Sun PF, Ouyang YW, Song DJ *et al.* Self-supervised task augmentation for few-shot intent detection. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(3): 527–538 May 2022. DOI 10.1007/s11390-022-2029-5

Self-Supervised Task Augmentation for Few-Shot Intent Detection

Peng-Fei Sun (孙鹏飞), Ya-Wen Ouyang (欧阳亚文), Ding-Jie Song (宋定杰), and Xin-Yu Dai* (戴新宇), *Member, CCF*

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

E-mail: {spf, ouyangyw, songdj}@smail.nju.edu.cn; daixinyu@nju.edu.cn

Received November 19, 2021; accepted April 8, 2022.

Abstract Few-shot intent detection is a practical challenge task, because new intents are frequently emerging and collecting large-scale data for them could be costly. Meta-learning, a promising technique for leveraging data from previous tasks to enable efficient learning of new tasks, has been a popular way to tackle this problem. However, the existing meta-learning models have been evidenced to be overfitting when the meta-training tasks are insufficient. To overcome this challenge, we present a novel self-supervised task augmentation with meta-learning framework, namely STAM. Firstly, we introduce the task augmentation, which explores two different strategies and combines them to extend meta-training tasks. Secondly, we devise two auxiliary losses for integrating self-supervised learning into meta-learning to learn more generalizable and transferable features. Experimental results show that STAM can achieve consistent and considerable performance improvement to existing state-of-the-art methods on four datasets.

Keywords self-supervised learning, task augmentation, meta-learning, few-shot intent detection

1 Introduction

Intent detection (ID) is a sub-field of text classification, which aims at classifying the user's utterance into predefined classes, that is, intents. In real-world scenarios, ID often suffers from a lack of training data, because new intents are frequently added to the existing collection and usually contain only a few samples. How to use a few samples for intent detection has become a challenging task. Few-shot intent detection (FSID) presents a promising solution for this challenge, and meta-learning is a mainstream paradigm to achieve the goal of few-shot learning.

Typically, meta-learning solves the few-shot learning problem by learning the prior knowledge from various meta-training tasks (episodes) in a so-called meta-training stage, so that the model can quickly adapt to new tasks with only a few samples. Although these methods have shown promising performance, the superior performance of these methods relies on enough meta-training tasks. In the scenario of insufficient meta-training tasks, meta-learners could overfit these meta-training tasks, which limits their generalization ability [1]. Unfortunately, the above issue is significant in FSID applications [2]. This is because there are many intents in real-world scenarios, and it is impractical to collect enough data for each intent and construct a large number of meta-training tasks.

Task augmentation, i.e., performing augmentation on each task to generate new tasks for auxiliary training, has been regarded as an explicit form of regularization to address the overfitting issue in meta-learning [2–5]. Numerous methods [1, 3] have theoretically and numerically justified that task augmentation effectively increases the types of meta-training tasks, improving generalization. However, existing methods are mainly on computer vision and have a little exploration in natural language. One potential reason is that, unlike images, text is discrete and few word substitutions may lead to significant semantic changes. The semantic feature is an essential feature of intent detection [6]. If task augmentation leads to the seman-

Regular Paper
Special Section on Self-Learning with Deep Neural Networks
This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 61936012 and 61976114.
*Corresponding Author

tic change or even affects the label to which the sample belongs, it will damage the model's performance. This makes task augmentation more challenging in FSID.

To address the challenge, in this paper, we propose a novel self-supervised task augmentation with meta-learning (STAM) framework for FSID. Firstly, STAM generates meta-training tasks that are semantically similar while having different features by static augmentation and dynamic augmentation. Concretely, as illustrated in Fig.1, static augmentation is to generate the augmented task for each original task by back-translation① before encoding. On the other hand, dynamic augmentation can obtain two different representations by forward-passing an original task twice with different dropout masks. Apart from the aforementioned task augmentation strategies, we also propose to weave self-supervision learning into meta-learning by adding two auxiliary losses in STAM (see Fig.2). The goal is to learn more generalizable and transferable features by using the structural information of data itself as the supervision signals to improve the generalization ability of meta-learning in meta-task insufficient scenarios. For this purpose, we devise the task consistency loss and the contrastive loss, respectively. The task consistency loss ensures that the predictions of meta-learner across different augmented tasks should be consistent. And the contrastive loss is devised through the maximization of mutual information between original and augmented samples. Furthermore, we demonstrate that these two auxiliary losses introduced in STAM are helpful to improve the generalization ability of meta-learning models. Our main contributions can be summarized as follows.

- Firstly, we introduce a simple yet effective task augmentation for text, which explores two different strategies and combines them to generate plenty of meta-training tasks and analyze their effects.
- We then propose a novel STAM model, which incorporates self-supervision learning to meta-learning by adding two additional learning objectives during the meta-training stage. We demonstrate that STAM can further improve the generalization capability under the insufficient meta-training tasks scenario.
- Finally, extensive experiments on four popular benchmark datasets show that our STAM achieves strong or even state-of-the-art performance.

2 Related Work

2.1 Self-Supervised Learning

For the past few years, self-supervised learning (SSL) has arisen much research interest in the field of natural language processing. It defines an annotation-free pretext task, which can enhance the model's learning capability without requiring any additional annotation effort. The main difference in existing SSL techniques is how the supervisory signal is obtained from the data. For example, in natural language processing, many studies [9–11] have explored masking part of a sentence and predicting the masked words from the remaining words. Other models [12, 13] adopt self-supervised learning objectives with unsupervised data by taking different views of the same sentence. Besides, SimCSE [14] predicts input sentence itself with

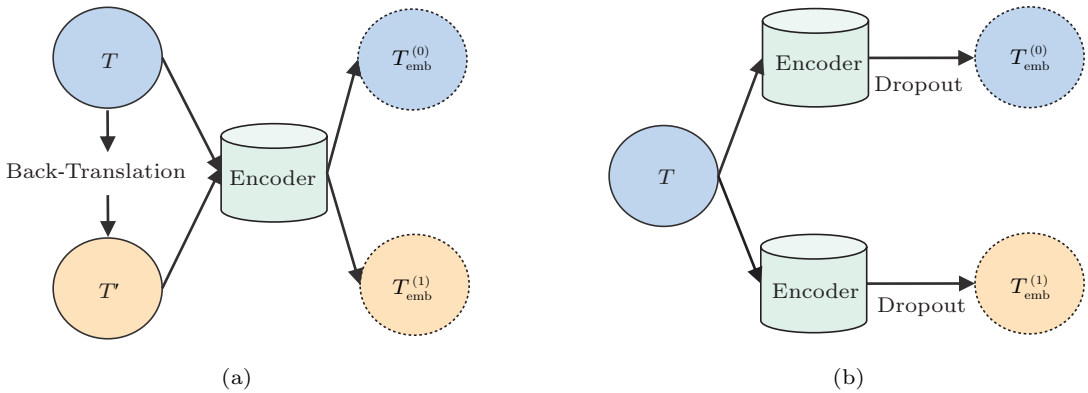


Fig.1. Comparisons among techniques. The blue denotes original tasks and the yellow denotes augmented tasks. (a) Static augmentation: with back-translation to create new tasks before encoding. (b) Dynamic augmentation: with only dropout as noise to generate new tasks in the encoding process.

①Back-translation [7, 8] refers to the procedure of translating an example into the target language using a machine translation model and then re-translating it back into the source language to obtain an augmented example.

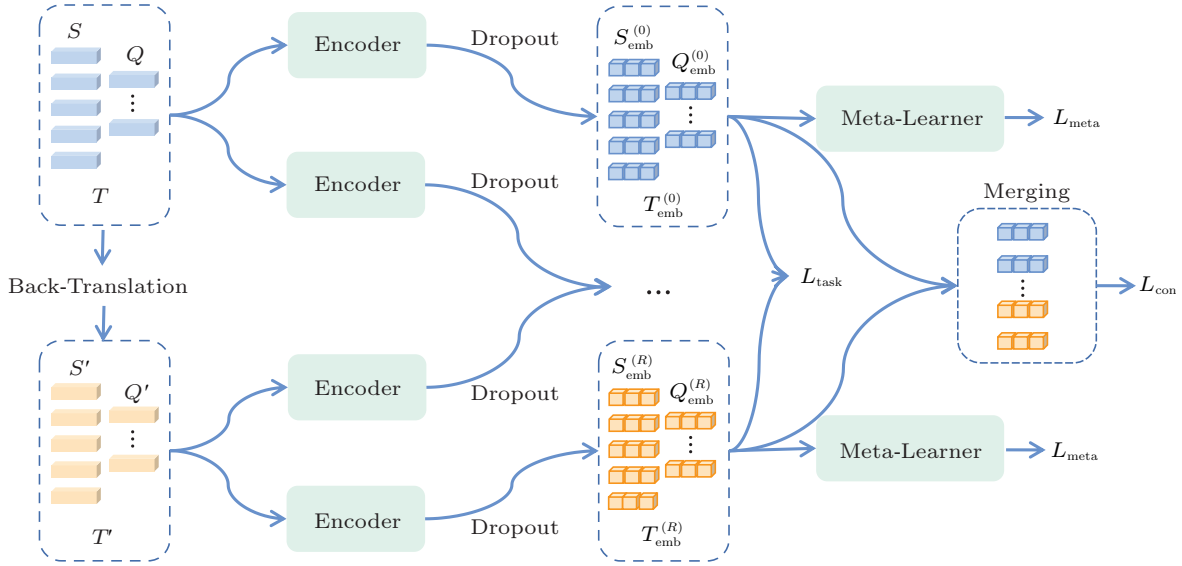


Fig.2. Schematic illustration of the proposed STAM. For clarity, only the 5-shot setting is presented here. The blue cuboids denote original samples and the yellow cuboids denote augmented samples.

dropout noise to produce superior sentence embeddings from either unlabeled or labeled data. Recently, the potential of SSL for few-shot learning has been explored in [15–17]. [15,16] propose to integrate SSL into few-shot learning by adding an auxiliary SSL pretext task in a few-shot model. In [17], authors introduced an episode-level pretext task to integrate SSL into the episode training in FSL. Different from the existing SSL approaches for few-shot learning, we propose new task augmentation and introduce two auxiliary losses to integrate SSL into meta-learning. The most related work is SMLMT [2], which pretrains the model on the pretext task and fine-tunes the model on the downstream task. In contrast, our approach proposes two different strategies to generate meta-training tasks and designs an end-to-end framework STAM that closely integrates SSL with meta-learning.

2.2 Meta Learning

Recently, meta-learning algorithms have received widespread attention in the few-shot learning settings. Generally, these methods can be roughly divided into three categories: "learning to optimize", "learning to model" and "learning to compare". The "learning to optimize" methods are known as optimization-based methods [18,19], which learn the parameter initialization for adapting the model to a new task with few steps of parameter updating. The "learning to model" methods are called as the model-based methods [20,21], which focus on designing either specific model structures or parameters capable of rapid updating. The "learning to compare" methods, namely metric learning based methods [22–25], attempt to learn a good embedding and an appropriate comparison metric. In the present study, although other methods can also be integrated into our STAM, we mainly use the "learning to compare" methods as the meta-learner for our STAM. This is because they are efficient and straightforward.

In addition, meta-learning can also be integrated with other machine learning frameworks. For instance, recent application of meta-learning integrates reinforcement learning for robots to learn basic skills and react timely to rare situations [26,27]. Other studies [28,29] incorporate graphs into meta-learning. GPN [28] is a novel meta-learner that explicitly relates tasks on a graph explaining the relations of predicted classes in few-shot learning. [29] proposes a meta-relational learning framework to transfer the general relational information from existing triples to incomplete triples. Besides, [30,31] have recently explored the integration of self-supervised learning into meta-learning. These approaches use the self-supervised training loss as an auxiliary task to improve the model. Our approach also follows this line of work. To further improve the generalization of the model, we design two auxiliary losses to integrate self-supervised learning into meta-learning.

2.3 Few-Shot Intent Detection

For FSID, several studies have attempted to solve it using different techniques. For instance, [32] pro-

poses the pseudo-labeling technique for few-shot intent detection, which devises the hierarchical clustering inspired method for assigning weighted pseudo-labels to unlabeled user utterances. [33] introduces intent detection methods backed by pretrained dual sentence encoders. In [34], authors applied data augmentation in conjunction with meta-learning to reduce sampling bias. In [35], authors proposed a simple yet effective few-shot intent detection method by leveraging contrastive pretraining and fine-tuning. Although these methods have achieved good results, they ignore the overfitting problem caused by insufficient meta-training tasks. Our work focuses on this problem by constructing STAM to alleviate it effectively.

3 Preliminaries

3.1 Problem Formulation

In this paper, we establish few-shot intent detection in the meta-learning framework [23]. Specifically, C_{train} denotes a set of seen classes, and C_{test} is a set of unseen classes, where $C_{\text{train}} \cap C_{\text{test}} = \emptyset$. Our goal is to learn a meta-learner on C_{train}, so that we can make predictions over novel classes, which belong to C_{test}.

To emulate the few-shot scenario, the meta-learner learns from a group of n-way, k-shot meta-training tasks sampled from C_{train} and is then evaluated in a similar way on C_{test}. Each meta-training task T can be divided into two sets: 1) support set $S = \{(\boldsymbol{x}_i, y_i), i = 1, 2, ..., n \times k\}$ is formed by randomly selecting k samples from each of the n classes and 2) query set $Q = \{(\boldsymbol{x}_i, y_i), i = 1, 2, ..., n \times m\}$ contains other m samples from the same n classes. For simplicity, we denote $l_s = n \times k$ and $l_q = n \times m$. In the meta-training stage, the training is done by feeding the support set S to the model and updating the model based on loss over the query set Q. During meta-testing, we apply the same mechanism to test whether our model can indeed adapt quickly to novel classes C_{test}.

3.2 Meta-Learner

We use the prototypical network (ProtoNet) [24] as our meta-learner. ProtoNet is a non-parametric method, which is composed of the encoder $f_\phi(\cdot)$ and the non-parametric classifier $g(\cdot)$. Concretely, for each task T, we first use the encoder $f_\phi(\cdot)$ to map all samples into the feature space, and then compute the mean feature embedding of support samples for each class $c \in C_{\text{train}}$ as the prototype $\boldsymbol{p}_c$:

$$\boldsymbol{p}_c = \frac{1}{k} \sum_{(\boldsymbol{x}_i, y_i) \in S} f_\phi(\boldsymbol{x}_i) \cdot \mathbb{I}(y_i = c),$$

where ϕ is the parameters of the encoder. $\mathbb{I}(\cdot)$ denotes the indicator function with its output being 1 if the input is true or 0 otherwise. Once the class prototypes are obtained from the support set, the distances between each query sample and the prototypes can be calculated. Therefore, for each query sample $\boldsymbol{x}_i \in Q$, the posterior probability given class c is as follows:

$$\begin{aligned} &P(y_i = c \mid \boldsymbol{x}_i, S; \phi) \\ &= \frac{\exp(-d(f_{\boldsymbol{\phi}}(\boldsymbol{x}_i), \boldsymbol{p}_c))}{\sum_{c' \in C_{\text{train}}} \exp(-d(f_{\boldsymbol{\phi}}(\boldsymbol{x}_i), \boldsymbol{p}_{c'}))}, \end{aligned} \tag{1}$$

where $d(\cdot, \cdot)$ represents the distance calculation function, and the Euclidean distance is used here. Finally, the loss function used to meta-learn the parameters ϕ is defined as:

$$L_{\text{meta}} = -\frac{1}{l_q} \sum_{(\boldsymbol{x}_i, y_i) \in Q} \log(P(y_i \mid \boldsymbol{x}_i, S; \phi)). \tag{2}$$

4 Proposed Method

In this section, we introduce the proposed STAM (Fig. 2). Firstly, we describe the details of task augmentation strategies (Subsection 4.1). Based on that, we present two auxiliary losses to incorporate self-supervision learning into the meta-learning (Subsection 4.2). Finally, we present the total loss (Subsection 4.3) and a complete algorithm for STAM (Subsection 4.4).

4.1 Task Augmentation

Similar to data augmentation, task augmentation aims at creating new and realistic-looking tasks by applying a transformation to the original task② [36]. In this paper, we introduce three task augmentation strategies: static augmentation, dynamic augmentation, and their combination.

Static Augmentation. The goal of this strategy is to create new tasks by performing task augmentation before training. To this end, we use the back-translation as transformation, since back-translation

②The main difference between data augmentation in classical machine learning and task augmentation in meta-learning is the minimum unit of augmentation. For data augmentation in classical machine learning, the aim is to generate more diverse examples in a single task. Task augmentation in meta-learning aims to generate more diverse tasks but not the number of examples in each task.

may maximize language diversity while maintaining semantic consistency. For each task T, we translate it to another language, and then translate it back to the source language as the augmented task T'. Finally, we feed the original task T and the augmented task T' into the transformer-based encoder to obtain the corresponding feature embeddings $T^{(0)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}_i)|\boldsymbol{x}_i \in T\}$ and $T^{(1)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}'_i)|\boldsymbol{x}'_i \in T'\}$.

Dynamic Augmentation. This strategy is to achieve task augmentation by randomly dropping out neurons during training. Concretely, given a task $T = \{(\boldsymbol{x}_i, y_i), i = 1, 2, ..., (l_s + l_q)\}$, we use different dropout masks to feed the same task T into the encoder twice. We therefore obtain two corresponding feature embeddings $T^{(0)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}_i, w)|\boldsymbol{x}_i \in T\}$ and $T^{(1)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}_i, w)|\boldsymbol{x}_i \in T\}$, where w is a pre-defined probability for dropout. $T^{(0)}_{\text{emb}}$ and $T^{(1)}_{\text{emb}}$ are the original and the augmented task feature embedding, respectively. We also use the transformer-based methods as the encoder here, thereby w is the standard dropout mask and we do not need to add any additional dropout.

Integration. From the above discussion, we conclude that two task augmentation strategies are performed in different ways. Therefore, we introduce their combination in STAM (see Fig.2). Specifically, for each original task T, we firstly apply static augmentation to generate augmented task T'. Then, we obtain the extended task feature embeddings $T^{(0)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}_i, w)|\boldsymbol{x}_i \in T\}$, $T^{(1)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}_i, w)|\boldsymbol{x_i} \in T\}$, $T^{(2)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}'_i, w)|\boldsymbol{x}'_i \in T'\}$ and $T^{(3)}_{\text{emb}} = \{f_\phi(\boldsymbol{x}'_i, w)|\boldsymbol{x}'_i \in T'\}$ by feeding the original task and augmenting the task into dynamic augmentation. Note that $T^{(0)}_{\text{emb}}$ is the original task feature embedding, $T^{(r)}_{\text{emb}}(r = 1, ..., R)$ is the augmented task feature embedding, and R is the number of augmented tasks ($R = 3$ in our implementation).

4.2 Auxiliary Losses

In order to make full use of the structural information of the data itself to learn more generalizable and transferable features, we design two auxiliary functions: task consistency loss L_{task} and contrastive loss L_{con}.

Task Consistency Loss. We design the task consistency loss by forcing the meta-learner learned over the different augmented tasks to produce consistent predictions. These are various options on how to enforce such consistency. Inspired by [37], we adopt the Kullback-Leibler (KL) divergence. More formally, for each task T, we first pass it through the task augmentation and its extended task feature embeddings $T^{(r)}_{\text{emb}}(r = 0, ..., R)$ are obtained. Then, given a task feature embedding $T^{(r)}_{\text{emb}} = (S^{(r)}_{\text{emb}}, Q^{(r)}_{\text{emb}})$, we compute the probability distribution $P^{(r)}_i$ with (1) over the query samples $\boldsymbol{x}^{(r)}_i \in Q^{(r)}_{\text{emb}}(i = 1, ..., l_q)$. Finally, the task consistency loss is computed with the bidirectional KL divergence:

$$L_{\text{task}} = \frac{1}{2}\sum_{r=1}^{R}\sum_{i=1}^{l_q}(D_{\text{KL}}(P^{(0)}_i||P^{(\text{r})}_i) + D_{\text{KL}}(P^{(\text{r})}_i||P^{(0)}_i)). \quad (3)$$

Contrastive Loss. Following [38], we adopt the normalized temperature-scaled cross-entropy loss as the contrastive loss. Concretely, we first merge the extended task and obtain a merged task. For each sample in the merged support set, we select the correlated augmented samples as positive, while all the other samples are regarded as negative. The corresponding loss for a positive pair of samples $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is shown as the following:

$$L_{i,j} = -\log \frac{\exp\left(\text{Sim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)/\tau\right)}{\sum_{k=1}^{M}\mathbb{I}_{[k\neq i]}\exp\left(\text{Sim}\left(\boldsymbol{x}_i, \boldsymbol{x}_k\right)/\tau\right)}.$$

Here, M is the number of samples in the merged support set, $\text{Sim}(\cdot,\cdot)$ denotes the inner (dot) product, and τ is a scalar temperature parameter. Finally, the contrastive loss L_{con} is computed across all positive pairs:

$$L_{\text{con}} = \frac{1}{l_s \times R}\sum_{k=1}^{l_s}\sum_{r=1}^{R} L_{k,k+r\times l_s}. \quad (4)$$

4.3 Total Loss

We argue that two auxiliary losses act as the regularization term to prevent overfitting for meta-learning. Thus, the total loss for STAM is finally stated as:

$$L_{\text{total}} = L_{\text{avg}} + \alpha \times L_{\text{task}} + \beta \times L_{\text{con}}, \quad (5)$$

where L_{avg} is the average of L_{meta} for all tasks. α and β are the loss weight hyper-parameters. Note that self-supervised learning is only exploited during meta-training.

4.4 Full Algorithm

As we have mentioned above, during the episode training, we first sample the task T from C_{train}. We then use task augmentation to obtain the extended feature embeddings $T^{(r)}_{\text{emb}}(r = 0, ..., R)$. Next, we compute the meta-learning loss L_{meta} with (2) for all $T^{(r)}_{\text{emb}}$ and average all L_{meta} as L_{avg}. Meanwhile, we use (3) to

compute the task consistency loss L_{task} for the original and augmented tasks, and merge the extended tasks to compute the contrastive loss L_{con} with (4). Finally, all losses are combined by (5) and gradient updates are performed. Thus, our STAM is summarized in Algorithm 1.

Algorithm 1. STAM

Input: labeled dataset C_{train}
hyper-parameters α, β
temperature parameter τ
while not converge **do**
Randomly sample original task T from C_{train}, where $T = (S, Q)$
Obtain the extended feature embeddings $T^{(r)}_{\text{emb}} = (S^{(r)}_{\text{emb}}, Q^{(r)}_{\text{emb}})(r = 0, ..., R)$ from T using task augmentation
Compute L_{meta} with (2) for all $T^{(r)}_{\text{emb}}$ and average all L_{meta} as L_{avg}
Compute L_{task} with (3)
Merge the extended tasks and compute L_{con} with (4)
Compute the total loss L_{total} with (5)
Update parameter $\boldsymbol{\phi}$ by minimizing L_{total}
end

5 Experiments

5.1 Datasets

To prove the effectiveness of our method, we evaluate STAM on three intent detection datasets: SNIPS [39], CLINC150 [40] and Liu [41]. Considering that intents contained in these datasets are from different domains, training and test data might not be from the same domain, which makes these datasets more challenging. In addition, to evaluate our method's generality, we also test it on the text classification dataset Huff [42]. Table 1 summarizes the statistics of the different datasets.

SNIPS[39]. This dataset is about personal voice assistants, and collected in a crowd sourced fashion. We select two intents (AddToPlaylist, RateBook) as the test set and the other five intents are divided into the training set (BookRestaurant, PlayMusic, SearchCreativeWork) and the validation set (GetWeather, SearchScreeningEvent), respectively. We evaluate the performance on the test set in 2-way-k-shot settings, where $k = \{1, 5\}$.

CLINC150[40]. This dataset is also about voice assistants, and supports both in-scope and out-of-scope data. In our experiments, we follow [43] to discard the out-of-scope data and only keep the in-scope data, i.e., 150 intents across 10 different domains. We choose 75 intents among five different domains (banking, kitchen, home, auto commute and small talk) as the training set and sample 45 intents from three domains (travel, work, meta) as the test set. The remaining intents are considered as the validation set.

Liu[41]. This dataset is collected from Amazon Mechanical Turk, and contains around 2.5k examples from 54 intents. We randomly choose nine intents (audiobook, currency, joke, podcasts, post, quirky, recipe, taxi, wemo_off) and eight intents (dontcare, game, repeat, ticket, factoid, dislikeness, radio, hue_lightchange) as the test set and the validation set, respectively. The remaining 37 intents are considered as the training set.

Huff[42]. The dataset is used for text classification, which contains around 200k new headlines published on HuffPost from 2012 to 2018. These headlines are split among 41 classes. We follow [42] to split the dataset into few-shot setting, and the training, test, and validation set are divided into 20, 16, and 5 classes, respectively.

5.2 Baselines

We compare against several baselines as follows. 1) We first compare it with ProtoNet [24] and RelationNet [25] (using a pretrained BERT as the text encoder), since [44] has shown that equipping a BERT model on the prototypical network and the relation network can achieve the significant results for FSID. 2) We also compare it with other state-of-the-art models, in-

Table 1. Main Statistics of Datasets

	Vocabulary Size	#Sentences	#Classes (Train/Validation /Test (Total))	Number of Tokens per Sentence (Mean Value ± Standard Deviation)	Number of Sentences per Class (Mean Value ± Standard Deviation)
SNIPS [39]	11 641	14 484	3/2/2 (7)	9.0 ± 3.2	2 069 ± 21
CLINC150 [40]	7 284	22 500	75/30/45 (150)	8.5 ± 3.3	150 ± 0
Liu [41]	12 046	25 478	37/8/9 (54)	7.5 ± 3.4	472 ± 823
Huff [42]	8 218	36 900	20/5/16 (41)	11.5 ± 0.9	900 ± 0

Note: # denotes the number of something, e.g., #Sentences is the number of sentences. The last two columns are the mean values and their standard deviation.

cluding Pseudo-Label [32] and SMLMT [2]. The Pseudo-Label is a semi-supervised few-shot intent detection model; hence we can easily apply it for our experiments. SMLMT is a self-supervised meta-learning approach, and we use the code of SMLMT③ and run it according to our experimental settings.

5.3 Implementation Details

Our implementation is based on Pytorch④. For a fair comparison, we adopt BERT [9] for all methods f_ϕ to calculate the sentence embedding. The Adam [45] optimizer is adopted with the initial learning rate of 1.0×10^{-3} for all the experiments. The hyper-parameters α and β are selected from $\{0.01, 0.1, 0.5, 1, 1.5\}$ and $\{0.001, 0.01, 0.1, 0.5, 1\}$ according to the performance on the validation set respectively. We use the same value of 0.07 as in [46] for temperature parameter τ. We repeat the experiments with five different seeds and report the mean classification accuracy because the few-shot experiments are sensitive to initialization and hyper-parameters. For STAM, we follow [7] to use an English-to-German [47] and a German-to-English [48] machine translation model for back-translation used in static augmentation. More specifically, we translate each English sentence into one German sentence and re-translate it to the English sentence. All experiments are performed on a GPU node with a 10-core Intel Xeon Silver 4114 CPU and four NVIDIA GeForce 1080Ti GPUs.

6 Main Results

Overall Performance. As shown in Table 2, the proposed STAM method achieves the best performance across all datasets. On average, STAM improves the 1-shot accuracy by 1.3% and the 5-shot accuracy by 2.2%, against the best baseline for each dataset. This validates the general effectiveness of our proposed STAM in addressing FSID.

Comparison with Meta-Learners. Compared with ProtoNet, STAM improves the performance by 5.05%, 0.77%, 2.08% and 7.32% on SNIPS, CLINC150, Liu, and Huff under the 5-shot setting, respectively. Note that on the text classification dataset Huff, STAM shows considerable improvement. Similar consistent gains of 1%–5% can be observed under the 1-shot setting.

Furthermore, we compare different meta-learning algorithms as meta-learner in our framework. For RelationNet, STAM-RelationNet is also effective, especially on SNIPS and Liu, and they increase by 1.93% and 0.86% on 5-shot, respectively. This shows that the meta-learners' generalization ability can be improved by equipping self-supervised task augmentation. Moreover, we find that STAM is better than STAM-RelationNet in most cases. This indicates that in STAM, the encoder that benefits from auxiliary losses can output more discriminative representation and reduce dependency on the additional metric network (we will discuss more in Section 7).

Comparison with State-of-the-Arts. We also compare STAM with other SOTA methods, i.e., Pesudo-Label [32] and SMLMT [2]. For instance, STAM outperforms SMLMT (similar to our method) by 1.68% and 5.21% on SNIPS and Huff under the 5-shot setting, respectively. On 1-shot, the same trend can be observed with consistent 2%–5% gains. From the results, we can clearly see that our STAM outperforms these methods.

Table 2. Comparison of Mean Accuracy (%) on Four Datasets

Method	SNIPS [39]		CLINC150 [40]		Liu [41]		Huff [42]	
	1-Shot	5-Shot	1-Shot	5-Shot	1-Shot	5-Shot	1-Shot	5-Shot
ProtoNet [24]	81.42±0.22	86.96±0.85	91.78±1.19	97.60±0.13	79.33±1.15	91.17±0.31	49.47±1.20	65.89±1.08
RelationNet [25]	82.01±1.01	86.23±0.65	92.04±0.91	97.19±0.22	80.47±1.36	91.98±0.56	50.23±1.36	64.16±1.40
Pseudo-Label [32]	83.50±0.27	90.50±0.94	92.90±0.26	97.18±0.43	81.50±1.27	92.33±0.17	52.11±0.03	67.28±0.83
SMLMT [2]	83.97±0.11	90.33±0.95	92.32±0.84	97.17±0.41	82.73±0.72	92.55±0.13	52.74±0.03	66.91±0.62
STAM	**85.13**±0.41*	92.01±0.50⋆	**93.73**±0.41⋆	**98.37**±0.12*	**84.40**±0.84⋆	93.25±0.35⋆	**53.69**±0.42*	**72.12**±0.45*
STAM-RelationNet	85.07±0.56⋆	**92.26**±0.33⋆	92.91±0.76	98.10±0.13*	84.01±0.43*	**93.41**±0.14*	52.92±0.02⋆	71.61±0.97*

Note: SNIPS is the 2-way setting, and the others are the 5-way setting. STAM-RelationNet means replacing ProtoNet in STAM with RelationNet. Most of our methods are significantly better than the baseline models with p-value < 0.01 (marked by $*$) and p-value < 0.05 (marked by $\star$) using t-test. The best results are bolded.

③https://github.com/iesl/metanlp, March 2022.

④https://pytorch.org/, March 2022.

This further demonstrates the effectiveness of STAM.

7 Ablation Studies and Analysis

Effect of Augmentation Strategies. We investigate the impact of different augmentation strategies on the model. These results are shown in Table 3. We find that the performance drops clearly when we independently remove static augmentation or dynamic augmentation. This shows that both augmentation strategies contribute to the model. Furthermore, we can also see that the performance can be further improved by simultaneously adopting both augmentation strategies, which confirms that two augmentation strategies are complementary.

Table 3. Ablation Study on the Different Augmentation Strategies Contributions in 5-Way Setting

Setting	Liu [41]		Huff [42]	
	1-Shot	5-Shot	1-Shot	5-Shot
STAM	84.40	93.25	53.69	72.12
w/o dynamic augmentation	81.71	92.54	51.99	70.38
w/o static augmentation	81.62	92.56	51.25	69.88

Note: Mean accuracy is reported on the Liu and Huff datasets (w/o means without).

Effect of Loss Function. We also study the contribution of task consistency loss L_{task} and contrastive loss L_{con} for our method. To this end, we conduct an ablation study on Liu and Huff datasets by setting the auxiliary loss weight as 0. The results are shown in Table 4. It can be observed that both losses contribute to the performance. Additionally, we also visualize the representation of the test set using t-SNE in Fig.3. We can see that representations learned with auxiliary losses exhibit a more clear and more compact cluster structure than those without auxiliary losses, such as intents Sports, Religion (see Fig.3(a) vs Fig.3(b)). This further validates the effectiveness of the two auxiliary losses.

Table 4. Ablation Study of the Objective Function in 5-Way Setting

L_{avg}	L_{con}	L_{task}	Liu [41]		Huff [42]	
			1-Shot	5-Shot	1-Shot	5-Shot
✓			81.11	92.29	49.48	65.89
✓	✓		81.47	92.36	52.89	72.08
✓		✓	81.69	92.54	51.42	68.78
✓	✓	✓	84.40	93.25	53.69	72.12

Note: Mean accuracy is reported on the Liu and Huff datasets.

Effect of Hyper-Parameters. In order to evaluate the influence of hyper-parameters α and β on our method, we train our STAM with different values of hyper-parameters on the Liu dataset. From results in Fig.4, with the increase of α values, the performance increases first and then falls. The highest accuracy is obtained on $\alpha = 1$. This indicates that if α is too small, L_{task} would not help much; on the other hand, if it is too large, it may overshoot the minima and fail to converge. Fig.4 shows that the performance of β shows the same trend. However, when $\beta > 0.1$, the accuracy falls sharply. We conjecture that the small β is necessary due to the difference in scale of the contrastive and the other losses.

Analysis of Training Efficiency. To analyze the effect of STAM on training efficiency, we compare STAM with three strategies: without dynamic augmentation, without static augmentation, and without task augmentation. The performance changes of the validation set as the training proceeds are shown in Fig.5. We can observe that the training process introduces little extra costs but distinct benefits. Specifically, our method only takes about 7 minutes more to bring a performance boost.

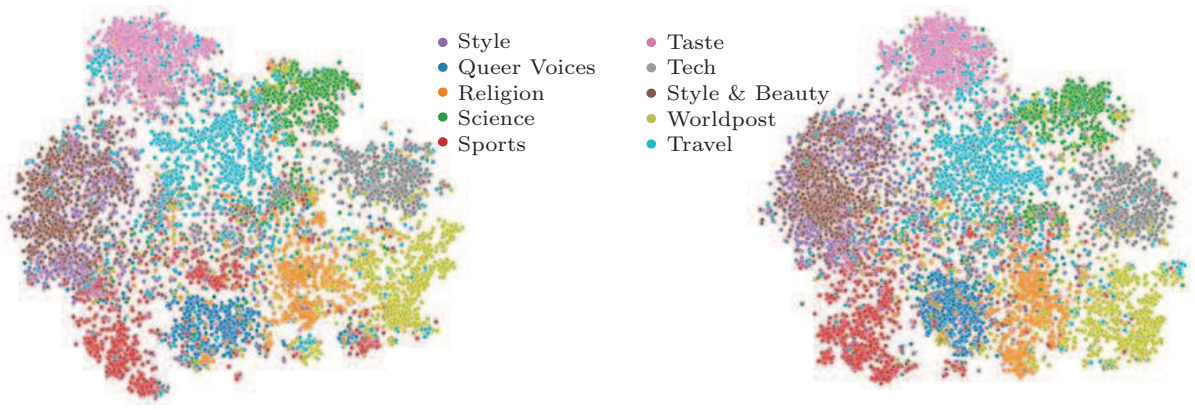


Fig.3. Visualizations of data distributions on Huff [42]. (a) Without two auxiliary losses. (b) With two auxiliary losses.

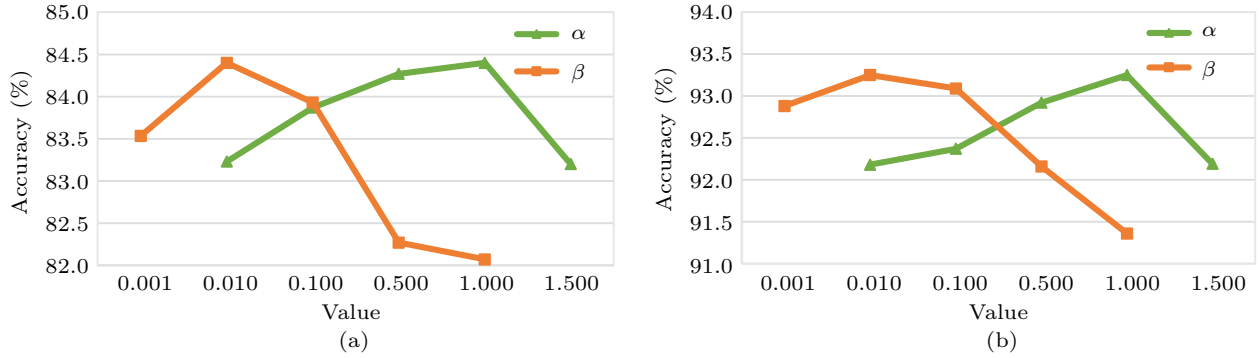


Fig.4. Performance of STAM with different hyper-parameters α and β on Liu [41]. (a) 5-way 1-shot. (b) 5-way 5-shot.

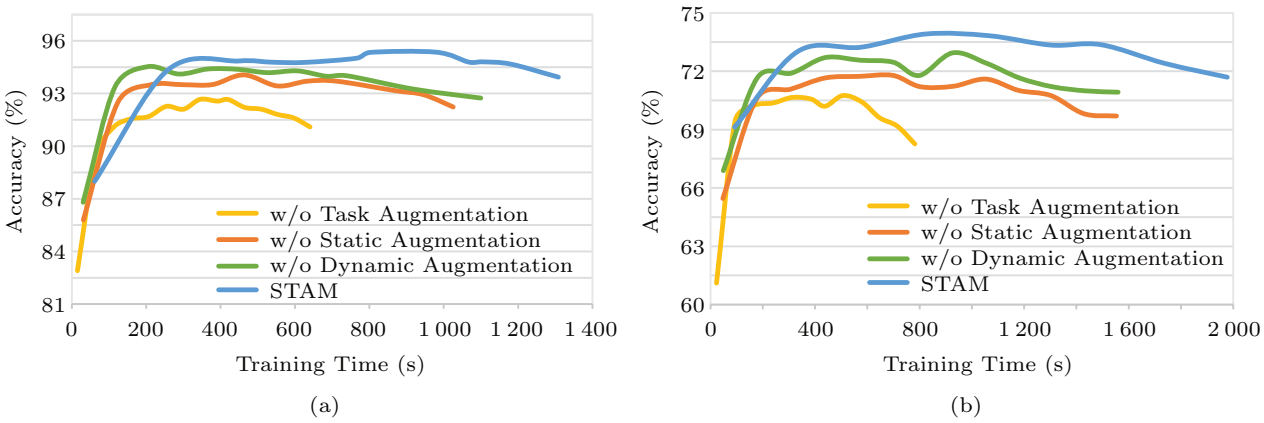


Fig.5. Comparisons of the training efficiency of STAM and others as the training proceeds in a 5-way 5-shot setting on two datasets. (a) Liu [41]. (b) Huff [42].

8 Conclusions

This paper focused on realistic few-shot intent detection and proposed a novel STAM model to overcome the potential overfitting problem caused by the insufficient meta training tasks. The STAM includes our proposed task augmentation to increase the meta-training tasks for auxiliary training and devises two auxiliary losses to incorporate self-supervision learning into meta-learning. The effectiveness of STAM is validated on four benchmark datasets in both 1-shot and 5-shot settings. Compared with some state-of-the-art methods, STAM improved the mean accuracy by about 1.3% and 2.2% in 1-shot and 5-shot settings, respectively.

The potential bottleneck of our research is that the task augmentation relies on back-translation, which in turn depends on machine translation models. Currently, machine translation models have weak support for long text, which will limit the application scenarios of our model.

For further work, we plan to explore different strategies to augment the number of meta-tasks, various designs of self-supervised tasks, and the effectiveness of STAM in other domains.

References

[1] Yao H, Zhang L, Finn C. Meta-learning with fewer tasks through task interpolation. arXiv:2106.02695, 2021. https://arxiv.org/abs/2106.02695, June 2021.

[2] Bansal T, Jha R, Munkhdalai T, McCallum A. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proc. the 2020 Conference on Empirical Methods in Natural Language Processing*, November 2020, pp.522-534. DOI: 10.18653/v1/2020.emnlp-main.38.

[3] Yao H, Huang L K, Zhang L *et al.* Improving generalization in meta-learning via task augmentation. In *Proc. the 38th International Conference on Machine Learning*, July 2021, pp.11887-11897.

[4] Wang H, Deng Z H. Cross-domain few-shot classification via adversarial task augmentation. In *Proc. the 30th International Joint Conference on Artificial Intelligence*, August 2021, pp.1075-1081. DOI: 10.24963/ijcai.2021/149.

[5] Murty S, Hashimoto T, Manning C D. DReCa: A general task augmentation strategy for few-shot natural language

inference. In *Proc. the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2021, pp.1113-1125. DOI: 10.18653/v1/2021.naacl-main.88.

[6] Xia C, Zhang C, Yan X, Chang Y, Yu P S. Zeroshot user intent detection via capsule neural networks. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 31-November 4, 2018, pp.3090-3099. DOI: 10.18653/v1/D18-1348.

[7] Edunov S, Ott M, Auli M, Grangier D. Understanding back-translation at scale. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 31-November 4, 2018, pp.489-500. DOI: 10.18653/v1/D18-1045.

[8] Sennrich R, Haddow B, Birch A. Improving neural machine translation models with monolingual data. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.86-96. DOI: 10.18653/v1/P16-1009.

[9] Devlin J, Chang M W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2019, pp.4171-4186. DOI: 10.18653/v1/N19-1423.

[10] Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692, 2019. https://arxiv.org/abs/1907.11692, July 2021.

[11] Conneau A, Khandelwal K, Goyal N, Chaudhary V, Wenzek G, Guzmán F, Grave E, Ott M, Zettlemoyer L, Stoyanov V. Unsupervised cross-lingual representation learning at scale. In *Proc. the 58th Annual Meeting of the Association for Computational Linguistics*, July 2020, pp.8440-8451. DOI: 10.18653/v1/2020.acl-main.747.

[12] Zhang Y, He R, Liu Z, Lim K H, Bing L. An unsupervised sentence embedding method by mutual information maximization. In *Proc. the 2020 Conference on Empirical Methods in Natural Language Processing*, November 2020, pp.1601-1610. DOI: 10.18653/v1/2020.emnlp-main.124.

[13] Meng Y, Xiong C, Bajaj P, Tiwary S, Bennett P, Han J, Song X. COCO-LM: Correcting and contrasting text sequences for language model pretraining. In *Proc. the 35th International Conference on Neural Information Processing Systems*, December 2021.

[14] Gao T, Yao X, Chen D. SimCSE: Simple contrastive learning of sentence embeddings. In *Proc. the 2021 Conference on Empirical Methods in Natural Language Processing*, November 2021, pp.6894-6910. DOI: 10.18653/v1/2021.emnlp-main.552.

[15] Gidaris S, Bursuc A, Komodakis N, Pérez P, Cord M. Boosting few-shot visual learning with self-supervision. In *Proc. the 2019 IEEE/CVF International Conference on Computer Vision*, October 27-November 2, 2019, pp.8059-8068. DOI: 10.1109/ICCV.2019.00815.

[16] Su J C, Maji S, Hariharan B. When does self-supervision improve few-shot learning? In *Proc. the 16th European Conference on Computer Vision*, August 2020, pp.645-666. DOI: 10.1007/978-3-030-58571-6_38.

[17] Zhang M, Zhang J, Lu Z, Xiang T, Ding M, Huang S. IEPT: Instance-level and episode-level pretext tasks for few-shot learning. In *Proc. the 9th International Conference on Learning Representations*, May 2021.

[18] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. the 34th International Conference on Machine Learning*, August 2017, pp.1126-1135.

[19] Nichol A, Achiam J, Schulman J. On first-order meta-learning algorithms. arXiv:1803.02999, 2018. https://arxiv.org/abs/1803.02999, October 2021.

[20] Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T. One-shot learning with memory-augmented neural networks. arXiv:1605.06065, 2016. https://arxiv.org/abs/1605.06065, May 2021.

[21] Munkhdalai T, Yu H. Meta networks. In *Proc. the 34th International Conference on Machine Learning*, August 2017, pp.2554-2563.

[22] Koch G, Zemel R, Salakhutdinov R *et al*. Siamese neural networks for one-shot image recognition. In *Proc. the 32nd International Conference on Machine Learning Deep Learning Workshop*, July 2015.

[23] Vinyals O, Blundell C, Lillicrap T, Kavukcuoglu K, Wierstra D. Matching networks for one shot learning. In *Proc. the 2016 Annual Conference on Neural Information Processing Systems*, December 2016, pp.3630-3638.

[24] Snell J, Swersky K, Zemel R S. Prototypical networks for few-shot learning. In *Proc. the 2017 Annual Conference on Neural Information Processing Systems*, December 2017, pp.4077-4087.

[25] Sung F, Yang Y, Zhang L, Xiang T, Torr P H, Hospedales T M. Learning to compare: Relation network for few-shot learning. In *Proc. the 2018 IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp.1199-1208. DOI: 10.1109/CVPR.2018.00131.

[26] Nagabandi A, Clavera I, Liu S, Fearing R S, Abbeel P, Levine S, Finn C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *Proc. the 7th International Conference on Learning Representations*, May 2019.

[27] Rakelly K, Zhou A, Finn C, Levine S, Quillen D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proc. the 36th International Conference on Machine Learning*, June 2019, pp.5331-5340.

[28] Liu L, Zhou T, Long G, Jiang J, Zhang C. Learning to propagate for graph meta-learning. In *Proc. the 2019 Annual Conference on Neural Information Processing Systems*, December 2019, pp.1037-1048.

[29] Chen M, Zhang W, Zhang W, Chen Q, Chen H. Meta relational learning for few-shot link prediction in knowledge graphs. In *Proc. the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, November 2019, pp.4216-4225. DOI: 10.18653/v1/D19-1431.

[30] Zhu K, Zhai W, Cao Y. Self-supervised tuning for few-shot segmentation. In *Proc. the 29th International Joint Conference on Artificial Intelligence*, July 2020, pp.1019-1025. DOI: 10.24963/ijcai.2020/142.

[31] Liu S, Davison A J, Johns E. Self-supervised generalisation with meta auxiliary learning. In *Proc. the 2019 Annual Conference on Neural Information Processing Systems*, December 2019, pp.1677-1687.

[32] Dopierre T, Gravier C, Subercaze J, Logerais W. Few-shot pseudo-labeling for intent detection. In *Proc. the 28th International Conference on Computational Linguistics*, December 2020, pp.4993-5003. DOI: 10.18653/v1/2020.coling-main.438.

[33] Casanueva I, Temčinas T, Gerz D, Henderson M, Vulić I. Efficient intent detection with dual sentence encoders. In *Proc. the 2nd Workshop on Natural Language Processing for Conversational AI*, July 2020, pp.38-45. DOI: 10.18653/v1/2020.nlp4convai-1.5.

[34] Kumar M, Kumar V, Glaude H, De Lichy C, Alok A, Gupta R. Protoda: Efficient transfer learning for few-shot intent classification. In *Proc. the 2021 IEEE Spoken Language Technology Workshop*, January 2021, pp.966-972. DOI: 10.1109/SLT48900.2021.9383495.

[35] Zhang J, Bui T, Yoon S, Chen X, Liu Z, Xia C, Tran Q H, Chang W, Yu P. Few-shot intent detection via contrastive pre-training and fine-tuning. In *Proc. the 2021 Conference on Empirical Methods in Natural Language Processing*, November 2021, pp.1906-1912. DOI: 10.18653/v1/2021.emnlp-main.144.

[36] Yuan P, Mobiny A, Jahanipour J, Li X, Cicalese P A, Roysam B, Patel V M, Dragan M, Nguyen H V. Few is enough: Task-augmented active meta-learning for brain cell classification. In *Proc. the 2020 Medical Image Computing and Computer Assisted Intervention*, October 2020, pp.367-377. DOI: 10.1007/978-3-030-59710-8_36.

[37] Wu L, Li J, Wang Y, Meng Q *et al.* R-drop: Regularized dropout for neural networks. In *Proc. the 2021 Annual Conference on Neural Information Processing Systems*, December 2021.

[38] Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. In *Proc. the 37th International Conference on Machine Learning*, July 2020, pp.1597-1607.

[39] Coucke A, Saade A, Ball A, Bluche T, Caulier A, Leroy D, Doumouro C, Gisselbrecht T, Caltagirone F, Lavril T, Primet M, Dureau J. Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces. arXiv:1805.10190, 2018. https://arxiv.org/abs/1805.10190, December 2021.

[40] Larson S, Mahendran A, Peper J J, Clarke C, Lee A, Hill P, Kummerfeld J K, Leach K, Laurenzano M A, Tang L, Mars J. An evaluation dataset for intent classification and out-of-scope prediction. In *Proc. the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, November 2019, pp.1311-1316. DOI: 10.18653/v1/D19-1131.

[41] Liu X, Eshghi A, Swietojanski P, Rieser V. Benchmarking natural language understanding services for building conversational agents. In *Proc. the 10th International Workshop on Spoken Dialogue Systems*, April 2019, pp.165-183. DOI: 10.1007/978-981-15-9323-9_15.

[42] Bao Y, Wu M, Chang S, Barzilay R. Few-shot text classification with distributional signatures. In *Proc. the 8th International Conference on Learning Representations*, April 2020.

[43] Li Y, Zhang J. Semi-supervised meta-learning for cross-domain few-shot intent classification. In *Proc. the 1st Workshop on Meta Learning and Its Applications to Natural Language Processing*, August 2021, pp.67-75. DOI: 10.18653/v1/2021.metanlp-1.8.

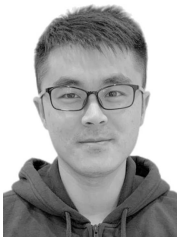
[44] Dopierre T, Gravier C, Logerais W. A neural few-shot text classification reality check. In *Proc. the 16th Conference of the European Chapter of the Association for Computational Linguistics*, April 2021, pp.935-943. DOI: 10.18653/v1/2021.eacl-main.79.

[45] Kingma D P, Ba J. Adam: A method for stochastic optimization. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[46] Khosla P, Teterwak P, Wang C, Sarna A, Tian Y, Isola P, Maschinot A, Liu C, Krishnan D. Supervised contrastive learning. In *Proc. the 2020 Annual Conference on Neural Information Processing Systems*, December 2020. pp.18661-18673.

[47] Ott M, Edunov S, Grangier D, Auli M. Scaling neural machine translation. In *Proc. the 3rd Conference on Machine Translation: Research Papers*, October 31-November 1, 2018, pp.1-9. DOI: 10.18653/v1/W18-6301.

[48] Ng N, Yee K, Baevski A, Ott M, Auli M, Edunov S. Facebook FAIR's WMT19 news translation task submission. In *Proc. the 4th Conference on Machine Translation*, August 2019, pp.314-319. DOI: 10.18653/v1/W19-5333.



Peng-Fei Sun received his M.E. degree in computer science from Chongqing University of Posts and Telecommunications, Chongqing, in 2013. Currently, he is working towards his Ph.D. degree with the National Key Laboratory for Novel Software Technology and Department of Computer Science and Technology at Nanjing University, Nanjing. His research interests lie primarily in natural language processing.

Ya-Wen Ouyang received his B.E. degree in software engineering from Dalian University of Technology, Dalian, in 2018. Currently, he is working towards his Ph.D. degree with the National Key Laboratory for Novel Software Technology and Department of Computer Science and Technology at Nanjing University, Nanjing. His research interests lie primarily in natural language processing.

Ding-Jie Song received his B.E. degree in software engineering from Nanjing University, Nanjing, in 2021. He is currently studying for his Master's degree with the National Key Laboratory for Novel Software Technology and Department of Computer Science and Technology at Nanjing University, Nanjing. His research interests include dialogue system and natural language processing.

Xin-Yu Dai received his B.E. and Ph.D. degrees in computer science from Nanjing University, Nanjing, in 1999 and 2005, respectively. He has been on leave from Nanjing University, Nanjing, from August 2010 to September 2011 to visit EECS Department and Statistics Department at UC Berkeley, Berkeley. He is currently a professor with the National Key Laboratory for Novel Software Technology and Department of Computer Science and Technology at Nanjing University, Nanjing. His research interests include natural language processing and knowledge engineering.