IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 4, 2019, accepted February 14, 2019, date of publication March 4, 2019, date of current version March 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902631

Marginal Deep Architecture: Stacking Feature Learning Modules to Build Deep Learning Models

GUOQIANG ZHONG[1], (Member, IEEE), KANG ZHANG[1], HONGXU WEI[1], YUCHEN ZHENG[2], AND JUNYU DONG[1], (Member, IEEE)

[1]Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China
[2]Department of Advanced Information Technology, Kyushu University, Fukuoka 819-0395, Japan

Corresponding author: Guoqiang Zhong (gqzhong@ouc.edu.cn)

This work was supported in part by the National Key R&D Program of China under Grant 2016YFC1401004, in part by the National Natural Science Foundation of China under Grant 41706010, in part by the Science and Technology Program of Qingdao under Grant 17-3-3-20-nsh, in part by the CERNET Innovation Project under Grant NGII20170416, and in part by the Fundamental Research Funds for the Central Universities of China.

ABSTRACT Recently, many deep models have been proposed in different fields, such as image classification, object detection, and speech recognition. However, most of these architectures require a large amount of training data and employ random initialization. In this paper, we propose to stack feature learning modules for the design of deep architectures. Specifically, marginal Fisher analysis (MFA) is stacked layer-by-layer for the initialization and we call the constructed deep architecture marginal deep architecture (MDA). When implementing the MDA, the weight matrices of MFA are updated layer-by-layer, which is a supervised pre-training method and does not need a large scale of data. In addition, several deep learning techniques are applied to this architecture, such as backpropagation, dropout, and denoising, to fine-tune the model. We have compared MDA with some feature learning and deep learning models on several practical applications, such as handwritten digits recognition, speech recognition, historical document understanding, and action recognition. The extensive experiments show that the performance of MDA is better than not only shallow feature learning models but also related deep learning models in these tasks.

INDEX TERMS Deep architectures, feature learning, marginal Fisher analysis, marginal deep architecture.

I. INTRODUCTION

Deep learning models have achieved significant results in many tasks, such as image classification, document analysis and recognition, natural language processing and video analysis [1]–[5]. With multiple hidden layers, deep learning methods can explore the internal structure of high dimensional data and learn data representation with multiple levels of abstraction [6]. For example, in the face recognition applications, the learned features of the first layer may be the edges, directions and some local information. The second layer typically detects some object parts which are combination of the edges and directions. Higher layers may further abstract the face image by combining the features of previous layers (outlines of the eyes, noses, lips).

The associate editor coordinating the review of this manuscript and approving it for publication was Choon Ki Ahn.

In recent years, many deep learning models have been proposed [7]–[10]. Nevertheless, there are several complex problems to be solved, for example, some parameters need to be properly initialized, such as the weight matrix of two successive layers in deep belief networks (DBNs) and the convolution kernels in convolutional neural networks (CNNs). Furthermore, to get high performance, traditional deep learning methods need a large scale of data to train them. To the end, many problems emerge during the training process. If we don't initialize the parameters properly, the optimization procedure might need a long training time and fall into inferior local minima.

Alternatively, many feature learning methods have been proposed to learn the low-dimensional representation of high-dimensional data and avoid the curse of dimensionality. In particular, most of them can be trained with limited amount of data and their learning algorithms are generally based on

closed-form solution or convex optimization. For example, marginal Fisher analysis (MFA) is one of the feature learning methods that is supervised and based on the graph embedding framework [11], [12]. It utilizes an intrinsic graph to characterize the intra-class compactness, and another penalty graph to characterize the inter-class separability. The optimal solution of MFA can be learned by generalized eigenvalue decomposition. Whereas, shallow feature learning models cannot achieve good performance if the structure of the data is highly nonlinear; on the other hand, the combinations of these shallow feature learning models have rarely been exploited to design deep models.

In order to simultaneously solve the existing problems in deep learning models and combine the advantages of feature learning models, we proposed a novel deep learning method based on stacked feature learning modules. Specifically, instead of using random initialization, stacked MFA layers are applied to initialize this deep architecture, so that the constructed deep learning models are called marginal deep architecture (MDA). At first, to increase the capacity of the architecture, we use a random weight matrix to project the input data to a higher dimensional space. Next, the stacked MFA layers are applied to learn the lower dimensional representations of the data layer by layer. At last, the softmax layer is connected to the final feature layer. During the implementation of MDA, we add some tricks in the training process to fine tune it, such as back propagation, dropout and denoising. We have compared MDA with some feature learning and deep learning models on different domains of datasets (including handwritten digits recognition, speech recognition, historical document understanding, image classification, action recognition and so on). Experiments show that the performance of MDA is better than not only shallow feature learning models, but also related deep learning models.

Please note that, although convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have played an important role in many image, video and natural language applications, feedforward neural networks are still important. For instance, they can be used to deal with vectorized data and as the fully connected layers of many deep learning architectures. Hence, how to design deep feedforward neural networks is still an important issue for the deep learning community.

The contributions of this work can be summarized as follows:

1. We propose a novel deep architecture called MDA. The neurons in the first hidden layer of MDA are twice or quadruple of that in the input layer. Next, several layers of feature learning models are stacked to learn the low dimensional representations of the input data. In the end, a softmax classifier is applied.
2. In general, traditional deep learning models require a large amount of training data to obtain good results. Whereas, MDA can achieve better performance than these models with limited amount of training data due to the supervised pre-training method rather than random initialization.
3. Experiments demonstrate that the proposed MDA can obtain good results in several fields of data sets, such as natural images, spoken letters and handwritten digits. These results show that MDA is a general model to handle data sets with different scales of data. In addition, for large size images, combining convolutional operations and MDA, we can obtain competitive results with existing deep learning methods.

The rest of this paper is organized as follows: In Section II, we give a brief overview of related work. In Section III, we present the proposed marginal deep architecture (MDA) in detail. The experimental settings and results are reported in Section IV. At last, Section V concludes this paper with remarks and future work.

II. RELATED WORK

Since 2006, many deep learning models have been proposed. Primitively, Hinton and Salakhutdinov proposed the deep autoencoder (AE) that is an effective way to learn the low-dimensional representations of high-dimensional data [10]. Based on AE, Vincent *et al.* [13] proposed the denoising autoencoder (DAE), which made the learned representations robust to partial corruption of the input data. Subsequently, Vincent *et al.* extended DAE to stacked DAE (SDAE), which works very well on natural images and handwritten digits. To prevent the weights in deep neural networks from co-adaptation, Hinton *et al.* [14] introduced the dropout technique, which delivers new records for many speech and object recognition applications. However, due to numerous parameters, previous deep learning models generally need a large scale of training data to obtain good learning results.

In recent years, to address many vision problems, the research on deep convolutional neural networks (CNN) develops very fast [4], [15]–[18]. Specifically, in the research of image classification, Krizhevsky, Sutskever and Hinton proposed a large, deep convolutional neural network (AlexNet) to classify the 1.2 million high-resolution images in the ImageNet data set. The authors use efficient GPU to speed AlexNet. The results show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging data set using purely supervised learning [4]. In order to transfer a trained deep convolutional neural network to new tasks, Donahue *et al.* [15] proposed the deep convolutional activation feature (DeCAF), which is extracted from a well trained deep convolutional neural network on a large object recognition data set. DeCAF provides a uniform framework for researchers, who can improve and change this framework on some specific tasks. However, its performance at scene recognition has not attained the same level of success. In order to alleviate this problem, Zhou *et al.* [17] introduce a new scene-centric database called Places with over 7 million labeled pictures of scenes. Then, they learn the deep features for scene recognition tasks using deep architectures, and achieve

excellent results on several scene-centric datasets. However, these methods based on convolutional operation need very large scale of training samples and can not work well with limited amount of data.

In many domains other than computer vision, deep learning methods also achieve good performances. In [19], Hinton *et al.* represent the shared views of four research groups in using deep neural networks (DNNs) for automatic speech recognition (ASR). The DNNs that contain many layers of nonlinear hidden units and a very large output layer can outperform Gaussian mixture models (GMMs) at acoustic modeling for speech recognition on a variety of data sets. In the area of genetics, Xiong *et al.* [20] use deep learning algorithms to derive a computational model that takes DNA sequences as input to predict splicing in human tissues. It reveals the genetic origins of disease and how strongly genetic variants affect RNA splicing. In the area of natural language understanding, deep learning models have delivered strong results on topic classification, sentiment analysis and so on. Amongst others, Sutskever *et al.* [21] proposed a general approach, multilayered long short-term memory (LSTM), which can solve the general sequence to sequence problems better than before.

On the other hand, in the field of feature learning models, dimensionality reduction plays a crucial role to handle the problems for visualizing high-dimensional data and avoiding the ''curse of dimensionality'' [22], [23]. Traditional dimensionality reduction can mainly be classified by three criteria: linear or nonlinear, e.g., principal components analysis (PCA) [24] and linearity preserving projection (LPP) [25] are linear methods, while stochastic neighbor embedding (SNE) [26] is a nonlinear method; supervised or unsupervised, e.g., marginal Fisher analysis (MFA) [11], [12] and linear discriminant analysis (LDA) [27] are supervised methods, and PCA is an unsupervised method; local or global, e.g., MFA and SNE are local methods, and PCA is a global method. Many feature learning models provide excellent solutions for the applications of dimensionality reduction. However, for large scale complex problems, feature learning models may not perform well. Considering this situation, we try to select some well-behaved feature learning models and combine them to deep architectures. Among others, MFA is one special formulation of the graph embedding framework [11]. It utilizes an intrinsic graph to characterize the intra-class compactness, and another penalty graph to characterize the inter-class separability. Our motivation of this work is to combine the advantage of MFA and deep architectures and propose a new supervised initialization method for deep learning algorithms.

There are also some work about feature learning models based on the deep architectures [28]–[30]. Yuan *et al.* [28] proposed an improved multilayer learning model to solve the scene recognition task. This model learn all features used for scene recognition in an unsupervised manner. George *et al.* [29] proposed the deep semi-Non-negative matrix factorization (NMF), which is able to learn hidden representations from different, unknown attributes of a given dataset. Whereas, this model is proposed for learning low-dimensional representations that are better suited for clustering. Ngiam [30] proposed a deep architecture, which is an unsupervised model to learn feature representations over multiple modalities. They argued that multi-modality feature learning is better than one modality and achieved good performance on video and audio data sets.

In this work, we combine the advantages of feature learning models and deep architectures [31], [32], which stack MFA to initialize the deep architecture as a supervised pre-training method. Then, we employ some deep learning techniques, like back propagation, denoising and dropout to fine-tune the network. The advantage of this deep architecture is that we can learn the desirable weight matrix even if the training data is not large enough. And compared with traditional deep learning models and shallow feature learning models, the proposed method perform better than them in most cases.

III. MARGINAL DEEP ARCHITECTURE (MDA)

In this section, we present an innovative architecture of deep learning models first; After that, we introduce the proposed marginal deep architecture (MDA) in detail. In addition, some deep learning techniques used for the training of MDA, including back propagation, denoising and dropout, are also presented.

A. A NOVEL FRAMEWORK OF DEEP ARCHITECTURE

The target of feature learning can be described as follows. If there are n input data, $\mathcal{X} = \{\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T\} \in R^D$, where D is the dimensionality of the data space. The learning objective is to search for the compact representations of these data, i.e. $\mathcal{Y} = \{\mathbf{y}_1^T, \ldots, \mathbf{y}_n^T\} \in R^d$, where d is the dimensionality of the low dimensional embeddings.

In this paper, we consider the feature learning problems from the perspective of deep learning, and stack shallow feature learning modules to build deep networks [31], [32]. In this case, the data maps from the original D-dimensional space to the d-dimensional space layer by layer. This deep architecture can be seen as a general framework for data representation learning. The data flow in the deep architecture can be abstracted as

$$D \Longrightarrow D_1 \Longrightarrow \cdots \Longrightarrow D_i \Longrightarrow \cdots \Longrightarrow D_{p-1} \Longrightarrow d, \quad (1)$$

where D_1 is the dimensionality of a high dimensional space mapped from the original space. In order to increase the capacity of the network, it is twice or quadruple as many as those neurons in the input layer. D_i stands for the dimensionality of the i-th intermediate representation space, and p is the total stages of mappings. In this framework, feature learning modules with various output dimensions are applied to learn the representations of data in each layer. The mapping functions between consecutive layers are obtained by the layer by layer optimization of the feature learning models. The framework of the proposed deep architectures is briefly

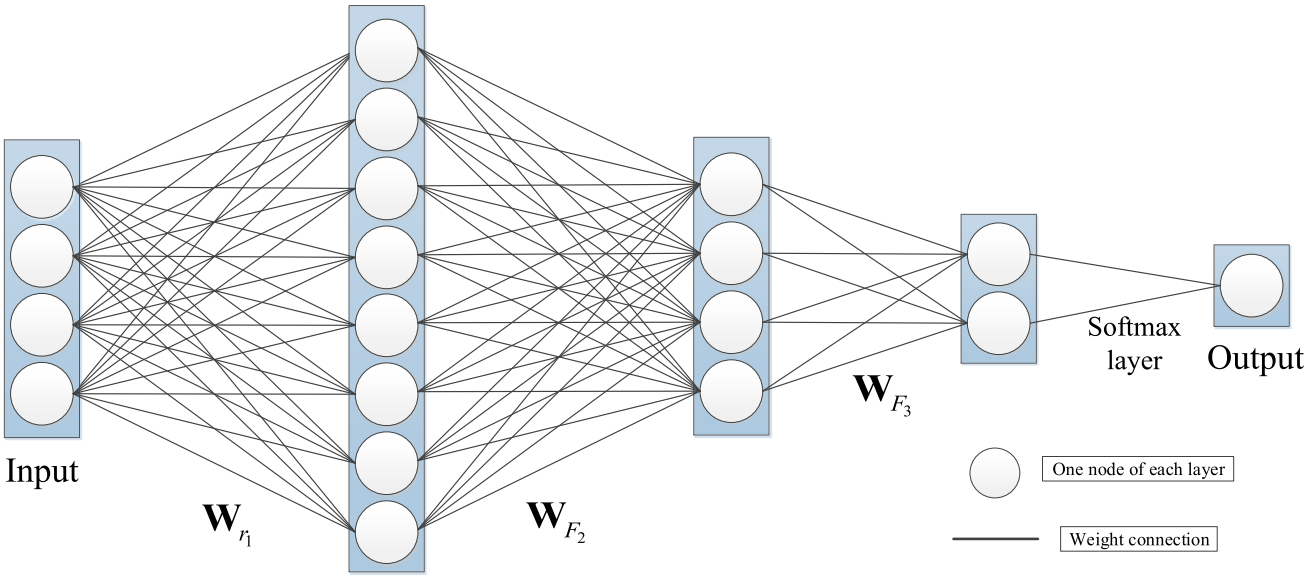


FIGURE 1. The uniform framework of the proposed deep architectures. $\mathbf{W}_{r_1}$ represents the first layer random weight matrix, while $\mathbf{W}_{F_2}$ and $\mathbf{W}_{F_3}$ represent the weight matrices learned by feature learning models. For simplicity, the bias terms are omitted.

presented in Fig. 1. In this figure, the first hidden layer is randomly initialized by matrix $\mathbf{W}_{r1}$, and the new representation of an input $\mathbf{x}$ can be written as

$$\mathbf{a}^1 = g(\mathbf{W}_{r_1}^T \mathbf{x} + \mathbf{b}), \tag{2}$$

where $g(.)$ is a non-linear activation function. After that, MFA or other feature learning models are used to initialize the subsequent layers. The outputs of the subsequent hidden layers are

$$\mathbf{a}^k = g(\mathbf{W}_{F_{k-1}}^T \mathbf{a}^{k-1} + \mathbf{b}). \tag{3}$$

For example, in Fig. 1, $\mathbf{W}_{F_2}$ and $\mathbf{W}_{F_3}$ are the weight matrices of the second layer and third layer learned from feature learning models. In the end, softmax regression is adopted as the last layer for the classification tasks. In the first hidden layer, the higher dimensional representations of input data can be learned. Afterwards, the following feature learning models can learn the lower dimensional embeddings step by step. The key point of this network is that the weight matrices of the hidden layers are initialized by feature learning modules except the matrix in the first hidden layer, which may deliver a better performance than other deep learning models initialized by random matrices.

B. MARGINAL FISHER ANALYSIS (MFA)

Based on our novel framework of deep architecture, we employ marginal Fisher analysis (MFA) to construct MDA. There are several advantages to use MFA. Compared with many traditional feature learning models, such as linear discriminant analysis (LDA), there is no assumption about the data distribution of each class, so that MFA is more general for discriminant analysis. In addition, the margins between classes can properly characterize the separability of the classes. Therefore, we apply MFA as the building blocks of MDA.

MFA follows the graph embedding framework to construct an intrinsic graph that characterizes the intra-class compactness and another penalty graph that characterizes the inter-class separability [11]. Suppose the input data is $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and the projection matrix is $\mathbf{W} = \{\omega_1, \ldots, \omega_d\}$. The intrinsic graph aims to connect each sample to its k-nearest neighbors in the same class. Suppose that we use $\mathcal{N}(i)$ to denote the k-nearest neighbors of $\mathbf{x}_i$ in its class. The intra-class compactness can be described as

$$\tilde{S_w} = \sum_i \sum_{j \in \mathcal{N}(i) \vee i \in \mathcal{N}(j)} \left\| \omega^T \mathbf{x}_i - \omega^T \mathbf{x}_j \right\|^2 \tag{4}$$

$$= \sum_i \sum_j \left\| \omega^T \mathbf{x}_i - \omega^T \mathbf{x}_j \right\|^2 \mathbf{A}_{ij} \tag{5}$$

$$= 2\omega^T \mathbf{X}(\mathbf{D} - \mathbf{A})\mathbf{X}^T \omega, \tag{6}$$

where $\mathbf{D}$ is a diagonal matrix with elements $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. The adjacency matrix $\mathbf{A}$ is given by

$$\mathbf{A}_{ij} = \begin{cases} 1, & \textit{if } j \in \mathcal{N}(i) \textit{ or } i \in \mathcal{N}(j), \\ 0, & \textit{otherwise}. \end{cases} \tag{7}$$

The penalty graph connects the marginal point pairs of different classes. We use $\mathcal{M}(C)$ to denote a set of input pairs that are k-nearest pairs among the set $\{(i,j)|i \in C \wedge j \notin C\}$, where C is class of an input $\mathbf{x}_i$. The inter-class separability can be defined as

$$\tilde{S_b} = \sum_i \sum_{(i,j) \in \mathcal{M}(C_i) \vee (j,i) \in \mathcal{M}(C_j)} \left\| \omega^T \mathbf{x}_i - \omega^T \mathbf{x}_j \right\|^2 \tag{8}$$

$$= \sum_i \sum_j \left\| \omega^T \mathbf{x}_i - \omega^T \mathbf{x}_j \right\|^2 \mathbf{A}_{ij}^p \tag{9}$$

$$= 2\omega^T \mathbf{X}(\mathbf{D}^p - \mathbf{A}^p)\mathbf{X}^T \omega, \tag{10}$$

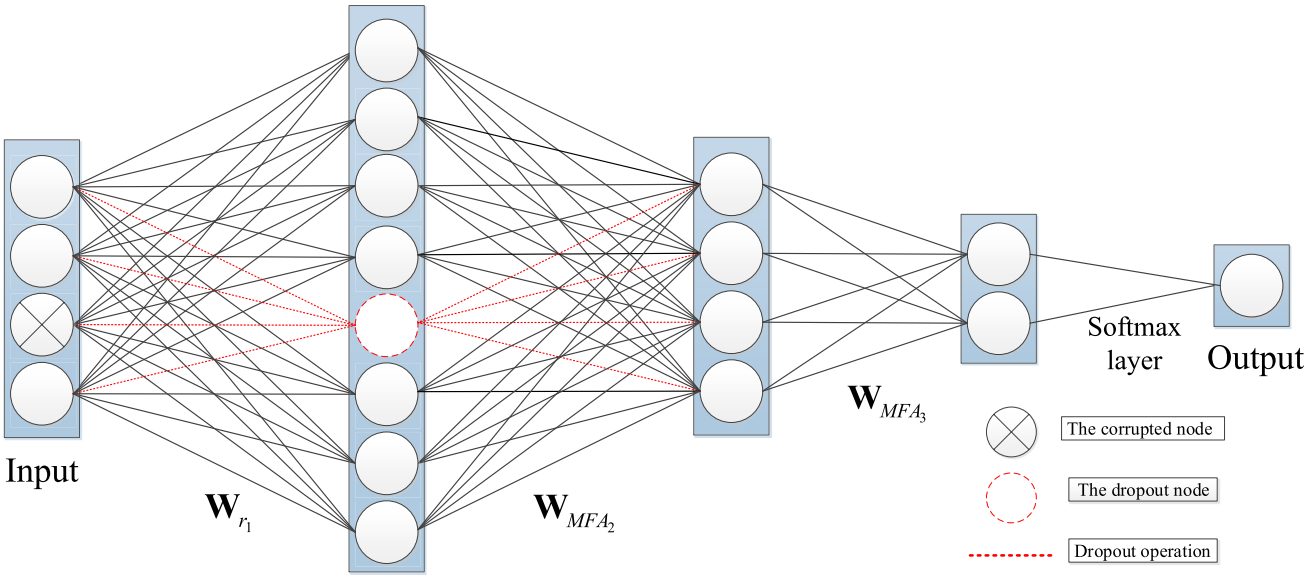


FIGURE 2. A brief representation of MDA. $\mathbf{W}_{r_1}$ stands for the first layer random weight matrix, while $\mathbf{W}_{MFA_2}$ and $\mathbf{W}_{MFA_3}$ represent the weight matrices learned by MFA. The dotted red lines represent the dropout operation, the dotted red circle is the dropout node, and the cross nodes are corrupted. The denoising and dropout operation are completely random. For simplicity, the bias terms are omitted.

where $\mathbf{D}^p$ is a diagonal matrix with elements $\mathbf{D}^p_{ii} = \sum_j \mathbf{A}^p_{ij}$. The adjacency matrix $\mathbf{A}^p$ is given by

$$\mathbf{A}^p_{ij} = \begin{cases} 1, & if\ (i,j) \in \mathcal{M}(C_i)\ or\ (j,i) \in \mathcal{M}(C_j), \\ 0, & otherwise. \end{cases} \tag{11}$$

The target of MFA is to minimize the intra-class compactness and maximize the inter-class separability simultaneously. Therefore, the marginal Fisher criterion is defined as

$$\mathbf{W}_{MFA} = \underset{\mathbf{W}}{argmin} \frac{\mathrm{tr}(\mathbf{W}^T\mathbf{X}(\mathbf{D}-\mathbf{A})\mathbf{X}^T\mathbf{W})}{\mathrm{tr}(\mathbf{W}^T\mathbf{X}(\mathbf{D}^p-\mathbf{A}^p)\mathbf{X}^T\mathbf{W})}. \tag{12}$$

In [11], to apply MFA for face recognition applications, the faces are firstly projected into a PCA subspace by the transformation matrix $\mathbf{W}_{PCA}$ to reduce noise. Since the features are learned by multiple layers in MDA and the whole deep architecture is fine-tuned by back propagation, it is not necessary to reduce the dimension of data by PCA at first. Hence, we compute the projection matrix $\mathbf{W}_{MFA}$ directly using Eq. (12) at each layer.

Here, MFA is used as the initialization method of the weight matrices in MDA. For different layers of MDA, the input $\mathbf{X}$ of $\mathbf{W}_{MFA}$ in Eq. (12) is the output of its previous layer. For example, in Fig. 2, $\mathbf{W}_{MFA_2}$ and $\mathbf{W}_{MFA_3}$ are computed using the output of their previous layers. In addition, the weight matrices calculated by MFA are only applied to initialize the weight matrices of MDA at the first iteration. Then, we apply back propagation to fine-tune these matrices.

C. MARGINAL DEEP ARCHITECTURE (MDA)

Based on the novel deep architecture framework and the benefits of MFA, we present MDA in the following. As depicted in Fig. 2, MDA is constructed by integrating MFA into the novel framework. Given an input vector $\mathbf{x} \in [0,1]^d$, it is firstly mapped to a higher dimensional space by a random weight matrix $\mathbf{W}_{r1}$. The activation output of the first hidden layer can be written as

$$\mathbf{a}^1 = s(\mathbf{W}^T_{r_1}\mathbf{x} + \mathbf{b}), \tag{13}$$

where $s(.)$ is the sigmoid function $s(x) = \frac{1}{1+e^{-x}}$, $\mathbf{b}$ is the bias terms, and $\mathbf{a}^1$ is the output of the first hidden layer. From the second layer to the $(n-1)$-th layer, the weight matrices are learned by MFA to initialize MDA layer by layer.

$$\mathbf{a}^k = s(\mathbf{W}^T_{MFA_{k-1}}\mathbf{a}^{k-1} + \mathbf{b}). \tag{14}$$

We use the softmax regression as the last layer of MDA for classification tasks, so that the number of neurons is the same as the number of classes. The cost function can be defined as

$$J(\mathbf{w}) = -\frac{1}{N}(\sum_{i=1}^{N}\sum_{j=1}^{K}\mathbf{I}(y_i = j)\log\frac{\exp(\mathbf{w}_j^T\mathbf{a}_i^{n-1})}{\sum_{l=1}^{K}\exp(\mathbf{w}_l^T\mathbf{a}_i^{n-1})}), \tag{15}$$

where N and K are the total number and class number of the input data, respectively. $\mathbf{I}(x)$ is the indicator function. If x is true, $\mathbf{I}(x) = 1$, else $\mathbf{I}(x) = 0$. y_i is the label of $\mathbf{x}_i$. $\mathbf{w}_j$ and $\mathbf{w}_l$ are weight vectors corresponding to class j and l. Hence, the probability that $\mathbf{x}_i$ is correctly categorized to class j is

$$p(y_i = j|\mathbf{x}_i, \mathbf{w}) = \frac{\exp(\mathbf{w}_j^T\mathbf{a}_i^{n-1})}{\sum_{l=1}^{K}\exp(\mathbf{w}_l^T\mathbf{a}_i^{n-1})}. \tag{16}$$

From the $(n-1)$-th layer to the last layer, we continue to use MFA to map it. To then end, we can consider that the MDA is initialized with a supervised pre-training method.

D. DEEP LEARNING TECHNIQUES APPLIED TO MDA

In order to improve the performance of MDA, we adopt some deep learning techniques to fine-tune MDA, including back propagation, denoising and dropout.

1) BACK PROPAGATION

Back propagation [33] is an efficient optimization algorithm to optimize MDA, which employs stochastic gradient descent to learn the weight matrices and the bias terms layer by layer. For every neuron i in the output layer (n-th layer), the error term can be described as

$$\delta_i^n = \frac{\partial J(\mathbf{w})}{\partial a_i^n} = -\frac{1}{N}\sum_{i=1}^{N}[(\mathbf{I}(y_i = j) - p(y_i = j|\mathbf{x}_i, \mathbf{w})] \quad (17)$$

where $J(\mathbf{w})$ is the cost function computed from (15), and a_i^n is the output of neuron i in the output layer. For every neuron i from the $(n-1)$-th layer to the second layer, the calculation of the error term is

$$\delta_i^k = (\sum_{j=1}^{k+1} \mathbf{w}_{ji}^k \delta_j^{k+1}) s'(a_i^k). \quad (18)$$

Then, the $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_{ij}^k}$ and $\frac{\partial J(\mathbf{w})}{\partial b_i^k}$ are calculated as,

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_{ij}^k} = a_j^k \delta_i^{k+1}, \quad (19)$$

$$\frac{\partial J(\mathbf{w})}{\partial b_i^k} = \delta_i^{k+1}. \quad (20)$$

By calculating the gradient of the cost function of MDA with respect to the parameters, the back propagation algorithm can update the weight matrices and bias terms in the layers of MDA. It starts from the output layer at the top and ends with the input layer at the bottom.

2) DENOISING OPERATION

Denoising is proposed in the denoising autoencoder to improve its robustness [13]. It can be viewed as a regularization method and avoids the ''overfitting'' problem. The main idea of it is that we can set the required proportion of ν ''destruction'' and corrupt partial input data. We can select a fixed percentage ν randomly for every input $\mathbf{x}$. The value of these inputs is fixed at 0, while the others remain unchanged. A partially destroyed version $\tilde{\mathbf{x}}$ of an initial input $\mathbf{x}$ can be obtained through a stochastic mapping,

$$\tilde{\mathbf{x}} \sim q_D(\tilde{\mathbf{x}}|\mathbf{x}), \quad (21)$$

where $q_D(\tilde{\mathbf{x}}|\mathbf{x})$ is the unknown data distribution. Hence, a hidden representation $\mathbf{h}$ can be computed as

$$\mathbf{h} = s(\mathbf{W}^T\tilde{\mathbf{x}} + \mathbf{b}). \quad (22)$$

In MDA, we use denoising to improve its performance. A concrete illustration can be seen in Fig. 2. With the denoising operation, the output of the first hidden layer is computed as

$$\mathbf{a}^2 = s(\mathbf{W}_{r_1}^T\tilde{\mathbf{x}} + \mathbf{b}_1), \quad (23)$$

where $\mathbf{W}_{r_1}$ and $\mathbf{b}_1$ are the random weight matrix and the bias term in the first hidden layer. The ''denoising'' method is proposed based on a hypothetical criterion for network design: robust to partial destruction of the input data. This criterion implies that good internal representations can be learned from an unidentified distribution of the input data. Therefore, this method is a benefit to learn more robust structure and avoids the overfitting problems in most cases.

3) DROPOUT

Similar with denoising operation, dropout is an efficient method to prevent overfitting [14]. Dropout has a dramatic effect on the test set when a deep learning model is trained on a small training set. It is a regularization technique to prevent the complex co-adaptations on the training data. The key point of dropout is that each neuron in the hidden layers is randomly excluded from the model with a probability of β. Besides, dropout can be seen as an efficient way of performing model averaging with deep learning models. Fig. 2 depicts the dropout operation in MDA.

IV. EXPERIMENTS AND DISCUSSIONS

To evaluate MDA, we performed several experiments on different sizes of data sets. We designed MDA in different structures in order to explore the optimal architecture of it. At first, we tested the MDA on five benchmark data sets to explore the best architecture of MDA and compared it with other feature learning and deep learning models. Then, in order to show the performance of MDA initialized by the supervised initialization method on different sizes of data sets, we applied MDA to a specific dataset with extremely limited data, CMU mocap, and a relatively large data set, CIFAR-10. In addition, we combined MDA with convolutional neural network (CNN) for addressing image classification tasks, and used the supervised initialization method in the pre-training phrase of deep CNNs on the CIFAR-10 data set.

A. DATE SET DESCRIPTIONS

We first evaluated the MDA on five benchmark datasets. Table 1 illustrates some characteristics of these data sets. The **USPS**[1] data set is composed of handwritten digits images, which contains 7291 training samples and 2007 test samples from 10 classes and each sample is represented with a 256 dimensional vector. The task is to identify the digits from 0 to 9. The **Isolet**[2] data set contains 6238 training samples and 1559 test samples from 26 classes with 614 dimensional features. It collects audio feature vectors of spoken letters from the English alphabet. Based on the recorded (and pre-processed) audio signals, the task aims to identify the exact letter which is spoken. **Sensor**[3] is a sensorless drive diagnosis data set, including 46816 training samples and 11693 test

[1] http://www.gaussianprocess.org/gpml/data/

[2] http://archive.ics.uci.edu/ml/datasets/ISOLET

[3] http://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis#

TABLE 1. Characteristics of the used data sets.

dataset	n	train	test	$\|\mathcal{Y}\|$	D(d)
USPS	9298	7291	2007	10	256(32)
Isolet	7797	6238	1559	26	614(308)
Sensor	58509	46816	11693	11	48(24)
Covertype	581012	15120	565892	7	54(27)
Ibnsina	20668	17543	3125	174	200(100)
CMU mocap	49	44	5	3	93(24)
CIFAR-10	60000	50000	10000	10	3072(64)

samples from 11 classes. Each one of the samples contains 48 dimensional features, which are extracted from electric current drive signals. The target is to classify the specific category through different conditions of the drive and its intact and faulty components. **Covertype**[4] is a geological and map-based data set chosen from four wilderness areas located in the Roosevelt National Forest of northern Colorado. It contains 15120 training samples and 565892 test samples from 7 classes with 54 dimensional features. The target is to recognize the categories of forest cover from cartographic variables. **IbnSina**[5] is an ancient Arabic document data set, we select 50 pages of the manuscript for training (17543 training samples) and 10 pages for testing (3125 test samples). There are 174 classes of subwords with 200 dimensions in this dataset.

In addition, we also tested MDA on a specific task, which uses the CMU motion capture (**CMU mocap**) data set.[6] The CMU mocap data set includes three categories, namely, jumping, running and walking. We chose 49 video sequences from four subjects. For each sequence, the features are generated using Lawrence's method,[7] with dimensionality 93 [34]. By reason of the few samples of this data set, we adopt 10-fold cross-validation in our experiments and use the average error rate and standard deviation to evaluate the performance. At last, we test MDA on a classic data set **CIFAR-10**[8] to test the performance of MDA on image classification applications. Furthermore, we combined MDA with CNN, and evaluated this model on CIFAR-10. The CIFAR-10 data set includes 60000 32 $\times$ 32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Fig. 5 shows some examples of the CIFAR-10 data set from the 10 categories.

B. CLASSIFICATION ON FIVE BENCHMARK DATA SETS

In this experiments, we compare MDA with several related deep learning models on the 5 benchmark data sets. These deep learning models include autoencoder (AE) [10], stacked autoencoders (SAE), denoising autoencoders (DAE) [13], stacked denoising autoencoders (SDAE) [35] and denoising autoencoders with dropout (DAE(dropout)) and a variant of MDA, PDA. Note that the architecture of PDA is the same as MDA but the feature learning module of PDA is PCA [24] instead of MFA [11], [12].

1) EXPERIMENTAL CONFIGURATIONS

All of these deep learning models have the same structure and configurations. The size of minibatch was set to 100, the learning rate and momentum were set to the default value 1 and 0.5, the number of epoch was set to 400, while the dropout rate β and the denoising rate ν were set to 0.1. In AE and SAE, the weight penalty of the $L2$ norm was set to 10^{-4}. For MFA, the number of nearest neighbors for constructing the intrinsic graph was set to 5, while that for constructing the penalty graph was set to 20. The target dimensions of data representations in MDA and PDA on these data sets are shown in the last column of Table 1.

2) CLASSIFICATION RESULTS

From the experimental results shown in Table 2, we can see that MDA performs best on four data sets except the Sensor data set, compared with other models. Meanwhile, it obtained the second best result on the Sensor dataset. Furthermore, PDA achieved the best result on the Sensor data set and the second best results on the other data sets. These results demonstrate that in most cases, the proposed deep learning models can achieve good performances on data sets with limited amount of training data. We can also conclude that the performance of MDA is better than not only the related deep learning models, but also some shallow feature learning methods, such as PCA and MFA as shown in Table 2. These results demonstrate that MDA and PDA based on stacked some feature learning models can learn better representations of the input data than shallow feature learning methods. However, it is not always true that deep learning models perform better than the feature learning models. For instance, the performance of MFA is better than AE, DAE, DAE with dropout and SDAE on the Sensor data set. This indicates that training with a limited amount of data, some feature learning methods may learn the representations of the input data better than deep learning models. MDA possesses the advantages of both deep learning and feature learning models. Experimental results show the advantages of MDA on data sets with limited amount of training data.

3) TIME CONSUMPTION

The time consumption on training and test process for 7 different deep architectures are shown in Table 3. Each experiment was carried out 5 times and the averaged results are reported. Note that all the experiments were performed on a 4-core intel(R) Core(TM)2 Quad Q9550 CPU with 2.83GHz clock frequency. We can see that the training times of PDA and MDA are similar with AE, DAE and DAE with dropout. However, they are much faster than SAE and SDAE. On the Isolet data set, the time consumptions of PDA and MDA are less than other deep architectures. This demonstrates that

[4]http://archive.ics.uci.edu/ml/datasets/Covertype
[5]http://www.causality.inf.ethz.ch/al_data/IBN_SINA.html
[6]http://http://mocap.cs.cmu.edu/
[7]http://is6.cs.man.ac.uk/~neill/mocap/
[8]http://www.cs.toronto.edu/ kriz/cifar.html

TABLE 2. The classification accuracy on five benchmark data sets. "ORIG" represents the results obtained by applying softmax directly to the original data space. The best result is highlighted with boldface.

Method	ORIG	PCA	MFA	AE	SAE	DAE(dropout)	DAE	SDAE	PDA	MDA
USPS	0.8366	0.9402	0.9392	0.9402	0.9402	0.9581	0.9532	0.9452	0.9586	**0.9601**
Isolet	0.9467	0.9237	0.9269	0.9519	0.9506	0.9596	0.9519	0.9543	0.9584	**0.9622**
Sensor	0.8151	0.8042	0.8234	0.7995	0.8325	0.8178	0.7764	0.7870	**0.8582**	0.8558
Covertype	0.5576	0.5596	0.6057	0.7405	0.5576	0.7093	0.7397	0.7440	0.7458	**0.7589**
Ibnsina	0.8957	0.9190	0.9206	0.9363	0.9184	0.9402	0.9370	0.9261	0.9421	**0.9491**

TABLE 3. Time consumption of 7 compared deep architectures. The test time is on all the test samples.

Methods	Training Times					Test Times				
	USPS	Isolet	Sensor	Covertype	Ibnsina	USPS	Isolet	Sensor	Covertype	Ibnsina
AE	24m 26s	76m 57s	16m 25s	9m 24s	32m 20s	0.16s	0.47s	0.13s	9.18s	0.20s
DAE	23m 21s	75m 12s	16m 4s	8m 54s	44m 38s	0.19s	0.48s	0.14s	9.34s	0.26s
DAE(dropout)	24m 50s	78m 56s	19m 1s	11m 13s	48m 15s	0.19s	0.53s	0.15s	9.16s	0.30s
SAE	61m 32s	201m 28s	43m 2s	30m 9s	77m 50s	0.18s	0.42s	0.13s	9.85s	0.18s
SDAE	61m 34s	214m 16s	46m 11s	31m 49s	74m 43s	0.19s	0.46s	0.14s	9.24s	0.18s
PDA	25m 13s	74m 9s	18m 53s	11m 11s	47m 29s	0.19s	0.47s	0.14s	9.62s	0.24s
MDA	29m 55s	75m 38s	19m 22s	16m 44s	48m 47s	0.18s	0.51s	0.14s	9.35s	0.23s

TABLE 4. The structures of MDA on the 5 benchmark data sets. "None" represents without second layer in MDA. "Twice" means the second layer's nodes are as twice as the input layer. "Quadruple" represents the second layer's nodes are as quadruple as the input layer. "Octuple" represents the second layer's nodes are as octuple as the input layer.

Dataset	None	Twice	Quadruple	Octuple
USPS	256-128-64-32	256-512-256-128-64-32	256-1024-512-256-128-64-32	256-2048-1024-512-256-128-64-32
Isolet	617-308	617-1324-617-308	617-2648-1324-617-308	617-5296-2648-1324-617-308
Sensor	48-24	48-96-24	48-192-96-24	48-384-192-96-24
Covertype	54-27	54-108-27	54-216-108-54-27	54-432-216-108-54-27
Ibnsina	200-100	200-400-200-100	200-800-400-200-100	200-1600-800-400-200-100

PDA and MDA can sometimes achieve good results with short training time because of their efficient weights initialization. The training periods of SAE and SDAE are very slow because every layer in SAE and SDAE is an autoencoder layer, and it requires a long optimization time for initializing the weight matrix. During the test procedure, all the methods have similar efficiency as their architectures are the same. These results show the efficiency of MDA. It's mainly because of that it can achieve good initial weight matrices by a short time and perform well on the learning tasks.

4) THE DENOISING AND DROPOUT RATIOS

In order to evaluate the influence of the denoising and dropout operations on MDA, we designed an experiment on the 5 benchmark data sets with different denoising ratios and dropout ratios. Firstly, we fixed the dropout ratio at 0.1, then adjusted the denoising ratio from 0.1 to 0.5. Next, we fixed the denoising ratio at 0.1, and modified the dropout ratio from 0.1 to 0.5. These experimental results are shown in Fig. 3. We can see that MDA achieved minimum error on all data sets when denoising ratio and dropout ratio are 0.1. The error was increasing with the increasing of denoising ratio and dropout ratio in most cases. For the USPS data set, the error decreased when dropout ratio was changed from 0.2 to 0.3. For the Sensor data set, the error decreased slightly when denoising ratio was changed from 0.2 to 0.3. The experimental results demonstrate that denoising and dropout operations can improve the performance of MDA when selecting appropriate values for them. Without the denoising and dropout operations, the experimental results is not as good as adopting these operations. In the following experiments, we set both of them to 0.1.

5) DIFFERENT STRUCTURES FOR MDA

In order to explore better structures of MDA, we constructed different structures of it by changing the number of nodes in each layer. For the USPS data set, we first got rid of the second layer, the structure of the model was $256 - 128 - 64 - 32$. Then, we set the number of the nodes in the second layer to twice of that in the input layer, so that the architecture changed to $256 - 512 - 128 - 64 - 32$. Next, the nodes in the second layer were quadruple as many as those in the input layer, and the architecture became to $256 - 1024 - 512 - 256 - 128 - 64 - 32$. Finally, the nodes were octuple as many as those in the input layer, so that the architecture was $256 - 2048 - 1024 - 512 - 256 - 128 - 64 - 32$. The structures of MDA on other data sets were changed similarly, as shown in Table 4.

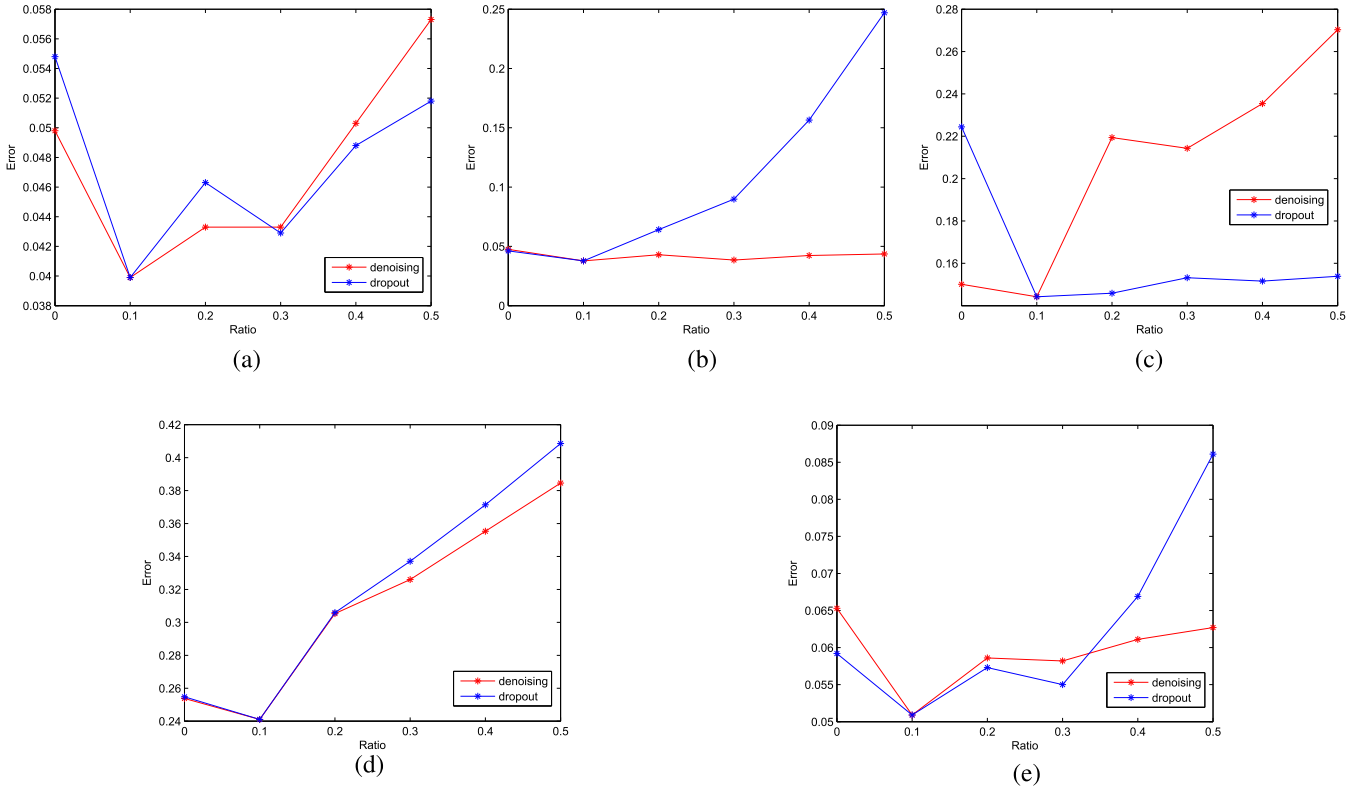


FIGURE 3. Error rates on 5 data sets with respect to different denoising ratios and dropout ratios. (a) USPS. (b) Isolet. (c) Sensor. (d) Covertype. (e) Ibnsina.

TABLE 5. The classification error with different structures of MDA on the 5 benchmark data sets. The best result (minimum error) is highlighted with boldface.

Dataset	None	Twice	Quadruple	Octuple
USPS	0.0463	**0.0399**	0.0433	0.0453
Isolet	0.0398	**0.0378**	0.0417	0.0430
Sensor	0.2172	**0.1442**	0.1559	0.7856
Covertype	0.3876	0.3878	**0.2411**	0.3806
Ibnsina	0.0643	**0.0509**	0.0614	0.0858

The experimental results with these different structures of MDA are shown in Table 5. We can see that MDA achieved the minimum classification error on all the data sets except the Covertype data set when the nodes in the second layer are twice of that in the input layer. Besides, MDA obtained the best performance on the Covertype data set when the nodes of the second layer are quadruple of that in the input layer. We can conclude that MDA can work well when the nodes in the second layer are twice or quadruple as many as the nodes in the input layer. Then we employed these structures that achieved best results to compare the performance of MDA with its related models on these data sets. In addition, except on the Covertype data set, when the nodes in the second layer increase from doubled nodes gradually, the errors increase at the same time.

6) DIFFERENT NUMBERS OF HIDDEN LAYERS FOR MDA

As a deep learning model, the depth of MDA is very important. With the architecture getting deeper and deeper, the training of the deep learning models may become more and more difficult. It means that we shall spend more computing resources if the architecture is very deep. In order to evaluate how many hidden layers are appropriate to different tasks, we designed different structures on the 5 benchmark data sets. We applied MDA with from 1 to 7 hidden layers on the USPS and Isolet data sets and from 1 to 5 hidden layers on the Covertype, Sensor and Ibnsina datasets. Other experimental settings are the same as previous experiments.

Table 6 shows the classification error on the 5 benchmark data sets with different numbers of hidden layers for MDA. All the data sets achieved the best results when the number of hidden layers is 3 except the USPS data set. On the USPS data set, MDA achieved the best result when the number of hidden layers was 5. When the number of hidden layers is from 1 to 3, with the increasing number of hidden layers, the classification error decreased on all the data sets. With limited amount of data, we don't need very deep architectures to handle them.

7) EFFECTS OF INITIALIZATION METHODS

To evaluate the effect of MFA as an effective network initialization method, we applied different network initialization methods to the same deep architecture on the 5 benchmark data sets. These initialization methods included

TABLE 6. The classification error on the 5 benchmark data sets with different structures of MDA.

The number of hidden layers	1	2	3	4	5	6	7
USPS	0.05780	0.05032	0.05082	0.05182	**0.03990**	0.05730	0.05132
Isolet	0.03977	0.04169	**0.03785**	0.03849	0.05452	0.04234	0.04683
Covertype	0.27171	0.25601	**0.24110**	0.26731	0.27491	–	–
Sensor	0.20457	0.15522	**0.14420**	0.17027	0.15567	–	–
Ibnsina	0.05696	0.05184	**0.05088**	0.06016	0.06720	–	–

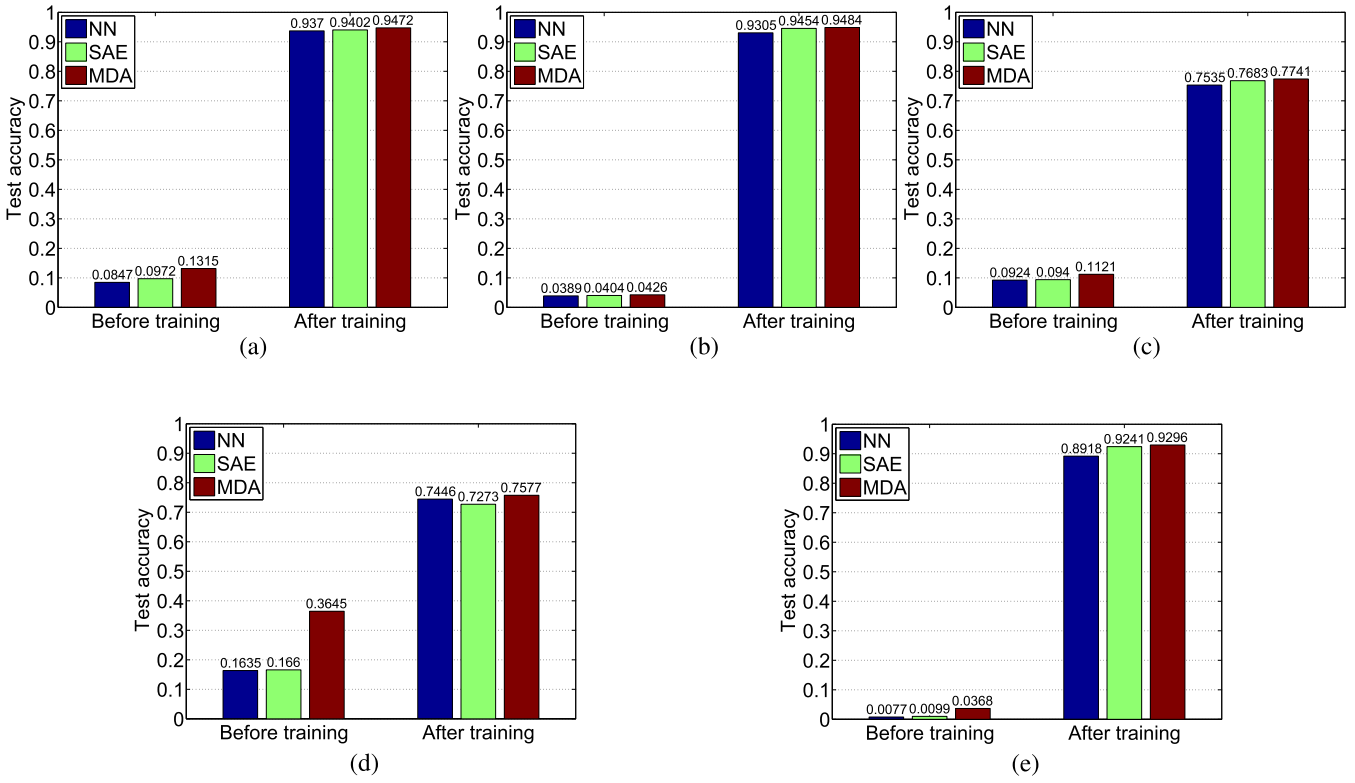


FIGURE 4. The test accuracies of MDA with different initialization methods on the 5 benchmark data sets. On each data set, the deep models have the same structure and hyperparameters. The classification results are evaluated only with initialization and after training the same epochs, respectively. (a) USPS dataset. (b) Isolet dataset. (c) Sensor dataset. (d) Covertype dataset. (e) Ibnsina dataset.

random initialization, SAE as the unsupervised pre-training method and MFA as the supervised pre-training method. Firstly, the deep architectures initialized by these methods are evaluated on the 5 benchmark data sets before the training phase. Then, we tested these models after the same training epochs on the same data sets. On each data set, the structures and experiment settings of these deep models were kept the same. The structures were $256-128-64-32$ for the USPS data set, $617-308$ for the Isolet data set, $48-24$ for the Sensor data set, $54-27$ for the Covertype data set and $200-100$ for the Ibnsina data set, respectively.

Fig. 4 illustrates the test accuracies of these models before and after the training period on the 5 benchmark data sets. NN is the neural network initialized by random initialization. SAE and MDA are the same neural network initialized by SAE and MFA, respectively. It is easy to see that, MDA achieved the best results in the pre-training period of the deep models without back propagation training and after the same training period. These experimental results demonstrate that compared with other initialization methods, the stacked MFA in MDA can initialize the deep architecture more effectively and obtain better performance after the same training epochs.

C. CLASSIFICATION ON THE CMU MOCAP DATASET

To test the performance of MDA on real world applications, we evaluated it on a specific data set, CMU mocap. CMU mocap is a very small data set including only 49 samples. Traditional deep learning methods cannot work well in this application. We compared MDA and PDA with PCA, MFA and other 5 deep learning models. The architectures of all the deep models (except the PDA) are $93-186-93-47-24$. Specially, since the CMU mocap data set only has 49 samples, the PCA method can only reduce the dimensionality

TABLE 7. The classification accuracy with standard deviation on the CMU mocap data set. The best results are highlighted with boldface.

Method	Accuracy
PCA	0.4061 ± 0.1540
MFA	0.4394 ± 0.1716
AE	0.6030 ± 0.1343
SAE	0.5894 ± 0.1648
DAE(dropout)	0.6030 ± 0.1343
DAE	0.5939 ± 0.1540
SDAE	0.6030 ± 0.1343
PDA	$\mathbf{0.6409 \pm 0.0815}$
MDA	$\mathbf{0.6364 \pm 0.0958}$

to 49 at most, so that the architecture of PDA was set to $93-186-49$. The denoising ratio and dropout ratio were set to 0.1 on all the deep learning models, including AE, DAE, DAE with dropout, SDAE, SAE, PDA and MDA. The weight penalty of AE was set to 10^{-4}. The learning rate was set to 0.01, the momentum was set to 0.5 and the number of epoch was set to 600. The experiment was tested based on 10-fold cross validation. The experimental results are shown in Table 7.

In Table 7, we can see that PDA and MDA achieved the best and second best results on this dataset and have lower standard deviation than other deep learning models. This demonstrates that PDA and MDA are more stable than other deep learning models. Moreover, deep learning models perform better than shallow feature learning models, such as PCA and MFA. Through these results, we can see that in applications with limited amount of training data, our proposed methods can achieve desired and stable results compared with other deep learning and feature learning models.

D. CLASSIFICATION ON THE CIFAR-10 DATA SET

In order to assess MDA on image classification applications, we chose CIFAR-10 as a relatively large scale data set to evaluate the performance of MDA.

In this section, we performed two experiments. In the first experiment, we tested MDA on the CIFAR-10 data set to test its performance on image classification applications. In the Second experiment, because MDA is a fully-connected network, we selected a CNN and replaced its fully-connected layers with MDA. This new model was named CNN-MDA. In this case, we tested the effect of MDA as a supervised initialization method in CNNs.

1) PERFORMANCE OF MDA ON THE CIFAR-10 DATA SET

In this experiment, we first transformed the color images in the CIFAR-10 data set to gray level images and named them gray CIFAR-10, so that the dimensionality of the images reduced to $32 \times 32 \times 1$. Fig. 5 shows some example images in the CIFAR-10 data set. Fig. 6 shows the corresponding

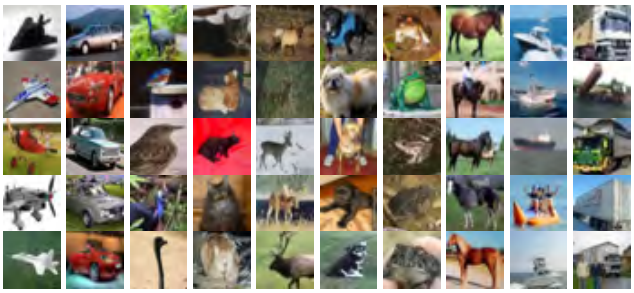


FIGURE 5. Example images in the CIFAR-10 data set. Each column corresponds to one category.

FIGURE 6. Gray level images corresponding to those shown in Fig. 5.

TABLE 8. Classification accuracy obtained on the CIFAR-10 data set.

Methods	Accuracy
PCA	0.3077
MFA	0.2655
AE	0.4883
SAE	0.4748
DAE(dropout)	0.4910
DAE	0.4824
SDAE	0.4887
PDA	**0.4915**
MDA	**0.5053**

gray level images. Then, we flattened each sample as a 1024 dimensional vector, which can be input to MDA. Based on the previous experiments, the architecture of MDA was designed as $1024-2048-1024-512-256-128-64$. The size of minibatch was set to 100. The dropout ratio and denoising ratio were set to 0.1, respectively. The number of epoch was set to 400. The learning rate was set to 1 and the momentum was set to 0.5. We compared MDA with 8 previous methods.

Table 8 shows the classification accuracy on gray CIFAR-10. Due to the high dimensionality and high complexity of the problem, PCA and MFA did not perform well. Furthermore, MDA achieved the best results in all the compared methods. However, due to working on gray level images and loss of the color and spatial information of the images, we didn't get the state-of-the-art result on this data set.

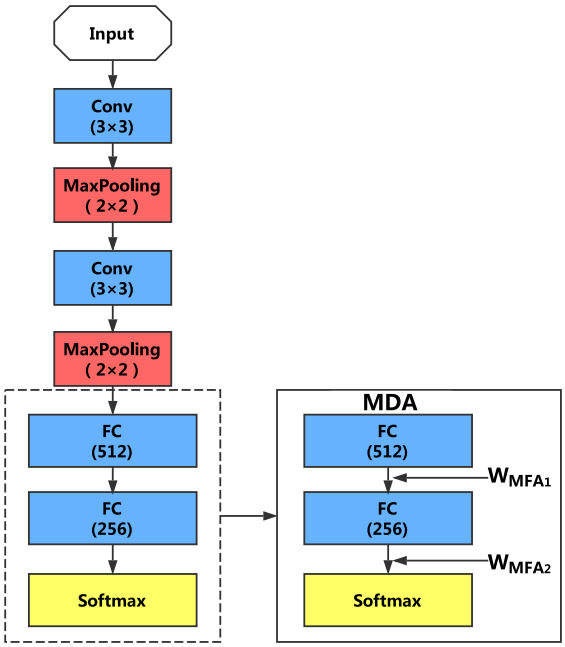


FIGURE 7. The architecture of CNN-MDA. The fully-connected part is replaced with MDA.

2) PERFORMANCE OF CNN-MDA ON THE CIFAR-10 DATA SET

In this experiment, we combined MDA and CNN by replacing the fully-connected part of CNN with MDA, which can be initialized by the supervised pre-training method. This model is named CNN-MDA and its structure is shown in Fig. 7. Compared with the original CNN, its fully connected part is initialized by MDA. Typically, we chose a classic CNN model, LeNet-5, and replaced its fully-connected part with MDA and named it LeNet-5-MDA. Note that the structures of CNN-MDA and LeNet-5-MDA remained the same as their original models. Then, we tested these CNNs on the CIFAR-10 data set without fine-tuning.

The performance of these models can be seen in Table 9. On the one hand, LeNet-5 and LeNet-5-MDA did not achieve good results on this data set, mainly because of their limited number of parameters. On the other hand, compared with their original architectures, both CNN-MDA and LeNet-5-MDA can optimize the original networks better and achieve higher accuracy by replacing the fully-connected parts with MDA. It can be concluded that the fully-connected parts of CNNs can be initialized effectively by MDA with the supervised pre-training method.

E. COMPARISON BETWEEN MDA AND DEEP JOINTLY INFORMED NEURAL NETWORKS (DJINN)

Deep jointly informed neural networks (DJINN) is an effective model, which can automatically construct feedforward neural networks based on decision trees [36]. Furthermore, it can initialize the constructed model by its tree-informed initialization method as a warm-start to the training process.

TABLE 9. Comparison between CNNs and CNN-MDA on the CIFAR-10 data set.

Methods	Accuracy
Lenet-5	0.4536
Lenet-5-MDA	**0.4894**
CNN	0.7099
CNN-MDA	**0.7236**

TABLE 10. Accuracy and computational cost of DJINN and MDA.

Datasets	Accuracy		Computational Cost	
	DJINN	MDA	DJINN	MDA
USPS	0.9297	**0.9601**	7m 4s	**19s**
Isolet	0.9305	**0.9622**	7m 47s	**7s**
Sensor	0.8211	**0.8558**	4m 50s	**10s**
Covertype	0.7447	**0.7589**	13m 47s	**31s**
Ibnsina	0.9338	**0.9491**	26m 26s	**57s**

DJINN has shown its high predictive performance for a variety of regression and classification tasks. Here, we construct these two models by the same number of layers and compare the test accuracies and computational cost of DJINN and the proposed MDA on the same data sets.

The initialization period of DJINN is to determine the architecture and weight utilizing the dependency structure of a decision tree trained on the data. At the same time, the initialization period of MDA is to compute the weight matrices using MFA. Since the main difference between DJINN and MDA is their initialization methods, the computational cost of the two models is evaluated with the time consumption of their initialization period under the same experiment conditions.

The experiment results are shown in Table 10. MDA achieved better results than DJINN on all the 5 benchmark data sets. In addition, the computational cost of MDA is much less than DJINN on these data sets. This is because MDA only needs to compute the weight matrices between different layers in the pre-training phase. It can be concluded that the performance of MDA is better than DJINN with less time consumption.

V. CONCLUSION

In this paper, we propose a novel deep learning architecture by stacking feature learning methods. We apply the feature learning method MFA to this framework and name it MDA. In this case, MDA can be initialized by a supervised pre-training method. Furthermore, some deep learning techniques, such as back propagation, denosing and dropout operation, are employed on MDA to improve its performance. Extensive experiments demonstrate that on data sets with limited amount of data, the performance of MDA is better than both shallow feature learning models and relevant deep learning models. Experiments on the CIFAR-10 data set show that MDA can be used in CNNs for the supervised initialization of their fully-connected layers. In the future work,

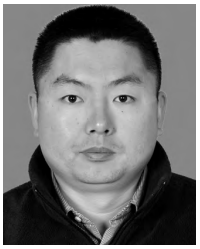
we intend to exploit some other feature learning methods for the deep architecture construction and explore the different structures of this novel deep learning framework.

ACKNOWLEDGMENT

The Titan X GPU used for this research was donated by the NVIDIA Corporation.

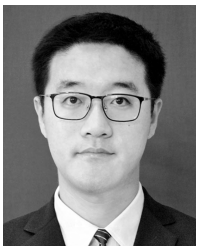
REFERENCES

[1] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. ICML*, 2008, pp. 160–167.
[2] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010.
[3] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Proc. CVPR*, Jun. 2011, pp. 3361–3368.
[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1106–1114.
[5] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A simple deep learning baseline for image classification?" *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.
[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[7] M. Ranzato, Y. Boureau, and Y. L. Cun, "Sparse feature learning for deep belief networks," in *Proc. NIPS*, 2007, pp. 1185–1192.
[8] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proc. NIPS*, 2009, pp. 1096–1104.
[9] H. Lee, R. B. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. ICML*, 2009, pp. 609–616.
[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[11] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[12] G. Zhong, Y. Chherawala, and M. Cheriet, "An empirical evaluation of supervised dimensionality reduction for recognition," in *Proc. ICDAR*, Aug. 2013, pp. 1315–1319.
[13] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. ICML*, 2008, pp. 1096–1103.
[14] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. (2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580
[15] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. ICML*, 2014, pp. 647–655.
[16] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. ICML*, 2015, pp. 97–105.
[17] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. NIPS*, 2014, pp. 487–495.
[18] G. Zhong, X. Ling, and L.-N. Wang, "From shallow feature learning to deep learning: Benefits from the width and depth of deep architectures," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 9, Jan./Feb. 2019, Art. no. e1255.
[19] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[20] H. Y. Xiong *et al.*, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, 2015, Art. no. 1254806.
[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.
[22] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," *J. Mach. Learn. Res.*, vol. 10, nos. 1–41, pp. 66–71, Oct. 2009.
[23] L. J. van der Maaten, "An introduction to dimensionality reduction using MATLAB," Univ. Maastricht, Maastricht, The Netherlands, Tech. Rep. MICC 07-07, 2007.
[24] I. T. Jolliffe, *Principal Component Analysis*. Hoboken, NJ, USA: Wiley, 2002.
[25] X. He and P. Niyogi, "Locality preserving projections," in *Proc. NIPS*, vol. 16, 2003, pp. 153–160.
[26] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Proc. NIPS*, 2002, pp. 833–840.
[27] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[28] Y. Yuan, L. Mou, and X. Lu, "Scene recognition by manifold regularized deep learning architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2222–2233, Oct. 2015.
[29] T. George, B. Konstantinos, Z. Stefanos, and B. W. Schuller, "A deep semi-NMF model for learning hidden representations," in *Proc. ICML*, 2014, pp. 1692–1700.
[30] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proc. ICML*, 2011, pp. 689–696.
[31] Y. Zheng, G. Zhong, J. Liu, X. Cai, and J. Dong, "Visual texture perception with feature learning models and deep architectures," in *Proc. CCPR*, 2014, pp. 401–410.
[32] Y. Zheng, Y. Cai, G. Zhong, Y. Chherawala, Y. Shi, and J. Dong, "Stretching deep architectures for text recognition," in *Proc. ICDAR*, 2015, pp. 236–240.
[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[34] G. Zhong, W.-J. Li, D.-Y. Yeung, X. Hou, and C.-L. Liu, "Gaussian process latent random field," in *Proc. AAAI*, 2010, pp. 1–6.
[35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[36] K. D. Humbird, J. L. Peterson, and R. G. McClarren, "Deep neural network initialization with decision trees," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. doi: 10.1109/TNNLS.2018.2869694.



GUOQIANG ZHONG (M'16) received the B.S. degree in mathematics from Hebei Normal University, Shijiazhuang, China, in 2004, the M.S. degree in operations research and cybernetics from the Beijing University of Technology, Beijing, China, in 2007, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2011.

From 2011 to 2013, he was a Postdoctoral Fellow with the Synchromedia Laboratory for Multimedia Communication in Telepresence, École de Technologie Supérieure, University of Quebec, Montreal, Canada. Since 2014, he has been an Associate Professor with the Department of Computer Science and Technology, Ocean University of China, Qingdao, China. He has published three books, four book chapters, and more than 50 technical papers in the areas of artificial intelligence, pattern recognition, machine learning, and data mining. His research interests include pattern recognition, machine learning, and image processing. He has served as a PC Member/Reviewer for many international conferences and top journals, such as the IEEE TNNLS, the IEEE TKDE, the IEEE TCSVT, *Pattern Recognition*, *Knowledge-Based Systems*, *Neurocomputing*, ACM TKDD, ICPR, and ICDAR. He is a member of the ACM and IAPR and a Professional Committee Member of CAAI-PR, CAA-PRMI, and CSIG-DIAR. He has been awarded as an Outstanding Reviewer by several journals, such as *Pattern Recognition*, *Neurocomputing*, and *Cognitive Systems Research*.



KANG ZHANG received the B.S. degree in communication engineering from the Ocean University of China, Qingdao, China, in 2014, where he is currently pursuing the M.S. degree in electronics and communication engineering. His research interests include data mining, machine learning, and image processing.

HONGXU WEI received the B.S. degree in electronic commerce from the University of Jinan, Jinan, China, in 2015. He is currently pursuing the M.S. degree in computer technology with the Ocean University of China, Qingdao, China. His research interests include machine learning, neural networks, and image processing.

YUCHEN ZHENG received the B.S. and M.S. degrees from the Department of Computer Science and Technology, Ocean University of China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Department of Advanced Information Technology, Kyushu University, Japan. His research interests include machine learning, document analysis, neural networks, and computer vision.

JUNYU DONG (M'09) received the B.S. and M.S. degrees from the Department of Applied Mathematics, Ocean University of China, Qingdao, China, in 1993 and 1999, respectively, and the Ph.D. degree in image processing from the Department of Computer Science, Heriot-Watt University, U.K., in 2003.

He joined the Ocean University of China, in 2004, where he is currently a Professor and the Head of the Department of Computer Science and Technology. His research interests include machine learning, big data, computer vision, and underwater vision.