Constructing Multiple Tasks for Augmentation: Improving Neural Image Classification With K-means Features

Tao Gui*, **Lizhi Qing***, **Qi Zhang, Jiacheng Ye, HangYan, Zichu Fei, Xuanjing Huang**
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{tgui16, lzqing18, qz, yejc19, hyan19, zcfei19, xjhuang}@fudan.edu.cn

Abstract

Multi-task learning (MTL) has received considerable attention, and numerous deep learning applications benefit from MTL with multiple objectives. However, constructing multiple related tasks is difficult, and sometimes only a single task is available for training in a dataset. To tackle this problem, we explored the idea of using unsupervised clustering to construct a variety of auxiliary tasks from unlabeled data or existing labeled data. We found that some of these newly constructed tasks could exhibit semantic meanings corresponding to certain human-specific attributes, but some were non-ideal. In order to effectively reduce the impact of non-ideal auxiliary tasks on the main task, we further proposed a novel meta-learning-based multi-task learning approach, which trained the shared hidden layers on auxiliary tasks, while the meta-optimization objective was to minimize the loss on the main task, ensuring that the optimizing direction led to an improvement on the main task. Experimental results across five image datasets demonstrated that the proposed method significantly outperformed existing single task learning, semi-supervised learning, and some data augmentation methods, including an improvement of more than 9% on the Omniglot dataset.

1 Introduction

Multi-task learning (Caruana 1997) is a learning paradigm in machine learning. Its goal is to improve the generalization performance of a single task by leveraging useful information contained in multiple related tasks (Zhang and Yang 2017). Multi-task learning has been used successfully across all applications of machine learning, from natural language processing (Collobert and Weston 2008) and speech recognition (Deng, Hinton, and Kingsbury 2013) to computer vision (Girshick 2015) and reinforcement learning (Wilson et al. 2007). Critical assumptions in these applications are that MTL typically involves very heterogeneous tasks and the tasks are closely related in some way (Ruder 2017; Zhang and Yang 2017).

Although the use of MTL to improve single task learning (STL) has achieved great success (Luong et al. 2015; Yim et al. 2015), these methods continue to have difficulty with two issues. **First**, we have always been hindered in searching a sufficient number of related tasks (Bingel and Søgaard 2017). In particular, in some domains, only a single task is available for training. **Second**, because of the lack of a good notion of when two tasks should be considered related (Ruder 2017), it is difficult to avoid introducing some unrelated tasks, which can result in the learned features being negatively influenced (Zhang and Yang 2017). Meyerson and Miikkulainen (2018) explored the pseudo-task augmentation method, which verified that MTL improved the generalization performance of single task. However, they only regarded the duplicates of a single task as multiple tasks, which lost the diversity of the task and could not take advantage of unlabeled data.

In this work, we considered an extreme situation where only a single task was available for training, but we expected to find sufficient related tasks and utilize the framework of deep MTL to improve the single task learning. To tackle the above issues, we used unsupervised clustering

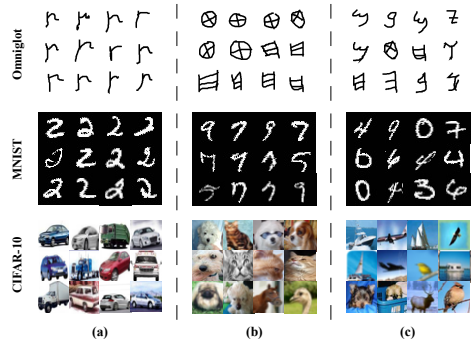

Figure 1: Examples of k-means clustering classes based on ACAI embeddings with random scaling (Berthelot et al. 2019), which can be used to perform multi-task learning. (a) Some of the clusters correspond well to true labels. (b) Some of the clusters exhibit semantic meaning, like the heads of the animals. (c) Others are non-ideal or are based on image artifacts, which could cause a performance deterioration. We proposed a meta-learning-based MTL to tackle this problem.

*Both authors contributed equally

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

methods on the unlabeled data or existing labeled data to automatically construct auxiliary tasks. Because the unsupervised clustering methods are fast implemented on a large scale, we can easily to obtain sufficient tasks. As for the *relatedness* of tasks, because we constructed the auxiliary tasks from the same data distribution as the main task, all of the tasks were $\mathcal{F}$-related and could yield provable multiple-task learning guarantees (Ben-David and Borbely 2008)[1]. We found that with simple k-means clustering mechanisms for partitioning the embedding space (Bojanowski and Joulin 2017; Donahue, Krähenbühl, and Darrell 2016), some of the clusters could construct reasonable related auxiliary tasks (Coates and Ng 2012). As shown in Figure 1 (a) and (b), some of the clusters correspond well to unseen labels or exhibit semantic meaning. Hence, these related tasks could benefit the MTL (Bonilla, Chai, and Williams 2008).

Unsupervised clustering would inevitably produce the non-ideal clusters, as shown in Figure 1 (c), which could cause a performance deterioration. To tackle this problem, we introduced a meta-learning-based multi-task learning framework (Meta-MTL), an novel variant of Model-Agnostic Meta-Learning (MAML) (Finn, Abbeel, and Levine 2017), to assist in the fast and robust adaptation of the MTL model, which drove the training of any auxiliary tasks to be beneficial to the main task. Specifically, Meta-MTL adopted a hard parameter sharing framework, which trained the shared hidden layers on a set of auxiliary tasks, while the meta-optimization objective was to minimize the loss on the main tasks, ensuring that the optimizing direction led to an improvement on the main task.

The main contributions of this paper can be summarized as follows: 1) we studied the multi-task learning in an extreme situation where only a single task was available, and used the k-means clustering to conduct multi-task learning to improve STL; 2) we proposed a novel meta-learning-based MTL method to assist in the fast and robust adaptation for the main task, which could effectively avoid the influence of the non-ideal clusters; and 3) experimental results across five image datasets indicated that the proposed method significantly outperformed the STL, semi-supervised learning, and some data augmentation methods.

2 Related Work and Background

This work is related to two research threads: multi-task learning and meta-learning. We first summarize the related work and then give some background. We also discuss the differences between the existing approaches and our own.

2.1 Multi-Task Learning

Many deep learning approaches require a large number of training samples and are becoming increasingly complex (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014; He et al. 2016). However, this requirement cannot be fulfilled in some applications because (labeled) samples are hard to collect. MTL is a good solution to this insufficient data problem when there are multiple related tasks each of which has limited training samples (Zhang and Yang 2017).

In MTL, a task $\mathcal{T}_t$ is usually accompanied by a training dataset D_t consisting of N_t training samples, i.e., $D_t = \{\mathbf{x}_i^t, \mathbf{y}_i^t\}_{i=1}^{N_t}$, where $\mathbf{x}_i^t \in \mathbb{R}^{d_t}$ is the ith training instance in $\mathcal{T}_t$ and y_i^t is its label. Although more sophisticated methods now exist, the most common approach is still based on the joint training of neural network models for multiple tasks (Caruana 1997), in which a joint model is decomposed into a feature extractor (shared layers) $\mathcal{F}$ that is shared across all tasks, and task-specific decoder $\mathcal{D}_t$ for each task. The model for the tth task is then defined as

$$\hat{\mathbf{y}}_i^t = \mathcal{D}_t(\mathcal{F}(\mathbf{x}_i^t; \theta_{\mathcal{F}}); \theta_{\mathcal{D}_t}), \tag{1}$$

where $\theta_{\mathcal{F}}$ and $\theta_{\mathcal{D}_t}$ are the parameters of $\mathcal{F}$ and $\mathcal{D}_t$, respectively. Then, the goal of the joint model is to find optimal parameters $\theta^* = (\{\theta_{\mathcal{D}_t}\}_{t=1}^T, \theta_{\mathcal{F}})$ such that

$$\theta^* = \operatorname*{argmin}_{\theta} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} \mathcal{L}(\hat{\mathbf{y}}_i^t, \mathbf{y}_i^t), \tag{2}$$

where $\mathcal{L}$ is a suitable loss function. More sophisticated deep MTL approaches can be characterized by "how to share," which specifies concrete ways to share knowledge among tasks, such as the low-rank approach (Han and Zhang 2016), task relation learning (Long et al. 2017), and so on. In this work, we consider a method to take advantage of the benefits of MTL to improve the STL, where all of the auxiliary tasks are generated by unsupervised clustering methods. Ideally, another potential benefit is that all of the MTL techniques can be used on our models.

2.2 Meta-Learning

Meta-learning was recently proposed and shown to achieve great performances on various few-shot learning tasks (Lake, Salakhutdinov, and Tenenbaum 2015). The intuition of meta-learning can be categorized by learning a matrix function that embeds data in the same class closer to each other (Vinyals et al. 2016; Snell, Swersky, and Zemel 2017), learning good weight initializations (Finn, Abbeel, and Levine 2017), and learning transferable optimizers (Andrychowicz et al. 2016). In this work, we adopt the idea from meta-learning for task generalization (Li et al. 2018), which is a novel variant of MAML for quickly and robustly adapting the newly constructed tasks to the main task. The proposed method utilizes a new meta-learning objective on shared layers that simulates transfer learning by fine-tuning, i.e., the shared layers are updated to minimize the loss on the main task based on a few examples and a few gradient descent steps on auxiliary tasks, while the task-specific decoders are updated to optimize the corresponding tasks. Our method can be applied to most deep STL architectures.

3 Meta-MTL with K-means Augmentation

In this work, we propose a novel MTL framework with a k-means augmentation framework to automatically construct

[1]Ben-David and Borbely (2008) proposed that two tasks were $\mathcal{F}$-related if the data for both tasks could be generated from a fixed probability distribution using a set of transformations $\mathcal{F}$. We constructed the tasks based on the same data distribution, which was a special case of $\mathcal{F}$-related.

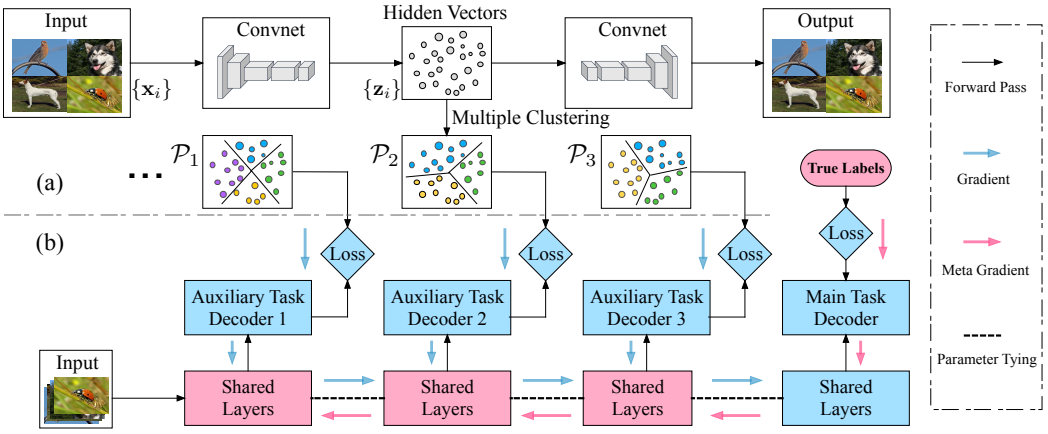

Figure 2: Graphical illustration of training process for proposed Meta-MTL with k-means augmentation. The entire training process is divided into two steps. (a) In the task generation step, we first run an unsupervised embedding learning algorithm to map the data into the embedding space $\mathcal{Z}$, producing $\{\mathbf{z}_i\}$. Then, we apply k-means T times to obtain T auxiliary tasks. (b) In the meta multi-task learning step, for each episode, one batch of images are sampled to update the task-specific decoders, while the meta-optimization objective of the shared layers is to minimize the loss on the main task. The blue boxes and arrows are mainly involved in task-specific learning, and those in purple are mainly involved meta-learning.

auxiliary tasks to improve single task learning, which is a modification to the basic MTL architecture. By using a novel meta-learning method for task generalization, the proposed method can quickly and robustly adapt the newly constructed tasks to the main task. The entire dataset construction and training process is illustrated in Figure 2.

3.1 Problem Statement

Assume that the human-specific main task to be learned is a classic supervised classification task. We have access to a small labeled dataset D_{main} and an unlabeled dataset D_{aux}. We expect to use the unsupervised clustering method on D_{aux} to effectively construct multiple auxiliary tasks $\{\mathcal{T}_t\}_{t=1}^{T}$. If we lack D_{aux}, it is also feasible to use D_{main} to construct auxiliary tasks. Our goal is to leverage the MTL framework on the automatically constructed tasks to improve the single task learning.

Concretely, we adopt a hard parameter sharing framework, which shares the hidden layers between the main task and all auxiliary tasks to obtain the shared features $\mathbf{h} = \mathcal{F}(\mathbf{x})$, while keeping several task-specific decoders to obtain task-specific outputs $\hat{\mathbf{y}}_t = \mathcal{D}_t(\mathbf{h})$. Given a main task and T newly constructed auxiliary tasks, the goal of joint training all of the tasks is to minimize the total loss, similar to that of the traditional MTL, as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N_0} \sum_{i=1}^{N_0} \mathcal{L}(\hat{\mathbf{y}}_i^0, \mathbf{y}_i^0) + \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} \mathcal{L}(\hat{\mathbf{y}}_i^t, \mathbf{y}_i^t)), \tag{3}$$

where $\theta = (\{\theta_{\mathcal{D}_t}\}_{t=0}^{T}, \theta_{\mathcal{F}})$. $\mathcal{D}_0$ refers to the output decoder of the main task, and $\{\mathcal{D}_t\}_{t=1}^{T}$ refers to the decoders of T auxiliary tasks. Baxter (1997) showed that the risk of overfitting the shared parameters is an order T smaller than that of overfitting the task-specific parameters. Thus, when learning more tasks simultaneously, there is less chance of overfitting on the main task, which can be verified in Section 4.3. In the next section, we discuss how we can construct tasks without ground-truth labels. In addition, we introduce a meta-learning-based MTL to assist in the fast and robust adaptation for the main task in Section 3.3.

3.2 Automatic Task Generation

We aim to construct multiple classification tasks from the unlabeled data and then learn how to efficiently utilize MTL to improve STL. If such tasks are adequately structured and related, then joint learning these tasks should make it possible to improve the learning of human-provided main tasks.

Notice that in the labeled dataset, the labels $\mathbf{y}$ can induce a partition $\mathcal{P} = \{C_l\}_{l=1}^{L}$ over D_0 by assigning all of the datapoints with label l to subset C_l. Then, subsets $\{C_l\}_{l=1}^{L}$ make up the entire data set, where L is the number of types of labels. Similarly, we can reduce the problem of constructing multiple auxiliary tasks to that of constructing different partitions over D_{aux}. However, simple partitioning methods may fail. For example, randomly partitioning the data shows no consistency between a task's training data and test data. Meanwhile, clustering in the pixel-space is also unappealing because of the poor correlation between the distances in the pixel-space and the clustering difficulty caused by the high dimensionality of the raw images (Hsu, Levine, and Finn 2019). Hence, all that's left is to find a reasonable alternative to human labels for defining the partition.

Fortunately, for certain architectures, like autoencoder, the latent embeddings have been shown to disentangle important factors of variation in the dataset, which makes such models useful for representation learning (Berthelot et al. 2019) and have been used for producing pseudo tasks with semantic meaning (Hsu, Levine, and Finn 2019). Thus,

we first run an unsupervised embedding learning algorithm $\mathcal{E}$ on D_{aux}, and then map the data $\{\mathbf{x}_i\}$ into the embedding space $\mathcal{Z}$, producing $\{\mathbf{z}_i\}$. We focus on a standard clustering algorithm, k-means, which takes a set of vectors as input, and clusters them into k distinct groups based on a geometric criterion. To produce a diverse task set, we apply random scaling to the dimensions of $\mathcal{Z}$ or random select half of the dimensions T times to induce T different matrices, and then generate T partitions $\{\mathcal{P}_t\}_{t=1}^{T}$ by running k-means T times. More precisely, for a single run of clustering, it jointly learns a $d \times k$ centroid diagonal matrix $\mathbf{U}$ and the cluster assignments $\mathbf{y}_l^*$ of each vector $\mathbf{z}_i$ such that

$$\mathcal{P}, \mathbf{U} = \underset{\mathbf{U}\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} \underset{\mathbf{y}_l^*\in\{0,1\}^k}{\operatorname{argmin}} \parallel \mathbf{z}_i - \mathbf{U}\mathbf{y}_l^* \parallel^2, \quad (4)$$

where $\mathbf{y}_l^{*\top} 1_k = 1$. Through the above function, we can obtain a set of optimal assignments $\{\mathbf{y}_l^*\}$ and a centroid matrix $\mathbf{U}$. These assignments are then used as pseudo-labels to construct the auxiliary task. As a result, the T partitions can correspond to T auxiliary tasks.

3.3 Meta-MTL with Newly Constructed Tasks

So far, we have obtained one main task and T auxiliary tasks. Next, we can directly apply the MTL framework to learn a joint neural classification function $\hat{\mathbf{y}}_i^t = \mathcal{D}_t(\mathcal{F}(\mathbf{x}_i;\theta_{\mathcal{F}}),\theta_{\mathcal{D}_t})$. However, this can be problematic when there are some non-ideal clusters, as shown in Figure 1 (c), which could cause a performance deterioration (Zhang and Yang 2017). For example, ACAI's pixel-wise reconstruction loss may prioritize information about the few "features" that dominate the reconstruction pixel count in complex images, resulting in clusters that only correspond to a limited range of factors such as the background color and pose (Hsu, Levine, and Finn 2019).

Therefore, to further improve the performance, we adopt a novel meta-learning-based multi-task learning method to achieve fast and robust task generalization. Instead of simply joint training $\mathcal{D}_t(\mathcal{F}(\mathbf{x}_i;\theta_{\mathcal{F}}),\theta_{\mathcal{D}_t})$, our method provides a way of leveraging the auxiliary tasks for better generalization on the main task in the MTL phase. More specifically, our method adopts a traditional hard sharing parameter structure, which trains each task-specific decoder $\mathcal{D}_t$ for its respective task while training the shared layers $\mathcal{F}$ on the auxiliary tasks to generalize to main task. In each training episode, we first uniformly sample a batch of tasks $\mathcal{T}$ from both the auxiliary tasks and main task $\{\mathcal{T}_t\}_{t=0}^{T}$, and conduct gradient descent based on the data from the sampled tasks $\mathcal{T}$ to optimize the task-specific decoders $\theta_{\mathcal{D}_\mathcal{T}}$. We simultaneously store the updated weights of shared layers $\theta_{\mathcal{F}}^*$ in preparation for the follow-up meta learning. For simplification, we use $\mathcal{L}$ to denote the loss function of our objective function. The update process is as follows:

$$\begin{aligned}\theta_{\mathcal{D}_\mathcal{T}} &= \theta_{\mathcal{D}_\mathcal{T}} - \alpha\nabla\mathcal{L}_{\mathcal{D}_\mathcal{T}}(\theta_{\mathcal{D}_\mathcal{T}})\\ \theta_{\mathcal{F}}^* &= \theta_{\mathcal{F}} - \alpha\nabla\mathcal{L}_{\mathcal{D}_\mathcal{T}}(\theta_{\mathcal{F}}).\end{aligned} \quad (5)$$

We then treat $\theta_{\mathcal{F}}^*$ as an initialized shared weight to optimize the original shared parameters on the main task based on the newly sampled datapoints in D_0. The final update on the shared parameters in each training episode can be written as follows:

$$\begin{aligned}\theta_{\mathcal{F}} &= \theta_{\mathcal{F}} - \beta\nabla\theta_{\mathcal{F}}\mathcal{L}_{\mathcal{D}_0}(\theta_{\mathcal{F}}^*)\\ &= \theta_{\mathcal{F}} - \beta\nabla\theta_{\mathcal{F}}\mathcal{L}_{\mathcal{D}_0}(\theta_{\mathcal{F}} - \alpha\nabla\mathcal{L}_{\mathcal{D}_\mathcal{T}}(\theta_{\mathcal{F}})),\end{aligned} \quad (6)$$

where both α and β are hyper-parameters of the two-stage learning rate. The above optimization can be conducted with stochastic gradient descent (SGD). We detail the task construction and Meta-MTL algorithm in Algorithm 1. In the meta-train stage, we optimize the task-specific decoders based on the sampled tasks $\mathcal{T}$, which ensures that the model has the ability for each task. Different from decoders, in the meta-test stage, we only optimize the shared layers based on the data from the main task, which ensures that the MTL optimizing direction leads to an improvement on the main task. In this way, the knowledge learned from $\mathcal{D}_t$ can provide a good initial representation that can be effectively fine-tuned using a few examples in $\mathcal{D}_0$, and thus achieves fast and robust adaptation for the main task.

Algorithm 1 Meta-MTL with K-means Augmentation

1: Run embedding learning algorithm $\mathcal{E}$ on D_{aux} and produce embeddings $\{\mathbf{z}_i\}$ from observations $\{\mathbf{x}_i\}$.
2: Run k-means on $\{\mathbf{z}_i\}$ T times (with random scaling or random selection on dimensions) to generate a set of partitions $\{\mathcal{P}_t = \{C^l\}_{l=1}^{L_t}\}_{t=1}^{T}$, which correspond to a set of auxiliary tasks $\{\mathcal{T}_t\}_{t=1}^{T}$.
3: **for** episode = 1, M **do**
4: Sample batch of tasks $\mathcal{T} \sim \{\mathcal{T}_t\}_{t=0}^{T}$.
5: **for all** $\mathcal{T}$ **do**
6: Sample K datapoints $D_\mathcal{T} = \{\mathbf{x}_j, \mathbf{y}_j\}$.
7: Evaluate $\nabla_{\theta_\mathcal{F}}$ and $\nabla_{\theta_{\mathcal{D}_t}}$ using $D_\mathcal{T}$ based on Equation 1.
8: Applying gradient decent to update the parameters of task-specific decoders $\theta_{\mathcal{D}_\mathcal{T}}$.
9: Compute updated parameters $\theta_{\mathcal{F}}^*$ with gradient descent based on Equation 5.
10: Sample datapoints $D_0 = \{\mathbf{x}_j, \mathbf{y}_j\}$ from $\mathcal{T}_0$ for the meta-update.
11: **end for**
12: Update the parameters of shared layers $\theta_\mathcal{F}$ based on Equation 6.
13: **end for**

4 Experiments

In this section, we aim to compare the proposed method with various baseline methods on five benchmarks. To avoid falsely embellishing the capabilities of our approach by overfitting to the specific datasets and task types, we did not perform any architecture engineering: we used architectures from prior work as-is, or lightly adapted them to our needs if necessary. We designed the experiments and showed the superiority in a range of settings: (1) the Meta-MTL can achieve a significant improvement in a limited training data setting; (2) the performance of the proposed model can be further improved when combined with unlabeled data; (3)

Method	Acc.
STL (Yang and Hospedales 2016)	65.72
ACAI embedding finetune (Berthelot et al. 2019)	67.91
MTL on all alphabets (Yang and Hospedales 2016)	70.98
MTL + Tasks with random labels, $T = 4$	60.98
MTL + Tasks with k-means labels, $T = 4$	61.26
PTA-F, $T = 4$ (Meyerson and Miikkulainen 2018)	70.63
PTA-F, $T = 10$ (Meyerson and Miikkulainen 2018)	71.52
Meta-MTL, $T = 4$	**72.04**
Meta-MTL, $T = 10$	**74.80**

Table 1: Accuracy on Omniglot dataset. THe test accuracy averaged across 50 alphabets is shown.

the proposed method can improve the performance of the model with data augmentation, and outperforms some semi-supervised methods; (4) the proposed method can also be used in challenging recent computer vision benchmarks, such as CIFIA-100 and miniImageNet. For the sake of simplicity, we used the same number of k-means clusters as that of the types of true labels for each dataset. Our code will be released at *https://github.com/Howardqlz/Meta-MTL*.

4.1 Omniglot Character Recognition

The Omniglot dataset (Lake, Salakhutdinov, and Tenenbaum 2015) contains handwritten characters in 50 different alphabets. Each alphabet with its own number of unique characters ($14 \sim 55$) induces its own character recognition task. In total, there are 1,623 unique characters, and each has exactly 20 instances. Here, each task corresponds to an alphabet, and the goal is to recognize its characters. To reduce variance and improve the reproducibility of the experiments, we used the same 50/20/30% train/validation/test split as Meyerson and Miikkulainen (2018) for each task. Methods were evaluated with respect to all 50 tasks. The underlying model $\mathcal{F}$ for all the baselines is a simple four layer convolutional network that has been shown to yield good performance on Omniglot (Yang and Hospedales 2016). Each of these four convolutional layers has 53 filters and 3×3 kernels, and is followed by a 2×2 maxpooling layer and dropout layer with a 0.5 dropout probability. Each task-specific decoder $\mathcal{D}_t$ has two fully connected layers with 848 and a specific number of classes of neurons, respectively. In our method, we used the k-means method on ACAI embeddings (Berthelot et al. 2019) to obtain the auxiliary tasks.

Table 1 reports the average accuracy values across all 50 tasks (alphabets). Our proposed Meta-MTL methods surpass the STL model by more than 9%. Not all of the baseline models work well when the training dataset is very small. Meyerson and Miikkulainen (2018) constructed an MTL method using duplicates of a single task, but did not introduce related tasks, which limited the performance[2]. In particular, constructing tasks with random labels also resulted in a huge performance degradation. In contrast, our method can achieve a significant improvement when the main task leverages auxiliary tasks constructed by unsupervised clustering on ACAI embeddings.

[2]In Table 1∼5, we evaluated six kinds of explicit controls to PTA trajectories (Meyerson and Miikkulainen 2018). The PTA-F reported in these tables is the best one.

Method	Acc.
STL (Yang and Hospedales 2016)	91.83
ACAI embedding finetune (Berthelot et al. 2019)	93.09
Self-training (Rosenberg, Hebert, and Schneiderman 2005)	92.20
Co-training (Chen, Weinberger, and Blitzer 2011)	91.87
MTL + Tasks with random labels, $T = 4$	92.27
MTL + Tasks with k-means labels, $T = 4$	92.86
PTA-F, $T = 4$ (Meyerson and Miikkulainen 2018)	92.67
PTA-F, $T = 10$ (Meyerson and Miikkulainen 2018)	91.98
Meta-MTL, $T = 4$	**93.37**
Meta-MTL, $T = 10$	**94.22**
Meta-MTL, $T = 4\dagger$	**93.76**
Meta-MTL, $T = 10\dagger$	**94.42**

Table 2: Accuracy on MNIST dataset. All of the models use 1% training data. The models marked with † use the remaining unlabeled data.

4.2 MNIST Number Classification

The MNIST dataset consists of 70,000 hand-drawn examples of the 10 numerical digits. The previous split used the original MNIST 60,000/10,000 training/testing split. In this work, we evaluated the proposed method using only 1% training data, and the remaining data were treated as unlabeled data. Because it is a simpler task, we used two layers of the CNN architecture as the shared layer $\mathcal{F}$. The first convolutional layer has 32 filters of size 5×5, followed by 2×2 max pooling. The second convolutional layer has 64 filters of size 4×4, and again a 2×2 max pooling, as shown in (Yang and Hospedales 2016).

Table 2 shows the accuracy of the baseline models and our model on the test dataset of MNIST. Similar to Omniglot, our method outperforms all of the baseline models when the training dataset is very small. In particular, when the number of the auxiliary tasks increases, the performance of PTA-F drops, but our model performance improves a lot. In addition, if we used the remaining unlabeled data to train the embeddings, our method can obtain better results. The Meta-MTL model also outperforms semi-supervised methods, such as the self-training and co-training methods by more than 3%.

To investigate why the proposed method can improve the STL, we visualized the behaviors of the STL and Meta-MTL when encountering somewhat ambiguous hand-drawn examples. We trained Meta-MTL using the same hyperparameters as the STL, while adding only one auxiliary task to improve the STL. Some typical examples are shown in Figure 3. From the figure, we can see that with a small training dataset, the STL is more likely to confuse some ambiguous samples, while the auxiliary task can help the model recognize them. For instance, as shown in the first row, the STL is confused if this picture is "2" or "8," while the auxiliary cluster can give key information that this image is gathered with many examples of "2." Then, the Meta-MTL can correct the STL's mistakes. In addition, we also visualized the outputs of the shared layers when encountering some ambiguous images. We resized the 1,024

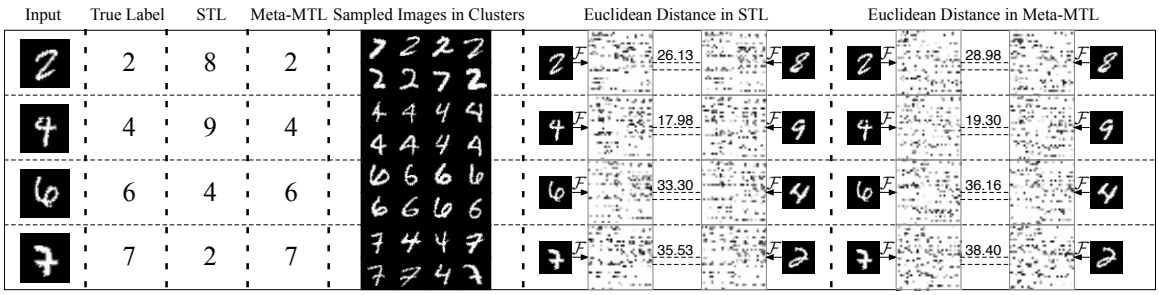

Figure 3: Visualization of behaviors of STL and Meta-MTL. The third and fourth columns are the labels predicted by STL and Meta-MTL, respectively. We randomly sample 8 images from the k-means cluster which contains the original image shown in the fifth column. In addition to exhibiting the output matrices of the shared layers, we also compute the Euclidean distance between the output vectors. The larger Euclidean distance shows stronger capacity to distinguish the ambiguous images.

Model	**Acc.**	
	CNN	CNN‡
STL (Yang and Hospedales 2016)	72.48	75.91
ACAI embedding finetune (Berthelot et al. 2019)	64.94	64.94
MTL + Tasks with random labels, $T = 4$	70.81	74.48
MTL + Tasks with k-means labels $T = 4$	71.06	74.61
PTA-F, $T = 4$ (Meyerson and Miikkulainen 2018)	69.20	72.54
PTA-F, $T = 12$ (Meyerson and Miikkulainen 2018)	68.48	70.96
Meta-MTL, $T = 4$	**75.02**	**78.65**
Meta-MTL, $T = 12$	**75.69**	**79.65**

Table 3: Accuracy on CIFAR-10 dataset. The models marked with ‡ apply data augmentation.

Model	**Acc.**	
	20 C	100 C
STL (Yang and Hospedales 2016)	55.94	44.19
ACAI embedding finetune (Berthelot et al. 2019)	44.37	34.40
MTL + Tasks with random labels	51.00	41.30
MTL + Tasks with k-means labels	51.84	42.30
PTA-F, $T = 8$ (Meyerson and Miikkulainen 2018)	51.67	45.86
PTA-F, $T = 20$ (Meyerson and Miikkulainen 2018)	51.69	47.43
Meta-MTL, $T = 8$	**59.66**	**47.01**
Meta-MTL, $T = 20$	**60.39**	**47.94**

Table 4: Accuracy on CIFAR-100 dataset. The "20 C" means that the 100 classes in the CIFAR-100 are grouped into 20 superclasses, and the "100 C" means the original 100 classes.

dimensional output vectors to the matrices of size 32×32, and then plotted the largest 200 values in the matrices. Comparing these pictures, we can observe that the outputs of the ambiguous images in STL have greater similarity than those in Meta-MTL. Therefore, the STL is more difficult to distinguish the ambiguous images. For a clearer comparison, we computed the Euclidean distance between the ambiguous images. We can see that our Meta-MTL can more easily distinguish the two images in the shared layer than STL by expanding the distance between the ambiguous images.

4.3 CIFAR-10 and CIFAR-100 Tiny Images Dataset

The CIFAR-10 dataset (Krizhevsky, Hinton, and others 2009) is composed of 10 classes of natural images with 50,000 training images, and 10,000 testing images. Each image is a RGB image of size 32×32. For this dataset, we applied the same CNN architecture and decoders as those in MNIST dataset, while we evaluated our method and all of the baseline methods using two settings. One setting was training the model on the original dataset without any data augmentation method. The other setting was training on the dataset with some data augmentation techniques, i.e., randomly performing horizontal flips and gray scale. Because the labels of the auxiliary tasks were generated by the whole image, we did not apply the randomly crop technique on the images.

The results are listed in Table 3. We can see that our method outperforms the other baseline methods by a large margin. In contrast to the PTA-F trained on the Omniglot and MNIST, when the number of decoders increased, the performance of this model dropped greatly by more than 4%. The performance of MTL models with random labels and k-means labels also dropped slightly. In contrast, our model exceeds the STL by more than 3%. In addition, the contribution of our model and data augmentation may be orthogonal, because whether or not the model uses data augmentation, our methods have similar improvements.

The CIFAR-100 dataset (Krizhevsky, Hinton, and others 2009) is the same in size and format as the CIFAR-10 dataset, but it contains 100 classes. Thus, the number of images in each class is only one-tenth of the CIFAR-10 dataset. This dataset also provides another label version that the 100 classes in the CIFAR-100 are grouped into 20 superclasses. We did not tune the hyper-parameters and used the same setting as the CIFAR-10 dataset. The results are listed in Table 4. Our Meta-MTL also outperforms the other baseline models by more than 4% in "20 C" and 3% in "100 C", respectively.

To verify the effect of our meta-MTL algorithm and k-means labels, we evaluated the performance of our method and several baseline methods with different numbers of auxiliary tasks on the CIFAR-10 dataset. The comparison of these methods is shown in Figure 4. From the figure, we can see that our model "K-means Labels + Meta" achieve a stable and outstanding performance for each

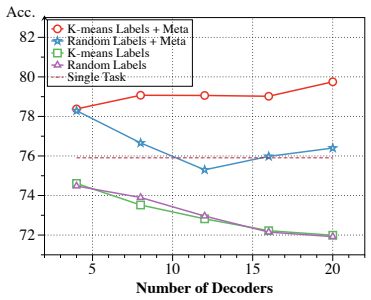

Figure 4: Comparison of MTL models trained on the CIFAR-10 datasets with different numbers of tasks.

number of decoders. As the number of decoders increases, the performance of the model also shows an upward trend. In contrast, the performance of the models with k-means or random labels but not using a meta learning algorithm gradually declines when the number of decoders increases. Comparing our method with model "K-means Lables", we can find that the proposed meta-MTL algorithm can suppress the impact of error labels and leverage ideal labels from auxiliary tasks, make the model achieve more than 8% higher accuracy than that without the meta-learning algorithm. In addition, comparing model "K-means Labels + Meta" with "Random Labels + Meta", we can see that k-means labels can exhibit semantic meanings, make the model achieve more than 4% higher accuracy.

4.4 MiniImageNet

In this subsection, we aim to verify the performance of the proposed method on a more difficult dataset. MiniImageNet dataset (Ravi and Larochelle 2016), which consists of 100 classes, each with 600 examples of size 84×84. The images are predominantly natural and realistic. We split the dataset into training and test sets in a 50,000/10,000 ratio, then trained and tested these methods using five-fold cross validation. To evaluate the performance of the proposed method based on different embedding spaces, we compared the DeepCluster (Caron et al. 2018) with ACAI (Berthelot et al. 2019) method to produce embeddings. Moreover, we applied random scaling and randomly select half dimensions of the embeddings for the k-means clusters to produce different partitions. The baseline model is the same as (Finn, Abbeel, and Levine 2017). The underlying model $\mathcal{F}$ is a simple four layer convolutional network, of which each layer has 32 filters and 3×3 kernels, and is followed by a 2×2 maxpooling layer. Each task-specific decoder $\mathcal{D}_t$ has one fully connected layer with 100 neurons.

The final results are shown in Table 5. The Meta-MTL model still outperforms all of the baseline models. Comparing the model finetuned on the DeepCluster embeddings with that on the ACAI embeddings, we found that the DeepCluster model has a much stronger performance than the ACAI. This is because ACAI's pixel-wise reconstruction loss may prioritize information about the background color in these complex images, while reducing the focus on the target objects (Hsu, Levine, and Finn 2019). Hence, the

Embedding	Model	Acc.
Random initialization	STL	38.98
	MTL, Random Labels	36.70
	PTA-F, $T = 4$	34.93
DeepCluster	Embedding finetune	35.14
	MTL, k-means, $T = 4, \diamondsuit$	37.24
	MTL, k-means, $T = 4, \heartsuit$	36.78
	Meta-MTL, $T = 4, \diamondsuit$	**41.07**
	Meta-MTL, $T = 4, \heartsuit$	**40.86**
ACAI	Embedding finetune	22.90
	MTL, k-means, $T = 4, \diamondsuit$	37.36
	MTL, k-means, $T = 4, \heartsuit$	36.98
	Meta-MTL, $T = 4, \diamondsuit$	**40.43**
	Meta-MTL, $T = 4, \heartsuit$	**40.60**

Table 5: Accuracy on miniImageNet dataset. The models marked with $\diamondsuit$ apply the random scaling on the embeddings to obtain the different tasks, while those marked with $\heartsuit$ apply random selection for half of the dimensions on the embeddings.

Meta-MTL model can achieve better results based on the clusters obtained by DeepCluster embeddings compared to those obtained by ACAI embeddings, which further leads to better performance of Meta-MTL. However, we could hardly distinguish which method to transform the embedding space for multiple partitions is better. The method with random scaling is not always better than that with random selecting half dimensions of embeddings. Therefore, both the random scaling and random selecting methods can produce different effective embeddings to achieve similar performances.

5 Conclusion

In this work, we proposed the use of an unsupervised clustering method to construct a variety of auxiliary tasks to improve single task learning. We found that with simple k-means clustering mechanisms for partitioning the embedding space, some of the clusters could construct reasonable related auxiliary tasks. We also proposed a novel meta-learning-based multi-task learning framework to assist the fast and robust adaptation for the main task, which can effectively reduce the influence of the non-ideal clusters. Because the unsupervised clustering methods can be quickly implemented on a large scale, we can easily construct a sufficient number of auxiliary tasks. Experimental results across five image datasets demonstrated that the proposed method significantly outperforms the previous methods.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key R&D Program (No. 2018YFB1005104, 2018YFC0831105), National Natural Science Foundation of China (No. 61976056, 61532011, 61751201), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), Science and Technology Commission of Shanghai Municipality Grant (No.18DZ1201000, 16JC1420401, 17JC1420200).

References

Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; and De Freitas, N. 2016. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 3981–3989.

Baxter, J. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning* 28(1):7–39.

Ben-David, S., and Borbely, R. S. 2008. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine learning* 73(3):273–287.

Berthelot, D.; Raffel, C.; Roy, A.; and Goodfellow, I. 2019. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *ICLR*.

Bingel, J., and Søgaard, A. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*, 164–169.

Bojanowski, P., and Joulin, A. 2017. Unsupervised learning by predicting noise. In *ICML*, 517–526. JMLR. org.

Bonilla, E. V.; Chai, K. M.; and Williams, C. 2008. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, 153–160.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *ECCV*, 132–149.

Caruana, R. 1997. Multitask learning. *Machine learning* 28(1):41–75.

Chen, M.; Weinberger, K. Q.; and Blitzer, J. 2011. Co-training for domain adaptation. In *Advances in neural information processing systems*, 2456–2464.

Coates, A., and Ng, A. Y. 2012. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*. Springer. 561–580.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 160–167. ACM.

Deng, L.; Hinton, G.; and Kingsbury, B. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8599–8603. IEEE.

Donahue, J.; Krähenbühl, P.; and Darrell, T. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1126–1135. JMLR. org.

Girshick, R. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

Han, L., and Zhang, Y. 2016. Multi-stage multi-task learning with reduced rank. In *Thirtieth AAAI Conference on Artificial Intelligence*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Hsu, K.; Levine, S.; and Finn, C. 2019. Unsupervised learning via meta-learning. In *International Conference on Learning Representations*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 1097–1105.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2018. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Long, M.; Cao, Z.; Wang, J.; and Philip, S. Y. 2017. Learning multiple tasks with multilinear relationship networks. In *Advances in neural information processing systems*, 1594–1603.

Luong, M.-T.; Le, Q. V.; Sutskever, I.; Vinyals, O.; and Kaiser, L. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Meyerson, E., and Miikkulainen, R. 2018. Pseudo-task augmentation: From deep multitask learning to intratask sharing-and back. In *International Conference on Machine Learning*, 3508–3517.

Ravi, S., and Larochelle, H. 2016. Optimization as a model for few-shot learning.

Rosenberg, C.; Hebert, M.; and Schneiderman, H. 2005. Semi-supervised self-training of object detection models. *WACV/MOTION* 2.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 4077–4087.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, 3630–3638.

Wilson, A.; Fern, A.; Ray, S.; and Tadepalli, P. 2007. Multi-task reinforcement learning: a hierarchical bayesian approach. In *ICML*, 1015–1022. ACM.

Yang, Y., and Hospedales, T. 2016. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391*.

Yim, J.; Jung, H.; Yoo, B.; Choi, C.; Park, D.; and Kim, J. 2015. Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 676–684.

Zhang, Y., and Yang, Q. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.