Final Project Report: Audio Event Recognition

Xinhao Chen / chenxinhao@sjtu.edu.cn
Shanghai Jiao Tong University / ACM Hornored Class

January 10, 2018

Abstract

In this report, I use various neural networks architectures to classify the features of a Audioset. And I compare three different deep learning frameworks, PyTorch, MxNet, TensorFlow, in a simple network. Besides, I tried a multi-LSTM network with skip connection to attempt to capture sequence information and improve the performance. At last, I use ensemble method to get my final result, about 0.944mAUC, 0.231mAP.

In order to facilitate testing, I wrote a test script(run.py) that can reproduces all my single model results.

1 Introduction

Audio event recognition, the human-like ability to identify and relate sounds from audio, is a nascent problem in machine perception. One of the comparable problems is object detection in image processing.

I use PyTorch as my main frameworks, and tried various network, after all, I find out that the simple two layer fully connection layer can reach a great result, even better than my multi-LSTM network, so I implemented this network in other two frameworks MxNet and TensorFlow, and compared the speed and performance of them.

2 Dataset

2.1 Data Format

Audioset [1] is used in this project. And we don't use the origin data, but the features provided by Google.

- Each utterance is about 10 seconds (can smaller than 10), each second is a 128 dimensional vectors.
- Each utterance may have several labels and there are 527 kinds of labels in total.

2.2 Data Processing

- After looked though the data, I found out that just small part of data is less than 10 seconds, in order to facilitate batch training, I padding(0 padding) all the data in to same length(10).
- I also noticed that all the data is a integer between $[0, 255]$, so I divide all the data by 255, map them to $[0, 1]$, which is much suitable for neural networks.
- In order to fine tune our network, I cut the train dataset into small train dataset(80%) and test dataset(20%), and all of the following experiments are based on these two data sets, which never use the eval dataset to determine any parameters of our network.
- After all, I use determined parameters and the whole train dataset(100%) to train our networks, then, use the eval dataset to eval our well trained model.

3 Evaluation Metric

- AP, also known as average precision. It computes the area under precision-recall curve. In the evaluation section, record the set segments classified for each category as A_i and denote the set of segments in each category as B_i. Precision p_i is defined as $p_i = \frac{|A_i \cap B_i|}{|A_i|}$ and recall is defined as $r_i = \frac{|A_i \cap B_i|}{|B_i|}$. Compute the area of precision-recall curve for recall from 0 to 1 gets the average precision for each category.
- AUC, computes the area under the ROC curve, which is a TPR-FPR curve. Denote the set of all segments as C. TPR(true positive rate) is defined as $\frac{|A_i \cap B_i|}{|B_i|}$ and FPR is defined as $\frac{|A_i \cap B_i|}{|C - B_i|}$. Compute the area of ROC for FPR from 0 to 1 gets the AUC score for each category.

4 Experiment and Analysis

4.1 Environment

All the experiments are conducted on a PC device (Intel(R) Core(TM) i7-6900K 3.2GHz, 16GB memory, NVIDIA GeForce(R) GTX 1080) and are implemented in Python 3.6.2)

4.2 Training Loss

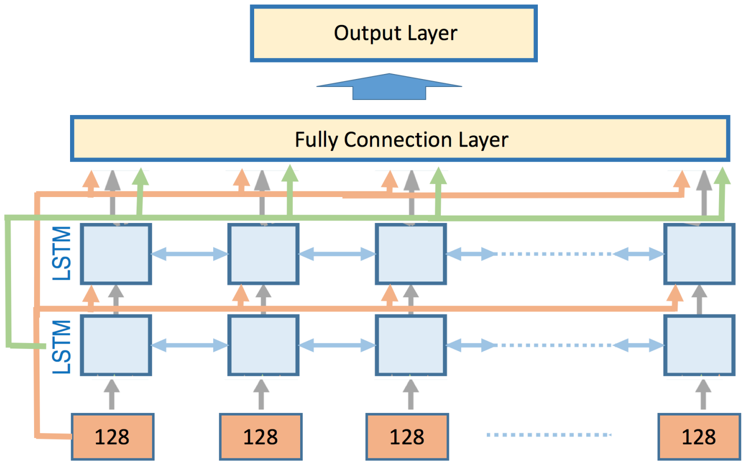
We define the training loss (to be minimized) as a multi-label one-versus-all loss based on max-entropy, between predict label x and input label y.

$$loss(x, y) = -\sum_i (y_i \times log(\frac{1}{1 + e^{-x_i}}) + (1 - y_i) \times log(\frac{e^{-x_i}}{1 + e^{-x_i}})$$

4.3 LSTM with Skip Connection

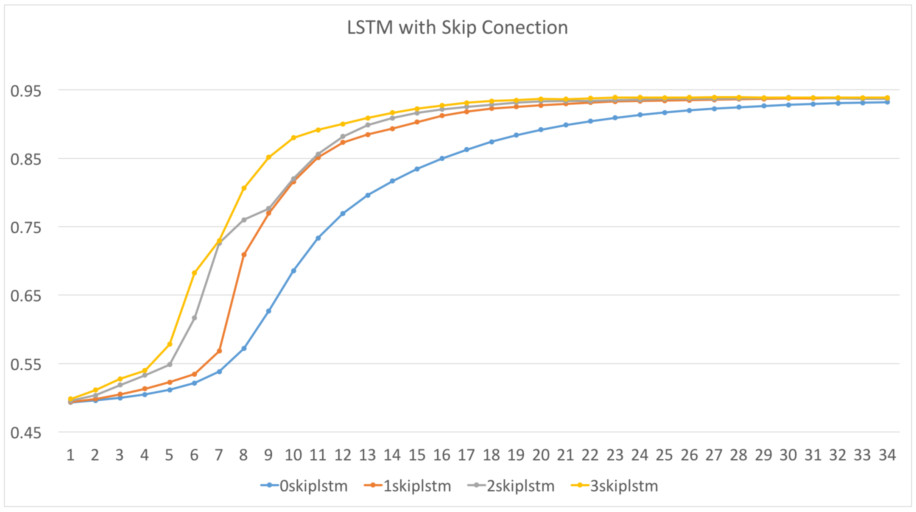
Inspired by the U-net [3], I proposed a new network which use the skip connection into LSTM [2], the idea is very simply, just connect the input and output of each layer LSTM as a new output.

So, this is my first model, first two layer are the bi-direction LSTM with skip connection layers, the last are layer is fully connection layers.



Our results shows that the LSTM with skip connection is effective. As the number of skip LSTM layers increases, the convergence rate of our model has been significantly improved, and achieved better results. This shows that our network has indeed successfully captured the sequence information of the data.

Besides, I also tried to improve this model, like single direction LSTM, masked LSTM, different initialization, but the result do not have a obvious improvement.

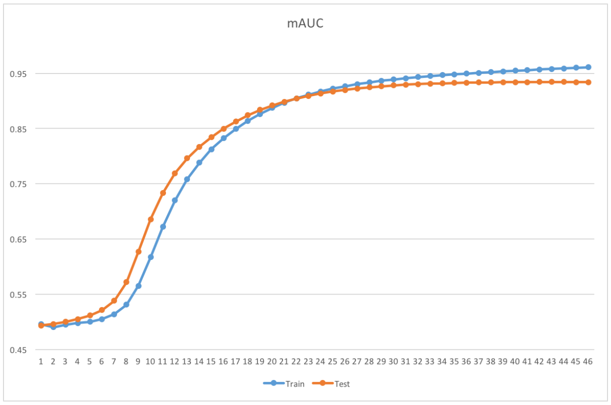


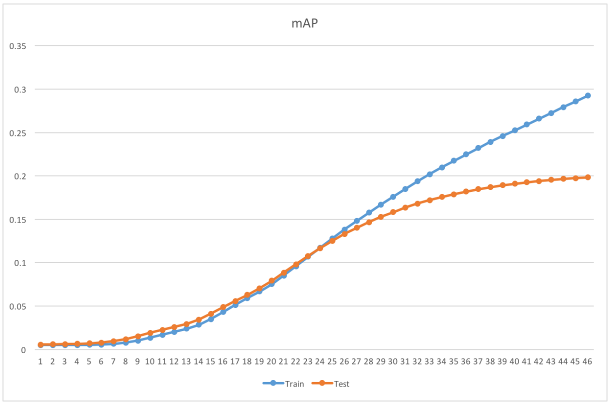
4.4 Comparisons

In order to compare the effect of different parameters on the model, I chose a simple two layer fully connection network as my model, which the result did not change significantly.

In the following experiments, I use the 2 fully connection layers(1280 * 500 * 527), trained by batch size of 500, 50 epoch, with xavier initialization, implemented in PyTorch as our base model.

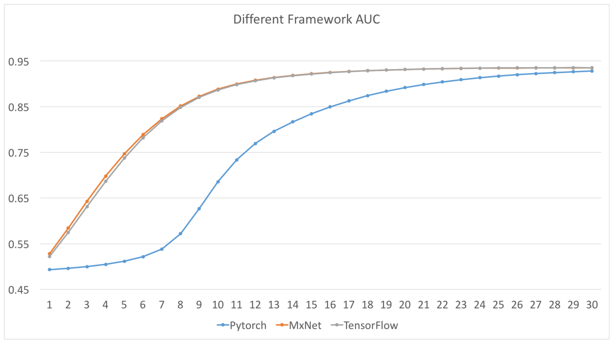
Below is the train curve and test curve of my base model.

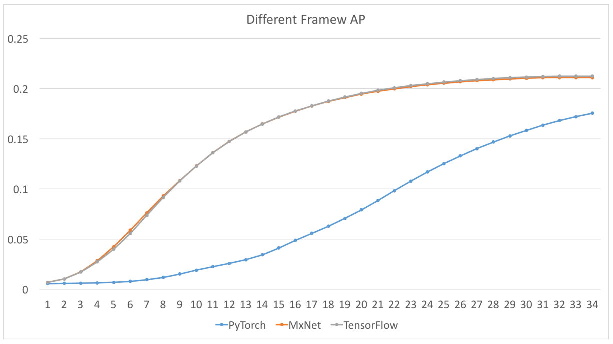




And, base on this model, I tried different frameworks, different batch size, and different initialization.

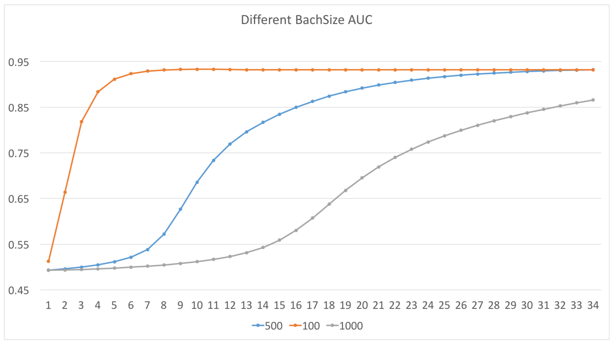
Different Framework

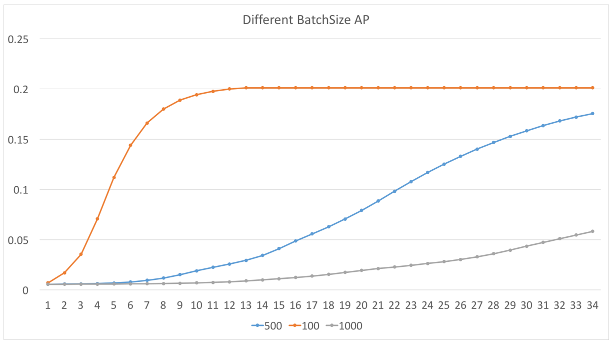




In our network, The speed of convergence of TensorFlow and MxNet are similar, but PyTorch is much slower. But if take into account the actual running time, MxNet is the slowest, PyTorch second, and TensorFlow is the fastest.

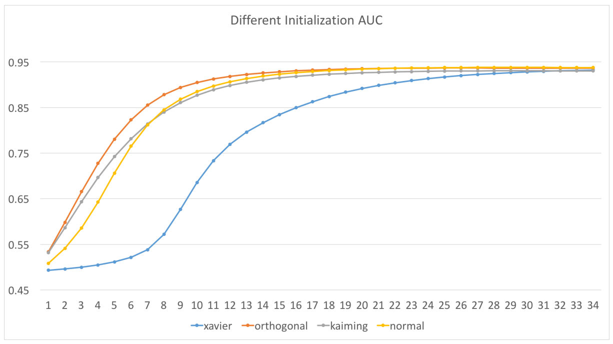
Different Batch Size

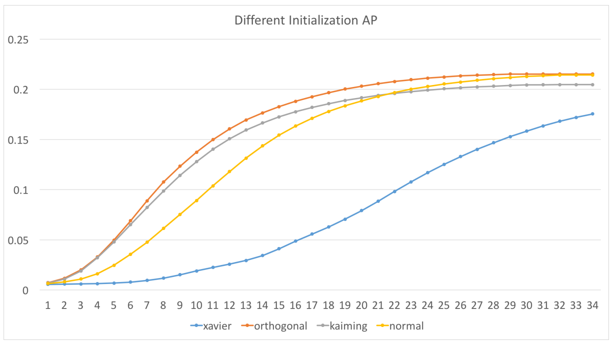




Smaller batch size can convergence in fewer epoch, but for the same number of epochs, it needs more time.

Different Initialization





Except xavier initialization, other initializations have similar convergence rates, but the result of xavier is a little better than others.

5 Conclusion

- Skip connection is a genius idea, which can really help with our training, no only in CNN, it is not only effective in CNN, but also in RNN.
- Initialization is important, and without a properly initialization, network will collapses easily.
- A framework, while providing the basic operation, should also have some convenient high-level interface, just like TFlearn and Gloun.

References

[1] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[3] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-net: A user-level network interface for parallel and distributed computing. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, SOSP '95, pages 40–53, New York, NY, USA, 1995. ACM.