Meta Learning for Unsupervised Clustering

Han-Ul Kim[1]
hanulkim@mcl.korea.ac.kr
Yeong Jun Koh[2]
yjkoh@cnu.ac.kr
Chang-Su Kim[1]
changsukim@korea.ac.kr

[1] School of Electrical Engineering,
Korea University,
Seoul, Korea

[2] Department of Computer Science & Engineering,
Chungnam National University,
Daejeon, Korea

Abstract

Learning an embedding space is essential in clustering. Deep learning has been used recently for this purpose, yielding impressive clustering results. However, it remains challenging to discover clusters in small data, which are insufficient to train deep networks. To address this challenge, we adopt the meta learning strategy, which learns to learn new tasks efficiently. We propose a novel meta learner, called MLC-Net, which mimics numerous clustering tasks during the training to learn an effective embedding space for new clustering tasks. MLC-Net has three building blocks: encoder, centroid, and prediction blocks. The encoder block transforms input patterns into an embedding space, while the centroid block estimates a representative feature for each cluster, called pseudo-centroid. It makes the embedding space more effective and more reliable, by learning an embedding space and a pseudo-centroid estimator jointly. Extensive experimental results on the Omniglot, MNIST, and Mini-ImageNet datasets demonstrate that MLC-Net achieves the state-of-the-art unsupervised clustering, as well as few-shot classification, performances.

1 Introduction

Real-world data are often in a high-dimensional space, in which it is not easy to define an effective metric. Therefore, embedding schemes, which map data into an embedding space with an appropriate metric (often resulting in dimensionality reduction), have been developed for clustering [18, 21]. Especially, motivated by the success of deep learning, recent techniques [7, 31, 32] focus on the design of embedding neural networks, yielding reasonable clustering performances on the datasets in [3, 14]. However, when there are insufficient data for clustering, these techniques fail to yield reliable results. According to [7], some deep-learning-based clustering algorithms provide inferior results to the simple K-means [16] on the Omniglot dataset [13], in which each class has only 20 sample patterns. Discovering clusters using small data still remains challenging.

In contrast to machine learning, human intelligence discovers new concepts efficiently from a small amount of data based on prior knowledge. This notable gap between human and machine intelligence motivated the study of meta learning [27, 28]. In meta learning, an intelligence model is trained on diverse learning tasks so that it can be generalized to solve

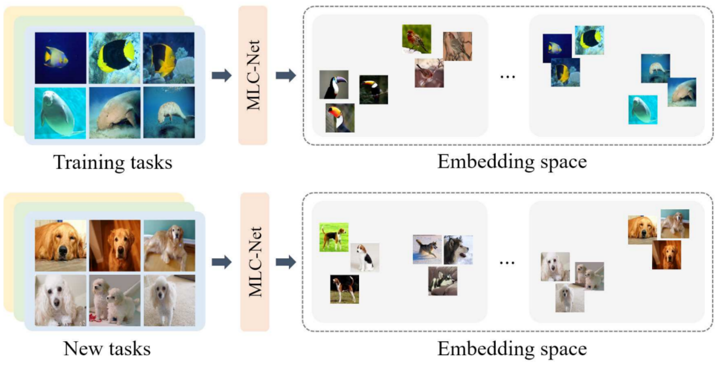

Figure 1: The proposed MLC-Net mimics numerous clustering tasks during the training to learn an embedding space, in which new clustering tasks can be performed effectively.

new tasks using only a few examples. Few-shot classification is a notable sub-branch of meta learning [4, 10, 22, 24, 30]. It learns a classifier to accommodate new class types, not seen in the training phase, by employing only a few examples per new class in the inference phase. However, at least one example per new class is required to achieve few-shot classification.

In this work, we propose a novel meta learner, called MLC-Net (Meta Learning for Clustering), to overcome the weakness of clustering algorithms in the case of small data [7, 31, 32]. As illustrated in Figure 1, the objective of MLC-Net is to learn an embedding space that can be generalized to new clustering tasks. To this end, we design MLC-Net to contain three building blocks: encoder block, centroid block, and prediction block. The encoder block transforms input patterns into an embedding space, while the centroid block estimates a representative feature for each cluster, called pseudo-centroid. Then, the prediction block computes the distances from each pattern to the clusters in the embedding space and performs the nearest neighbor classification. During the training, we jointly learn the embedding space and the pseudo-centroid estimator to minimize the sum of cross-entropy, self-augmentation loss, and self-ensemble loss. We experimentally show that this joint learning makes the embedding space more effective and more reliable for new clustering tasks.

We assess the clustering performances of the proposed MLC-Net on the Omniglot [13] and MNIST [14] datasets. Omniglot includes only a few patterns per class, whereas MNIST has sufficient data for training the conventional algorithms. On Omniglot, MLC-Net provides excellent generalization performance for new tasks and outperforms the state-of-the-art clustering algorithms [7, 31] significantly. On MNIST, MLC-Net yields competitive performance to the conventional algorithms, even though it is trained on the Omniglot dataset without any fine-tuning on the MNIST dataset. Furthermore, we also compare MLC-Net with the few-shot classifiers on Mini-ImageNet [30]. In this dataset, MLC-Net yields comparable or even better performance than the state-of-the-art few-shot classifiers.

To summarize, this work has three main contributions:

- We introduce the notion of meta learning for clustering to address the challenging scenario, in which only a small amount of data are available for clustering.
- We propose MLC-Net to learn an effective embedding space for clustering.
- MLC-Net achieves excellent unsupervised clustering, as well as few-shot classification, performances on the Omniglot, MNIST, and Mini-ImageNet datasets.

2 Related Work

2.1 Clustering

For unsupervised clustering, the classical K-means algorithm [16] has been used extensively due to its simplicity and effectiveness. K-means works well when data are distributed compactly around distinctive centroids, but real-world data often fail this condition. Hence, attempts [18, 21] have been made to transform data into an embedding space in which K-means can perform clustering successfully. For example, [21] developed a kernel-based algorithm for nonlinear principal component analysis (PCA), and considered the K-means clustering in a feature space obtained by a nonlinear kernel. Also, [18] proposed a spectral clustering algorithm in an embedding space, which was obtained using the K largest eigenvectors of the Laplacian matrix, derived from affinity scores between data points.

Deep learning also has been used for unsupervised clustering, since it can process high-dimensional data effectively by learning embedding spaces [6, 29]. Xie *et al.* [31] developed a deep embedding method to learn feature representation and cluster assignment simultaneously. Yang *et al.* [32] proposed a method to learn a K-means-friendly space, which performs the network update step and the K-means step alternately. Jiang *et al.* [9] introduced the generative clustering based on a variational autoencoder. Hu *et al.* [7] adopted a data augmentation scheme. They trained deep neural networks to maximize the mutual information between input and predicted representation, while constraining the predicted representation of augmented data to be close to those of original data. However, the need for huge data for network training is a common problem in [7, 9, 31, 32]. When insufficient data are available, these unsupervised clustering algorithms cannot train the networks properly.

2.2 Meta Learning

The objective of meta learning [27, 28] is to enable an intelligence model to learn a new task using only a few data. In particular, few-shot classification aims at learning a classifier that can adapt to new classes, not seen in the training, using only a few examples per class. For this purpose, we take an approach to learn an embedding space for a labeled set of few examples, so as to perform the classification of queries accurately in the space. Thus, we summarize only the most relevant methods [10, 22, 24, 30] to the proposed algorithm.

Koch *et al.* [10] trained Siamese networks to determine whether two examples belong to the same class and used them for one-shot classification. Vinyals *et al.* [30] designed the matching networks, which compute the cosine-distances between input and support patterns to perform the nearest neighbor one-shot classification. Also, they devised the episodic training, which samples only subclasses of a training set to mimic one-shot classification tasks in the inference phase. By matching the training and test conditions, the episodic training makes the matching networks effective for various classification tasks. The episodic training has been employed also in [22, 24]. Snell *et al.* [22] developed the prototypical networks to represent each class as a prototype, which is the mean feature of the corresponding few examples, and predicted the label of a query using the prototypes. Sung *et al.* [24] proposed the relation network to determine similarities between input and support patterns, instead of using conventional metrics such as the cosine-similarity. Notice that these few-shot classifiers [10, 22, 24, 30] require at least one example per class (not seen in the training), whereas the proposed MLC-Net can assign each query to one of the clusters without requiring any annotation.

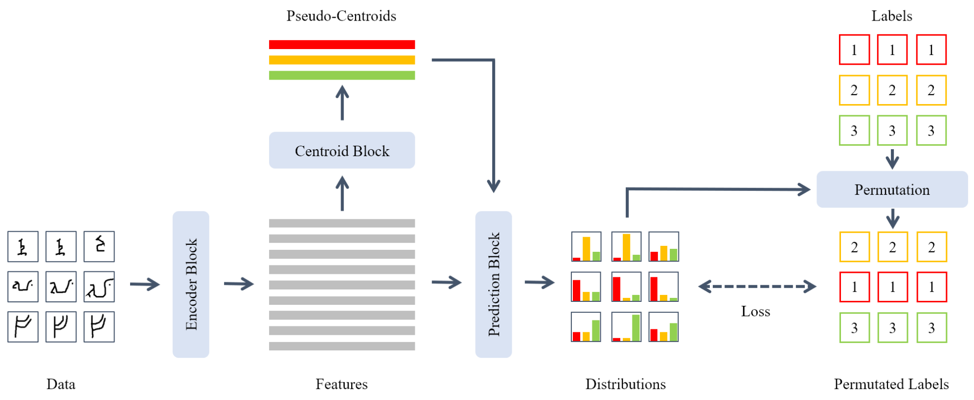

Figure 2: An overview of the proposed MLC-Net.

3 MLC-Net

The proposed MLC-Net yields effective features for clustering based on meta learning. MLC-Net transforms patterns $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ into an embedding space (or feature space), while generating a representative feature for each cluster, referred to as *pseudo-centroid*. It then estimates cluster labels $\{y_1, \cdots, y_N\}$ of the patterns based on the nearest neighbor criterion, where $y_i \in \{1, \cdots, K\}$ and K is the number of clusters.

Unlike the meta learning techniques for few-shot classification [4, 10, 20, 22, 24, 30], it is not straightforward for MLC-Net to determine representative features for clusters since it cannot utilize support pairs of patterns and their ground-truth labels. Unless the representative features are reliable, cluster labels may be estimated inaccurately, and the resultant embedding space may be ineffective. To overcome this difficulty, MLC-Net generates the representative feature for each cluster by linearly combining input patterns in the embedding space. Then, MLC-Net jointly learn how to embed the patterns and how to determine the representative features in the training phase. During the training, MLC-Net employs known but independent classes, which are readily available but different from clusters in the inference phase, and their patterns to learn the embedding space for clustering tasks.

3.1 Model

Figure 2 illustrates the proposed meta learner, MLC-Net, which has three building blocks to emulate various clustering tasks during the training: encoder, centroid, and prediction blocks. Let us describe each block subsequently.

3.1.1 Encoder Block:

Let $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ be D-dimensional input patterns. The encoder block transforms these patterns into M-dimensional feature vectors $\{\mathbf{z}_1, \cdots, \mathbf{z}_N\}$ through an embedding function $f : \mathbb{R}^D \rightarrow \mathbb{R}^M$. To implement f, we employ a convolutional neural network [11], which is suitable for handling image tasks. Then, we train it to yield the embedding space, in which clustering can be performed effectively in the inference phase.

3.1.2 Centroid Block:

We determine the representative feature, called *pseudo-centroid*, of each cluster by linearly combining the input patterns in the embedding space. Let $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_K] \in \mathbb{R}^{M \times K}$ be the centroid matrix, whose columns are pseudo-centroids $\mathbf{c}_1, \cdots, \mathbf{c}_K$ for K clusters. Given the input feature matrix $\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_N] \in \mathbb{R}^{M \times N}$, we compute the centroid matrix by

$$\mathbf{C} = \mathbf{Z}\mathbf{A} \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{N \times K}$ is the attention matrix, whose element a_{ij} indicates the contribution of pattern $\mathbf{z}_i$ to pseudo-centroid $\mathbf{c}_j$.

The attention matrix is also obtained by analyzing the input feature matrix $\mathbf{Z}$. To this end, we design the attention function g, given by

$$\mathbf{A} = g(\mathbf{Z}) = h_n(h_r(\mathbf{Z}^T\mathbf{W})) \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{M \times K}$ is a matrix of trainable parameters, the rectifier h_r applies the rectified linear unit (ReLU) to each element in $\mathbf{Z}^T\mathbf{W}$, and the normalizer h_n makes the sum of elements in each column in $h_r(\mathbf{Z}^T\mathbf{W})$ equal to 1 using the softmax function. By combining (1) and (2), we have

$$\mathbf{C} = \mathbf{Z} \times h_n(h_r(\mathbf{Z}^T\mathbf{W})). \tag{3}$$

Thus, the objective is to train the parameter matrix $\mathbf{W}$ so that the resultant pseudo-centroids in $\mathbf{C}$ represent the corresponding clusters faithfully.

Various functions can be used as the attention function g in (2). However, we adopt the particular g in (2), which is a simple nonlinear neural network, for the following reasons. For the moment, let us ignore the nonlinear components h_n and h_r in (2). Then, (3) can be written as

$$\mathbf{C} \simeq \mathbf{Z}\mathbf{Z}^T\mathbf{W} = ((N-1)\sum_{\mathbf{z}} + N\mathbf{m}_{\mathbf{z}}\mathbf{m}_{\mathbf{z}}^T)\mathbf{W} \tag{4}$$

where $\mathbf{m}_{\mathbf{z}}$ and $\sum_{\mathbf{z}}$ are the sample mean vector and the sample covariance matrix of the feature vectors in $\mathbf{Z}$, respectively [19]. Assuming zero mean $\mathbf{m}_{\mathbf{z}} = \mathbf{0}$, we have

$$\mathbf{C} \simeq (N-1)\sum_{\mathbf{z}}\mathbf{W} = (N-1)\mathbf{Q}\Lambda\mathbf{Q}^T\mathbf{W} \tag{5}$$

where $\sum_{\mathbf{z}} = \mathbf{Q}\Lambda\mathbf{Q}^T$ is the eigenvalue decomposition of the covariance matrix $\sum_{\mathbf{z}}$, *i.e.* the principal component analysis of the feature space [1, 23]. Note that, by controlling the parameter matrix $\mathbf{W}$, we can locate the pseudo-centroids within the subspace spanned by the first few principal components. For example, when the first column of $\mathbf{W}$ is set to $\mathbf{Q}[\beta, 0, \cdots, 0]^T$, pseudo-centroid $\mathbf{c}_1$ is located on the line along the first principal component. In other words, we can achieve the dimensionality reduction, if the embedding function f leaves some redundancy in the feature space. In addition to the dimensionality reduction, we can locate the pseudo-centroids evenly within the reduced space, again by selecting $\mathbf{W}$ appropriately, so that they are far from one another in the space.

Approximating the attention function g in (2) to a linear one, while ignoring the nonlinearity, offers insight into pseudo-centroids. However, the nonlinear components h_n and h_r play important roles. The softmax normalizer h_n ensures that each pseudo-centroid in $\mathbf{C}$ is a convex combination of the feature vectors in $\mathbf{Z}$. In other words, all weighting coefficients are non-negative and their sum equals 1. Thus, h_n guarantees that all pseudo-centroids are located within the convex hull of the feature vectors. Also, the rectifier h_r increases the learning capacity of MLC-Net, since stacking a series of linear and non-linear operations enables a neural network to achieve its objective more effectively in general [5, 11, 15]. Section 4 will analyze the impacts of the non-linearity in the attention function in more detail.

3.1.3 Prediction Block:

Using the output of the encoder and centroid blocks, the prediction block estimates the probability $p(y_i = k|\mathbf{x}_i)$ that pattern $\mathbf{x}_i$ belongs to cluster k based on the nearest neighbor criterion.

$$p(y_i = k|\mathbf{x}_i) = \frac{e^{-\|\mathbf{z}_i - \mathbf{c}_k\|_2^2}}{\sum_{k'=1}^{K} e^{-\|\mathbf{z}_i - \mathbf{c}_{k'}\|_2^2}} \tag{6}$$

Notice that the centroid block and the prediction block are employed only in the training phase, whereas the K-means clustering is performed straightforwardly in the learned embedding space in the inference phase. These two blocks, however, are also essential. During the training, we learn the embedding space jointly with $\mathbf{W}$ in (2), which generates pseudo-centroids for clustering tasks. This joint learning of embedding space and pseudo-centroids is less prone to overfit. In contrast, if we learn the embedding space by performing the K-means clustering for each clustering task, the space may be biased for the training tasks and may fulfill the objective of meta learning for new tasks less successfully. Section 4 will confirm that MLC-Net yields a more effective embedding space, by employing the centroid block and the prediction block.

3.2 Learning

To train MLC-Net, we adopt the episodic training [30], where each episode is composed of subclasses of a training dataset. In this work, each episode is a task to cluster patterns in known classes, which are, however, not used in the test phase. We generate diverse combinations of classes, so that numerous episodes improve the generalization performance of MLC-Net to new clustering tasks. We construct an episode by randomly selecting a subset of classes from the training dataset and choosing a subset of patterns from each selected class.

Given a training episode, MLC-Net performs clustering and the performance is quantified by three loss functions: cross-entropy, self-augmentation loss, self-ensemble loss. The cross-entropy is for learning an effective embedding space, where clusters are well separated but patterns in each cluster are compactly distributed. The self-augmentation and self-ensemble losses are for improving the generalization performance. We minimize the sum of the loss functions to train the set θ of parameters in MLC-Net via the stochastic gradient descent. Note that θ consists of the parameters in the embedding function f in the encoder block and the parameter matrix $\mathbf{W}$ in the centroid block.

3.2.1 Cross-Entropy:

It penalizes prediction errors of MLC-Net, which estimates the softmax probabilities of clusters via (6). The estimated clusters may not correspond to the ground-truth classes, since MLC-Net determines them without resorting to the ground-truth labels. Hence, it is necessary to reorder the ground-truth labels to define the cross-entropy. We adopt the permutation function $\pi(\cdot)$, which is determined to maximize the unsupervised accuracy [2, 7, 9, 31, 32],

$$\pi^* = \arg\max_{\pi} \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\hat{y}_i = \pi(l_i)) \tag{7}$$

where $\hat{y}_i$ denotes the predicted label of pattern $\mathbf{x}_i$ to maximize the softmax probability in (6), and l_i denotes the ground-truth label of $\mathbf{x}_i$. The optimal permutation π^* can be obtained by the Hungarian algorithm [12]. Then, we compute the cross-entropy

$$\mathcal{L}_{\text{ce}}(\mathbf{x}_i, l_i; \theta) = -\log p(y_i = \pi^*(l_i)|\mathbf{x}_i). \tag{8}$$

This cross-entropy helps to discriminate patterns in different clusters, by introducing the permutation function and mimicking classification tasks.

3.2.2 Self-Augmentation Loss:

It attempts to make MLC-Net invariant to small perturbations in input patterns. We perform data augmentation and constrain MLC-Net to provide consistent results for original and augmented data. The self-augmentation loss is given by

$$\mathcal{L}_{\text{sa}}(\mathbf{x}_i; \theta) = \frac{1}{K} \sum_{k=1}^{K} (p(y_i = k|\mathbf{x}_i) - p(y_i = k|\mathbf{x}'_i))^2 \tag{9}$$

where $\mathbf{x}'_i$ denotes an augmented pattern derived from an original pattern $\mathbf{x}_i$.

3.2.3 Self-Ensemble Loss:

We use the self-ensemble loss to prevent unstable parameter fluctuations during the training. We adopt the mean-teacher method [26] using the errors between the predictions of a model and its temporal ensemble;

$$\mathcal{L}_{\text{se}}(\mathbf{x}_i; \theta) = \frac{1}{K} \sum_{k=1}^{K} (p(y_i = k|\mathbf{x}_i; \theta) - p(y_i = k|\mathbf{x}_i; \theta'))^2 \tag{10}$$

where θ' denotes a temporal ensemble. Specifically, let θ_t and θ'_t denote parameters and their temporal ensemble at training step t, respectively. The temporal ensemble is the exponential moving average of successive parameters,

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t \tag{11}$$

where α is a hyper-parameter to control the learning speed.

4 Experiments

We assess the proposed MLC-Net on three datasets: Omniglot [13], MNIST [14], and Mini-ImageNet [30]. We first perform ablation studies to analyze MLC-Net. We then compare MLC-Net with the state-of-the-art clustering algorithms [7, 9, 31, 32]. Although MLC-Net is designed for clustering tasks, it can be used for few-shot classification as well. Thus, we compare MLC-Net also with the few-shot classifiers [4, 17, 20, 22, 24, 30]. We use the classification accuracy and the unsupervised accuracy in (7) to evaluate few-shot learning and clustering algorithms, respectively.

Table 1: Unsupervised accuracies when the attention function in (2) is varied.

Model	Attention function $g(\mathbf{Z})$	5-class	20-class
Linear	$\mathbf{Z}^T\mathbf{W}$	0.200	0.050
ReLU	$h_r(\mathbf{Z}^T\mathbf{W})$	0.200	0.050
Softmax	$h_n(\mathbf{Z}^T\mathbf{W})$	0.991	0.947
MLC-Net	$h_n(h_r(\mathbf{Z}^T\mathbf{W}))$	0.997	0.974

Table 2: Comparison with baseline networks.

Model	5-class	20-class
Baseline I	0.943	0.884
Baseline II	0.991	0.945
MLC-Net	0.997	0.974

4.1 Implementation Details

We adopt the same network structure in [22, 30] for the proposed encoder block. It repeats the convolution block four times, which includes 64 convolution filters of size 3×3, batch normalization [8], ReLU non-linearity, and 2×2 max-pooling. Also, for the centroid block, we employ a single dense layer and a softmax layer to implement the attention function in (2).

For training, we use the NAG optimizer [25] with an initial learning rate of 0.1 and a momentum of 0.9. The training is iterated for 100,000 episodes. We decrease the learning rate by a factor of 0.1 every 25,000 episodes. We construct a training episode to include 200 classes and 20 classes for Omniglot and Mini-ImageNet, respectively. Then, we extract 5 images from each class. For evaluation, we extract features from input patterns by employing the encoder block and use the K-means algorithm [16] to perform the clustering in the embedding space. As done in the conventional clustering algorithms [7, 31], we try 10 different initializations for the K-means clustering and select the result with the minimum K-means cost.

4.2 Ablation Studies on Omniglot

Table 1 compares accuracies when different attention functions are used instead of g in (2). “Linear” and “ReLU” yield poor results since the corresponding networks fail to converge during the training. In contrast, the softmax normalizer h_n enables stable learning, by constraining that pseudo-centroids are located within the convex hull of input features. Combining the rectifier h_r with the normalizer h_n, the proposed MLC-Net provides an effective embedding space.

Table 2 compares the proposed MLC-Net with two baseline networks. First, Baseline I has the architecture of the standard classification network [5, 11]. It removes the centroid block and replaces the prediction block with a single dense layer followed by a softmax layer. As in [30], we train Baseline I to classify an image into one of the training classes. Second, Baseline II substitutes the centroid block with the K-means algorithm during the training. Similarly to MLC-Net, Baseline I and Baseline II are used in the training phase only, while the K-means clustering is performed in the learned embedding spaces, obtained by the respective encoder blocks, in the inference phase.

Table 3: Comparison with the conventional clustering algorithms on the Omniglot.

Model	**5-class**	**20-class**
DEC [31]	0.093	0.067
IMSAT [7]	0.315	0.273
MLC-Net	**0.997**	**0.974**

Table 4: Comparison with the conventional clustering algorithms on the MNIST.

Model	**10-class**
DEC [31]	0.843
DCN [32]	0.830
VaDE [9]	0.945
IMSAT [7]	**0.984**
MLC-Net	0.947

Baseline I performs the worst since its embedding space is less suitable for clustering tasks. In contrast, both Baseline II and MLC-Net use the episodic training to carry out numerous clustering tasks. Thus, they can be generalized for new clustering tasks effectively. However, the lower performance of Baseline II indicates that the joint training of the embedding space and the pseudo-centroid estimator helps to improve the generalization performance, as compared with training the embedding space only. Also, the centroid block is computationally simpler than K-means, since it does not require iterations. MLC-Net processes 29.1 episodes per second (EPS), whereas Baseline II does 4.8 EPS using an NVIDIA 1080-Ti GPU.

4.3 Comparison on Omniglot

Table 3 compares unsupervised accuracies of the proposed MLC-Net and the state-of-the-art clustering algorithms [7, 31] on the Omniglot dataset. MLC-Net outperforms the conventional algorithms substantially. As reported in [7], the conventional algorithms fail to learn suitable embedding networks due to the lack of data in Omniglot, yielding poor results. Even in this challenging case, MLC-Net performs well through the meta learning, which learns the embedding space where K-means works reliably and effectively.

4.4 Comparison on MNIST

In Table 4, we compare 10-class unsupervised accuracies of MLC-Net and the conventional clustering algorithms on the MNIST dataset. Contrary to Omniglot, MNIST is well structured; each class has the same number of patterns that are sufficient to train deep neural networks. Thus, recent clustering algorithms yield promising results on MNIST. However, MLC-Net is better than [9, 31, 32] and comparable to [7]. It is worthy to point out that, while the conventional algorithms are trained for the MNIST dataset, MLC-Net is trained on Omniglot and is directly used on MNIST without any tuning. This confirms excellent generalization capability of MLC-Net.

Table 5: Comparison with the conventional few-shot classifiers on the Mini-ImageNet.

Model	**5-class**
Matching Network [30]	0.434
Meta-Learner LSTM [20]	0.434
MAML [4]	0.487
Prototypical Network [22]	0.461
Relation Network [24]	0.505
SNAIL [17]	<u>0.550</u>
MLC-Net	0.508
MLC-Net*	**0.563**

4.5 Comparison on Mini-ImageNet

Let us consider the few-shot classification scenario, in which one support pattern is given per class. This support pattern is regarded as the centroid of the class. Then, an input pattern is classified based on the nearest neighbor criterion. Table 5 reports few-shot classification performances of the proposed MLC-Net and the conventional algorithms on the Mini-ImageNet dataset. Although MLC-Net is initially designed for clustering, it outperforms the conventional classifiers, except for SNAIL [17]. However, SNAIL uses a more complicated network to embed input patterns than the other algorithms do. In Table 5, MLCNet* indicates the performance of the proposed algorithm, the encoder block of which is replaced with the same network as SNAIL. MLC-Net* surpasses SNAIL.

5 Conclusions

We proposed MLC-Net to perform reliable clustering, even when few data per class are available in the training phase but no support information about new classes is given in the inference phase. MLC-Net consists of the encoder, centroid, and prediction blocks. We designed the centroid and prediction blocks to emulate numerous clustering tasks during the training so that the encoder block produces the embedding space, in which input patterns in unseen classes are clustered effectively. We trained MLC-Net by minimizing the cross-entropy, self-augmentation, and self-ensemble losses. Experiments showed that MLC-Net provides excellent clustering performance on Omniglot and MNIST. Furthermore, MLC-Net also yields comparable or even better classification results than the conventional few-shot classifiers on Omniglot and Mini-ImageNet.

Acknowledgement

This work was supported by ‘The Cross-Ministry Giga KOREA Project’ grant funded by the Korea government (MSIT) (No. GK18P0200, Development of 4D reconstruction and dynamic deformable action model based hyperrealistic service technology), and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2018R1A2B3003896).

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[2] Deng Cai, Xiaofei He, and Jiawei Han. Locally consistent concept factorization for document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 23(6): 902–913, 2011.

[3] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *ICAIS*, 2011.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[7] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *ICML*, 2017.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[9] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *IJCAI*, 2017.

[10] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Workshops*, 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[12] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

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

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436, 2015.

[16] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[17] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

[18] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

[19] A. Papoulis and S. U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 4th edition, 2002.

[20] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[21] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[22] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[23] G. Strang. *Linear Algebra and Its Applications*. Brooks/Cole, 4th edition, 2006.

[24] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

[25] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[26] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017.

[27] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to Learn*. Springer, 1998.

[28] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

[29] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11 (Dec):3371–3408, 2010.

[30] Oriol Vinyals, Charles Blundell, Tim Lillicrap, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[31] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

[32] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards K-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, 2017.