Unsupervised Feature Learning by Autoencoder and Prototypical Contrastive Learning for Hyperspectral Classification

Zeyu Cao[1], Xiaorun Li[1], Liaoying Zhao[2]

Unsupervised learning methods for feature extraction are becoming more and more popular. We combine the popular contrastive learning method (prototypical contrastive learning) and the classic representation learning method (autoencoder) to design an unsupervised feature learning network for hyperspectral classification. Experiments have proved that our two proposed autoencoder networks have good feature learning capabilities by themselves, and the contrastive learning network we designed can better combine the features of the two to learn more representative features. As a result, our method surpasses other comparison methods in the hyperspectral classification experiments, including some supervised methods. Moreover, our method maintains a fast feature extraction speed than baseline methods. In addition, our method reduces the requirements for huge computing resources, separates feature extraction and contrastive learning, and allows more researchers to conduct research and experiments on unsupervised contrastive learning.

***Index Terms*—unsupervised learning, autoencoder, contrastive learning, hyperspectral classification**

I. Introduction

With the rapid development of deep learning, much progress has been made in the hyperspectral classification field. Recent years, many great supervised deep learning methods for hyperspectral classification were proposed, such as one-dimensional convolutional neural networks(1DCNN)[1], three-dimensional convolutional neural networks(3DCNN)[2], deep recurrent neural networks (RNN)[3], deep feature fusion network(DFFN)[4], spectral-spatial residual network(SSRN)[5]. Nowadays, it is widely admitted that the fusion of spatial information and spectral information is the best strategy to extract features for hyperspectral classification. Furthermore, deep learning models showed great power in finding the two kinds of information. With enough labeled samples, many supervised learning methods can effectively learn the spatial and spectral information simultaneously(e.g., SSRN) and update the models' parameters, thus getting good hyperspectral classification results. However, there are some disadvantages to supervised learning methods. Firstly, severe data dependence is the notorious disadvantage of supervised deep learning. Without enough labeled samples, many supervised deep learning methods can not work or only get bad results. Secondly, many supervised deep learning methods only rely on the labels to supervise the extraction of features. As a result, with the increase of labeled data, the models need to be retrained entirely, which seems expensive in the practical circumstances.

Researchers are also paying their attention to unsupervised learning methods to avoid the disadvantages of supervised deep learning. Unsupervised learning methods are mostly feature extraction algorithms, with a suitable feature extractor, we can use a simple classifier to get good classification results. To some extent, unsupervised learning methods are more convenient to be applied to practice. Speak of unsupervised learning, there are some classic methods widely applied to all kinds of fields, such as principal component analysis(PCA)[6] and independent component analysis(ICA)[7]. These traditional algorithms remain efficient in the pre-process or feature extraction field.

Generally speaking, unsupervised learning aims to dig out information from data other than labels; there are two major ways to achieve the objective: representative learning and discriminative learning. There are two famous representative learning models: autoencoder[8] and generative adversarial network(GAN)[9]. The two methods both use the self-similarity of data and then construct a model that can map the original data to some certain distributions. With the generated distributions, we can get distributions similar to the distributions of real samples. As a result, we can get forgery samples that are similar to real samples or get good feature extractors. Inspired by autoencoder, stacked sparse autoencoder(SSAE)[10] used autoencoder for sparse spectral feature learning and multiscale spatial feature learning, respectively and then fused these two kinds of feature for hyperspectral classification. Furthermore, GAN[11] also proved to be useful in extracting spectral-spatial features for hyperspectral classification. Except for these, 3D convolutional autoencoder(3DCAE)[12], semi-supervised GAN[13], GAN-Assisted CapsNet(TripleGAN)[14] and many other papers also originated from the representative learning. Generally, many researchers have acknowledged the utility of the representative learning methods in the remote sensing field.

On the other hand, discriminative learning is also an original idea in exploring the information of data. A typical way of using discriminative learning is the contrastive learning method. Unlike autoencoder and GAN, the contrastive learning method does not care about the details of the data. It cares about the difference between the different samples. In other words, contrastive learning aims to discriminate different samples and does not aim to find samples' exact distributions. By comparing different samples, a discriminator is trained and can

Manuscript received December 1, 2012; revised August 26, 2015. Corresponding author: Xiaorun Li (email:lxr@zju.edu.cn). [1]Zhejiang University, College of Electrical Engineering, 866 Yuhangtang Rd, Hangzhou, China, 310058,[2]HangZhou Dianzi University, China Institute of Computer Application Technology, Xiasha Higher Education Zone, Hangzhou, China, 310018

be used for classification or other tasks. Because by finding the difference between two randomly selected samples, somehow the discriminator has learnt to extract features that are helpful for downstream tasks. Contrastive learning can be applied in supervised as well as unsupervised learning. For supervised contrastive learning, contrastive loss[15] is proposed to optimize the discriminator. In the hyperspectral classification field, siamese convolutional neural networks[16] adopted the original siamese network for scene classification. The end-to-end siamese convolutional neural network[17] proposed a 1D siamese CNN network for hyperspectral classification. Dual-path siamese convolutional neural network[18] proposed a dual-path siamese network composed of 2D and 1D CNN for hyperspectral classification. Supervised learning has drawn many researchers' attention.

For unsupervised contrastive learning, there are no label information can be utilized at all. So a paradigm is widely used for unsupervised learning. Because contrastive learning requires the contrast between two samples, we use two kinds of data augmentation methods to transform a single sample into two transformed samples. In this way, we create a kind of label: the two samples originate from the same sample. Then with a proper structure and a well-designed objective function, we can train an encoder that can map the samples to latent encoding space, and the features from the encoding space can well represent the original samples. When the encoder acts as a feature extractor, many downstream tasks can be done. Much work has been done in the computer vision field, such as momentum contrast[19] and a simple framework for contrastive learning[20]. They proposed some original structures for unsupervised contrastive learning and got promising results in many visual datasets. Furthermore, prototypical contrastive learning(PCL)[21] combined cluster algorithm with contrastive learning, got better performances than other methods, even outperformed some supervised methods. Generally, unsupervised contrastive learning has shown its great power in the computer vision field.

Considering the similarity between hyperspectral images and normal images, we think it is possible to transfer the unsupervised contrastive learning methods to hyperspectral classification. However, as far as we know, few related methods have been reported in the hyperspectral classification field. There are some problems to address to transfer the unsupervised contrastive learning to the process of hyperspectral image. Firstly, the computer vision field's typical data augmentation methods seem not reasonable in the hyperspectral classification field. For example, color distortion to the normal images will not change objects' spatial features, so it is an acceptable transformation. When distorting hyperspectral images' spectral value, the spectral information is destroyed, and the spectral information is essential for hyperspectral classification, so this is not an acceptable transformation for hyperspectral images. Secondly, unsupervised contrastive learning for normal images demands large batch size and lots of GPUs, if it is directly applied to hyperspectral images, the requirement for more computing resources will be too expensive.

To apply the unsupervised contrastive learning to hyperspectral classification, we addressed the two problems and proposed ContrastNet, a deep learning model for unsupervised feature learning of hyperspectral images. Our innovations can be concluded as follows:

1) We introduced discriminative learning methods into hyperspectral images. Moreover, we applied the prototypical contrastive learning to unsupervised feature learning of hyperspectral classification.
2) We designed an excellent pipeline to process the hyperspectral images, which combines two kinds of unsupervised learning methods (representative learning and contrastive learning). Furthermore, the pipeline only demands a single GPU, which means it is more affordable than the original prototypical contrastive learning method.
3) Enlighted by the autoencoder structure, we designed two autoencoder-based modules: adversarial autoencoder module(AAE module), variational autoencoder module(VAE module). The two modules can work alone for feature extraction. When using the extracted features for hyperspectral classification, they outperformed many baselines, even supervised methods.

II. Related Work

A. Variational Autoencoder

Autoencoder structure(AE)[8] is a widely used unsupervised feature extraction model. The standard autoencoder is composed of an encoder and a decoder. The encoder encodes the input images to features with fixed shape. Moreover, the features are called latent code. The decoder decodes the latent code to original input images. The autoencoder's objective function is usually reconstruction loss(e.g., Euclidean distances between output and input). Furthermore, the whole model is optimized by the backpropagation algorithm.

However, original AE does not restrict the distribution of latent code. Thus variational autoencoder(VAE)[22] is proposed to optimize the distribution of latent code. VAE used KL-divergence and the reparameterization trick to restrict latent code distribution so that randomly sampled value from a normal distribution can be decoded to an image with similar distribution to the training samples. The latent code generated by VAE performs better in the classification task, so VAE is widely used for unsupervised feature extraction.

B. Adversarial Autoencoder

Adversarial autoencoder(AAE)[23] is another AE-based structure which adopts adversarial restriction on the latent code. Unlike VAE, AAE does not use KL-divergence to measure the distance between latent code distribution and normal distribution. AAE chooses to use the idea of adversarial to restrict the distribution of latent code. By training a generative adversarial network(GAN)[9] with the latent code and randomly selected samples from a normal distribution, the distribution of latent code will become an expected normal distribution. As a result, the latent code generated by AAE will be great for classification, and the distributions of it will be similar but different from that generated by VAE.

C. Prototypical Contrast Learning

Usual unsupervised contrastive learning methods aim to find an embedding space where every sample can be discriminated among other instances in the whole dataset, such as momentum contrast(MoCo)[19]. But this objective may be too hard to achieve without enough computing resources or sufficient iteration. Although this kind of distribution may be beneficial in some downstream tasks, it is not very fit for classification. Excellent distribution for classification demands for clear clusters composed of the data in the same classes. And typical unsupervised contrastive learning can't satisfy this demand.

Prototypical contrastive learning(PCL)[21] is an unsupervised representation learning method based on contrastive learning. Unlike most contrastive learning methods, PCL introduces prototypes as latent variables to help find the maximum-likelihood estimation of the network parameters in an expectation-maximization framework. PCL uses ProtoNCE loss, a generalized version of the InfoNCE[24] loss for contrastive learning by encouraging representations to be closer to their assigned prototypes. The experiments show that PCL can get better performances than other baselines in the classification tasks, so we choose PCL as our method's base structure.

III. Proposed Method

A. Motivation

PCL showed great power of unsupervised learning in processing normal images. However, as mentioned above, it is hard to apply PCL to hyperspectral images directly. So we must make some adjustments to PCL in order to apply it to hyperspectral classification. With the blossom of AE-based methods in the hyperspectral images field, we find that the encoder in an autoencoder structure can be seen as a transformation, just like other data augmentation methods. They both map the original images from the original distribution to another distribution. The difference is, data augmentations usually do not change the shape of images, while encoder usually encodes the images to vectors with less dimensions. Another essential truth is that data augmentation methods usually keep most information of the original images to make them able to be rightly classified. Moreover, the encoder in a trained autoencoder also can keep most information in the original images.

As a result, we find it reasonable to treat the encoder as a transformation function that plays the role of data augmentation. Then the question is to find two encoders with different distributions, just like two kinds of data augmentations. So we choose designed AAE and VAE modules to act as two kinds of data augmentation because AAE and VAE can generate relatively fixed and slightly different distributions.

In this way, we can not only apply PCL to hyperspectral images but also lower the demand for deep networks. By encoding the images to shape fixed vectors(e.g., 1024-d), we can assign the feature extraction and contrastive learning into two parts, and each part will not be too complicated. Through this method, we make contrastive learning methods affordable for those who have limited computing resources.

B. VAE Module

To process the hyperspectral images, we design a VAE network to extract features from the original hyperspectral images. The structure of the VAE module is shown in Fig.1. As the figure shows, we first use PCA preprocess the whole hyperspectral image to cut down the channels. Then sliding windows are adopted to split the image into small patches with the shape $S \times S \times K$. S is the size of sliding windows, and K is the channel of the preprocessed hyperspectral image. The structure of the encoder network is enlighted by the idea of hybrid convolution in HybridSN[25]. We used several 3D convolutional layers and 2D convolutional layers to extract the input patches' spectral and spatial information. We also used several 3D transposed convolutional layers and 2D transposed convolutional layers to build the decoder network.

Furthermore, to avoid the risk of overfitting, we adopted batch normalization[26] after every convolutional layer. Every linear layer is combined with a rectified linear unit(ReLU) activation layer. We use a global pooling layer to fix the shape of the feature map, thus fix the shape of extracted features(e.g., 1024-d) after reshaping operation.

According to the original VAE, we let the encoder output two variables μ and σ. Then the latent code can be computed by Eq. 1. z is the latent code, and ε is a random value drawn from the normal distribution. The loss function for VAE network can be divided into two parts: $\mathcal{L}_{\mu,\sigma^2}$ and $\mathcal{L}_{recon}$. As Eq. 2 shows, N is the number of samples, $\mathcal{L}_{\mu,\sigma^2}$ computes the KL divergence between latent code and normal distribution. Moreover, Eq. 3 shows the detail of reconstruction loss $\mathcal{L}_{recon}$, N is the number of samples, the input image patch is $I^{x,y,z}$ with the shape of $s \times s \times k$, and $\hat{I}^{x,y,z}$ is the output image patch of the decoder. The whole loss function can be represented by Eq. 4.

$$z = \mu + \varepsilon \times \sigma \tag{1}$$

$$\mathcal{L}_{\mu,\sigma^2} = \frac{1}{2}\sum_{i=1}^{N}\left(\mu_{(i)}^2 + \sigma_{(i)}^2 - \log\sigma_{(i)}^2 - 1\right) \tag{2}$$

$$\mathcal{L}_{recon} = \sum_{i=1}^{N}\left(\frac{1}{s\times s\times k}\sum_{x=0}^{s-1}\sum_{y=0}^{s-1}\sum_{z=0}^{k-1}\left(I^{x,y,z} - \hat{I}^{x,y,z}\right)^2\right) \tag{3}$$

$$\mathcal{L} = \mathcal{L}_{\mu,\sigma^2} + \mathcal{L}_{recon} \tag{4}$$

When we finished the VAE module training, we can map all the patches to VAE features. Thus we can use these VAE features to build the training dataset for ContrastNet.

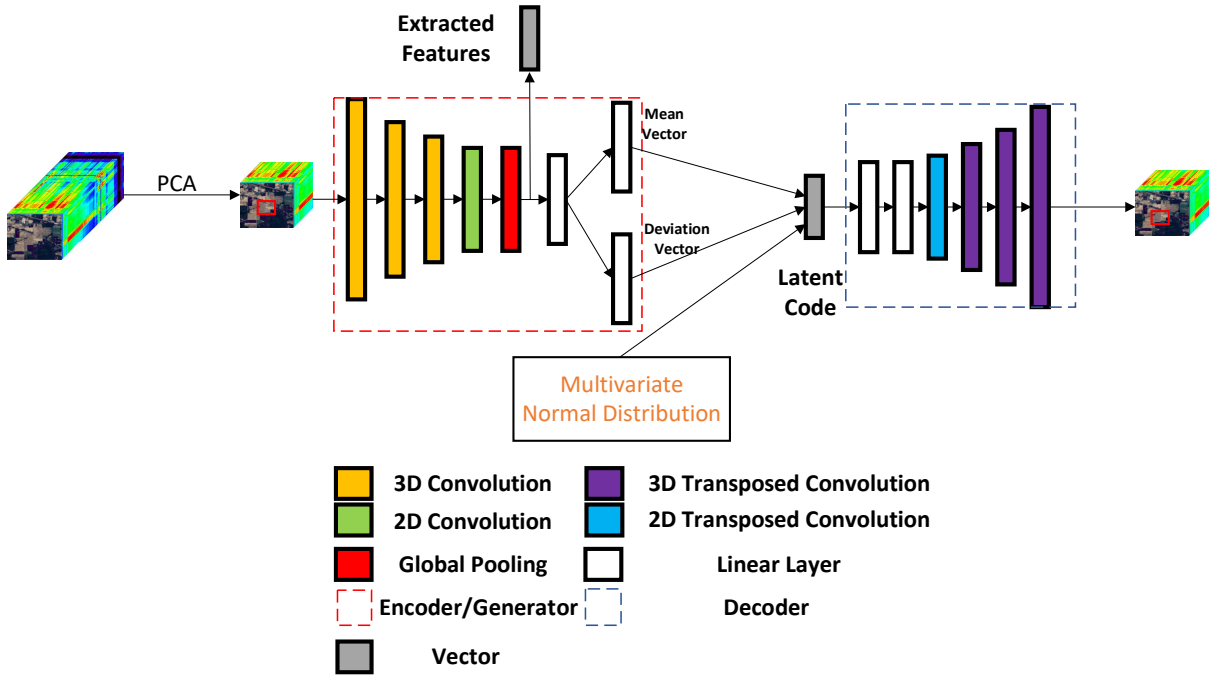


Fig. 1: Structure of VAE module. Every (transposed) convolution comprises a (transposed) convolutional layer and a batch normalization layer. Every linear layer means a linear layer and a ReLU layer.

C. AAE Module

Similar to the VAE module, the whole hyperspectral image is preprocessed by PCA in the AAE module. The structure of the AAE network is shown in Fig.2. The encoder and the decoder in the AAE module are the same with that in the VAE module except for the linear layers before the latent code. The encoder in the AAE module directly outputs the latent code and acts as a GAN generator.

According to the original AAE, the latent code is the output of the generator and the discriminant input. Another input of the discriminant is a randomly drawn variable from the normal distribution. The training of the AAE can be divided into two phases: reconstruction phase and regularization phase. In the reconstruction phase, we use reconstruction loss to optimize the parameters in the network. Reconstruction loss is shown in Eq. 5, it is the same as the reconstruction loss in the VAE module. $\mathcal{L}_{recon}$ is reconstruction loss, N is the number of samples, $I^{x,y,z}$ represents the input image patch, $s \times s \times k$ represents the shape of the input patch and $\hat{I}^{x,y,z}$ means reconstructed image patch.

In the regularization phase, the discriminator and the encoder are optimized. At first, we optimize the discriminator and then the generator(encoder). We adopt Wasserstein GAN(WGAN)[27] loss as our optimization function for the discriminator and the generator. Discriminant loss and generator loss are shown in Eq. 6 and 7, where $\mathcal{L}_D$ represents the loss of the discriminant, and $\mathcal{L}_G$ represents the loss of the generator. P_g is the distribution of the generated samples(latent code), and P_r is the distribution of the real samples(samples from a normal distribution). x represents random samples from P_g and P_r.

$$\mathcal{L}_{recon} = \sum_{i=1}^{N} \left(\frac{1}{s \times s \times k} \sum_{x=0}^{s-1} \sum_{y=0}^{s-1} \sum_{z=0}^{k-1} \left(I^{x,y,z} - \hat{I}^{x,y,z} \right)^2 \right) \tag{5}$$

$$\mathcal{L}_D = \mathbb{E}_{x \sim P_g} \left[\mathbf{D}(x) \right] - \mathbb{E}_{x \sim P_r} \left[\mathbf{D}(x) \right] \tag{6}$$

$$\mathcal{L}_G = -\mathbb{E}_{x \sim P_g} \left[\mathbf{D}(x) \right] \tag{7}$$

Similarly, when we finished the training of the AAE module, we can map all the patches to AAE features. Thus we can use these AAE features to build the training dataset for ContrastNet.

D. ContrastNet

Before we introduce the ContrastNet, the specific description of typical unsupervised contrast learning should be clarified. Let the traing set $X = \{x_1, x_2, ..., x_n\}$ have n images. And our goal is to find an embedding function f_θ(realized via a deep neural network) that maps X to another space $V = \{v_1, v_2, ..., v_n\}$ where $v_i = f_\theta(x_i)$, and v_i can describe x_i best. Usually the objective is achieved by the InfoNCE loss function[28] as Eq. 8 shows. v_i' is a positive embedding for instance i, and v_j' includes one positive embedding and r negative embeddings for other instances, and τ is a temperature hyper-parameter. In MoCo[19], these embeddings are obtained by feeding x_i to a momentum encoder parametrized by θ', $v_i' = f_{\theta'}(x_i)$, where θ' is a moving average of θ.

When minimizing the Eq. 8, the distance between v_i and v_i' are becoming close, and the distance between v_i and r negative embeddings are becoming far. However, original InfoNCE loss

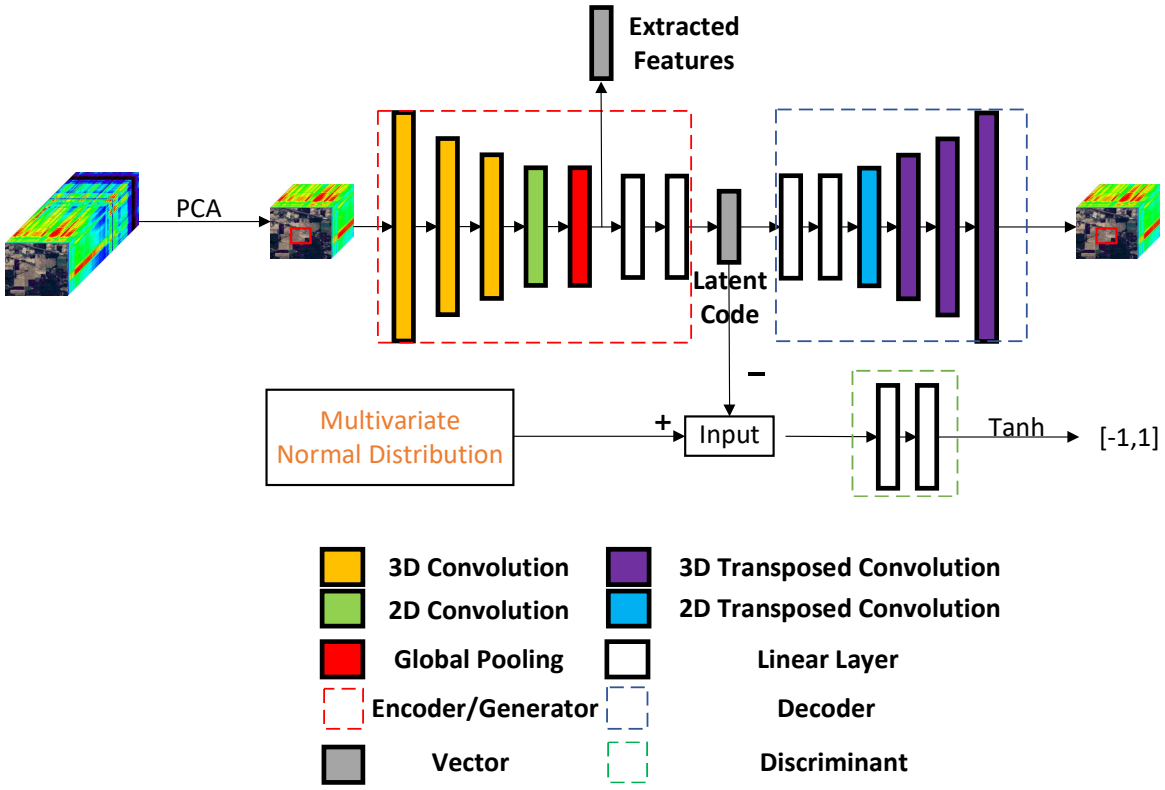


Fig. 2: Structure of AAE module. Every (transposed) convolution comprises a (transposed) convolutional layer and a batch normalization layer. Every linear layer means a linear layer and a ReLU layer.

uses a fixed τ, which means the same concentration on a certain scale. Moreover, it computes the distance between v_i and all kinds of negative embeddings, some of which are not representative.

As a result, prototypical contrastive learning proposed a new loss function named ProtoNCE. As Eq. 9 and Eq. 10 show, based on InfoNCE, ProtoNCE contains a second part $\mathcal{L}_{Proto}$. It use prototypes c instead of v', and replace the fixed temperature τ with a per-prototype concentration estimation ϕ.

$$\mathcal{L}_{\text{InfoNCE}} = \sum_{i=1}^{n} -\log \frac{\exp\left(v_i \cdot v_i'/\tau\right)}{\sum_{j=0}^{r} \exp\left(v_i \cdot v_j'/\tau\right)} \tag{8}$$

$$\mathcal{L}_{Proto} = \sum_{i=1}^{n} -\left(\frac{1}{M}\sum_{m=1}^{M} \log \frac{\exp\left(v_i \cdot c_s^m/\phi_s^m\right)}{\sum_{j=0}^{r} \exp\left(v_i \cdot c_j^m/\phi_j^m\right)}\right) \tag{9}$$

$$\mathcal{L}_{ProtoNCE} = \mathcal{L}_{Proto} + \mathcal{L}_{\text{InfoNCE}} \tag{10}$$

The detailed algorithm of prototypical contrastive learning can be found in Algorithm 1. There are two encoders in the PCL algorithm, one encoder maps x_i to v_i, and the momentum encoder maps x_i to v_i'. Practically, we do not use the same x_i to get v_i and v_i', two images with proper data augmentation will be the input of two encoders for better performance. The first encoder parameters can be represented by θ, and the parameters in the momentum encoder can be represented by θ' which meets Eq. 11. In algorithm 1, $k-means()$[29] is the GPU implementation of a widely used clustering algorithm. And C^m is the sets of k_m prototypes(center of clusters). $Concentration()$ is the function of concentration estimation ϕ, as Eq. 12 shows, ϕ is calculated by the momentum features $\{v_z'\}_{z=1}^{Z}$ that are within the same clusters as a prototype c, α is a smooth parameter to ensure that small clusters do not have overly-large ϕ, and the typical value is 10. $SGD()$[30] is a kind of optimization function to update the parameters in the encoder.

$$\theta' = 0.999 * \theta' + 0.001 * \theta \tag{11}$$

$$\phi = \frac{\sum_{z=1}^{Z} \|v_z' - c\|_2}{Z \log(Z+\alpha)} \tag{12}$$

The proposed ContrastNet is similar to the original PCL algorithm, but there are several differences. Firstly, ContrastNet uses AAE and VAE features as inputs, not two perturbed images. Secondly, the structure of ContrastNet is more straightforward and specialized in the "contrastive" part, not the "encoding" part. Thirdly, we adopt the projection head trick in sim-CLR[20] to improve the performance.

Fig. 3 is an overview of the ContrastNet. Expectation-Maximization (EM) algorithm is adopted to train the ContrastNet. In the E-step, VAE features are fed into the momentum

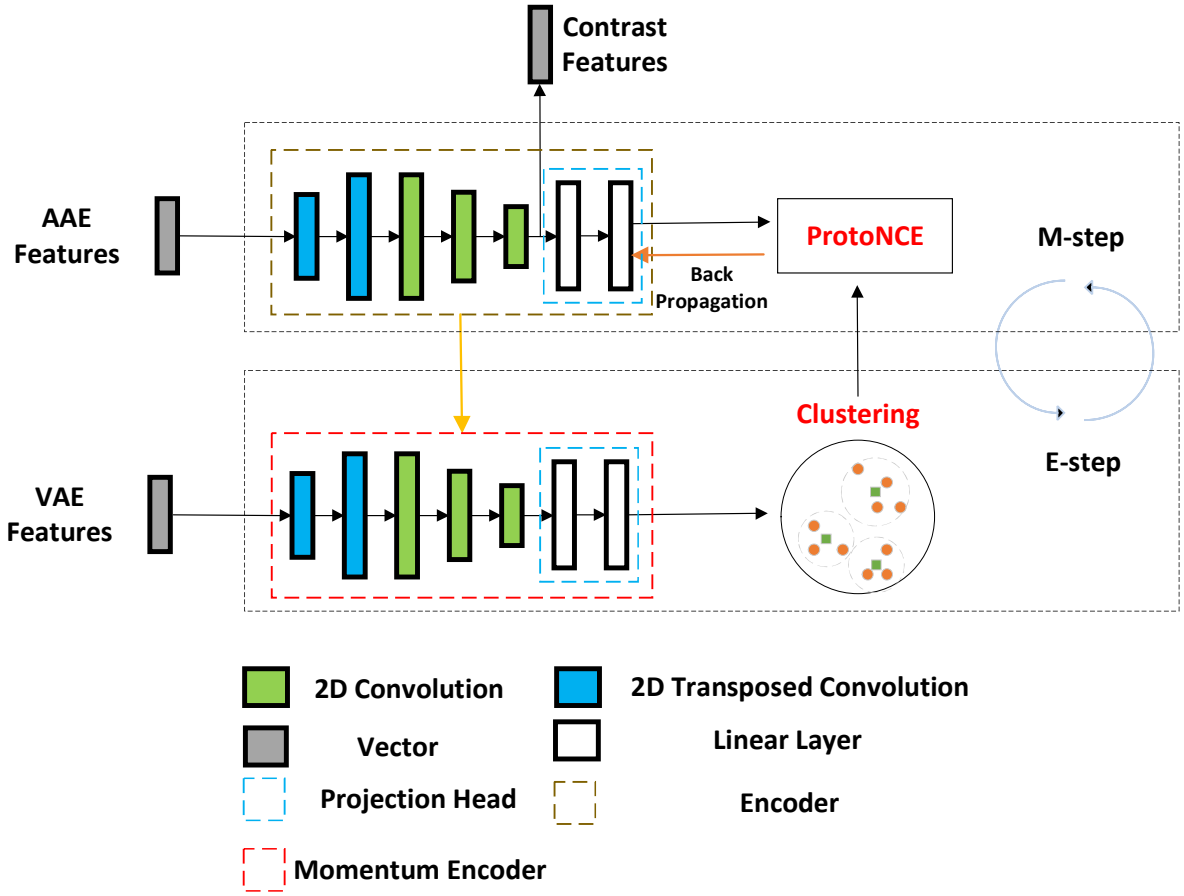


Fig. 3: Structure of ContrastNet. Every (transposed) convolution comprises a (transposed) convolutional layer and a batch normalization layer. Every linear layer means a linear layer and a ReLU layer. Clustering means k-means algorithm, and ProtoNCE is $\mathcal{L}_{ProtoNCE}$ in Eq. 10. M-step and E-step are the phases of Expectation-Maximization algorithm.

Algorithm 1 Prototypical Contrastive Learning.

Input: encoder f_θ, training dataset X, number of clusters $K = \{k_m\}_{m=1}^{M}$

$\theta' = \theta$
while not MaxEpoch **do**
  $V' = f'_\theta(X)$
  for $m = 1$ **to** M **do**
    $C^m = k - means(V', k_m)$
    $\phi^m = Concentration(C^m, V')$
  end for
  for x **in** $Dataloader(X)$ **do**
    $v = f_\theta(x), v' = f'_\theta(x)$
    $\mathcal{L}_{ProtoNCE}(v, v', \{C^m\}_{m=1}^{M}, \{\phi^m\}_{m=1}^{M})$
    $\theta = SGD(\mathcal{L}_{ProtoNCE}, \theta)$
    $\theta' = 0.999 * \theta' + 0.001 * \theta$
  end for
end while

encoder, then C^m and ϕ^m are calculated. In the M-step, $\mathcal{L}_{ProtoNCE}$ is calculated based on the updated features and variables in the E-step, then the two encoders are updated via backpropagation. Similarly, all convolutions in the ContrastNet is composed of a convolutional layer and a batch normalization layer, and every linear layer is combined with a ReLU layer. The 1024-d AAE and VAE features will be reshaped to $4\times4\times64$ feature maps first, in this way, the spatial information extracted can be preserved. And then, the feature maps are fed into the ContrastNet. The structure of ContrastNet is light and straightforward because the previous process has achieved the feature extraction. The ContrastNet only needs to pay attention to the contrastive learning. We adopted the projection head structure in the ContrastNet, that is, we use the output of the projection head for training, and use the features before the head for testing. In this way, the extracted contrast features will contain more original information. In the testing phase, we need to extract contrast features in Fig. 3 from the primary encoder, which can save much testing time for the ContrastNet.

IV. Experiments and Analysis

A. Datasets

Experiments are conducted using three publicly available HSI datasets: Indian Pines(IP), University of Pavia(PU), and Salinas Scene(SA). The IP dataset was gathered using the Airborne Visible / Infrared Imaging Spectrometer (AVIRIS)

TABLE I: Structure of AAE Encoder. The shape of data is defined in pytorch style. -1 means batchsize in the shape array.

AAE Encoder	
Layer(type)	Output Shape
input	[-1,1,15,27,27]
Conv3d	[-1,8,9,25,25]
BatchNorm3d	[-1,8,9,25,25]
ReLU	[-1,8,9,25,25]
Conv3d	[-1,16,5,23,23]
BatchNorm3d	[-1,16,5,23,23]
ReLU	[-1,16,5,23,23]
Conv3d	[-1,32,3,21,21]
BatchNorm3d	[-1,32,3,21,21]
ReLU	[-1,32,3,21,21]
Conv2d	[-1,64,19,19]
BatchNorm2d	[-1,64,19,19]
ReLU	[-1,64,19,19]
AdaptiveAvgPool2d	[-1,64,4,4]
Linear	[-1,512]
ReLU	[-1,512]
Linear	[-1,128]

TABLE II: Structure of AAE Decoder. The shape of data is defined in pytorch style. -1 means batchsize in the shape array.

AAE Decoder	
Layer(type)	Output Shape
input	[-1,128]
Linear	[-1,256]
ReLU	[-1,256]
Linear	[-1,23104]
ReLU	[-1,23104]
ConvTransposed2d	[-1,96,21,21]
BatchNorm2d	[-1,96,21,21]
ReLU	[-1,96,21,21]
ConvTransposed3d	[-1,16,5,23,23]
BatchNorm3d	[-1,16,5,23,23]
ReLU	[-1,16,5,23,23]
ConvTransposed3d	[-1,8,9,25,25]
BatchNorm3d	[-1,8,9,25,25]
ReLU	[-1,8,9,25,25]
ConvTransposed3d	[-1,1,15,27,27]
BatchNorm3d	[-1,1,15,27,27]

hyperspectral sensor. It consisted of 145×145 pixels and 224 spectral reflectance bands in the wavelength range of $0.4 \sim 2.5 \times 10^{-6}$ meters. After removing bands covering the region of water absorption, 200 bands were left, and the ground truth was composed of 16 classes. The PU dataset was a hyperspectral image with 610×340 pixels and 103 spectral bands in the wavelength range of $0.43 \sim 0.86 \times 10^{-6}$ meters. The ground truth contained nine classes. The SA dataset was an image with 512×217 pixels and 224 spectral bands. This dataset was in the wavelength range of $0.36 \sim 2.5 \times 10^{-6}$ meters. Twenty water-absorbing bands were discarded from the SA dataset, so it was a dataset with 204 spectral bands and 16 landcover classes in the ground truth.

B. Experiments Details

All experiments were implemented on a PC with 16GB RAM and a GTX 1080 GPU. The coding environment was pytorch[31]. The shape of patches is $27 \times 27 \times 15$ in PU and SA dataset, and $27 \times 27 \times 30$ in the IP dataset. When we use $27 \times 27 \times 15$ patches, our networks' details are shown as follows.

Tab. I and Tab. II show the structure of AAE. The structure of the discriminant is simple, so it is omitted. The latent code dimension is 128 and extracted AAE features are flattened outputs of the pooling layer in the encoder(1024-d). In the training phase, we use two Adam[32] optimizers to optimize the encoder and the decoder, and two SGD[30] optimizers to optimize the generator and the discriminant. Two Adam optimizers are set with the learning rate of 0.001, and weight decay is set to 0.0005. The optimizer for the generator is set with a 0.0001 learning rate and no weight decay. The optimizer for the discriminant is set with a 0.00005 learning rate and no weight decay. The batch size is set to 128. We train the AAE module for 20 epochs, and only save the parameters of the encoder.

Tab. III and Tab. IV show the structure of VAE. The dimension of latent code is also 128 and extracted VAE features are flattened outputs of the pooling layer in the encoder(1024-d)

TABLE III: Structure of VAE Encoder. The shape of data is defined in pytorch style. -1 means batchsize in the shape array.

VAE Encoder	
input	[-1,1,15,27,27]
Conv3d	[-1,8,9,25,25]
BatchNorm3d	[-1,8,9,25,25]
ReLU	[-1,8,9,25,25]
Conv3d	[-1,16,5,23,23]
BatchNorm3d	[-1,16,5,23,23]
ReLU	[-1,16,5,23,23]
Conv3d	[-1,32,3,21,21]
BatchNorm3d	[-1,32,3,21,21]
ReLU	[-1,32,3,21,21]
Conv2d	[-1,64,19,19]
BatchNorm2d	[-1,64,19,19]
ReLU	[-1,64,19,19]
AdaptiveAvgPool2d	[-1,64,4,4]
Linear	[-1,512]
ReLU	[-1,512]
Linear/Linear	[-1,128]/[-1,128]

too. In the training phase, we use two Adam[32] optimizers to optimize the encoder and the decoder. Two Adam optimizers are set with the learning rate of 0.001, and weight decay is set to 0.0005. The two outputs of VAE encoder are used to compute the latent code. Then the latent code is fed into the decoder. The batch size is set to 128. We train the VAE module for 30 epochs, and only save the parameters of the encoder.

The two encoders in ContrastNet have the same structure, which is shown in Tab. V. Two inputs are 1024-d vectors, and the dimension of contrast features and projected features are both 128-d. We train the ContrastNet with an SGD optimizer, and the learning rate is 0.003, weight decay is set to 0.001. Number of negative samples $r = 640$, and batch size is 128, τ in Eq. 8 is 0.01, number of clusters $K = [1000, 1500, 2500]$. According to the original PCL method, we only train the network with Eq. 8 for 30 epochs, then train the network with Eq. 10 for 170 epochs. The learning rate will be multiplied with 0.1 when the trained epoch is more than 120 and 160.

C. Results

We conducted classification experiments in the three datasets. Eleven other methods were adopted as baselines

TABLE IV: Structure of VAE Decoder. The shape of data is defined in pytorch style. -1 means batchsize in the shape array.

VAE Decoder	
Layer(type)	Output Shape
input	[-1,128]
Linear	[-1,256]
ReLU	[-1,256]
Linear	[-1,23104]
ReLU	[-1,23104]
ConvTransposed2d	[-1,96,21,21]
BatchNorm2d	[-1,96,21,21]
ReLU	[-1,96,21,21]
ConvTransposed3d	[-1,16,5,23,23]
BatchNorm3d	[-1,16,5,23,23]
ReLU	[-1,16,5,23,23]
ConvTransposed3d	[-1,8,9,25,25]
BatchNorm3d	[-1,8,9,25,25]
ReLU	[-1,8,9,25,25]
ConvTransposed3d	[-1,1,15,27,27]
BatchNorm3d	[-1,1,15,27,27]

TABLE V: Structure of ContrastNet Encoder. The shape of data is defined in pytorch style. -1 means batchsize in the shape array.

ContrastNet Encoder	
Layer(type)	Output Shape
input	[-1,1024]
ConvTransposed2d	[-1,64,6,6]
BatchNorm2d	[-1,64,6,6]
ReLU	[-1,64,6,6]
ConvTransposed2d	[-1,64,8,8]
BatchNorm2d	[-1,64,8,8]
ReLU	[-1,64,8,8]
Conv2d	[-1,128,6,6]
BatchNorm2d	[-1,128,6,6]
ReLU	[-1,128,6,6]
Conv2d	[-1,64,4,4]
BatchNorm2d	[-1,64,4,4]
ReLU	[-1,64,4,4]
Conv2d	[-1,32,2,2]
ReLU	[-1,32,2,2]
Linear	[-1,128]
ReLU	[-1,128]
Linear	[-1,128]

to compare with our methods. They are linear discriminant analysis(LDA)[33], locality-preserving dimensionality reduction(LFDA)[34], sparse graph-based discriminant analysis(SGDA)[35], sparse and low-rank graph for discriminant analysis(SLGDA)[36], deep convolutional neural networks(1D-CNN)[1], supervised deep feature extraction(S-CNN)[37], tensor principal component analysis(TPCA)[38], stacked sparse autoencoder(SSAE)[10], unsupervised deep feature extraction(EPLS)[39], and 3D convolutional autoencoder(3DCAE)[12]. To illustrate the great feature learning ability of ContrastNet, we used SVM to classify the features we learned by ContrastNet. Moreover, features learned by AAE and VAE were also used to compare with the ContrastNet. 10% samples of each class are randomly selected for training SVM in the IP and PU datasets, and 5% samples of each class are used for training in the SA dataset, the rest in each dataset is used for testing. The quantitative results are evaluated by average accuracy (AA) and overall accuracy (OA). The results in the Indian Pines dataset are shown in Tab. VI. It is clear that the structures we proposed performed best in many classes. AAE got the best results in five classes while ContrastNet got the best results in six classes, and VAE also performed best in 3 classes.

To some extent, AAE and VAE are two better feature extractors than 3DCAE in terms of AA. However, ContrastNet, which learns information from AAE and VAE, showed a significant increase in OA. It can be evidence indicating that ContrastNet can extract some core information from two distributions by contrast learning. As a result, it got the highest OA. However, ContrastNet did not perform well enough in AA in the Indian Pines dataset. We guess the reason for it may be the imbalance of the number of samples in the Indian Pines dataset. Classes with large numbers dominated the location of clusters. Thus the ContrastNet tended to perform well in these classes, which decreased accuracy in other classes.

The results in the SA dataset are shown in Tab. VII. Similar to the IP dataset results, AAE performed best in five classes, VAE and ContrastNet both performed best in three classes. Though all methods got good performances in this dataset, ContrastNet still got the highest OA and got the highest AA. Unlike the IP dataset, the number of samples in the SA dataset is relatively balanced, so ContrastNet got a higher AA than AAE and VAE. The same conclusion can be drawn from the results in the PU dataset, which are shown in Tab.VIII. ContrastNet still outperformed any other methods in terms of OA and AA.

Generally speaking, the proposed AAE, VAE, and ContrastNet in this paper are good models that outperform many supervised and unsupervised methods. Among the three methods, Autoencoder-based AAE and VAE are models that make use of reconstruction information and concentrate on the single sample itself. ContrastNet is a model that learns the similarity between two different distributions of a single sample, and also learn the dissimilarity between positive samples and negative samples. According to the experiment results, ContrastNet performs better than AAE and VAE in OA and AA when the number of samples is balanced. When the number of samples is imbalanced, ContrastNet tends to get higher OA and lower AA than AAE and VAE.

To intuitively show the clustering effect of ContrastNet, we use t-SNE[40] method to visualize the features extracted by AAE, VAE, and ContrastNet. The visualizations are shown in Fig. 4, 5, 6. It is not hard to find that the features extracted by the ContrastNet are easier to classify. Because the inner-class distances are small, and the inter-class distances are big in the ContrastNet features.

The computational analysis is shown in Tab. IX. Because the training of ContrastNet demands extracted AAE and VAE features, which means the demand for AAE and VAE, the training time of ContrastNet is considerably longer than other algorithms. However, When extracting features, the time consumption of ContrastNet is fairly fast than others. Because for AAE and VAE, only encoders are used in the feature extraction phase. Moreover, the structure of AAE and VAE is relatively simple and efficient, so the ContrastNet performs fast in the feature extraction phase.

TABLE VI: Comparison in the Indian Pines dataset. 10% labeled samples are used for training, and the rest are used for testing. AA menas average accuracy and OA means overall accuracy. The best values in each row are in bold. And the methods in bold are structures proposed in this paper. Some of the data in the table is quoted from [12].

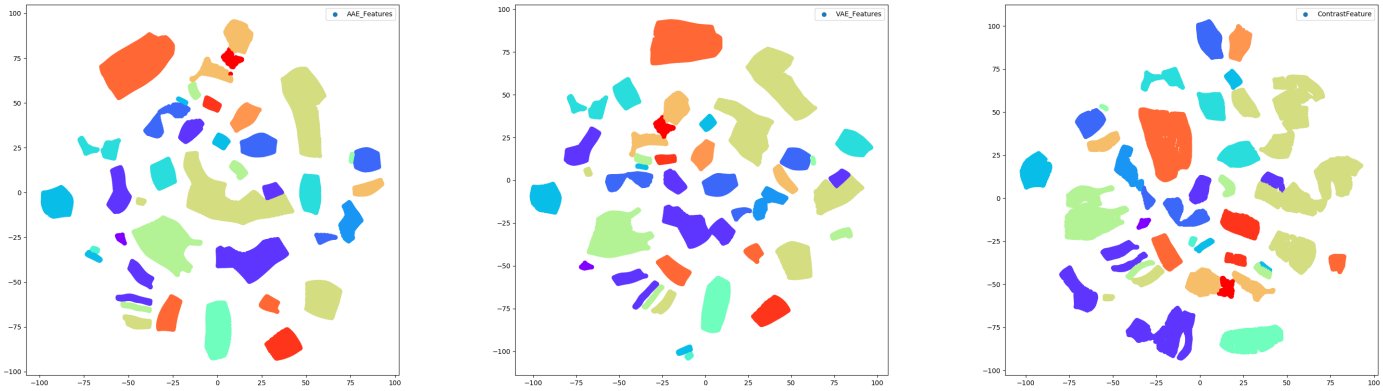
Class	Supervised Feature Extraction						Unsupervised Feature Extraction							
	LDA	LFDA	SGDA	SLGDA	1D-CNN	S-CNN	PCA	TPCA	SSAE	EPLS	3DCAE	**AAE**	**VAE**	**ContrastNet**
1	58.54	29.63	42.59	39.62	43.33	83.33	39.02	60.97	56.25	58.72	90.48	**100.00**	**100.00**	85.37
2	69.88	75.59	80.89	85.56	73.13	81.41	72.30	87.00	69.58	59.91	92.49	81.63	78.78	**97.15**
3	65.86	75.42	65.71	74.82	65.52	74.02	72.02	94.51	75.36	71.34	90.37	95.27	92.37	**97.95**
4	73.71	58.12	64.10	49.32	51.31	71.49	55.87	79.34	64.58	74.31	86.90	**99.22**	97.34	95.62
5	90.32	95.17	94.57	95.35	87.70	90.11	93.09	93.08	88.81	97.95	94.25	95.17	93.87	**96.09**
6	92.09	96.12	98.39	95.59	95.10	94.06	94.67	96.34	87.00	96.44	97.07	**98.73**	98.27	96.80
7	96.00	11.53	50.00	36.92	56.92	84.61	80.00	76.00	90.00	54.02	91.26	96.00	**98.67**	70.67
8	98.14	93.87	99.80	99.75	96.64	98.37	98.37	99.76	89.72	88.99	97.79	**99.84**	99.77	98.68
9	11.11	0.00	0.00	5.00	28.89	33.33	88.89	**100.00**	**100.00**	58.89	75.91	96.30	98.15	70.37
10	73.80	80.89	59.30	69.11	75.12	86.05	74.49	79.51	77.19	73.10	87.34	87.01	78.86	**97.45**
11	55.41	83.02	84.44	89.91	83.49	82.98	69.58	85.42	77.58	70.78	90.24	89.08	81.75	**98.40**
12	76.92	86.32	77.04	86.78	67.55	73.40	65.29	84.24	72.00	57.51	**95.76**	93.51	90.64	93.57
13	91.30	79.25	99.09	**99.51**	96.86	87.02	98.37	98.91	87.80	99.25	97.49	98.56	98.56	95.32
14	93.32	88.87	92.50	96.45	96.51	94.38	91.39	98.06	93.48	95.07	96.03	95.73	93.24	**98.51**
15	67.72	60.00	68.68	61.79	39.08	75.57	48.99	87.31	72.36	91.26	90.48	**97.31**	97.02	96.73
16	90.36	53.68	85.26	84.16	89.40	79.76	87.95	96.38	97.22	91.27	98.82	98.02	**98.81**	79.76
AA(%)	76.89	66.72	72.65	73.10	71.66	84.44	76.89	89.31	81.18	77.43	92.04	**95.09**	93.51	91.78
OA(%)	76.88	81.79	80.05	85.19	79.66	80.72	76.88	88.55	79.78	77.18	92.35	91.80	88.03	**97.08**

TABLE VII: Comparison in the Salinas dataset. 5% labeled samples are used for training, and the rest are used for testing. AA menas average accuracy and OA means overall accuracy. The best values in each row are in bold. And the methods in bold are structures proposed in this paper. Some of the data in the table is quoted from [12].

Class	Supervised Feature Extraction						Unsupervised Feature Extraction							
	LDA	LFDA	SGDA	SLGDA	1D-CNN	S-CNN	PCA	TPCA	SSAE	EPLS	3DCAE	**AAE**	**VAE**	**ContrastNet**
1	99.16	99.20	99.65	98.14	97.98	99.55	97.48	99.88	**100.00**	99.99	**100.00**	99.98	99.74	99.93
2	99.94	99.19	99.33	99.44	99.25	99.43	99.52	99.49	99.52	99.92	99.29	**100.00**	99.79	99.80
3	99.79	99.75	99.30	99.29	94.43	98.81	99.41	99.04	94.24	98.75	97.13	**100.00**	**100.00**	99.95
4	99.77	99.71	99.21	99.57	99.42	97.45	99.77	**99.84**	99.17	98.52	97.91	99.09	99.57	98.01
5	98.98	98.47	99.07	98.06	96.60	97.96	98.70	98.96	98.82	98.33	98.26	99.42	**99.71**	99.48
6	99.89	99.09	99.57	99.32	99.51	99.83	99.65	99.80	**100.00**	99.92	99.98	99.97	99.86	99.94
7	**99.97**	99.66	99.27	99.33	99.27	99.59	99.94	99.84	99.94	97.69	99.64	99.93	99.96	99.79
8	81.84	86.89	89.78	89.48	86.79	94.40	83.90	84.11	80.73	78.86	91.58	91.21	86.91	**99.53**
9	99.90	97.34	**100.00**	99.65	99.08	98.85	99.97	99.60	99.47	99.54	99.28	99.69	99.99	99.71
10	96.31	95.85	97.99	97.94	93.71	97.35	96.89	95.76	92.12	95.98	96.65	98.46	96.54	**99.80**
11	99.61	97.94	99.53	99.06	94.55	97.71	96.84	96.14	96.62	98.60	97.74	99.57	**100.00**	99.80
12	99.67	99.64	**100.00**	**100.00**	99.59	98.73	99.95	99.07	97.75	99.44	98.84	**100.00**	99.44	99.98
13	99.20	97.38	98.47	97.82	97.50	96.72	99.54	**100.00**	95.81	98.85	99.26	99.89	97.24	98.20
14	96.56	93.18	96.07	90.47	94.08	95.22	97.24	95.74	96.65	98.56	97.49	**99.08**	96.49	98.62
15	73.60	63.04	65.40	69.51	66.52	95.61	76.68	79.54	79.73	83.13	87.85	93.69	87.90	**99.53**
16	98.48	99.00	99.34	99.00	97.48	99.44	97.90	98.40	99.12	99.50	98.34	**99.73**	99.61	99.57
AA(%)	96.42	95.33	96.37	96.01	94.73	97.39	96.46	93.24	95.61	96.55	97.45	98.73	97.67	**99.48**
OA(%)	92.18	91.22	91.82	93.31	91.30	97.92	92.87	96.57	92.11	92.35	95.81	97.10	95.23	**99.60**

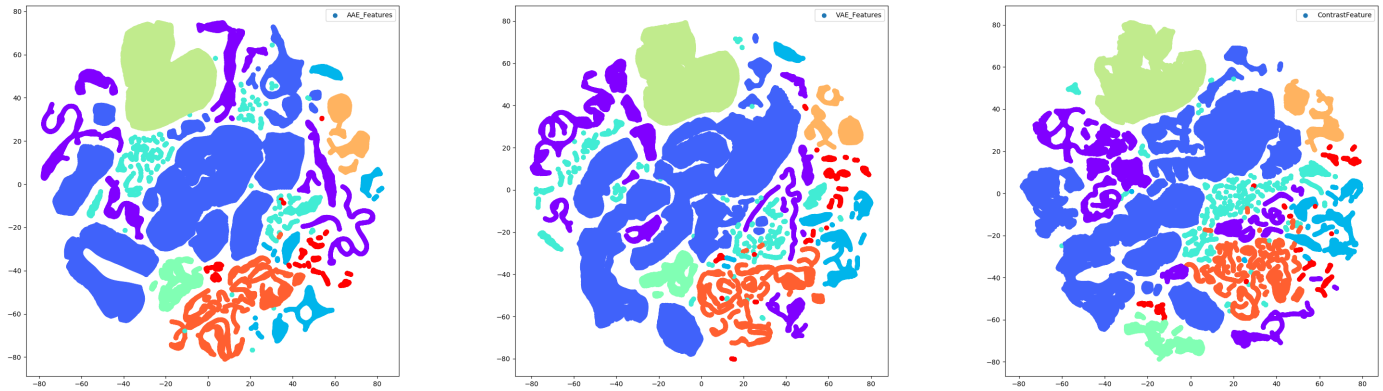
TABLE VIII: Comparison in the University of Pavia dataset. 10% labeled samples are used for training, and the rest are used for testing. AA menas average accuracy and OA means overall accuracy. The best values in each row are in bold. And the methods in bold are structures proposed in this paper. Some of the data in the table is quoted from [12].

Class	Supervised Feature Extraction						Unsupervised Feature Extraction							
	LDA	LFDA	SGDA	SLGDA	1D-CNN	S-CNN	PCA	TPCA	SSAE	EPLS	3DCAE	**AAE**	**VAE**	**ContrastNet**
1	78.41	93.45	91.37	94.66	90.93	95.40	90.89	96.17	95.72	95.95	95.21	95.39	81.31	**99.49**
2	83.69	97.36	97.23	97.83	96.94	97.31	93.27	97.95	94.13	95.91	96.06	98.96	97.65	**99.98**
3	73.02	71.41	66.08	77.27	69.43	81.21	82.60	86.50	87.47	94.33	91.32	97.49	91.00	**99.06**
4	93.68	91.00	91.19	93.18	90.32	95.83	92.41	94.84	96.91	**99.28**	98.28	96.94	94.79	97.75
5	**100.00**	97.99	99.33	98.51	99.44	99.91	98.98	**100.00**	99.76	99.92	95.55	99.94	99.94	99.81
6	88.51	87.55	80.31	90.08	73.69	95.29	92.00	94.76	95.76	93.57	95.30	99.29	**99.91**	99.90
7	85.75	80.23	75.26	85.34	83.42	87.05	85.83	91.89	91.18	98.17	95.14	**100.00**	99.42	99.83
8	74.49	86.98	84.11	90.49	83.65	87.35	82.96	89.04	82.47	91.23	91.38	97.90	94.77	**98.79**
9	99.11	99.26	95.14	99.37	98.23	95.66	**100.00**	98.94	**100.00**	99.78	99.96	96.32	88.62	94.84
AA(%)	86.29	89.47	86.67	91.86	87.34	94.75	90.99	95.64	93.71	96.33	95.36	98.03	94.16	**98.83**
OA(%)	83.75	92.77	90.58	94.15	89.99	92.78	91.37	94.45	93.51	95.13	95.39	98.14	94.53	**99.46**



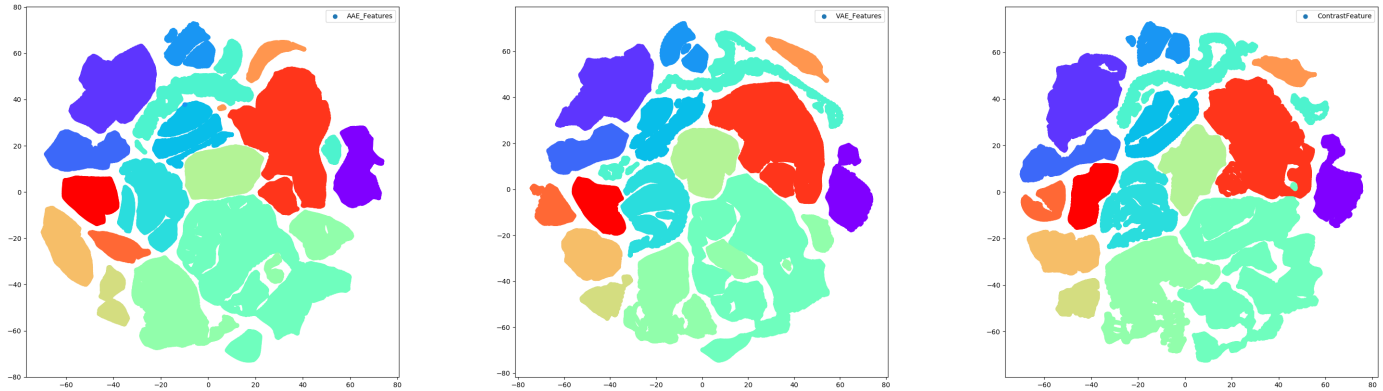
(a) AAE (b) VAE (c) ContrastNet

Fig. 4: Visualization of extracted features in the Indian Pines dataset. Each class is corresponding to a kind of color.



(a) AAE (b) VAE (c) ContrastNet

Fig. 5: Visualization of extracted features in the University of Pavia dataset. Each class is corresponding to a kind of color.



(a) AAE (b) VAE (c) ContrastNet

Fig. 6: Visualization of extracted features in the Salinas dataset. Each class is corresponding to a kind of color.

TABLE IX: Computational analysis of different feature extration algorithms. The best value in each row is in bold. And the methods in bold are structures proposed in this paper. Some of the data in the table is quoted from [12].

Indian Pines dataset							
	1D-CNN	S-CNN	TPCA	SSAE	EPLS	3DCAE	**ContrastNet**
Training(s)	**20.6**	119	44	240.1	111.9	1156	2073
Feature Extraction (s)	17.4	3.2	150	**2.76**	47.3	5.22	12.77
Salinas dataset							
	1D-CNN	S-CNN	TPCA	SSAE	EPLS	3DCAE	**ContrastNet**
Training(s)	**43.1**	134	241	513	103.5	1159	4784
Feature Extraction (s)	83.8	62	816	**16.3**	192.2	26.4	36.65
University of Pavia dataset							
	1D-CNN	S-CNN	TPCA	SSAE	EPLS	3DCAE	**ContrastNet**
Training(s)	**32.1**	332	44	491	127.13	1168	3778
Feature Extraction (s)	150.07	106	150	31.9	141.5	32.04	**27.82**

V. Conclusion

In this paper, we proposed an unsupervised feature learning method based on autoencoder and contrastive learning. This method combines unsupervised representative methods and unsupervised discriminative methods, learning to extract better hyperspectral classification features than other baseline methods. In the proposed method, we designed two efficient autoencoder structure: VAE and AAE, and design a ContrastNet for contrastive learning, which reduces the computing resources demand of contrastive learning. Our experiments show that the proposed method can extract more representative features and keep a high feature extraction speed in the testing phase. Our work shows that unsupervised learning still has great potential in the remote sensing field, and we hope others can get more exciting ideas through our exploration.

Appendix A

Acknowledgment

The authors would like to thank...

References

[1] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, 2015.

[2] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 6232–6251, 2016.

[3] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, 2017.

[4] W. Song, S. Li, L. Fang, and T. Lu, "Hyperspectral image classification with deep feature fusion network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 6, pp. 3173–3184, 2018.

[5] Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral–spatial residual network for hyperspectral image classification: A 3-d deep learning framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 847–858, 2017.

[6] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[7] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, "Hyperspectral image classification with independent component discriminant analysis," *IEEE transactions on Geoscience and remote sensing*, vol. 49, no. 12, pp. 4865–4876, 2011.

[8] D. H. Ballard, "Modular learning in neural networks." in *AAAI*, 1987, pp. 279–284.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[10] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectralspatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2438–2442, 2015.

[11] L. Zhu, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Generative adversarial networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 9, pp. 5046–5063, 2018.

[12] S. Mei, J. Ji, Y. Geng, Z. Zhang, X. Li, and Q. Du, "Unsupervised spatialspectral feature learning by 3d convolutional autoencoder for hyperspectral classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 6808–6820, 2019.

[13] Y. Zhan, D. Hu, Y. Wang, and X. Yu, "Semisupervised hyperspectral image classification based on generative adversarial networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 2, pp. 212–216, 2018.

[14] X. Wang, K. Tan, Q. Du, Y. Chen, and P. Du, "Caps-triplegan: Gan-assisted capsnet for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 7232–7245, 2019.

[15] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[16] X. Liu, Y. Zhou, J. Zhao, R. Yao, B. Liu, and Y. Zheng, "Siamese convolutional neural networks for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 8, pp. 1200–1204, 2019.

[17] M. Rao, L. Tang, P. Tang, and Z. Zhang, "Es-cnn: An end-to-end siamese convolutional neural network for hyperspectral image classification," in *2019 Joint Urban Remote Sensing Event (JURSE)*, 2019, pp. 1–4.

[18] L. Huang and Y. Chen, "Dual-path siamese cnn for hyperspectral image classification with limited training samples," *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, 2020.

[19] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[21] J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. H. Hoi, "Prototypical contrastive learning of unsupervised representations," 2020.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[23] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[24] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3733–3742.

[25] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "Hybridsn: Exploring 3-d-2-d cnn feature hierarchy for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, 2019.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[27] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[28] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
[29] J. Johnson, M. Douze, and H. Jegou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, p. 11, 2019. [Online]. Available: http://dx.doi.org/10.1109/tbdata.2019.2921572
[30] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.
[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[33] Q. Du, "Modified fisher's linear discriminant analysis for hyperspectral imagery," *IEEE geoscience and remote sensing letters*, vol. 4, no. 4, pp. 503–507, 2007.
[34] W. Li, S. Prasad, J. E. Fowler, and L. M. Bruce, "Locality-preserving dimensionality reduction and classification for hyperspectral image analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 4, pp. 1185–1198, 2012.
[35] N. H. Ly, Q. Du, and J. E. Fowler, "Sparse graph-based discriminant analysis for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 7, pp. 3872–3884, 2014.
[36] W. Li, J. Liu, and Q. Du, "Sparse and low-rank graph for discriminant analysis of hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 7, pp. 4094–4105, 2016.
[37] B. Liu, X. Yu, P. Zhang, A. Yu, Q. Fu, and X. Wei, "Supervised deep feature extraction for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 1909–1921, 2018.
[38] Y. Ren, L. Liao, S. J. Maybank, Y. Zhang, and X. Liu, "Hyperspectral image spectral-spatial feature extraction via tensor principal component analysis," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1431–1435, Sep. 2017.
[39] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1349–1362, 2016.
[40] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Zeyu Cao received the B.S. degree in automation from Zhejiang University, Hangzhou, China, where he is currently pursuing the Ph.D. degree in control theory and control engineering. His research interests include object detection and machine learning.

Xiaorun Li received the B.S. degree from the National University of Defense Technology, Changsha, China, in 1992, and the M.S. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1995 and 2008, respectively. Since 1995, he has been with Zhejiang University, where he is currently a Professor with the College of Electrical Engineering. His research interests include hyperspectral image processing, signal and image processing, and pattern recognition.

Liaoying Zhao received the B.S. and M.S. degrees from Hangzhou Dianzi University, Hangzhou, China, in 1992 and 1995, respectively, and the Ph.D. degree from Zhejiang University, Hangzhou, in 2004. Since 1995, she has been with Hangzhou Dianzi University, where she is currently a Professor with the College of Computer Science. Her research interests include hyperspectral image processing, signal and image processing, pattern recognition, and machine learning.