CSCI 635 Introduction to Machine Learning

Project 2

Krystian Derhak & Michael Lee

Professor Avenoso
Due: 05/02/2022

File Structure

- project2.pdf
- project2.zip
  - project2_code.py
  - LSTM.model
  - BLSTM.model
  - Bonus.model
  - README.txt

Overview

This project is a continuation of the first project which endeavoured to use neural networks to classify single note recordings of music from the NSynth dataset [1]. The first project focused on using Convolutional Neural Networks (CNN's) which in our best case yielded an accuracy of 57.15%. The parameters and requirements of the project remained the same with an accuracy between 50%-60% and included figures below. This time we completed the goal by using Long Short-term Memory, or LSTM networks which typically handle sequential data better.

Since we could rely heavily on most of our code structure and familiarity with the dataset from project 1, we focused the bulk of our energy and time on understanding LSTMs, especially in how to implement them in keras. This involved specifically reading and relying on the documentation from both Tensorflow and Keras teams [2] [3] but also additional articles that walked through some of the particulars [4] [5]. As for construction each network had 2 LSTM/BLSTM layers followed by dropout layers with a rate of 0.2, Dense layers with relu activation functions and a final dense layer with a softmax activation function. See the Figures section for full details of each network.

Our normal LSTM model was able to achieve an accuracy of 55.22% and our BLSTM achieved an accuracy of 55.10% and our bonus network achieved an accuracy of 66.28%. While it might seem odd that both the LSTM and BLSTM achieved similar accuracies, this was impressive as the BLSTM took less memory and had fewer parameters and was still able to achieve the same accuracy. As for the bonus network, its higher accuracy was achieved due to us increasing the amount of parameters that were trained on.

Training

In order to speed up the training process and focus on the important parts, we trimmed all examples to only include the first second. Additionally we reduced the shape through 1D average pooling. We tried our best to keep as many parameters similar between this and project one in order to allow for the best comparison. Therefore we kept the Adam optimizer, learning rate of 0.001, a batch size of 1024, and 30 epochs. We also tried to keep our parameters around 1,000,000.

Training our CNNs took approximately 15 minutes but training our LSTMs took approximately 35 minutes. Due to the similarities to our first project, we did not have to do as much trial and error to achieve our accuracies but we did have to spend more time training as they took nearly twice as long to train.

When training our first LSTM network was relatively easy, but our first attempt at a BLSTM ended up being too large for our memory, with nearly 2.5 times the amount of parameters. Therefore we dropped down the BLSTM layer from a size of 256 to 128 which had less parameters and could actually run.

Results

As previously mentioned in our overview our best network based on accuracy was our LSTM with an accuracy of 55.22%. However, comparing confusion matrices (fig 5) and learning curves (Fig 2, 4) they are very similar and thus requires one to ask the question which is actually the "best" network considering that the accuracies are within a tenth of a percent from another. Looking at the total parameters the LSTM was nearly 850,762 parameters and took 2127 seconds, or 35.45 minutes to train. On the other hand the BLSTM was able to achieve a similar accuracy with only 588,618 parameters and took only 1810 seconds, or 30.17 minutes to train. With this in mind we air on the side of the BLSTM which was faster, simpler, and was more efficient with memory usage.

We took this understanding that BLSTMs were the more robust of the two networks with the understanding that more parameters generally means higher accuracy as the network learns more about the data and created the bonus network which is a BLSTM but has more parameters. Given the goal of separating the instruments into the three different groups of accoustic, electronic, and synthetic. Provided this task and learned knowledge our BSLTM was able to achieve an accuracy of 66.23%.

Discussion

The largest conclusion we came to was the impressive robustness that BLSTMs offer compared to LSTMs considering the comparable accuracies and memory usage. That being said, if we had the setup and time to test the differences between the same layer sized LSTM vs BLSTM we would likely see a higher accuracy from the BLSTM. This can be seen in our results from the bonus network, which was a BLSTM that contained more parameters.

Compared to the CNN networks found in project 1, the analysis becomes more nuanced. On the on hand the CNNs took significantly less time to train, nearly 50% less time at 15 minutes compared to the LSTMs which generally took around 30 minutes to train. However, the CNNs had over a million parameters, 1,135,338, while our best LSTM network had nearly half of that with only 588,618. Each able to achieve comparable accuracies. In our specific use case of students with tight deadlines the speed of the CNN was helpful but for larger projects where memory constraints might be a higher concern might look more toward the LSTM family of networks.

We discussed at length in our first report about the waveforms and will only give a brief synopsis in this report as it is more of the same. Due to the size and make up of the NSynth dataset, certain instruments are just represented more and are therefore learned more. Biggest example is Bass instruments which both sets of networks learned really well. It can be seen in the Confusion Matrices and various waveforms that the LSTM networks appear to have less confusoin between Guitar, Keyboard, and Mallet but the overall accuracy does not reflect this specific finding.

In conclusion depending on which you prioritize, speed or memory usage will persuade your decision on which network to use for a given problem set. Besides the specific edge cases mentioned, no specific network proved to be significantly better or worse than the other.

Figures

LSTM Network Construction

```
my_model.add(tf.keras.layers.LSTM(256, input_shape=train.shape[1:], return_sequences=True))
my_model.add(tf.keras.layers.Dropout(rate=0.2))
my_model.add(tf.keras.layers.LSTM(256))
my_model.add(tf.keras.layers.Dense(128, activation='relu'))
my_model.add(tf.keras.layers.Dropout(rate=0.2))
my_model.add(tf.keras.layers.Dense(64, activation='relu'))
my_model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

BLSTM Network Construction

```
my_model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128, return_sequences=True),
input_shape=train.shape[1:]))
my_model.add(tf.keras.layers.Dropout(rate=0.2))
my_model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128)))
my_model.add(tf.keras.layers.Dense(128, activation='relu'))
my_model.add(tf.keras.layers.Dropout(rate=0.2))
my_model.add(tf.keras.layers.Dense(64, activation='relu'))
my_model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

BLSTM Bonus Network Construction

```
my_model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128, return_sequences=True),
input_shape=train.shape[1:]))
my_model.add(tf.keras.layers.Dropout(rate=0.2))
my_model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128)))
my_model.add(tf.keras.layers.Dense(256, activation='relu'))
my_model.add(tf.keras.layers.Dropout(rate=0.2))
my_model.add(tf.keras.layers.Dense(64, activation='relu'))
my_model.add(tf.keras.layers.Dense(3, activation='softmax'))
```

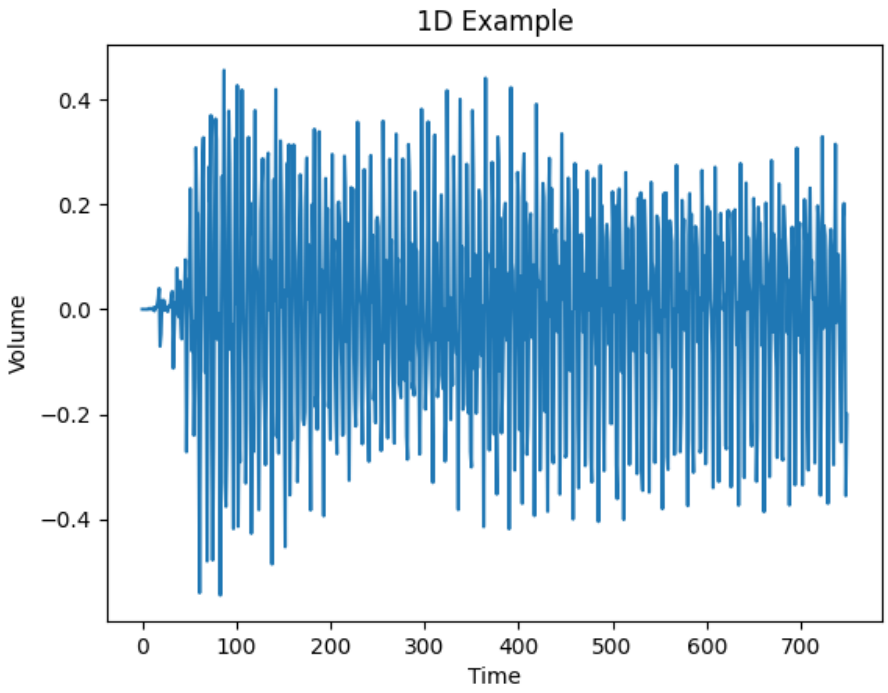
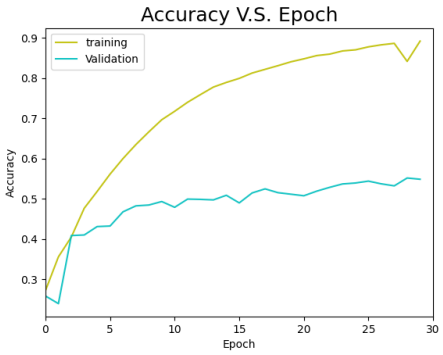
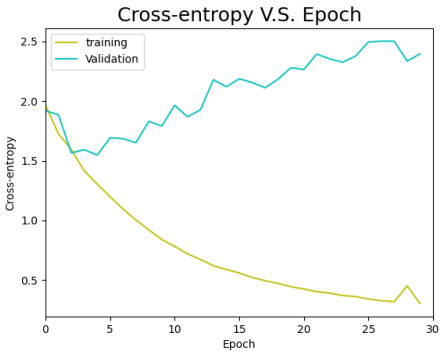


Figure 1: 1-D audio waveform

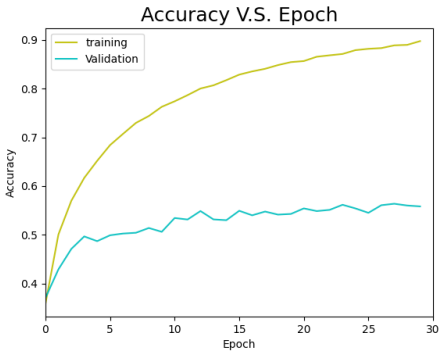


(a) Learning Curve

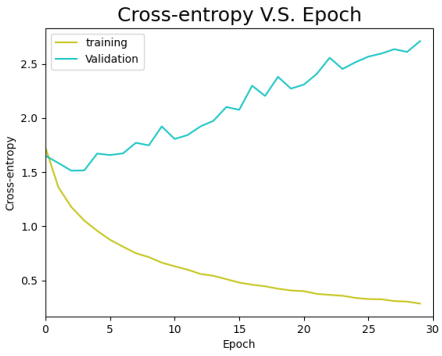


(b) Cross Entropy

Figure 2: Learning Curves Network 1 LSTM

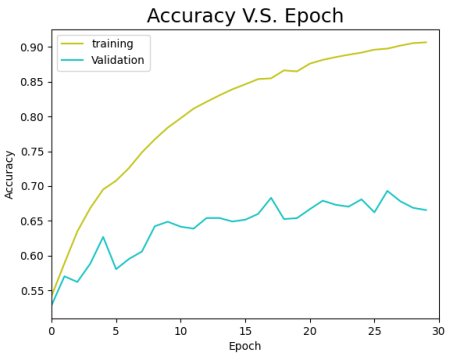


(a) Learning Curve

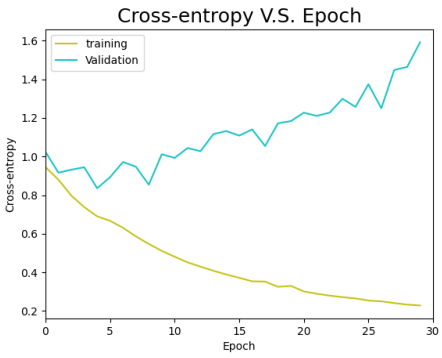


(b) Cross Entropy

Figure 3: Learning Curves Network 2 BLSTM

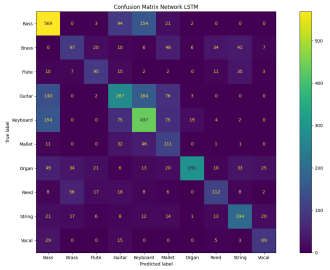


(a) Learning Curve

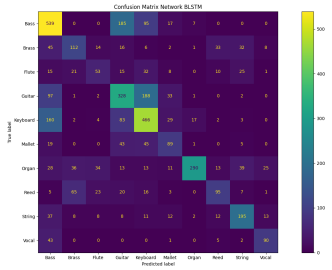


(b) Cross Entropy

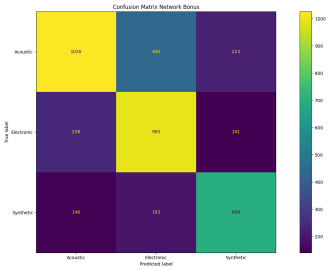
Figure 4: Learning Curves Network 2 BLSTM



(a) Confusion Matrix Network 1 (LSTM)

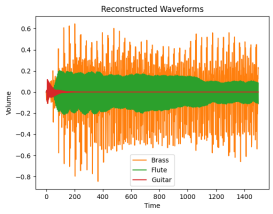


(b) Confusion Matrix Network 2 (BLSTM)

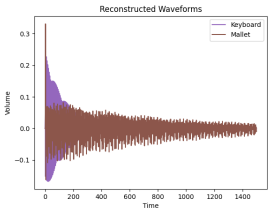


(c) Confusion Matrix Bonus Network

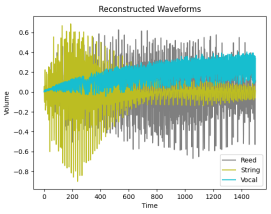
Figure 5: Confusion Matrices



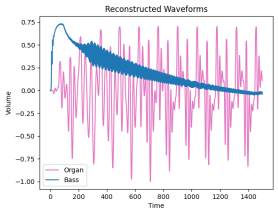
(a) Group 1



(b) Group 2

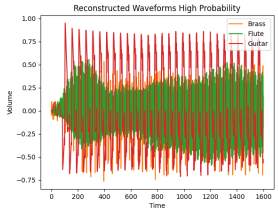


(c) Group 3

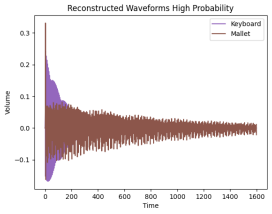


(d) Group 4

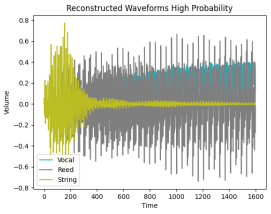
Figure 6: Waveforms of each class



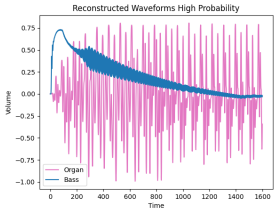
(a) Group 1



(b) Group 2

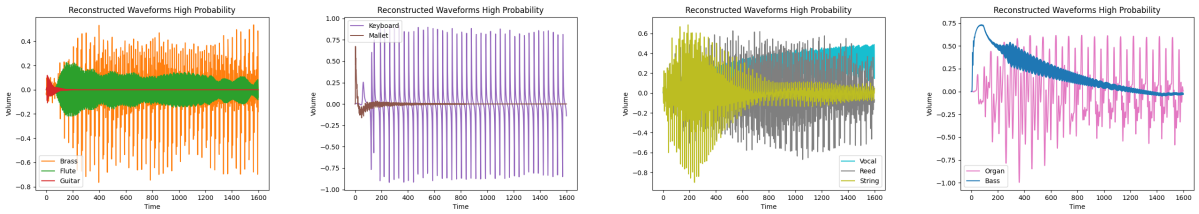


(c) Group 3



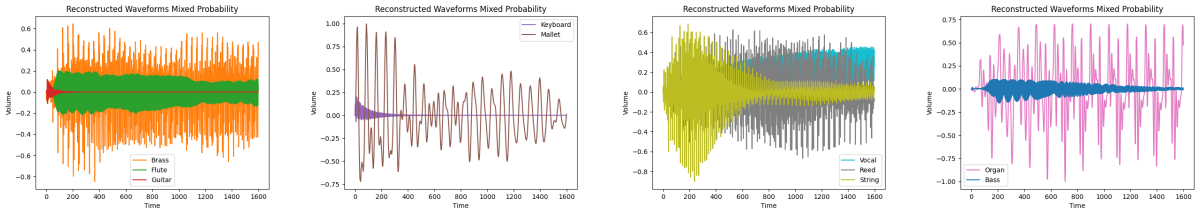
(d) Group 4

Figure 7: Waveforms with a high probability network 1 LSTM



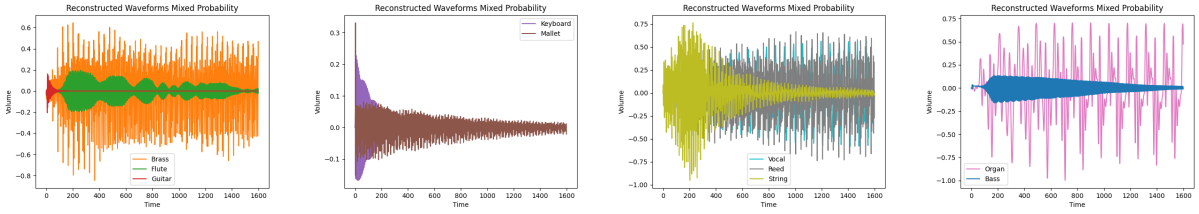
(a) Group 1 (b) Group 2 (c) Group 3 (d) Group 4

Figure 8: Waveforms with a high probability network 2 BLSTM



(a) Group 1 (b) Group 2 (c) Group 3 (d) Group 4

Figure 9: Waveforms with a mixed probability network 1 LSTM



(a) Group 1 (b) Group 2 (c) Group 3 (d) Group 4

Figure 10: Waveforms with a mixed probability network 2 BLSTM

References

[1] J. Engel, "The nsynth dataset," Apr 2017. [Online]. Available: https://magenta.tensorflow.org/datasets/nsynth

[2] T. Keras, "Keras documentation: Lstm layer," 2022. [Online]. Available: https://keras.io/api/layers/recurrent_layers/lstm/

[3] T. TensorFlow, "recurrent neural networks with keras tensorflow core 2022," 2022. [Online]. Available: https://www.tensorflow.org/guide/keras/rnn

[4] "Music genre classification using lstm." [Online]. Available: https://www.servomagazine.com/magazine/article/music-genre-classification-using-lstm

[5] C. Versloot, "Machine learning articles," 2022. [Online]. Available: https://github.com/christianversloot/machine-learning-articles/blob/main/bidirectional-lstms-with-tensorflow-and-keras.md