Perceptual Data Compressor Based on Progressive Image Compression

Simian Wang
Research School of Computer Science, Australian National University, Canberra, Australia, 0200
u6165791@anu.edu.au

Abstract. In modern society, the core algorithms of many applications are based on the analysis of high-dimensional data. As the dimension of data increases, the computation cost will be increased exponentially, which becomes a major obstacle for the development of algorithms. At the same time, saving and loading large amounts of high-dimensional data is also space-consuming for electronic devices. For solving this issue, I propose a perceptual data reduction model named Progressive Perceptual Data Compressor (PPDC), based on the use of neural networks. The whole model has one hidden layer and consists of 2 parts: the compressor model, and the decompressor part. The compressor part will compress the input vectors into a lower dimension, while the decompressor can decompress the low-dimensional vectors generated by the corresponding compressor. The idea of perceptual loss (Johnson, et al., 2016) is introduced in training stage. The dataset to apply PPDC on, is the SARS dataset which comes from the study of aggregation in fuzzy signature (Mendis, Gedeon, & Koczy, 2005). It contains the clinical symptoms of 4000 patients, 1000 of them are normal(healthy), 1000 have high blood pressure, 1000 have pneumonia, and the rest 1000 have SARS. Each patient record has 23 symptoms and can be viewed as 23-dimensional data.

Keywords: Neural Networks, Data Dimension Reduction, Unsupervised Learning, Perceptual Loss.

1 Introduction

Due to the epidemics of COVID-19, the medical system of each country across the world are under significantly high pressure trying to bring the situation under control. In general, that means their IT systems need to save massive high-dimensional data such as the clinical symptoms of the potential COVID-19 patients for determining their states of health. Saving high-dimensional data like these will consume a high amount of storage space and make the future big-data analysis computation-consuming. This concern drives me to choose the SARS dataset, and consider if there is a way to reduce the consumption of using and saving high-dimensional data in the medical area.

Apparently, data compression can solve this problem well, by reducing the high-dimensional data to a relatively lower dimension. The problem my model will solve is: With the use of neural networks, high-dimensional data can be compressed to low-dimensional data, and the low-dimensional data can be recovered back to high dimension while the recovered data is close to the original high-dimensional data in Euclidean distance. There are several existing data compression methods based on neural networks: progressive image compression method (Gedeon & Harris, 1992) and image compression based on shared weights and bidirectional networks (Gedeon, et al., 1997). Since the data dimension in our case is lower than the cases of image compression, the progressive image compression method is preferred since it requires lower computational resources, and capable of dealing with 23-dimensional data.

The progressive image compression is made up of 2 parts: one is the compressor, while the other one is the decompressor. Both parts are made up of fully connected neural networks. The procedure of compression is: the input image matrix is divided into non-overlapping fixed-sized square matrixes; and then, the compressor will flat each square matrix into a high-dimensional vector, then compress each vector to a lower-dimensional vector. For decompression, the decompressor part will convert the compressed vector back to the same dimension as the input of the compressor.

As for the idea of 'progressive': after one epoch of training the compressor and decompressor, some weight vectors of the compressor will be pruned based on distinctiveness between the hidden units in the hidden layer they created. If at least one vector is pruned, the output dimension of the compressor will be reduced, then compress the input vector in a lower dimension than in this round. The general workflow is shown in Fig. 1..

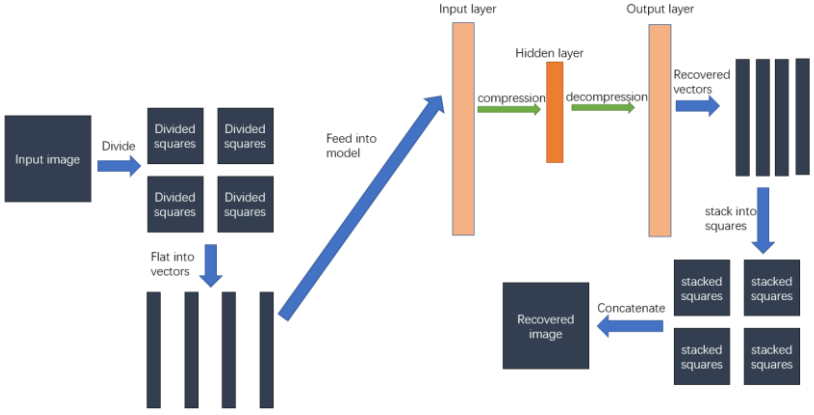

Fig. 1. The workflow of Progressive Image Compression method

My proposed model PPDC uses a similar idea progressive image compression. In my method, there will be a single hidden layer in the neural network, one input layer, and one output layer. And, there will be no operation applied to input and output vectors since the data is originally in vector form instead of an image. The overview structure and workflow are illustrated in Fig. 2. The optimization is based on the mean squared error between the recovered vector from the decompressor and the original input vector of the compressor.

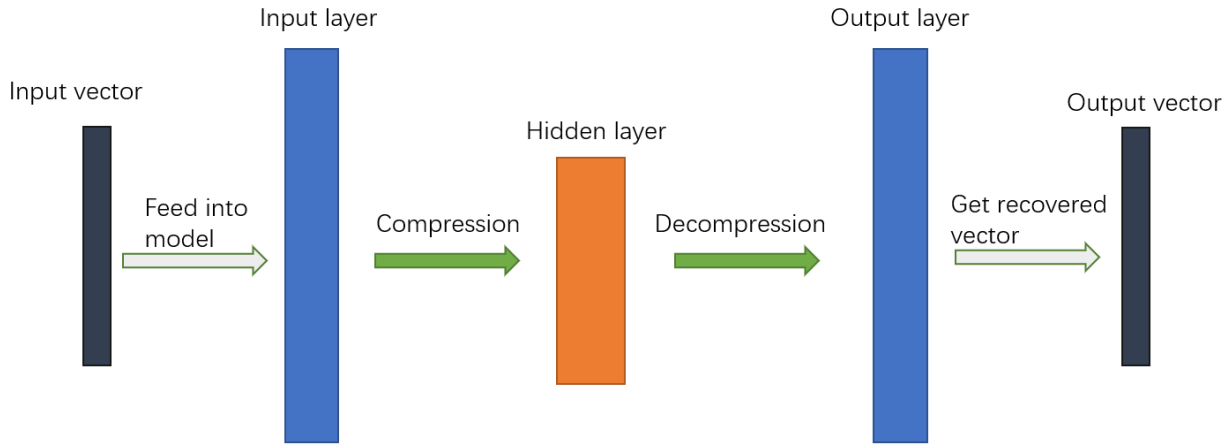

Fig. 2. The workflow of Perceptual Progressive Data Compressor (PPDC).

After training the PPDC model, the original 23-dimensional data will be compressed to lower than 10 dimensions, while keeping the mean squared error between original vector and the decompressed vector in a relative low range. The detail of experiments will be discussed in future parts.

Another innovation of the proposed PPDC model is: the perceptual loss (Johnson, et al., 2016) is introduced during the training procedure. The training of our PPDC model is unsupervised, thus the only principle of our training is to make the output of decompressor to be 'as similar as possible' to the original input vector. Perceptual loss can be used to measure the similarity between them. Specifically, a neural network works as a classifier is trained: given a 23-dimensional vector representing clinical symptoms of a patient, the classifier should predict the disease category of the patient: 0 means normal (no disease), 1 means HighBP, 2 means pneumonia and 3 means SARS. Thus, the dimension of its output is 4, each value represents the probability of the disease category on this index (from 0 to 3). After training this classifier, its output 4-dimensional vector can be viewed as a perceptual representation of a clinical symptoms vector. Then, unlike using average Euclidean distance in the original paper of perceptual loss (Johnson, et al., 2016), mean square error will be used instead.

The implementation of the model is based on the use of PyTorch deep learning framework.

2 Method

2.1 Dataset Details

The dataset used for analysis comes from the study of fuzzy signature aggregation (Mendis, Gedeon, & Koczy, 2005). Each record contains the fuzzy signature of patients' temperature at 4 different times (8 a.m., 12 a.m., 4 p.m., 8 p.m.) with 3 degrees, systolic blood pressure with 3 degrees, diastolic blood pressure with 3 degrees, 3 degrees of nausea, and the 2 degrees for showing the existence abdominal pain. In total, 23 values are showing the clinical symptoms of a recorded patient. All the entries in each data are fuzzy values, which means they are all in the range of 0 to 1. Based on this, due to the lack of prior medical knowledge, no data is preprocessing applied on the data. There is no significant evidence showing that applying normalization or other data preprocessing techniques on the given fuzzy-valued dataset will lead to a definite improvement to data compression. Thus, the data in SARS dataset will be directly processed into the PPDC model.

As for the disease label of each record, it is not a part of compression. The design goal of the PPDC model is only to compress the clinical symptoms. There are 2 reasons for this design, the first reason is: the disease label is a definite integer class value, while the others are decimal fuzzy values. They are not in the same value category. The second reason is: if the disease label is also considered, the accuracy of compressing and recovering this value in good manner weighs much more than any other value. It is more proper to take the disease label as a single definite value and compress the rest of the other 23 fuzzy indefinite values in actual use.

2.2 Network Structure

As shown in Fig. 2., there are 2 parts of the PPDC model: compressor for compression, decompressor for decompression. Both compressor and decompressor are represented by a fully connected layer. At the initialization part,

the input dimension and output dimension for each part are the same as the input vector. After pruning the weight vector of the compressor, the pruned weight matrix will be the weight matrix of the new compressor, and a new decompressor will be initialized for training. Obviously, the output dimension of the compressor, and the input dimension of the decompressor, will decrease as the training goes on.

As for the structure of the classifier, it is a 3-layer neural network including 1 layer for activation. The intermediate hidden layer has dimension half to the input vector, which is 11 in this case. The workflow of the classifier is shown in Fig. 3.

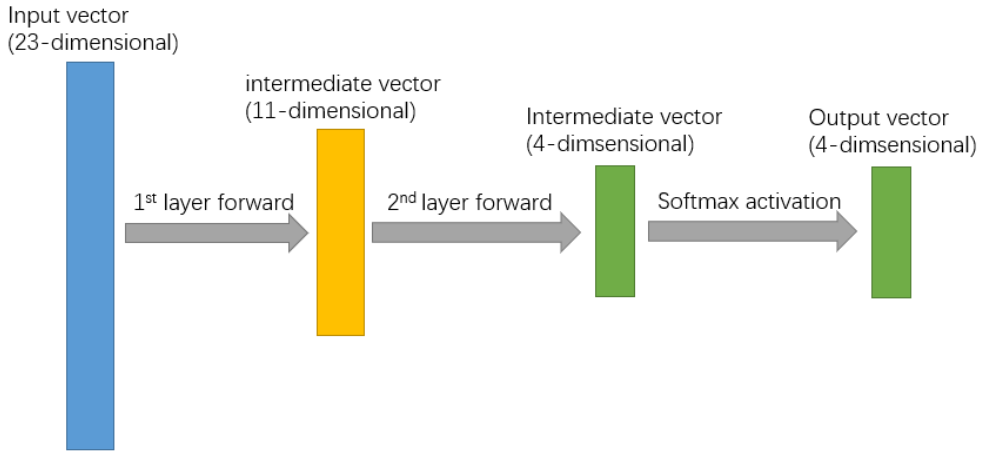

Fig. 3. The workflow of the classifier.

2.3 Training Hyperparameters Setup.

In my implementation, the random seed is set to 46608420. The optimizer used to follow up the optimization is the built-in PyTorch stochastic gradient descent optimizer. The learning rate is set to 0.001, and the rest of other settings in optimizer are default options.

As for training the classifier for perceptual loss, built-in PyTorch cross entropy loss is used as the loss function. 3200 records from dataset will be used for training, and 800 records will be used for validation. The classifier will be trained for 10 epochs, the model with highest accuracy on validation set will be saved to calculate perceptual loss during the training of PPDC.

As for training PPDC, the maximum number of rounds for pruning the weight matrix of compressor is set to 15, to make the compressor fully prune its weight matrix. For each round, the compressor and decompressor will be trained in an end-to-end manner for 20 epochs. There will be training result comparison between training for 10, 15, 20, 25 and 30 epochs in each round that will be analyzed in discussion part. At the first training round, the compressor will have the output dimension set to 23, same as the input 23-dimensional vector. After pruning, the decompressor will be re-initialized to fit the new output dimension of compressor. For instance, if after pruning the new compressor will compress the input 23-dimensional vector into 16-dimensional vector, then the new decompressor will be initialized to take 16-dimensional compressed vector as the input and output the 23-dimensional decompressed vector.

2.4 Training Procedure.

Training procedure of our model follows the backpropagation manner. The loss function is the addition of two parts: one is the mean squared error (MSE) between the decompressed vector and the original input vector of compressor, the functions is:

$$MSE(x,\ y) = \frac{1}{d}\sum_{i=1}^{d}(x_i - y_i)^2$$

Equation 1. mean squared error function.

Where d stands for the dimension of vector to be compressed, which is 23 in our case, x stands for the decompressed vector produced by the decompressor, and y stands for the original input vector x_i stands for the elements in the decompressed vector, and y_i stands for the elements in the original input vector; another part is the perceptual loss, which is the MSE between the outputs of classifier, with input set to original input vector and decompressed vector. Thus, the perceptual loss function is:

$$perceptual\_loss(x, y) = MSE(classifier(x), classifier(y))$$

Equation 2. perceptual loss function.

In conclusion, the loss equation for one sample used for training is:

$$training\ loss = MSE(x, y) + \ perceptual\_loss(x, y)$$

Equation 3. training loss function.

After one round of training, the compressor and decompressor are fine-tuned with this dimension setup. Then, weight vector pruning is carried out. Pruning is based on the distinctiveness between hidden units. For computational efficiency and lack of RAM space, it is sufficient to use the first 200 input vectors from the training set to generate the hidden units for pruning. Hidden units come from the outputs of the compressor. Assume there are N input vectors with d dimension, then the outputs of the compressor can form a matrix with shape Nxh, where h stands for the output dimension of the compressor. Each column of the output matrix represents a hidden unit. Distinctiveness used in the PPDC model is the cosine distance, and this idea comes from. A distance matrix of cosine distance is firstly calculated. For entry on i'th row, j'th column, it represents the cosine distance between i'th hidden unit and j'th hidden unit. The threshold is set to 0.96, which means: (a) if the cosine distance between two hidden units is above 0.96, that means they are too similar to each other, which results in producing similar information. For compressing data, one of the 2 weight vectors producing these 2 hidden units will be added to another vector, then it will be pruned out. (b) if the cosine distance between two hidden units is below -0.96, it means these two vectors are almost opposite, and their hidden units can cancel out. For this situation, 2 weight vectors producing these 2 hidden units will both be canceled out.

Pruning creates the new weight matrix for the compressor, but there is no pruning or any corresponding adjustment for the decompressor to follow up on the change of compressor. Thus, the only possible solution is to re-initialize the decompressor, for the training in the next round. The performance of the best-performed compressor in each round will be saved for future analysis.

2.5 Training Workflow Overview

In conclusion, the training procedure for one sample in the dataset is: Firstly, one patient vector from the dataset will be proceeded to the PPDC model, then the decompressed vector is obtained; then, those two vectors will be the input of the classifier model to obtain the perceptual output of them, used for the calculation of perceptual loss; after that the training loss is obtained, backpropagation can be applied to update the parameters in PPDC model. Here in Fig. 4., the whole workflow is illustrated.

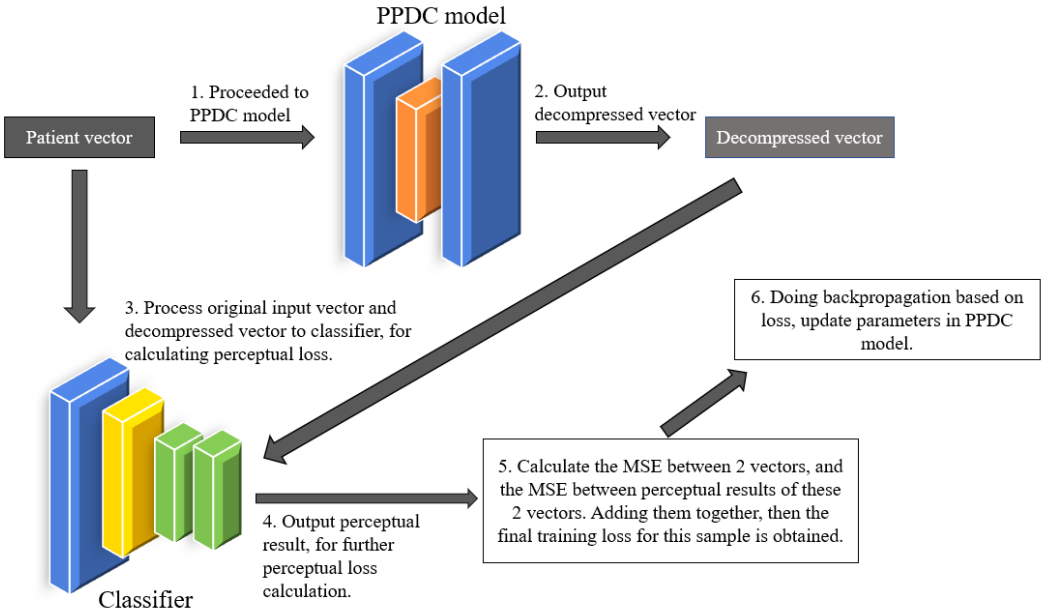

Fig. 4. The training workflow for one sample

3 Results and Discussion

3.1 Quality of Compression

For performance evaluation, both the average mean square error (MSE) and perceptual loss are used for evaluating the performance. We calculate the perceptual loss and average MSE between the original input vector and decompressed vector. For assessment, only the performance of best compressor for each training round is recorded: the one that achieved the lowest training loss over all inputs, among all epochs. I will show the relationship between the

number of rounds, compression dimension, perceptual loss and the average mean squared error over all inputs in each round in Fig. 5. For this test, the number of epochs for each round is set to 20.

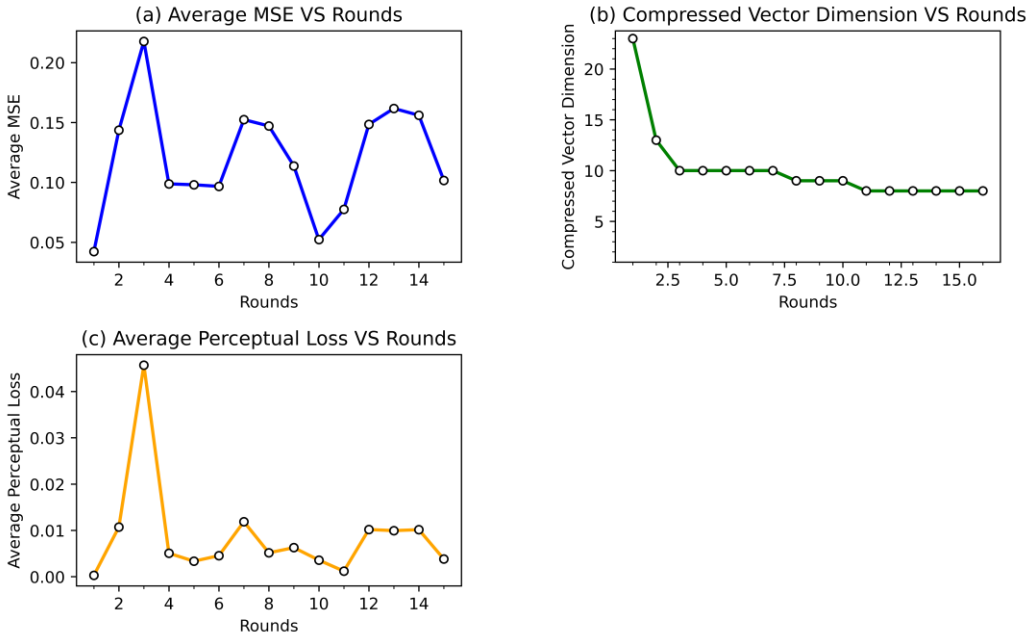

Fig. 5. Graph (a) shows the change of Average MSE of each data w.r.t rounds, (b) shows the change of the dimension of compressed vector w.r.t rounds, and (c) shows the average perceptual loss of each data w.r.t rounds.

According to graph (a), it shows that the change of the lowest achieved average MSE is not consistent, but in the last round, as the dimension of the compressed vector reaches 6, the average MSE is around 0.117, which is not as good as the average MSE 0.042 of the baseline: PPDC model which does not compress the input vector dimension (hidden layer dimension set to be the same as the input vector). As for perceptual loss, the final average perceptual loss is around 0.006, while the baseline achieves around 0.0003. The PPDC model finally achieves

Obviously, the MSE and perceptual loss from the final PPDC model are not as good as non-compressing baseline model, but the main task of PPDC model is to compress the data. That means, if the final loss is reasonable, and the data is fully compressed, then the model performance is acceptable. In our case, average perceptual loss of 0.005 means: for an experienced doctor (the well-trained fine-tuned classifier in our case) with four given disease types, the decompressed data will only cause slight difference on doctor's judgement on the probability that the patient has a certain disease. This evidence strongly supports that the decompressed vector has high similarity to the original input vector. Thus, since the perceptual loss is reasonable, and average MSE is acceptable, se can conclude that our PPDC model can solve the problem of compressing high-dimensional patients' clinical symptoms data and decompress low-dimensional representation to support doctors' decision making.

3.2 Effective of Hyperparameter

Number of epochs for each round is an important hyperparameter for training. It determines the quality of the trained compressor and decompressor in each round, and this can affect the pruning process significantly. For following experiments, the number of maximum rounds will be set to 15, and the number of epochs will be set to 10, 15, 20, 25, and 30 respectively. In following paragraphs, I will talk about the variation in compression dimension, perceptual loss and the average mean squared error as the number of epochs changes.

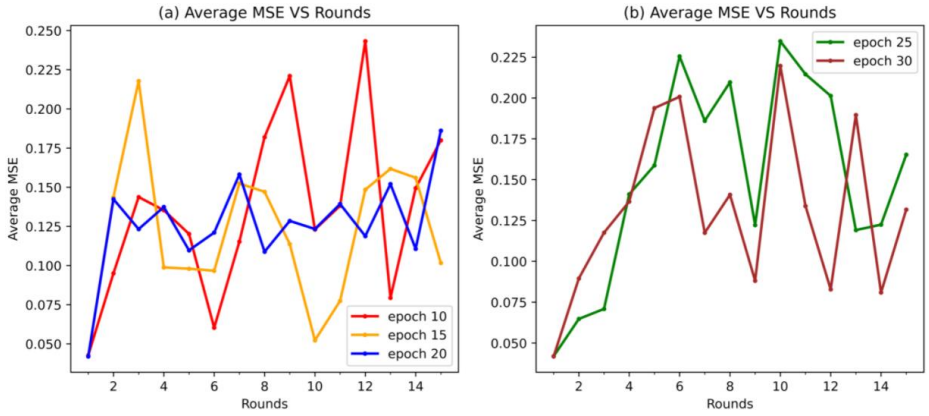

Fig. 6. Graph (a) shows the change of average MSE of each data w.r.t rounds for setting epoch to 10, 15 and 20, while (b) shows the average MSE for setting epoch to 25 and 30.

According to Fig. 6., we can see that the number of epochs does not make consistent contribution on the average MSE. The lowest average MSE 0.101 is achieved when the number of epochs is set to 15. That means when the actual model needs to be trained, the larger epoch number does not guarantee to produce the model with lower average MSE.

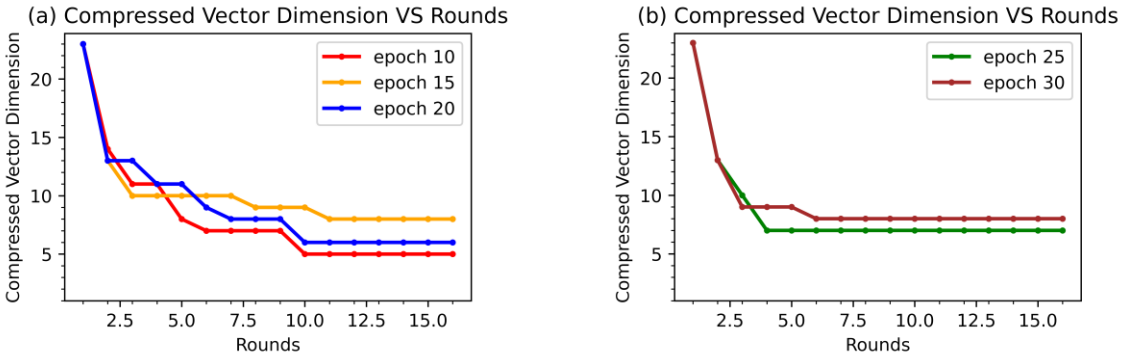

Fig. 7. Graph (a) shows the change of the dimension of compressed vector w.r.t rounds when setting epoch to 10, 15 and 20, while (b) shows the change of the dimension of compressed vector when setting epoch to 25 and 30.

According to Fig. 7, the total dimension reduced by our PPDC model does not increase or decrease linearly with the number of epochs. The highest dimension reduction is achieved when epoch is set to 10. The final compressed vector is 5-dimensional. But, by comparing the graph (a) and (c), we can see that: as for the rounds required to reach reduce the dimension, as the epoch increases, the number of dimensions being reduced in early rounds (before round 4) are much high, which means the convergence speeds up as the number of epoch for each training round increases.

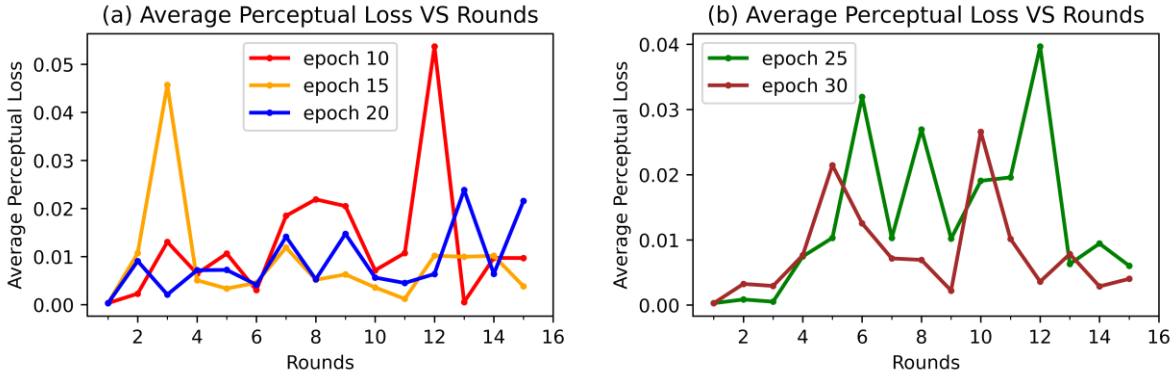

Fig. 8. Graph (a) shows the average perceptual loss of each data w.r.t rounds while setting epoch to 10, 15 and 20, while (b) shows the average perceptual loss of each data while setting epoch to 25 and 30.

As shown in Fig. 8., similar to average MSE, the lowest perceptual loss is achieved when epoch is set to 15, which is 0.0038. The perceptual loss when number of epochs set to 30 is 0.0040, which is slightly higher than lowest result. That means the perceptual loss is not guaranteed to be lower as the number of epoch for each round is set to be higher.

4. Conclusion and Future Work

4.1 Conclusion

In this paper, I proposed a neural network approach for data compression by dimension reduction, and the model is called Progressive Perceptual Data Compressor (PPDC). It consists of 2 parts: one is the compressor part; another one is the decompressor part. Both are represented as fully-connected neural layer. According to the experiment result, our model achieves significant improvements comparing to the baseline method. It preserves most features of the original input, while compressed the input data to at most 21.7% of its original size with specific hyperparameter setup.

The main innovation of the PPDC model is the use of modified version perceptual loss (Johnson, et al., 2016). Perceptual loss is applied in both training procedure and model evaluation procedure. The purpose of introducing perceptual loss is to model the similarity between 2 high-dimensional vectors. Combining both MSE and perceptual loss, the model is capable of compressing data taking both external and latent features into consideration.

4.2 Future Work

The first problem is the unstableness of PPDC, which leads to the difficulty for finding the most appropriate hyperparameter setup to train the model. It is shown in part 3.2, that there is no transparent relationship between the training hyperparameter setup and the performance of fine-tuned model. One possible solution could be the use of genetic algorithms. There are existing works such as using genetic algorithm to choose hyperparameters for convolutional neural networks (Xiao, Yan, Basodi, Ji, & Pan, 2020).

The second problem is to improve the perceptual loss. Currently the perceptual loss is the MSE between the outs of classifier with input as original vector and decompressed vector. MSE may not be the best option for showing the difference between these two outputs. Since the output values are activated with softmax activation function, measurements such as cross entropy and Kullback–Leibler divergence (Kullback & Leibler, 1951) could be applied. Another potential improvement could be changing the classifier to a deeper network and calculate the difference in several layers instead of only calculating the MSE of the outputs. This choice of function representing the 'difference' requires further research.

References

Gedeon, T., & Harris, D. (1992). Progressive Image Compression. Proceedings International Joint Conference on Neural Networks, 4, pp. 403-407.

Gedeon, T., Catalan, J., & Jin, J. (1997, 9 17-19). Image Compression using Shared Weights and Bidirectional Networks. Proceedings 2nd International ICSC Symposium on Soft Computing, pp. 374-381.

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. European Conference on Computer Vision, pp. 694-711.

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. Annals of Mathematical Statistics., pp. 79–86.

Mendis, B. S., Gedeon, T., & Koczy, L. (2005, 12 14). Investigation of Aggregation in Fuzzy Signatures. International Conference on Computational Intelligence, Robotics and Autonomous Systems.

Xiao, X., Yan, M., Basodi, S., Ji, C., & Pan, Y. (2020, 1 23). Efficient Hyperparameter Optimization in Deep Learning Using a Variable Length Genetic Algorithm. arXiv. Retrieved from https://arxiv.org/abs/2006.12703