MEDA: Meta-Learning with Data Augmentation for Few-Shot Text Classification

Pengfei Sun, **Yawen Ouyang**, **Wenming Zhang** and **Xin-yu Dai***
National Key Laboratory for Novel Software Technology, Nanjing University
{spf, ouyangyw, zhangwm}@smail.nju.edu.cn, daixinyu@nju.edu.cn

Abstract

Meta-learning has recently emerged as a promising technique to address the challenge of few-shot learning. However, standard meta-learning methods mainly focus on visual tasks, which makes it hard for them to deal with diverse text data directly. In this paper, we introduce a novel framework for few-shot text classification, which is named as **ME**ta-learning with **D**ata **A**ugmentation (MEDA). MEDA is composed of two modules, a ball generator and a meta-learner, which are trained jointly. The ball generator is to increase the number of shots per class by generating more samples, so that meta-learner can be trained with both original and augmented samples. It is worth noting that ball generator is agnostic to the choice of the meta-learning methods. Experiment results show that on both datasets, MEDA outperforms existing state-of-the-art methods and significantly improves the performance of meta-learning on few-shot text classification.

1 Introduction

Deep learning has achieved great success in many areas, including computer vision, speech recognition, and natural language processing. However, supervised deep learning models require a large amount of labeled data to achieve competitive performance. This requirement limits the generalizability of models to novel classes with few examples. In contrast, humans can learn a novel concept with very few examples. Inspired by this observation, researchers have begun to study few-shot learning problem.

Recent efforts to address the few-shot learning problem have leveraged a meta-learning paradigm [Snell *et al.*, 2017; Finn *et al.*, 2017]. Meta-learning methods train a meta-learner, which extracts knowledge from many related tasks during meta-training and leverages that knowledge to learn new tasks during meta-testing quickly. Meta-learners are trained by sampling a small training set (support set) and test set (query set) from a large universe of labeled examples, feeding the support set to the learner to obtain a classifier, and then computing the loss of the classifier on the query set. The goal of meta-learner is to classify novel classes with few examples.

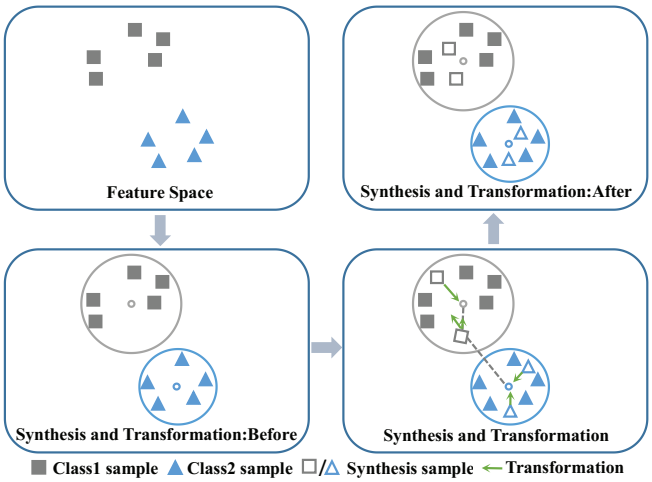

Figure 1: An illustration of ball generator. The first block shows the two classes of samples in the feature space. The second and third blocks illustrate two key parts. Firstly, we calculate the smallest enclosing ball [Welzl, 1991] of each class according to the existing samples. After that, we synthesize new samples in the smallest enclosing ball. Secondly, a transformation module is proposed to make synthetic samples close to their own ball center and away from other ball centers. The last block shows the results of synthesis and transformation.

Although these methods develop fast, most of these works concentrate on image classification, and less attention has been paid to text classification. Different from images, text in the same class has diverse expressions, which makes the performance of meta-learner more easily limited by the number of samples per class (i.e., the number of shots). Thus, the current meta-learning methods might be difficult to directly extend to few-shot text classification.

To tackle this problem, we introduce data augmentation for meta-learning, which brings in the benefits that we can produce augmented samples to increase the shot numbers for novel classes and diversify training samples. However, this poses a new challenge that producing high-confidence samples is non-trivial in the few-shot settings.

*Contact Author

In order to address this issue, we propose a novel data augmentation method named ball generator, and Figure 1 shows a simple illustration for it. Firstly, we calculate the smallest enclosing ball [Welzl, 1991] of the support set and synthesize samples in this ball. We argue that samples in this ball have higher confidence because all support set samples are contained in this ball, and their farthest distance from the ball center is minimized. Secondly, to avoid the effect of synthetic sample bias, we introduce the transformation module to make synthetic samples close to their own ball center and away from other ball centers. The whole process is called "ball generator". Besides, we also propose a novel framework **ME**ta-Learning with **D**ata **A**ugmentation (MEDA) to jointly train the ball generator and the meta-learner, and make them evolve synergistically. The experimental results show that compared with vanilla meta-learning, increasing the number of shots for novel classes can effectively improve the generalization ability of meta-learner on the few examples.

In summary, the main contributions of our work are as follows:

- We propose a novel data augmentation method named ball generator to increase the number of shots for novel classes. Notably, ball generator is agnostic to the choice of the meta-learning methods. We have successfully applied it to prototypical networks and relation networks, and verified the effectiveness of it.
- We propose a novel framework **ME**ta-Learning with **D**ata **A**ugmentation (MEDA) for few-shot text classification, which jointly optimizes the ball generator and the meta-learner, such that the ball generator can learn to produce augmented samples that best fit the meta-learner.
- We conduct a lot of experiments to prove the benefits of our proposed method. Experiments on two benchmark datasets demonstrate the effectiveness and superiority of the model.

2 Related Work

2.1 Meta Learning

Meta-learning intends to design models that can learn new skills or adapt to new environments rapidly with a few training examples. Metric-based meta-learning is one of the classical techniques used, where learns a distance function between data points so that it classifies test instances by comparing them to the K labeled examples. Typical work includes siamese networks [Koch *et al.*, 2015], matching networks [Vinyals *et al.*, 2016], prototypical networks [Snell *et al.*, 2017], relation networks [Sung *et al.*, 2018]. Although these works acquire fine results, most achievements are concentrated on computer vision field, the research and applications in text classification field are limited. Zhang et al. [2019] propose a meta-learning model which are the first to bridge the pretraining strategy with meta-learning methods for few-shot text classification. Unlike previous studies, our method equips meta-learning with data augmentation, which increases the number of shots by augmenting the minimally available samples and generates more diverse samples to extend to new tasks more effectively.

2.2 Data Augmentation

Data augmentation methods are explored in the field of text processing for improving the performance of models. At present, the mature methods of data augmentation can be divided into two categories: one is original text-oriented augmentation research; another is feature space data augmentation research. The former is mainly augmented by synonym replacement or deletion of words in the raw text. Such as Easy Data Augmentation (EDA) [Wei and Zou, 2019] augments the raw text based on four operations: synonym replacement, random insertion, random swap, and random deletion. Although these operations can augment text data, they may alter the text purport, and the noise brought will affect the performance of the model. More generally, a recent class of methods tries to explore augmentation in the feature space. They mainly process feature space of raw text, and inject noise into the text representation through different methods to improve the generalization ability of the model [Malandrakis *et al.*, 2019; Fedus *et al.*, 2018]. Although these methods have achieved good results, in the case of few examples, there are some problems such as the confidence of the augmented samples is not high, and the model training is difficult.

For few-shot text classification, Kumar et al. [2019] study six feature space data augmentation methods to improve classification performance in few-shot setting in combination with both supervised and unsupervised representation learning methods. Our approach also follows this line of work, but we redesign the model to make it simpler and easier to train. Meanwhile, our approach generates additional training samples by combining a meta-learner module with a generator module. These aforementioned modules are trained in an end-to-end manner to facilitate training few-shot text classifiers.

3 Preliminaries

3.1 Problem Formulation

In this paper, we establish few-shot text classification in a meta-learning framework [Vinyals *et al.*, 2016]. Specifically, we are given labeled examples from a set of classes C_{train}. Our goal is to learn a classification algorithm on C_{train}, so that we can make predictions over novel classes, which have only a few examples. These novel classes belong to a set of classes C_{test} and disjoint from C_{train}.

To emulate the few-shot scenario, meta-learning algorithms learn from a group of N-way, K-shot tasks sampled from C_{train} and are then evaluated in a similar way on C_{test}. During meta-training, we sample N classes (N-way) from C_{train} to create a single training episode. For each of these N classes, we sample K examples (K-shot) as the support set S and T examples as the query set Q, i.e., $S = \{(\boldsymbol{X_i}, \boldsymbol{y_i})\}_{i=1}^{N\times K}$ and $Q = \{(\boldsymbol{X_j}, \boldsymbol{y_j})\}_{j=1}^{N\times T}$. We update our model based on loss over the query set Q. During meta-testing, we apply the same episode-based mechanism to test whether our model can indeed adapt quickly to novel classes C_{test}.

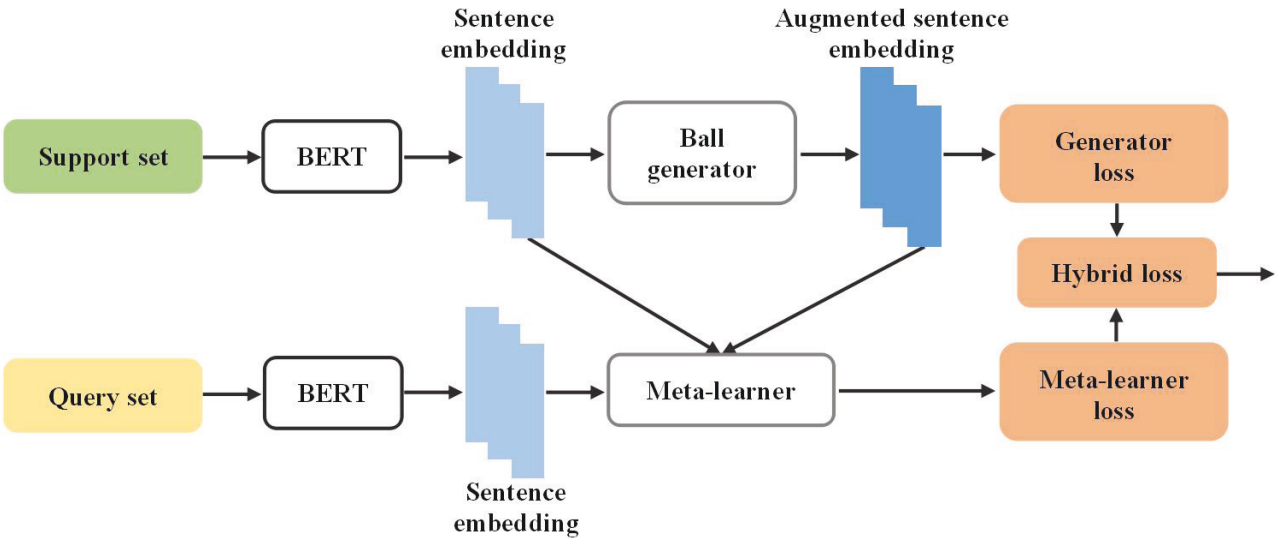

Figure 2: The overall framework of MEDA: Meta-learning with Data Augmentation.

3.2 Meta-learner

In meta-learning , meta-learners are trained according to episodes. In each episode, meta-learners calculate the probability distribution of each query instance $\hat{\boldsymbol{X}} \in Q$ under the support set S, denote as $P(\hat{\boldsymbol{y}}|\hat{\boldsymbol{X}}, S; \phi)$. The difference between different meta-learners is how to implement $P(\hat{\boldsymbol{y}}|\hat{\boldsymbol{X}}, S; \phi)$. Specifically, we choose prototypical networks and relation networks as meta-learners for MEDA.

Prototypical Networks. The prototypical networks [Snell *et al.*, 2017] use support set S to calculate a prototype for each class, and then classify the query instance according to the distance between the query instance and the prototypes. More specifically, given a query instance $\hat{\boldsymbol{X}}$, it belongs to the probability distribution of class c:

$$P(\hat{y} = c \mid \hat{\boldsymbol{X}}, S; \phi) = \frac{\exp(-d(f_{\phi}(\hat{\boldsymbol{X}}), \boldsymbol{p_c}))}{\sum_t \exp(-d(f_{\phi}(\hat{\boldsymbol{X}}), \boldsymbol{p_t}))}, \quad (1)$$

where $d(\cdot, \cdot)$ represents the distance calculation function, and the Euclidean distance is used here. $\boldsymbol{p_c}$ represents the prototype, and its calculation formula is as follows:

$$\boldsymbol{p_c} = \frac{1}{|S_c|} \sum_{(\boldsymbol{X_i}, y_i) \in S_c} f_{\phi}(\boldsymbol{X_i}). \quad (2)$$

where $S_c \subset S$ represents the subset corresponding to class c in the support set S.

Relation Networks. Compared with the prototypical networks, relation networks [Sung *et al.*, 2018] measure the distance between an unlabeled query instance and a few labeled instances by constructing neural networks. It consists of two parts: a feature extractor $f(\cdot)$ and a relation module $g(\cdot)$. First, the feature extractor generates representations of the query and training instances. Then the probability of the query instance $\hat{\boldsymbol{X}}$ belonging to class c is calculated by the relation module. The details are as follows:

$$P(\hat{y} = c \mid \hat{\boldsymbol{X}}, S; \phi, \varphi) = g_{\varphi}(\mathcal{C}(\boldsymbol{p_c}, f_{\phi}(\hat{\boldsymbol{X}}))), \quad (3)$$

where $\mathcal{C}(\cdot, \cdot)$ is concatenating operation. $\boldsymbol{p_c}$ represents the feature vector corresponding to class c, its calculation is as follows:

$$\boldsymbol{p_c} = \sum_{(\boldsymbol{X_i}, y_i) \in S_c} f_{\phi}(\boldsymbol{X_i}). \quad (4)$$

4 MEDA

This section introduces the general architecture and principal modules of the proposed model. Figure 2 shows the overall architecture of MEDA. MEDA has two modules: one is ball generator module; the other is meta-learner module. The generator module generates the samples based on the feature vector of the support S samples (Section 4.1). We take the set of augmented samples S' and add it to the support set S to produce an extended support set $\tilde{S} = S \cup S'$. The meta-learner module is to make predictions over the query set Q, after learning from the extended support set $\tilde{S}$ (Section 3.2). These aforementioned modules are trained in an end-to-end manner (Section 4.2 and Section 4.3).

4.1 Ball Generator

Our method is a kind of feature space data augmentation. The key idea is to synthesize samples in feature space, and then alleviate the synthetic sample bias. To achieve so, we design a generator network, which is composed of the synthesis module and transformation module. As shown in Figure 2, we first learn data representation using a powerful pre-training language model BERT [Devlin *et al.*, 2019]. Then, all samples are projected into the D-dimensional feature space through BERT.

Synthesis module. In feature space, we restrict the sampling space into smallest enclosing ball of the support set. Specifically, for support set S_i, we use the randomization method proposed by Welzl [1991] to calculate the smallest enclosing ball $B(S_i) = \{\boldsymbol{C_i}, R_i\}$ corresponding to each class y_i, where $\boldsymbol{C_i} \in R^D$ is the center of the ball and R_i is the

radius. After obtaining $B(S_i)$, we synthesize the sample $\boldsymbol{X'} \in R^D$ using the formula below:

$$\boldsymbol{X'} = \boldsymbol{C_i} + u^{1/D} R_i \frac{\boldsymbol{Z}}{\|\boldsymbol{Z}\|_2}, \tag{5}$$

where u is uniformly sampled in the range of [0-1]. $\boldsymbol{Z}$ is from standard normal distribution, i.e. $\boldsymbol{Z} \sim \mathcal{N}(0, \mathbf{I})$.

Transformation module. To avoid the effect of synthetic sample bias, we apply a transformation in feature space to make synthetic samples close to their own ball center and away from other ball centers. We therefore write the transformation as a function $\boldsymbol{X''} = G(\boldsymbol{X'}; \boldsymbol{\theta})$ that takes a synthetic point $\boldsymbol{X'}$ as input, and produces an augmented sample $\boldsymbol{X''}$ as output. $\boldsymbol{\theta}$ is the parameters of the transformation. We use a MLP with three fully connected layers for $G(\cdot; \boldsymbol{\theta})$.

We call the whole process the ball generator. For each class, we use the generator to generate M augmented samples and get the set of augmented samples S'. These augmented samples are added to the support set S to produce an extended support set $\tilde{S} = S \cup S'$, where $S = \{(\boldsymbol{X}_{p,i}, \boldsymbol{y}_{p,i})\}_{p=1,i=1}^{N,K}$ and $S' = \{(\boldsymbol{X''}_{p,j}, \boldsymbol{y}_{p,j})\}_{p=1,j=1}^{N,M}$. Therefore, it is easy to integrate data augmentation into meta-learning by rewriting meta-learner from $P(\hat{\boldsymbol{y}}|\hat{\boldsymbol{X}}, S; \boldsymbol{\phi})$ to $P(\hat{\boldsymbol{y}}|\hat{\boldsymbol{X}}, \tilde{S}; \boldsymbol{\phi})$.

4.2 Hybrid Loss

To jointly optimize the ball generator and the meta-learner, we introduce the hybrid loss. With the help of hybrid loss, the ball generator can produce more suitable samples for the meta-learner. Therefore, the hybrid loss is defined as follows:

$$\mathcal{L} = \lambda \mathcal{L}_{GEN} + \mathcal{L}_{ML}, \tag{6}$$

where λ is a trade-off parameter to control the relative importance of each term. $\mathcal{L}_{GEN}$ is the generator loss, and $\mathcal{L}_{ML}$ is the meta-learner loss.

Generator Loss. To generate higher confidence augmented samples, we design the generator loss to make augmented samples close to their own ball center and away from other ball centers. The specific definition is as follows:

$$\mathcal{L}_{GEN} = \sum_{\substack{\boldsymbol{x''} \in y_i \\ \boldsymbol{x''} \notin y_j}} \max\{0, d(\boldsymbol{X''}, \boldsymbol{C_i}) + r - d(\boldsymbol{X''}, \boldsymbol{C_j})\}, \tag{7}$$

where $\boldsymbol{X''}$ is the augmented sample by the generator, $\boldsymbol{C_i}$ and $\boldsymbol{C_j}$ are the ball center corresponding to the i-th and j-th classes respectively, $d(\cdot, \cdot)$ is the Euclidean distance and r is a margin.

Meta-learner Loss. To maximize learning, we design different loss functions for different meta-learners. Specifically, we use cross entropy loss for prototypical networks and mean square error loss for relation networks. Empirically, we find that using the generator loss speeds up the convergence and improves the classification performance than using the meta-learner loss alone.

Algorithm 1 Training Strategy

Input: The number of ways, shots, queries, augmented samples N, K, T, M, labeled dataset C_{train}
Output: $\boldsymbol{\theta}, \boldsymbol{\phi}$

for each episode iteration **do**
  Randomly sample N classes from C_{train}
  for $i = 1$ to N **do**
    Randomly sample K examples for the class i-th as support set $S_i = \{(X_s, y_s)\}_{s=1}^{K}$
    Randomly sample T examples for the class i-th as query set $Q_i = \{(X_q, y_q)\}_{q=1}^{T}$
    $B(S_i) = \{\boldsymbol{C_i}, R_i\}$ # Calculate the smallest enclosing ball using S_i
    $S' = \emptyset$ # Initialize the extended support set
    for $j = 1$ to M **do**
      Compute the synthetic sample $\boldsymbol{X'}$ by Eq. 5
      Use $G(\boldsymbol{X'}; \boldsymbol{\theta})$ to generate the augmented sample $\boldsymbol{X''}$
      $S' = S' \cup \{(\boldsymbol{X''}, y_i)\}$
    end for
    $\tilde{S} = S \cup S'$
  end for
  Use the augmented support set S' to calculate the generator loss by Eq. 7
  Feed the extended support set $\tilde{S}$ to the meta-learner, and calculate the meta-learner loss in the query set Q
  Update the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ by minimizing the hybrid loss
end for

4.3 Training Strategy

Algorithm 1 summarizes the end-to-end training strategy. We construct the episode to calculate the gradient and update the parameters in each training iteration. We first sample the N-way-K-shot training episode to produce S and Q from C_{train}. We use the generator to take the set of augmented samples S' and add it to the support set S to produce an extended support set $\tilde{S} = S \cup S'$. The augmented support set S' is used to calculate the generator loss by Eq. 7. We then feed the extended support set $\tilde{S}$ to the meta-learner, and calculate the meta-learner loss in the query set Q. We minimize the hybrid loss to update the parameters. In this way, we can optimize the MEDA in an end-to-end manner.

5 Experiments

5.1 Datasets

To prove the effectiveness of our proposed model, we evaluate the MEDA on two publicly datasets for the few-shot scenario: SNIPS[1] [Coucke *et al.*, 2018] and ARSC[2] [Yu *et al.*, 2018]. SNIPS is a personal voice assistant dataset which is collected in a crowd sourced fashion. ARSC is a sentiment classification dataset which is constructed by Amazon product reviews. Table 1 summarizes the statistics of the above mentioned datasets.

[1]https://github.com/snipsco/nlu-benchmark/
[2]https://github.com/Gorov/DiverseFewShot_Amazon

Datasets	SNIPS	ARSC
Vocab Size	11641	206913
Average Sentence Length	9.05	98.62
Train Samples	9888	119745
Test Samples	3914	18627

Table 1: For few-shot text classification, statistics of SNIPS and ARSC datasets.

SNIPS. As SNIPS is not a benchmark for few-shot learning, we first construct few-shot splits to simulate the few-shot scenario. We divide the original 7 intents into 5 intents as C_{train} and 2 intents (AddToPlaylist, RateBook) as C_{test}. The C_{train} and C_{test} are used as training set and test set respectively. Thus, we evaluate the performance on C_{test}, i.e., 2-way-K-shot settings, where K=3,5,10.

ARSC. For ARSC dataset, we partition datasets following [Yu *et al.*, 2018]. The dataset comprises English reviews for 23 types of products on Amazon. For each product domain, there are three different binary classification tasks. These buckets then form $23 \times 3 = 69$ tasks in total. Following [Yu *et al.*, 2018], we also select 12 (4×3) tasks from four domains (Books, DVD, Electronics, Kitchen) as the test set, with only five examples as support set for each label in the test set. We create 5-shot learning models on this dataset.

5.2 Baselines

We compare against several baselines and competitors as follows. (1) From the perspective of data augmentation, our method is a kind of feature space data augmentation. Therefore, in order to illustrate the effectiveness of the proposed method, we compare it with other feature space data augmentation methods. There are mainly Extrapolation (EXTRA) [DeVries and Taylor, 2017] and Random Perturbation (RP) [Kumar *et al.*, 2019]. In order to ensure fairness, we also increase the number of shots by the above methods and then classify them by prototypical networks. (2) For meta-learner, our overall framework is generic, and we can employ different meta-learning algorithms as meta-learner. Therefore, we compare the two methods described in Section 3.2, including prototypical networks (PN) and relation networks (RN). (3) From the perspective of few-shot text classification, in order to illustrate the effectiveness of MEDA, we also compare it with other state-of-the-art models, including ROBUSTTC-FSL [Yu *et al.*, 2018] and Induction Networks (IN) [Geng *et al.*, 2019].

5.3 Implement Details

Our implementation is based on Pytorch[3]. We experiment with pre-trained BERT [Devlin *et al.*, 2019] using Sentence-Transformers codebase [Reimers and Gurevych, 2019]. Adam [Kingma and Ba, 2015] is used to train the MEDA in an end-to-end manner. The initial learning rate is 1e-3. In the loss function, we set $\lambda = 1$ and $r = 1$. To avoid overfitting, we use dropout with 0.2 dropout rate. We generate 10 samples using different augmentation methods for the given class. We evaluate the model performance after each epoch and stop the training when the error on the validation set starts increasing. All hyper-parameters of the MEDA are cross-validated on the validation set using a coarse grid search. For EXTRA, we use $\lambda = 0.5$. For RP, we sample noise from a normal distribution with $\mu = 0$, $\sigma = 1$.

[3]https://pytorch.org/

Method	Mean Acc		
	K=3	K=5	K=10
Augmentation			
EXTRA-PN	86.14	89.53	92.44
RP-PN	86.39	88.79	92.62
Meta-learners			
PN	84.83	86.96	91.01
RN	84.07	86.23	91.16
Ours			
MEDA-PN	87.52	**91.13**	**94.34**
MEDA-RN	**88.13**	90.62	93.61

Table 2: Comparison of mean accuracy (%) on SNIPS. K indicates the number of training examples per class (K-shot).

Method	Mean Acc
Augmentation	
EXTRA-PN	83.09
RP-PN	82.15
Meta-learners	
PN*	68.17
RN*	83.07
Competitors	
ROBUSTTC-FSL*	83.12
IN*	85.63
Ours	
MEDA-PN	85.68
MEDA-RN	**87.21**

Table 3: Comparison of mean accuracy (%) on ARSC. $*$ indicates a result borrowed from [Geng *et al.*, 2019].

6 Results

Overall Performance. Experiment results on all two datasets are presented in Table 2 and Table 3. From the results, we can clearly see that our proposed MEDA outperforms all other models on standard few-shot settings. Particularly, on SNIPS, the classification accuracy of MEDA is superior to the prototypical networks by more than 2.5% when K=3;5, showing the effectiveness of MEDA for few-shot text classification. With more data (e.g., K=10), although all baselines have performed very well, our MEDA is still improved by about 2.5% compared with others. For ARSC, the improvements of MEDA on this dataset are consistent with SNIPS. This further validates the general effectiveness of our proposed MEDA in addressing few-shot text classification.

MEDA vs. meta-learner. The comparison results between MEDA versions and vanilla versions of prototypical networks and relation networks are summarized in Table 2 and Table 3. For ARSC, we find that MEDA versions improve the classification accuracy significantly, by up to 17% and 4% compared with prototypical networks and relation networks, re-

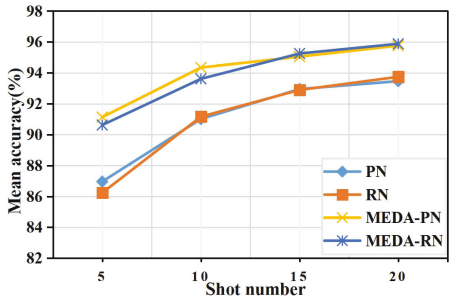

Figure 3: The mean accuracy on SNIPS with varying number of shots.

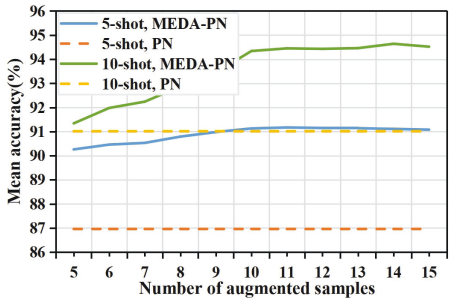

Figure 4: Comparing the 5-shot and 10-shot settings, the mean accuracy of different model (MEDA-PN vs. PN) changes with the number of augmented samples on SNIPS.

spectively. For SNIPS, MEDA versions are also effective, especially when K=10, they increase by 3.33% and 2.45%, respectively. This shows that the performance of conventional meta-learners can be further boosted when equipped with our data augmentation. Meanwhile, this also suggests that our proposed method is general and can work with different meta-learners.

MEDA vs. data augmentation. We also compare our generation strategy with common forms of data augmentation (EXTRA and RP) in Table 2 and Table 3. From the results, we can clearly see that our MEDA outperforms other data augmentation approaches. This confirms that our generation strategy produces more diverse and useful training examples than simple data augmentation. It is interesting to note that all data augmentation methods have achieved good results. This further shows that on the basis of meta-learning, data augmentation can expand prior knowledge and improve model performance.

7 Analysis

K-shot evaluations. In this section, we analyze the effectiveness of MEDA from the perspective of K-shot in different settings. So we compare our model's accuracy with other models such as prototypical networks and relation networks on the 2-way SNIPS task, and the shot number ranges from 5 to 20. As shown in Figure 3, we could observe that MEDA achieves the best classification performances in all settings. Notably, the margin of MEDA increases as the shot number decreases, showing that the model achieves performance improvement when the shot number is relatively small.

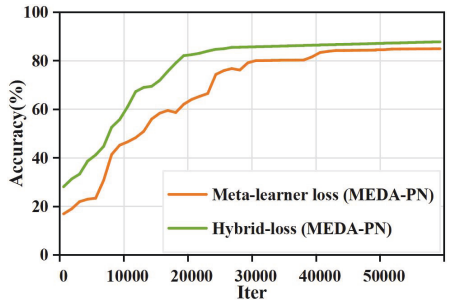

Figure 5: Accuracy of different versions of $\mathcal{L}$ on ARSC validation set.

Number of augmented samples. We also study how the accuracy of the model changes with the number of augmented samples on SNIPS. We therefore plot the 5-shot and 10-shot accuracy in Figure 4. The results show that as the number of augmented samples increases, the accuracy of the model goes up. The performance saturates after 10 samples. For this reason, we use 10 augmented samples in all of our experiments. Comparing the 5-shot and 10-shot settings, we find that adding five augmented samples has almost the same effect as using five real samples. This may suggest that our proposed MEDA is not simply copying samples, but generating meaningful samples for the model.

Hybrid loss analysis. As described in Section 4.2, we use the hybrid loss $\mathcal{L}$ to guide the training of our MEDA. To demonstrate its superiority, we compare it with the alone meta-learner loss $\mathcal{L}_{ML}$, i.e., without the generator loss $\mathcal{L}_{GEN}$. Figure 5 shows the accuracy curves of different loss functions with the number of iterations on ARSC. Comparing the different plots in Figure 5, we can see that using the hybrid loss function has a faster convergence speed than using the meta-learner loss function alone. At the same time, in Figure 5, we also analyze the change of the accuracy with the number of iterations. We find that, using the hybrid loss function has higher accuracy than using the meta-learner loss alone. This further shows that using the hybrid loss speeds up the convergence and improves the classification performance than using the meta-learner loss alone.

8 Conclusion

In this paper, we propose a novel framework MEDA for few-shot text classification that uses ball generator to increase the number of shots for meta-learner. MEDA is an end-to-end framework that jointly optimizes ball generator and meta-learner. The proposed model achieves state-of-the-art performance on two publicly datasets. In the future, we will explore the effectiveness of MEDA in other domains.

Acknowledgements

We thank all anonymous reviewers who provided valuable feedback. Our research was supported by the NSFC (No. 61936012,61976114) and the National Key R&D Program of China (No. 2018YFB1005102).

References

[Coucke *et al.*, 2018] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, 2018.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[DeVries and Taylor, 2017] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space. In *ICLR (Workshop)*, 2017.

[Fedus *et al.*, 2018] William Fedus, Ian Goodfellow, and Andrew M. Dai. MaskGAN: Better text generation via filling in the _____. In *International Conference on Learning Representations*, 2018.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[Geng *et al.*, 2019] Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. Induction networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3895–3904, 2019.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Koch *et al.*, 2015] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[Kumar *et al.*, 2019] Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. A closer look at feature space data augmentation for few-shot intent classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1–10, 2019.

[Malandrakis *et al.*, 2019] Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, Angeliki Metallinou, and Amazon Alexa AI. Controlled text generation for data augmentation in intelligent artificial agents. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 90–98, 2019.

[Reimers and Gurevych, 2019] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983, 2019.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[Sung *et al.*, 2018] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208. IEEE Computer Society, 2018.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[Wei and Zou, 2019] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, 2019.

[Welzl, 1991] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In *New results and new trends in computer science*, pages 359–370. Springer, 1991.

[Yu *et al.*, 2018] 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.

[Zhang *et al.*, 2019] Ningyu Zhang, Zhanlin Sun, Shumin Deng, Jiaoyan Chen, and Huajun Chen. Improving few-shot text classification via pretrained language representations. *CoRR*, abs/1908.08788, 2019.