Unsupervised feature learning for data classification

Shixuan An\textsuperscript{1, a, †}, Ruicheng Lu\textsuperscript{2, b, †} and Tianyi Zhang\textsuperscript{3, c, †}

\textsuperscript{1}School of Mathematics and Statistics, Central China Normal University, Luonan Street, Wuhan, Hubei 430079, China;
\textsuperscript{2}School of Mathematics and Statistics, Dalhousie University, Halifax, NS, B3H 4R2, Canada;
\textsuperscript{3}School of Economics and Management, Harbin Institute of Technology, 92 Xidazhi Street, Harbin, Heilongjiang 150001, China.

Corresponding author e-mail: \textsuperscript{a}rc418815@dal.ca, \textsuperscript{b}anshix@mails.ccnu.edu.cn, \textsuperscript{c}1181000808@stu.hit.edu.cn

\textsuperscript{†}These authors contributed equally.

Abstract. Data classification is a critical task for data analysis. However, recent methods aim to model this task as a classification or regression task, which needs ground truth for discriminative representation learning. This paper aims to learn a more efficient and effective feature for data analysis in an unsupervised learning manner. Our method consists of two main components: a customized autoencoder network (C-AENet) and a customized squeeze-and-excitation network (C-SENet). The C-AENet aims to reconstruct the feature using the fewer dimension, and C-SENet is to improve the representation by providing channel attention. Experiment on the Avila dataset shows that both modules are effective in data classification with fewer feature dimension.

1. Introduction

Deep learning is a method in machine learning, which is an essential branch of artificial intelligence. The word “deep” comes from a large number of hidden layers in the deep learning model. It is a complex algorithm for learning multi-layer feedforward neural networks. The essence of deep learning is to train the neural network model stepwise to hold the optimal parameters for objective tasks \cite{1}. Therefore, the feature information extracted by each layer of the model represents a ubiquitous data characteristic on which the prediction targets depend highly.

In recent years, there is a surge in the research field of deep learning. Consequently, deep learning has now become a hotspot in the field of machine learning. But among all the interesting subfields of deep learning, the focus points of this paper are listed below:

1. Integrating feature selection strategies into Neural Network (NN) models
2. Integrating feature reduction or compression using unsupervised learning methods into NN models

Feature selection is to select the representative feature elements in the feature set and keep the original one. It can also be regarded as a dimension reduction operation, that is, compressing high dimensional data to low dimensional data to reduce feature space. The neural network is also widely used in feature selection due to nonlinear function mapping, large-scale parallel data processing, and good self-learning and self-adaptive characteristics. In this paper, the core task is to integrate feature
selection strategies into NN models to select the features that contribute the most to the information of labels and visualize them to intuitively know the most important features that affect the results.

Before using neural networks to perform multi-classification tasks, here are some data preprocessing methods. Because there are many irrelevant or redundant features in high-dimensional data set, which will affect the learning performance of NN models, dimensional feature reduction is a necessary unsupervised learning method. There are some common dimension reduction algorithms in unsupervised learning, such as the linear methods, including PCA and SVD, and nonlinear methods such as Autoencoder, LLE. Lots of research has attempted to reduce the dimensionality of features through effective unsupervised learning methods in front of NN models. For example, they use autoencoder and convolutional neural network technology to complete image dimensionality reduction and classification tasks. They first denoise the image components, then use autoencoder to segment and reduce the dimension, and finally use the CNN model to realize the classification problem. In another research, the first use PCA to reduce the dimension of the data. They then generate the mathematical framework of convolutional autoencoder (CAE) [3] from PCA and make a more in-depth analysis of the behavior of CAE and CNN's PCA initialization network.

There is quite a lot of research that concentrates on feature selection strategies. For example, they propose a simple module named Bottleneck Attention Module (BAM) [4], which can be placed at each bottleneck of models to do the feature enhancement. It can achieve great results proved by the experiments. In addition, some research proposes a simple forward convolution neural network attention module named Convolutional Block Attention Module (CBAM) [5]. It can conduct end-to-end training with basic CNN. They tested and verified CBAM through many experiments and found that its different models have a consistent improvement in classification and detection performance. And some other research designs a dynamic selection mechanism for CNN and a building block called selective kernel unit (SK) [6]. Multiple SK accumulates into a deep network of SKNet, which can capture different multi-dimensional targets through information search and fusion of different branches using SoftMax attention and verify the ability of neurons to adaptively adjust the size of the receptive field according to input.

In this paper, the goal is based on neural networks to perform multi-classification tasks on dataset. By experimenting with different methods to preprocess the data or modifying the models, their resulting effects could be easily compared. On the one hand, the key investigation point is to reduce the dimensionality of features and compress it through effective unsupervised learning methods for data preprocessing methods. Through this way, the effect of eliminating redundant information contained in the original dataset is expected. On the other hand, it is also helpful to search the most discriminant feature to our classification task or find the discriminant feature ranking. Also, the degree of perfection to the compression of original data could be observed by comparing the correlation of discriminant power between different features compressed data with raw data.

2. Related works and models
In this section, two kinds of standard NN modules and their corresponding models will be briefly discussed as well as an overview of their basic structures.

2.1. Autoencoder (AE)
AE was first introduced in 1986 [7] as a special kind of NN designed to reconstruct its input through training. In this way, the whole learning process inside AE could be considered an unsupervised learning method since there is no target label for each data involved during the training. The general key objective hidden behind the seemingly implausible reconstruction process is that AE intends to acquire an “informative” hidden representation of the raw data. Such hidden representation could be later used to achieve various other tasks according to the user’s final goal. Accordingly, many regularizations during the construction of the hidden representation could be imposed to accommodate the needs of different tasks.
2.1.1. AE module Structure

As a non-linear unsupervised method, AE could be versatile and compatible for various jobs like described above. In this subsection, a closer look at the basic structure of AE and introduce several concepts will be given to prepare the customized model in later sections.

The two critical components of AE are called encoder \((f)\) and decoder \((g)\), as shown in Figure 1. The whole training process is to search the optimal parameters contained in \(f\) and \(g\). Here, the optimality is in the sense of achieving minimal reconstruction loss. In a more mathematical term, \(f\) and \(g\) could be properly defined as two functions parametrized by multiple parameters, i.e.,

\[
\begin{align*}
    f : \mathbb{R}^n &\rightarrow \mathbb{R}^m \\
    g : \mathbb{R}^m &\rightarrow \mathbb{R}^n
\end{align*}
\]

where assuming that \(\text{dim}(X) = \text{dim}(X') = n\) and \(\text{dim}(h) = m\).

With the help of equation (1) & (2), It is an appropriate time to define the training loss (reconstruction loss) as mentioned above to be:

\[
\mathcal{L}(\text{recon}) = \mathbb{E}_{X}[\mathcal{L}_R(X, g \circ f(X))]
\]

Where \(\mathbb{E}_{X}\) is the expectation over the distribution of \(X\), and a common choice for the reconstruction loss function \(\mathcal{L}_R\) in AE is the \(L_2\) norm or the \(L_1\) norm or the Squared Error function (SE). The detailed equations for these two loss functions are shown below in equation (4), (5) and (6):

\[
\begin{align*}
    \mathcal{L}_2(x, y) &= \sqrt{\|x - y\|^2} \\
    \mathcal{L}_1(x, y) &= \|x - y\| \\
    \text{SE}(x, y) &= \|x - y\|^2
\end{align*}
\]

Where \(x\) and \(y\) are the data points with same dimension. Finally, the optimal \(f\) and \(g\) could be acquired by stepwise solving the optimization problem:

\[
\begin{align*}
    f^{opt}, g^{opt} &= \arg\min_{f,g} \mathcal{L}(\text{recon}) \\
    &= \arg\min_{f,g} \mathbb{E}_{X}[\mathcal{L}_R(X, g \circ f(X))] \\
    &= \arg\min_{f,g} \mathbb{E}_{X}[\text{SE}(X, g \circ f(X))] \\
    &= \arg\min_{f,g} \mathbb{E}_{X}[\|X - g \circ f(X)\|^2] \\
    &= \arg\min_{f,g} \text{MSE}[X, g \circ f(X)]
\end{align*}
\]

Thus, It is not hard to see that the final reconstruction loss function of AE is simply the Mean Squared Error (MSE) of input data and reconstructed data when using SE as the loss function from the above derivation.
After elucidating the general structure of AE, a reasonable concern regarding the latent representation might be proposed. That is, it is possible to just get the identity function for both $f$ and $g$ after training, since it always reduces the reconstruction loss to exactly 0 in such setting, i.e.:

$$f = g = I_n$$

(8)

Where $I_n$ represents the n-dimensional identity function. If this is the case, the final latent representation $h$ (as shown in Figure 1) will simply be the input data itself. Though, in this case, the $h$ could also be considered as “informative”, but such result is in some sense useless. Hence, to overcome this drawback, some regularization about the hidden layer must be done for most AE use cases. Based on the different regularization factors, it is easy to divide AE into several subcategories such as Sparse Autoencoder (SAE) [8], Denoising Autoencoder (DAE) [8], Contractive Autoencoder (CAE) [8], etc. and apply them to targeted tasks. However, only one of the simplest regularization methods is needed to be considered for later experiments, which makes the dimension of the hidden representation $h$ smaller than the input. i.e., make $dim(h) < dim(X)$. In other words, a bottleneck is imposed on AE. Notice that this particular regularization method has many direct and obvious applications such as feature extraction, data compression, etc.

![AE example for data compression](image)

**Figure 2: AE example for data compression [8]**

In Figure 2, a visual example of applying AE to one of the MNIST data for compression purposes was displayed. Here, the hidden layer contains the compressed representation of input pixels. As shown above, the reconstructed data pixels become obscure compare to the original input pixels. It is inevitable to lose some information during the compression, but the key outline of the data has been kept. In many real-life scenarios, the AE often uses a highly symmetrical architecture between the encoder ($f$) and the decoder ($g$) to achieve a better performance since $f$ and $g$ are highly symmetrical in a mathematical sense. The AE model in this paper will refer to the AE model with a compressional hidden layer.

2.1.2. **AENet**

So far, it is merely an investigation of the structure and usage of a single AE model. Now, the topic will concentrate on concatenating AE as a feature compressor with a Fully Connected Neural Network (FCNN) model to achieve a classification task. In this paper, such a model will be denoted as the AENet model.
As described above, the basic structure of two different kinds of AENet could be visualized in Figure 3. Before moving on, one worth noting difference to the simple AE model is that in AENet, the task labels are necessary. The whole network is built to fulfill the final task contained in the NN block (Figure 3) while maintaining the compression task of AE in some level \( \lambda \) which explains how much reconstruction loss the task could tolerate. In other words, AENet uses the AE model as a regularization technique for the final task in the NN block. More detailed explanations could be found in the following subsection.

Much research [9,10] as attempted to apply AE to a classification task. One of the most related networks was illustrated in Figure 3 (path 2), where two different blocks are connected to the encoder module in AE. They [9,10] tried to train the classification network with the compressed hidden representation directly. Thus the new classification loss would be the original loss with the reconstruction regularizer. i.e.:

\[
\mathcal{L}^{\text{new}} = \mathcal{L}(\text{classification}) + \lambda \cdot \mathcal{L}(\text{recon})
\]

where \( \ast \) denotes the usual multiplication operator.

You might notice a great similarity by looking at path 1 and path 2 in Figure 3. They both use the AE module as a regularizer for another supervised learning task. Nevertheless, one essential difference between them is: if they both perform a classification task as a final goal, path 1 keeps the dimension of data at the classification stage. In contrast, path 2 reduces the dimension of input data when entering the NN block since the latter feeds the compressed data directly into the classification network. For the purpose of later experiment, it is important to keep the same dimension of the data and use the AE as a light-weighted compressor module that can be applied multiple times without interfering with the pre-defined task model. For this particular reason, path 1 is selected, and from now on, such an AE compressor module will be called the AE module in later sections. Meanwhile, the AENet will specifically refer to AENet-path 1 as drawn in Figure 3 from this moment.

2.2. Squeeze-and-Excitation Network (SENet)

The SENet was an award-winning network in ILSVRC 2017 classification contest. It was later officially introduced by their research team [11]. As its name suggests, the SENet is a special kind of NN formed by stacking together multiple SE modules similar to the AE module, a powerful light-weighted module used in NN. But unlike the AE module, it manages to enhance the representational power of NN by applying a channel-wise attention mechanism instead of simply compressing the data. More details about the SE module, SENet, will be discussed in this subsection.
2.2.1. Squeeze-and-Excitation (SE) module structure

The original SE block published in their paper [11] was based on Convolutional Neural Network (CNN), which is a specially designed NN for fusing both spatial and channel information within a certain perceptible area of the data at each layer using the convolution operator. Here is an illustration of the basic structure of SE block in CNN in Figure 4, where \(\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}\) denote the input data with spatial dimension \(H' \times W'\) and channel dimension \(C'\). Similarly, \(\mathbf{U} \in \mathbb{R}^{H \times W \times C}\) denote the transformed feature maps with spatial dimension \(H \times W\) and channel dimension \(C\). Next, for the sake of simplicity, a natural adaptation of the notation in their paper [11] will be employed with the same idea of omitting all the bias terms in NN.

Thus, for a specific channel in the feature map \(u_c \in \mathbb{R}^{H \times W}\), the following formula can be formed:

\[
u_c = v_c \odot X = \sum_{s=1}^{C'} v^s_c \odot X^{s}
\]

where \(v_c\) refers to the \(c\)-th learned filter in this convolutional layer and \(\odot\) denotes the convolution operator. In addition, the \(F_{ex}\) denote a convolutional mapping layer.

![Figure 4: Basic structure of SE block in CNN [11]](image)

After obtaining the feature maps \(\mathbf{U}\), the SE block is ready for drop-in use without any further assumptions. It also makes the SENet by stacking multiple SE blocks after the convolutional layer possible. Now, it is time to scrutinize the detailed structure of the SE block in CNN. The kernel of the SE block could be divided into three parts: Squeezing mapping, Excitation mapping, and scaling mapping.

The first part of the SE block is designed to acquire global spatial information since there is always a loss in global spatial information when convolutional layers only work on a perceptive area within the spatial field. Thus, Squeezing mapping using the global average pooling technique could extract an overview of the global spatial information. i.e.: \(F_{sq}: \mathbb{R}^{H \times W} \rightarrow \mathbb{R}\), notice the squeezing mapping operate channel-wise through the whole feature maps after the convolutional layer. Let’s denote the \(c\)-th channel-wise statistics by \(z_c\), the corresponding formula is then:

\[
z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)
\]

After gathering the global information and squeeze it into the dimension \(\mathbb{R}^{C}\), a typical NN is used to capture the channel-wise dependencies. In some sense reallocates the computational focus to more discriminant channels of the task within the CNN. Such operation is done by the Excitation mapping \(F_{ex}: \mathbb{R}^{C} \rightarrow \mathbb{R}^{C}\). After squeezing (global pooling), the channel-wise statistics are fed to an NN starting with a Fully Connected (FC) layer. But notice, a bottleneck is proposed with reduction factor \(r > 1\) in this NN to reduce the complexity and serves as a regularizer. Then, an activation function ReLU is applied. After that, to return to the same dimension as the input statistics, another FC layer is followed. Lastly, a simple gating mechanism is employed here. To capture and emphasize multiple channels, the Sigmoid activation function is applied after the second FC layer to extract the channel dependency weights. By observing the extracted weights, it is easy to distinguish which feature(s) has(have) the most discriminant power to the final goal in the main learning task. In a mathematical term, by denoting the whole channel-wise statistics as \(z = [z_1, z_2, ..., z_c]\), the \(F_{ex}\) mapping could be represented as:
\[ t = F_{ex}(z, W) = \sigma(W_2 \cdot z) \]  

where \( W \) denotes the total weights in the two FC layers in the Excitation mapping procedure, \( W_1 \in \mathbb{R}^{C \times C} \) and \( W_2 \in \mathbb{R}^{C \times C} \) denotes the weight matrices for the first FC layer and the second FC layer.

The final stage of the entire SE block is rescaling mapping. After getting the dependency weight of each channel, this weight could then be applied to the original input feature map \( u_c \) by simply multiplying them channel-wise, and the result will be a refined version of feature maps \( \tilde{X} \) (as shown in Figure 4), which could dramatically reduce the final task error [11] when used in CNN for many different tasks. The equation for the scaling mapping is as follows:

\[ \tilde{X} = F_{scale}(U, t) = U \otimes t \]  

where \( t \) is derived from equation (12) and \( \otimes \) denotes the element-wise multiplication.

2.2.2. Effective SE (eSE) module

In this small subsection, one small variant of the SE module will be discussed shortly. The eSE module was first introduced by this paper [12] when they tried to boost the performance of VoVNet. As they mentioned in their [12] paper, the original SE module uses the reduction factor \( r \) to decrease the complexity of the model in the Excitation mapping as described above. However, during the reduction process, the loss of channel information is inevitable. To this end, they [12] propose an eSE module to only use one FC layer and keep the input dimension after Squeezing mapping. Besides this, everything else in the SE module remains. Hence, the new eSE module keeps the Squeezing mapping the same (equation (11)). Nevertheless, the only change happens in the excitation mapping \( F_{ex} : \mathbb{R}^C \rightarrow \mathbb{R}^C \). Instead of equation (12) for \( F_{ex} \), equation (14) will represent the new excitation mapping since only one FC layer exists, and there is no \( W_2 \) in the formula. Meanwhile, in this way, the new module reduces the number of parameters needed and thus increase the module effectiveness as its name suggested. After this, the scaling mapping also keeps the same as state in equation (13).

\[ t = F_{ex}(z, W) = \sigma(W_1 \cdot z) \]  

where \( W_1 \in \mathbb{R}^{C \times C} \) is the weight parameter in FC of the eSE module.

2.2.3. SENet structure

This subsection is devoted to concentrating on SENet. As mentioned above, the SE module could act as a light-weighted drop-in module for a suitable NN after each layer as an attentive reinforcer. Using this definition, any SENet could be regarded as an NN that has one for multiple SE modules. In the previous subsection, how the SENet could be formed in CNN has been carefully discussed, but for later experiments, focusing on the traditional NN is enough. For a simple NN, an urgent need is to somehow remould the SE module structure since there is no global spatial feature. Thus, the squeezing stage is useless for this purpose. But the other two parts (i.e., excitation part and scaling part) need to be kept. The general model of SENet in NN is shown in Figure 5. Since the SE module neither alters NN’s final task nor participates in regularization, it is sufficient to simply keep the loss function the same as the original NN.
3. Methods

This section will build two different types of customized models (denote with C-) and a combined model that combines the previous two customized models in a particular order for later experiments.

3.1. C-AENet

The first customized sub-model needed for later experiments is based on the AENet. To prepare this sub-model, a proper loss function must be chosen. Since in Figure 3, the two main tasks in path 1 and path 2 are the same, the general loss function of this sub-model will be no different to equation (9), but the details of this loss function are waiting to be filled. Based on some empirical tests on the dataset [13], the regularization factor ($\lambda$) is chosen to be 6 and the reconstruction loss function to be the $\ell_1$ norm, i.e.:

$$L_R(X, g \circ f(X)) = L_1(X, g \circ f(X))$$

since dealing with the classification as a final task, a most common classification loss function could be chosen: Cross Entropy Loss Function (CEL) in the task. Before pushing any further, let’s refresh our memory of the definition of the Cross Entropy (CE). The CE of distribution $Q$ relative to distribution $P$ over some set is given by:

$$H(P, Q) = -\mathbb{E}_P[\log Q]$$

where $H(P, Q)$ denotes the CE of $Q$ relative to $P$, since the CE has a similar explanation with Kullback-Leibler divergence (K-L divergence, AKA: Relative Entropy), measures the difference between two distribution, and K-L divergence is also a nice way to define CE, i.e.:

$$H(P, Q) = H(P) + D_{KL}(P||Q)$$

where $H(P)$ is the entropy of $P$.

Now consider a single sample $i$ in the training set of a classification problem. Suppose there are $M$ classes in total. This sample has the class label $c$. Then the Probability Mass Function (PMF) for this particular sample label is the discrete indicator function $\mathcal{I}_c : S \to \{0, 1\}$, where $S = \{0, 1, \ldots, M - 1\}$. Now denote this conditional label distribution as $Y_{ic} = Y_c(\cdot | X = i)$. Hence, it is natural to define the whole classification loss using CE by:

$$\mathcal{L}^{\text{classification}} = \mathbb{E}_X[H(Y_c(\cdot | X), \tilde{Y}(\cdot | X))]$$

where $\tilde{Y}_i = \tilde{Y}(\cdot | X = i)$ denotes the conditional distribution generated by usual Softmax function in the output layer by inputting the same sample $i$ at the start of AENet. For simplicity, let’s rewrite equation (18) as:

$$\mathcal{L}^{\text{classification}} = \mathbb{E}_X[H(Y_c, Y)]$$
With the convention that \( Y_c = Y_c(\cdot |X) \) and \( \hat{Y} = \hat{Y}(\cdot |X) \).

Finally, the new loss function of the customized sub-model could be defined according to equation (9):

\[
\mathcal{L}^{\text{AENet}} = \mathcal{L}(\text{classification}) + \lambda \cdot \mathcal{L}(\text{recon})
\]

by e.q. (9)

\[
= \mathbb{E}_X[H(Y_c, \hat{Y})] + \lambda \cdot \mathbb{E}_X[L_R(X, g \circ f(X))]
\]

by e.q. (3) \& (19)

\[
= \mathbb{E}_X[H(Y_c, \hat{Y}) + \lambda \cdot \mathcal{L}_R(X, g \circ f(X))]
\]

by linearity of \( \mathbb{E}_X \)

\[
= \mathbb{E}_X[-\mathbb{E}_{Y_c}[\log \hat{Y}] + \lambda \cdot \mathcal{L}_R(X, g \circ f(X))]
\]

by e.q. (15) \& (16)

\[
= \mathbb{E}_X[-\mathbb{E}_{Y_c}[\log \hat{Y}] + \lambda \cdot \|X, g \circ f(X)\|]
\]

by e.q. (5)

Then plug in \( \lambda = 6 \) as chosen above:

\[
\mathcal{L}^{\text{AENet}} = \mathbb{E}_X[-\mathbb{E}_{Y_c}[\log \hat{Y}] + 6 \cdot \|X, g \circ f(X)\|]
\] (20)

Until now, such customized model could be summarized as a sub-model of AENet that has the same structure as shown in Figure 3 (path 1) with a specific loss function defined in equation (20) and designed for solving the supervised classification problem. From now on, this customized model and the AE module in it will be referred as C-AENet and C-AE module respectively in the rest of this paper.

3.2. C-SENet

In this subsection, a customized sub-model based on SENet will be proposed. Because the main job in later sections focuses on a classification problem, the customized model will be designed to accommodate all the needs of experiment to solve a classification problem in the end. Though applying SE module multiple times is still possible, due to the purpose of comparison with other model and the consideration of effectiveness, the SE module will only be used once in SENet. A detailed illustration about this model can be found in Figure 6.

![C-SENet structure](image)

Figure 6: C-SENet structure

As Figure 6 shows, by putting the SE module right after the input layer, it is easy for this model to extract the most discriminant input feature(s) relative to the task label. such a goal could be easily achieved by observing the dependency weights encoded in SE module as analyzed in section 2.2.1.

A small detail must be pointed out before moving any further, you might notice the C-SE module is in place of SE module. It’s because this special SE module is also customized module formed by combining the idea of eSE module and SE module. This customized SE model is referred as C-SE module with detailed structure shown in Figure 8. Like the eSE module creator said, the loss of information is inevitable when the bottleneck is introduced in excitation mapping. Still, for the sake of later experiment, it is better to preserve or even magnify the original SE module’s learning ability with minimum modification. Thus, by introducing an expanding factor \( e > 1 \) instead of reduction factor \( r \) in the excitation mapping, the SE module are given a better learning ability controlled by expanding factor...
$e$ to capture the feature dependency weight. Besides this, all the other components in SE module remain the same. It is also not hard to see that all the equations described in the SE module are still valid in C-SE module, except a minor change has to be made in denoting the FC weights. i.e.: $W_1 \in \mathbb{R}^{e \times C}$ and $W_2 \in \mathbb{R}^{C \times e \times C}$. The visual structure of C-SE module could be found in Figure 7.

**C-SE Module for NN**

![C-SE Module for NN](image)

**Output Dimension**

$C \quad e \cdot C \quad e \cdot C \quad C \quad C \quad C \quad C'$

Figure 7: C-SE Module for NN

As shown in the green square in Figure 6, a bunch of FC layers are followed by the crucial SE module to complete the whole NN model. The loss function for such a model is defined by CE as described in equation (19) since the main task is the classification. From this moment, the C-SENet is used to denote this specific SENet, as shown in Figure 7.

3.3. Combined model

This last subsection will discuss the combined model of the AENet introduced in section 2.1.2 and the SENet introduced in section 2.2.3. It is the characteristics of light-weight, drop-in and versatile of these two modules that make the model combination possible. Thus, in theory, it is possible to combine them in many different ways by permuting these two modules based on the depth of the NN. However, only one of those many possibilities is chosen here to propose a customized AESENNet model for later sections.

### 3.3.1. C-AESENNet

At last, the final customized sub-model will be given, of which the basic construction is given by Figure 8. This particular model is chosen because of the intention of deducing the degree of perfection to compression by the weights of input features, as mentioned in the introduction section. Such model could take advantage of the SE module and determine whether the feature dependency weight(s) or ranking has changed after compression. Finally, to make a fair comparison model, it is necessary to keep all the customized parameters as the same. Hence, this model will use the C-AE module and C-SE module as described previously. The total loss function will be the same as the equation (20). From now on, this highly customized model with all the above hyperparameters should be called as C-AESENNet. The detailed structure is shown in Figure 8.

**C-AESENNet**

![C-AESENNet](image)

Figure 8: C-AESENNet structure
4. Experiment
In this section, a detailed discussion about the experiments using the methods described in the previous section will be presented.

4.1. Base Model
A two-layer neural network is built to learn from raw data and predict the classification. The input of this model is the original data (raw data), which contains 10 features and one label. The training dataset has 12 classes, and the task is to divide the samples into these 12 classes.

As is shown in Figure 9, in the linear part of the Base Model in the first hidden layer, the number of input features is 10 (corresponding to the 10 dimensions of raw features), and the number of output features is 20. Afterwards, the Rectified Linear Unit (ReLU) activation function was used. In the second hidden layer, the number of input features in the linear part corresponds to the 20 output features of the previously hidden layer, and the number of output nodes is 12 (#class). Finally, use the Softmax function, which is a standard way to generate weights, as the activation function of the second hidden layer to calculate the probability of each category and select the category with the largest probability.

In the process of backpropagation, a standard setup for multi-class NN model training is chosen. Set the Cross Entropy as the Loss function of the Base Model, see formula (19) for details, and choose the training optimizer to be Adam [14] with customized initial learning rate 1e-2.

Through many experiments, it is found that the result of the Base model will converge when the epoch is 1000. Therefore, an epoch of 1000 is used as the experimental basis to obtain test accuracy results. The accuracy of prediction based on Baseline floats around the average level of 76.84%.

4.2. Models for data compression
(1) PCA
“Principal Component Analysis” (PCA) is a typical dimension reduction method applied to extract the main feature components of the data and reduce the number of the attributes. PCA transforms the possibly related features into a group of linear independent variables by orthogonal transformation. The mathematical derivation of PCA can be carried out from the perspective of maximum separability.

Here is an overview of the derivation. Suppose that there are N data records with dimension D, and such dataset is denoted with \( \mathbf{X} \in \mathbb{R}^{N \times D} \). The symmetrical covariance matrix \( \Sigma_X \in \mathbb{R}^{D \times D} \) of such dataset \( \mathbf{X} \) is given by:

\[
\Sigma_X = \frac{1}{N} (\mathbf{X} - \mu_X)^T (\mathbf{X} - \mu_X)
\]

(21)

where \( \mu_X \in \mathbb{R}^{N \times D} \) is the mean matrix of \( \mathbf{X} \) averaged over all data points for each feature. Using the spectral theorem from linear algebra, it’s known that there are D distinct eigenvectors and D corresponding eigenvalues of \( \Sigma_X \). Hence, the eigenvector matrix could be formed by placing each eigenvector in a column with descending order of corresponding eigenvalues. i.e.:

\[
\mathbf{V} = \begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_D \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_D \end{bmatrix}
\]

(22)
where $\hat{v}_i$ is the $i$-th eigenvector and $\lambda_i$ is the corresponding $i$-th eigenvalue with $\lambda_1 > \lambda_2 > \cdots > \lambda_D \geq 0$ (positive definite property). Now, if defining the following equation as the non-reduction PCA transformation:

$$ Z = XV $$

with the transformed dataset $Z \in \mathbb{R}^{N \times D}$ and $V$ from equation (22), it’s prepared to calculate:

$$ \Sigma_Z = \frac{1}{N} (Z - \mu_Z)^T (Z - \mu_Z) \quad \text{by Def of } \Sigma_Z $$

$$ = \frac{1}{N} (XV - \mu_X V)^T (XV - \mu_X V) \quad \text{by e.q. (23)} $$

$$ = \frac{1}{N} [(X - \mu_X)V]^T [(X - \mu_X)V] \quad \text{by distribution law} $$

$$ = \frac{1}{N} V^T (X - \mu_X)^T (X - \mu_X) V $$

$$ = \Lambda V \Lambda \quad \text{by e.q. (21)} $$

$$ = \Lambda \quad \text{by Def of eigenvalue} $$

Overall:

$$ \Sigma_Z = \Lambda $$

As it could be seen from the above calculation, the covariance of the transformed dataset $Z$ is simply the eigenvalue matrix with descending order. Thus, the features in the transformed dataset do not correlate with each other since non-diagonal items in $\Sigma_Z$ are 0. This way, each feature is maximally separated. Meanwhile, the $i$-th ($i \leq D$) eigenvalue represents how much information is contained in the $i$-th transformed feature. In practice, it’s possible to chop off the minor eigenvalues in $\Lambda$ as well as the corresponding eigenvectors in $V$ and only keep the most important features in the dataset to reduce the dimensionality of data. If the dropped feature is considered small enough to reconstruct the original dataset with a minimum loss, then the PCA process is also a feature compression process. One worth-noting fact is that the PCA could be considered a special case of the AE module mentioned in section 2.1.1, where the encoder $f$ and decoder $g$ are linear. In other words, PCA only finds a low dimensional hyperplane to compressed data while the general AE module learns a non-linear manifold [8].

In this work, PCA algorithm is used to preprocess the original data. In the processed data, the principal components are arranged in descending order of contribution and all the dimensions from the 1st dimension to the 9th dimension are kept for further experiments. In each experiment, starting from the principal components with the largest contribution rate, $n$ (1-9) dimensions of the principal components are selected sequentially as the input to the Base Model mentioned in 4.1 to get the following results.

It can be seen from the results of the PCA algorithm that every time the principal component of a dimension is added, the accuracy of the test will be improved compared to the result of the previous dimension. Retaining the top nine principal components obtained an average test accuracy rate of 75.54%, which is slightly lower than the Baseline’s 76.84% test accuracy level. This shows that the 10-dimensional features selected all contribute information to the result of the label. When the selected features are almost all related to the label and have a certain information contribution rate, the subsequent feature extraction work is necessary.

(2) SVD

“Singular Value Decomposition” (SVD) [15] algorithm is a method to reduce the dimensionality of raw data. Assuming that the raw dataset is a matrix $X \in \mathbb{R}^{N \times D}$ ($D$ is the number of raw features, $N$ is the number of samples in the training set), then the formula (25) could be used to decompose $X$. $U \in \mathbb{R}^{N \times N}$ in the formula represents a set of orthogonal column vector bases, expressing the description information extracted from samples. $S \in \mathbb{R}^{N \times D}$ represents a matrix composed of singular values arranged in descending order on the diagonal, which can reflect the weight of each column vector base.
in the original matrix. \( \mathbf{V}^T \in \mathbb{R}^{D \times D} \) represents a set of orthogonal row vector bases and expresses the descriptive information extracted from features.

\[
\mathbf{X} = \mathbf{USV}^T
\]

\[
\mathbf{X} = \begin{bmatrix}
\mathbf{u}_1 & \cdots & \mathbf{u}_N \\
0 & \cdots & 0
\end{bmatrix}
\begin{bmatrix}
\sigma_1 & \cdots & 0 \\
0 & \cdots & \sigma_D
\end{bmatrix}
\begin{bmatrix}
\mathbf{v}_1^T & \cdots & \mathbf{v}_D^T
\end{bmatrix}
= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \cdots + \sigma_D \mathbf{u}_D \mathbf{v}_D^T
\]

As it could be seen from the above calculation, it is possible to approximate the original dataset \( \mathbf{X} \) directly by an appropriate number of features \( k \) using the following formula:

\[
\hat{\mathbf{X}}_t = \sum_{\ell=1}^{k} \sigma_\ell \mathbf{u}_\ell \mathbf{v}_\ell^T
\]

where the \( \hat{\mathbf{X}}_t \) represent the \( t \) ranks reserved approximation of original dataset \( \mathbf{X} \).

Sequentially, \( n \) (1-9) dimensions of reserved ranks are chosen as the input to Base Model mentioned in 4.1 to get the following outcome.

The results of the SVD algorithm are quite consistent with the PCA algorithm as shown in Figure 10, and the test accuracy rate gradually increases with the increase of the retained dimension. From Figure 10, it can be found that the PCA process retains the components of the first seven dimensions, and the SVD process retains the ranks of the first seven dimensions can achieve better test accuracy results. When the PCA and SVD processes are retained to 9 dimensions, the compressed data has been close to the effect obtained by raw data in the Base Model and is better than retaining 7 and 8 dimensions.

Figure 10: Text accuracy of PCA, SVD and Baseline

(3) AENet

According to the structure of the AENet shown in Figure 3-path 1, Base Model is used in the position of the N block of the NN block to form a C-AENet for the experiment. The specific content of the model C-AENet is shown in section 3.1. It is hoped to achieve information compression through this unsupervised learning module and input the compressed information into the Base Model for learning.
It can be seen in Figure 11 that when the data is compressed to each dimension, the test accuracy of C-AENet is better than the PCA and SVD methods, especially when the data is compressed to low dimensions such as 1 to 3 dimensions. C-AENet uses data compression to reduce dimensionality. Instead of discarding the last dimension similar to PCA and SVD algorithms, there is less information loss and a better interpretation of the label. Also, PCA and SVD process both linearize raw data, while the C-AE module’s processing of raw data is non-linear. Generally speaking, nonlinear processing can compress data more optimally than linearization processing, and retain more explanatory information components for labels.

As shown in Figure 11, the test accuracy of the C-AENet model increases as the number of remaining dimensions in the reconstruction part increases. This is because all the features in this data have relatively significant information contributions to the label. C-AENet can already achieve the same test accuracy as the Base Model when the data is compressed to 8 dimensions, and the test accuracy when the data is compressed to 9 dimensions has reached 77.07%. Taking the error of the experiment into account, it is believed that when the data is compressed to 9 dimensions, C-AENet and Base Model have the same level of effect. It means that when the data in C-AENet is compressed to 9 dimensions, the information contained in the compressed data can retain the level of interpretation of the label by the original data.

In summary, it can be found that comparing PCA and SVD, C-AENet has obvious advantages in data compression. Therefore, C-AE module is selected as the selected data compression model for subsequent experiments. At the same time, choosing to compress the raw data to 9 dimensions in C-AENet, so that the compressed data will eventually interpret the label at the same level as the raw data as is reflected in Figure 11.

Now the parameter of C-AENet is adjusted to accommodate the experiment. First of all, raw data is put into C-AENet as the input. To create the encoder structure shown in Figure 3, the original 10-dimensional data is amplified to 20 dimensions in the first hidden layer of the C-AENet model. Due to the conclusion made from Figure 10, the 20-dimensional data is compressed to 9 dimensions through this hidden layer shown in Figure 3. Because the C-AE module is a neural network with symmetric hidden layers, data must continue to enter the decoder process. The compressed 9-dimensional data is
symmetrically enlarged to 20 dimensions and returns to the original 10 dimensions in the next layer. All activation functions in C-AE module are ReLU.

In this experiment, the hyperparameter $\lambda$ is set to 6 and the Loss formula (20) is applied. It is worth mentioning that the average reconstruction loss in the experiment is 0.07458. Since Z-score normalizes the original data, the size of the original data is between 0-1. The reconstruction loss obtained through the C-AE module is less than 0.1, which can indicate to a certain extent that the C-AE module controls the loss after data compression in a small range.

4.3. C-SENet for feature selection

Referring to Figure 6, the C-SENet model is used. Base Model is added to the position of the NN block. The raw data in this experiment is input into the C-SE module as the input layer in Figure 6.

After establishing the C-SENet model in the experiment, referring to Figure 9, the hyperparameters of $\epsilon=1.6$ are used in the fully connected layer to expand the features. Therefore, in the C-SE module, the number of input nodes is 10 (the number of raw features), and the number of output nodes is 16 for the first hidden layer. In the second hidden layer of the C-SE module, the number of input nodes is 16, and the number of output nodes is 10.

Through the two-layer C-SENet network, the relative information contribution degree of each feature, weights, can be obtained. In the C-SENet model, C-SENet generates a set of weights for the ten input features and assigns weights at the feature level. The output weights of the excitation stage of C-SENet are regarded as the importance of each feature channel after the feature selection process.

As shown in Figure 12, the weights of each feature in the C-SENet model is arrayed in descending order. On average, the feature “lower margin” has the highest information weight, followed by the upper margin, row number, intercolumnar distance, exploitation, peak number, modular ratio, weight, modular ratio/interlinear spacing and interlinear spacing.

![Figure 12: Average weights of features for C-SENet](image)

Through Figure 13 of comparison of the average test accuracy with C-SENet and Base Model. The C-SENet has an average accuracy of 85.78%, which far exceeds the average test accuracy of Baseline 76.84%.

The test accuracy is improved compared to Base Model because the C-SENet process added in front of the Base Model selects the most discriminant features through the process of excitation. Also, C-
SENet uses different weights to reweight the features to filter out the processed data that is more suitable for the labels than the original features. The features with a higher contribution rate are relatively enlarged when they enter the Base Model after the C-SE module is processed. This part of the features has better explanatory power. The features with a low relative contribution rate are either not selected during the excitation process or are given a lower weight during the reweighting process, thereby reducing the influence of the information they contain on the test results.

After the above process, C-SENet acquires and amplifies the information contained in the features that are more related to the label and also removes some of the information contained in the features that are less related to the label. At this point, the information entering the neural network where the baseline is located can better interpret and fit in the label, and higher test accuracy could be reached by the C-SENet model.

![Baseline Vs SENet](image)

**Figure 13: Test accuracy of Baseline and C-SENet**

### 4.4. C-AESENet for feature selection after data compression

As shown in Figure 8, C-AESENet sequentially combines the C-AE module, C-SE module and Base Model (put in NN block). Raw data act as the input layer in Figure 8 and enter the C-AE module directly. After raw features are compressed by the C-AE module, they are then used as input to the C-SE module for feature selection and information contribution rate acquisition. Finally, after the Base Model in the NN block, the test accuracy of C-AESENet will be generated and compared with Baseline.
Figure 14 shows the average test accuracy results of Base Model, C-AENet, C-AESENet and SENet. It can be seen that at the same level, the effect of C-SENet is better than C-AESENet and better than AENet. Under the same number of epochs (1000), the test accuracy of C-AENet and Base Model is at the same level, while test accuracy of C-AESENet and C-SENet is significantly higher than that of Base Model. It can be proved that the feature attention models used are all effective. Information that is more closely related to the label can be extracted through C-SENet. It is possible to reduce the dimensionality of the data when the intensity of the data interpretation remains unchanged.

Figure 15 shows the average weights of features for C-SENet and C-AESENet.
From Figure 15&16, it’s easy to tell that C-SENet and C-AESENet have roughly the same selection of features. In fact, the weights of each feature obtained in C-SENet and C-AESENet have experimental errors. Contingency and experimental error will affect the weights of features obtained in each experiment. In general, the features extracted from C-SENet and C-AESENet are highly uniform. "Lower margin", "upper margin", "row number", "intercolumnar distance", "exploitation" and "peak number" are features that have a greater contribution to label and stronger discriminant power. The other four "modular ratio", "weight", "modular ratio/interlinear spacing" and "interlinear spacing" have relatively low influence on the label.

Since C-SENet and C-AESENet have a high similarity to the weights produced by each feature, it’s appropriate to think that the data compressed by the C-AE module and the raw data also have high similarity in the retained information. This can prove that in the process in the C-AE module of compressing the information, the reconstruction loss is tiny. In other words, the C-AE data compression method used in the experiment has a relatively small degree of damage and loss of useful information. These all reflect the degree of perfection to compression and reconstruction in the C-AE data dimensionality compression method are quite high.

5. Conclusion
In this paper, an unsupervised feature learning method is proposed for the data classification task. The model consists of two main parts, namely a Customized autoencoder network (C-AENet) and a customized squeeze-and-excitation network(C-SENet). The C-AENet compresses the feature map while maintaining most information, and C-SENet builds a more effective representation by outputting a channel weight. Experimental results on the Avila dataset show that the method has great potential to provide a robust feature representation for data classification, which can be widely adopted in most classification tasks.

References
[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, “Deep learning”, MIT press, pp 800,2016.
[2] Madhumitha Ramamurthy,Y. Harold Robinson, S.Vimal,A.Sureshd, “Auto encoder based dimensionality reduction and classification using convolutional neural networks for hyperspectral images”, Microprocessors and Microsystems, Volume 79, Nov,2020.
[3] Mathias Seuret, Michele Alberti, Rolf Ingold, and Marcus Liwicki. “PCA-Initialized Deep Neural Networks Applied To Document Image Analysis”, arXiv:1702.00177v1 [cs.LG] 1 ,Feb, 2017.
[4] Jongchan Park, Sanghyun Woo, Joon-Young Lee, In So Kweon, “BAM: Bottleneck Attention Module”, arXiv:1807.06514 [cs.CV], 17, Jul.2018, Available: https://arxiv.org/abs/1807.06514

[5] Sanghyun Woo, Jongchan Park, Joon-Young Lee, In So Kweon, ”CBAM: Convolutional Block Attention Module”, arXiv:1807.06521[cs.CV], Tue, 17, Jul. 2018, Available: https://arxiv.org/abs/1903.06586.

[6] Xiang Li, Wenhai Wang, Xiaolin Hu, Jian Yang, “Selective Kernel Network”, arXiv:1903.06586 [cs.CV], 15, Mar. 2019, Available: https://arxiv.org/abs/1903.06586.

[7] Rumelhart, D.E., Hinton, G.E., Williams, R.J, “Parallel distributed processing: Explorations in the microstructure of cognition”, vol. 1. Foundations, pp. 318–362, Jan. 1986.

[8] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” 2020.

[9] Lei Le, Andrew Patterson, Martha White, “Supervised autoencoders: Improving generalization performance with unsupervised regularizers”. Neural Information Processing Systems (NIPS), pp 107-117, Dec. 2018.

[10] Yuting Zhang, Kibok Lee, Honglak Lee, “Augmenting supervised neural networks with unsupervised objectives for large-scale image classification”, Proceedings of The 33rd International Conference on Machine Learning, vol. 48, pp 612-621, 21, Jun. 2016.

[11] Hu J, Shen L, Albanie S, Sun G, Wu E, “Squeeze-and-Excitation Networks” IEEE transactions on pattern analysis and machine intelligence [IEEE Trans Pattern Anal Mach Intell]. Vol. 42 (8), Aug. 2020.

[12] Y. Lee and J. Park, “CenterMask : Real-Time Anchor-Free Instance Segmentation,” arXiv:1911.06667 [cs], Apr. 2020, Available: https://arxiv.org/abs/1911.06667.

[13] C. De Stefano, M. Maniaci, F. Fontanella, A. Scotto di Freca, “Reliable writer identification in medieval manuscripts through page layout features: The ‘Avila’ Bible case”, Engineering Applications of Artificial Intelligence 72, p 99-110, 2018.

[14] D. Kingma and J. Lei Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,” Jan. 2017. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf.

[15] G.H. Golub and C. Reinsch, “Singular Value Decomposition and Least Squares Solutions”, Numerische Mathematik, Number. Math. 14, pp 403-420, April. 1970.