Video Action Classification

Yuting Tseng (A0212195L)
Department of Computer Science
National University of Singapore
e0503474@u.nus.edu

Zhe Chen (A0091882W)
Department of Information Systems
National University of Singapore
e0293911@u.nus.edu

Bihui Han (A0191517A)
Department of Computer Science
National University of Singapore
e0338161@u.nus.edu

Xinyan He (A0191564Y)
Department of Computer Science
National University of Singapore
e0338208@u.nus.edu

Abstract

Our task is to perform a video action classification on the `Breakfast Actions dataset`, which is composed of numerous videos. Each video is composed of multiple sub-actions. The goal is to classify each segment to any of the 47 sub-actions (excluding SIL) through various methods.

1 Introduction

Our goal is to experiment with different methods of segment representations and to preform action classification over those segments. We have gotten the extracted feature vectors with dimension $(400,)$ for each frame in the video. Therefore, we assumed that there is no need to go through a convolution neural network; furthermore, we focus more on deep neural network (DNN) and recurrent neural network (RNN).

2 Methodology

2.1 Deep Neural Network (DNN)

Deep neural network (DNN), so-called multi-layer perceptron (MLP), is the underlying model architecture for machine learning problems. Due to the memory of the GPU, our model architecture contains four hidden layers. The detail would be shown in the third section.

2.2 Recurrent Neural Network (RNN)

The appearing frames in a video segment are related to time series. We assume that recurrent neural networks (RNN) might be feasible to deal with these sequential problems. We can consider not only the results of the above layer but also the information we get more previously via RNN. The followings are two types of RNN we adopt in this project.

2.2.1 Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) includes three individual gates, input gate, forget gate, and output gate. The forget gate determines whether the previous message would be pass to the next nodes. Furthermore, we take a try of bidirectional LSTM to learn more information from both sides to improve accuracy.

Submitted to CS5242 Neural Network and Deep Learning, National University of Singapore.

2.2.2 Gated Recurrent Units (GRU)

GRU is simplified from LSTM, which has only two gates, update gate and output gate. Comparing to LSTM, a fewer parameter is needed, so we might be able to reduce overfitting. In addition, we train the model both in one direction and bidirectional.

3 Experiments

3.1 Data Preprocessing

Different videos are composed of various number of frames; this situation is not acceptable in a neural network. In order to generate the data with the same length, there are several possible methods of data augmentation 1. pad "0" at the end till the maximum of length or 2. crop the redundant frames after the minimum length. We decide to combine those two ways. First, we set the threshold to be the average length of all segments. Next, we pad 0 at last if it is shorter than the threshold; otherwise, we crop the segment.

3.2 Basic Setting

For criterion, we choose to use `CrossEntropyLoss` since this is a classification problem, which is more suitable for the mentioned error compute methods.

For optimization, we select `Adam` algorithm. This algorithm combines the advantages of `Adagrad`, `RMSprop`, and `Momentum`. The parameters we use are the recommended ones in the original paper.

For epoch, we first observe when the accuracy of validation converges and choose the most reasonable number. For the sake of comparison between different methods, we consider it to be 100.

For the input feature vector, we preprocess the data, as mentioned before. For the output vector, we regularize with `softmax` and select the specific action with the largest possibility.

For training process, we randomly choose 20% of data to be validation set. We compute the loss and accuracy score on both training set and validation set. For testing process, we get the results csv file through our done-trained model; moreover, we ensemble the results of different model.

3.3 Model Structure

3.3.1 Deep Neural Network (DNN)

Table 1: DNN

Layer (Type)	Output Shape	Param #
Linear-1	[-1, 1024]	163,021,824
ReLU-2	[-1, 1024]	0
BatchNorm1d-3	[-1, 1024]	2048
Dropout-4	[-1, 1024]	0
Linear-5	[-1, 256]	262,400
ReLU-6	[-1, 256]	0
BatchNorm1d-7	[-1, 256]	512
Dropout-8	[-1, 256]	0
Linear-9	[-1, 64]	16,448
ReLU-10	[-1, 64]	0
BatchNorm1d-11	[-1, 64]	128
Dropout-12	[-1, 64]	0
Linear-13	[-1, 48]	3,120
Softmax-14	[-1, 48]	0

Total params: 163,306,480
Trainable params: 163,306,480
Non-trainable params: 0

3.3.2 Recurrent Neural Network (RNN)

Table 2: LSTM

Layer (Type)	Output Shape	Param #
LSTM-1	[-1, 398, 256]	673,792
LSTM-2	[-1, 398, 64]	82,432
Linear-3	[-1, 4096]	104,337,408
ReLU-4	[-1, 4096]	0
Dropout-5	[-1, 4096]	0
Linear-6	[-1, 1024]	4,195,328
ReLU-7	[-1, 1024]	0
Dropout-8	[-1, 1024]	0
Linear-9	[-1, 256]	262,400
ReLU-10	[-1, 256]	0
Dropout-11	[-1, 256]	0
Linear-12	[-1, 48]	12,336
Softmax-13	[-1, 48]	0

Total params: 109,563,696
Trainable params: 109,563,696
Non-trainable params: 0

Table 3: GRU

Layer (Type)	Output Shape	Param #
GRU-1	[-1, 398, 256]	505,344
GRU-2	[-1, 398, 64]	61,824
Linear-3	[-1, 4096]	104,337,408
ReLU-4	[-1, 4096]	0
Dropout-5	[-1, 4096]	0
Linear-6	[-1, 1024]	4,195,328
ReLU-7	[-1, 1024]	0
Dropout-8	[-1, 1024]	0
Linear-9	[-1, 256]	262,400
ReLU-10	[-1, 256]	0
Dropout-11	[-1, 256]	0
Linear-12	[-1, 48]	12,336
Softmax-13	[-1, 48]	0

Total params: 109,374,640
Trainable params: 109,374,640
Non-trainable params: 0

3.3.3 Bidirectional Recurrent Neural Network (BiRNN)

Table 4: Bidirectional LSTM

Layer (Type)	Output Shape	Param #
LSTM-1	[-1, 398, 128]	238,592
Linear-2	[-1, 4096]	208,670,720
ReLU-3	[-1, 4096]	0
Dropout-4	[-1, 4096]	0
Linear-5	[-1, 1024]	4,195,328
ReLU-6	[-1, 1024]	0
Dropout-7	[-1, 1024]	0
Linear-8	[-1, 256]	262,400
ReLU-9	[-1, 256]	0
Dropout-10	[-1, 256]	0
Linear-11	[-1, 48]	12,336
Softmax-12	[-1, 48]	0

Total params: 213,379,376
Trainable params: 213,379,376
Non-trainable params: 0

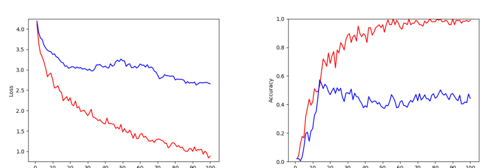
Table 5: Bidirectional GRU

Layer (Type)	Output Shape	Param #
GRU-1	[-1, 398, 128]	178,944
Linear-2	[-1, 4096]	208,670,720
ReLU-3	[-1, 4096]	0
Dropout-4	[-1, 4096]	0
Linear-5	[-1, 1024]	4,195,328
ReLU-6	[-1, 1024]	0
Dropout-7	[-1, 1024]	0
Linear-8	[-1, 256]	262,400
ReLU-9	[-1, 256]	0
Dropout-10	[-1, 256]	0
Linear-11	[-1, 48]	12,336
Softmax-12	[-1, 48]	0

Total params: 213,319,728
Trainable params: 213,319,728
Non-trainable params: 0

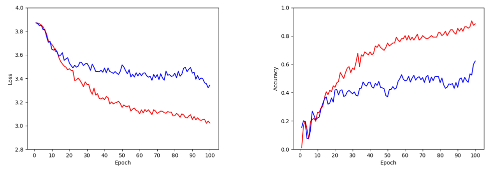
3.4 Results

3.4.1 Deep Neural Network (DNN)

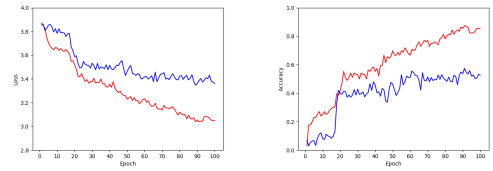


Training Results of DNN
Public Score: 0.4

3.4.2 Recurrent Neural Network (RNN)

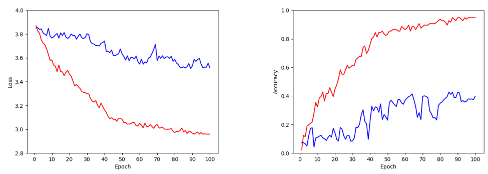


Training Results of LSTM
Public Score: 0.41199

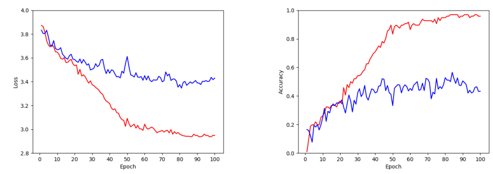


Training Results of GRU
Public Score: 0.38862

3.4.3 Bidirectional Recurrent Neural Network (BiRNN)



Training Results of bidirectional LSTM
Public Score: 0.40186



Training Results of bidirectional GRU
Public Score: 0.38064

The figures on the left and right are *Epoch v.s. Loss* and *Epoch v.s. Accuracy*, respectively. The blue lines indicate the result values of *training data*; the red lines show those of *validation data*.

4 Analysis

Table 6: Comparison within Each Model *(Epoch=100)*

	DNN	LSTM	GRU	biLSTM	biGRU
Layer #	14	13	13	12	12
Param #	$\approx 163 * 10^6$	$\approx 109 * 10^6$	$\approx 109 * 10^6$	$\approx 213 * 10^6$	$\approx 213 * 10^6$
Loss (Validation)	2.6593	3.3246	3.4055	3.5041	3.3886
Accuracy (Validation)	0.43069	0.5898	0.5655	0.4099	0.4461
Public Score	0.42211	0.41199	0.38862	0.40186	0.38064

4.1 DNN v.s. RNN

The validation's accuracy score of DNN is much worse; however, the public score is better within two types of neural networks. The other worth mentioning point: the validation scores of both LSTM and GRU models are better, compared to DNN model. The possible cause might be that RNN model is not complicated enough for dealing with various datasets.

4.2 RNN v.s. biRNN

The number of parameters of biRNN is about two times more than that of RNN. It indicates that biRNN has higher capacity and complexity, which means: it tends to be overfitting more easily within the same epochs. As a result, we try to select reasonable epochs for each model and indeed get the better accuracy.

4.3 LSTM v.s. GRU

LSTM model is extremely similar to GRU model, including the basic concepts and the structure; the fact, therefore, results in similar validation accuracy and public score. We can still note that the performance of GRU is not as good as LSTM. After thinking deeply, it is the known theory - LSTM outperforms GRU in tasks with long-series sequential data, might account for this situation.

5 Conclusion

We have done some trials through different methods, including DNN, LSTM, GRU, biLSTM, and biGRU. We get the best public score via DNN model and LSTM for the second highest. There are some potential issues over the performance: 1. the model might be underfitting with a small number of parameters; 2. the model might be overfitting with a large number of parameters; 3. the model might perform better if suitable for coping long-series sequential data.

Due to memory issues, we are not able to make use of known pre-trained models, such as ResNet, VGG, and InceptionNet, which are perhaps helpful for the performance. Besides, we split the data into two parts for training, which may lead to the validation data are not select randomly enough.

6 Appendix

6.1 Contribution

Coding - Yuting Tseng

Debugging and executing: Zhe Chen, Bihui Han, Xinyan, He

Report: All teammates

6.2 File

`README.md` - includes the environment, language and some detailed information.

`Train.py` - the main program for training; calculates loss as well as accuracy score on both training and validation datasets, and eventually saves the model with the best validation accuracy.

`Test.py` - the main program for testing; gets the result through the model saved by `Train.py` and eventually creates csv file for submission.

`Parsing.py` - parses the arguments for executing the code (have default values).

`Preprocessing.py` - calculates the length of each segment in order to help us select a reasonable threshold for padding or cropping.

`Loading.py` - converts the numpy data into the wanted format.

`Model.py` - includes the model structures of deep neural network and recurrent neural network.

7 Reference

[1] H. Kuehne, A. B. Arslan & T. Serre. (2014) *The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities.*