Performance of Autoencoder with Bi-Directional Long-Short Term Memory Network in Gestures Unit Segmentation

Ziwenhan Song

Australian National University, Canberra ACT, 0200, Australia
ziwenhan.song@anu.edu.au

Abstract. Nowadays, gestures analysis has an increasing need on human computer interaction. And the gestures segmentation is one of the basic procedures to do gestures analysis. As we know that the gestures video is a sequence of frames playing along with time spots. The dataset is about gesture unit segmentation which is preprocessed by its authors in advance. They carried the experiments with MLP (Multi-Perceptron). Originally, they regarded several continuous frames as one mini-batch, and trained them without messing up the order. However, their method did not address the issue that each frame is derived from previous frame which means it should contain the information from previous frame. But this paper applies Bi-Directional Long-Short Term Memory Network (Bi-LSTM) to embed the information of previous frame and successor into current frame and do classification to segment the gestures unit. Also, this paper compares the performance of LSTM with and without autoencoder, which accelerating the speed of training. What is more, the LSTM does improve 5% and 10% performance on the metrics of accuracy and F1-score.

Keywords: Autoencoder, Bi-LSTM, Gestures Unit Segmentation.

1 Introduction

Currently, gestures recognition becomes widely used on human computer interaction. The purpose of recognizing gestures of human hands is to build a quick and convenient approach for interacting with computers. In order to separate different gestures in a frames of videos, we need to analyze each frame to make a decision about what state the gesture belongs to. To address this problem, Wagner et al. (2014) designed a model by using MLP (Multi-Perceptron) to predict the gesture unit based on current state, like velocity. However, this method does not contain too much history and also does not consider the future. Thus, we introduce the LSTM and attempt to make each frame of video or gesture unit contain not only current state but also the information from previous states and future states which should be more accurate than before to recognize gestures. In addition, the autoencoder accelerating the training of classification. This paper compares the training time between model with and without autoencoder, and also compares the model with and without the LSTM. All in all, the autoencoder does make the training faster, in details, for each epoch of training, before encoding the training time is about 1.35 secs, and after encoding, the training time decreased to 1.30 secs. On the other hand, after using LSTM, the test result increased about 5%-10% in both accuracy and F1-score. Furthermore, the original paper of this dataset got the F1-score on about 0.92 which is much higher than this paper, we cannot repeat their result. For further study, the parameter of neural network should be adjusted for overcoming the original paper.

2 Background

2.1 Concepts of Neural Networks

Basic Feedforward Neural Networks

Human got the inspire from our brain which is constructed by massive neurons and neurons connected with each other. Electric signals and chemical signals pass through those neurons. Neural networks are this kind of connected computing system which can receive input patterns and output classification results or regression values (van Gerven & Bohte, 2018). To put it simply and mathematically, neural network creates a mapping function from input space to output space. We can regard a single hidden layer neural network as a universal function approximator which was proved by Cybenko (1989). What is more, Hornik (1991) proposed that the ability of approximation is not from the choice of activation function, but the multilayer neural network itself.

For instance, the simplest neural network is constituted with one input layer, one hidden layer and one output layer. Assume we use $X = \{x_1, x_2, ..., x_n\}$ to represent input, $Y = \{y_1, y_2, ..., y_m\}$ to represent output and $H = \{h_1, h_2, ..., h_k\}$ to represent hidden units. Therefore, the mathematical representation of a neural network is defined by:

$$H = f_1(X^T Weight_1 + Bias_1) \qquad (1)$$

$$Y = f_2(H^T Weight_2 + Bias_2) \qquad (2)$$

For which Weight is a transformation matrix that mapping from one vector space to another vector space, and the function f_1 and f_2 are some kinds of activation function which would make the results nonlinear. Finally, we have an error function to measure the distance between predicted output and the expected output which is labeled manually in training data.

However, how to determine the weights is the procedure called training. Popularly, at first we give small initial values to weights, and use backpropagation to update the weights iteratively. The main idea is to do chain rule on error function and pass the gradients back to the input (Werbos, 1990). Then, for each weight, we own a gradient value, we can update the current weight according to this gradient which may be multiplied by a minor value called learning rate. After several iterations, we may observe the reduction of errors, and stop the training at proper time. In the end, we use this trained model to do prediction on test dataset, which is just another feedforward operation as equation (1) and (2) do, and the only different is that we do not do backward this time.

As the computational power rapidly developing, people found that the neural network can be deeper which performs better results on computer vision and natural language processing. This idea, intuitively, motivated by regarding the output of each layer as a high level representation from previous layer (Bengio, 2009). Therefore, we have the belief and theory of doing this work.

Concepts of Recurrent Neural Network (RNN)

Basic RNN

RNN was inspired by human's thinking that when we human beings are reading an article we could understand the current word based on the previous words that we have read. A feedforward neural network can map a current input to an output, but RNN can map all the previous inputs with current input into current output which means it can obtain the history and embed them into current state for comprehensive understanding (Kawakami, 2008).

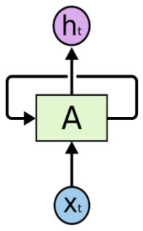

Fig. 1. Recurrent Neural Network (Olah, 2015)

As we can see in **Fig.1**, for each output of current layer, it directly goes back to the input. Thus, the previous layer's information would be wrapped into next layer. We can unfold the RNN like a chain, showing in the **Fig.2**. This chain-like network, therefore, can accept a sequence of data which is very nature for applying it on sequence of words data or temporal related data, such as neural machine translation, temporal classification and so on.

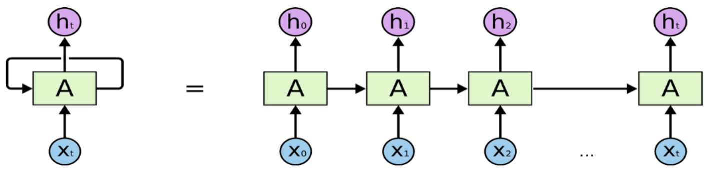

Fig. 2. Unfold Recurrent Neural Network (Olah, 2015)

Long-Short Term Memory Network (LSTM)

Long-short term memory network was firstly designed by Hochreiter and Schmidhuber (1997). As we can see in **Fig.3**, each layer of LSTM is more complicated than RNN. Basically, the LSTM composed by four cells, the input gate, the output gate, the forget gate and the memory cell. Those gates own the abilities of letting information go through or block that simulates the human's forgetting. In addition, the cell in the Bi-directional LSTM is not only memorizing the previous information but also containing the future states' information which provides more context for current state.

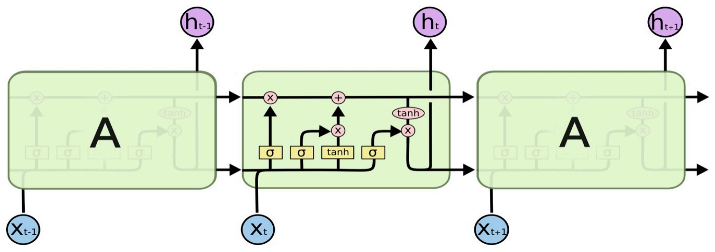

Fig. 3. The Structure of Long-Short Term Memory Network (Olah, 2015)

2.2 Concepts of Autoencoder

Auto-associative, currently, we call it autoencoder. According to Gedeon (1998), auto-associative is a kind of compression technique that can do dimension reduction so that accelerate the training speed. Also, it can extract the important features from free redundant parameters (Gedeon, 1998). The basic idea is to set a symmetrical neural network which owns less dimensions between the input and output layer. We can regard the left half part as encoder, and regard the right half part as decoder. Thus, the procedure is like compressing input into small number of features and then attempting to recover them back. Furthermore, we try to make the output close to input itself which means the middle layer keeps the most effective information.

From the view of Bengio (2009), we can regard the output of each layer as a higher level of features than previous layer. Thus, in the encoder part, the hidden layer is composed by low level features from previous layer and make them more concise. And in the decoder part, we try to decompose the high level features into low level features.

Both ideas are rational and we can find the thought changing when time flies by. However, the essential part is not change.

Three Topology of Autoencoder

The three topology network structures are described in Geneon's work (1998).

Non-shared Weights.

Non-shard weights autoencoder is the most basic neural network that constructed symmetrically with smaller unit size in hidden layers. The first step is to initial small number of weights and do feedforward. The second step is to do back-propagation and update those weights. The weights at symmetrical positions are not shared which means they can be different. Finally, we only use encoder part (the left half of neural network) for transforming original data into hidden layer space.

Shared Weights.

Shared weights autoencoder is a little different with non-shared weights which shared the weights at symmetrical positions. From the view of linear algebra, we just transform a matrix into another space with the weights matrix and then use the its transpose to recover it back. Obviously, we cannot recover it back completely, and it just an approximation.

Bidirectional.

The bidirectional autoencoder is based on the non-shared weights autoencoder. We get the output from the output layer in shared weights model as the input. Then, we utilize the original input as target to train a new model. Intuitively, we recover the original data, and doing another encoding and decoding to make it more concise.

This paper focuses on the non-shared weights autoencoder which is the simplest topology.

2.3 Dataset Description

This dataset is a series gesture features from several streams of video that captured from Microsoft Kinect (Wagner et al., 2014). In details, the data features are constructed according to the velocity and acceleration for left hand, right hand and corresponding wrist. What is more, this is a classification problem that includes five classes which represent gestures, such as Rest, Preparation, Stroke, Hold and Retraction. And the dataset contains seven files which are the processed data from seven videos.

2.4 Review of Gestures Unit Segmentation Recognition

According to Wagner et al. (2014), they designed a single hidden layer neural network, and did not shuffle the data because of chronologically recording videos. The windows size in their paper, we can think it as a mini-batch size which determines a size for a group of data patterns that would be input to neural network together. From my points of view, the authors do not provide any new idea for neural network or just for learning patterns. They present a practice for recognizing gestures from videos based on some handcraft features which are talked in above. And they believe it plays a good performance on this task, which gets the F-score up to 0.9.

3 Model Design and Implementation

3.1 Autoencoder

We design a five-layer autoencoder to train an encoder for compressing the original 32-dimension data into 16 dimensions which could make the training phase faster than before. The autoencoder structure is as below, see in **Fig.4**. We do not make the weights of decoder part equal to the encoder part. In addition, the final autoencoder trains 2000 epochs, the number of epochs is defined arbitrarily because of the very low loss after training.

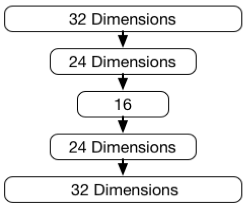

Fig. 4. The Structure of Autoencoder

3.2 Autoencoder with LSTM model

To contain more information about previous and future information, we introduce the Bi-LSTM to wrap history and future of each data unit. And then we do general linear neural network classification. The number of LSTM's hidden units is 16 which is equal to the data dimensions after encoding. And the output of Bi-LSTM owns 32 dimensions for each data unit, twice than the number of hidden units, because of the Bi-directional setting. The overall structure shows in **Fig. 5**. Then the two-layer linear model composes the 5-class classification task.

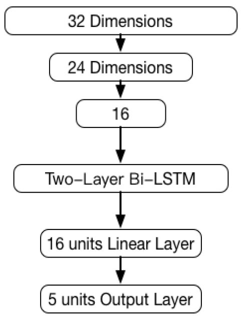

Fig. 5. The Structure of Autoencoder with LSTM Model

4 Experimental Results and Discussion

4.1 Experiments Setting

The experiments are mainly carried on one video of gestures unit segmentation dataset. We separate the video frames into two parts, the first 70% as train set, and the remaining 30% as test set, just the same as Wagner et al. (2014) did. The baseline model has no-LSTM layer and has 18 units in hidden layer. We carry on experiment to measure the efficiency of autoencoder and the improvement of using LSTM.

4.2 Experimental Metrics

Accuracy

Accuracy is a metrics to measure the correct predictions of experiment, the calculation is as follow:

$$Accuracy = \frac{number\ of\ correct\ classification}{number\ of\ video\ frames} \qquad (3)$$

F1-Score

F1-Score is a kind of weighted average of accuracy and recall (Han & Kamber & Pei, 2011), and recall is the rate of true predications over all correct predications for each class. Thus the F1-Score is as follow:

$$F1 - Score = \frac{2}{\frac{1}{accuracy} + \frac{1}{recall}}$$

4.3 The Performance of LSTM with and without Autoencoder

Methods	Time of Training per Epoch (secs)
With Autoencoder	1.30
Without Autoencoder	1.35

Table 1. The Time of Training per Epoch of Model with and without Autoencoder

As you can see in **Table 1.**, the time per epoch in training decreases from 1.35 secs to 1.30 secs, if we assume we need to train 1, 0000 epochs, it will save about 8.3 minutes.

Methods	Accuracy	F1-Score
Without Autoencoder	0.7031	0.7175
With Autoencoder	0.6667	0.6009

Table 2. The Accuracy and F1-Score of Model with and without Autoencoder

However, see in **Table 2.**, the accuracy and F1-score of model with autoencoder decrease about 4% and 9%, which means the information loses a little due to that the encoding procedure does not keep all of the information from original data.

4.4 The Accuracy and F1-Score of Gestures Unit Segmentation with and without LSTM

Methods	Accuracy	F1-Score
Without LSTM	0.6602	0.6143
With LSTM	0.7031	0.7175

Table 3. The Accuracy and F1-Score of Model with and without LSTM

According to the results showing in **Table 3.**, the performance of model based on LSTM is improved 4% in accuracy and nearly 10% in F1-score, which means that the recall for each class rose as well. Therefore, we could know that the LSTM does keep more information of history and future for each state or frame and make the prediction better.

However, the original paper from Wagner et al. (2014) got the F1-score up to about 0.9, they use about 20 units in hidden layer and train up to 1300 epochs. I have set the similar settings, but have not got the similar performance yet.

5 Conclusion and Future Work

This paper introduces bi-directional long-short term memory network to wrap the history and future information into current frame. By comparing with the baseline, this paper with LSTM achieves 5% and 10% improvement on accuracy and F1-score. On the other hand, the autoencoder accelerates the training process and reduces the complexity of neural network. Unfortunately, this paper does not perform as good as the original paper did which only used multi-perceptron method.

For future work, we should carry on more experiments for figuring out better parameters for the model to make the result as close as the original paper.

References

1. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4), 303-314.
2. Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550-1560.
3. Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2), 251-257.
4. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
5. Gedeon, T. D. (1998, October). Stochastic bidirectional training. In Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on (Vol. 2, pp. 1968-1971). IEEE.
6. Kawakami, K. (2008). Supervised Sequence Labelling with Recurrent Neural Networks (Doctoral dissertation, Ph. D. thesis, Technical University of Munich).
7. Bengio, Y. (2009). Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2(1), 1-127.
8. Wagner, P. K., Peres, S. M., Madeo, R. C. B., de Moraes Lima, C. A., & de Almeida Freitas, F. (2014, May). Gesture Unit Segmentation Using Spatial-Temporal Information and Machine Learning. In FLAIRS Conference (Vol. 98).
9. Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.
10. Olah, C. (2015). Understanding LSTM Networks—colah's blog. [online] Colah.github.io. Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs [Accessed 30 May 2018].
11. van Gerven, M., & Bohte, S. (2018). Editorial: Artificial Neural Networks as Models of Neural Information Processing. Artificial Neural Networks as Models of Neural Information Processing, 5.