applied sciences

Article

Training Autoencoders Using Relative Entropy Constraints

Yanjun Li [1,*] and Yongquan Yan [2]

[1] School of Information, Shanxi University of Finance and Economics, Taiyuan 030006, China
[2] School of Statistics, Shanxi University of Finance and Economics, Taiyuan 030006, China
[*] Correspondence: 20181084@sxufe.edu.cn

Abstract: Autoencoders are widely used for dimensionality reduction and feature extraction. The backpropagation algorithm for training the parameters of the autoencoder model suffers from problems such as slow convergence. Therefore, researchers propose forward propagation algorithms. However, the existing forward propagation algorithms do not consider the characteristics of the data itself. This paper proposes an autoencoder forward training algorithm based on relative entropy constraints, called relative entropy autoencoder (REAE). When solving the feature map parameters, REAE imposes different constraints on the average activation value of the hidden layer outputs obtained by the feature map for different data sets. In the experimental section, different forward propagation algorithms are compared by applying the features extracted from the autoencoder to an image classification task. The experimental results on three image classification datasets show that the classification performance of the classification model constructed by REAE is better than that of the classification model constructed by other forward propagation algorithms.

Keywords: autoencoder; relative entropy; image classification

check for updates

Citation: Li, Y.; Yan, Y. Training Autoencoders Using Relative Entropy Constraints. *Appl. Sci.* **2023**, *13*, 287. https://doi.org/10.3390/app13010287

Academic Editors: Muhammad Imran and Neal N. Xiong

Received: 3 November 2022
Revised: 16 December 2022
Accepted: 20 December 2022
Published: 26 December 2022

1. Introduction

The autoencoder is a hot research topic in the field of computer vision [1–10]. It can be used for feature extraction and data dimensionality reduction. In the deep learning framework, autoencoders can be used as the basic modules to form deep and complex networks [11–16]. A deep autoencoder composed of stacking multiple autoencoders can extract more complex and abstract features than a shallow autoencoder. Inspired by the hierarchical information processing mechanism of the human visual system in cognitive neuroscience, Hiton and Salakhutdinov propose an autoencoder-based deep neural network model [17]. Through layer-by-layer greedy pre-training and fine-tuning, this autoencoder-based deep neural network model achieves significantly better classification correctness than traditional methods on image classification datasets. During the pre-training process, the autoencoder is used to initialize the weights of the deep neural network. This highlights the importance of training autoencoders.

The classic autoencoder training algorithm is the backpropagation algorithm [18]. However, this algorithm has the problem of slow convergence. In addition, the backpropagation algorithm requires the user to specify parameters such as step size, momentum factor and maximum number of iterations when optimizing the parameters of the neural network model. The setting of these parameters has an important impact on the training effect of the neural network, but there is no clear theoretical guidance on how to set these parameters. For a specific problem, the user needs to repeatedly debug based on experience to achieve a good result. When the dataset size is large, debugging the parameters based on iterative trial and error can be an extremely time-consuming process.

To solve the problems in the backpropagation algorithm, researchers propose to train the autoencoder in a forward way [19,20]. These methods first map the input data into the hidden layer space, then solve for the model parameters of the decoder, and finally derive the model parameters of the encoder using weight bundling. Different datasets have different characteristics. In the case of image datasets, for example, the type and complexity of images contained in different datasets also vary. However, existing forward propagation algorithms do not fully consider the characteristics of the data itself.

This paper proposes an autoencoder training algorithm based on relative entropy constraints, called relative entropy autoencoder (REAE). It uses relative entropy to constrain the average activation value of the feature mapping on the hidden nodes. When REAE solves for the feature map parameters, different constraints are imposed on the average activation value of the hidden layer outputs obtained by the feature map for different data sets. To verify the validity of REAE, it is applied to the image classification task. The autoencoder cannot be used directly for classification, so it is common practice to combine it with a Softmax classifier to form a new classification model. The experimental results on image classification datasets show that the classification performance of the classification model constructed by REAE is better than that of the classification model constructed by other forward propagation algorithms.

2. Background

The autoencoder is a special kind of neural network that consists of two parts: an encoder and a decoder. The encoder converts the input data into a different representation, and the decoder converts this new representation back to the original format.

The model parameters of the autoencoder can be trained by either a backpropagation algorithm or a forward propagation algorithm. The autoencoder forward propagation algorithm is summarized below.

The training process of the autoencoder forward propagation algorithm is shown in Figure 1.

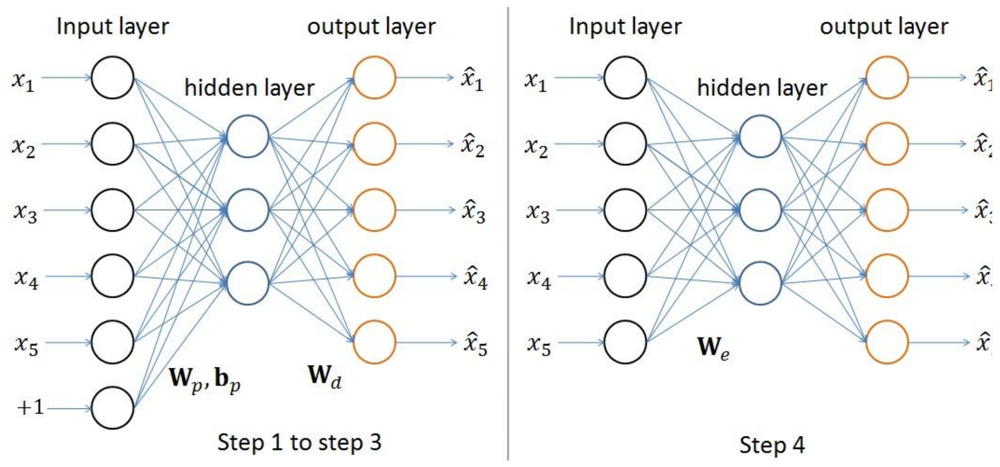


Figure 1. The training process of the autoencoder forward propagation algorithm.

From the perspective of the autoencoder implementation, the feature mapping parameters mentioned in the first step of Algorithm 1 are the parameters of the weights from the input layer to the hidden layer of the autoencoder. The feature mapping parameters are only used to map the input data to the hidden layer space of the autoencoder and do not encode the input data. To avoid confusion with the model parameters of the autoencoder, they are named feature mapping parameters in this paper. In the third step, the decoder model parameters do not include bias terms, so the encoder model parameters obtained through weight bundling also do not include the bias term.

Algorithm 1 Autoencoder forward propagation algorithm.

```
1: Solve for feature mapping parameters $\mathbf{W}_p$ and $\mathbf{b}_p$.
2: Forward propagate the input data, and use the model parameters $\mathbf{W}_p$ and $\mathbf{b}_p$ to map
   the input data to the hidden layer space.
3: Solve for the decoder model parameters $\mathbf{W}_d$.
4: Use weight bundling to calculate encoder model parameters $\mathbf{W}_e$.
```

When designing an autoencoder, it is sometimes necessary to control the parameter scale of the autoencoder according to application requirements. The commonly used method is to use weight bundling in the autoencoder, that is, make the following constraints on the autoencoder's encoder weight parameters $\mathbf{W}_e$ and decoder weight parameters $\mathbf{W}_d$ [19,20]:

$$\mathbf{W}_e^T = \mathbf{W}_d. \tag{1}$$

The difference between different autoencoder forward propagation training algorithms is how to solve for the feature mapping parameters. Wang et al. [19] solve for the feature map parameters by calculating the low-rank approximation of the pseudo-inverse of the input matrix. The advantage of this method is its excellent stability. Since there is no random initialization, the model parameters obtained for a given data set are also the same each time. Kasun et al. [20] solve for feature mapping parameters by randomly initializing weights. Random initialization does not require additional calculations, so the advantage of this method is that the parameters can be solved quickly.

The existing autoencoder forward training algorithms do not fully consider the characteristics of the data itself. The algorithm proposed by Wang et al. [19] uses the low-rank approximation of the pseudo-inverse of the input data as the feature mapping parameter, but its main purpose is to obtain stable feature map parameters. The algorithms proposed by Kasun et al. [20] are data-independent.

To make full use of the characteristics of the data itself, this paper proposes to train autoencoders using relative entropy as a constraint. When REAE solves for the feature mapping parameters of the autoencoder, it restricts the average activation value of the hidden layer outputs according to the characteristics of the dataset.

It should be noted that REAE is different from the classic sparse autoencoder. There are two main differences between them: (1) The constrained objects are different. The sparse autoencoder imposes constraints on the hidden layer outputs generated by the encoder. REAE imposes constraints on the hidden layer outputs generated by the feature mapping; (2) different constraint requirements. The sparse autoencoder imposes a sparse constraint on the average activation value, and requires that the average activation value of the hidden layer outputs generated by the encoder is very small. The constraints of REAE on the average activation value of the hidden layer outputs depend on the dataset to which REAE is applied. Generally, the average activation value of the hidden layer outputs of the sparse autoencoder is set to a number close to 0, e.g., 0.05. When REAE does feature mapping with different data sets, the constraint value for the average activation of the hidden layer output may be a larger number or a smaller number. Taking the MNIST dataset and the CIFAR-10 dataset as examples, the best average activation value for REAE is 0.2 on the MNIST dataset and 0.8 on the CIFAR-10 dataset.

3. Relative Entropy Autoencoder

Like other autoencoder forward propagation training algorithms, the model parameters of REAE are composed of three parts: feature mapping parameters, decoder model parameters and encoder model parameters. Since the encoder model parameters can be obtained by bundling the decoder model parameters, this section is divided into two parts: solving for feature mapping parameters and solving for decoder parameters.

3.1. Solving for Feature Mapping Parameters

Suppose the training set is $\mathcal{D} = \{(\mathbf{x}_m, \mathbf{o}_m) | \mathbf{x}_m \in \mathbf{R}^d, \mathbf{o}_m \in \mathbf{R}^k\}_{m=1}^M$, where $\mathbf{x}_m$ represents the input feature of the training sample, $\mathbf{o}_m$ represents the label of the training sample, and M represents the number of training samples. Let $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_M]$ denote the training sample feature matrix and $\mathbf{O} = [\mathbf{o}_1, \cdots, \mathbf{o}_M]$ denote the training sample label matrix. The input features and labels of the training samples are all represented by column vectors, and each column corresponds to one datum. The autoencoder takes the input data as the reconstruction target, that is, $\mathbf{O} = \mathbf{X}$. In this case, the training set can be expressed as $\mathcal{D} = \{\mathbf{x}_m | \mathbf{x}_m \in \mathbf{R}^d\}_{m=1}^M$. Let J denote the number of hidden nodes of the autoencoder. Let $\mathbf{h}_m = [h_{m1}, \cdots, h_{mJ}]^T$, h_{mj} denote the activation of the input data $\mathbf{x}_m$ on the hidden layer node j through feature mapping:

$$h_{mj} = \sigma(\mathbf{w}_{pj}\mathbf{x}_m + b_{pj}), \tag{2}$$

where $\mathbf{w}_{pj}$ is a row vector consisting of the jth row of $\mathbf{W}_p$, and b_{pj} is the bias consisting of the j item of vector $\mathbf{b}_p$. The average activation value of the training set $\mathcal{D}$ on the hidden node j through feature mapping is:

$$\hat{\rho}_j = \frac{1}{M} \sum_{m=1}^{M} h_{mj}. \tag{3}$$

REAE imposes the following constraint on the average activation of feature mapping on the hidden node j:

$$\hat{\rho}_j = \rho, \tag{4}$$

where ρ is a dataset-dependent average activation parameter, which represents the average activation value required by the constraint. In addition, this paper assumes that the average activation constraints take the same value on all hidden nodes, that is, $\hat{\rho}_j = \rho, j = 1, \cdots, J$.

REAE uses relative entropy to constrain the average activation of hidden nodes. According to the definition of relative entropy, the loss function of feature mapping can be defined as:

$$\mathcal{L}(\mathcal{D}; \mathbf{W}_p, \mathbf{b}_p) = \sum_{j=1}^{J} \left(\rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \right). \tag{5}$$

The loss function (5) has no analytical solution, so the feature mapping parameters $\mathbf{W}_p$ and $\mathbf{b}_p$ can be solved for by gradient descent.

3.2. Solving for Decoder Parameters

An autoencoder is an unsupervised learning algorithm. If the data itself is regarded as a kind of label, that is, $\mathbf{O} = \mathbf{X}$, then the autoencoder can be trained by the mode of supervised learning. Assuming that the loss function of the autoencoder adopts the squared error loss function, and the activation function of the output layer adopts a linear activation function, the optimization objective of the autoencoder can be defined as:

$$\min_{\mathbf{W}_d} \|\mathbf{W}_d\mathbf{H} - \mathbf{X}\|^2 + \lambda \|\mathbf{W}_d\|^2, \tag{6}$$

where $\lambda (\lambda > 0)$ is a weight parameter, which is used to control the importance of the first item and the second item in the optimization objective.

The optimization problem (6) has an analytical solution. When the number of columns of the hidden layer output matrix $\mathbf{H}$ is greater than the number of rows ($M > J$), the calculation formula of $\mathbf{W}_d$ is:

$$\mathbf{W}_d = \mathbf{X}\mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}, \tag{7}$$

where $\mathbf{I}$ is an identity matrix of size $J \times J$. When the number of rows of the hidden layer output matrix $\mathbf{H}$ is greater than or equal to the number of columns ($J \geq M$), the calculation formula of $\mathbf{W}_d$ is:

$$\mathbf{W}_d = \mathbf{X}(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T, \tag{8}$$

where $\mathbf{I}$ is an identity matrix of size $M \times M$.

When the forward algorithm is used to train the autoencoder, the encoder model parameters are obtained by bundling the decoder model parameters. Therefore, the calculation formula of the encoder model parameters is:

$$\mathbf{W}_e = \mathbf{W}_d^T. \tag{9}$$

In summary, taking the number of training samples greater than the number of hidden nodes ($M > J$) as an example, the autoencoder forward training algorithm based on relative entropy constraints, REAE (Algorithm 2), can be described as:

Algorithm 2 Forward training algorithm for autoencoders based on relative entropy constraints.

```
1: Solve for the feature mapping parameters $\mathbf{W}_p$ and $\mathbf{b}_p$ by optimizing the loss function (5)
   using gradient descent.
2: Forward propagate data, and use the model parameters obtained in step (1) to calculate
   the hidden layer output matrix $\mathbf{H}$.
3: Calculate the decoder model parameters $\mathbf{W}_d$ according to the Formula (7).
4: Use the Formula (9) to calculate the encoder model parameters $\mathbf{W}_e$.
```

When the number of hidden nodes is greater than or equal to the number of training samples ($J \geq M$), the third step of Algorithm 2 can use the Formula (8) to calculate the decoder model parameters $\mathbf{W}_d$.

4. Experiments

This section verifies the effectiveness of the REAE algorithm on the image classification data sets MNIST [21], CIFAR-10 [22] and SVHN [23]. Direct performance comparisons between autoencoders trained by different forward propagation algorithms are not possible. Therefore, autoencoders are applied to image classification tasks, and classification accuracy is used as a basis for evaluating forward propagation algorithms. Autoencoders do not have a classification function and need to be combined with a Softmax classifier to form a classification model. A common method of building a classification model is to remove the decoder from the autoencoder and use the output of the encoder as the input feature to the Softmax classifier.

For ease of presentation, the classification model constructed by the autoencoder and Softmax classifier is referred to as an autoencoder network. In this paper, the classification model based on REAE is named relative entropy autoencoder network (REAN), the classification model based on the algorithm proposed in Ref. [19] is named pseudo inverse autoencoder network (PIAN), and the classification models based on the two algorithms proposed in Ref. [20] are named random autoencoder network (RAN) and random orthogonal autoencoder network (ROAN), respectively.

The experiments consist of two parts: analysis of factors affecting classification performance and comparison of algorithm performance.

4.1. Experimental Setup

The MNIST dataset is a large digital image recognition database consisting of handwritten digits [21]. It contains a total of 70,000 handwritten digital images consisting of numbers 0 to 9. Each image in the MNIST dataset is standardized and aligned to an image patch of a fixed size of 784 pixels. The CIFAR-10 dataset is a color image dataset, which contains 60,000 color images of size 32 × 32 [22]. The image content includes 10 categories, consisting of planes, cars, birds, cats, deer, dogs, frogs, horses, boats and trucks. The SVNH dataset is a real image dataset for developing machine learning and object recognition algorithm [23]. Its images are derived from the house numbers in Google Street View images.

The experimental software environment is as follows: 64-bit version of Windows 7, matlab 2014b. The experimental hardware platform is as follows: Intel(R) Core(TM) i3-550 CPU, 8G memory.

4.2. Analysis of Factors Affecting Classification Performance

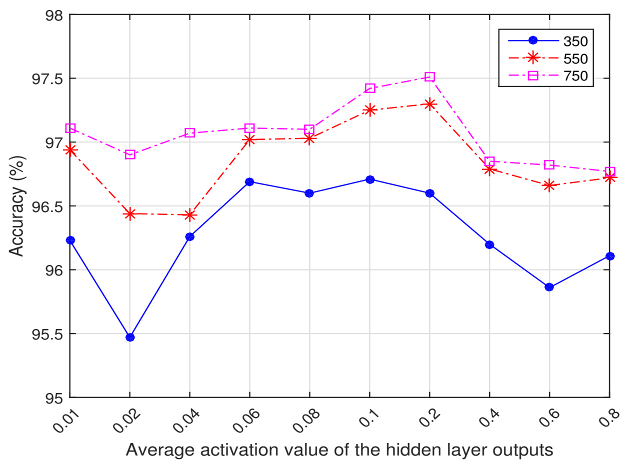
4.2.1. Average Activation Value of the Hidden Layer Outputs

The purpose of training an autoencoder is to extract effective features for application services. This subsection takes the classification application as an example, and analyzes the influence of the average hidden layer activation value ρ on the classification performance of the REAN autoencoder network under different settings of the number of hidden nodes. The number of hidden nodes is set to 350, 550, and 750 on the MNIST dataset, and 450, 700, and 950 on the CIFAR-10 dataset, respectively. The average hidden layer activation value ρ is taken on the grid $\{0.01, 0.02, 0.04, 0.06, 0.08, 0.1, 0.2, 0.4, 0.6, 0.8\}$. In the REAN autoencoder network, the search range of the regularization parameter λ of the autoencoder and the regularization parameter τ of the Softmax classifier is the grid $\{10^{-5}, 10^{-4}, \cdots, 10^{2}\}$. The specific parameter values are determined by cross-validation.

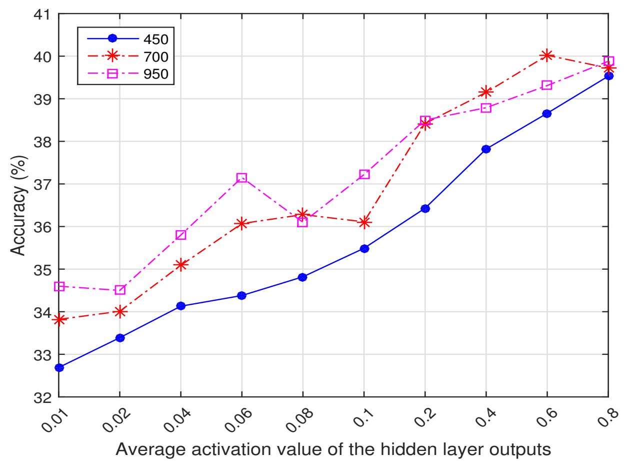
The experimental results are shown in Figure 2. The blue polyline, red polyline, and magenta polyline in Figure 2a correspond to the experimental results when the number of hidden nodes on the MNIST data set is 350, 550, and 750, respectively. The blue polyline, red polyline, and magenta polyline in Figure 2b correspond to the experimental results when the number of hidden nodes on the CIFAR-10 data set is 450, 700, and 950, respectively.

Figure 2 shows that given the number of hidden nodes, the average hidden layer activation value ρ has a direct impact on the classification performance of the REAN autoencoder network. It can be seen from Figure 2a that, on the MNIST data set, when the average hidden layer activation value ρ is in the interval $[0.1, 0.2]$, the classification performance of the REAN autoencoder network is the best. When the average hidden layer activation value ρ is greater than 0.2, the classification performance of the REAN autoencoder network generally declines. When the number of hidden nodes is 350, the classification performance of the REAN autoencoder network is the best at $\rho = 0.1$, and the classification accuracy rate is 96.7%. When the number of hidden layer nodes is 550 and 750, the classification performance of REAN autoencoder network is the best at $\rho = 0.2$, and the classification accuracies are 97.3% and 97.5%, respectively. Observing Figure 2b, it can be found that the classification performance of the REAN autoencoder network shows an obvious trend of improvement with the increase in the average hidden layer activation value ρ. The best classification performance of the REAN autoencoder network is obtained when the average hidden layer activation value ρ takes a larger value (0.6 or 0.8). The interpretation of this paper is that the data complexity of CIFAR-10 is much greater than that of MNIST. When the number of hidden nodes is 700, the classification performance of the REAN autoencoder network is the best at $\rho = 0.6$, and the classification accuracy rate is 40.0%. When the number of hidden layer nodes is 450 and 950, the classification performance of the REAN autoencoder network is the best at $\rho = 0.8$, and the classification accuracies are 39.5% and 39.8%, respectively.

It can be seen from the value of the optimal average hidden layer activation value ρ on the MNIST dataset and the CIFAR-10 dataset that the selection of ρ is closely related to the dataset where the REAE algorithm is applied. This is the desired result of this paper, i.e., to represent the characteristics of different datasets by means of a hidden layer average activation value constraint.



(**a**) MNIST dataset



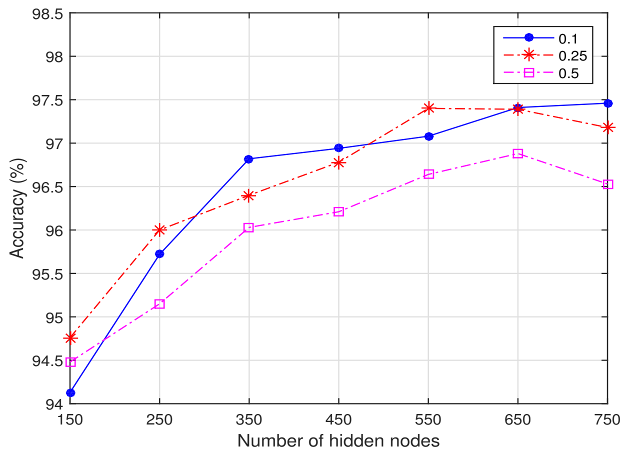
(**b**) CIFAR-10 dataset

Figure 2. The relationship between the average activation value of the hidden layer outputs and classification performance in the REAN autoencoder network.

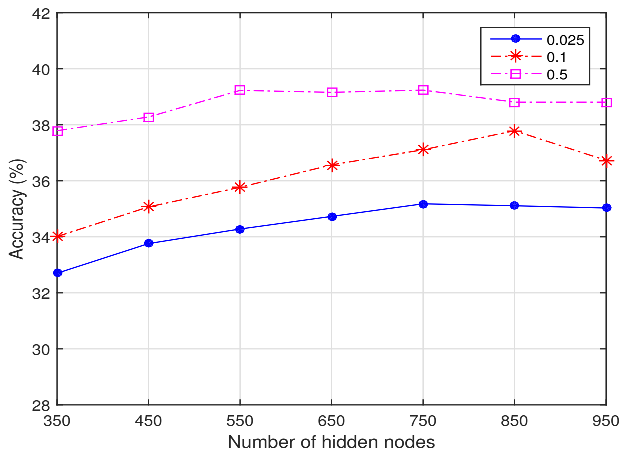
4.2.2. Number of Hidden Nodes

The learning ability of a neural network is closely related to the number of hidden nodes. The greater the number of hidden nodes, the stronger the learning ability of the network. The sample dimension of the MNIST dataset is 784, so the number of hidden nodes is taken on a grid $\{150, 250, 350, 450, 550, 650, 750\}$. The sample dimension of the CIFAR-10 dataset is 1024 (the color images in the CIFAR-10 dataset are converted to grayscale images), so the number of hidden nodes is taken on a grid $\{350, 450, 550, 650, 750, 850, 950\}$. In the REAN autoencoder network, the search range for both the regularization parameter λ of the autoencoder and the regularization τ of the Softmax classifier is grid $\{10^{-5}, 10^{-4}, \cdots, 10^{2}\}$, the specific parameter values are determined by cross-validation. The experimental results are shown in Figure 3.

It can be seen from Figure 3 that when the number of hidden nodes is small, the classification performance of the REAN autoencoder network increases with the increase in the number of hidden nodes. This conclusion holds for different average hidden layer activation value constraints. This is because as the number of hidden nodes increases, the autoencoder trained by REAE can extract more effective features. Thus, the classification performance of the REAN autoencoder network is improved. After the number of hidden nodes increases to a threshold, if the number of hidden nodes continues to increase, the improvement of the classification performance of the REAN autoencoder network will slow down or even decrease.



(**a**) MNIST dataset



(**b**) CIFAR-10 dataset

Figure 3. The relationship between the number of hidden nodes and classification performance in the REAN autoencoder network.

Figure 3a shows that, on the MNIST dataset, when the number of hidden nodes exceeds 650 and the average activation value of the hidden layer is constrained to $\rho = 0.1$, the classification performance improvement of the REAN autoencoder network becomes very low. When the average activation value of the hidden layer is constrained by $\rho = 0.25$ and $\rho = 0.5$, the classification performance of the REAN autoencoder network is not improved but decreased. The best classification effect on the MNIST dataset is obtained when the average activation value of the hidden layer is constrained to $\rho = 0.1$ and the number of hidden nodes is set to 750. Its best classification accuracy is 97.4%. Figure 3b shows that, on the CIFAR-10 data set, when the average hidden layer activation value constraint ρ is equal to 0.025, 0.1 and 0.5, respectively, the best classification effect of the REAN autoencoder network is obtained when the number of hidden nodes is 750, 850 and 550, respectively. The best classification effect on the CIFAR-10 dataset is obtained when the average activation value of the hidden layer is constrained to $\rho = 0.5$ and the number of hidden nodes is set to 550. Its best classification accuracy is 39.2%.

4.3. Comparison of Algorithm Performance

This subsection compares the classification performance of the REAN autoencoder network with the autoencoder networks constructed by other forward algorithms through experiments. Both network structure and loss function will affect the classification performance of an autoencoder network. For fairness, all autoencoder networks adopt the same network structure and loss function. On the MNIST, CIFAR-10 and SVHN data sets, the number of hidden nodes is set to 750. The average hidden layer activation value ρ is selected on a grid $\{0.1, 0.2, \cdots, 0.9\}$. The regularization parameter λ of the autoencoder and the regularization τ of the Softmax classifier are searched on a grid $\{10^{-5}, 10^{-4}, \cdots, 10^{2}\}$. All candidate parameters (ρ, λ, τ) are determined by cross-validation. On the three datasets, each method was repeated 10 times. The results with the highest average test accuracy on each dataset are shown in bold. The experimental results are presented in Table 1.

Table 1. Comparative experiment results on MNIST, CIFAR-10 and SVHN datasets.

Name	MNIST			CIFAR-10			SVHN		
	Accuracy	Parameter		Accuracy	Parameter		Accuracy	Parameter	
		λ	τ		λ	τ		λ	τ
REAN	**97.3**	10^{-2}	10^{-1}	**40.0**	10^{-2}	10^{-2}	**62.4**	10^{-3}	10^{-3}
PIAN	97.0	10^{-5}	10^{0}	37.0	10^{-4}	10^{-1}	56.5	10^{-5}	10^{-1}
RAN	96.9	10^{-2}	10^{-1}	39.3	10^{-2}	10^{-1}	60.2	10^{-3}	10^{-5}
ROAN	96.8	10^{-2}	10^{-1}	39.5	10^{-2}	10^{0}	60.7	10^{-3}	10^{-2}

It can be seen from the test accuracy given in Table 1 that the classification performance of the REAN autoencoder network is better than that of other autoencoder networks. The average classification accuracy of REAN autoencoder network on MNIST data set is 97.3%, which is 0.3%, 0.4%, and 0.5% higher than that of the PIAN autoencoder network, the RAN autoencoder network and the ROAN autoencoder network, respectively. The average classification accuracy of the REAN autoencoder network on the CIFAR-10 data set is 40.0%, which is 3.0%, 0.7% and 0.5% higher than that of the PIAN autoencoder network, the RAN autoencoder network and the ROAN autoencoder network, respectively. The average classification accuracy of the REAN autoencoder network on the SVHN data set is 62.4%, which is 5.9%, 2.2% and 1.7% higher than that of the PIAN autoencoder network, the RAN autoencoder network and the ROAN autoencoder network, respectively.

To show the significant differences among the experimental results of the various methods, the results of paired *t*-test for each dataset are also reported. The significance level α was set to 0.05. H = 1, which indicates that there is a statistically significant difference, while H = 0 indicates that there is no significant difference. The smaller the P-value (probability) is, the more obvious the difference is between the different methods. Experimental results are shown in Table 2. As can be seen from Table 2, there is a statistically significant difference between the REAE autoencoder network and the other autoencoder networks in terms of classification accuracy.

Table 2. Paired *t*-tests on MNIST, CIFAR-10 and SVHN datasets.

Name	REAN vs. PIAN		REAN vs. RAN		REAN vs. ROAN	
	H	P	H	P	H	P
MNIST	1	1.34e-05	1	4.12e-07	1	5.03e-06
CIFAR-10	1	3.34e-09	1	2.13e-04	1	2.69e-02
SVHN	1	3.54e-10	1	5.55e-06	1	4.42e-05

5. Discussion and Conclusions

This paper presents a new autoencoder forward training algorithm, which is named REAE. Existing autoencoder forward propagation algorithms do not take into account the properties of the data itself when solving for the feature mapping parameters. According to the data set, REAE uses relative entropy to constrain the average activation value of hidden layer outputs. The REAE algorithm consists of four steps; firstly, solving for the feature mapping parameters, then mapping the input data into the hidden space, then solving for the decoder model parameters, and finally, obtaining the encoder model parameters by weight binding.

There is no direct performance comparison between autoencoders trained by different forward algorithms. Autoencoders are mainly used for data feature extraction, so they are applied to image classification tasks, and the classification accuracy is used as a criterion to evaluate the performance of the algorithm. The autoencoder itself does not have a classification function. In this paper, the decoder of the autoencoder is removed and the output of the encoder is used as the input features of the Softmax classifier to build a classification model.

The experiments consist of two parts: analysis of factors affecting classification performance and comparison of algorithm performance. The average activation value of the hidden layer outputs and the number of hidden nodes affect the classification performance. The experimental results on the average activation value of the hidden layer outputs show that different datasets correspond to different optimal hidden layer activation values. This is exactly what is expected in this paper, i.e., to characterize the different datasets by means of a hidden layer average activation value constraint. The experimental results on hidden nodes show that the classification performance of the REAN autoencoder network improves as the number of hidden nodes increases. This is due to the fact that, as the number of hidden nodes increases, REAE can extract a richer set of features. It should be noted that there is no clear functional relationship between the average activation value of the hidden layer and the number of hidden nodes, and cross-validation is required to determine the values of these two hyperparameters in specific applications. This is due to the fact that as the number of hidden nodes increases, REAE can extract a richer set of features. It should be noted that there is no explicit functional relationship between the average activation value of the hidden layer outputs and the number of hidden nodes, and cross-validation is required to determine the values of the two hyperparameters in a specific application. The experimental results of the algorithm performance comparison show that given the network structure and loss function, the classification model constructed by REAE is superior to the classification models constructed by other forward algorithms in classification performance.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.Y.; validation, Y.Y.; formal analysis, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing,Y.L. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Natural Science Youth Foundation of Shanxi Province (No. 201901D211415).

Data Availability Statement: Publicly available datasets were analyzed in this study. The data used in the article can be found here: MNIST: http://yann.lecun.com/exdb/mnist/ (accessed on 1 November 2022). CIFAR-10: http://www.cs.toronto.edu/~kriz/cifar.html (accessed on 1 November 2022). SVHN: http://ufldl.stanford.edu/housenumbers (accessed on 1 November 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhou, Y. Rethinking Reconstruction Autoencoder-Based Out-of-Distribution Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 7379–7387.
2. Liu, X.; Ma, Z.; Ma, J.; Zhang, J.; Schaefer, G.; Fang, H. Image Disentanglement Autoencoder for Steganography Without Embedding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 2303–2312.
3. Kim, M. Gaussian Process Modeling of Approximate Inference Errors for Variational Autoencoders. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 244–253.
4. Yu, M.; Quan, T.; Peng, Q.; Yu, X.; Liu, L. A model-based collaborate filtering algorithm based on stacked autoencoder. *Neural Comput. Appl.* **2022**, *34*, 2503–2511. [CrossRef]

5. Yang, J.; Ahn, P.; Kim, D.; Lee, H.; Kim, J. Progressive Seed Generation Auto-Encoder for Unsupervised Point Cloud Learning. In Proceedings of the IEEE International Conference on Computer Vision, Online, 11–17 October 2021; pp. 6413–6422.
6. Wang, C.; Lucey, S. PAUL: Procrustean Autoencoder for Unsupervised Lifting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Online, 19–25 June 2021; pp. 434–443.
7. Parmar, G.; Li, D.; Lee, K.; Tu, Z. Dual Contradistinctive Generative Autoencoder. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Online, 19–25 June 2021; pp. 823–832.
8. Preechakul, K.; Chatthee, N.; Wizadwongsa, S.; Suwajanakorn, S. Diffusion Autoencoders: Toward a Meaningful and Decodable Representation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 10619–10629.
9. Meng, Q.; Catchpoole, D.; Skillicom, D.; Kennedy, P.J. Relational Autoencoder for Feature Extraction. In Proceedings of the 2017 International Joint Conference on Neural Networks, Anchorage, AK, USA, 14–19 May 2017; pp. 364–371.
10. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and Composing Robust Features with Denoising Autoencoders. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 1096–1103.
11. Wu, B.; Nair, S.; Martin-Martin, R.; Fei-Fei, L.; Finn, C. Greedy Hierarchical Variational Autoencoders for Large-Scale Video Prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Online, 19–25 June 2021; pp. 2318–2328.
12. Ashfahani, A.; Pratama, M.; Lughofer, E.; Ong, Y.S. DEVDAN: Deep evolving denoising autoencoder. *Neurocomputing* **2020**, *390*, 297–314. [CrossRef]
13. Zhou, C.; Paffenroth, R.C. Anomaly Detection with Robust Deep Autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 665–674.
14. Qiao, C.; Hu, X.Y.; Xiao, L.; Calhoun, V.D.; Wang, Y.P. A deep autoencoder with sparse and graph Laplacian regularization for characterizing dynamic functional connectivity during brain development. *Neurocomputing* **2021**, *456*, 97–108. [CrossRef]
15. Jian, L.; Rayhana, R.; Ma, L.; Wu, S.; Liu, Z.; Jiang, H. Infrared and visible image fusion based on deep decomposition network and saliency analysis. *IEEE Trans. Multimed.* **2022**, *24*, 3314–3326. [CrossRef]
16. Shi, C.; Pun, C.M. Multiscale superpixel-based hyperspectral image classification using recurrent neural networks with stacked autoencoders. *IEEE Trans. Multimed.* **2020**, *22*, 487–501. [CrossRef]
17. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]
18. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
19. Wang, K.; Guo, P.; Xin, X. Autoencoder, Low Rank Approximation and Pseudoinverse Learning Algorithm. In Proceedings of IEEE 2017 International Conference on Systems, Man, and Cybernetics, Banff, AB, Canada, 1–4 October 2017; pp. 948–953.
20. Kasun, L.L.C.; Zhou, H.; Huang, G.B.; Vong, C.M. Representational learning with elms for big data. *IEEE Intell. Syst.* **2013**, *28*, 31–34.
21. The Mnist Database of Handwritteen Digits. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 1 November 2022).
22. The Cifar-10 Database. Available online: http://www.cs.toronto.edu/~kriz/cifar.html (accessed on 1 November 2022).
23. The Street View House Numbers Dataset. Available online: http://ufldl.stanford.edu/housenumbers (accessed on 1 November 2022).