Multiple Event Audio Detection

Arth Banka
Indian Institute of Technology Kanpur
arth20@iitk.ac.in

Abstract

Audio tagging is the task of identifying and predicting qualitative tags on audio data. Multiple event audio detection is performed to label audio sequences with possible classes it is constituted of. In this report, I present the different computational methods that can be used to perform the task of multi label tagging on mel spectrograms. The report contains introduction, literature survey, my methodology, a tabular representation of the precision, recall, F1 score, confusion matrix obtained for the various methods, observations and discussion.

1 Introduction

Multiple-event audio detection is the recognition of the type of audio amongst a set of established classes. For specific environments and applications, the types of events to categorise can vary. For the particular problem statement tackled here, audio tagging over the classes of ten sounds that are commonly observed in urban residential areas. The classes by which audio signals have been tagged are- 'Alarm_bell_ringing', 'Blender', 'Cat', 'Dishes' ,'Dog', 'Electric_shaver_toothbrush', 'Frying', 'Running_water', 'Speech', 'Vaccum_Cleaner'. The data-set provided in the problem statement to train the computational learning based systems consisted of a collection of ten thousand spectrograms and their corresponding labels in the form of multi-hot vectors. A validation data-set consisting of two thousand spectrograms was also provided. The spectrograms were mel-spectrograms that were single channel, with thousand time frames and nmels=64. The data was then processed and was used to train different learning based systems.

2 Literature Survey

An essential element of audio pattern classification is audio tagging, which focuses on determining whether audio tags are present in a particular audio sample. Literature survey revealed that acceptable methods to train computational models for audio tagging are D-prime, mean area under the curve (mAUC), and mean average precision (mAP). Several different sources were referred to in the process. IEEE conference proceedings, IEEE journals, ICASSP conference, NeurIPS, ICML, ICLR, DCASE challenges and DCASE workshops were studied to understand and implement the state of the art in audio tagging techniques. Literature survey helped in developing an understanding the accepted convention in multi-label classification taks. Other reference papers have also been added in the 'References' section.

3 Methodology

3.1 Data loading and processing

A function "data _loader" was defined to import the relevant data into the variables X and Y. The mel spectrograms were modified in the original data-set .The Y axis of the spectrograms was transformed into log scale or decibels by using the function "librosa.power _to _db" from the librosa library.

In some cases, the spectrograms were further transformed into mel frequency cepstral coefficients (MFCC). MFCC is obtained after performing a discrete cosine transform on the log scaled mel spectrogram. Since the MFCC is a further compression of the mel spectrogram performance is often worse on it compared to a mel spectrogram, however MFCCs are compressed forms of mel spectrograms so they can be used in networks to prevent resource exhaustion errors. In the computational models made for audio tagging I have used mel spectrograms since they lead to better performance. The spectrograms were zero-padded before loading into X, by using another self defined function "padder" which was designed to ensure that the data was of uniform dimensions. The labels were encoded into multi-hot vectors for training the learning based systems and was achieved through the eventroll_to_multihot_vector. The data was converted into and stored as numpy arrays.

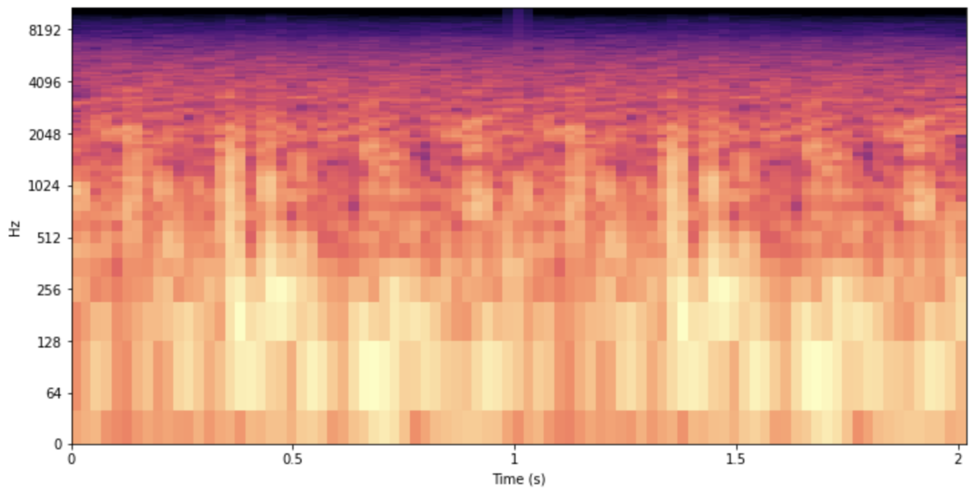


Figure 1: A sample image of a mel spectrogram after being modified to log-scale.

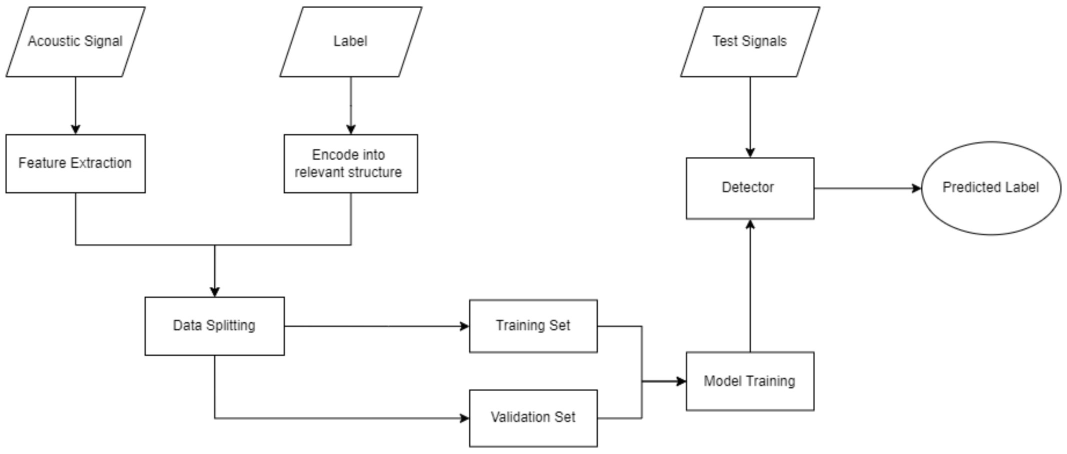


Figure 2: Standard pipeline of a single event audio detector.

3.2 ANN Model

Artificial neural networks are capable of learning complex non-linear functions from the input audio features to their classes, improving performance. A feed-forward artificial neural network consists of an input layer of the dimension of features, an output layer of the dimension of classes, and multiple hidden layers consisting of neurons. The weights of these layers transform the inputs to a linear combination upon which an activation function like tanh or ReLU can be applied. Back-propagation is used to learn the weights and biases for these layers, with the gradient being propagated through the binary cross-entropy loss. Gradient descent on the derivatives obtained by back-propagation leads to learning optimal classification weights.

The network architecture for the neural network was obtained after hyper-parameter variation. I used a layer to flatten the input MFCC data, followed by four dense layers and one dense output layer in my model. The first four layers had 1024, 512, 256, and 128 neurons, respectively, each having ReLU activation. The output layer had ten neurons to represent the ten different classes and used sigmoidal activation.

3.3 RNN Model

As they can accommodate long-term temporal dependence, recurrent neural networks have been shown to be successful in making predictions incorporating time series data.Traditional neural networks have inputs and outputs that are independent of one another, but there is a need to remember the previous words in situations where it is necessary to anticipate the next word in a sentence. They stand out due to their "memory," which allows them to affect the current input and output by using data from previous inputs.

In the model architecture that I have used the input layer in an LSTM with 128 neurons. Overfitting has been prevented via a dropout layer that has been placed between the time dispersed layer and the LSTM layers. Then a sequence of dense layers with ReLu activation have been used. The number of neurons in these dense layers progressively decreased as 512, 256, 128, 64 and 32. After the dense layers an ELU activation layer performs the identity operation on positive inputs and an exponential nonlinearity on negative inputs. This is followed by a droput layer for preventing over-fitting and a sigmoidal layer for tagging with the ten different labels.

For temporal data, LSTMs return sequences. The time dispersed layers create the relationship between the current and prior neuronal outputs. RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.

3.4 CNN Model

Convolutional Neural networks (CNN) are widely used in tasks such as object detection, which is also a version of the classification problem. In sound event detection problems, the kernels are used to learn patterns in a sequence of spectrogram frames characteristic to a particular class. In CNN, we input an image of height × width × dimension. Therefore we modify the input data by adding an extra dimension using the reshape function. The input data used in this case were log scaled mel spectrograms since MFCCs are more decorrelated which does not mesh well with strong classifiers such as CNN.
The model architecture was arrived upon after hyper-parameter variation and trial and error. The final network architecture that gave the best result on the data contains consisted of five convolution layers of 32, 64, 64, 128, 256 kernels respectively, each having ReLU (Rectified Linear Unit) activation linearity. Each layer has strides of (1,1) and kernel size of (3,3) except the second convolution layer which has utilises a kernel of size (2,2). Max pooling has been used to since our data size is quite large and kernel size of (2,2) prevents loss of information. The increasing number of kernels ensure that the network learns sufficient specific features from the basic features for classification. Dropout layers have been added to prevent over-fitting during training. The flattened output is then passed through a dense layer of 512 neurons with the ReLU activation and finally into a dense layer of 10 neurons with the softmax activation. Batch size of 32 was used for training and batch normalization was used in the layers to ensure that inputs were standardised for each batch. The loss function used in the model was categorical cross entropy and optimizer was 'adam'.

3.5 CRNN Model

The Convolutional Recurrent Neural Networks are two of the most well-known neural networks combined. CRNN combines CNN (convolutional neural network) and RNN (recurrent neural network)

(Recurrent neural networks). Similar to the CRNN, the suggested network produces superior or optimal outcomes, particularly for audio signal processing. To extract useful features, convolutions and max-poling processes are consecutively applied. The gated recurrent unit (GRU) is then fed these in order to extract the temporal data. The outputs of the network are sigmoidal scores, which represent the variety of active acoustic events in the audio signal.

The network architecture used starts with a 2D CNN network that consisted of five convolution layers of 32, 64, 64, 128, 256 kernels respectively, each having ReLU (Rectified Linear Unit) activation linearity. Each layer has strides of (1,1) and kernel size of (3,3) except the second convolution layer which has utilises a kernel of size (2,2). Max pooling has been used to since our data size is quite large and kernel size of (2,2) prevents loss of information. The data was then reshaped and fed into a GRU layer with 128 units. This was followed by a dense layer of 256 neurons which fed the data into the sigmoidal output layer.

4 Results and Confusion Matrix

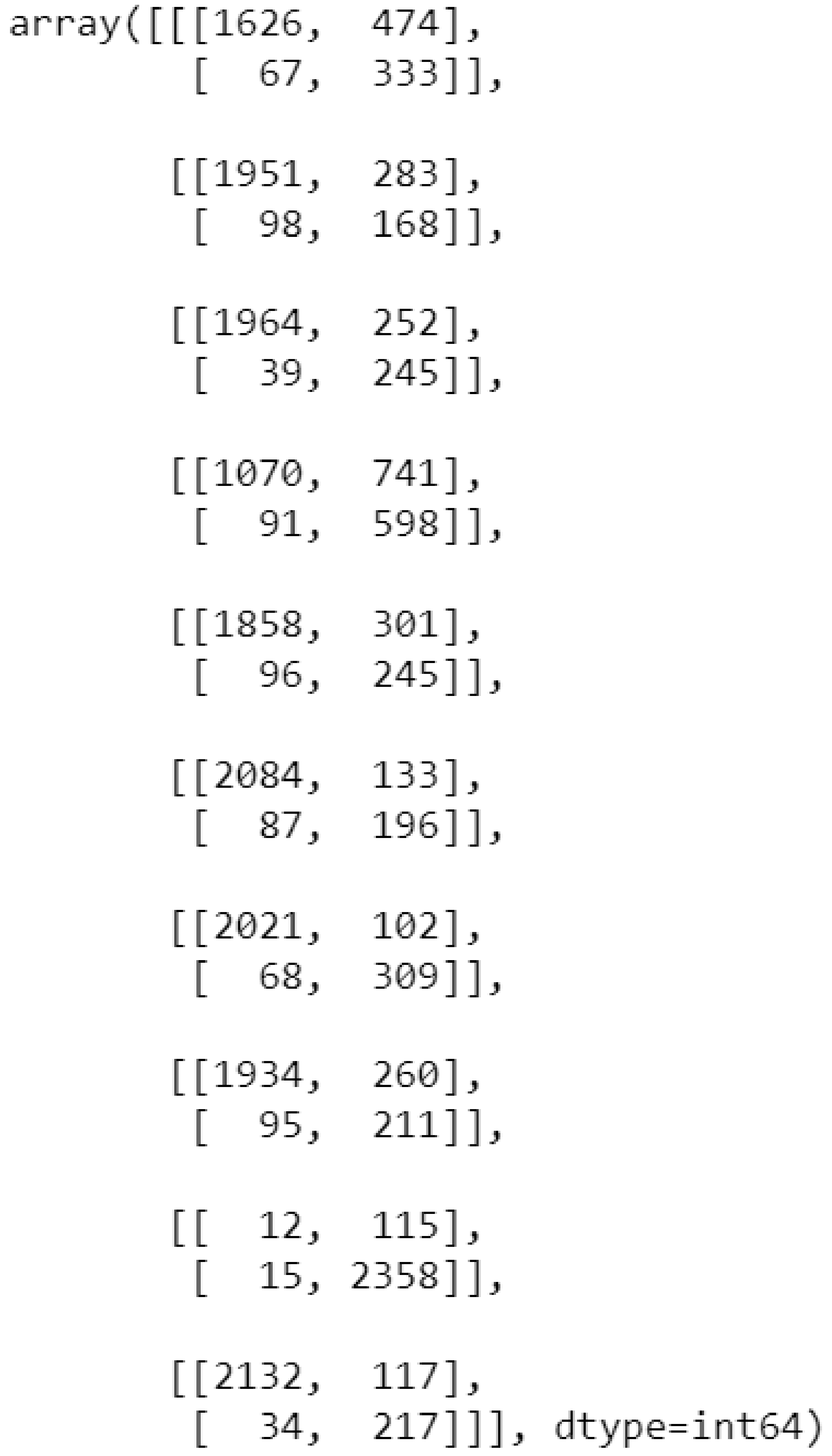
A test set consisting of 2500 samples was released to verify the performance of the learning based systems. The results obtained have been shown in tabular form below.

CRNN Evaluation Metrics (Test Set)				
Metrics	Micro-Avg	Macro-Avg	Weighted-Avg	Samples-Avg
Precision	0.64	0.56	0.70	0.70
Recall	0.88	0.80	0.88	0.89
F1 Score	0.74	0.65	0.76	0.76

CNN Evaluation Metrics (Test Set)				
Metrics	Micro-Avg	Macro-Avg	Weighted-Avg	Samples-Avg
Precision	0.60	0.52	0.66	0.66
Recall	0.82	0.78	0.84	0.84
F1 Score	0.66	0.57	0.71	0.71

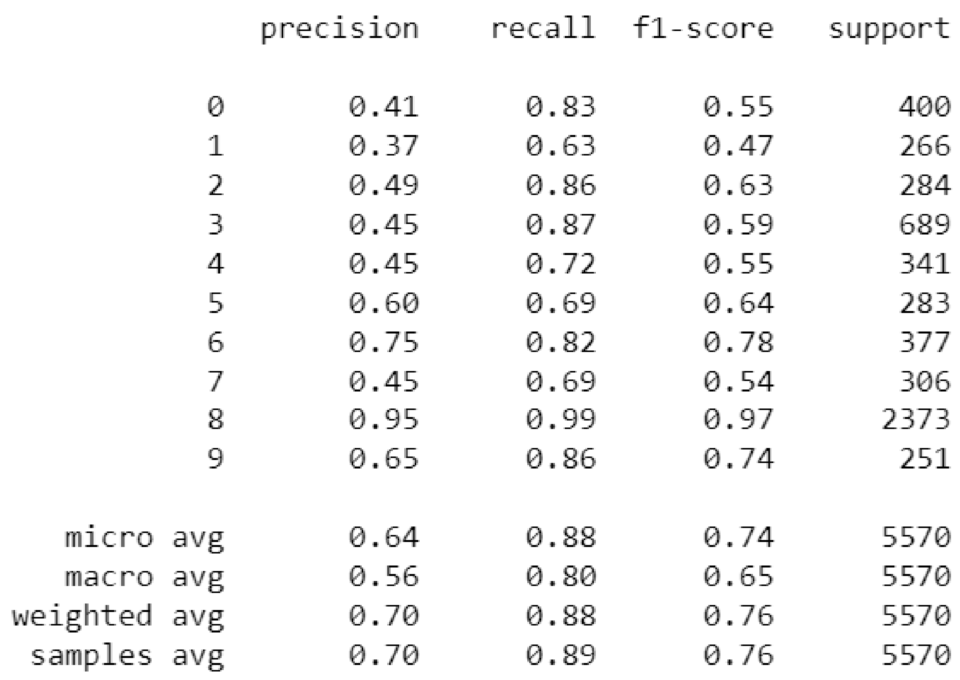
RNN Evaluation Metrics (Test Set)				
Metrics	Micro-Avg	Macro-Avg	Weighted-Avg	Samples-Avg
Precision	0.34	0.22	0.37	0.39
Recall	0.62	0.48	0.61	0.62
F1 Score	0.39	0.28	0.44	0.45

NN Evaluation Metrics (Test Set)				
Metrics	Micro-Avg	Macro-Avg	Weighted-Avg	Samples-Avg
Precision	0.17	0.13	0.19	0.19
Recall	0.40	0.33	0.46	0.49
F1 Score	0.24	0.20	0.26	0.27



```
array([[[1626,  474],
        [  67,  333]],

       [[1951,  283],
        [  98,  168]],

       [[1964,  252],
        [  39,  245]],

       [[1070,  741],
        [  91,  598]],

       [[1858,  301],
        [  96,  245]],

       [[2084,  133],
        [  87,  196]],

       [[2021,  102],
        [  68,  309]],

       [[1934,  260],
        [  95,  211]],

       [[  12,  115],
        [  15, 2358]],

       [[2132,  117],
        [  34,  217]]], dtype=int64)
```

Figure 3: The confusion matrix heat map for the CRNN model(best model) for ground truth/test set.



```
              precision    recall  f1-score   support

           0       0.41      0.83      0.55       400
           1       0.37      0.63      0.47       266
           2       0.49      0.86      0.63       284
           3       0.45      0.87      0.59       689
           4       0.45      0.72      0.55       341
           5       0.60      0.69      0.64       283
           6       0.75      0.82      0.78       377
           7       0.45      0.69      0.54       306
           8       0.95      0.99      0.97      2373
           9       0.65      0.86      0.74       251

   micro avg       0.64      0.88      0.74      5570
   macro avg       0.56      0.80      0.65      5570
weighted avg       0.70      0.88      0.76      5570
 samples avg       0.70      0.89      0.76      5570
```

Figure 4: The classification report for the CRNN model(best model) for ground truth/test set.

5 Discussion and Conclusion

Several different methods have been implemented for multiple event audio detection. Deep learning systems demonstrate descent performance at tagging of audio inputs with their respective classes. Significant variations in recording style and even modelling noise can only be captured by huge data-sets, and without that, severe performance deterioration is seen.

From literature survey I had found that accuracy and binary accuracy are inefficient methods of training for neural networks since they match the high number of similar zeroes in the multihot vector leading to very high accuracy scores and this led me to train my models under two metrics- f1_score and AUC. The influence of the true negatives can be seen by the area under the false positive rate and true positive rate (recall), or AUC. F1 score allowed me to ensure that the precision and recall were also scaling well in the model. All metrics are computed for each class separately before being averaged across all classes. They are also known as macro metrics and are more adept for multi-label classification.

We had the best results with our CRNN as can be seen in the figures and tables above. The CNN proved to be the second best metric followed by RNNs and ANNs. Part of this success stems from the CNN sharing its weights across multiple features. Therefore it was not as susceptible to overfitting as the other models I tried. Due to high number of parameters which need to be trained for ANNs we find that adequate performance can only be attained for sufficiently large datasets. ANNs mostly hardcode learning so they don't perform adequately for time variant audio data sets. The same holds for RNNs. While the CNN does have better performance and is more robust, we should also note that it can be as much as three times slower than the ANN. A very interesting application of the data-set given to us was its use in security systems in households and to detect suspicious activity in public places. Future work could focus on newer directions of research which seek to learn better representations for data like contrastive learning which also leverages vast unlabelled datasets by learning under self supervision.

6 References

1. H. Liu and C. Zhang, "Reinforcement Learning based Neural Architecture Search for Audio Tagging," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207530.

2. G. Guo and Stan Z Li, “Content-based audio classification and retrieval by support vector machines,” IEEE Transactions on Neural Networks, vol. 14, no. 1, pp. 209–215, 2003

3. Iqbal, Kong, Q., Plumbley, M. D., and Wang, W. (2018). General-purpose audio tagging from noisy labels using convolutional neural networks.212–216. https://openresearch.surrey.ac.uk/esploro/outputs/conferencePresentation/General-purpose-audio-tagging-from-noisy-labels/99514815102346file-0

4. E. Cakır, T. Heittola, and T. Virtanen, “Domestic audio tagging with convolutional neural networks,” IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE 2016), 2016

5. T. Lidy and A. Schindler, “Cqt-based convolutional neural networks for audio scene classification,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016), vol. 90. DCASE2016 Challenge, 2016, pp. 1032–1048.

6. http://dcase.community/challenge2018/task-large-scaleweakly-labeled-semi-supervised-sound-event-detection

7. D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, “Detection and classification of acoustic scenes and events,” IEEE Transactions on Multimedia, vol. 17, no. 10, pp. 1733–1746, 2015.

8. K. Choi, G. Fazekas, and M. Sandler, “Automatic tagging using deep convolutional neural networks,” arXiv preprint arXiv: 1606.00298, 2016.

9. Guangxiao Song, Zhijie Wang, Fang Han, Shenyi Ding, Muhammad Ather Iqbal, Music auto-tagging using deep Recurrent Neural Networks, Neurocomputing, Volume 292, 2018, Pages 104-110, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2018.02.076.

7 Appendix

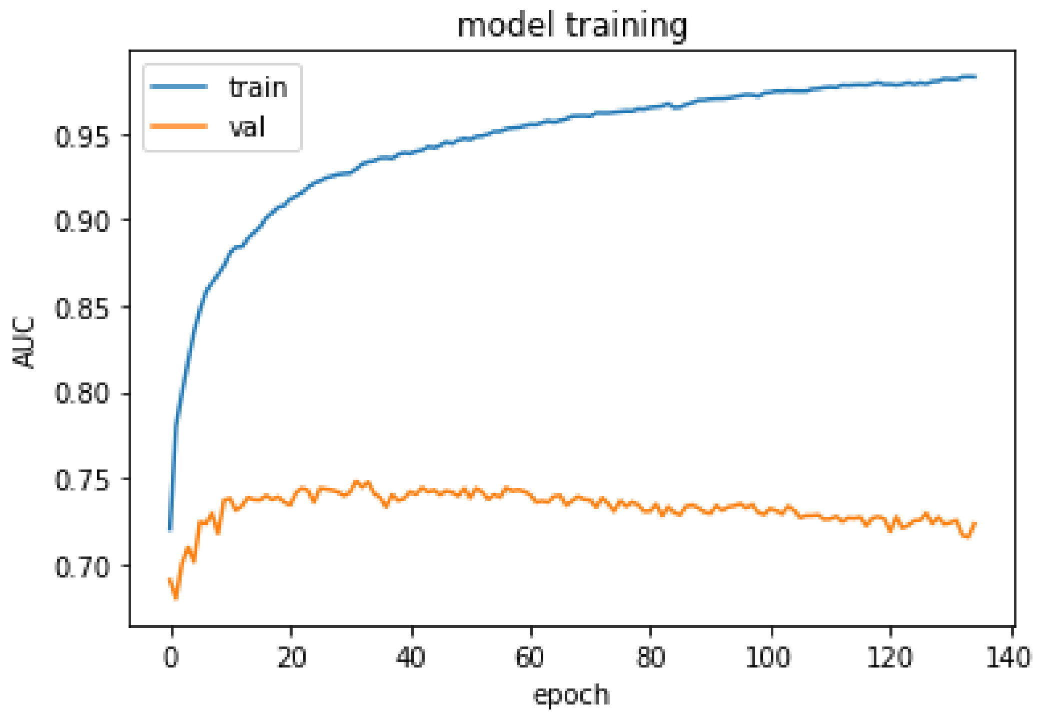


Figure 5: The AUC as a function of epochs for RNN.

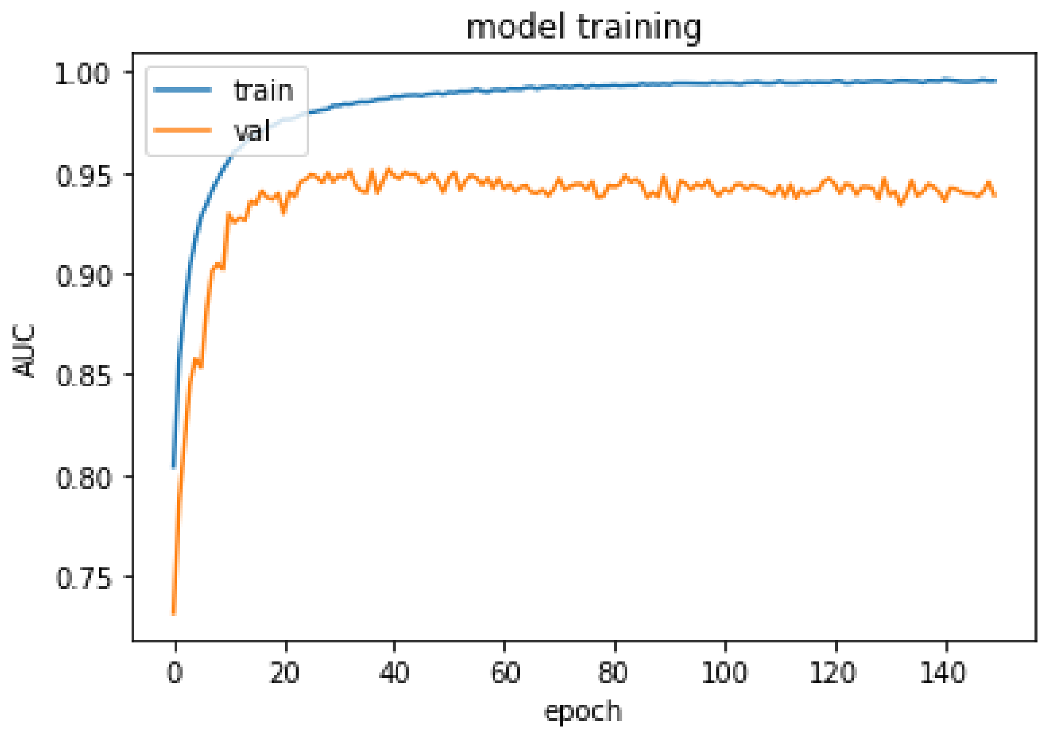


Figure 6: The AUC as a function of epochs for CRNN.

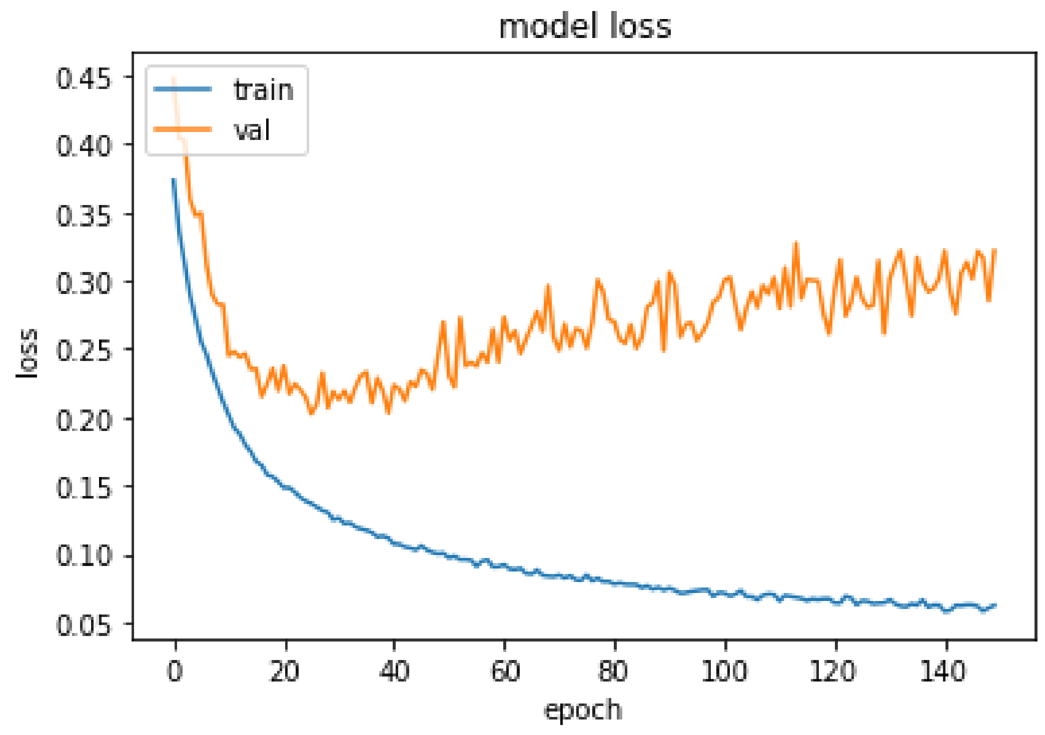


Figure 7: The loss curve of binary cross-entropy as a function of epochs for CRNN.