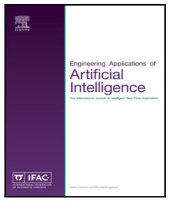
Engineering Applications of Artificial Intelligence 133 (2024) 108612

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Research paper

Transformer Autoencoder for K-means Efficient clustering

Wenhao Wu, Weiwei Wang *, Xixi Jia, Xiangchu Feng

School of Mathematics and Statistics, Xidian University, Xi'an, 710126, China

ARTICLE INFO

Keywords:
Deep clustering
K-means
Transformer

ABSTRACT

As a fundamental unsupervised learning task, clustering has been widely applied in exploratory data analysis in the fields of computer vision, pattern recognition, and data mining. Among existing clustering methods, K-means is the most popular one due to its simplicity and computational efficiency. However, the ubiquitous high dimensionality challenges the effectiveness and the efficiency of the K-means algorithm. Fortunately, the deep neural network provides a powerful resolution for learning low dimensional feature. To optimize the feature learning and the K-means clustering jointly, we present a new deep clustering network called Transformer AutoEncoder for K-means Efficient clustering (TAKE). It consists of two modules: the Transformer AutoEncoder (TAE) for feature learning and the KNet for clustering. The TAE incorporates the transformer structure to learn global features and the contrastive learning mechanism to enhance feature discrimination. The KNet is constructed by unrolling the accelerated projected gradient descent iterations of the relaxed K-means model. The network is trained in two phases: pretraining and clustering. In pretraining, the TAE is optimized by minimizing the cosine similarity-based reconstruction loss, the contrastive loss (CL) and the convex combination loss (CCL). The CCL encourages features of augmented neighbor data to lie in a convex hull, thus K-means friendly. In the clustering phase, the TAE and the KNet are optimized jointly by minimizing the reconstruction loss and the K-means clustering loss. The clustering results are obtained by the forward inference of the KNet. Extended experiments show that our proposed method is highly effective in unsupervised representation learning and clustering.

1. Introduction

Clustering analysis aims to segment a collection of unlabeled data into several clusters so that similar data are grouped into the same clusters and dissimilar data are separated into different clusters. It is a fundamental unsupervised learning task with wide applications in machine learning, computer vision, and data mining (Liu et al., 2022; Xie et al., 2022; Montero et al., 2022; Ezugwu et al., 2022). The classical clustering algorithms, such as the K-means (Lloyd, 1982) and the spectral clustering (Ng et al., 2001), have demonstrated their effectiveness in low dimensional data clustering. The K-means algorithm is especially popular in practice due to its simplicity and computation efficiency. However, these methods degenerate severely in performance when applied to high-dimensional data because the distance-based similarity is unreliable in high-dimensional space. Moreover, the computation cost is raised by the high dimensionality.

In practice, high-dimensional data usually contain redundant information and corruption. To deal with high-dimensional raw data, it is necessary to learn inherent low-dimensional features for the effectiveness and computation efficiency of the classical clustering methods. In recent years, deep neural network (DNN) has shown powerful capacity in representation learning. This inspires a large number of deep clustering methods (Lu et al., 2021; Wang and Jiang, 2021), which use DNN to exploit the latent discriminative low-dimensional features of the high dimensional raw data and then apply a classical clustering algorithm to the low-dimensional features to obtain the final cluster segmentation.

The AutoEncoder (AE) is commonly utilized for unsupervised feature learning. The Stacked AutoEncoder (SAE) (Vincent et al., 2010) consists of multiple layers of autoencoder in which the outputs of each layer are wired to the inputs of the successive layer, and it uses greedy layer-wise training to obtain a suitable initialization parameter. The Denoising AutoEncoder (DAE) (Vincent et al., 2008) aims to improve the robustness of the network by conducting random perturbation to the input. The AE is widely used for image clustering. For example, the Adversarial Multiview Clustering (AMvC) (Wang et al., 2022) uses a multiview encoder to extract latent features from multiview data and then applies the spectral clustering to obtain the final cluster assignment. The Self-supervised Deep Multiview Spectral Clustering

* Corresponding author.
E-mail addresses: whh@stu.xidian.edu.cn (W. Wu), wwwang@mail.xidian.edu.cn (W. Wang), xxjia@xidian.edu.cn (X. Jia), xcfeng@mail.xidian.edu.cn (X. Feng).

https://doi.org/10.1016/j.engappai.2024.108612
Received 14 August 2023; Received in revised form 8 March 2024; Accepted 10 May 2024
Available online 24 May 2024

(SDMvSC) (Zong et al., 2022) also uses spectral clustering, yet it extracts the fused features of multiview data through joint optimization of multiple autoencoders and a feature fusion network. The Autoencoder Constrained Clustering with Adaptive Neighbors (ACC_AN) (Li et al., 2020b) embeds the structured graph learning with adaptive neighbors into the deep autoencoders. In recent years, the self-attention mechanism and the transformer architecture have achieved great success in computer vision. For example, in the Vision Transformer (ViT) (Dosovitskiy et al., 2020), images are divided into non-overlapping patches, and the self-attention mechanism is used to interact between image patches so that the long-range dependence can be captured. In contrast, the convolution adopted in convolutional AE is a local operation and, thus, less effective in capturing the long-range dependencies inherent in input data.

Most existing AE-based deep clustering methods (Cai et al., 2022; Guo et al., 2017; Xie et al., 2016; Yang et al., 2017) adopt a two-phase training strategy. First, the AE is pre-trained by minimizing the reconstruction loss to obtain the feature representation, and the K-means algorithm is applied to the feature to obtain an initial cluster assignment. Then, feature learning and clustering are updated in an end-to-end fashion. The differences between these methods are mainly in the second phase. For example, the Deep Embedding Clustering (DEC) (Xie et al., 2016) uses the K-means clustering results obtained in the first phase to obtain the initial soft cluster assignment. After that, the decoder is dropped, and the encoder is tuned by minimizing the KL divergence of the soft assignment and a target assignment. Improved Deep Embedded Clustering (IDEC) (Guo et al., 2017) improves the clustering phase of the DEC by preserving the decoder and minimizing the KL divergence and the reconstruction loss altogether. The Efficient Deep Embedded Subspace Clustering (EDESC) (Cai et al., 2022) initializes the base proxy for each subspace using the K-means clustering results of the first phase. Then it trains the network by minimizing the KL divergence between the soft and the target cluster assignments, the base proxy's regularization constraint, and the autoencoder's reconstruction loss. The Deep Clustering Network (DCN) (Yang et al., 2017) optimizes the reconstruction loss and the K-means clustering loss jointly in the second phase. To be specific, DCN alternatively updates the representation, the cluster assignment, and the centroids.

The above methods have two major limitations. First, in the pre-training phase, by optimizing only the reconstruction loss, the learned feature may not be well distributed for the K-means clustering. Thus, the initial cluster assignment may not be correct, which leads to sub-optimal final cluster segmentation. An empirical analysis of this observation will be provided in Section 4. Moreover, in the clustering phase of the DCN, the discrete cluster assignment of the K-means makes the optimization of the K-means hard to incorporated in the stochastic gradient descent (SGD) of the total loss.

To overcome the above problems, we propose a deep clustering method called Transformer Autoencoder for K-means Efficient Clustering (TAKE). The network consists of two modules: a transformer-based autoencoder (TAE) and a K-means clustering network (KNet). Compared with the commonly used convolutional AE, the TAE can effectively learn the global features of images by using attention mechanism. To make the features K-means friendly, we pre-train the TAE by minimizing not only the conventional reconstruction loss but also the Contrastive Loss (CL) and the Convex Combination Loss (CCL). The CL can effectively enhance the discrimination of the features. The CCL encourages the feature data of the same cluster to group within a convex hull, thus in favor of K-means clustering. The KNet conducts the K-means clustering and assimilates the cluster centers as parameters. In the training phase, the KNet and the TAE are jointly optimized by using the SGD. Such a training strategy has two significant advantages. First, the final feature data is more fit for the K-means clustering. Second, the whole network is trained for data in batch, not for data in whole. Therefore, our method can be used to segment unseen data or online data.

Table 1
Notation definition.

Symbol	Description
$\boldsymbol{f}_{\theta_e}$	encoder with parameter set θ_e
$\boldsymbol{g}_{\theta_d}$	decoder with parameter set θ_d
$\boldsymbol{h}$	contrastive head
$\boldsymbol{x}_j, \boldsymbol{x}$	original data
$\boldsymbol{x}'$	$\boldsymbol{x}' = \boldsymbol{g}_{\theta_d}(\boldsymbol{f}_{\theta_e}(\boldsymbol{x}))$, reconstruction of $\boldsymbol{x}$
$\boldsymbol{z}_j, \boldsymbol{z}$	$\boldsymbol{z}_j = \boldsymbol{f}_{\theta_e}(\boldsymbol{x}_j)$, $\boldsymbol{z} = \boldsymbol{f}_{\theta_e}(\boldsymbol{x})$, feature of $\boldsymbol{x}_j$, $\boldsymbol{x}$ extracted by the encoder
$\boldsymbol{x}_j^i$	ith augmented data of data $\boldsymbol{x}_j$
$\boldsymbol{z}_j^i$	$\boldsymbol{z}_j^i = \boldsymbol{f}_{\theta_e}(\boldsymbol{x}_j^i)$ the features of $\boldsymbol{x}_j^i$
$\boldsymbol{h}_j^i$	$\boldsymbol{h}_j^i = \boldsymbol{g}(\boldsymbol{h}_j^i)$, features of $\boldsymbol{z}_j^i$ obtained by the contrastive head
B	the batch size
p	the number of samples used in the convex combination loss
c_i	the combination coefficients in convex combinations loss
$\mathcal{L}_r$	reconstruction loss
$\mathcal{L}_{ccl}$	convex combination loss
$\mathcal{L}_{cl}$	contrastive loss
K	number of clusters
$\boldsymbol{m}_i$	cluster center of the ith clusters
$\boldsymbol{M}$	$\boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_K]$, matrix collects the cluster centers
$\boldsymbol{s}_j$	clustering assignment vector corresponding to the data $\boldsymbol{x}_j$
s_{ji}	ith entry of $\boldsymbol{s}_j$
$\lambda, \lambda_1, \lambda_2, \lambda_3$	tuning parameters of the loss functions
$\mathcal{P}_s$	projection operator
α	step size of gradient descent
$\boldsymbol{s}_j^{(t)}$	value of $\boldsymbol{s}_j$ at the tth iteration in the accelerated gradient descent
$\gamma^{(t)}, \eta^{(t)}$	parameters in the tth step of the accelerated projected gradient descent

Our main contribution can be summarized as follow:

- The transformer-based autoencoder TAE is designed for effective feature learning.
- The convex combination loss is defined to enforce convex distribution of features.
- The KNet is designed by unrolling the iterative solution of K-means clustering.
- The end-to-end clustering network TAKE is proposed by joining TAE and KNet.
- Extended experiments demonstrate TAKE outperforms the state-of-the-art baselines.

2. Related work

Firstly, we define the notations used throughout the paper in Table 1.

2.1. Autoencoder-based deep clustering

Autoencoder (AE) provides a popular neural infrastructure for unsupervised learning (Guo et al., 2017; Sheng et al., 2022; Vincent et al., 2010; Xie et al., 2016). Generally, an AE network consists of two modules: an encoder $\boldsymbol{z} = \boldsymbol{f}_{\theta_e}(\boldsymbol{x})$ and a decoder $\boldsymbol{x}' = \boldsymbol{g}_{\theta_d}(\boldsymbol{z}) = \boldsymbol{g}_{\theta_d}(\boldsymbol{f}_{\theta_e}(\boldsymbol{x}))$. The encoder maps the data $\boldsymbol{x}$ into a low dimensional feature representation $\boldsymbol{z}$, and the decoder reverses it. By minimizing a properly defined reconstruction error $\mathcal{L}_r(\boldsymbol{x}, \boldsymbol{x}')$, AE can effectively extract meaningful representation. Fully-connected neural networks and convolutional neural networks are two commonly used structures for AEs.

The DCN (Yang et al., 2017) uses a stacked AE network for feature extraction and takes K-means for clustering. Suppose we have a collection of N raw data $\{\boldsymbol{x}_i \in \mathbb{R}^D, i = 1, 2, \ldots, N\}$ to be segmented into K clusters. let $\boldsymbol{f}_{\theta_e} : \mathbb{R}^D \to \mathbb{R}^d (d \ll D)$ and $\boldsymbol{g}_{\theta_d} : \mathbb{R}^d \to \mathbb{R}^D$ denote the encoder (with parameter sets θ_e) and the decoder (with parameter sets θ_d), respectively. let $\boldsymbol{m}_k \in \mathbb{R}^d$ be the centroid of the kth cluster, $k = 1, 2, \ldots, K$, and $\boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_K] \in \mathbb{R}^{d \times K}$. $\boldsymbol{s}_j \in \{0, 1\}^K$ denotes the one-hot clustering assignment vector corresponding to the

data $\boldsymbol{x}_j$. DCN optimizes the reconstruction loss and K-means loss jointly as follows:

$$\begin{aligned}\min_{\theta_e,\theta_d,\boldsymbol{M},\{\boldsymbol{s}_j\}} \sum_{j=1}^{N} \Big\{ & \mathcal{L}_r\left(\boldsymbol{g}_{\theta_d}\left(\boldsymbol{f}_{\theta_e}\left(\boldsymbol{x}_j\right)\right),\boldsymbol{x}_j\right) + \\ & \frac{\lambda}{2}\left\|\boldsymbol{f}_{\theta_e}\left(\boldsymbol{x}_j\right)-\boldsymbol{M}\boldsymbol{s}_j\right\|_2^2 \Big\} \\ s.t. \boldsymbol{s}_j \in \{0,1\}^K, & \boldsymbol{s}_j^T\mathbf{1}=1 \quad \forall j\end{aligned} \tag{1}$$

In the training process, the network parameters and the K-means clustering are alternatively optimized. Specifically, the parameters θ_e and θ_d are updated with $\boldsymbol{M}$ and $\{\boldsymbol{s}_j\}$ being fixed, then $\boldsymbol{M}$ and $\{\boldsymbol{s}_j\}$ are updated with fixed θ_e, θ_d.

Different from the alternating optimization scheme in DCN, we design a novel deep architecture, KNet, by mapping the accelerated projected gradient descent algorithm (Bubeck et al., 2015) for optimizing the K-means loss into a deep network. In our method, θ_e, θ_d, and $\boldsymbol{M}$ are updated by using the SGD algorithm, and $\{\boldsymbol{s}_j\}$ can be calculated by forward propagation of KNet.

2.2. Transformer

The ViT (Dosovitskiy et al., 2020) is one successful transformer architecture used in computer vision. In ViT, the input image is divided into small non-overlapping patches, and all patches are arranged in a specific order. The patch sequence is input to the linear projection layer to produce a patch embedding sequence, which then enters into the position embedding to infuse the relative position information in each patch of the sequence. An additional learnable class embedding is annexed to the patch embedding to predict the class of the input image after being converted by the Transformer Encoder. Several Transformer-based models (Caron et al., 2021; Carion et al., 2020; Strudel et al., 2021) have been created using this straightforward yet effective backbone architecture, and some have emerged as the standard architectures for a range of vision tasks such as image classification (Dosovitskiy et al., 2020), object detection (Carion et al., 2020), and semantic segmentation (Strudel et al., 2021).

2.3. Unrolling network

Unrolling (Gregor and LeCun, 2010) is an emerging technique for constructing interpretable neural network structures and has been applied to various signal processing, e.g., image deblurring (Li et al., 2020a), super-resolution (Zhang et al., 2020), and clustering (Lin et al., 2021; Tankala et al., 2020). A comprehensive review can be found in Ref. Monga et al. (2021). The Mixture Model Auto-Encoders (MixMate) (Lin et al., 2021) unrolls the Expectation-Maximization (EM) algorithm to construct a clustering network. The K-Deep Simplex (KDS) (Tankala et al., 2020) unrolls the accelerated projective gradient descent algorithm for clustering.

In our model, we present a relaxed K-means model and unroll the accelerated projective gradient descent algorithm for it into our KNet, with the cluster centroids assimilated as parameters.

3. Proposed method

In this work, we propose a deep clustering model called Transformer Autoencoder for K-means Efficient Clustering (TAKE). As illustrated in Fig. 1, our TAKE consists of two modules: a transformer-based autoencoder (TAE) and a K-means clustering network (KNet). The former mainly aims to learn K-means-friendly features, while the latter conducts the K-means clustering, with the cluster centers assimilated as parameters. We first pre-train the TAE by jointly minimizing the reconstruction loss, as well as the contrastive learning loss (CL) and the convex combination loss (CCL). The contrastive learning loss can effectively enhance the discrimination of the features, and the convex combination loss encourages the feature data of the same cluster to group within a convex hull, thus in favor of K-means clustering. The pretraining provides good initialization for the TAE and the KNet. After pretraining, the TAE and the KNet are jointly optimized by using the stochastic gradient descent (SGD) algorithm. The proposed model has at least two advantages. First, the learned feature data are well fit for the K-means clustering. Second, the whole network is trained for data in batch, not for data in whole. Therefore our model can be used for segmenting unseen data or online clustering.

This section will provide a detailed explanation of the proposed method. Firstly, we introduce the overall model framework. Then, we explain the details of the TAE, the Contrastive Head, and the KNet. Finally, we describe the training strategy of the whole model.

3.1. Transformer-based AE (TAE)

3.1.1. Infrastructure of TAE

Inspired by the advantages of AE and ViT, we developed a transformer-based AE, denominated TAE. As illustrated in Fig. 2, our TAE consists of two components: Encoder and Decoder. The encoder architecture is based on the ViT. To maintain the symmetric structure of the TAE, we concatenate all patches (the blue module in the encoder) rather than using an additional patch for downstream tasks as done in ViT. The concatenated features are converted to low-dimensional features by a linear projection layer. The decoder has a similar structure to the encoder but has fewer transformer modules.

To our best knowledge, the self-attention deep subspace clustering (SADSC) (Chen et al., 2021) is the first one exploiting the self-attention mechanism in AE for image clustering. It configures convolutional layers and self-attention layers alternatively in the AE. Our TAE is different in that we use the transformer layer only.

3.1.2. Pretraining loss of TAE

When pretraining the TAE, we use three losses: the reconstruction loss, the contrastive loss, and the convex combination loss.

Let $\{\boldsymbol{x}_j, j = 1,2,\ldots,B\}$ be a mini-batch of input data, we use the following reconstruction loss:

$$\mathcal{L}_r = \frac{1}{B}\sum_{j=1}^{B}[1 - cos(\boldsymbol{x}_j, \boldsymbol{x}'_j)] \tag{2}$$

where $cos(\cdot,\cdot)$ is the cosine similarity.

Note that the previous AE-based clustering methods (Xie et al., 2016; Yang et al., 2017) use the Mean Squared Error (MSE) reconstruction loss, yet the input data are preprocessed by various rescaling operations. In our method, we prefer to take the raw data (without preprocessing) as input, and we use the cosine similarity reconstruction loss in Eq. (2) to avoid the influence of data scales on the results. In Section 4.2.4, we compare the MSE loss and the cosine similarity loss. The clustering results on seven datasets are presented in Tables 6 and 7. On five of the seven datasets, the clustering results favor the cosine similarity loss.

As well known, the k-means clustering performs well on convex datasets (Ahmed et al., 2020). Our method segments the learned features by the K-means clustering. To encourage the learned features of similar data to gather in a convex hull, thus facilitating the k-means clustering, we define the following convex combination loss (CCL) to train the TAE. Let $\{\boldsymbol{x}_j^i\}_{i=1}^p$ is the augmentation of data $\boldsymbol{x}_j$, $\boldsymbol{z}_j^i = \boldsymbol{f}_{\theta_e}(\boldsymbol{x}_j^i)$, $\boldsymbol{z}_j = \boldsymbol{f}_{\theta_e}(\boldsymbol{x}_j)$ be the feature representation of $\boldsymbol{x}_j^i$ and $\boldsymbol{x}_j$, respectively. The CCL is defined as follows:

$$\mathcal{L}_{ccl} = \frac{1}{B}\sum_{j=1}^{B}\| \sum_{i=1}^{p} c_i \boldsymbol{z}_j^i - \boldsymbol{z}_j \|_2^2 \tag{3}$$

where, $\boldsymbol{c} = (c_1, c_2, \ldots, c_p)^T$ denote the coefficients of the convex combination, satisfying $\sum_{i=1}^{p} c_i = 1$, and $c_i \geq 0$. p is the number of augmentations. The CCL encourages the feature representation of similar data (data and its augmentations) to lie in a convex hull. To

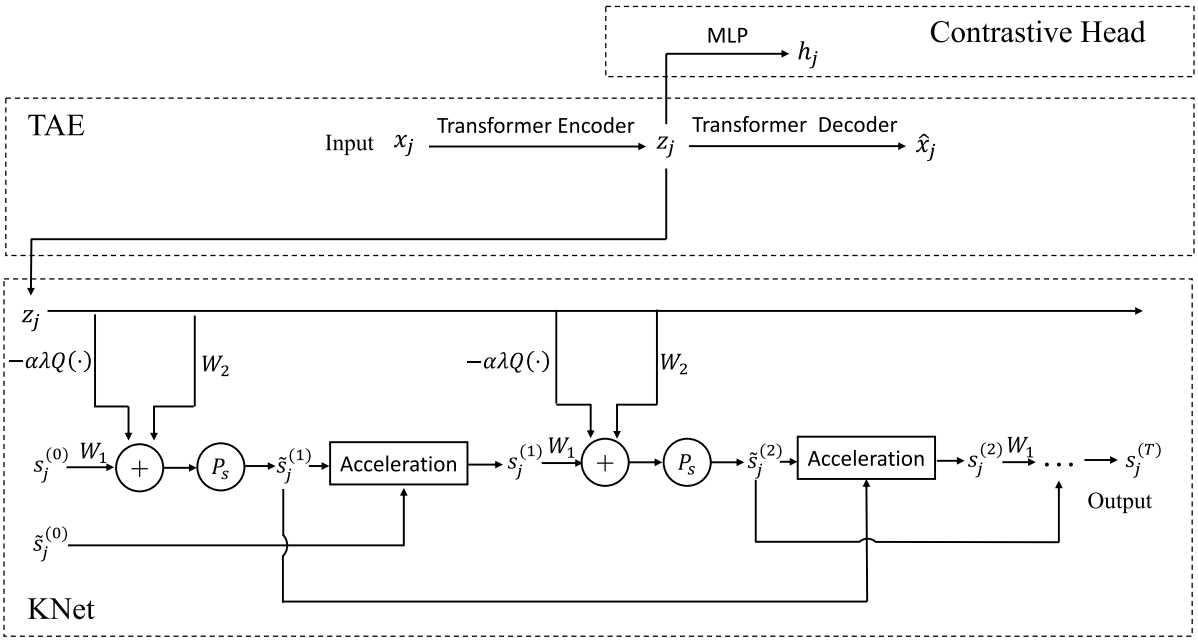


Fig. 1. The overall framework of TAKE: consisting of the TAE and the KNet. In TAE, we also include the contrastive head to extract discriminative features.

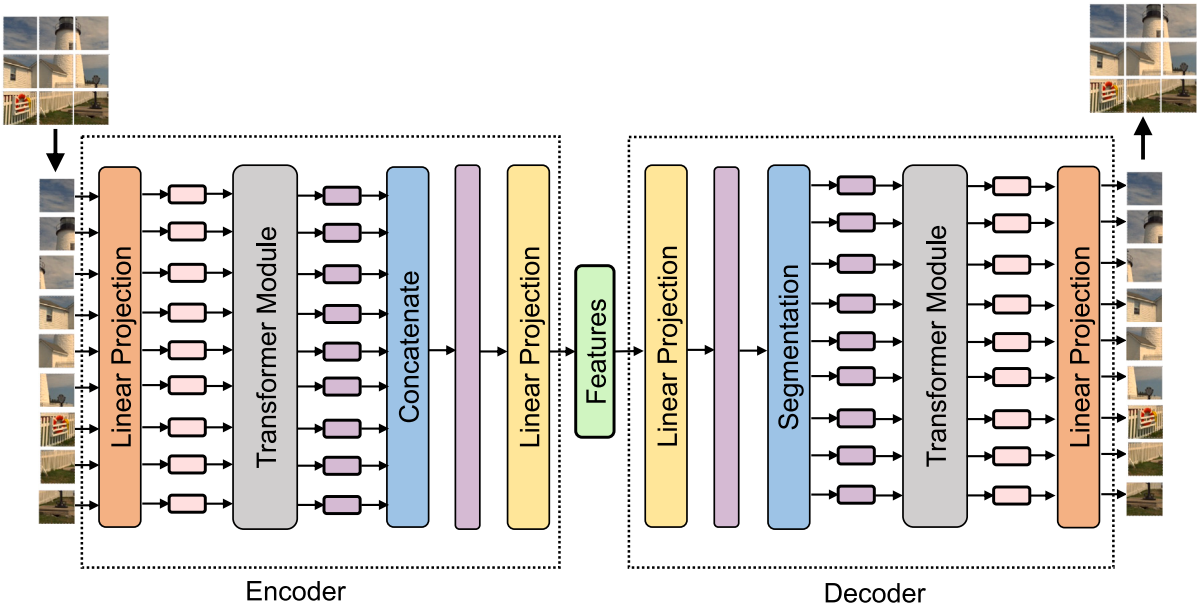


Fig. 2. An overview of our proposed Transformer Autoencoder.

satisfy the constraint, we additionally introduce a set of learnable parameters $\tilde{\boldsymbol{c}} = (\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_p)^T$ and produce the coefficients of the convex combination by using the softmax function, i.e. $\boldsymbol{c} = Softmax(\tilde{\boldsymbol{c}})$.

In pretraining the TAE module, the convex combination loss is computed for each original and augmented data. The ablation experimental results in Tables 10 and 11 confirm that our network trained with the convex combination loss plus the reconstruction loss significantly outperforms that only trained with the reconstruction loss.

According to SimCLR (Chen et al., 2020), the contrastive learning tends to group similar data while departing dissimilar data in feature space. To enhance the learning ability of our TAE, we incorporate the contrastive head (an MLP with one hidden layer) (Chen et al., 2020), which is used to project the feature representation into a new space in which the contrastive loss (CL) is applied. Formally, the contrastive head can be formulated as $\boldsymbol{h} = h(\boldsymbol{z}) = W^{(2)}\sigma(W^{(1)}\boldsymbol{z})$ where σ is ReLU activation function. Let $\boldsymbol{x}_j^1$, $\boldsymbol{x}_j^2$ be two augmentations of $\boldsymbol{x}_j$, $\boldsymbol{z}_j^i = \boldsymbol{f}_{\theta_e}\left(\boldsymbol{x}_j^i\right), i = 1, 2$ be their feature representations, and $\boldsymbol{h}_j^i = g\left(\boldsymbol{z}_j^i\right), i = 1, 2$ be output of the contrastive head, the contrastive loss for a positive pair $(\boldsymbol{x}_j^1, \boldsymbol{x}_j^2)$ is defined as

$$l(\boldsymbol{x}_j^1, \boldsymbol{x}_j^2) = -log \frac{exp(cos(\boldsymbol{h}_j^1, \boldsymbol{h}_j^2)/\tau)}{\sum_{k=1}^{B} exp(cos(\boldsymbol{h}_j^1, \boldsymbol{h}_k^2)/\tau) + \sum_{k=1, k\neq j}^{B} exp(cos(\boldsymbol{h}_j^1, \boldsymbol{h}_k^1)/\tau)} \tag{4}$$

where B is the batch size, τ is a temperature parameter. Note that the numerator of Eq. (4) measures the similarity between $\boldsymbol{x}_j^1$ and $\boldsymbol{x}_j^2$ in feature space (treated as positive samples in contrastive learning), while the denominator quantifies the similarity between $\boldsymbol{x}_j^1$ and other samples in the batch (treated as negative samples in contrastive learning) in

feature space. The total contrastive loss for a batch is

$$\mathcal{L}_{cl} = \frac{1}{2B}\sum_{j=1}^{B}[l(\boldsymbol{x}_j^1, \boldsymbol{x}_j^2) + l(\boldsymbol{x}_j^2, \boldsymbol{x}_j^1)] \tag{5}$$

3.2. The clustering module KNet

Let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_B \in \mathbb{R}^d$ be the low-dimensional representation of a batch of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_B$. To obtain the clustering assignment, the K-means algorithm optimizes the following programming:

$$\begin{aligned} &\mathcal{L} = \sum_{j=1}^{B} \|\boldsymbol{z}_j - \boldsymbol{M}\boldsymbol{s}_j\|_2^2 \\ &s.t. \boldsymbol{s}_j \in \{0,1\}^K, \boldsymbol{s}_j^T\mathbf{1} = 1 \end{aligned} \tag{6}$$

where $\boldsymbol{s}_j \in \{0,1\}^K$ denotes the hard cluster assignment of $\boldsymbol{z}_j$ and each column of $\boldsymbol{M}$ corresponds a cluster centroid. However, due to the discreteness of the hard cluster assignment, it is difficult to integrate the k-means algorithm into a network and take advantage of the end-to-end training. We propose to relax the cluster assignment into soft one $\boldsymbol{s}_j \in [0,1]^K$, and relax the model as follows:

$$\begin{aligned} &\mathcal{L} = \sum_{j=1}^{B}\left[\|\boldsymbol{z}_j - \boldsymbol{M}\boldsymbol{s}_j\|_2^2 + \lambda\sum_{i=1}^{K} s_{ji}\|\boldsymbol{z}_j - \boldsymbol{m}_i\|_2^2 + \frac{1-\lambda}{2}\sum_{i=1}^{K} s_{ji}^2\right] \\ &s.t. \boldsymbol{s}_j \in [0,1]^K, \boldsymbol{s}_j^T\mathbf{1} = 1 \end{aligned} \tag{7}$$

where the s_{ji} is the ith entry of $\boldsymbol{s}_j$, and $\boldsymbol{m}_i$ denotes the ith centroid. The second term in the loss function aims to guarantee the sparsity of the soft cluster assignment. To be specific, if $\boldsymbol{z}_j$ is far away from $\boldsymbol{m}_i$, then s_{ji} tends to vanish. The third term aims to regularize the problem. The tuning parameter $\lambda \in [0,1]$ balances the sparsity and the regularization term.

The objective function in Eq. (7) is differentiable with respect to $\boldsymbol{s}_j$, so we can solve it using the accelerated projected gradient descent method when $\boldsymbol{M}$ is fixed. Specifically, let $\boldsymbol{s}_j^{(0)} = \tilde{\boldsymbol{s}}_j^{(0)} = \mathbf{0}$ and conduct the following iterations:

$$\text{projected gradient descent:} \quad \tilde{\boldsymbol{s}}_j^{(t+1)} = \mathcal{P}_s\left(\boldsymbol{s}_j^{(t)} - \alpha\nabla_{s_j}\mathcal{L}\left(\boldsymbol{s}_j^{(t)}\right)\right) \tag{8}$$

$$\text{Acceleration:} \quad \boldsymbol{s}_j^{(t+1)} = \tilde{\boldsymbol{s}}_j^{(t+1)} + \gamma^{(t)}(\tilde{\boldsymbol{s}}_j^{(t+1)} - \tilde{\boldsymbol{s}}_j^{(t)}) \tag{9}$$

for $0 \le t \le T$. The α is a step size of gradient descent, and the $\gamma^{(t)}$ are given by

$$\eta^{(0)} = 0, \quad \eta^{(t+1)} = \frac{1+\sqrt{1+4\eta^{(t)}}}{2}, \quad \gamma^{(t)} = \frac{\eta^{(t)}-1}{\eta^{(t+1)}}. \tag{10}$$

The operator $\mathcal{P}_s$ projects a vector onto the probability simplex $\{\boldsymbol{s} \in [0,1]^K : \boldsymbol{s}^T\mathbf{1} = 1\}$. We adopt the definition in Tankala et al. (2020), or

$$\mathcal{P}_s(\boldsymbol{s}) = \mathrm{ReLU}(\boldsymbol{s} + b(\boldsymbol{s})) \tag{11}$$

where the function $b : \mathbb{R}^K \to \mathbb{R}$.

Substituting the gradient into Eq. (8), we have:

$$\begin{aligned} \tilde{\boldsymbol{s}}_j^{(t+1)} &= \mathcal{P}_s(\boldsymbol{s}_j^{(t)} - \alpha\nabla_{s_j}\mathcal{L}(\boldsymbol{s}_j^{(t)})) \\ &= \mathcal{P}_s\Big\{\left[(1-\alpha+\alpha\lambda)\boldsymbol{I} - 2\alpha\boldsymbol{M}^T\boldsymbol{M}\right]\boldsymbol{s}_j^{(t)} + \\ &\qquad 2\alpha\boldsymbol{M}^T\boldsymbol{z}_j - \alpha\lambda\sum_{i=1}^{K}\|\boldsymbol{z}_j - \boldsymbol{m}_i\|_2^2\boldsymbol{e}_i\Big\} \end{aligned} \tag{12}$$

where $\boldsymbol{e}_i$ is the K-dimensional unit vector. Let $\boldsymbol{W}_1 = (1-\alpha+\alpha\lambda)\boldsymbol{I} - 2\alpha\boldsymbol{M}^T\boldsymbol{M}$, $\boldsymbol{W}_2 = 2\alpha\boldsymbol{M}^T$, $\boldsymbol{Q}(\boldsymbol{z}) = \sum_{i=1}^{K}\|\boldsymbol{z} - \boldsymbol{m}_i\|_2^2\boldsymbol{e}_i$. The iterations can be unrolled into the KNet in Fig. 1, with the clustering centers $\boldsymbol{M}$ assimilated as parameters. In the KNet module of Fig. 1, $\tilde{\boldsymbol{s}}_j^{(1)}$ is obtained by applying the projection operator $\mathcal{P}_s$ to $\boldsymbol{W}_1\boldsymbol{s}_j^{(0)} + \boldsymbol{W}_2\boldsymbol{z}_j - \alpha\lambda\boldsymbol{Q}(\boldsymbol{z}_j)$ and $\boldsymbol{s}_j^{(1)}$ is calculated according to Eq. (9) (Acceleration module).

In the clustering phase, the KNet is trained jointly with the TAE by using the cosine similarity reconstruction loss and the following k-means loss:

$$\begin{aligned} \mathcal{L}_k = \frac{1}{B}\sum_{j=1}^{B}\Bigg[&\|\boldsymbol{z}_j - \boldsymbol{M}\boldsymbol{s}_j^{(T)}\|_2^2 + \\ &\lambda\sum_{i=1}^{K} s_{ji}^{(T)}\|\boldsymbol{z}_j - \boldsymbol{m}_i\|_2^2 + \frac{1-\lambda}{2}\sum_{i=1}^{K}(s_{ji}^{(T)})^2\Bigg] \end{aligned} \tag{13}$$

3.3. Training strategy

Similar to the Autoencoder-based clustering methods (Xie et al., 2016; Guo et al., 2017; Sheng et al., 2022), the whole network TAKE is trained in two phases: pretraining and clustering. The pretraining phase mainly aims to provide good initialization of the TAE by training it with the aid of the contrastive head to learn discriminative features; The clustering phase trains the whole TAKE (the TAE and the KNet altogether) end-to-end to obtain the cluster assignments of the input data. Fig. 3 illustrates the training strategy. In the pretraining phase, the parameters of the TAE and the contrastive head are updated by minimizing the loss in Eq. (14) (marked as yellow). In the clustering phase, the TAE and the KNet are jointly optimized by end-to-end training with the loss function in Eq. (15) (marked as green).

Pretraining phase. In pretraining TAE, we use the following loss:

$$\mathcal{L}_{pre} = \mathcal{L}_r + \lambda_1\mathcal{L}_{ccl} + \lambda_2\mathcal{L}_{cl} \tag{14}$$

where λ_1 and $\lambda_2 \ge 0$ are tuning parameters to balance the three losses. To be more specific, the TAE is firstly started up by training for 150 epochs using only the reconstruction loss computed on the original data and then updated by training for 50 epochs using the total loss in Eq. (14) on both the original data and the augmented data. The starting-up process plays the dominant role in learning essential features of the input data and tends to stabilize the whole training process. The updating process further improves the discriminability of features.

Clustering phase. Once pre-trained, the TAE outputs K-means-friendly feature representations. We apply the k-means clustering to the feature representation and obtain the cluster centroids $\boldsymbol{M}$, which is then used to initialize the parameters $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and $\boldsymbol{Q}(\cdot)$ of the KNet. Subsequently, the whole TAKE (TAE and KNet altogether) is trained end-to-end by minimizing the following loss with balance parameters λ_3:

$$\mathcal{L}_{clu} = \mathcal{L}_r + \lambda_3\mathcal{L}_k \tag{15}$$

4. Experiments

In this section, we first evaluate the clustering performance of our proposed method on five image datasets (MNIST,[1] Fashion-MNIST,[2] 2MNIST, STL-10[3] and CIFAR-10[4]) and two text dataset (REUTERS-10k[5] and Agnews[6]). We also compare it against related traditional and deep clustering methods. Subsequently, we perform ablation studies to verify the effectiveness of key components in our model and conduct hyper-parameter analysis to show the robustness of the proposed model to the hyper-parameters.

[1] http://yann.lecun.com/exdb/mnist/
[2] https://github.com/zalandoresearch/fashion-mnist
[3] https://cs.stanford.edu/acoates/stl10/
[4] http://www.cs.toronto.edu/kriz/cifar.html
[5] https://keras.io/api/datasets/reuters/
[6] https://paperswithcode.com/dataset/ag-news

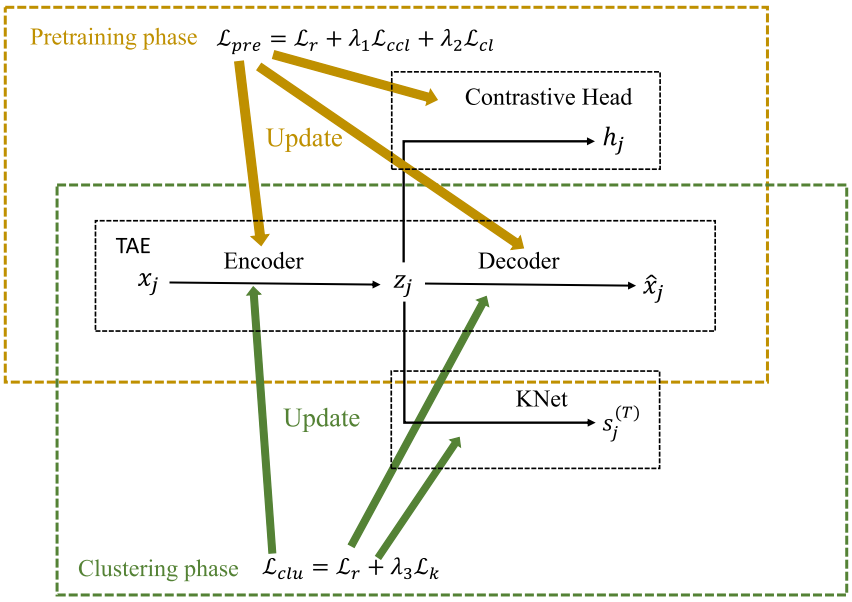


Fig. 3. Illustration of the training strategy. In the pretraining phase, the parameters of the TAE and the contrastive head are updated; In the clustering phase, the TAE and the KNet are jointly optimized by end-to-end training.

4.1. Experimental setting

Benchmark datasets. We evaluate our method on five image datasets and two text datasets. A summary of dataset statistics is shown in Table 2.

- The MNIST dataset comprises 70,000 grayscale images of handwritten digits ranging from 0 to 9, each with a resolution of 28 × 28 pixels. MNIST is partitioned into MNIST-train, and MNIST-test sets, comprising 60,000 and 10,000 images, respectively.
- The Fashion-MNIST dataset consists of 70,000 grayscale images of various fashion products, classified into 10 categories. Each image has 28 × 28 pixels. Fashion-MNIST is also partitioned into Fashion-MNIST-train, and Fashion-MNIST-test sets, comprising 60,000 and 10,000 images, respectively.
- The 2MNIST dataset is a more challenging dataset that is formed by concatenating images from both the MNIST and Fashion-MNIST datasets. This dataset comprises 140,000 images with a resolution of 28 × 28 pixels, categorized into 20 clusters. Note that 2MNIST is not a public dataset. We use it for comparison with the approaches presented in Refs. Sadeghi and Armanfard (2021a), as they employed it in their evaluations.
- The STL-10 dataset comprises 13,000 RGB images of 10 distinct objects, each with a resolution of 96 × 96 pixels. The STL-10 dataset is segregated into two sets, namely, STL-10-train and STL-10-test, which encompass 5,000 and 8,000 images, respectively.
- The CIFAR-10 dataset is composed of 60,000 RGB images depicting 10 distinct objects, with each image having a resolution of 32 × 32 pixels.
- The REUTERS-10k dataset comprises 10,000 English news articles classified into four clusters. The text data is represented using the term frequency-inverse document frequency (tf-idf) feature on the 2000 most frequently occurring word stems.
- AgNews (Rakib et al., 2020) contains 8,000 news titles covering the four categories of world, sport, business, and science.

Evaluation Metrics. We utilize two commonly accepted evaluation metrics: clustering accuracy (ACC) and normalized mutual information (NMI), as defined below.

$$ACC = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{n} \tag{16}$$

$$NMI = \frac{I(\boldsymbol{l};\boldsymbol{c})}{\max\{H(\boldsymbol{l}), H(\boldsymbol{c})\}} \tag{17}$$

where the $\boldsymbol{l} = (l_1, l_2, \ldots, l_n)$ is the ground-truth label of n samples and $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ is the predicted assignment. $H(\cdot)$ and $I(\cdot)$ is entropy and mutual information, respectively. The function m maps all possible one-to-one correspondences between the ground-truth label and predicted assignment. The optimal mapping function can be effectively computed using the Hungarian algorithm (Kuhn, 1955).

Implementation details. To form acceptable input for TAE, we need to preprocess the images or texts. For the grayscale image datasets, including MNIST, Fashion-MNIST, and 2MNIST, each image is represented by pixel values within the range of $[0, 1]$. The patch size and thus the number of the patches depend on the size of images. In Dosovitskiy et al. (2020), the patch sizes are set to 16 × 16 for 224 × 224 images. For the images with size 28 × 28 in MNIST, we set the patch size to 7 × 7, so the number of non-overlapping patches is (28/7)×(28/7) = 16. For the two real-world image datasets (STL-10 and CIFAR-10), we adopt the ResNet50 (He et al., 2016), as done in the previous work (Cai et al., 2022), to extract 2,048-dimensional features. These features are expanded to 2,304 dimensions by padding with zeros and subsequently reshaped into an "image" of dimensions 48 × 48. The reshaped "image" is further divided into 36 non-overlapping 8 × 8 patches. For the Reuters10K, we expand the tf-idf features into 2025-dimensions by padding with zeros and then reshaping them into a 45 × 45 "image". Subsequently, we divide the 45 × 45 "image" into 9 non-overlapping 15 × 15 patches. For AgNews, we use BERT (Devlin et al., 2018) to extract 768-dimensional features, which are expanded into 786-dimensions by padding with zeros and then reshaped into 28 × 28 "image". We divide the 28 × 28 "image" into 16 non-overlapping patches, with each patch consisting of 7 × 7 pixels.

For both grayscale images and text data, the depth of the transformer module and the number of multi-head attention heads are set to 8 in the encoder. For the real images from STL-10 and CIFAR-10, we utilize features extracted by Resnet50 (He et al., 2016), and the depth and the number of attention heads in the encoder module are set to 2. For all datasets, the number of depth and attention heads in the decoder module are set to 2. Empirically, we have observed that reducing the depth of the transformer modules in the encoder module has a negative impact on the clustering performance, whereas reducing the depth

Table 2
Statistics of the benchmark datasets.

Dataset	Modality	Total samples	Number of clusters	Dimension of the data
MNIST	grayscale image	70,000	10	28 × 28
Fashion-MNIST	grayscale image	70,000	10	28 × 28
2MNIST	grayscale image	140,000	20	28 × 28
STL-10	color image	13,000	10	96 × 96 × 3
CIFAR-10	color image	6,000	10	32 × 32 × 3
REUTERS-10k	text	10,000	4	2,000
Agnews	text	8,000	4	768

in the decoder module has little influence on the clustering performance. Considering the trade-off between computational efficiency and clustering accuracy, we have utilized an asymmetric structure in our experiments with raw grayscale images and text data.

For STL-10, CIFAR-10, and Agnews, we set $\lambda_1 = \lambda_2 = 0.1$, and for the grayscale image datasets, we set $\lambda_1 = \lambda_2 = 0.01$. The temperature τ in Eq. (4) is always set to 0.07. For all datasets, the parameter p in Eq. (3) is set to 2. Each original image is cropped to generate two augmented images, which are used in the contrastive loss; Each image is also rotated to produce two augmented images, which are used in the convex combination loss. For the text dataset REUTERS-10k, each data is represented by using the term frequency-inverse document frequency (tf-idf) feature. The original text data are not available, so we cannot do augmentation, and the TAE of our model TAKE is pre-trained only by minimizing the reconstruction loss of the tf-idf feature. For Agnews, we use PPDB synonym replacement[7] to obtain 4 augmented samples, which are used for contrastive loss and convex combination loss. To obtain the initial cluster centroids and assignments of the features output by the pre-trained TAE, we apply the classical K-means algorithm implemented by the scikit-learn[8] package with the default settings.

During the clustering phase, we conduct 50 training epochs using $\mathcal{L}_{clu}$. For MNIST, Fashion-MNIST, and 2MNIST, we set $\lambda_3 = 10^{-3}$; while for the STL-10, CIFAR-10, REUTERS-10k and Agnews, we set $\lambda_3 = 10^{-5}$. Larger λ_3 makes the cluster centers fluctuate sharply, and the training crash. To stabilize the training process, we also use the gradient clipping technique. The step size α in Eq. (8) is set to 10^{-3}. Moreover, for all datasets, the parameter λ in Eq. (7) is set to 0.5. In both phases, we specify the batch size as 256 and the learning rate as 10^{-3}. We use the Adam optimizer to train the network.

Comparison algorithms. We compare the proposed method with three traditional methods: K-means (Lloyd, 1982), large-scale spectral clustering (LSSC) (Chen and Cai, 2011), locality preserving non-negative matrix factorization (LPMF) (Cai et al., 2009); six autoencoder-based deep clustering methods: deep embedding clustering (DEC) (Xie et al., 2016), variational deep embedding (VaDE) (Jiang et al., 2016), improved deep embedding clustering (IDEC) (Guo et al., 2017), deep successive subspace learning (DSSL) (Sadeghi and Armanfard, 2021a), deep embedding clustering based on contractive autoencoder (DECCA) (Diallo et al., 2021), improved deep embedding clustering with deep fuzzy supervision (IDECF) (Sadeghi and Armanfard, 2021b); Two K-means-based deep clustering methods: deep clustering network (DCN) (Yang et al., 2017) and deep K-means (DKM) (Fard et al., 2020); Four state-of-the-art deep clustering methods: mixture model auto-encoders (MixMate) (Lin et al., 2021), self-expressive network (SENet) (Zhang et al., 2021), contrastive deep embedded clustering (CDEC) (Sheng et al., 2022) and efficient deep embedded subspace clustering (EDESC) (Cai et al., 2022).

The source codes of LSSC (Chen and Cai, 2011), LPMF (Cai et al., 2009), DSSL (Sadeghi and Armanfard, 2021a), DECCA (Diallo et al., 2021), and IDECF (Sadeghi and Armanfard, 2021b) are not available, so we directly quote the results in relevant papers. For other methods, the results are obtained by running the codes with default settings provided by the authors.

[7] https://github.com/makcedward/nlpaug
[8] https://scikit-learn.org/stable/

Table 3
The clustering performance on three grayscale image datasets and one text dataset. The best results are in bold font and the second best results are underlined.

Methods	MNIST		Fashion-MNIST		2MNIST		REUTERS-10k	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
K-means	0.5320	0.5000	0.4740	0.5120	0.3231	0.4400	0.5240	0.3120
LSSC	0.7246	0.7060	0.4960	0.4970	0.3977	0.5122	0.4384	–
LPMF	0.4710	0.4520	0.4340	0.4250	0.3468	0.3869	0.5139	–
VaDE	0.9446	0.8760	0.5780	0.6300	0.5660	–	0.7983	0.4160
DEC	0.8430	0.8372	0.5180	0.5463	0.4120	0.5312	0.7217	0.3140
IDEC	0.8813	0.8672	0.5290	0.5570	0.4042	0.5356	0.7564	0.4981
DCN	0.8300	0.8100	0.5122	0.5547	0.4135	0.4689	0.6820	0.4106
DKM	0.8400	0.7960	0.5131	0.5557	0.4175	0.4658	0.6239	0.3310
DSSL	0.9217	0.8421	0.5911	0.6205	0.4380	0.5552	0.7454	–
DECCA	0.9637	0.9074	0.6099	0.6698	–	–	0.8299	**0.6343**
IDECF	0.8717	–	0.5863	–	0.4368	–	–	–
MixMate	0.9573	0.8974	0.5625	0.6286	0.7629	0.8219	0.4482	0.1689
SENet	**0.9679**	0.9169	0.5889	0.6416	0.6421	0.7398	0.7275	0.5832
CDEC	0.9330	0.8580	0.5351	0.5404	0.6757	0.7230	0.7435	0.4963
EDESC	0.9130	0.8620	0.6310	**0.6700**	0.6372	0.6420	0.8250	0.6110
TAKE (our)	0.9654	**0.9275**	**0.6522**	0.6636	**0.7730**	**0.8385**	**0.8433**	0.6217

4.2. Results and discussion

4.2.1. Clustering performance analysis

The metrics for the datasets MNIST, Fashion-MNIST, 2MNIST, and REUTERS-10k are presented in Table 3; While the metrics for the datasets STL-10, CIFAR-10 and Agnews are presented in Table 4. It can be seen that, on 2MNIST, Reuters10K, STL-10, and CIFAR-10, our TAKE obtains significantly better metrics (including ACC and NMI) than the baselines. On the datasets MNIST and Agnews, either ACC or NMI of our TAKE ranks the best, and the other metric ranks the second best.

To compare the capability of related deep networks in learning discriminative feature representation, we apply t-distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten and Hinton, 2008) to visualize the final features generated by VaDE, DEC, IDEC and our TAKE on MNIST-test. The results are shown in Fig. 4, where the points in the same color correspond to data from the same cluster. It can be seen that the feature representations obtained by our TAKE are K-means-friendly, or the points from the same cluster are well accumulated, and the points from different clusters are well separated, except for a small number of challenging samples. In contrast, the feature points extracted by VaDE and DEC exhibit more cross-cluster overlapping, and IDEC exists over-segmentation (the two clusters of green points should have been in one group).

Fig. 5 illustrates some reconstructed images by our TAE trained on 2MNIST. The total reconstruction loss is as low as 0.02956. Both the reconstruction loss and the visual results indicate that our TAE has good reconstruction capability.

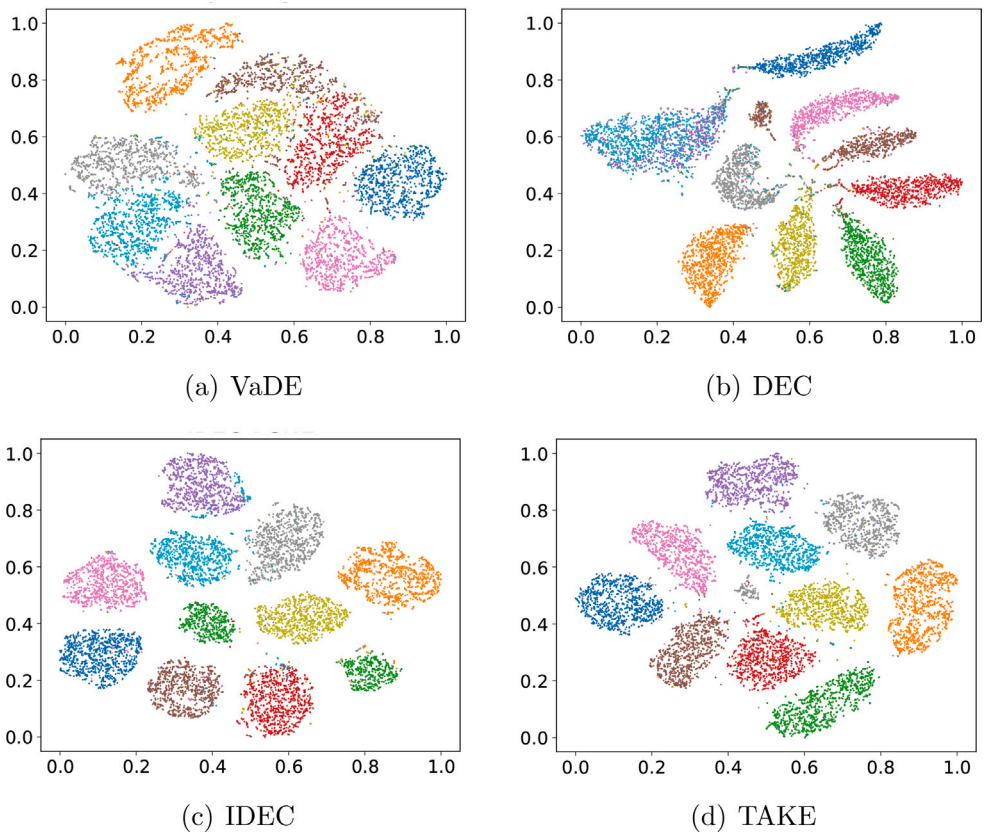


Fig. 4. t-SNE visualization on MNIST-test. The ground truth classes of the data are labeled using different colors.

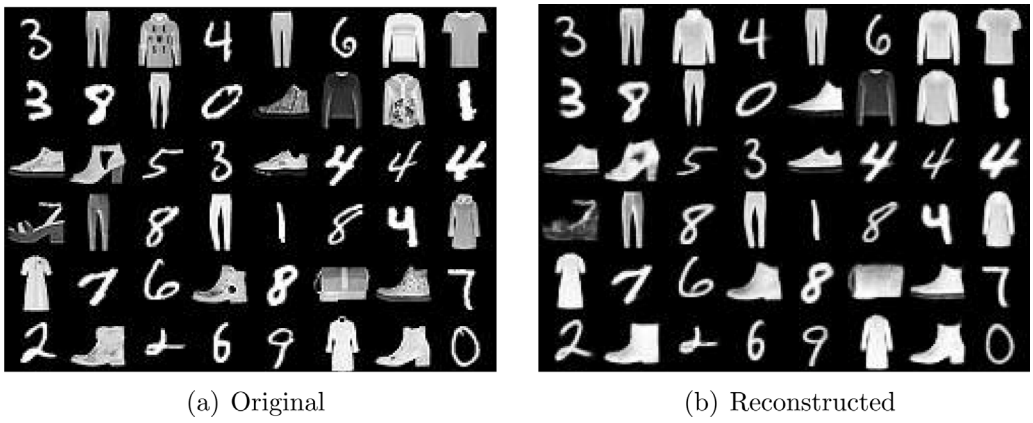


Fig. 5. Reconstruction capability of our TAE. (a) Samples from 2MNIST. (b) Reconstructed images.

4.2.2. Comparison with DCN

Among the existing deep clustering methods, DCN is mostly related to our approach. So we compare DCN and our method in more detail. Both methods use AE for feature representation and K-means for clustering. The difference is that DCN uses stacked AE (SAE), while our method uses transformer-based AE; Moreover, DCN pre-trains the SAE by optimizing only the MSE reconstruction loss, while we use cosine similarity-based reconstruction loss coupled with the convex combination loss and the contrastive loss. We only compare DCN and our TAE for feature representation by pretraining. We randomly select 3 classes of images from Fashion-MNIST and take 300 samples from each class. For visualization, we set the dimension of feature representation by 2. Fig. 6 shows the feature obtained by SAE in DCN and the features obtained by our TAE. It can be seen that the feature representations produced by SAE are far from K-means-friendly. We pre-train the SAE by jointly optimizing the MSE reconstruction loss, CCL, and CL,

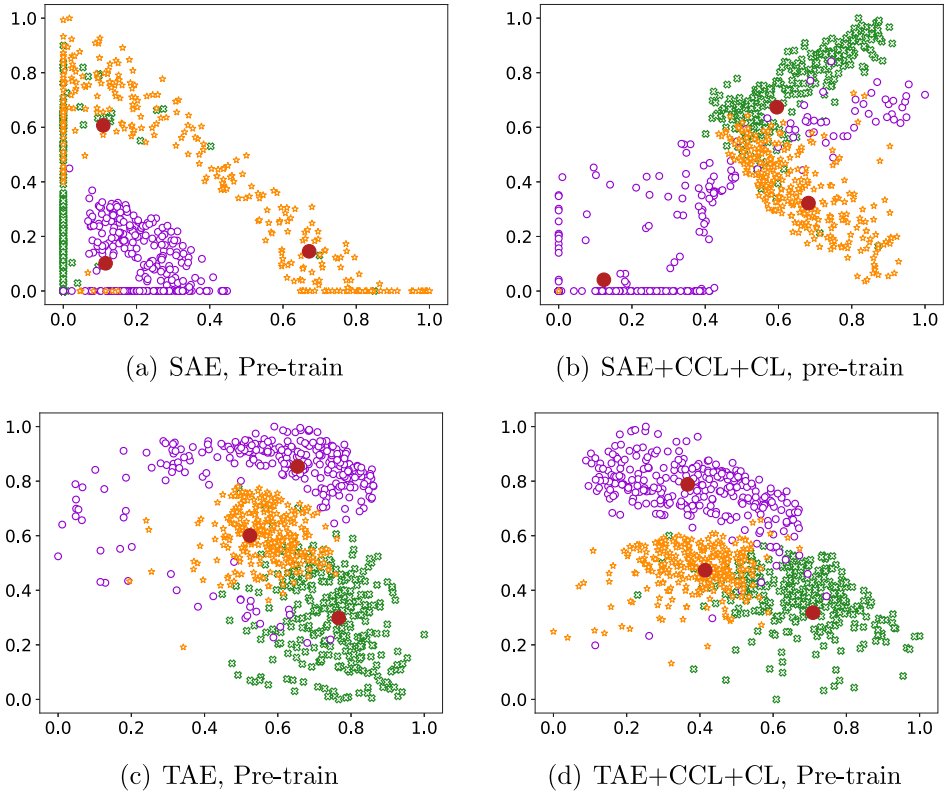


Fig. 6. Comparison of feature representation by DCN and our TAE.

Table 4
The clustering performance on two color image and one text dataset. The best results are in bold font and the second best results are underlined.

Methods	STL-10		CIFAR-10		Agnews	
	ACC	NMI	ACC	NMI	ACC	NMI
K-means	0.1920	0.1250	0.2290	0.8700	0.6478	0.3081
LSSC	0.1875	0.1168	–	–	–	–
LPMF	0.1800	0.0960	–	–	–	–
VaDE	0.2810	0.2000	0.1560	0.3600	0.5472	0.3298
DEC	0.3590	0.2760	0.3010	0.2570	0.5231	**0.4823**
IDEC	0.3253	0.1885	0.3160	0.2730	0.5184	0.2499
DCN	0.3384	0.2412	0.4769	0.4123	0.5000	0.1928
DKM	0.3261	0.2912	0.4920	0.4056	0.5376	0.3349
DSSL	0.6375	0.6403	0.5615	0.4530	–	–
MixMate	0.7739	0.7210	0.6521	0.5911	0.3290	0.2359
SENet	0.8232	0.7541	0.7469	0.6494	0.3721	0.1000
CDEC	0.7328	0.7183	0.5640	0.5778	0.6261	0.3350
EDESC	0.7130	0.6110	0.6960	0.5910	0.6955	0.3416
Our	**0.8765**	**0.7850**	**0.7523**	**0.6543**	**0.7632**	0.4451

Table 5
Generalization performance of TAKE on Fashion-MNIST and STL-10.

Train Data	Fashion-MNIST-test		STL-10-test	
	ACC	NMI	ACC	NMI
500	0.5533	0.5788	0.8556	0.7545
1000	0.5670	0.5873	0.8591	0.7559
2000	0.5616	0.6217	0.8645	0.7633
4000	0.5959	0.6128	0.8671	0.7669
5000	0.5912	0.6231	0.8536	0.7537

Table 6
Comparison of clustering performance between MSE loss and cosine similarity loss on MNIST, Fashion-MNIST, 2MNIST and REUTERS-10k.

	MNIST		Fashion-MNIST		2MNIST		REUTERS-10k	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
MSE	0.9218	0.8632	**0.6638**	**0.6852**	0.7238	0.7729	**0.8522**	**0.6032**
Cosine similarity	**0.9654**	**0.9275**	0.6522	0.6636	**0.7730**	**0.8385**	0.8433	0.6217

Table 7
Comparison of clustering performance between MSE loss and cosine similarity loss on STL-10, CIFAR-10 and Agnews.

	STL-10		CIFAR-10		Agnews	
	ACC	NMI	ACC	NMI	ACC	NMI
MSE	0.8431	0.7679	0.7499	0.6232	0.7396	0.431
Cosine similarity	**0.8765**	**0.7850**	**0.7523**	**0.6543**	**0.7632**	**0.4451**

and the feature distribution shows some improvement, but it remains insufficiently discriminative. On the contrary, the features derived from our TAE exhibit a superior distribution: features belonging to the same cluster are more closely clustered, while features originating from distinct clusters are dispersed.

4.2.3. Generalization performance of TAKE

We evaluate the generalization ability of TAKE to unseen data from Fashion-MNIST and STL-10. Specifically, we randomly select 500,1000,2000,4000,5000 data points from Fashion-MNIST-train and train TAKE for 200 epochs with a batch size of 100. Then, the trained TAKE is used to calculate the cluster assignment of the Fashion-MNIST-test (same as STL-10). Experimental results are reported in Table 5. It can be seen that our method is able to effectively segment unseen data.

4.2.4. Comparison of cosine similarity reconstruction loss and MSE loss

To validate the effectiveness of the cosine similarity-based reconstruction loss in our method, we compare it with the commonly used MSE loss, with other settings in common. The clustering results are shown in Tables 6 and 7. On five of the seven datasets, the clustering

Table 8
The ablation study of the AEs on STL10.

Setting	ACC	NMI	Params
TAKE (our)	**0.8765**	**0.7850**	3.49M
conv-AE	0.3940	0.5140	3.52M
mlp-AE	0.8225	0.7035	3.33M

Table 9
The ablation study of the AEs on Fashion-MNIST.

Setting	ACC	NMI	Params
TAKE (our)	**0.6522**	**0.6636**	3.49M
conv-AE	0.5988	0.6492	3.52M
mlp-AE	0.5384	0.5551	3.33M

results favors in the cosine similarity-based reconstruction loss. Considering this observation, we adopt the cosine similarity reconstruction loss in our method.

4.3. Ablation study

To show the effectiveness of our TAE, CCL, and CL in improving the clustering performance, we conduct some ablation studies on STL-10 and Fashion-MNIST.

To verify the effectiveness of our TAE, we compare three AE structures: TAE, convolutional AE (conv-AE), and fully-connected AE (mlp-AE). To ensure a fair comparison, we set the number of parameters of the latter two to be almost the same as our TAE. The mlp-AE's encoder has five hidden layers with 500, 500, 2000, and 10 neurons, respectively. Except for input and output, all internal layers are activated by the ReLU nonlinearity function (Glorot et al., 2011). The decoder and encoder have a symmetrical structure. This structure is commonly used in clustering (Xie et al., 2016; Guo et al., 2017). The conv-AE's encoder structure is $Conv_{16} - Conv_{16} - Conv_{32} - Conv_{32} - Fc_{1024} - Fc_{1024}$ where $Conv_k$ denotes a convolutional layer with k filters, kernel size of 3×3 and stride length 1 as default. The Fc_k denotes the fully connected layer with k neurons. BatchNorm (Ioffe and Szegedy, 2015) and GELU (Hendrycks and Gimpel, 2016) nonlinear activation are used after each convolution. The decoder is a mirror of the encoder. In addition, all other settings are kept the same. Tables 8 and 9 shows the results of these three settings on STL-10 and Fashion-MNIST, respectively. Our method achieves the highest ACC and NMI. This suggests that our TAE really facilitates the feature representation and the later clustering.

To analyze the effectiveness of the CCL and the CL, we conducted pretraining of our TAE in four scenarios: optimizing the reconstruction loss, CCL and CL jointly; with one of CCL and CL being removed from the loss; with both CCL and CL being removed from the loss. The clustering phase keeps the same for all the cases. Tables 10 and 11 present the metrics of the clustering results. It can be observed that, by adding the CCL or the CL, the clustering performance can be effectively improved. For example, with the CCL and the CL, the ACC increases from 0.6736 to 0.8765 on STL-10. Especially, the CL seems more effective than the CCL, and integration of $\mathcal{L}_r$, $\mathcal{L}_{ccl}$ and $\mathcal{L}_c$ obtains the best performance among all cases.

In order to confirm the efficacy of our proposed KNet, we conduct ablation experiments on 2MNIST and STL-10. We compare two cases: "Pretraining + K-means" uses the pre-trained TAE to obtain the features and then uses the K-means algorithm to obtain the clustering results; "Pretraining + KNet" means our proposed method. Table 12 shows the results, from which we can observe that, by training the KNet and the TAE jointly, our method can significantly improve the clustering performance, compared with "Pretraining + K-means".

Table 10
The ablation experiments of loss function on STL-10.

Setting	ACC	NMI
$\mathcal{L}_r$	0.6736	0.6845
$\mathcal{L}_r + \mathcal{L}_{cl}$	0.8641	0.6845
$\mathcal{L}_r + \mathcal{L}_{ccl}$	0.8545	0.7634
$\mathcal{L}_r + \mathcal{L}_{ccl} + \mathcal{L}_{cl}$	**0.8765**	**0.7850**

Table 11
The ablation experiments of loss function on Fashion-MNIST.

Setting	ACC	NMI
$\mathcal{L}_r$	0.5932	0.6192
$\mathcal{L}_r + \mathcal{L}_{cl}$	0.6322	0.6412
$\mathcal{L}_r + \mathcal{L}_{ccl}$	0.6234	0.6477
$\mathcal{L}_r + \mathcal{L}_{ccl} + \mathcal{L}_{cl}$	**0.6522**	**0.6636**

Table 12
ACC/NMI on the 2MNIST and STL-10.

Datasets	Pretraining + K-means	Pretraining + KNet
2MNIST	0.7248/0.8034	**0.7730/0.8385**
STL-10	0.8431/0.7432	**0.8765/0.7850**

4.4. Hyperparameter settings and loss attenuation

In this section, we investigate how the pretraining hyperparameters λ_1, λ_2, the clustering hyperparameters λ_3, and the number of transformer depths and heads used in the encoder influence the final clustering results.

Fig. 7(a) 7(b) and 8(a) 8(b) demonstrate the variation of the metrics ACC and NMI with the hyperparameters λ_1 and λ_2, on the datasets STL-10 and MNIST, respectively. The hyperparameters range from 10^{-4} to 10^0 while all other hyperparameters are fixed as described in Section 4.1. On both datasets, small values of λ_1 and λ_2 lead to good metrics. For STL-10, the clustering results have little fluctuation when the hyperparameters λ_1 and λ_2 increase. We set $\lambda_1 = \lambda_2 = 0.1$ on color images (including STL-10) and the text data, which produces promising clustering performance. For the MNIST, the clustering metrics fluctuates stronger when λ_1 and λ_2 increase. Consequently, we set smaller $\lambda_1 = \lambda_2 = 0.01$ for MNIST as well as Fashion-MNIST and 2MNIST.

Fig. 7(c) 7(d) and 8(c) 8(d) illustrates the impact of the number of the depth and the heads in TAE, on STL-10 and MNIST, respectively. We consider the number of depths and the heads in the range of $\{2, 4, 6, 8, 10\}$. It can be observed that, within this range, the clustering performance on STL-10 remains high and stable. However, for MNIST, a large number of depths and heads lead to good clustering performance. This inspires us to set the number of depths and the number of heads to 2 for STL-10 and 8 for MNIST.

Fig. 9 depicts the impact of λ_3 on ACC and NMI. It is evident that the clustering results are fairly consistent when λ_3 is below 0.01. However, as λ_3 exceeds this value, the clustering performance deteriorates. We determined the optimal value for the parameter λ_3 to be 10^{-5} for STL-10, CIFAR-10, and REUTERS-10k and 10^{-3} for MNIST, Fashion-MNIST, and 2MNIST, resulting in the attainment of the most favorable outcomes.

Figs. 10(a) and 11(a) illustrate the evolution of the reconstruction loss over the first 200 epochs of pretraining and the last 50 epochs of clustering for STL-10 and MNIST, respectively. The plots demonstrate that the reconstruction loss converges effectively. Furthermore, Figs. 10(b) and 11(b) display the change of $\mathcal{L}_{ccl}$ over the course of pretraining, while Figs. 10(c) and 11(c) depict the decay of $\mathcal{L}_{cl}$ during pretraining, both for the STL-10 and MNIST datasets.

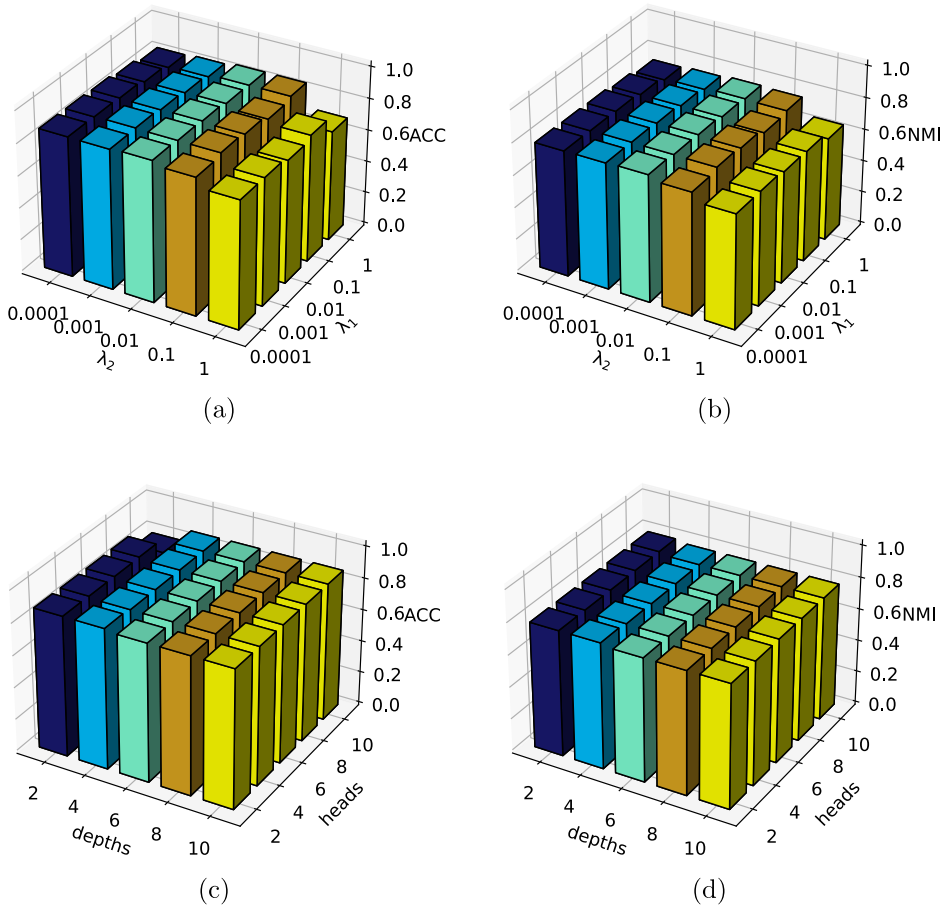


Fig. 7. Parameter sensitivity study on STL10. (a)(b) The effect of λ_1 and λ_2 on the metrics ACC and NMI. (c)(d) The effect of the depths and heads of TAE on the metrics ACC and NMI.

Finally, Figs. 10(d) and 11(d) exhibit the evolution of $\mathcal{L}_k$ during clustering phase. The experimental results demonstrate the efficacy of our proposed loss function in effectively guiding the learning process of the model.

5. Conclusion

In this paper, we propose an end-to-end deep clustering network called Transformer Autoencoder for K-means Efficient Clustering (TAKE). It consists of two modules: Transformer Autoencoder (TAE) and KNet. In order to extract discriminative features that are fit for K-means clustering, the TAE is pre-trained by using the proposed reconstruction loss, convex combination loss, and contrastive loss. In the clustering phase, the KNet and the TAE are jointly optimized to produce K-means-friendly features and cluster assignments. Extended experiments show the advantageous performance of our proposed model over the baselines. In particular, our approach achieves notable results for color images. For example, on the dataset STL-10, our method obtains ACC 87.65% and NMI 78.50%, which are 5.33% and 3.09% higher than the second-best indexes. Another advantage is that our network can be used for unseen data or online data clustering.

While our proposed method exhibits notable strengths, it has some limitations. Firstly, the number of clusters K needs to be known in advance, which is also a common limitation of many clustering methods. Additionally, the transformer-based autoencoder consumes more GPU memory and demands longer running times, potentially limiting its application in real-time or resource-constrained scenarios. How to solve these questions opens venues for future research and practical applications.

CRediT authorship contribution statement

Wenhao Wu: Conceptualization, Methodology. **Weiwei Wang:** Conceptualization, Funding acquisition, Methodology, Supervision. **Xixi Jia:** Conceptualization, Writing – review & editing. **Xiangchu Feng:** Conceptualization, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to thank the editors and the anonymous reviewers for their constructive comments and suggestions. This paper is supported by the National Natural Science Foundation of China (Grant Nos. 61972264, 62072312, 62372359, 62372302), Natural Science Foundation of Guangdong Province (Grant No. 2019A1515010894), Natural Science Foundation of Shenzhen (Grant No. 2020080716523 5002) and Natural Science Basic Research Program of Shaanxi (Program No. 2024JC-YBMS-527).

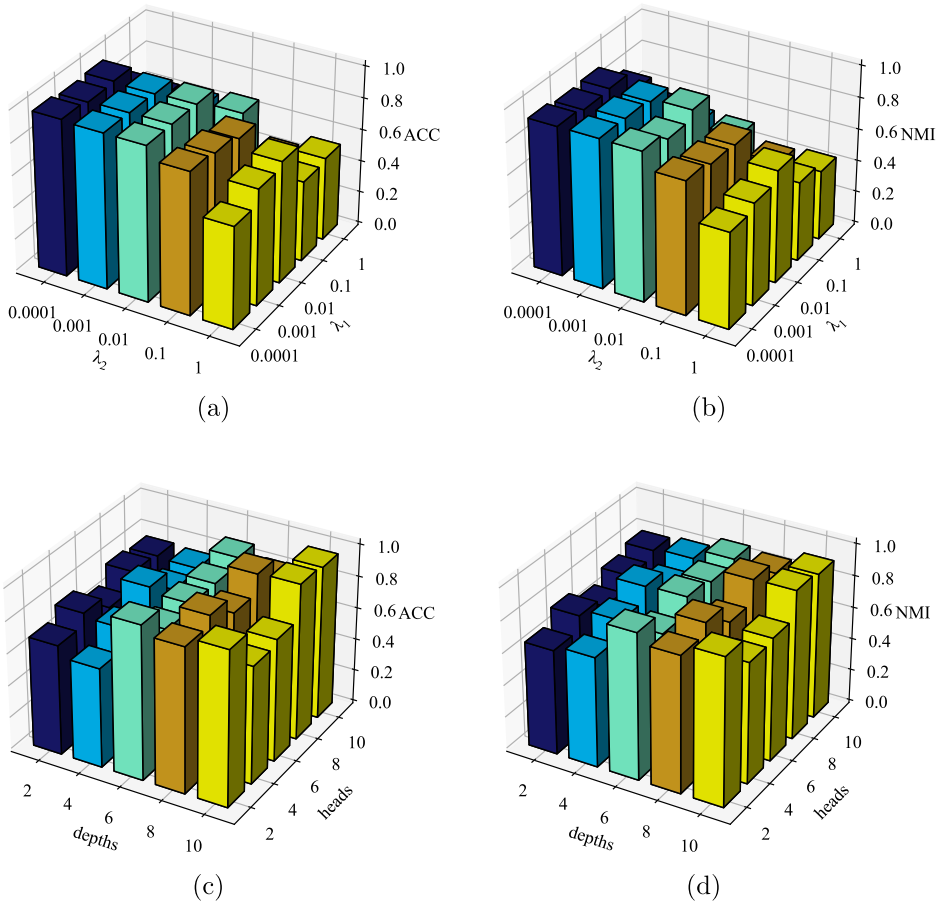


Fig. 8. Parameter sensitivity demonstration on MNIST. (a)(b) The λ_1 and λ_2 effect the clustering ACC and NMI. (c)(d) The depths and heads effect the clustering ACC and NMI.

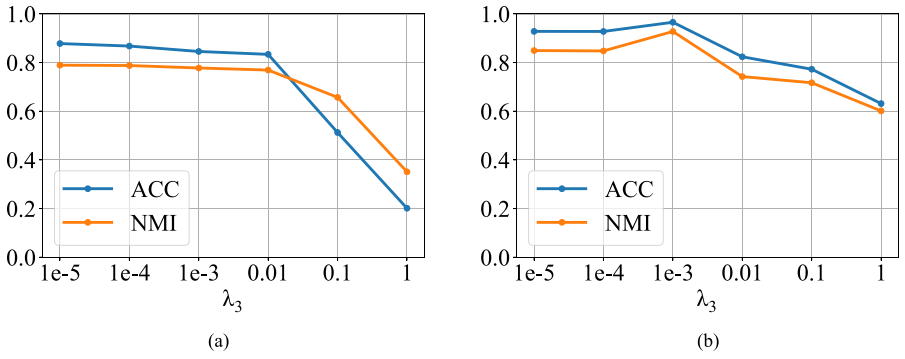


Fig. 9. Influence of λ_3 on the clustering results of (a) STL-10. (b) MNIST.

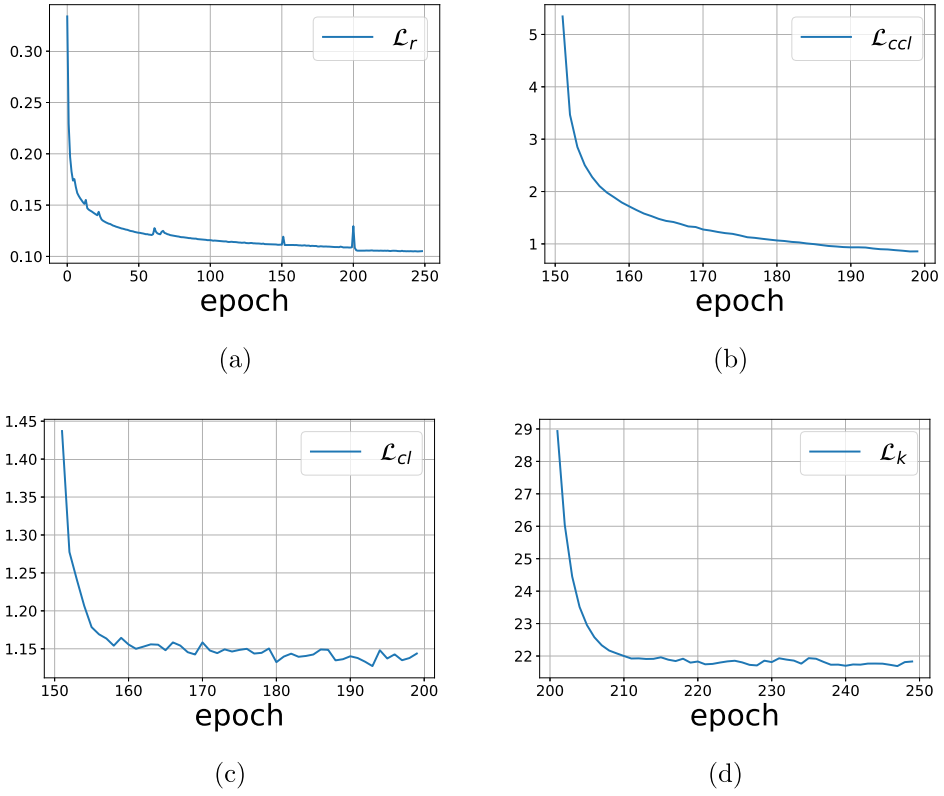


Fig. 10. Evolution of the loss function during training on STL-10.

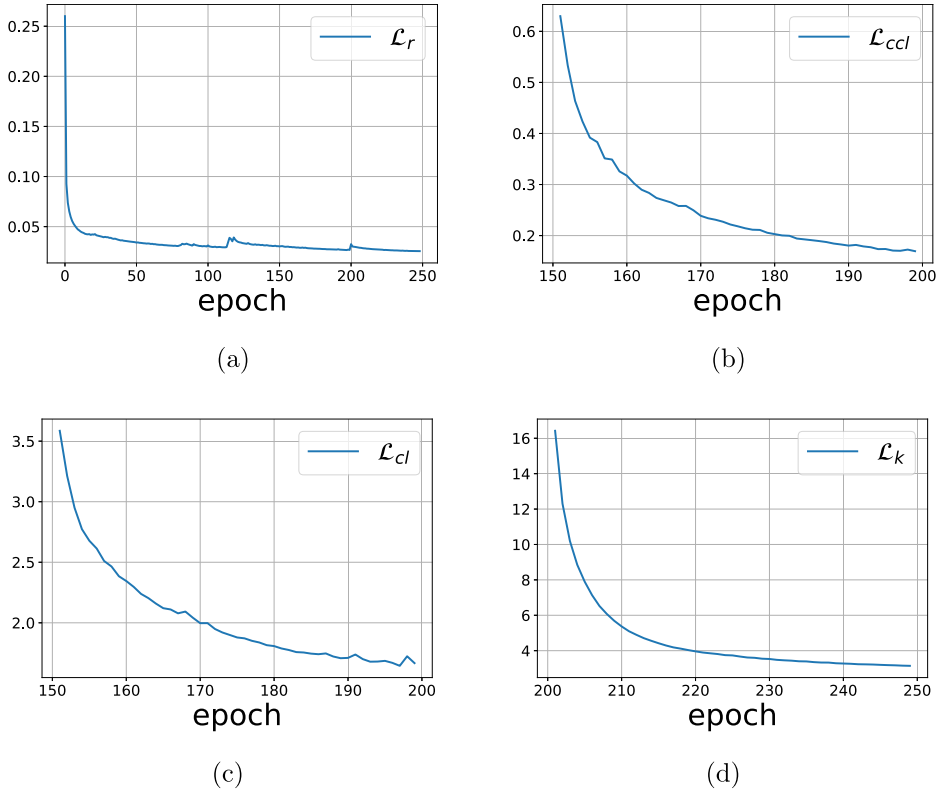


Fig. 11. Evolution of the loss function during training on MNIST.

References

Ahmed, M., Seraj, R., Islam, S.M.S., 2020. The k-means algorithm: A comprehensive survey and performance evaluation. Electronics 9 (8), 1295.

Bubeck, S., et al., 2015. Convex optimization: Algorithms and complexity. Found. Trends® Mach. Learn. 8 (3–4), 231–357.

Cai, J., Fan, J., Guo, W., Wang, S., Zhang, Y., Zhang, Z., 2022. Efficient deep embedded subspace clustering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1–10.

Cai, D., He, X., Wang, X., Bao, H., Han, J., 2009. Locality preserving nonnegative matrix factorization. In: Twenty-First International Joint Conference on Artificial Intelligence. pp. 1010–1015.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A., 2021. Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9650–9660.

Chen, X., Cai, D., 2011. Large scale spectral clustering with landmark-based representation. In: Twenty-Fifth AAAI Conference on Artificial Intelligence. pp. 313–318.

Chen, Z., Ding, S., Hou, H., 2021. A novel self-attention deep subspace clustering. Int. J. Mach. Learn. Cybern. 12 (8), 2377–2387.

Chen, T., Kornblith, S., Norouzi, M., Hinton, G., 2020. A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning. pp. 1597–1607.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Diallo, B., Hu, J., Li, T., Khan, G.A., Liang, X., Zhao, Y., 2021. Deep embedding clustering based on contractive autoencoder. Neurocomputing 433, 96–107.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Ezugwu, A.E., Ikotun, A.M., Oyelade, O.O., Abualigah, L., Agushaka, J.O., Eke, C.I., Akinyelu, A.A., 2022. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Eng. Appl. Artif. Intell. 110, 104743.

Fard, M.M., Thonet, T., Gaussier, E., 2020. Deep k-means: Jointly clustering with k-means and learning representations. Pattern Recognit. Lett. 138, 185–192.

Glorot, X., Bordes, A., Bengio, Y., 2011. Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 315–323.

Gregor, K., LeCun, Y., 2010. Learning fast approximations of sparse coding. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. pp. 399–406.

Guo, X., Gao, L., Liu, X., Yin, J., 2017. Improved deep embedded clustering with local structure preservation. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. pp. 1753–1759.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Hendrycks, D., Gimpel, K., 2016. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., Zhou, H., 2016. Variational deep embedding: An unsupervised and generative approach to clustering. arXiv preprint arXiv:1611.05148.

Kuhn, H.W., 1955. The hungarian method for the assignment problem. Naval Res. Logist. Q. 2 (1–2), 83–97.

Li, Y., Tofighi, M., Geng, J., Monga, V., Eldar, Y.C., 2020a. Efficient and interpretable deep blind image deblurring via algorithm unrolling. IEEE Trans. Comput. Imaging 6, 666–681.

Li, X., Zhang, R., Wang, Q., Zhang, H., 2020b. Autoencoder constrained clustering with adaptive neighbors. IEEE Trans. Neural Netw. Learn. Syst. 32 (1), 443–449.

Lin, A., Song, A.H., Ba, D., 2021. Mixture model auto-encoders: Deep clustering through dictionary learning. arXiv preprint arXiv:2110.04683.

Liu, W., Liu, L., Zhang, Y., Wang, H., Feng, L., 2022. Adaptive multi-view multiple-means clustering via subspace reconstruction. Eng. Appl. Artif. Intell. 114, 104986.

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Trans. Inform. Theory 28 (2), 129–137.

Lu, R.-k., Liu, J.-w., Zuo, X., 2021. Attentive multi-view deep subspace clustering net. Neurocomputing 435, 186–196.

Monga, V., Li, Y., Eldar, Y.C., 2021. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. IEEE Signal Process. Mag. 38 (2), 18–44.

Montero, D., Aginako, N., Sierra, B., Nieto, M., 2022. Efficient large-scale face clustering using an online mixture of Gaussians. Eng. Appl. Artif. Intell. 114, 105079.

Ng, A., Jordan, M., Weiss, Y., 2001. On spectral clustering: Analysis and an algorithm. Adv. Neural Inf. Process. Syst. 14.

Rakib, M.R.H., Zeh, N., Jankowska, M., Milios, E., 2020. Enhancement of short text clustering by iterative classification. In: 25th International Conference on Applications of Natural Language To Information Systems. pp. 105–117.

Sadeghi, M., Armanfard, N., 2021a. Deep successive subspace learning for data clustering. In: 2021 International Joint Conference on Neural Networks. IJCNN, IEEE, pp. 1–8.

Sadeghi, M., Armanfard, N., 2021b. IDECF: Improved deep embedding clustering with deep fuzzy supervision. In: 2021 IEEE International Conference on Image Processing. ICIP, pp. 1009–1013.

Sheng, G., Wang, Q., Pei, C., Gao, Q., 2022. Contrastive deep embedded clustering. Neurocomputing 514, 13–20.

Strudel, R., Garcia, R., Laptev, I., Schmid, C., 2021. Segmenter: Transformer for semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7262–7272.

Tankala, P., Tasissa, A., Murphy, J.M., Ba, D., 2020. K-deep simplex: Deep manifold learning via local dictionaries. arXiv preprint arXiv:2012.02134.

Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9 (11).

Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A., 2008. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning. pp. 1096–1103.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., Bottou, L., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. 11 (12).

Wang, J., Jiang, J., 2021. Unsupervised deep clustering via adaptive GMM modeling and optimization. Neurocomputing 433, 199–211.

Wang, Q., Tao, Z., Xia, W., Gao, Q., Cao, X., Jiao, L., 2022. Adversarial multiview clustering networks with adaptive fusion. IEEE Trans. Neural Netw. Learn. Syst..

Xie, J., Girshick, R., Farhadi, A., 2016. Unsupervised deep embedding for clustering analysis. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48. pp. 478–487.

Xie, K., Huang, L., Wei, Z., Zhang, W., Qin, Q., 2022. WCATN: Unsupervised deep learning to classify weather conditions from outdoor images. Eng. Appl. Artif. Intell. 113, 104928.

Yang, B., Fu, X., Sidiropoulos, N.D., Hong, M., 2017. Towards K-means-friendly spaces: Simultaneous deep learning and clustering. In: International Conference on Machine Learning. pp. 3861–3870.

Zhang, K., Gool, L.V., Timofte, R., 2020. Deep unfolding network for image super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3217–3226.

Zhang, S., You, C., Vidal, R., Li, C.-G., 2021. Learning a self-expressive network for subspace clustering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12393–12403.

Zong, L., Miao, F., Zhang, X., Liang, W., Xu, B., 2022. Self-supervised deep multiview spectral clustering. IEEE Trans. Neural Netw. Learn. Syst..