Here I have listed all models that I have tried on the given dataset. There is a table at the end consisting of the model’s train loss and accuracy and validation loss and accuracy.

1. **~~Dataset~~ Input Representations**  
   2 types of datasets were used
   1. Standardized dataset (SD) - Here each pitch contour was standardized to have mean = 0 and standard deviation = 1
   2. Normalized dataset (ND) - Here, each pitch contour was normalised to occur between 0 and 1.

The test and validation dataset refer to the same dataset and are used interchangably in this document.

1. **Model training**:  
   The model weights from the epoch with lowest validation loss was stored and used to report the model metrics.
   1. Each model architecture is assigned an experiment name. With each model architecture, there can be several models generated by using a different dataset (SD or ND), training with a different batch size, optimizer, number of epochs etc.

**Models**

Below, I have listed model architectures used. Each architecture can use different input representations, training hyperparameters such as batch size, epochs, learning rate, optimizer or learning rate schedules etc. These are listed as sublists beneath every model architecture

1. Test
   1. Model\_3 (this is the model name, on my system – maintained for reference purposes)  
      This model architecture is an exact replica of the model presented in [1].
      1. **Training params**: This model was trained with a batch size of 1, 50 epochs and SGD optimizer (Stochastic Gradient Descent). All layers use an l2 regularizer with lambda = 0.00025. All pool layers used average pooling and activation of all conv and dense layers were sigmoid. The Standard dataset was used as used in the paper.

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) (None, 1200, 1) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_1 (Conv1D) (None, 1196, 8) 48

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

average\_pooling1d\_1 (Average (None, 598, 8) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_2 (Conv1D) (None, 594, 4) 164

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

average\_pooling1d\_2 (Average (None, 297, 4) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 1188) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

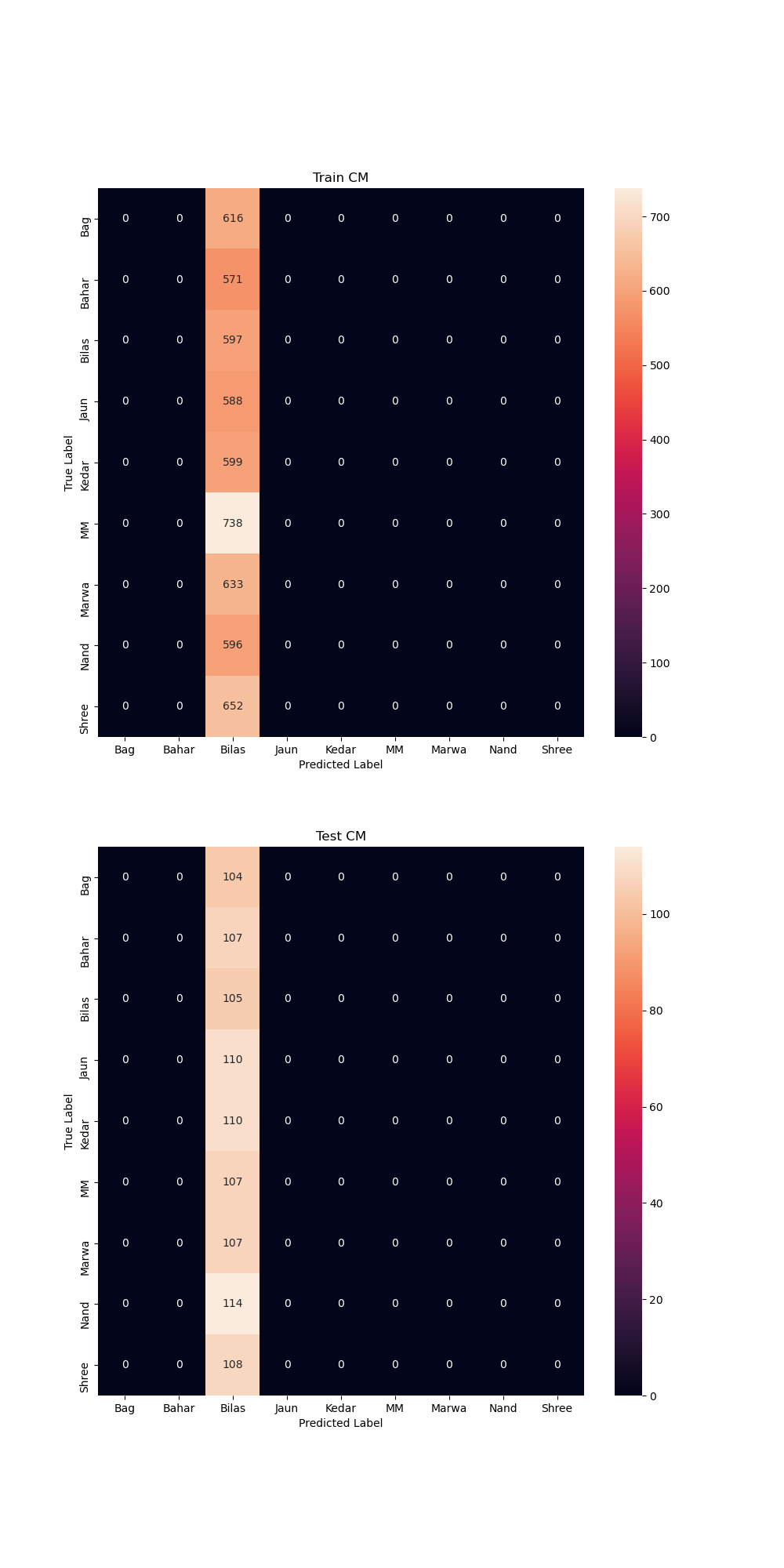
dense\_1 (Dense) (None, 732) 870348

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

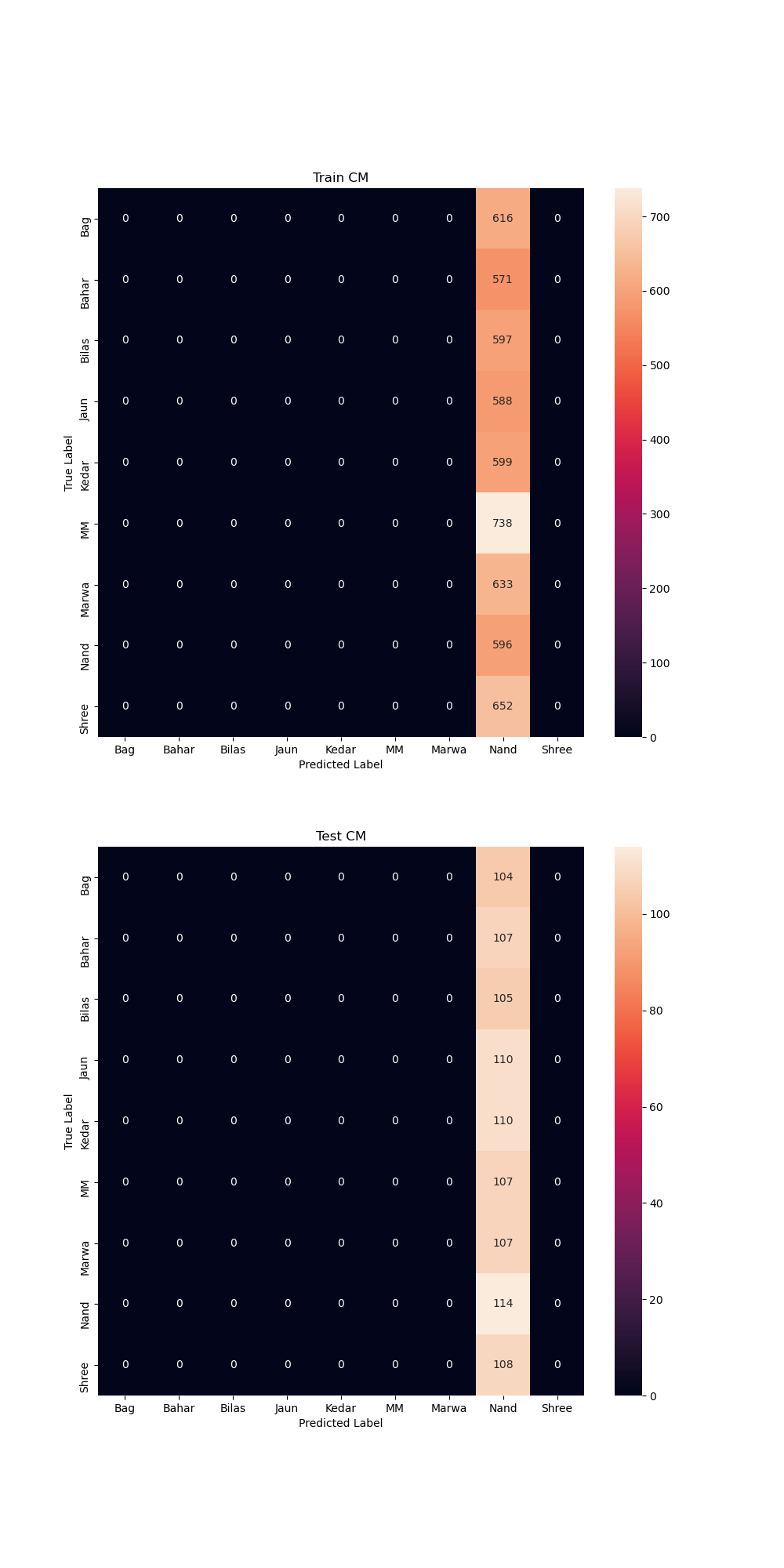
dense\_2 (Dense) (None, 9) 6597

=================================================================

* + 1. **Results:** This model has a train accuracy of 11.8% and a test accuracy of 10.8% implying that there is a random guess occuring by the model. This is evident from the confusion matrices of the predictions of the model on the train and test set. In the CM it is clear that the model is just guessing bilaskhani todi as the output for all inputs in the train and test set.

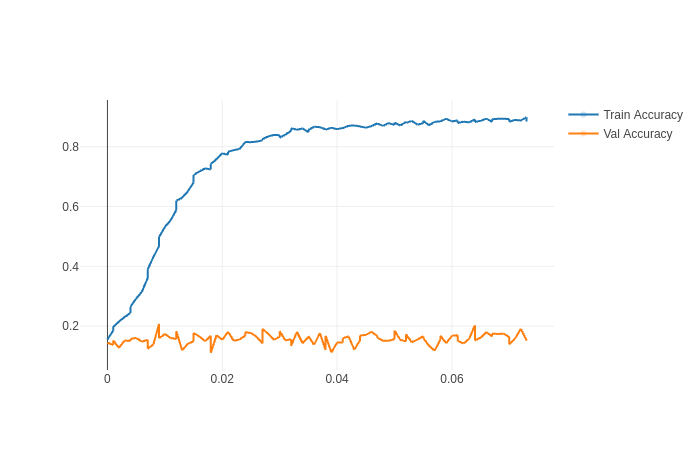
Figure 2: CM for model\_3, test

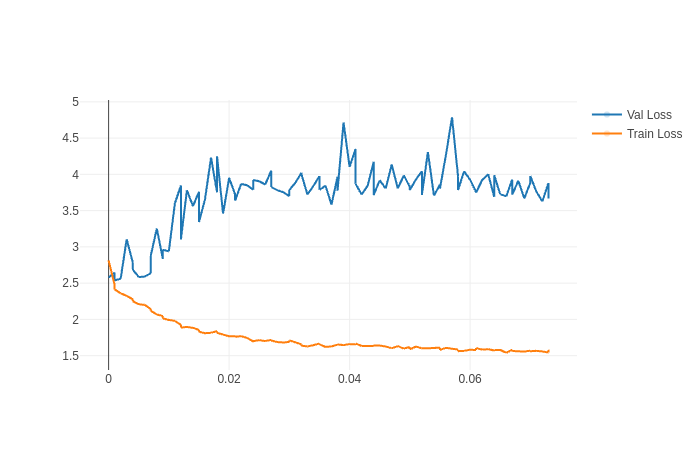
1. Zhang2015
   1. Model\_0  
      [2] is a follow up paper to [1]. In this paper they also explore the use of max pool and relu activation (as opposed to avg pool and sigmoid used previously) and deem it to be better. Hence this is the same architecture but with max pool used for all pooling layers and relu used as activation of convolution and dense layers.
      1. **Training Params (TP)–**  SGD was used with a batch size of 1 for 50 epochs.
      2. **Results –** Here also, the train accuracy is 11.8% and test accuracy is 10.7%. As is evident from the CM, only raag Nand is being predicted for the train and test data.
2. Zhang2015-mod1

Figure 3: CM for model\_0, Zhang2015

* 1. Model\_0  
     Given that our input is much longer than the input used in [2] (which was 3 time series with length 256), the length of our time series is 1200. To account for this I tried to double the number of filters (to capture possibly more patterns from a longer sequence). After each conv layer, batch normalization was added after which relu was used. Dropout layers were also added after the flatten layer and the dense layer. The size of the dense layer was also increased from 732 to 1000.  
     1. **TP:**  Adam optimizer was used along with a batch size of 32 was used with training for 100 epochs. The dataset used was ND. Following is the architecture.
     2. **Results –** This model goth a train accuracy of 19.8% and Val accuracy of 15.2%. However, through the training procedure, the model very strongly overfitted on the train set as is visible from the loss and accuracy plots below. The train accuracy ended with 88.4% and validation accuracy ended with 15.1% on the 100th epoch. However these values were not recorded as the train and val accuracy since the validation loss was very high for this epoch, and we record model metrics from the epoch with the lowest validation loss. The model metrics were recorded from epoch #2 during the training.

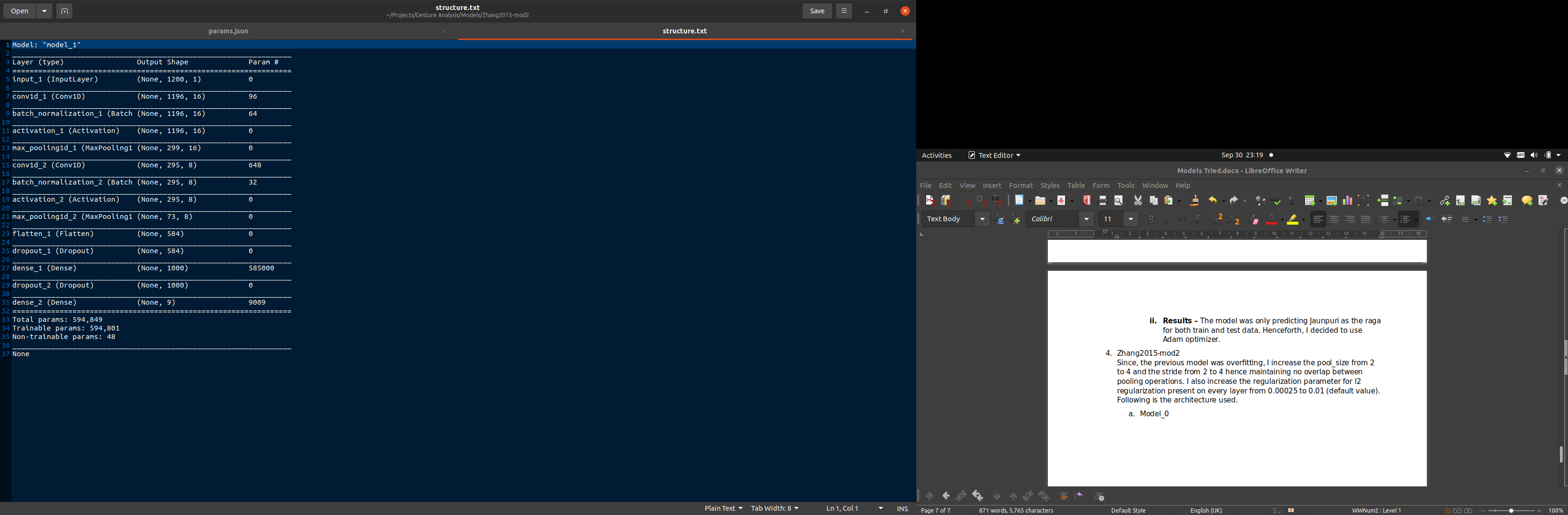
The model here very clearly has overfitted on the training set.

Figure 5: Train and val accuracy from zhang2015-mod1, model\_0

Figure 6: Train and val loss for Zhang2015-mod1, model\_0

* 1. Model\_1
     1. **TP –** With the same architecture, SGD was used with a batch size of 1, learning rate of 0.01 and momentum of 0.9 (as in Zhang2015 and test models), same optimizer and training parameters used in [2].
     2. **Results –** The model was only predicting Jaunpuri as the raga for both train and test data. Henceforth, I decided to use Adam optimizer.

1. Zhang2015-mod2  
   Since, the previous model was overfitting, I increase the pool\_size from 2 to 4 and the stride from 2 to 4 hence maintaining no overlap between pooling operations. I also increase the regularization parameter for l2 regularization present on every layer from 0.00025 to 0.01 (default value). Following is the architecture used.
   1. Model\_0

Figure 7: Model architecture used for Zhang2015-mod2

* + 1. **TP –** Adam was used with a batch size of 32, for 100 epochs. ND was used.
    2. **Results –** model metrics were recorded at epoch #11, because the validation loss is very noisy. The train accuracy is 22.8%and test accuracy 19.4%. This implies that the model has learnt more than in the previous experiments.

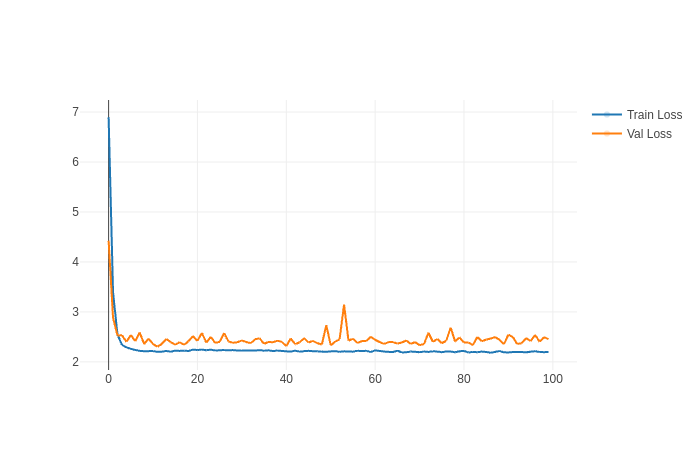
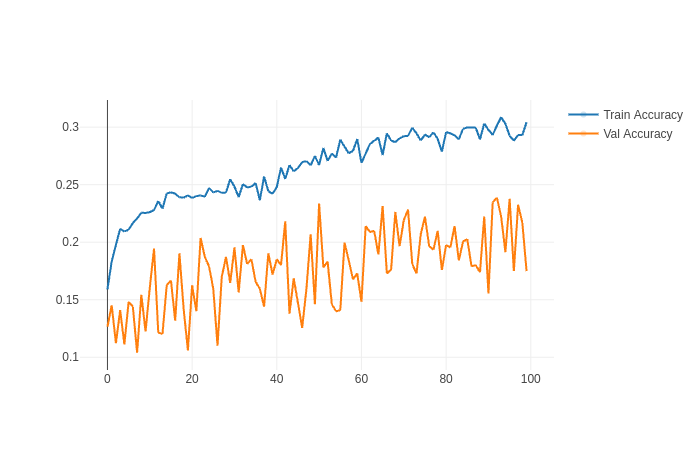
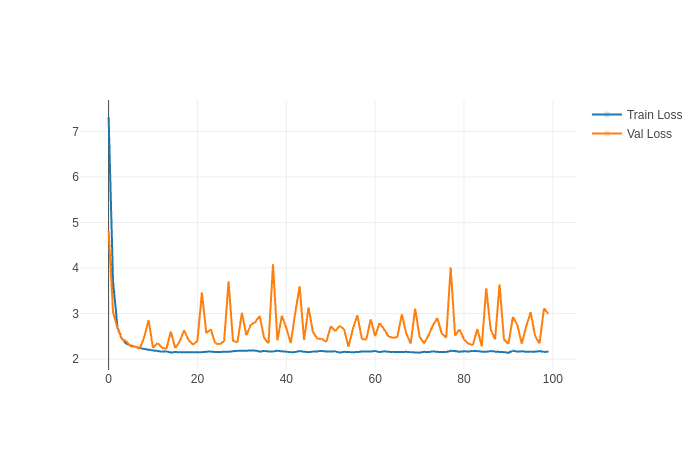
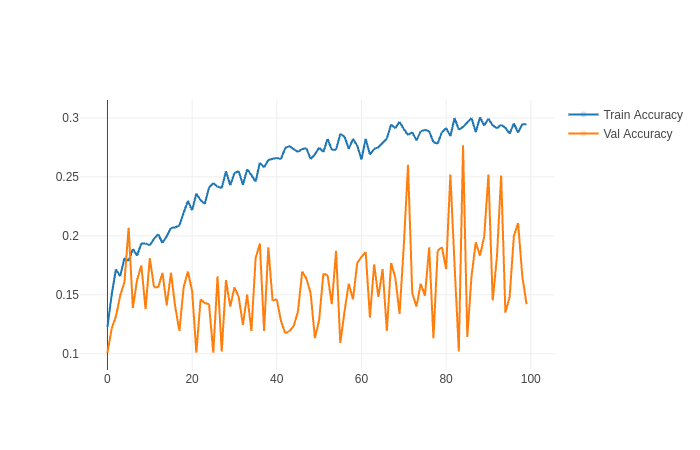


Figure 8: Loss plot for Zhang2015-mod2, model\_0

Figure 10: Accuracy plot for Zhang2015-mod2, model\_0

* 1. Model\_1
     1. **TPA –** All parameters are same as model\_0 but the SD dataset was used
     2. **Results –** model metrics were noted from epoch #13, and are similar to the results of model\_0, with a test acc of 16.9% and train accuracy of 19.4.

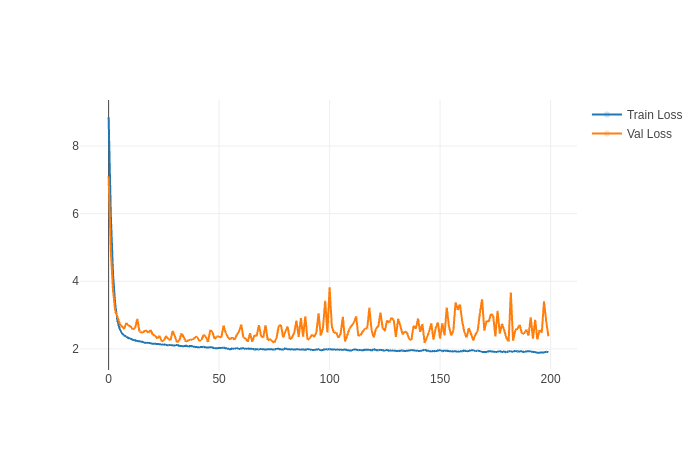
Figure 11: Loss plot for Zhang2015-mod2, Model\_1

Figure 12: Accuracy plot for Zhang2015-mod2, model\_1

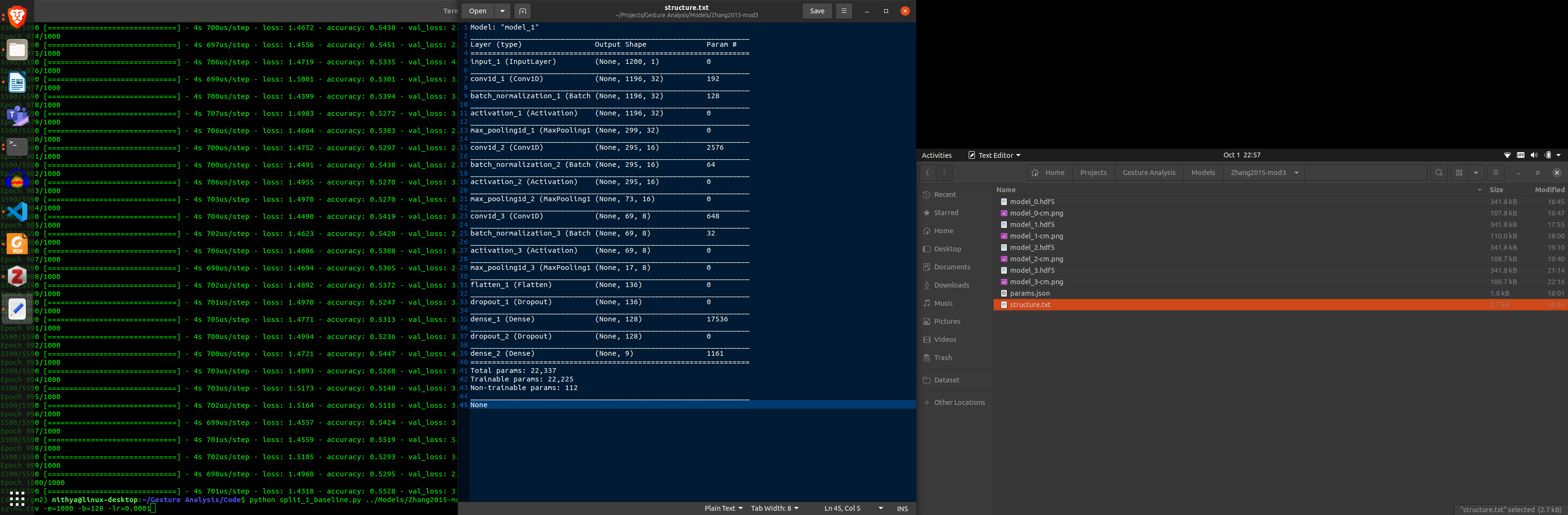
The validation loss in both model\_1 are more noisy than in model\_0. In addition model\_0 has performed slightly better in terms of validation accuracy and loss.

* 1. Model\_4  
     I tried increasing the batch size so that weight updates would happen over a more generalized error, and probably make the training less noisy.
     1. **TPA –** SD dataset was used, along with a batch size of 128, with a learning rate of 0.001 and Adam over 200 epochs.
     2. **Results –** The model train accuracy was 37.4% and test accuracy was 27.5% which is significantly better than previous models. Model parameters were stored from the 143/200 epochs. However the validation loss plot is still a little noisy

1. Zhang2015-mod3  
   I doubled the number of conv filters in the conv layers and added another conv layer after it. Below is a summary of the model architecture.

Figure 13: Loss plot of Zhang2015-mod2, model\_4

It seems like the validation set is performing very badly with this model throughout. The loss is fluctuating a lot. So it seems like the model is not able to generalize well. I think I will increase model complexity and try other regularization methods on the model.



* 1. Model\_2
     1. **TPA -**  Uses batch size of 128 and 1000 epochs to train, Adam optimizer with a learning rate of 0.001 (default)
     2. **Results –** The validation loss is still very noisy and doesn’t seem to settle at all. The model achieved a test acc of 38.9% and train acc of 50.8%.

Next I want to check if the validation dataset performs this badly across all three singers. All experiments until this point have been done on AG.

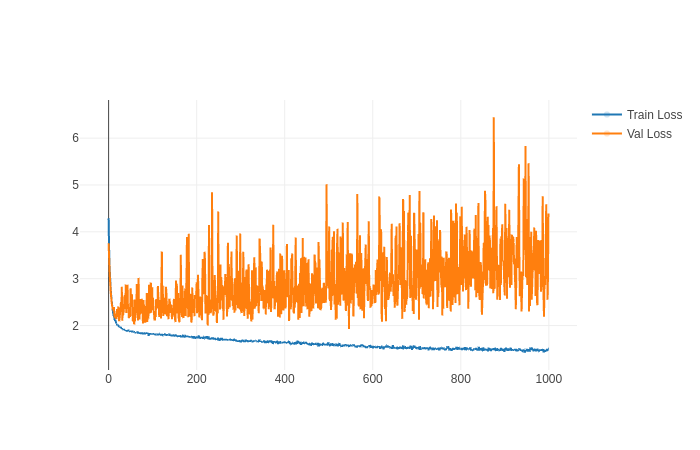
Figure 14: Loss plot for Zhang2015-mod3, model\_2

Table 1: Results from all models

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment Name | Model Name | Train Loss | Train Accuracy | Validation Loss | Validation Accuracy | Epoch with min val loss |
| test | Model\_3 | 2.22 | 11.9 | 2.21 | 10.8 | 10/50 |
| Zhang2015 | Model\_0 | 2.22 | 11.8 | 2.2 | 11.7 | 31/50 |
| Zhang2015-mod1 | Model\_0 | 2.22 | 11.4 | 2.2 | 11.3 | 16/100 |
| Zhang2015-mod2 | Model\_0 | 2.2 | 22.8 | 2.3 | 19.4 | 11/100 |
| Zhang2015-mod2 | Model\_1 | 2.17 | 19.4 | 2.22 | 16.9 | 13/100 |
| Zhang2015-mod2 | Model\_4 | 1.98 | 37.4 | 2.17 | 27.5 | 143/200 |
| Zhang2015-mod3 | Model\_2 | 1.55 | 50.8 | 1.93 | 38.9 | 546/1000 |