Abstract—Few-Shot classification aims at solving problems that only a few samples are available in the training process. Due to the lack of samples, researchers generally employ a set of training tasks from other domains to assist the target task, where the distribution between assistant tasks and the target task is usually different. To reduce the distribution gap, several lines of methods have been proposed, such as data augmentation and domain alignment. However, one common drawback of these algorithms is that they ignore the similarity task selection before training. The fundamental problem is to push the auxiliary tasks close to the target task. In this paper, we propose a novel task selecting algorithm, named Optimal Transport Task Selecting (OTTS), to construct a training set by selecting similar tasks for Few-Shot learning. Specifically, the OTTS measures the task similarity by calculating the optimal transport distance and completes the model training via a self-supervised strategy. By utilizing the selected tasks with OTTS, the training process of Few-Shot learning becomes more robust and effective. Other proposed methods including data augmentation and domain alignment can be used in the meantime with OTTS. We conduct extensive experiments on a variety of datasets, including MiniImageNet, CIFAR, CUB, Cars, and Places, to evaluate the effectiveness of OTTS. Experimental results validate that our OTTS outperforms the typical baselines, i.e., MAML, matchingnet, protonet, by a large margin (averagely 1.72% accuracy improvement).

Index Terms—Task selecting, Data mining, few-shot learning, self-supervised learning, optimal transport.
methods, a teacher model which is trained in other domains is used to design the curriculum. The Reinforcement learning teacher employs reinforcement learning to measure the difficulty of each sample and sort them as curriculum. The existing Transfer Teacher methods show the probability of using a teacher model to sorts the samples with the difficulty. Similar with the transfer teacher models, we can select tasks for Few-Shot learning if we can train a teacher model to measure the similarity between task.

In this paper, we propose a novel task selecting method to solve the Few-Shot task-selecting problem based on optimal transport, which is called Optimal Transport Task Selecting (OTTS). As a Few-Shot problem, the relationships between tasks are more likely contained in the samples instead of the unknowable distribution. And the relationship between samples could be another major difference except for the distribution of samples. Therefore, OTTS measures the similarity of tasks with the relationship between the features of the samples. In the process of OTTS, we build a graph for each task, where the features of the sample are represented as the nodes and the distance between of features are denoted as the edge. After building the graph, a distance based on optimal transport is designed to measure the similarity between tasks. The training of the feature encoder is based on self-supervised comparative learning with the optimal transport distance, which forces the similar tasks to gather while the unsimilar tasks scatter. Then, the OTTS can select the tasks before or during the training of classifier, which is similar to the teacher model of curriculum learning [13]–[16]. In this paper, we conduct a property experiment to verify the correctness of the training process and the distance of tasks. We also conduct an application experiment for the usage of OTTS on Cross-Domain Few-Shot learning. In the properties experiment, OTTS shows its superiority in selecting similar tasks not only from the same dataset but also from other datasets. By utilizing the tasks selected by OTTS, the training process become smoother with higher accuracy and lower loss. The result of the application experiment validates that the OTTS can benefit the Cross-Domain Few-Shot learning by selecting appropriate training tasks. The accuracy of models including protonet and matchingnet on cub, cars and places data sets has been improved by 1.72%.

The main contributions of this paper can be summarized as 4 points below:

- We propose a novel training strategy based on self-supervised learning algorithm called OTTS that can detect the similarity between Few-Shot tasks.
- We propose an application of graph optimal transport distance to measure the similarity of tasks. In the OTTS, this distance is used as loss function during training and selecting criterion.
- We conduct experiments to exam the correctness and verify the property of the OTTS algorithm. Find out the similarity of tasks included in the datasets including CUB, Cars, Places, CIFAR, and MiniImageNet in the view of the network.
- We propose a usage of OTTS for the Cross-Domain Few-Shot learning method, and verify the effect of the usage.

Fig. 1. (a) shows the optimal transport distance (or similarity) between 30000 training tasks with another 1 target task. All of the tasks are from the MiniImageNet dataset. It can be seen that the distribution of the distance is approximate to a Gaussian distribution. (b) is the distribution of 1000 tasks that are randomly selected from the 30000 tasks. The most randomly selected tasks have low similarity with the target task. (c) is the distance distribution between the target task and the tasks selected by our OTTS. The shorter distance means higher similarity, which can benefit the training of the classifier.

network is the most common strategy in recent years. For example, Task2Vec [8] firstly utilizes a probe network which is trained on the task requiring measurement. Then it applies the embedding of the probe network parameters as the feature of the task and measures the distance of tasks with the feature. The measuring method with distribution [10] utilizes Gaussian distribution to fit the distribution of the task samples. Then it adopts the optimal transpose [7], [11], [12] to compute the distance between distributions as the distance between tasks. Although the above mentioned two strategies can measure the distance with tasks, both kinds of methods have high requirements for the number of samples. The tasks with adequate samples have a lower necessity of selecting tasks to assist training. Especially, Few-Shot tasks, which urgently need assist training tasks, are unable to process with the existing methods.

In this case, we noticed Automatic Curriculum Learning (ACL) [13]–[16], which is a learning method that can select samples with the difficulty of samples. The ACL can be roughly divided into three categories, including Self-paced learning [17], [18], Transfer Teacher [19], [20], and Reinforcement learning teacher [21], [22]. With the Self-paced learning methods, model can automatically selecting samples with difficulty for further training. While in Transfer Teacher
In 1998, Rubner et al. proposed a probabilistic embedding method which uses optimal transport distance to compare the distributions. Frogner et al. [12] designed similar OT distance to measure the discrete measures. In 2019, Graph Optimal Transport (GOT) [12] utilized the GOT to alignment two graphs which belong to different domains. Following GOT [7], [12], we utilize GOT to measure the distance between two tasks.

2.2 Contrastive Self-Supervised Learning

Contrastive self-supervised learning [27], which belongs to unsupervised learning, completes the training process by utilizing the training samples information rather than the label information. This method is especially valuable in the situation that plenty of unlabeled samples are available in the training phase. Contrastive methods [28], [29] are commonly used to assemble the same image (“positive pairs”) together and simultaneously scatter the different ones (“negative pairs”).

A slow-moving average target network [29] designs a mixing process to update the parameter of the target network, which is used in many self-supervised learning models. This mixing process can produce stable output of the target network. The mixing process is commonly represented as

\[ \xi \leftarrow \tau \xi + (1 - \tau)\theta, \tau \in (0, 1) \]  

where \( \xi \) is the parameter of the target network which is slow-moving. The \( \theta \) is a set of parameter to update the target network.

Bootstrap Your Own Latent (BYOL) [29] is a typical method of contrastive learning. In BYOL, the feature \( Z_\theta \) is extracted from the training sample \( X \) by the online model \( F_\theta \) with parameter \( \theta \). The feature \( Z_\theta \) is used to predict the feature \( Z_\xi \) extracted by the target model \( F_\xi \). After obtaining the similarity between \( Z_\theta \) and \( Z_\xi \) as the contrastive loss, an optimizer is used to update the parameters of the online model and the parameter of the target model is updated with a mixing process.

\[ \theta \leftarrow \text{optimizer}(\theta, \nabla_\theta L_{\theta,\xi}^{BYOL}, \eta), \]  

\[ \xi \leftarrow \tau \xi + (1 - \tau)\theta \]  

where \( L_{\theta,\xi}^{BYOL} \) is the contrastive loss used in BYOL, \( \eta \) is the target decay rate.

In existed self-supervised learning methods, the samples normally used in the training process are the unlabeled images, sound signals, or sentences. Therefore a natural thought is that the tasks can be processed in a similar way if we can extract their effective feature. Following the same process as contrastive self-supervised learning, we...
can determine which tasks are similar like determining the similarity of samples as shown in Fig. 2.

2.3 Transfer Teacher Model
Transfer Teacher is a popular method in curriculum learning. In Transfer Teacher methods, a strong teacher model is pretrained to measure and sort the difficulty of samples with the performance of the teacher. In existed works, the teacher model could be a larger model than the student [19] or have the same structure with student model [20]. Generally, the teacher models are trained with a large enough dataset to obtain knowledge of label predicting and thus compute the loss of each sample. It can be recognized that the difficulty of the sample is based on the prediction of teacher and label.

With the optimal transport distance and self-supervised learning method mentioned above, we propose OTTS, a task selecting method trained with the self-supervised method, and optimal transport distance by regarding few-shot tasks as graphs. The main idea of the training process of OTTS is similar to self-supervised learning but on the plane of tasks. In OTTS, the positive pairs are tasks which include the same classes of samples. By training with the self-supervised method, OTTS can assemble similar graphs which represent the “features” of similar tasks, and scatter the unsimilar ones. The major difference lies that the teacher models measure the difficulty of samples, while OTTS measures the similarity between tasks.

2.4 Cross-Domain Few-Shot Learning
Cross-Domain Few-Shot (CDFS) Learning [30], [31] is a novel field with increasing popularity recent years. In most CDFS learning methods, researchers adopt the domain generation [6] or transfer learning [32] methods to force the classifier model to adapt to a new domain. In the experimental part of this paper, we use training tasks selected by OTTS to decrease the difficulty of adaptation and improve the performance of existing models.

3 Optimal Transport Task Selection
In this section, we introduce the details of OTTS method. We introduce the definition of the task selecting problem and the structure of OTTS in the subsection 3.1 and 3.2, respectively. Then we describe the training strategy of the feature extractor in the subsection 3.3. In subsection 3.4, we introduce the optimal transport loss function, and make a short prove for the relationship between the optimal transport distance and the loss of classifier.

3.1 Problem Definition
In a Few-Shot task selecting problem, the model need to select a set of support tasks \( \{T_{S_i}^{n}\}_{i} \) from tasks \( \{T_{S}\}_{i} \) in source domain \( D_S \) according to a testing set of target \( \{T_{T_i}\}_{i} \), where \( n_i \) is the selected task “ID” in the \( T_S \). Particularly, each support task is represented as \( T_{S} = \{(x_1^s, y_1^s), \ldots, (x_N^s, y_N^s), (x_{N+K}^s+1, \cdot), \ldots, (x_{N+K+M}^s, \cdot)\} \) and target task is represented as \( T_{T} = \{(x_1^t, y_1^t), \ldots, (x_N^t, y_N^t), (x_{N+K+1}^t, \cdot), \ldots, (x_{N+K+M}^t, \cdot)\} \).

In this case, each task contains K labeled samples for each of the N classes, which is commonly called N-way K-shot task. We assume the target tasks are from an unknowable target domain \( D_T \). The task-selecting model targets to select...
tasks from the source domain $D_S$ which are similar to the target task. By doing this, the OTTS model selects a set of training tasks in which samples are most possibly have the same features or distribution with the target tasks shown in 3.

### 3.2 Structure of OTTS

The structure of OTTS includes a feature extractor $F_\xi$ and a graph builder. The feature extractor is designed to extract the feature of each sample of a task. Then the graph builder can graph with the features. Each graph in this process represents the feature of a task, where the nodes donate the samples feature $Z$ and the edges represent the relationship between samples. The feature extractor is the trainable part of OTTS while the output of the graph builder only depends on the similarity between the features. In the training process, we also use an online model $F_\theta$ which has the same structure with the feature extractor to assist training.

### 3.3 Training Strategy of the Feature Extractor

During the training process, we use a self-supervised training strategy like BYOL but on the planes of tasks. As a task selecting problem, what we need during the training process is a huge number of tasks but not samples. To train OTTS, we prepare $N$-way 2-shots tasks and split each of the tasks into two $N$-way 1-shot tasks $T'$ and $T''$. The samples of $N$-way 1-shot tasks are fed as the input for the online and target models $F_\theta$, $F_\xi$ separately as a pair of positive samples.

The features $\{Z\}_{i=1}^N$ which extracted from each of these samples are then constructed into graphs with the graph builder. In this step, the relationship between samples of the tasks is computed and can be used in the calculation of the loss function. For each $N$-way 1-shot task, we build a graph $G$ with $N$ nodes to represent the “feature of the task”. We consider each pair of the features of the tasks as a pair of different representations of the same tasks. Intuitively, the OTTS model can find the same information between the representation and ignore the differences.

Therefore, in the training process of OTTS, we build an online model $F_\theta$ which is the same as the feature extractor to assist training and the feature extractor is seen as the target model $F_\xi$. We employ the graph $G_\theta$ of task feature $Z_\theta$ which is extracted by the online model $F_\theta$ to predict the graph $G_\xi$ of feature $Z_\xi$ extracted by the target model $F_\xi$. Then exchange the order of the tasks to obtain another pair of graphs $G_\theta$ and $G_\xi$.

$$L_{OTTS} = L_{OT}(G_\theta, G_\xi) + L_{OT}(G_\theta', G_\xi')$$

(6)

And updating the model with the with optimal transport loss function below

$$\theta \leftarrow \text{optimizer}(\theta, \nabla_\theta L_{OTTS}, \eta),$$

$$\xi \leftarrow \tau \xi + (1 - \tau)\theta$$

(7)

(8)

where the $L_{OT}(G_\theta, G_\xi)$ is the optimal transport loss function. The training strategy is shown in Fig. 4.

### 3.4 Optimal Transport Loss Function

The optimal transport loss function used in this paper is the summation of Wasserstein distance and the Gromov-Wasserstein distance. As mentioned above, each task is embedded as a graph, which can be viewed as discrete measures in the graph space. Therefore, the distance between two graphs can be measured by Wasserstein distance:

$$L_W(G_\theta, G_\xi) = \min_{T(G_\theta, G_\xi) \in T} \sum_{i=1}^N \sum_{j=1}^M c_{ij} T_{ij}$$

$$= \min_{T(G_\theta, G_\xi) \in T, T_{ij} \sim T} \sum_{i=1}^N \sum_{j=1}^M \|Z_\theta^i - Z_\xi^j\|$$

(9)

where $c_{ij} = \|Z_\theta^i - Z_\xi^j\|$ is the cost of transport from the feature $Z_\theta^i$ in graph $G_\theta$ and $Z_\xi^j$ in graph $G_\xi$. The Wasserstein distance can compute the similarity between the node of two graphs, which represent the feature of samples. During the training process of the classifier, the classifier loss of each task should be the same if the tasks are similar. The reason is that similar tasks theoretically share the same classifier parameters. Thus we can prove the relationship between the optimal transport distance and the similarity of tasks for classifier:

**Th. 3.1.** Given two graphs $G_\theta, G_\xi$ of sample feature from different tasks. $Z_\theta \in G_\theta$ and $Z_\xi \in G_\xi$ are the features in the graph. For $\forall \epsilon \in \mathbb{R}^+$, $\exists \delta$ such that for all $G_\theta, G_\xi$,

$$L_W(G_\theta, G_\xi) = \min_{T(G_\theta, G_\xi) \in T} \sum_{i=1}^N \sum_{j=1}^M c_{ij} T_{ij} < \delta$$

(10)

implies that

$$|p(y = Y|G_\theta, w) - p(y = Y|G_\xi, w)| < \epsilon$$

(11)

where $w$ is the parameter of the model.

**Proof:** Given $c_{ij}$ is the cost that transport from $Z_\theta^i$ to $Z_\xi^j$. For $T(G_\theta, G_\xi) \in T$ that minimize the loss.

$$|p(y = Y|G_\theta, w) - p(y = Y|G_\xi, w)|$$

$$=|\prod_i p(y^i|Z_\theta^i, w) - \prod_{T_{ij} \sim T(G_\theta, G_\xi)} p(y^i|Z_\xi^j, w)|$$

$$=|\prod_i p(y^i|Z_\theta^i, w) - \prod_{T_{ij} \sim T(G_\theta, G_\xi)} p(y^i|Z_\theta^i, w)p(Z_\xi^j|Z_\theta^i)|$$

$$=|1 - \prod_i p(Z_\xi^j|Z_\theta^i)p(y^i|Z_\theta^i, w)|$$

(12)

The distribution of $Z_\xi^j$ with noise is a Gaussian distribution [33] with $\mu = Z_\xi^j$:

$$p_{Z_\xi^j}(x) = N(Z_\xi^j, \Sigma^j)$$

(13)

The possibility for $x = Z_\theta^i$ is

$$p(Z_\theta^i|Z_\xi^j) = p_{Z_\xi^j}(Z_\theta^i)$$

$$= (2\pi)^{-\frac{d}{2}} |\Sigma^j|^{-1/2} \exp(-\frac{1}{2}(Z_\theta^i - Z_\xi^j)^T \Sigma^j^{-1}(Z_\theta^i - Z_\xi^j))$$

(14)
Set $\lambda_{\text{max}}$ is the max value of the characteristic value of the positive definite matrix $\Sigma^{-1}$, for $c(Z_{\theta}^{i}, Z_{\xi}^{j}) = (Z_{\theta}^{i} - Z_{\xi}^{j})^T (Z_{\theta}^{i} - Z_{\xi}^{j})^{1/2}$

$$
(2\pi)^{-\frac{D}{2}} |\Sigma|^{-(1/2)} \exp(-\frac{1}{2} (Z_{\theta}^{i} - Z_{\xi}^{j})^T (Z_{\theta}^{i} - Z_{\xi}^{j})^{-1} (Z_{\theta}^{i} - Z_{\xi}^{j}))
$$

$$
> (2\pi)^{-\frac{D}{2}} |\Sigma|^{-(1/2)} \exp(-\frac{1}{2} \lambda_{\text{max}}^{T} (Z_{\theta}^{i} - Z_{\xi}^{j})^T (Z_{\theta}^{i} - Z_{\xi}^{j}))
$$

$$
= (2\pi)^{-\frac{D}{2}} |\Sigma|^{-(1/2)} \exp(-\frac{1}{2} \lambda_{\text{max}}^{T} c(Z_{\theta}^{i}, Z_{\xi}^{j})^2)
$$

Substitute (16) into (13), $\lambda_{\text{max}} = \max_{j} \lambda_{\text{max}}^{j}$.

$$
|1 - \prod_{T_{ij}} (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2} \lambda_{\text{max}}^{T} c(Z_{\theta}^{i}, Z_{\xi}^{j})^2} \prod_{i} p(y^{i}|Z_{\theta}^{i}, w)|
$$

$$
< |1 - K e^{-\frac{1}{2} \lambda_{\text{max}}^{T} \Sigma_{T_{ij}} c(Z_{\theta}^{i}, Z_{\xi}^{j})^2} \prod_{i} p(y^{i}|Z_{\theta}^{i}, w)|
$$

$$
< |1 - K e^{-\frac{1}{2} \lambda_{\text{max}}^{T} \Sigma_{T_{ij}} c(Z_{\theta}^{i}, Z_{\xi}^{j})^2} \prod_{i} p(y^{i}|Z_{\theta}^{i}, w)|
$$

$$
< |1 - K e^{-\frac{1}{2} \lambda_{\text{max}}^{T} \Sigma (\delta)^2} \prod_{i} p(y^{i}|Z_{\theta}^{i}, w)| = \epsilon
$$

where $K = ((2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}})^{T}$. For every $\epsilon$, exist $\delta$ such that $\forall G_{\theta}, G_{\xi}$, $\sum_{T_{ij}} c_{ij} = \sum_{T_{ij}} c(Z_{\theta}^{i}, Z_{\xi}^{j}) < \delta$, the $|p(y = Y|G_{\theta}, w) - p(y = Y|G_{\xi}, w)| < \epsilon$.

In this theory, we proved that if the distance between two tasks is small, the losses of the classifier with the same parameters for each task are also similar. The similar losses on each point would lead to the same solution of classifier parameters. Therefore, training with the tasks which cause similar losses can bring a stable and effective training process for the classifiers.

However, the Wasserstein distance can not express all the information contained in the graph. It can only measure the similarity between features. We hence employ the Gromov-Wasserstein distance to represent the relationship of nodes, which implies the information of the relationship between samples from different classes. Wasserstein distance can help the OTTS model confirm the position of each sample in the task, further correcting the similarity between tasks. The correcting effect has a positive relationship with the number of samples used in the graph.

$$
\mathcal{L}_{GW}(G_{\theta}, G_{\xi}) = \min_{\gamma} \sum_{i,j} \gamma_{ij} \mathcal{L}(G_{\theta}^{i}, G_{\xi}^{j})
$$

$$
\mathcal{L}_{GW}(G_{\theta}, G_{\xi}) = \min_{\gamma} \sum_{i,j} \gamma_{ij} \mathcal{L}(G_{\theta}^{i}, G_{\xi}^{j})
$$

Therefore, the final loss is the sum of WD loss and GWD loss:

$$
\mathcal{L}_{GW}(G_{\theta}, G_{\xi}) = r * \mathcal{L}_{W} + (1 - r) \mathcal{L}_{GW}
$$

where $r \in (0, 1)$. With this optimal transport loss function, the $\mathcal{L}_{GW}$ is more like a regularization term to further limit the structure of the graphs. By training with the loss function with the self-supervised learning method. The tasks which have similar features can be assembled. After the training process, the target model in self-supervised learning and a graph builder are able to extract the task features. The similarity between tasks is calculated with the same similarity function as the training loss. After computing the similarities between the target task and the source tasks, OTTS can select tasks from source tasks as needed.

The details of the training and application process of OTTS is shown in the Algorithm 1.

### 4 Experiment

In this section, we run experiments on several datasets which are commonly used in the Few-Shot and Cross-Domain method [30, 31, 34, 35] to verify the performance of OTTS on Few-Shot and Cross-Domain Few-Shot learning.
The samples in each class is divided into 500 training ages. Each of the classes contains 600 samples of size 32*32 images and testing classes with 8041 images. The dataset is divided into a training set with 8144 which are consist of 16185 images with different kinds of birds images with totally 11788 samples. These samples are the proportion of 64:16:20.

MiniImageNet: The MiniImageNet [36] dataset is a subset of the ImageNet dataset. Specifically, the MiniImageNet contains totally 60000 images, belonging to 100 classes, which have 600 samples per class. The size of each image is 84*84. For fair comparison, we randomly divided the MiniImageNet into training, validation, and testing set with the proportion of 64:16:20.

CIFAR100: The CIFAR [37] includes 100 classes of images. Each of the classes contains 600 samples of size 32*32. The samples in each class is divided into 500 training samples and 100 testing samples.

CUB: The CUB [38] dataset is consist of 200 classes of birds images with totally 11788 samples. These samples are divided into a subset with 5994 samples for training and the rest 5794 samples for testing.

Cars: The Cars [39] dataset contains 196 classes of images which are consist of 16185 images with different kinds of cars. The dataset is divided into a training set with 8144 images and testing classes with 8041 images.

Algorithm 1: The proposed OTTS algorithm

| Input: A set of N-way 2-shot tasks \( T \), Online model \( F_\theta \) and target model \( F_\xi \). A set of target tasks \( T'_S \) and a large set \( T'_S \), which includes a lots of tasks |
| Output: A set of training tasks \( \{T'_S\}_i \) |

1 /*The training process of OTTS */
2 for A batch of \( T \in \{ T \} \) do
3     Divide each N-way 2-shot tasks \( T \) into 2 N-way 1-shot tasks \( T'^0 \), \( T''^0 \);
4     Extract the features \( \{Z_0\} \) of samples in \( T' \) by \( F_\theta \) and the feature \( \{Z'_0\} \) in \( T'' \) by \( F_\xi \);
5     Build graph \( G_\theta \) with \( \{Z_0\} \) and \( G_\xi \) with \( \{Z'_0\} \);
6     Exchange \( T' \) and \( T'' \) and build graphs \( G'_\theta \) and \( G'_\xi \) with the same process above;
7     Compute the optimal transport loss \( \mathcal{L}_{\text{OTTS}} = \mathcal{L}_{\text{OT}}(G_\theta, G_\xi) + \mathcal{L}_{\text{OT}}(G'_\theta, G'_\xi) \);
8     Update the parameter of online model \( F_\theta \) with \( \theta \leftarrow \text{optimizer}(\theta, \nabla_\theta \mathcal{L}_{\text{OTTS}}(G_\theta, G_\xi, \eta)) \);
9     Update the parameter of target model \( F_\xi \) with \( \xi \leftarrow \xi + (1 - \tau) \theta \);
10    Gain a set of parameter \( \xi \) for the target model;
11   /*The application of OTTS */
12   Extract the features \( \{Z'_n\}_i \) of \( T_S \in \{ T'_S \} \), with \( F_\xi \) and build a graph \( G'_n \) for each feature;
13   Compute a set of optimal transport distance \( \{\mathcal{L}_{\text{OT}}(G_i, G'_n)\}_i \) between \( G_i \) and \( G'_n \in \{G'_n\}_i \);
14   Sort the \( \{\mathcal{L}_{\text{OT}}(G_i, G'_n)\}_i \) in an increased order \( \{\mathcal{L}_{\text{OT}}(G_i, G'_n)\}_i \);
15   Select the top \( M \) elements, i.e. \( \{\mathcal{L}_{\text{OT}}(G_i, G'_n)\}_i^{M} \) and the corresponding tasks \( \{T'_S\}_i^{M} \);
16   return Top M elements of the sorted list \( \{T'_S\}_i^{M} \).

4.1 Dataset Description

In this section, we introduced the datasets used in the experiments. We use datasets which are commonly used in few-shot learning and transfer learning to verify the effects of OTTS. The details of the datasets are shown below:

MiniImageNet: The MiniImageNet [36] dataset is a subset of the ImageNet dataset. Specifically, the MiniImageNet contains totally 60000 images, belonging to 100 classes, which have 600 samples per class. The size of each image is 84*84. For fair comparison, we randomly divided the MiniImageNet into training, validation, and testing set with the proportion of 64:16:20.

CIFAR100: The CIFAR [37] includes 100 classes of images. Each of the classes contains 600 samples of size 32*32. The samples in each class is divided into 500 training samples and 100 testing samples.

CUB: The CUB [38] dataset is consist of 200 classes of birds images with totally 11788 samples. These samples are divided into a subset with 5994 samples for training and the rest 5794 samples for testing.

Cars: The Cars [39] dataset contains 196 classes of images which are consist of 16185 images with different kinds of cars. The dataset is divided into a training set with 8144 images and testing classes with 8041 images.

Places: The Places [40] dataset comprises over 10 million images. The images are categorized into more than 400 unique scene classes. Each class includes 5000 to 30000 training images.

The size of the images in CIFAR100, CUB, Cars, and Places are resized to 84*84 to suit the size of images in MiniImageNet.

4.2 Baseline Description

We compare our method with typical baselines, including both metric-based Few-Shot models, i.e., matchingnet [41], protonet [2] and optimization-based model, i.e., MAML [42].

Matchingnet: Matchingnet adopts pair-wise comparisons between the samples in the support set and the new input testing set. The predictions of the Matchingnet are based on the weighted similarity of samples in the support set and testing set.

Protonet: Protonet predicts the label of the testing samples by calculating the distance between the feature of the testing sample and the prototype of each class. The prototype of each class is obtained by all the samples belonging to the class.

MAML: MAML is a Model-Agnostic optimization-based model that uses inner and outer optimization procedure to obtain the second-order gradient. The key idea of MAML is to obtain the ability to fast optimize to new tasks. This ability is tightly related to the similarity between the training and testing tasks.

4.3 Details of OTTS

The training process of OTTS has been introduced above. Resnet-50 [43] is used as the online and target model in the self-supervised training process. In the training process, we used 25600 5-way 2-shot tasks from the training set of MiniImageNet. In each epoch, we input a batch of tasks into the self-supervised model and compute the optimal transport loss of the batch. The size of each batch of tasks is 64. The optimizer used in this process is Adam [44].

After training the OTTS model, we randomly select 30000 tasks from the training set of MiniImageNet as the training set. In the experiments below, these tasks are used if we need any tasks from the training set of MiniImageNet.

4.4 Implementation Details

In the experiments, we utilize both optimization-based and metric-based few-shot models include MAML [42], Matchingnet [41], and Protonet [2] as the baselines. For MAML, we utilize a four-layer CNN (Conv4) as the backbone network. We set the learning rate \( \alpha = 0.02 \) while the meta-learning rate \( \beta = 0.002 \) of MAML. The model is trained for 10 epochs with 400 training tasks from the training set. After the training process of each epoch, we utilize 10 set of testing tasks from MiniImageNet or CIFAR. Each set includes 5 tasks from the testing set to evaluate the performance of the model. The other hyperparameters are the same as the setting in the paper of MAML. To compare the effect of OTTS, we utilize selected training tasks that are similar to testing tasks while the control group uses the random training tasks.
4.5.1 Property experiment

This experiment aims to verify the property of the OTTS method. In this experiment, we randomly select 30 tasks from three datasets in CUB, Cars, and Places, measuring the similarity between every two tasks after a data augmentation for each sample. The final results are shown in Table 1 and Fig. 5.

Table 1 shows the average, maximum and minimum distance in each group of tasks. Each number in the table is computed 10 times, and the mean results are reported. The tasks from the same domain don’t have essential differences with the tasks from the other domains. The Max and Min values show that the similarity of tasks is not totally related with domain. The tasks which are most similar may not be consist in the same domain. The Average distance and the Max distance also show that the cross-domain tasks are more likely to be unsimilar to each other. For example, the tasks in CUB domain have an average similarity of 62.05, while similarity of tasks in CUB and other domains are larger. These reasons provide the probability to select training tasks from the other domains.

Fig. 5 also explains similar property as Table 1. Tasks from the same domain are sometimes less similar than cross-domain tasks. Even worse, we also observe some tasks, which show higher self divergences than either intra-domain or cross-domain tasks. These tasks have lower accuracy for a classifier that trained with the random training set because the feature or distribution of these tasks is different from the training set. This means that if we randomly select training tasks, the distance between the training tasks and the target tasks is expected to be larger than the minimum distance. As we introduced above, some training tasks are unsimilar to each other, the training process may not converge to a fixed optimal result. This may cause the parameters of the model to fluctuate largely. Therefore, random tasks selection may cause high uncertainty during training.

By incorporating the OTTS method, the distance between training and testing tasks can be minimized. Therefore, the optimized training tasks are more likely to be similar with the testing tasks, and guarantee a stable training process. The detail is shown in the next section.

For the Matchingnet and Protonet, we also utilize a Conv4 as the backbone network. The training strategy of Matchingnet and Protonet is Feature-wise Transformation (FT) [45]. Particularly, we train each model via 8 epochs while each epoch includes 10 couples of training tasks. The other settings are the same as the settings in FT [45]. After training the model, we utilize a set of testing tasks from the testing set of the target domain (including CUB, Cars, and Places) to test the result of the training. We utilize the testing tasks and similar training tasks selected by OTTS and the random testing and training tasks to make comparisons.

We adopt two quantitative measurements, i.e., classification accuracy and model loss, to evaluate the classification performance.

For the Matchingnet and Protonet, we also utilize a Conv4 as the backbone network. The training strategy of Matchingnet and Protonet is Feature-wise Transformation (FT) [45]. Particularly, we train each model via 8 epochs while each epoch includes 10 couples of training tasks. The other settings are the same as the settings in FT [45]. After training the model, we utilize a set of testing tasks from the testing set of the target domain (including CUB, Cars, and Places) to test the result of the training. We utilize the testing tasks and similar training tasks selected by OTTS and the random testing and training tasks to make comparisons.

We adopt two quantitative measurements, i.e., classification accuracy and model loss, to evaluate the classification performance.

4.5 Experimental Results

In this section, we compare the OTTS enhanced baseline model with the basic baseline model to verify the property of the OTTS. Then we propose the application of the OTTS on the cross-domain few-shot classification learning.

### Table 1: Distance between datasets.

| Dataset     | Average | Max    | Min    |
|-------------|---------|--------|--------|
| CUB         | 62.05   | 133.57 | 19.86  |
| Cars        | 64.32   | 123.07 | 18.06  |
| Places      | 66.01   | 137.11 | 19.74  |
| CUB vs Cars | 68.37   | 137.17 | 21.09  |
| Cars vs Places | 65.23   | 138.46 | 23.79  |
| Places vs CUB | 69.29   | 141.12 | 23.95  |
| CUB vs MiniImageNet | 64.05 | 136.13 | 20.74  |
| Cars vs MiniImageNet | 68.01 | 130.11 | 25.18  |
| Places vs MiniImageNet | 67.70 | 143.37 | 24.36  |

Fig. 5. Similarity between tasks cross domain: This figure shows the similarity between different cross domains. The darker the color, the more similar the tasks are. Each domain contains 30 tasks, while the diagonal shows the similarity with itself.
In the Table 2, we can find out that the selected tasks can achieve lower loss value in most cases. For the total testing set which contains all 10 sets, our OTTS outperforms the random selection strategy in both metrics, i.e., accuracy and the loss. Because the testing sets from CIFAR have larger differences with training tasks, the improvements of both accuracy and loss value are also larger. This proved that OTTS can benefit training process even though the tasks are from different domains, and the model is not specially designed for cross domain problems. The selected tasks can achieve higher results especially since the domains are different are whole but contains similar tasks.

Fig. 6 reflects the change of loss during the training process of MAML. Owing to the OTTS, the loss attenuation is faster than using the random tasks. In the meantime, selected tasks can make the training process more stable which can be seen as the shadow area of the Fig. 6. It can be noticed that the blue shadow area shrinks fast during the training process while the orange shadow doesn’t follow similar law. This represents that the selected tasks can cause a smoother learning process than random tasks.

These results validate the effectiveness of OTTS with MAML. It can be seen that OTTS can improve the stability of the training process. The multiple examinations validate the stability of OTTS. Selecting multiple training tasks can further reduce the randomness of the task selected by OTTS. During the training process, the training loss converged more stable, and the final loss is lower in most cases by using OTTS selected tasks. In the next section, we will discuss the effect of OTTS on the metric-based model on Cross-domain problems [34], [35].

4.5.3 Cross-Domain Few-Shot experiment
The Cross-Domain Few-Shot experiment selects the tasks from MiniImageNet as training set, while the target tasks are from CUB, Cars, or Places. In this experiment, similar tasks

In this experiment, we compare how similar tasks selected by OTTS influence the training process of MAML. Particularly, we compare the training processes by employing the randomly selected tasks and the OTTS selected tasks. The results of this experiment are shown in Table 2 and Fig. 6.

Table 2 represents the accuracy and the loss with different testing sets from MiniImageNet of CIFAR. Each testing set includes 5 testing tasks that are similar to each other. The training set includes 400 random tasks or selected tasks which are similar to the testing sets. Besides, Table 2 shows that the OTTS can benefit the training process of MAML.

4.5.2 How similar tasks influence the training process
In this experiment, we compare how similar tasks selected by OTTS influence the training process of MAML. Particularly, we compare the training processes by employing the randomly selected tasks and the OTTS selected tasks. The results of this experiment are shown in Table 2 and Fig. 6.
or random tasks are utilized to show the effect of OTTS in Cross-Domain learning. The models used in this experiment are Protonet and Matchingnet. The training strategy is FT. The final results are reported in Table 3, Fig. 7 and Table 4.

In Table 3, the accuracy of each model with OTTS selected training set on Cars, Cub, and Places are higher than the model with a random training set. This elaborates that OTTS can select a training set from the source domain to assist the training for the target domain. By using OTTS, the accuracy of the models increase by 1.72% in average.

Fig. 7 shows the distance relationship between the source support (Ss) tasks and target testing (Tt) tasks. The "✓" in the table represents the distance relationship is limited otherwise with no limits. The source support tasks are selected according to the target support (Ts) tasks. By selecting the source support tasks, the distance between the target testing tasks and the source support tasks can be partly minimized. Additionally, the OTTS selected training set preserves a smaller distribution distance than the random selection strategy. Therefore, the accuracy is higher with the tasks selected by OTTS. It is worth mentioning that the OTTS selected tasks can also deliver more stable distance and accuracy during training.

Furthermore, we investigate the relationship between task similarity and classification accuracy in Table 4. "Ss-Ts" means that we limit the distance between the source support (Ss) tasks and the target support (Ts) tasks, while the "Ss-Tt" is source support (Ss) tasks with target testing (Tt) tasks. It can be recognized that the similarity between the source support (Ss) tasks and target testing (Tt) tasks can largely improve the accuracy of the model. But the similarity between source support (Ss) tasks with target support (Ts) tasks is not significant and would cause a negative impact when we limit both similarities.

In the cross-domain experiments, we prove the effectiveness of OTTS in cross-domain learning. By using OTTS, the tasks from the source domain can be selected to construct a smaller training set which is more efficient for training. The large average distance between sets can be reduced the accuracy of target testing sets. The usage in cross-domain learning area is also a strength of OTTS than the transfer teacher methods, because OTTS can directly optimize the distribution of the training samples.

5 Conclusion

Few-Shot learning problems have received wide attention in recent years aiming to solve a problem that not enough samples can be used in the learning process. Therefore, the Few-Shot problem require extensive samples from other tasks to train the model. However, the samples from other tasks usually have different distribution with the task we wanted to train. Although there are multiple methods to minimize the differences of the tasks, like data augmentation and domain alignment, these methods require either the distribution or the amount of the of samples. To solve these problems, we proposed a novel method, i.e., OTTS, to minimize the differences by selecting similar tasks from the known datasets. In this method, we employed the self-supervised training method and graph optimal transport

### TABLE 3
The accuracy comparison between random strategy and our OTTS algorithm on metric-based models.

| model    | Cars      |          | CUB      |          | Places   |          |
|----------|-----------|----------|----------|----------|----------|----------|
|          | random    | OTTS     | random   | OTTS     | random   | OTTS     |
| matchingnet | 24.74% + 0.33% | 27.70% + 0.42% | 26.91% + 0.39% | 31.31% + 0.41% | 30.74% + 0.47% | 30.11% + 0.45% |
| protonet  | 25.04% + 0.37% | 26.00% + 0.37% | 27.36% + 0.37% | 28.23% + 0.36% | 27.79% + 0.40% | 29.56% + 0.47% |

### TABLE 4
The accuracy of different similarity settings on Cars.

| Ss-Ts | Ss-Tt | Accuracy    |
|-------|-------|-------------|
| ✓     | ✓     | 25.22% + 0.35% |
| ✓     | ✓     | 25.77% + 0.33% |
| ✓     | ✓     | 27.70% + 0.42% |
| ✓     | ✓     | 24.71% + 0.33% |
| ✓     | ✓     | 25.48% + 0.36% |
| ✓     | ✓     | 26.00% + 0.37% |
distance to measure the similarity between tasks. By using OTTS in multiple experiments, we concluded the properties of this method and proposed the application of the OTTS in Cross-Domain Few-Shot classification problems. These experiments validated the effectiveness of OTTS and achieved better results than the baseline methods. Furthermore, OTTS algorithm can not only select the tasks which are most similar to the target tasks but also be employed in a training strategy that use the tasks with different similarities efficiently. We are also looking forward to the further usages of the task selecting methods, e.g., the training process of OTTS can be carried out simultaneously with the training process of classifier.

REFERENCES

[1] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” ACM Computing Surveys (CSUR), vol. 53, no. 5, pp. 1–34, 2020.
[2] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” arXiv preprint arXiv:1703.05175, 2017.
[3] J. Li, B. Chiu, S. Feng, and H. Wang, “Few-shot named entity recognition via meta-learning,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2020.
[4] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, pp. 13001–13008.
[5] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” Journal of Big Data, vol. 6, no. 1, pp. 1–48, 2019.
[6] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, “Domain generalization: A survey,” arXiv preprint arXiv:2103.02503, 2021.
[7] L. Chen, Z. Gan, Y. Cheng, L. Li, L. Carin, and J. Liu, “Graph optimal transport for cross-domain alignment,” in International Conference on Machine Learning. PMLR, 2020, pp. 1542–1553.
[8] A. Achille, M. Lam, R. Tewari, A. Ravichandran, S. Maji, C. C. Fowlkes, S. Soatto, and P. Perona, “Task2vec: Task embedding for meta-learning,” in Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition, 2021, pp. 1245–1254.
[9] D. Alvarez-Melis and N. Pusi, “Geometric dataset distances via optimal transport,” arXiv preprint arXiv:2002.02923, 2020.
[10] G. Peyré, M. Cuturi et al., “Computational optimal transport: With applications to data science,” Foundations and Trends® in Machine Learning, vol. 11, no. 5–6, pp. 355–607, 2019.
[11] V. Titouan, N. Courty, R. Tavenard, and R. Flamary, “Optimal transport for structured data with application on graphs,” in International Conference on Machine Learning. PMLR, 2019, pp. 6275–6284.
[12] W. Huang, J. Liu, T. Li, T. Huang, S. Ji, and J. Wen, “Feddsr: Daily schedule recommendation in a federated deep reinforcement learning framework,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2021.
[13] W. Wang, G. Xu, W. Ding, Y. Huang, G. Li, J. Tang, and Z. Liu, “Representation learning from limited educational data with crowdsourced labels,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2020.
[14] B. Han, I. W. Tsang, X. Xiao, L. Chen, S.-F. Fung, and C. P. Yu, “Privacy-preserving stochastic gradual learning,” IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 8, pp. 3129–3140, 2021.
[15] X. Wang, Y. Chen, and W. Zhu, “A survey on curriculum learning,” IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2021.
[16] X. Guo, X. Liu, E. Zhu, X. Zhu, M. Li, X. Xu, and J. Yin, “Adaptive self-paced deep clustering with data augmentation,” IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 9, pp. 1680–1693, 2019.
[17] Y. Tang, Y. Xie, X. Yang, J. Niu, and W. Zhang, “Tensor multi-elastic kernel self-paced learning for time series clustering,” IEEE Transactions on Knowledge and Data Engineering, 2019.
[18] D. Weinshall, G. Cohen, and D. Amir, “Curriculum learning by transfer learning: Theory and experiments with deep networks,” in Proceedings of the 35th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 2535–2544. [Online]. Available: https://proceedings.mlr.press/v97/weinshall19a.html
[19] G. Hacohen and D. Weinshall, “On the power of curriculum learning in training deep networks,” in Proceedings of the 36th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 2535–2544. [Online]. Available: https://proceedings.mlr.press/v97/hacohen19a.html
[20] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, “Automated curriculum learning for neural networks,” in international conference on machine learning. PMLR, 2017, pp. 1311–1320.
[21] T. Mattissen, A. Oliver, T. Cohen, and J. Schulman, “Teacher-student curriculum learning,” IEEE transactions on neural networks and learning systems, vol. 31, no. 9, pp. 3732–3740, 2019.
[22] Y. Rubner, C. Tomasi, and L. J. Guibas, “A metric for distributions with applications to image databases,” in Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271). IEEE, 1998, pp. 59–66.
[23] G. Peyré, M. Cuturi, and J. Solomon, “Gromov-wasserstein averaging of kernels and distance matrices,” in International Conference on Machine Learning. PMLR, 2016, pp. 2134–2142.
[24] B. Muzellec and M. Cuturi, “Generalizing point embeddings using the wasserstein space of elliptical distributions,” in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 10258–10269.
[25] C. Frogner, F. Misra, and J. Solomon, “Learning embeddings into entropic wasserstein spaces,” in International Conference on Learning Representations, 2018.
[26] L. Jing and Y. Tian, “Self-supervised visual feature learning with deep neural networks: A survey,” IEEE transactions on pattern analysis and machine intelligence, 2020.
[27] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[28] J. H. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar et al., “Bootstrap your own latent: A new approach to self-supervised learning,” arXiv preprint arXiv:2006.07733, 2020.
[29] Y. Guo, N. C. Codella, L. Karlinsky, J. V. Codella, J. R. Smith, K. Saenko, T. Rosing, and R. Feris, “A broader study of cross-domain few-shot learning,” in European Conference on Computer Vision. Springer, 2020, pp. 124–141.
[30] N. Adler, R. Brandstetter, M. Widrich, A. Mayr, D. Kreil, M. Kopp, G. Klambauer, and S. Hochreiter, “Cross-domain few-shot learning by representation fusion,” arXiv e-prints, pp. arXiv–2010, 2020.
[31] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1399, 2010.
[32] C. P. Yu, S. Xie, and J. Fei-Fei, “3d object representations for fine-grained categorization,” in Proceedings of the IEEE international conference on computer vision workshops, 2013, pp. 554–561.
[40] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 6, pp. 1452–1464, 2017.

[41] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra et al., “Matching networks for one shot learning,” Advances in neural information processing systems, vol. 29, pp. 3630–3638, 2016.

[42] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in International Conference on Machine Learning. PMLR, 2017, pp. 1126–1135.

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

[44] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv e-prints, pp. arXiv–1412, 2014.

[45] H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang, “Cross-domain few-shot classification via learned feature-wise transformation,” in International Conference on Learning Representations, 2020.

Renjie Xu is a master of engineering student in the College of Control Science and Engineering, China University of Petroleum (East China), China. His research interests include meta-learning, Few-Shot learning, and Cross-Domain learning.

Xinghao Yang received the B.Eng. degree in electronic information engineering and M.Eng. degree in information and communication engineering from the China University of Petroleum (East China), Qingdao, China, in 2015 and 2018, respectively. His research interests include multi-view learning and adversarial machine learning with publications on IEEE TKDE, AAAI, IJCAI, Information Fusion and Information sciences.

Baodi Liu received the Ph.D. degree in Electronic Engineering from Tsinghua University. Currently, he is an assistant professor in College of Information and Control Engineering, China University of Petroleum, China. His research interests include computer vision and machine learning.

Kai Zhang received the Ph.D. degree in petroleum engineering from the China University of Petroleum (East China), Qingdao, China, in 2008. From June 2007 to May 2008, he studied with the University of Tulsa, Tulsa, OK, USA. He has been a Teacher with the China University of Petroleum (East China) since 2008. He teaches courses, including fluid flow in porous media and reservoir engineering. As a Project Leader, he has been in charge of three projects supported by the Natural Science Foundation of China, one project supported by the National Natural Science Foundation of Shandong Province, and 20 projects supported by SINOPEC, CNOOC, and CNPC. He has already published more than 60 papers. His research focuses on reservoir simulation, production optimization, history matching, and development of nonconventional reservoir.

Weifeng Liu (M’12-SM’17) is currently a Professor with the College of Control Science and Engineering, China University of Petroleum (East China), China. He received the double B.S. degree in automation and business administration and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2002 and 2007, respectively. His current research interests include pattern recognition and machine learning. He has authored or co-authored a dozen papers in top journals and prestigious conferences including 10 ESI Highly Cited Papers and 3 ESI Hot Papers. Dr. Weifeng Liu serves as associate editor for Neural Processing Letter, co-chair for IEEE SMC technical committee on cognitive computing, and guest editor of special issue for Signal Processing, IET Computer Vision, Neurocomputing, and Remote Sensing. He also serves dozens of journals and conferences.