DELIVERABLE 3

Final Training Results:

Before presenting the final training results, the model architecture and initial hyperparameters are shown below in Figure 1 and Table 1 respectively.

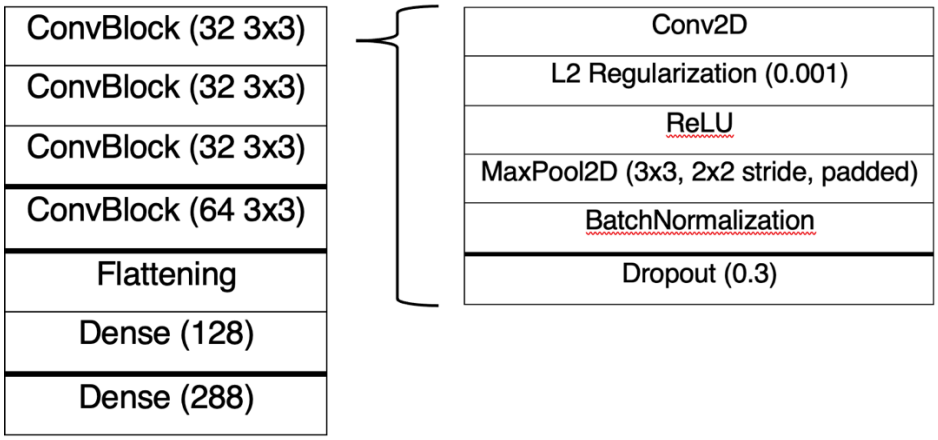


Figure 1: *CNN Model architecture*

Table 1: *Preliminary model hyperparameters*

Dropout Rate	L2 Regularization Weight	Adam Optimizer Learning Rate
0.3	0.001	0. 0001

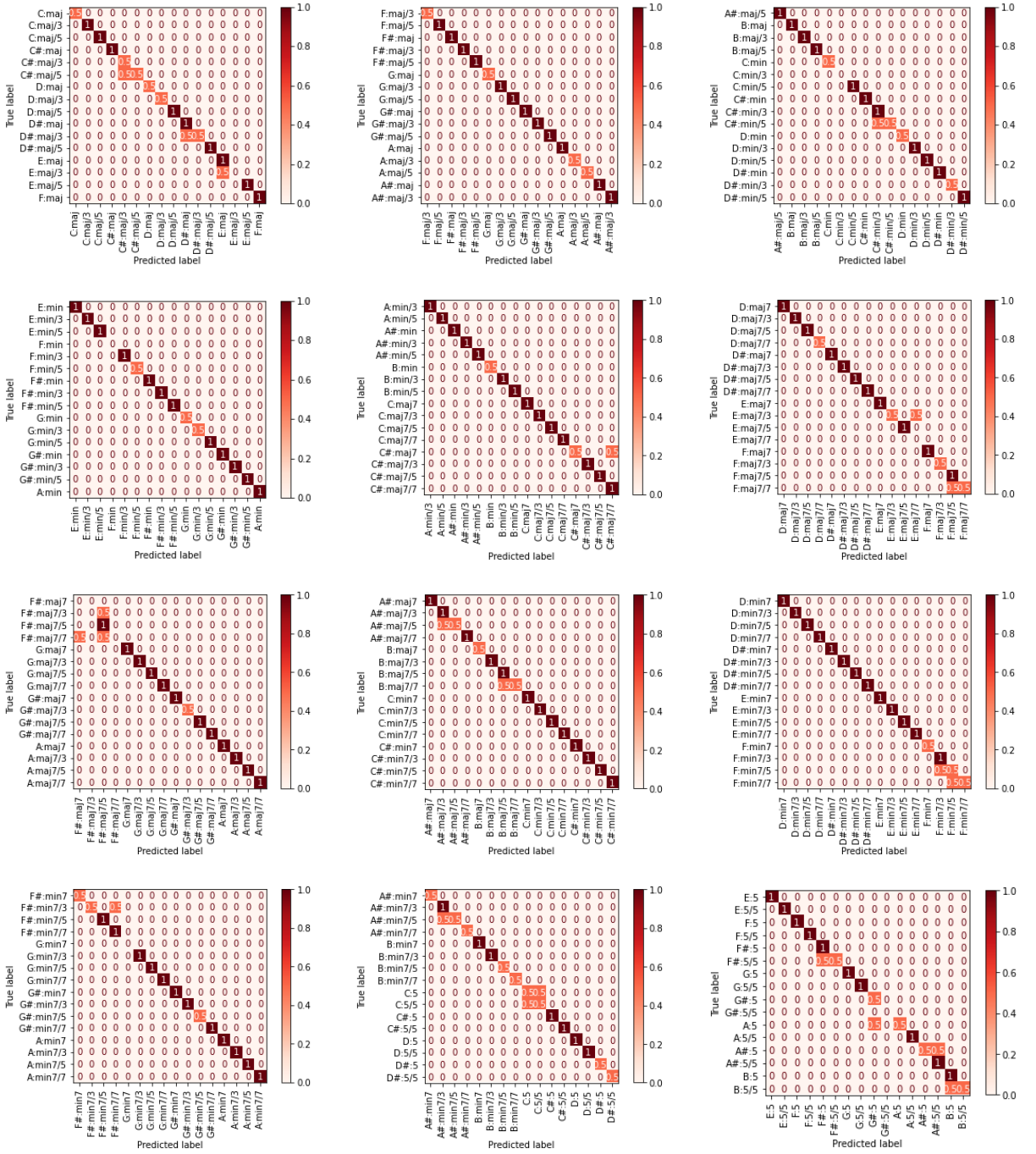
With this configuration, the initial model achieved 98% and 61% training and test accuracy respectively, clearly overfitting.

In an attempt to reduce overfitting, I adjusted the train, validation, and test splits to be 0.75, 0.15, and 0.1 respectively. This is an increase in size of the training set compared to the preliminary model. Furthermore, Bayesian optimization was performed to search for the optimal hyperparameters, with the target being to maximize the test accuracy. The search space for the hyperparameters was dynamic, allowing it to converge with each training iteration (effectively speeding up the search for the set hyperparameter to be used for the next training instance). After running this optimization for 10 iterations, the following hyperparameters, were found to be optimal (shown in Table 2).

Table 2: *Final model hyperparameters*

Dropout Rate	L2 Regularization Weight	Adam Optimizer Learning Rate
0.3404	0.002728	0.000296

With these hyperparameters, the model achieved 97% and 81% training and test accuracy respectively (with a spare categorical cross-entropy loss of 1.0501. Confusion matrices are shown below in Figure 2.



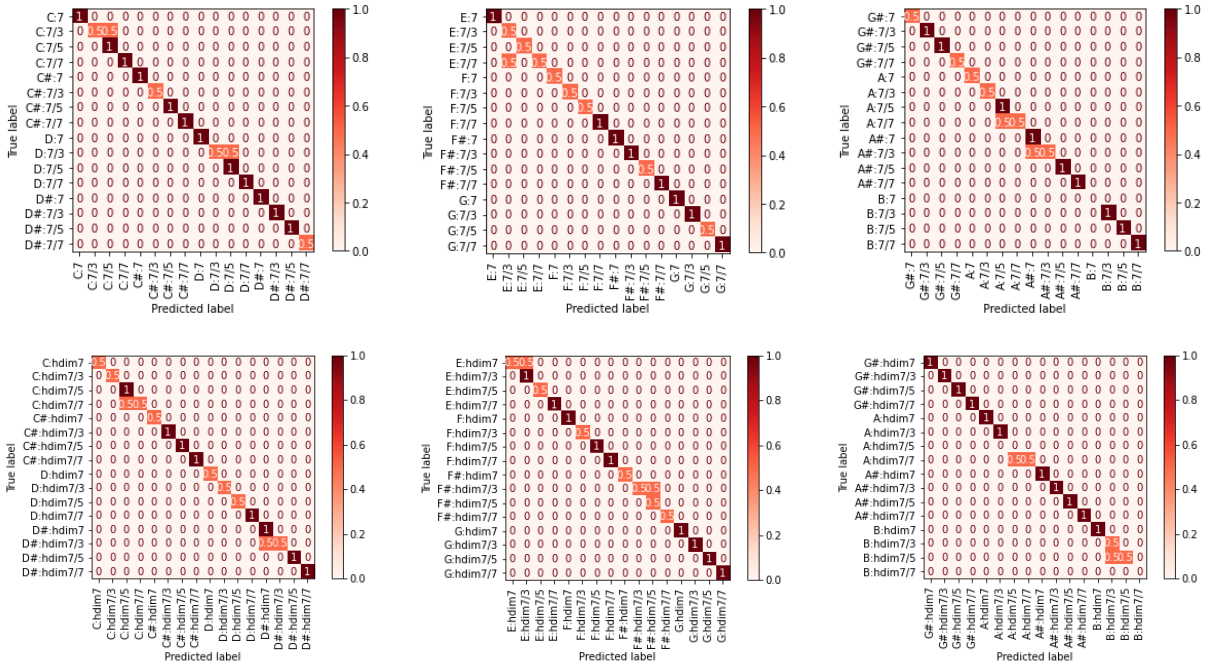


Figure 2: *Confusion matrices*

As can be seen from the confusion matrices, many of the mispredictions are cases of predicted the incorrect inversion of the *same* chord. This means that for the final integration, the prediction displayed to the user/client should only be for the chord and not for its specific inversion.

Final Demonstration Proposal:
Having no web development experience, I will be closely following the workshop and repository created by Youssef (TPM) and William (TPM). The vision for this landing page is a framework based around server-side rendering (as opposed to front-end) for the purpose of simplicity. The Flask library will be used build the landing page. I plan to host the landing page on either GitHub Pages or netlify.

For the user to get a prediction, they can either choose one of the sample chord recordings available on the landing page or they can record their own chord sample. This recording will be submitted to the server and a prediction will be requested. Based on the prediction, the server will send the chord name and guitar chord diagram to the client for the user to see. If time permits, I will look into using React instead of flask to potentially develop a mobile application.

Acknowledgements:
The improvements seen in the final model were made possible under the guidance of Zedian (TPM).