Connor McCurley

EEE 6814 **Homework 3** Fall 2019

Problem Description

The goal of this assignment was to train two taxonomies of neural networks to process temporal data. The idea is that we had grammar sequences of variable length and we wanted the networks to learn to identify whether a new sequence followed the grammar structure. In our case, the in-grammar hypothesis followed the structure of {0*,1}. This means that a sequence could contain any number of zeros, as long as it was followed by a single one. Any other binary pattern was considered to fall outside of the grammar.

Time Delay Neural Network (TDNN)

The first network taxonomy that was implemented was a time delay neural network. While I would have liked to test a multitude of network topologies, I just did not have time for this assignment. Because of that, I simply fixed the networks to have a single hidden layer (with ReLU activations) and a single output neuron (with sigmoid to provide 0/1 labels). Window sizes of 7 and 20 were compared. Simple block diagrams for the two architectures are shown in Figures 1 and 2.

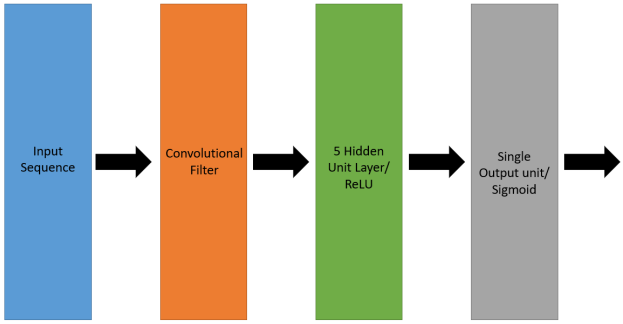

Figure 1: Architecture of a TDNN with a window size of 7. A 1D convolutional layer creates a feature vector for the input sequence before passing it along to a hidden layer and a subsequent output layer. A single label is predicted for the entire sequence.

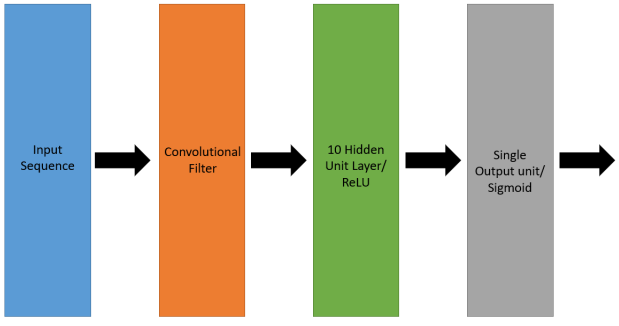

Figure 2: Architecture of a TDNN with a window size of 20. A 1D convolutional layer creates a feature vector for the input sequence before passing it along to a hidden layer and a subsequent output layer. A single label is predicted for the entire sequence.

While the homework specified that we generate data sequences of variable length, the network needed to take in a fixed feature map. To resolve this issue, the zeros in the sequence were changed to -1 (which is also just good practice as to actually utilize the full network), and the sequences were zero-padded (if needed). A 1D convolution was applied on the entire input sequence to simulate a sliding window. Both networks were then trained in batch 10 times each (with random initialization). Each network (TDNN and RNN) used the Adam optimizer with binary cross-entropy loss. The learning rate was fixed at $\eta = 0.001$. The best-performing model was then saved and used on the posted test set. Learning curves demonstrating the binary cross-entropy loss for training and validation sets are shown in Figures 3 and 4. The confusion matrices in Figures 5 and 6 demonstrate classification performance for each model on the held-out test set. As can be seen from the figures, performance from the network with a window size of 20 slightly outperformed that of window-size 7. My intuition is that, for these simple binary sequences, the network can capture relevant features more easily from a global perspective. However, this is highly data dependent. These results demonstrates the importance of properly selecting the window hyper-parameter. Additionally, I presume the zero-padding had a detrimental affect on performance (since the longer test sequences did not have padding). This likely caused both networks to under-perform.

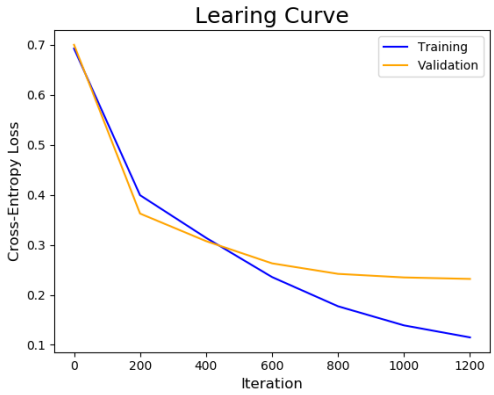

Figure 3: Learning curves for the TDNN with a window size of 7. The validation loss begins to increase at approximately training epoch 1200.

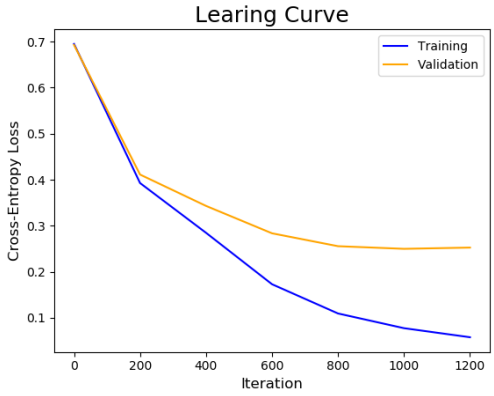

Figure 4: Learning curves for the TDNN with a window size of 20. The validation loss begins to increase at approximately training epoch 900.

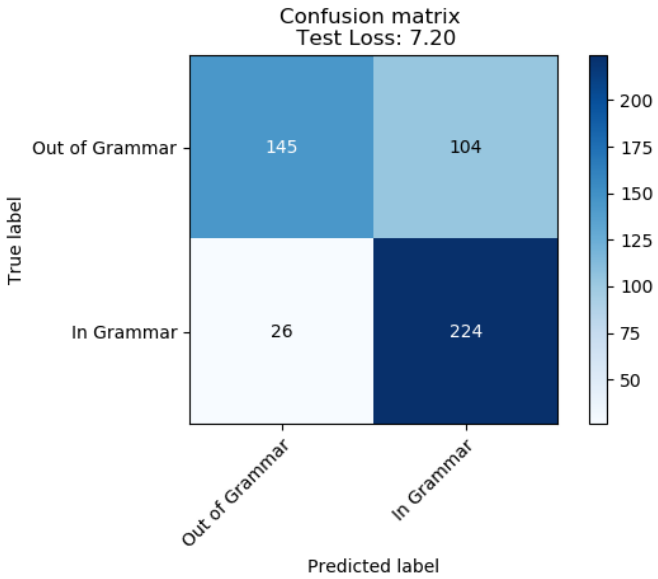

Figure 5: Confusion matrix for the TDNN with a window size of 7. Out of the 500 test sequences, this particular network mis-classified 130. 26 in-grammar sequences were predicted as out of grammar, while 104 out of grammar sequences were predicted in-grammar.

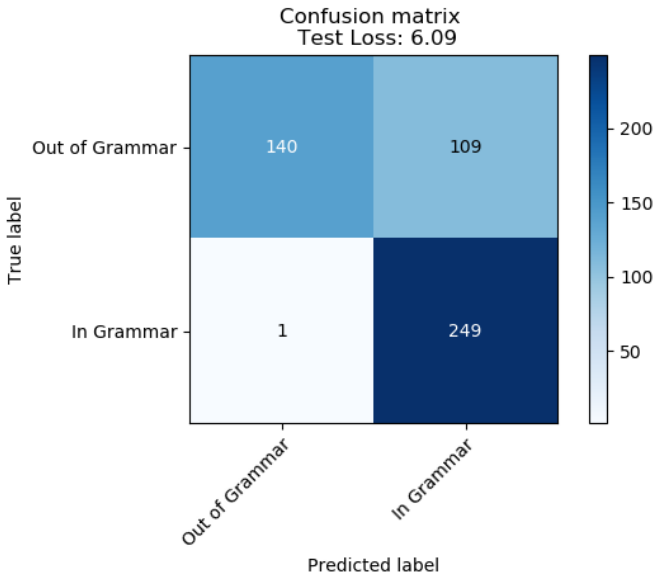

Figure 6: Confusion matrix for the TDNN with a window size of 20. Out of the 500 test sequences, this particular network mis-classified 110. 1 in-grammar sequence was predicted as out of grammar, while 109 out of grammar sequences were predicted as in-grammar.

Recurrent Neural Network (RNN)

The second taxonomy implemented was a simple recurrent network. Again, while I would have liked to try many configurations/ hyper-parameters, I simply did not have time. For this experimentation, I chose a RNN with two hidden-state units and a single output-state unit. Learning curves and the test confusion matrix for the best model are shown in Figures 7 and 8, respectively. As can be seen from the figures, this network implementation greatly outperformed the TDNN architectures. In this case, only 41 test samples were mis-classified (however all of the true in-grammar samples were correctly classified). Additionally, we were not required to select a window size. This greatly facilitated implementation and network tuning.

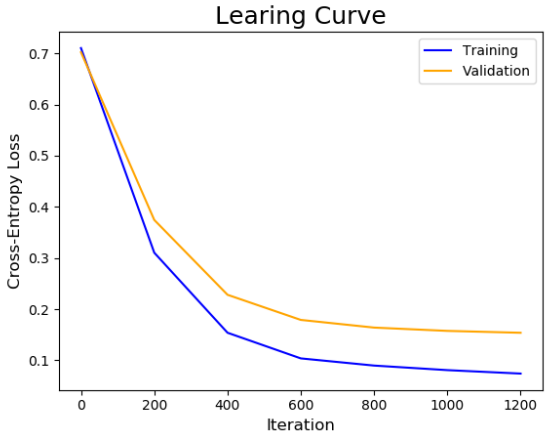

Figure 7: Learning curves for the RNN. The validation loss begins to increase at approximately training epoch 1200.

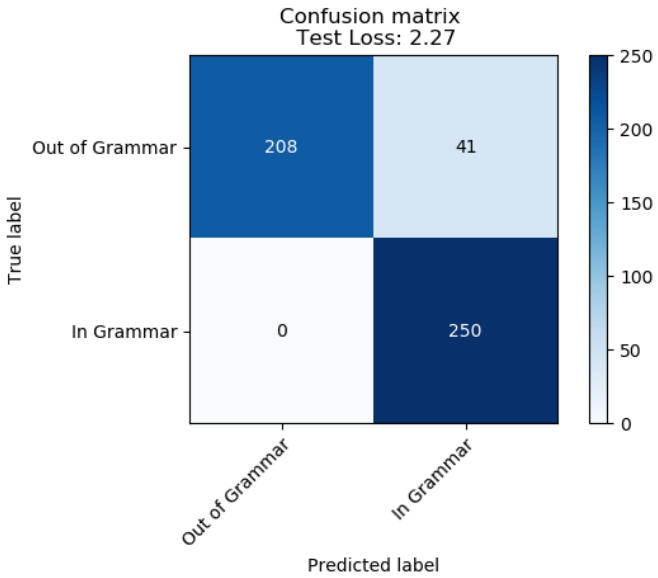

Figure 8: Confusion matrix for the RNN. Out of the 500 test sequences, this particular network mis-classified 41. 0 in-grammar sequences were predicted as out of grammar, while 41 out of grammar sequences were predicted in-grammar.

References

[1] Principe, Jose C., Euliano, Niel R., Lefebvre, W. Curt. "Chapter X- Temporal Processing with Neural Networks," in Neural and Adaptive Systems: Fundamentals Through Simulation, 1997.

[2] Principe, Jose C., Euliano, Niel R., Lefebvre, W. Curt. "Chapter XI- Training and Using Recurrent Networks," in Neural and Adaptive Systems: Fundamentals Through Simulation, 1997.