Speaker Diarization: Using Recurrent Neural Networks

Vishal Sharma, Zekun Zhang, Zachary Neubert

December 14, 2017

Abstract

Speaker Diarization is the problem of separating speakers in an audio. There could be any number of speakers and final result should state when speaker starts and ends. In this project, we analyze given audio file with 2 channels and 2 speakers (on separate channel). We train Neural Network for learning when a person is speaking. We use different type of Neural Networks specifically, Single Layer Perceptron (SLP), Multi Layer Perceptron (MLP), Recurrent Neural Network (RNN) and Convolution Neural Network (CNN) we achieve ∼92% of accuracy with RNN.

1 Introduction

Speaker diarization is the task of determining "who spoke when?"[ABE+10] in an audio with unknown number of speakers and unknown length of audio. It is of great importance in the domain of speech research and has several applications in the field of information retrieval [ABE+12] [CTS17].

2 Dataset

Our dataset contains 37 audio files approximately of 15 minutes each with sampling rate of 44100 samples/second, recorded in 2 channels with exactly 2 speakers on 2 different microphones. Each audio file has been hand annotated for speakers timings. Annotating timing (in seconds) they start and stop speaking. We use this dataset and split in 3 parts for training, validation and testing.

3 Preprocessing

3.1 Data Normalization

We perform normalization of audio files after observing recorded audio was not in the same scale. Few audio files were louder than others and normalization can help bring all audio files to same scale.

3.2 Sampling Audio

With frame rate being high, we have a lot of data. To give an example, in a 15 min audio file we get about 40M samples in each channel. To reduce data without loosing much information, we down sample audio files by every 4 sample. [Gia15]

3.3 Cleaning Labels

Provided labels needed some cleaning described below:

1. Names of the speakers was not consistent throughout the data file, we cleaned it and made sure name is consistent.
2. File also contained unicode, which needed to be cleaned. Python goes crazy with unicodes lol
3. There were miss alignments as well in the data and needed to be removed and fixed.

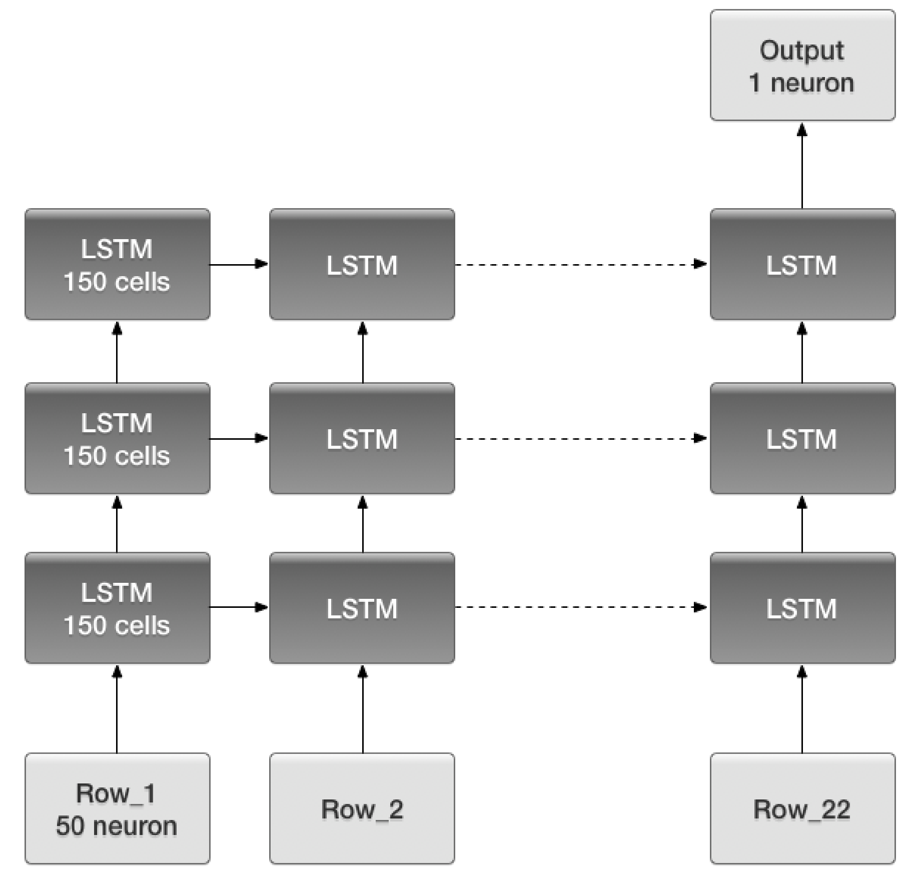

Figure 1: Structure of RNN using TensorFlow.

4 Approach

4.1 Multi-layer Perceptron

We start with a basic single layer perceptron model. We implement 3 different models with hidden layer of different sizes 100, 200, 500 neurons. We achieve approximately 86% accuracy. We next move to multi-layer perceptron model and try models with 2 layers deep. First layer had 100 and second 50 neurons and another with higher number of neurons (First Layer: 200, Second Layer: 100) (First Layer: 300, Second Layer: 50). For all the networks used in this project, the hidden neurons are ReLu [XWCL15] and the output neuron are sigmoid. The cost function used is cross entropy and mini-batch gradient descent with Adam optimization is used to train network.

4.2 Recurrent Neural Network (RNN)

Next we try Recurrent Neural Network [MKB+10] on the classification problem. The RNN gives us the best result with 3 layers each with 150 Long short-term memory (LSTM) cells. The LSTM in the graph means a LSTM layer which consists of 150 LSTM cells. The output only has one neuron with sigmoid to predict 0 or 1.

4.3 Convolution Neural Network

To apply CNN [KSH12], we at first compute the spectrogram for each row of the data matrix, then store them into a new file by using pickle. In this way we don't need to compute spectrogram online and hence can save a lot of training time. Function scipy. signal.spectrogram is used to compute the spectrogram for each segment. The recomputed spectrogram of each segment then is organized to a 3 dimension matrix with shape (number of segments, height, width). For example, the down sampled data matrix of a channel returned by get data has the shape (100, 1102) for a channel with 100 segments, then the shape of recomputed spectrogram matrix is (100,129,4). The

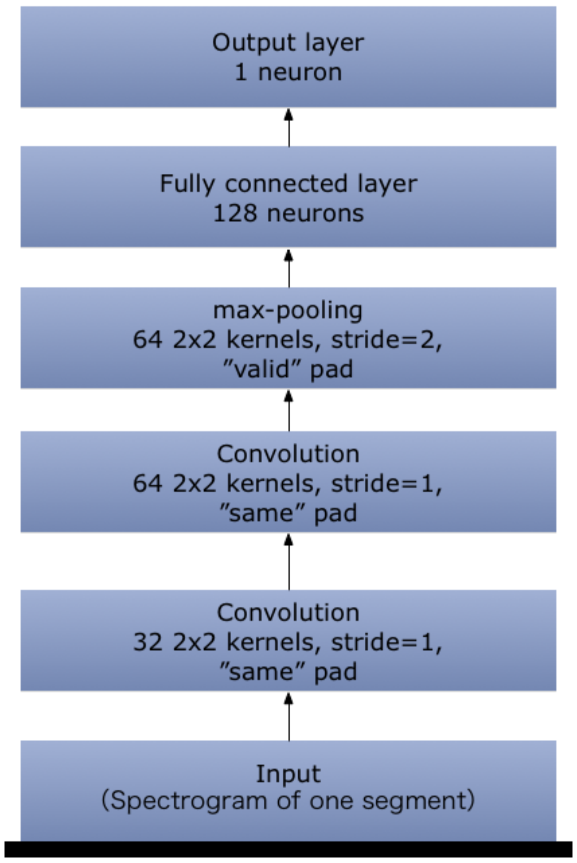

Figure 2: Structure of CNN using TensorFlow.

number of segments remains the same. The height 129 and width 4 come from using the default parameters of function scipy. signal.spectrogram. Spectrogram matrices are computed and stored by using code in Spectrogram Generator.

References

[ABE+10] Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. Speaker diarization: A review of recent research. 2010.

[ABE+12] Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370, 2012.

[CTS17] Pawel Cyrta, Tomasz Trzcinski, and Wojciech Stokowiec. Speaker diarization using deep recurrent convolutional neural networks for speaker embeddings. *CoRR*, abs/1708.02840, 2017.

[Gia15] Theodoros Giannakopoulos. pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12):e0144610, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[MKB+10] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[XWCL15] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

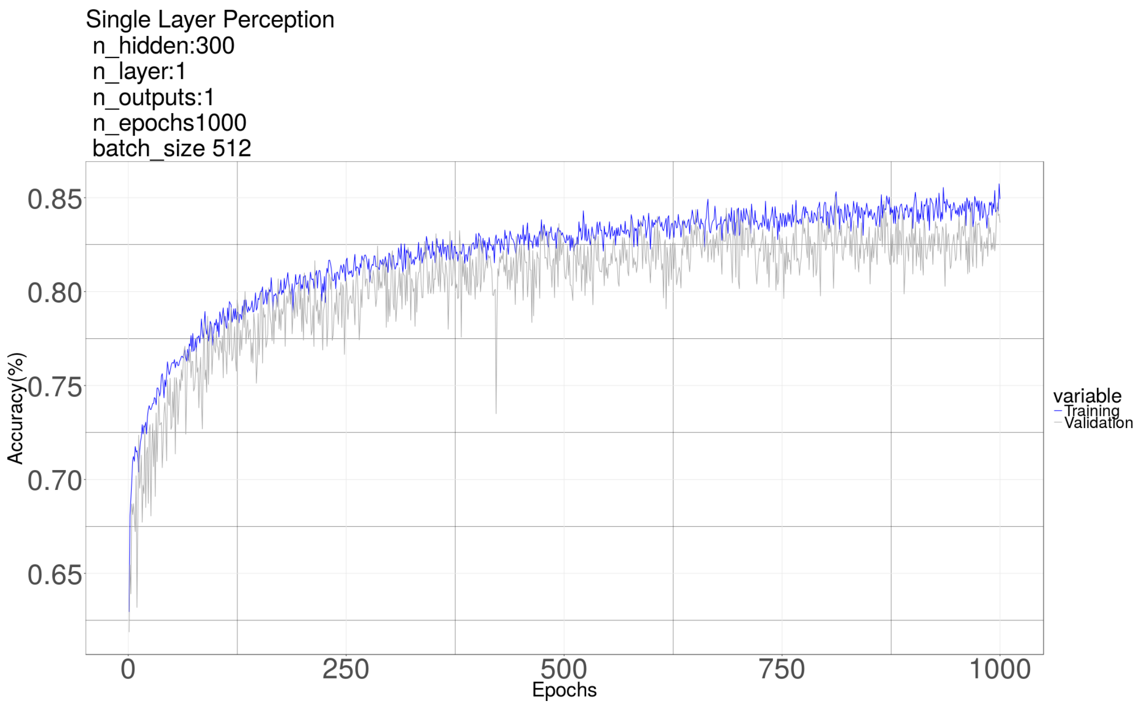

Figure 3: Results of Single Layer Perceptron on Training and Validation set. Title contains information about configuration of the network.

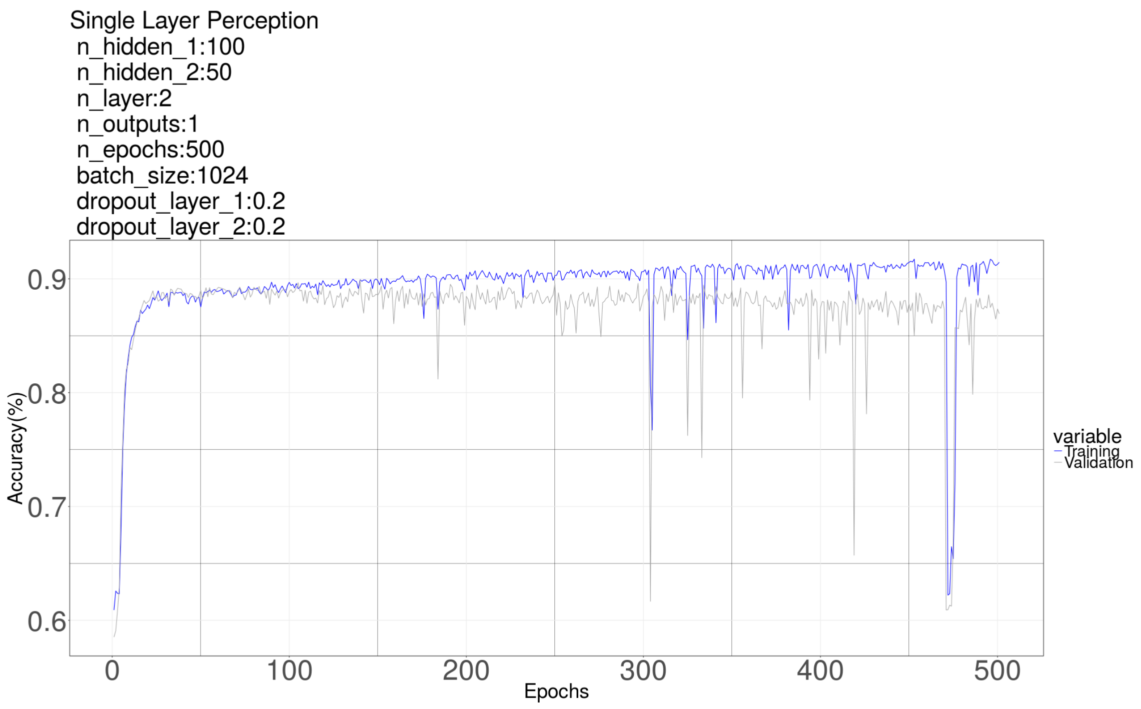

Figure 4: Results of Multi Layer Perceptron on Training and Validation set. Title contains information about configuration of the network.

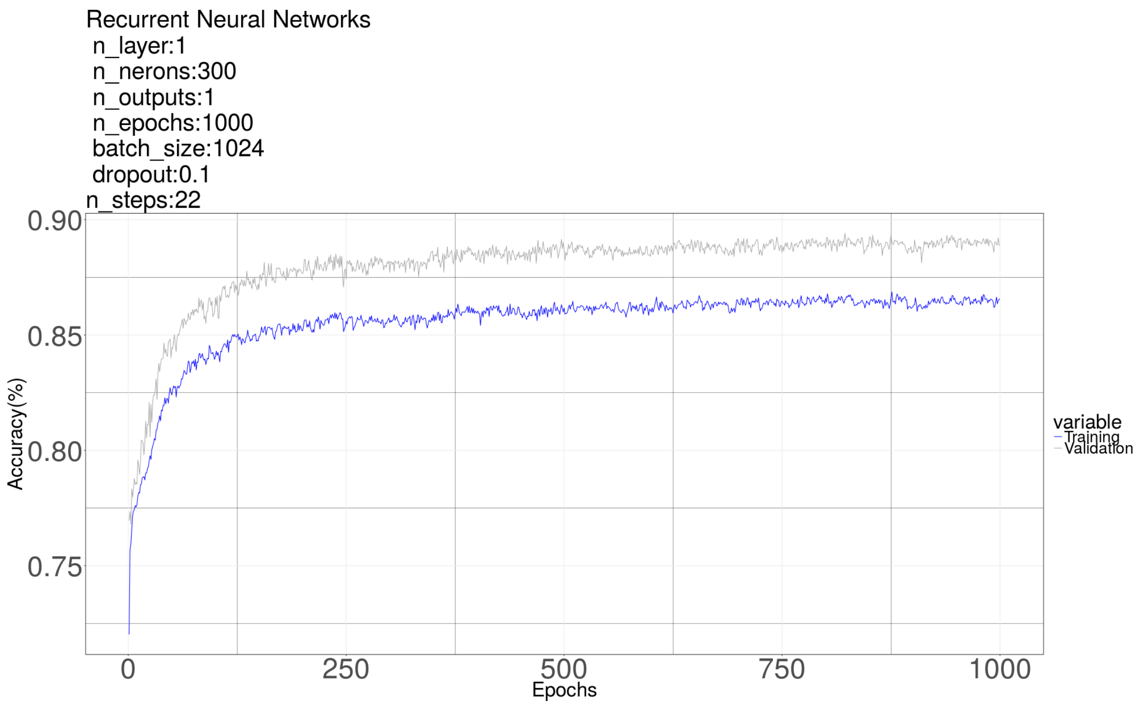

Figure 5: Results of Recurrent Neural Network on Training and Validation set. Title contains information about configuration of the network.

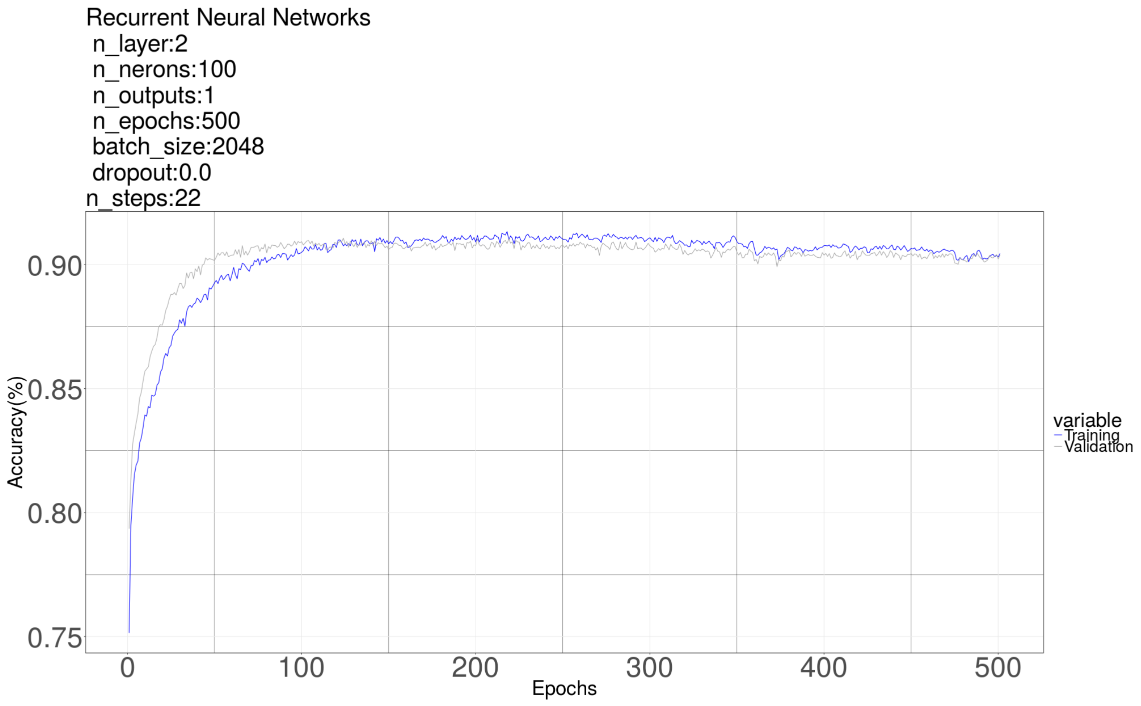

Figure 6: Results of Recurrent Neural Network on Training and Validation set. Title contains information about configuration of the network.

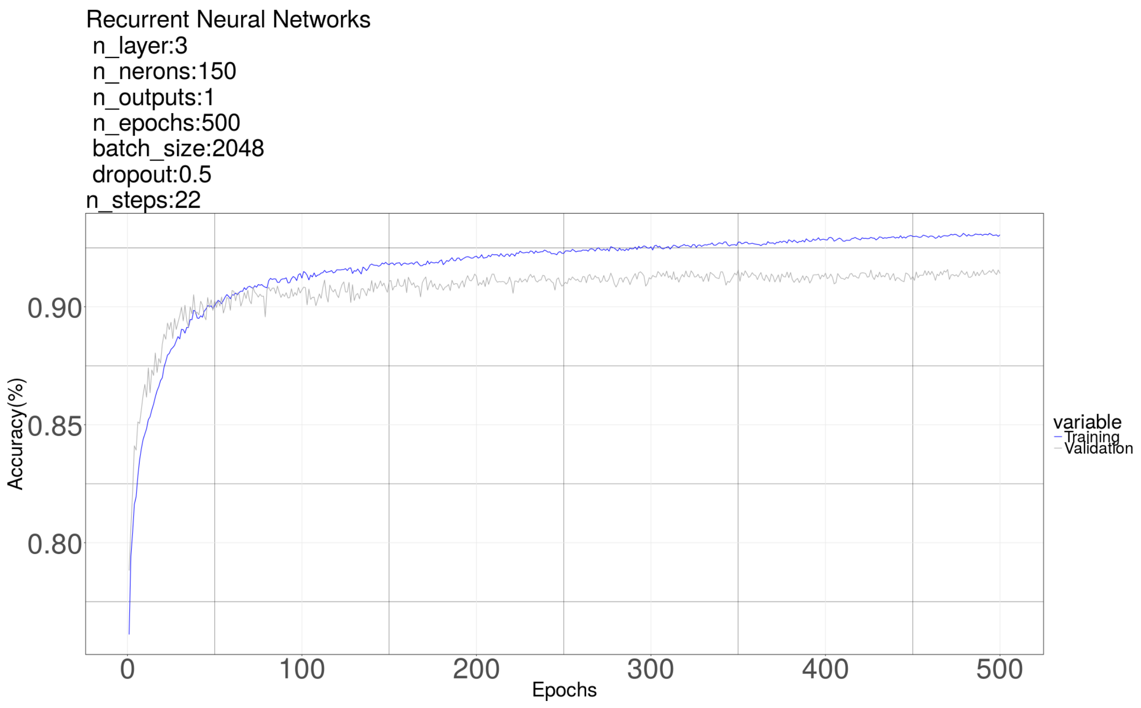

Figure 7: Results of Recurrent Neural Network on Training and Validation set. Title contains information about configuration of the network.

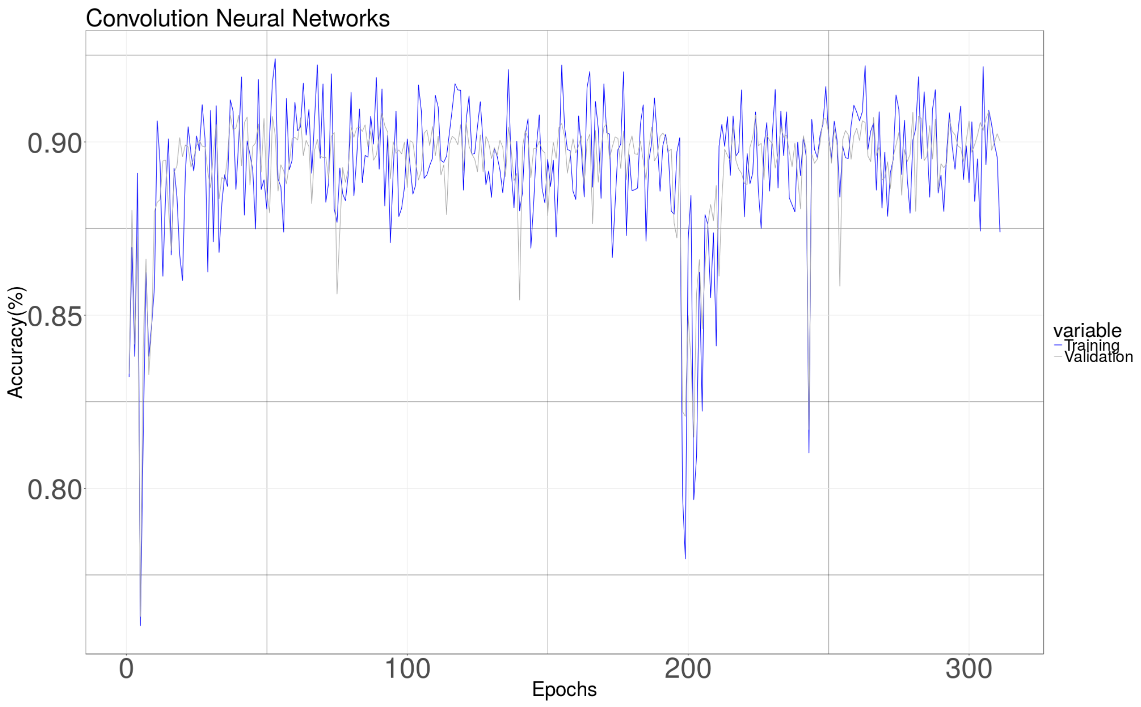

Figure 8: Results of Convolution Neural Network on Training and Validation set. Title contains information about configuration of the network.

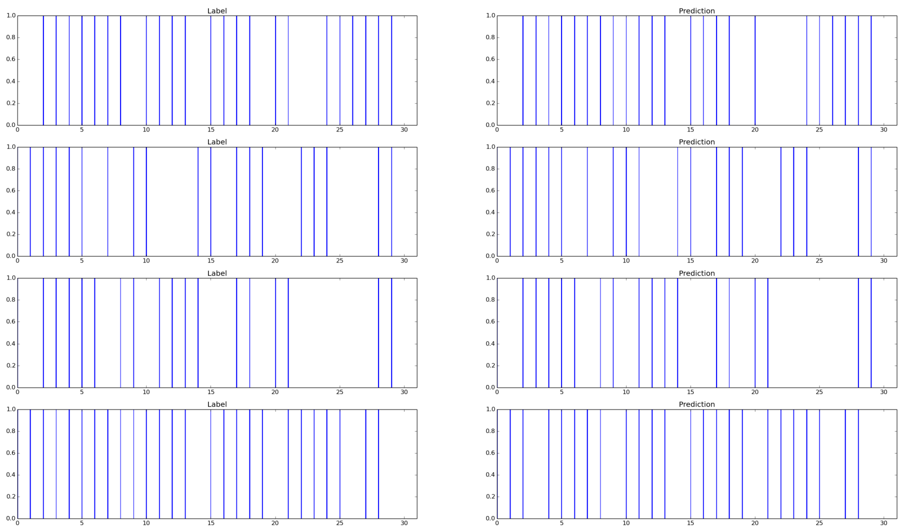

Figure 9: Compare Predicted with given Label.