An Comparative Study on Audio Event Recognition

Jiacheng Yang
Shanghai Jiao Tong University
kipsora@gmail.com

Abstract

As a human-level behaviour, audio event recognition is more and more popular and vital for artificial intelligence nowadays. Recently, Google has released AudioSet[1], an equivalent of ImageNet[2] in audio field. In this paper, we will have a comparative study on several basic network artitectures including DNN (including residual links[3]), CNN[4] and RNN[5, 6] on AudioSet using three the most famous deep learning frameworks, Tensorflow[7], MXNet[8] and PyTorch. Additionally, we will also use ensemble methods to enhance our models.

1 Introduction

As deep learning methods are increasingly powerful, more and more problems can be solved by deep neural networks. Nevertheless, due to lacking of sufficient data, audios, as another dimension of real world inputs for humans, can still be a tremendous challenge for deep learning. For example, audio event recognition models always requires large amount of supervised training labels.

Recent works has shown that several methods in computer vision, for example, convolutional neural networks, can be successfully applied to audio classification problems[9]. However, besides computer vision whether basic methods/network architectures in other artificial intelligence subfields such like recurrent neural networks (or LSTM) frequently used in natural language processing will work on audio event recognition problem is still obscure.

Recently, Google has released AudioSet[1], which includes 2 million audio data with high-quality human labels. This dataset can be seen as the counterpart of ImageNet in audio event recognition field. In this paper, we will have several comparative experiments on AudioSet for basic network architectures and machine learning methods.

2 Methodology

2.1 Formulation

Audio event recognition can be seen as a multi-label classification problem where each label stands for a tag like "Boom" and "Swing Music". Each data entry has an input feature extracted from the bottom layer of ResNet[1, 3] and a human-labeled ground truth. Formally speaking, during training we want to train a classifier f_θ under supervised dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$ by minimize a objective loss function $\mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})$, where θ is the parameters of the classifier; $\mathbf{x}$ stands for the input feature vector and $\mathbf{y}$ is the target list consists of several tags. In testing/predicting phase, given an input $\mathbf{x}$, this classifier will produce an tag lists (or probabilities for every tags) $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$.

2.2 Objective Loss Function

We used Sigmoid Binary Cross Entropy(SBCE for abbreviation) as our objective loss function, which regards a multi-label classification problem as a set of binary classification problems. Other methods described in [10] are remained as future works.

Given a classifier f_θ), SBCE on data batch $(\mathbf{x}_i, \mathbf{y}_i)$ is defined as:

$$\mathcal{L}_{\text{SBCE}}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} \mathbf{y}_{ij} \log \hat{\mathbf{y}}_{ij} + (1 - \mathbf{y}_{ij}) \log (1 - \hat{\mathbf{y}}_{ij})$$

where $\mathbf{y}_{ij}$ means the value of the j-th class of one-hot vector; $\hat{\mathbf{y}}_{ij}$ means the counterpart for predict vector; n is the batch size and C is the number of classes.

2.3 Network Architectures

We compared three network architectures, DNN, CNN and RNN on AudioSet using three mainstream deep learing frameworks, Tensorflow, MXNet and PyTorch.

Deep Neural Networks

We will use single layer to 8-layers neural networks with ReLU as their activation function. Batchnorm layer[11] and Dropout layer[12] as well as Residual Links[1] are also seen as variables. The network architecture is illustrated in figure 1 where the dashed lines means it is an experimental variable and star symbol means that the architecture inside the bracket will repeats for 0 or more times.

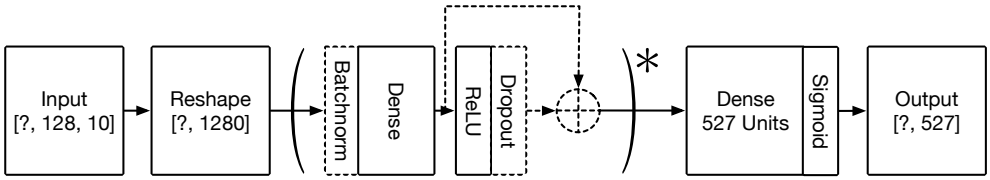


Figure 1: The Architecture of Deep Neural Networks

Convolutional Neural Networks

According to [9], CNN can be also performs well on audio classification. However, in our settings, considering the input feature is extracted from the bottom layer of ResNet, it is hard to say whether CNN will outperform DNN under this circumstance without experiments. We tried the network structure illustrated in the figure 2.

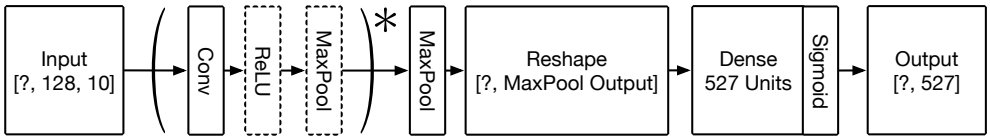


Figure 2: The Architecture of Convolutional Neural Networks

Recurrent Neural Networks

As another powerful tool dealing with temporal data like text, RNN is also selected for a model to be compared with. We used LSTM as well as GRU as our RNN cell. The figure 3 illustrates the basic RNN structure we used.

[1] We may add residual connections skipping more than one layers to improve the performance.

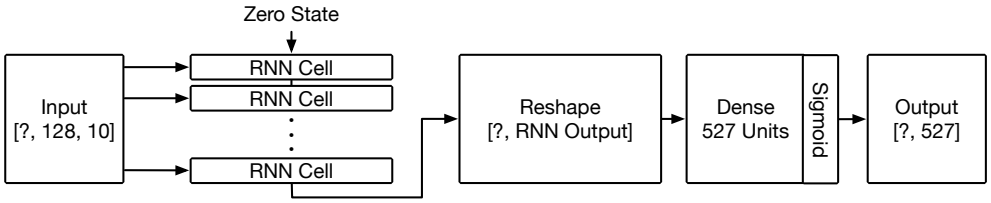


Figure 3: The Architecture of Recurrent Neural Networks

2.4 Metrics

We use two metrics to measure models' power:

1. **AP(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.
2. **AUC(Area Under ROC Curve):** 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.

3 Experiments and Results

3.1 Deep Neural Networks

Training Settings

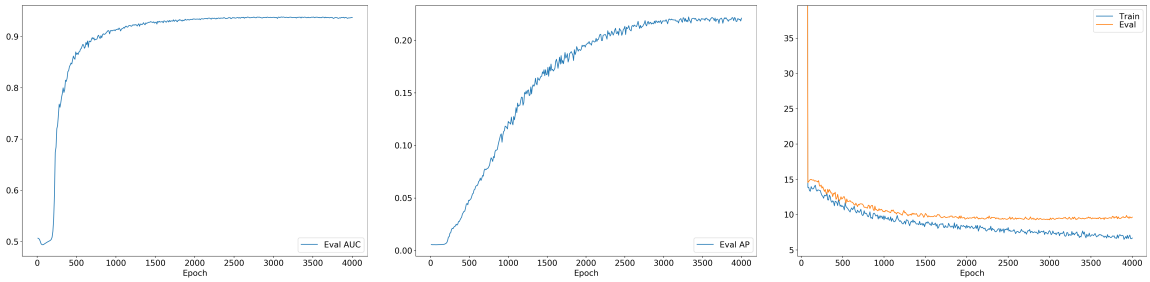
We used RMSProp as our optimizer and the learning rate is 5×10^{-5} and L2 loss as regularizer, the strength of which is 1×10^{-2} for every experiments. Initializer is truncated normal initializer, the stddev of which is about 1×10^{-3}(1×10^{-2} for deep networks with residual connections). In addition, we separate 30% of the training data as validation set to select the best hyperparameters (e.g. training epochs, dropout rate, batch size and hidden units in each layer). The final model will be trained with all of the training data.

Results

The results on evalutaion set of different networks are listed in the table 1 and the performance curves are shown in figure 4. As we can see in the table, the single layer NN performs the best. And the deep neural networks with residual links are not much worse than single layer neural networks. Besides, the baseline classifier logisitic regression also performs not bad on this dataset. These results prove that the dataset is relatively simple and a very deep network is liable to be overfitting here.

Layers	None	BN	Dropout	BN-Dropout	Residual
0	90.5/0.174	90.4/0.169	N/A	N/A	N/A
1	94.4/0.236	94.5/0.241	94.5/0.242	94.4/0.240	93.5/0.207
2	93.7/0.213	93.8/0.216	93.5/0.203	93.7/0.212	93.8/0.209
4	93.4/0.193	93.5/0.195	92.9/0.173	93.2/0.185	93.5/0.226
7^2	N/A	N/A	N/A	N/A	93.7/0.216

Table 1: Experiments on Deep Neural Networks



(a) AUC (b) AP (c) Loss

Figure 4: Figures on Deep Neural Networks

3.2 Convolutional Neural Networks

Training Settings

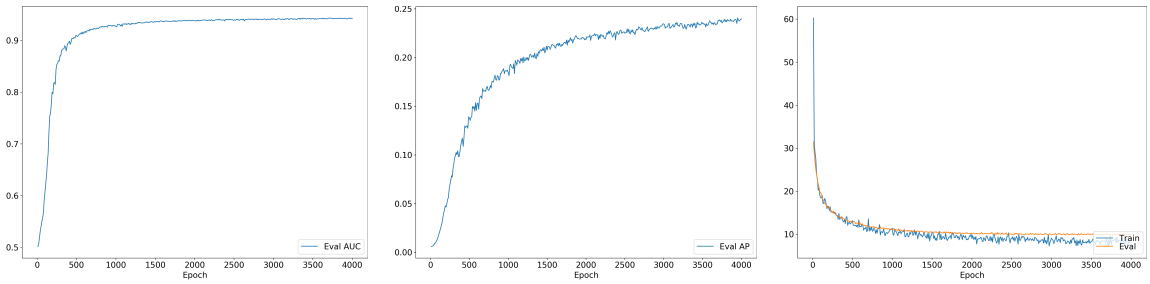
We used three type of filters, the number of each of which is 128 and the kernel size are swept over $[2, 3, 5]$ with stride 1 followed by a maxpooling layer (if used) whose pool size is $[5, 3, 2]$ respectively. Other training settings is the same as DNN.

Results

The performance of CNN is listed in the table 2 and the curves are shown in figure 5. Most of AUCs are slightly below the baseline line methods single layer DNN. But the APs are almost the same.

	None	(Conv-ReLU)*-Maxpool	(Conv-Maxpool-ReLU)*
Single Layer DNN	94.5/0.242	N/A	N/A
2 Conv Layers	N/A	94.2/0.242	94.2/0.244
3 Conv Layers	N/A	94.2/0.235	94.1/0.231

Table 2: Experiments on Convolutional Neural Networks



(a) AUC (b) AP (c) Loss

Figure 5: Figures on Convolutional Networks

3.3 Recurrent Neural Networks

Training Settings

We also tried another basic neural network architecture, recurrent neural networks, on AudioSet. Multi-layers LSTM [5] is used as RNN cell in this experiment. The hyperparameters' settings are the same as the experiment 3.1.

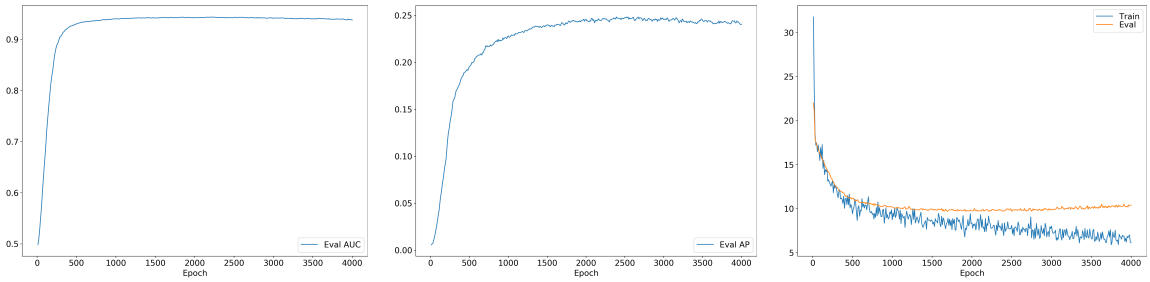
[2]Deep network will suffer from gradients vanishing problem if residual links are not added.

Results

The results on recurrent neural networks are shown in table 3 and the training curve is shown in figure 6. As we can see in the table, the good performances are stable with varying hyperparameters. Actually we do not fine tune the parameters for RNN. Hence it means that RNN is potential a better method than simple layer DNN.

	None	1024 Hidden Units	2048 Hidden Units
Single Layer DNN	94.5/0.242	N/A	N/A
6 Layers	N/A	94.2/0.234	94.4/0.246
10 Layers	N/A	94.1/0.233	94.4/0.244

Table 3: Experiments on Recurrent Networks



(a) AUC (b) AP (c) Loss

Figure 6: Figures on Recurrent Networks

3.4 Preventing Overfitting

The version of this dataset we used is a small subset of the original AudioSet consisting of only about 20 thousands entries, which means that overfitting problem can be really serious. As the training procedure goes through, the loss on training set is decreased much more quickly than the counterpart on evalution set. We provide the following two idea to prevent overfitting.

Adding Noise to Training Data

Adding noise to the training data can slightly extenuate the problem. In our experiments, we made 5 copies of the training data with different Gaussian noise. However, despite this method can only increase the robustness of the training process, the performance does not benefit from it. One hypothesis is that neural networks can easily learn to know that the data is noisy and can automatically calculate the deviation and the mean value of the noise. Proving this hypothesis is remained as future works.

Double Training Data by Time Inversion

This idea is inspired by the similar idea in the computer vision field that the problem of lacking data can be eased by flipping and adjusting light. However, though audio is sequential, this method does not bring any improvement on the final performance.

3.5 Platform Speed Comparation

Environment Settings

We compared the speed of DNN using three mainstream deep learning programming frameworks on a machine with 16 CPU and one GeForce GTX 1080 Ti GPU card.

Results

The table 4 shows elapsed CPU time and performances of the three frameworks.

Batch Size	Tensorflow	MXNet	PyTorch
Performance	94.5/0.242	94.4/0.242	93.3/0.206
Elapsed Time(s)	45.85	37.78	26.46

Table 4: Experiments on Comparing Frameworks' Performance

3.6 Ensemble Experiments

We use the ensemble methods of averaging weights among all of the good models (having AUC larger than 94). Finally we get the best model with AUC 94.9 and AP 0.257 compared to the best model with AUC 94.4 and AP 0.241 listed in table 5.

Model Type	AUC	AP
3-Conv-Layers CNN	94.2	0.236
10-Layers 2048-Units RNN	94.3	0.247
Single Layer DNN	94.4	0.241
Total	94.9	0.257

Table 5: Ensemble Models

4 Discussion and Future works

In this paper, we compared three basic neural network architectures, DNN, CNN and RNN on Google AudioSet, among which single layer DNN performs the best. However, with fine tunning the parameters, RNN and CNN are not worse than the single-layer DNN.

Besides, ensemble methods can truly improve the performance and this performance is better than the best performance among all models. However, due to the networks learning homogeneously, the averaging ensemble method did not bring a remarkable improvement on the final performance.

In the experiments of platform comparison, PyTorch is the fastest framework. But here we only tested single layer neural network while Tensorflow performs the best. Comparing more sophiscated networks is remained as future works.

Future works includes trying other complicated neural networks such like the combination layer of convolutional layer and recurrent layer as well as the other multi-label problem transformations such as learn the pairwise probability that label i is more likely to be selected compaired with lable j. Besides, having exploration on other datasets and trying the other ensemble methods are also worth a try.

References

[1] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[2] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[5] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014.

[6] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[7] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[8] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

[9] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[10] Jesse Read. Multi-label classification, July 2013.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.