Image compression and classification using 3-layer neural network and application of SVD

Yufei Lu
Research School of Computer Science, Australian National University, Canberra Australia
U6920336@anu.edu.au

Abstract

For the image compression, we would use a 3-layer-neural network to compress a set of image features and set the target equal to the input.[1] In addition, for the pruning strategy, the Singular Value Decomposition will be used to compress the weights between the hidden layer and the output layer. The parameters such as the number of epochs, the number of units of the hidden layer in the original network and the required accuracy of the SVD will be tested by different values and displayed in the paper.

Key words*: image compression, 3-layer neural network, SVD*

Introduction

Image compression:

Image compression is a traditional and developed technique. In the process of image compression, the features of an image will be extracted first, then the information of the images will be encoded or represented using less information.

Taking JPEG for example, it uses 8x8 block in Discrete Cosine Transform (DCT), then adapts run-length encoding to exploit the sparsity pattern of extracted frequency coefficients. Quantization is applied on coefficients to realize different compression level. [5]

Dataset:

In our dataset, SFEW (Static Facial Expressions in the Wild), the features have been provided in the form of .xlsx tables. Each term in the table consists of 12 columns. The first column represents the image names. The second column indicates the label of the image, the number(from 1 to 7) indicates the category of the image. The 3rd to the 7th column are the First 5 Principal Components of Local Phase Quantization (LPQ) features Local Phase Quantization (LPQ) features. The 8th to the 12th columns are the First 5 Principal Components of Pyramid of Histogram of Gradients (PHOG) features.[4]

Local phase quantization (LPQ) is an extension of local binary pattern (LBP), which is the basic of facial expression recognition. The pyramid of histogram of oriented gradients (PHOG) is an extension of HOG, which counts occurrences of gradient orientation in certain portions in an image.[4]

SVD:

SVD is a kind of decomposition of a m*n matrix(m<n), which can be represented in the form of a m*m orthogonal matrix multiply a m*n matrix, consists of a diagonal matrix and rest numbers are zeros, multiply a n*n orthogonal matrix. We can decompose the whole formula to make an approximation of the m*n matrix by a m*k matrix multiply a k*k matrix multiply a k*n matrix where k<m. [2]

Using SVD, we can decompress the weights matrix(m*n) between the hidden layer and the output layer to 3 matrices and the number of weights is (m+n)*k to make (m+n)*k<m*n. [2]

$$A_{m\times n} = U_{m\times m}S_{m\times n}{V^T}_{n\times n}$$
$$A_{m\times n} \approx {U'}_{m\times k}{S'}_{k\times k}{(V^T)'}_{k\times n}$$

To get $S'_{k\times k}$, we sort the singular values on diagonal of $S_{m\times n}$, which consists of a diagonal matrix and the rest are zeros. And extract the first k values then combine them with corresponding values in $U_{m\times m}$ and $V^T{}_{n\times n}$.[2]

Model:

Our network is the 3-layer fully connected model made up with input layer, hidden layer and the output layer. The numbers of units in input layer and the output layer are both 11. The number of units in hidden layer is less than 11. We will discuss the selection of the number below according to the accuracy and the loss. The input is the features (label, the First 5 Principal Components of LPQ and the First 5 Principal Components of PHOG) of a certain image in the SFEW dataset. The target output is identical to the input. For the first unit of input and output, what we apply is classification. For all units, we will apply image compression. Adam will be the optimizer of the model.

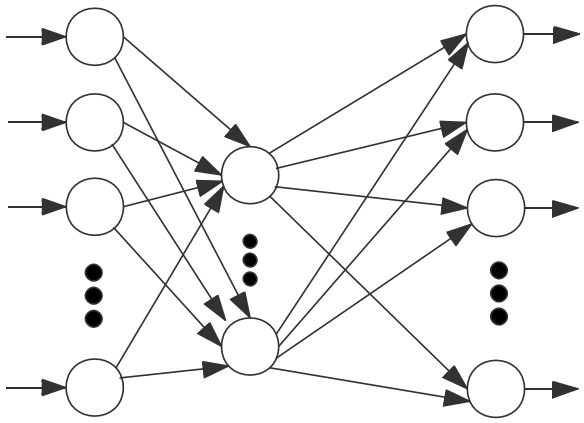


Fig. 1. Feed-forward network of three layers of processing units. The layer on the left hand is input layer; the layer on the right hand is output layer; they have the same neuron number. The hidden layer in the middle has fewer neurons.

Method

Preprocessing:

Ignoring the 1st column, we extract the rest 11 columns as our dataset. Then we fill 0s to all the NaN values in our dataset. Our inputs are the 11 features of data, which is identical to our target output. We will split our dataset as training set and test set by the in-built function.

As for the model, we define an input layer with 11 units, a hidden layer and an output layer with 11 units. The number of neurons in the hidden layer is less than 11. We will compare the results of different value of it below. As described, the network is fully connected. We choose to use Sigmoid as the activation function of the hidden layer.

Train:

In training process, we just train the network using back-propagation. The back-propagation method would be slow, the shortcomings will be discussed blow.

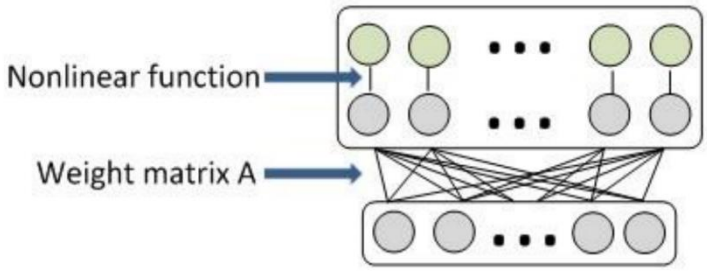
Decomposition:

For our trained model, we can regard the weights from the hidden layer to the output layer as a weight matrix. We may apply SVD to the weight matrix to reduce the number of the weights to improve the speed. The SVD process can be accomplished by an in-built function. We then can choose k value to decompose the m*n

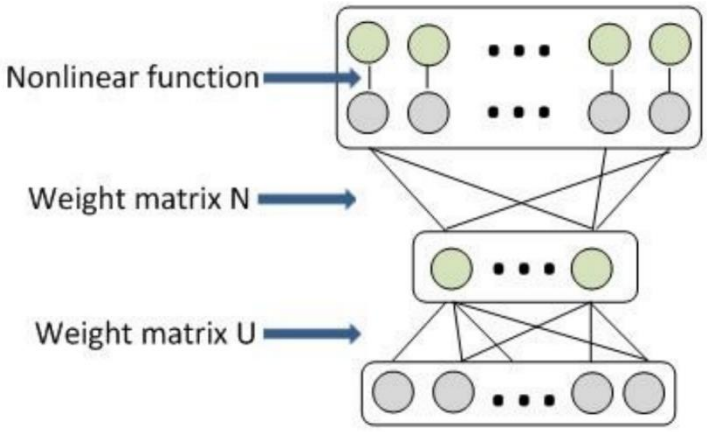
matrix to 3 matrices (m*k, k*k, k*n). In the model, we may add an extra layer to represent the singular values. We can set the bias of the extra layer as the k singular values. Besides, the weight matrix between the hidden layer and the extra layer can be indicated by the m*k matrix. The weight matrix from extra layer to the output layer can be illustrated as the k*n matrix.

Test:

We can test our original network (the one without SVD) and compressed model (the model with SVD) then compare the result. In this experiment, I chose the accuracy of classification (only for the first output of the output layer) and the mean square loss. Additionally, for the classification, as the target output is integer, we will convert our first output in the output layer to integer then compare the value with the target output. If they are equal, the result is correct. If not equal, the result will not be considered as correct. The classification accuracy is (number of correct outputs/number of all outputs).



(a) One layer in an original model. [3]



(b) Two corresponding layers in a new compressed model. [3]

Fig. 2. Model conversion in a neural network before and after SVD. [3]

Results and discussion

Decide the number of training epochs:

Firstly, we fix the compression accuracy to 0.9 and the number of units in hidden layer to 10, then try different number of epochs from 10 to 500, the step is 10(10, 20, 30, ..., 500). For each group, we iterate for 50 times then we use the mean values of classification accuracy and mean square loss.

The result is displayed here:

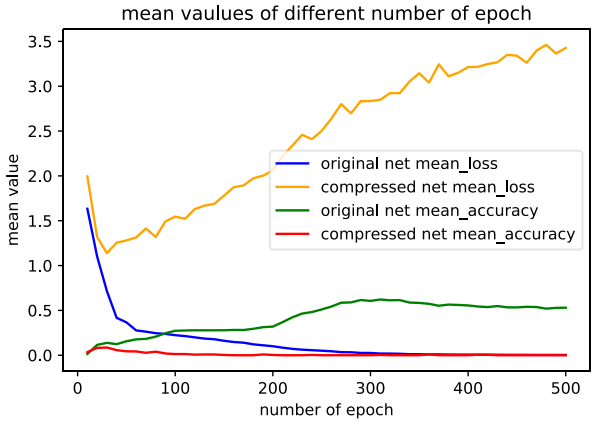


Fig. 3. Mean value of loss and classification accuracy using different number of epochs, with compression accuracy set to 0.9 and hidden units number set to 10.

We can observe that for the original network, the classification accuracy will go up first then decrease at where the number of epochs is about 300. The test loss will go down first then slightly climb at about where epoch number is 300. For the compressed network, our classification accuracy will increase then reduce at where epoch number is about 30. The loss will decrease first then obviously increase at the point where epoch number is 30.

Since our goal is to compress the network, we can choose the number of epochs 30.

Decide the number of units in hidden layer:

Firstly, we fix our compression accuracy as 0.9 and the number of training epochs as 30. We try different unit number in hidden layer, from 10 to 2.

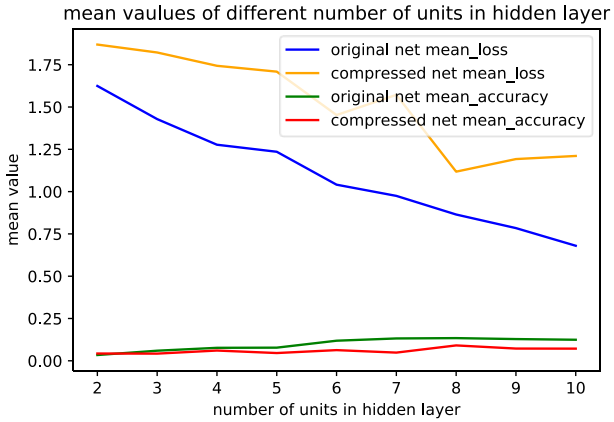


Fig. 4. Mean value of loss and classification accuracy using different number of hidden units, with compression accuracy set to 0.9 and epoch number set to 30.

We can observe from the figure that as the number of units in hidden layer grows, the loss of original and compressed network will both decrease. The classification accuracy of the two networks will both go up. We can observe that when unit number is 8, there is an obvious decrease in loss for compressed network.

So, we may choose 9 as our hidden unit number.

Decide the required compression accuracy:

Firstly, we fix our num of hidden layer to 9 and the number of epochs to 30. We try different accuracy from 0.5 to 0.9, the step size is 0.1. We will compare the time cost of original network and compressed network using different required compression accuracy. For each accuracy value, we loop for 50 times then calculate the mean value.

The results are shown below:

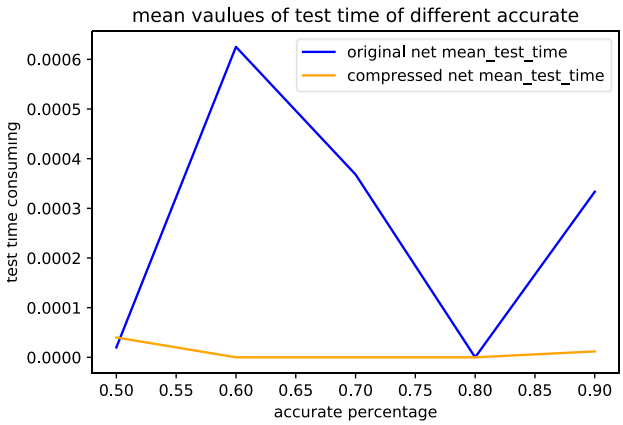


Fig. 5. Mean value of test time using different compression accuracy, with hidden unit number set to 8 and epoch number set to 30.

We can observe that if we choose our compression accuracy as 0.7 the compressed network will be less time-consuming, where the k value is 3. The 9*11 weight matrix in the original network will be replaced by a 9*3matrix and a3 81 matrix. The total weight number will be compressed from (9*11) to (9+11)*3.

So, we may choose 0.7our required classification accuracy. Notice that because the train and test set are different every time the program runs, the results might be slightly different. For example, when we choose compression accuracy as 0.7, the k value night be 3 or 4. But the improvement of the test speed can still be observed.

Discussion:

There are some drawbacks of back-propagation. The main shortcoming of the back-propagation is that it can be slow to train networks, and that the architecture required for a solution to a problem is not currently determinable *a priori*.[1] In our SVD process, we can only improve the speed in the test section after the training process.

And there is no comparison with the distinctiveness of the original paper (T.D. Gedeon & D. Harris, “PROGRESSIVE IMAGE COMPRESSION”). In this paper, the comparison of angles between vectors is applied to determine which vector to prune. If the angle between two vectors are too small or too large to complementary, one of the two will be prune in the network.

Conclusion and future work

Conclusion:

The SVD can be applied to the 3-layer network image compression. And it is efficient for saving time because we have tested it.

Future Work:

We may compare the effects using SVD and the pruning method(distinctiveness) in the original paper about Progressive Image Compression using angles between vectors to judge the distinctiveness of vectors to

prune.

Another aspect is that the SVD is usually used in Deep leaning, so we may try to create a deep learning model to apply the SVD and compare its effects with our 3-layer model.[3]

References

[1] T.D. Gedeon & D. Harris, “PROGRESSIVE IMAGE COMPRESSION”

[2] Bernhard Bermeitinger, Tomas Hrycej, Siegfried Handschuh, “Singular Value Decomposition and Neural Networks”, 2019

[3] Changliang Liu, Jinyu Li, Yifan Gong, “SVD-based Universal DNN Modeling for Multiple Scenarios”, 2015.

[4] Abhinav Dhall, Roland Goecke, Simon Lucey and Tom Gedeon, “Static Facial Expression Analysis in Tough Conditions: Data, Evaluation Protocol and Benchmark”.

[5] Jing Zhou, Akira Nakagawa, Keizo Kato, Sihan Wen, Kimihiko Kazui, Zhiming Tan, “Variable Rate Image Compression Method with Dead-zone Quantizer”, 2004.