1. Overview

The networks used this time are all Recurrent Neural Networks.

LSTM → The model consists of a single multi layer Long Short Term Memory with **2** layers. The input size is **160**. The sequence length is **100**. The hidden size is **128**. The batch size is **768**. The classes to classify are **10** for the 10 instrument families. The input and output channels are as follows, LSTM → (160, 128, 2) Fc1 → (128, 10)

BLSTM→ The model consists of a single multi layer Bidirectional Long Short Term Memory with **2** layers. The input size is **160**. The sequence length is **100**. The hidden size is **128**. The batch size is **768**. The classes to classify are **10** for the 10 instrument families. The input and output channels are as follows, BLSTM → (160, 128, 2) Fc1 → (128, 10)

Both networks use the same input size and sequence length as specified in the project statement. The size of the input data is **16,000** since we are looking at only the 1st second of the waveform. This size allows the network to be trained in a reasonable amount of time while looking at the features of one second of the waveform across families.

Findings → RNNs are best suited to data which occurs in a sequence, such as the data we have at hand which is sound data.

→ LSTMs and BLSTMs train faster than CNNs in general.

→ LSTMs train faster relative to BLSTMs as they are not bidirectional, hence a two way computational overhead is eliminated.

→ The accuracy of the LSTM is higher than BLSTM. (More in discussion)

→ Batch size affects the overall accuracy because the smaller the mini batches the more the weight updates. Very large batch size (2048) helped to train the model faster but at the cost of accuracy. An optimal trade-off between accuracy and batch size was to select 768 as the batch size.

→ Hidden size also affects training time and accuracy, as a very large number of hidden units slows down training (with respect to time) and also the model might learn as well due to issue of vanishing gradients.

2. Training

For both the networks, a batch size of 768 was used as this size utilized a significant portion of the memory and also sped up the training times more.

LSTM → The **optimizer** used was **Adam Optimizer**. The criterion is **Cross Entropy Loss (CLE).**

To train the LSTM, the learning rate was set to **0.003**, this learning rate performed well with the Advanced Net from project 1 and hence was a suitable candidate for this implementation as well. This learning rate performed well and the model decreased **training loss** per epoch as can be seen in the loss curve below.

BLSTM → The **optimizer** used is **Adam Optimizer**. The criterion is **Cross Entropy Loss** (**CLE**).

To train the BLSTM, the learning rate was set to **0.003**, same as the LSTM.

The main benefits (while training) with LSTMs and BLSTMs is that they can be trained faster as compared to CNNs. This was definitely seen in the training time per epochs. For our advanced net, we saw per epoch timings of 6:26 vs 3:02 for the BLSTM.

With LSTM the training time per epoch was even less with timings of 2:03 for a batch size of 768.

With 283704 examples in the training dataset and a batch size of 768, the number of iterations per epoch were 283704/768 = 369.40625 → 369 (since last batch is dropped).

The waveforms presented below are for frame size of 160. Looking at the waveforms for size 160 rather than 16,000 helps to see if the patterns at this size help the model learn better or not.

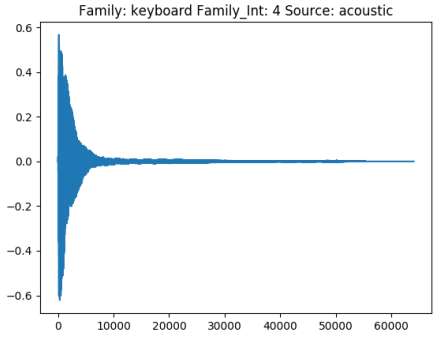
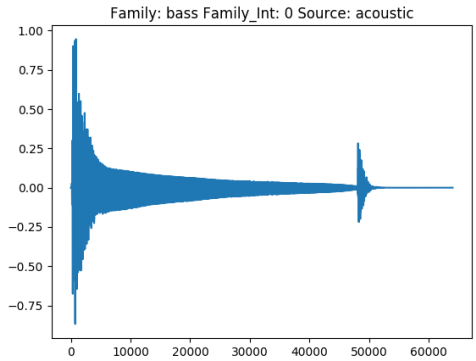
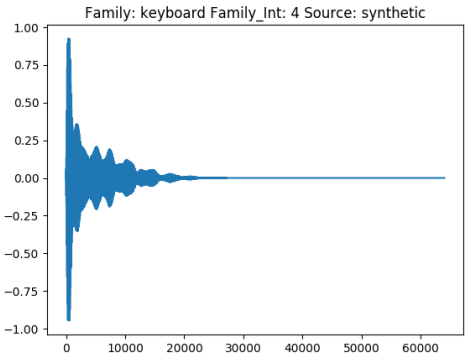
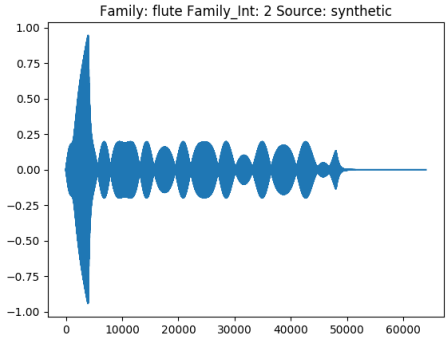
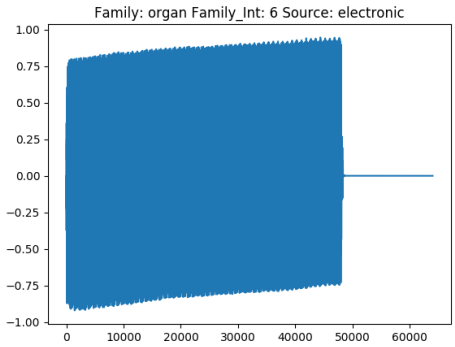
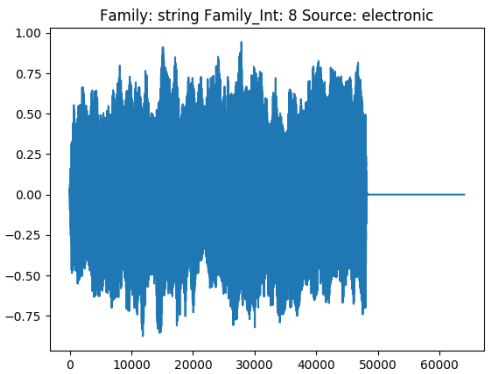
```
Epoch [3/30], Step [100/369], Loss: 1.5675
Training Accuracy : 41 %
Epoch [3/30], Step [200/369], Loss: 1.5286
Training Accuracy : 43 %
Epoch [3/30], Step [300/369], Loss: 1.4258
Training Accuracy : 44 %
Epoch 3, validation_loss: 1.4782496690750122
Validation Accuracy : 44 %
```

Monitoring progress in an epoch

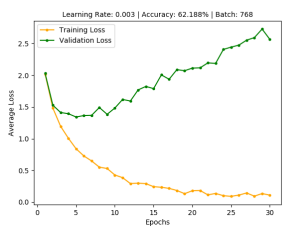
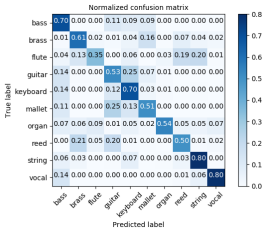
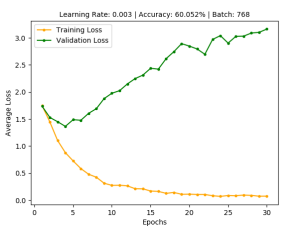
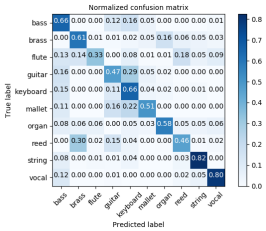
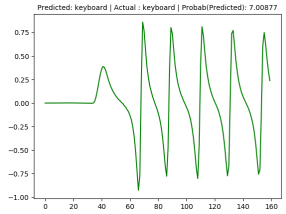
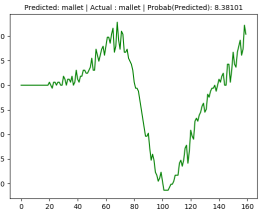
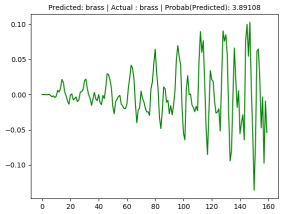
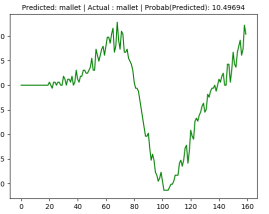
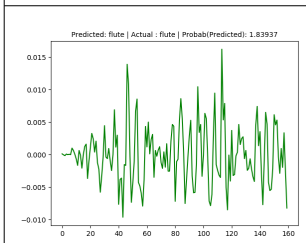
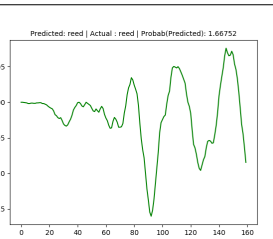
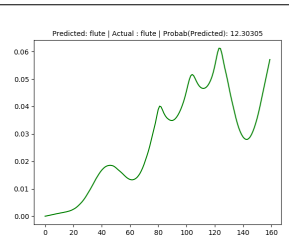
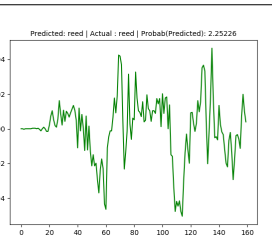
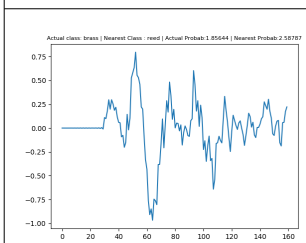
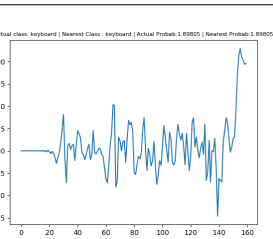
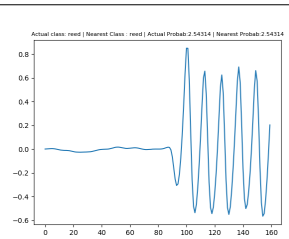
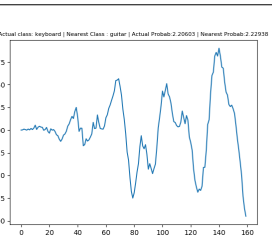
3. Results

Waveform examples across 3 sources and any 2 instrument families -

(Waveforms shown without transforms picked directly from train dataset)

Acoustic Source		
Synthetic Curves		
Electronic Curves		

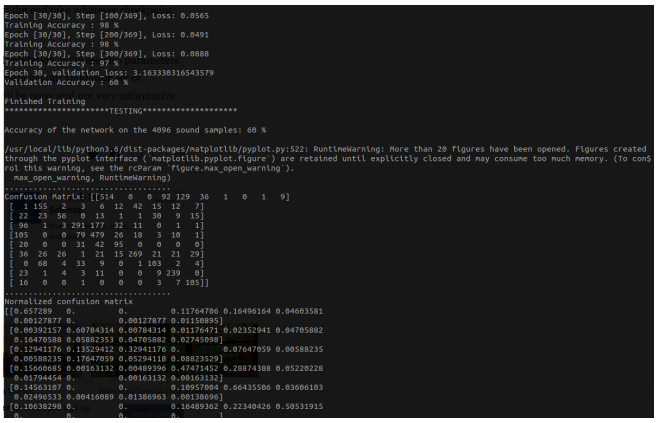
Accuracy results included in the Loss curves.

LSTM		**BLSTM**	
			
Loss Curve	Confusion Matrix	Loss Curve	Confusion Matrix
Waveform for samples where correct class probability is very high. (160 input size)			
			
			
Waveform for samples near decision boundary. (160 input size)			
			

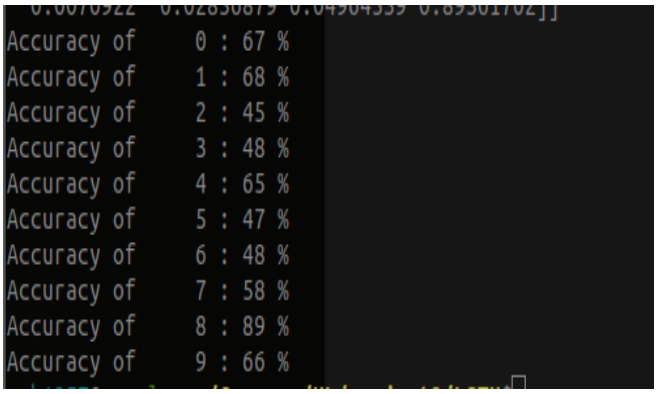
LSTM	BLSTM

Please zoom to view

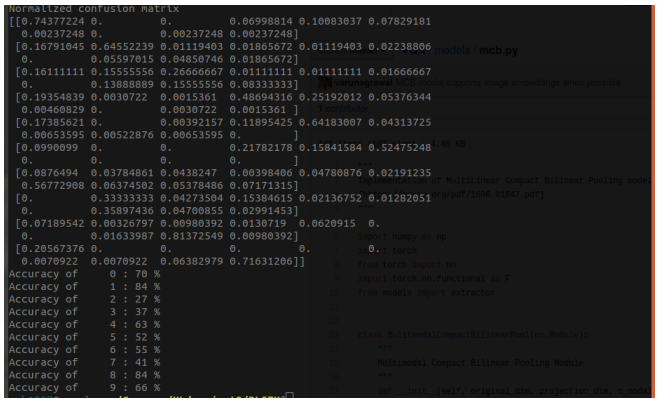
Overall Accuracy



Overall Accuracy



Per Class Accuracy



Per Class Accuracy

4. Discussion

Using the Adam optimizer [2] which varies the learning rate with the changes in parameters the curve is more gradual and reflective of expected training behavior with the training loss decreasing after each epoch. However the validation loss decreases at the beginning but then increases. This suggests that the model is overfitting to the train data.

From the confusion matrix, it can seen that both the LSTM and BLSTM are getting confused at similar classes. For example in the row for bass both the LSTM and BLSTM are confusion it for class guitar and keyboard.

In terms of speed and accuracy, as discussed above in the training section, both these models training much faster and also improved upon the accuracy from project1. For project1 the reported accuracy was 57% and with

the these 2 models we are seeing an accuracy of 62% which is a gain of 5%. The per epoch timings were about half of what we saw for advanced net from project 1.

Weight initialization used is random weight initialization. As with our advanced model, on subsequent runs our LSTM and BLSTM model wasn't being thrown of too much in terms of our accuracy, whereas in our advanced model there was a variability of about 5% on subsequent runs. Hence a Xavier weight initialization as with our advanced model didn't seem necessary. [4]

For a run of Epochs → 30 | Learning Rate → 0.003 | Batch Size → 768 | Hidden Units → 128 | Input Size → 100 | Sequence length → 160

The final accuracies were

LSTM → 62 %
BLSTM → 60 %

Finally, dropout layers with a p of 0.5 (drops 50% of the nodes), were added to perform regularization to prevent overfitting and penalizes the loss function but these caused the model not to learn at all and was getting stuck in a local minima (judged by the loss curves). [1] Hence, dropout was later removed and the accuracies presented here are for runs without dropout.

In short both LSTM and BLSTM performed similarly in terms of accuracies, per class accuracies and decision boundaries. This can be seen in the confusion matricies, loss curves and waveform visualizations seen above.

Again it can be seen that the "memory" aspect of RNN helps them to learn the sequence data better. However, the accuracies are still low and this raises questions towards the quality of the dataset which also affected the accuracies in project 1. [5]

Data → Analyzing a few waveforms from the different families(refer per family per source waveforms above (more were analyzed)), a few waveforms look similar to the waveforms from other families.[3] This might be a reason as to why the model confuses one family with another as can be seen with the case of guitar in the confusion matrix. It is difficult to say exactly how many instances of such waveforms are present in the dataset without going through it entirely. The distribution of data is also uneven which dips the per class accuracy.

With the data being "noisy" with waveforms similar across families, makes the models job difficult to differentiate between instrument families and hence the overall accuracy is relatively low.

Since softmax was not used for the fully connected layer, the "probabilities" are at a different metric and not between 0 and 1 per class. But the logic to derive the highest probability remains the same as project 1.

6. References

[1] Galeone, P. (2017). *Analysis of Dropout*. [online] P. Galeone's blog. Available at: https://pgaleone.eu/deep-learning/regularization/2017/01/10/anaysis-of-dropout/.

[2] Kingma, D. and Ba, J. (2014). *Adam: A Method for Stochastic Optimization*. [online] Arxiv.org. Available at: https://arxiv.org/abs/1412.6980.

[3] Shuvaev, S., Koulakov, A. and Giaffar, H. (2017). *Representations of Sound in Deep Learning of Audio Features from Music*. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1712.02898.pdf.

[4] Hochreiter Sepp, Koulakov, A.(1997). *Long Short Term Memory*. [online] Available at: http://www.bioinf.jku.at/publications/older/2604.pdf

[5] Yin, W., Kann, K., Yu, M. and Schütze, H. (2017). *Comparative Study of CNN and RNN for Natural Language Processing*. [online] Arxiv.org. Available at: https://arxiv.org/abs/1702.01923