we split its samples into two parts equally, to update the model and base learner jointly. Two separate splits are used: 1) 50%, 20%, and 30% classes for training, validation and testing, 2) 30%, 20%, and 50% classes for training, validation and testing. For 1), k$-shot$,N-way classification is performed. 2) is a challenging setting since the number of classes in meta-train is smaller than meta-test. We use all classes in $\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$, which is different from k-shot,N-way setting. For both splits, the validation set is used to adjust the hyper-parameters. To have a fair comparison, all the baselines trained with $k*2$ samples per class according to their own recipes.

For SCS, all 10 tasks are used in the evaluation and the results are shown in Table 4. All the results are measured after 10 runs. It is notable that for 3 vs. 5 split, meta-learner LSTM is not able to be employed. In both 2 and 4 shot, our model outperforms the other methods. The performance of

meta-learner LSTM is slightly better than Matching Network. The classification accuracy of meta metric-learner with FCE is higher than the basic version, which shows that FCE can improve the basic one on SCS. On the other hand, it is obvious to see that the 3 vs. 5 split is a more challenging task. Comparing the results in both cases, the performance of 3 vs. 5 is around 10% lower than 5 vs. 3 cases.

Table 4: Average classification accuracies on SCS of different approaches in single task setting.

Model	**Matching Fn**	**5 vs. 3 split**		**3 vs. 5 split**	
		2-shot	4-shot	2-shot	4-shot
Matching Network	Basic	59.42%	68.65%	48.15%	57.28%
Matching Network	FCE	59.57%	68.91%	48.74%	57.31%
Meta-learner LSTM	Basic	60.15%	69.06%	-	-
Meta Metric-learner	Basic	60.81%	69.14%	50.23%	58.44%
Meta Metric-learner	FCE	61.27%	69.58%	50.56%	59.02%

On Omniglot set, we randomly choose 20 tasks from the whole tasks in the evaluation. Same setting and data split are used as in SCS. The results are reported after 15 runs and described in Table 5. Similar trend is observed: the performance of meta metric-learners are better than others. Meta-learner LSTM and matching network achieve almost the same performance for 5 vs. 3 split. Different from SCS, considering the accuracies of two Meta Metric-learner, FCE function seems does not help a lot here.

Table 5: Average classification accuracies on Omniglot of different approaches in single task setting.

Model	**Matching Fn**	**5 vs. 3 split**		**3 vs. 5 split**	
		2-shot	4-shot	2-shot	4-shot
Matching Network	Basic	96.02%	96.83%	93.94%	94.88%
Matching Network	FCE	96.50%	97.39%	94.14%	95.26%
Meta-learner LSTM	-	96.54%	97.45%	-	-
Meta Metric-learner	Basic	97.24%	98.33%	95.77%	96.32%
Meta Metric-learner	FCE	97.38%	98.47%	95.69%	96.24%

5 Conclusion

In this paper, we proposed a meta metric learner for few-shot learning, which is a combination of an LSTM meta-learner and a base metric classifier. The proposed method takes several advantages such as is able to handle unbalanced classes as well as to generate task-specific metrics. Moreover, as shown in the results, using the meta-learner to guide gradient optimization in matching network seems to be a promising direction. We evaluate our model on several datasets, in both single task and multi-tasks settings. The experiments demonstrate that our approach outperforms is very competitive to the state-of-the-art approaches for few-shot learning.

There are several directions for future work. First, we will focus on selecting the data from related domains/resources to support the training of meta metric learners. Secondly, it would be interesting to propose an end-to-end framework of the meta-learner to leverage the data from different domains/sources/tasks for the training, instead of the current two-stages procedure. Finally, we would like to move forward to apply the current framework in other applications, such as language modeling [19], machine translation [20] and vision applications [21].

References

[1] Fei-Fei Li, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[2] Gregory Koch. Siamese neural networks for one-shot image recognition. 2015.

[3] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[4] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.

[5] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, volume 1, page 6, 2017.

[6] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *arXiv preprint arXiv:1703.00837*, 2017.

[7] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. 1:464–471, 2000.

[8] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[9] Yoon Kim. Convolutional neural networks for sentence classification. pages 1746–1751, October 2014.

[10] Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using one-hot lstm for region embeddings. *stat*, 1050:7, 2016.

[11] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[12] Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.

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

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

[15] Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215, 2018.

[16] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.

[17] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. 14:1532–1543, 2014.

[18] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proc. of ACL*, 2011.

[19] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083, 2016.

[20] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. Dual learning for machine translation. pages 820–828, 2016.

[21] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogério Schmidt Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1131–1140, 2017.