1. Overview

This project is an implementation of CNN using Keras to classify single note recordings of instruments into one of ten instrument family names with NSynth dataset [1]. The input dataset will go through a preprocessing process for reducing size. In the training process, two convolutional neural network model has been created, the first model is a simple structured (two convolutional layers) CNN developed in our preliminary approach, and the second model is a well-structured CNN optimized by empirical and theoretical approaches. The first model obtained testing accuracy of 33.69%, and the improved model increased the accuracy to 60.47%. More detailed information will be presented in the following sections. This project gives us a better understanding of the convolutional neural network structure; besides, we find that adding dropout [2] layers can effectively reduce overfitting for the second model.

2. Training

2.1. Preprocessing Data

The NSynth dataset is split into three sets: the training set with 289,205 examples, validation set with 12,678 examples, and testing set with 4,096 examples. Each example consists of 4 seconds long recording of the sound with 16kHz rate, which makes 64,000 soundwave values. The full dataset is with a size of around 73GB, which is difficult to run on a regular machine. To solve the memory issue, we processed the data through a size reduction program, which simply takes the mean value for every 100Hz of the original data (each down-sized soundwave has 640 values). After this process, the size of the dataset was reduced to 734MB.

2.2. First approach

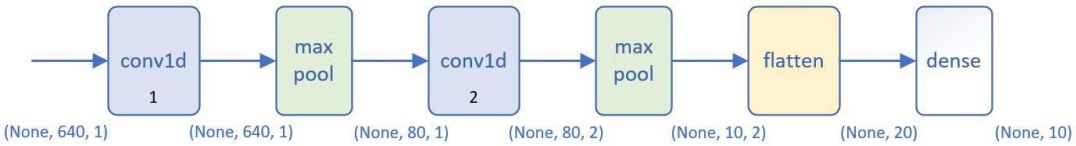

Figure 1. Structure graph of the first CNN model.

In our first approach, we created a CNN model with a simple structure. The structure of this CNN model can be seen in Figure 1. The first 1D CNN layer defines 1 filter with kernel size 10 and ReLU for activation function, which will output a 640 x 1 neuron matrix. The output will then be fed into a max-pooling layer to reduce the complexity of the output and prevent overfitting of the data [3]. The pool size for the max-pooling layer is 8, so the output of this layer will be at size 80 x 1. The second 1D CNN layer has 2 filters with kernel size 10 and ReLU for activation function, followed by a max-pooling layer with pool size 8. The output after those two layers will be at size 10 x 2. The output will be flattened to size 20 and fed into the fully connected layer with softmax activation function.

Categorical cross entropy was selected to be the loss function for this model. Categorical cross entropy is a loss function that is used for single label categorization (each example can only belong to one class) [4], which fits current problem nicely. Classification accuracy was selected as the metric to be evaluated by the model during training and testing. In addition, Adam optimizer was selected as the optimizer of the model. Adam optimizer is an adaptive learning rate optimization algorithm designed for deep learning and was proven to be effective (especially in terms of training speeds) empirically [5]. The model was trained for 60 epochs with batch size 1000. The detailed results can be seen in the following *Results* section.

2.3. Improved CNN model

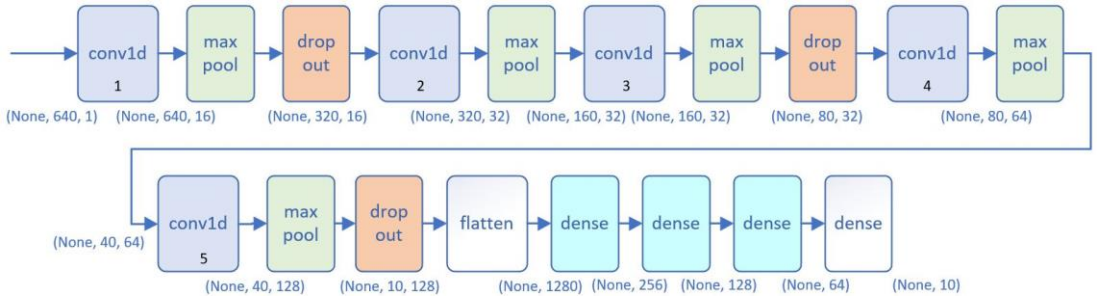

Figure 2. Structure graph of the improved CNN model.

The improved CNN model structure (showing in Figure 2) is developed empirically. There are 5 1D CNN layers created with 16, 32, 32, 64, and 128 filters, kernel size 10, and tanh as the activation function. To be mentioned, we are using tanh other than ReLU since it gives us better performance in experiments. The output of each 1D CNN layer will then be fed into a max-pooling layer with pool sizes 2, 2, 2, 2, and 4. Besides, there are three Dropout layers with rate 0.15, 0.2, and 0.2 created between those 1D CNN layers. Dropout will randomly set a fraction of units to zero at each training update, and thus prevents hidden units from relying on specific inputs [2]. To be mentioned, the first design of this CNN model doesn't include these Dropout layers. These layers were introduced when we detected overfitting issues. A more detailed explanation will be presented in the *Discussion* section below. The output of the last Dropout layer is at size 10x128. This output will be flattened to size 1280 and fed into the fully connected layers (three dense layers with units 256, 128, and 64 with activation function tanh. The last layer is a dense layer with units 10 and softmax activation function).
This model shares the same training configurations as the first one presented in section 2.2. The only change we made is changing loss function from categorical cross entropy to sparse categorical cross entropy, which is an integer-based categorical cross entropy loss function. This loss function doesn't require one-hot encoding of targets, and seems to work better in experiments (more experiment needed).

3. Results

Visualization of 1-D audio waveforms for each class can be seen in Figure 3.

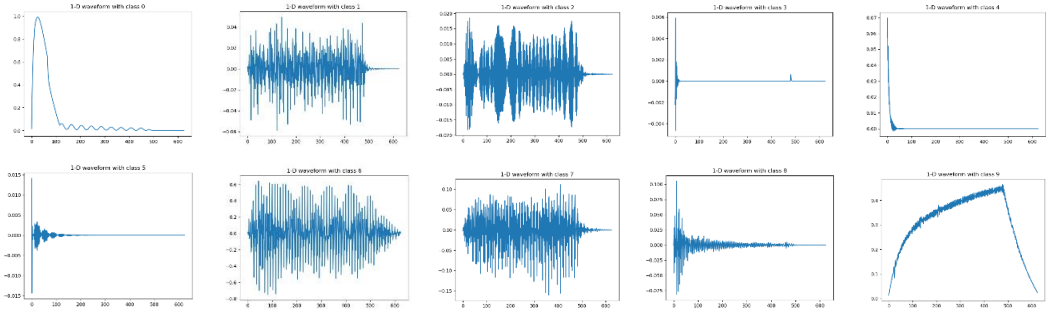

Figure 3. Visualization of 1-D audio waveform examples for each class.

3.1. First approach results

The test results for the first CNN model is showing below. The test accuracy is 33.69%.
The learning curves for training and validation data is showing in Figure 4.

The confusion matrix is showing in Figure 5. The waveform for samples where the correct class probability is very high, and very low for each class is showing in Figure 6 and 7. The waveform for samples near the decision boundary for each class is showing in Figure 8.

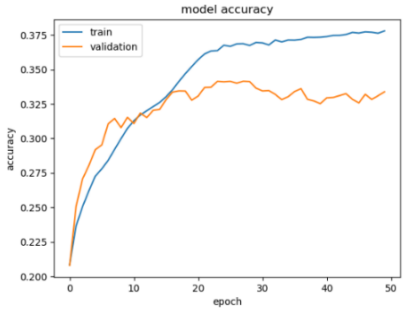

Figure 4. First CNN learning curves.

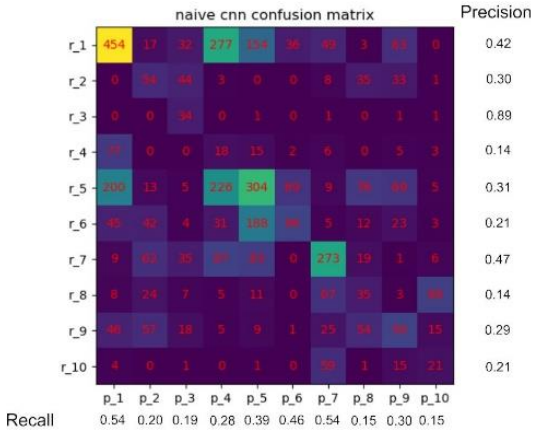

Figure 5. First CNN confusion matrix.

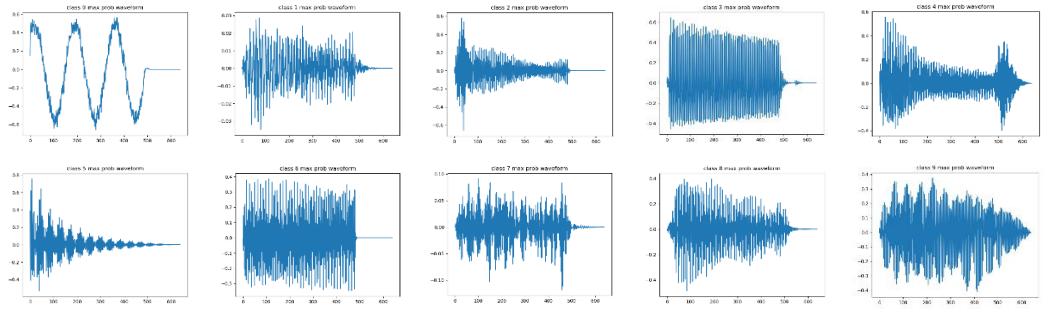

Figure 6. First CNN 1-D audio waveform where the correct class probability is very high for each class.

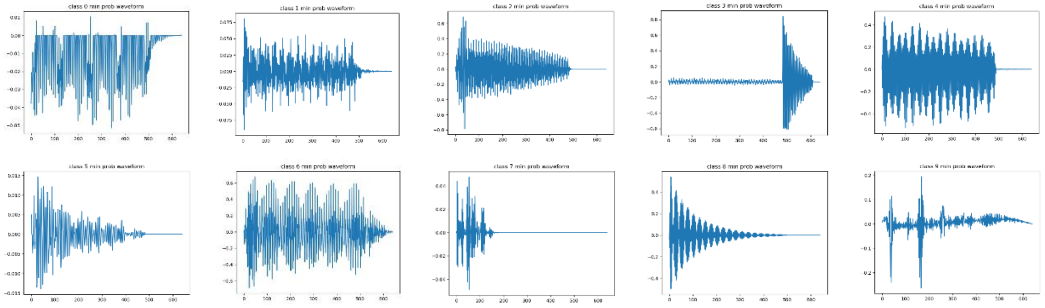

Figure 7. First CNN 1-D audio waveform where the correct class probability is very low for each class.

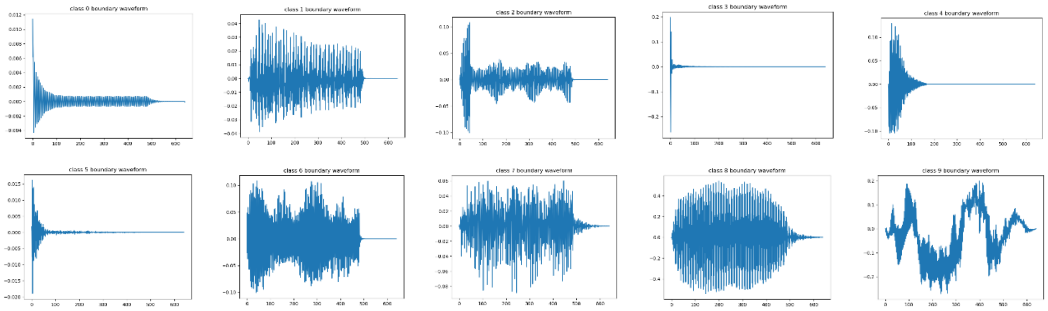

Figure 8. First CNN 1-D audio waveform for samples near the decision boundary for each class.

3.2. Improved CNN model results

The test results for the improved CNN model is showing below. The test accuracy is 60.47%. The learning curves for training and validation data is showing in Figure 9. The confusion matrix is showing in Figure 10. The waveform for samples where the correct class probability is very high, and very low for each class is showing in Figure 11 and 12. The waveform for samples near the decision boundary for each class is showing in Figure 13.

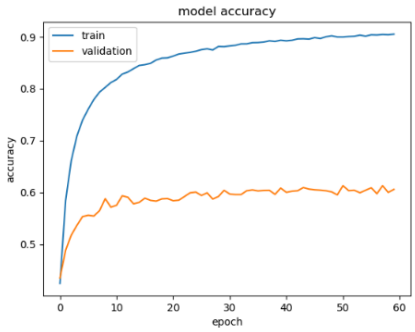

Figure 4. Improved CNN learning curves.

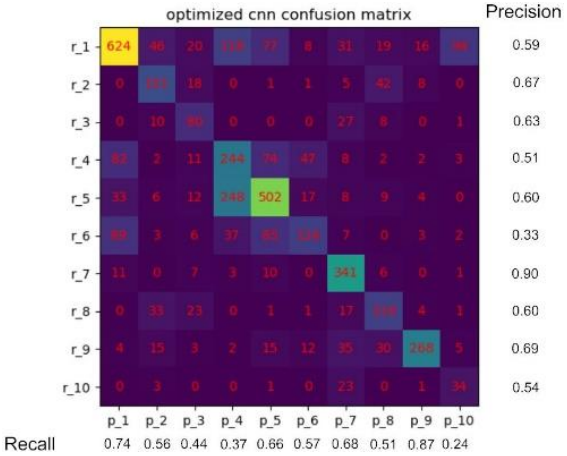

Figure 5. Improved CNN confusion matrix.

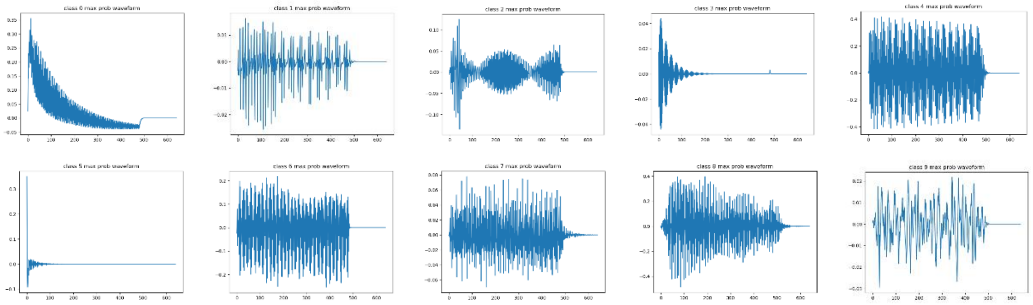

Figure 6. Improved CNN 1-D audio waveform where the correct class probability is very high for each class.

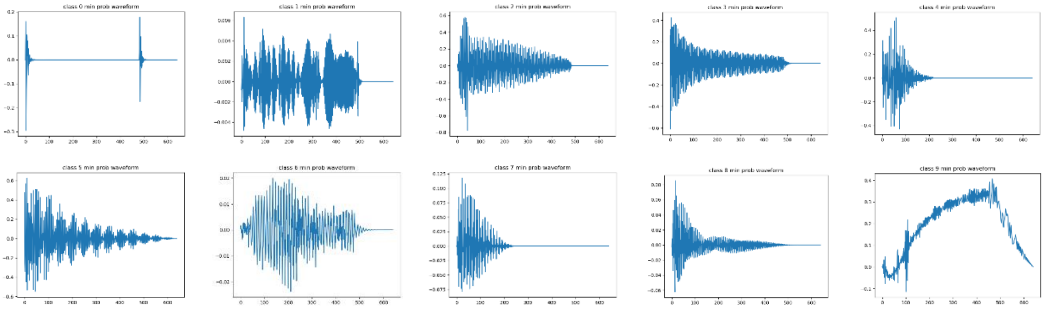

Figure 7. Improved CNN 1-D audio waveform where the correct class probability is very low for each class.

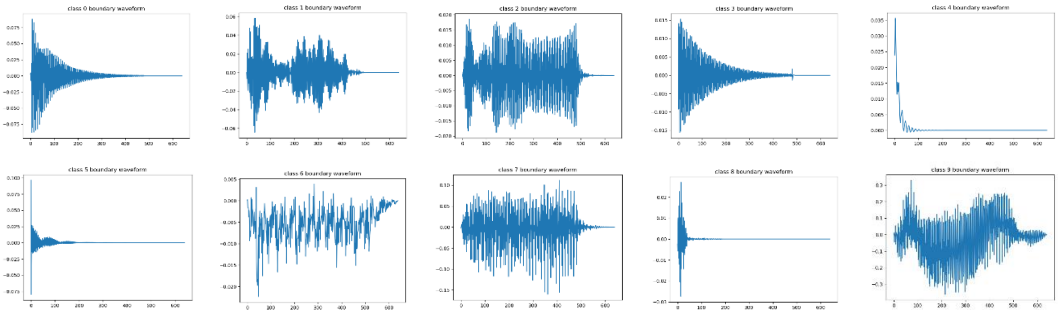

Figure 8. Improved CNN 1-D audio waveform for samples near the decision boundary for each class.

4. Discussion

From the result, we can see that the improved CNN model performs much better than the first CNN model (Increasing test accuracy from 33.69% to 60.47).

In our development process to improve the first CNN model. Our first move is to increase the number of CNN layers, and the number of filters for each CNN layer, since the training accuracy for the first model is very low (can be seen in Figure 4, around 37.5%), which means this model may not be complex enough to fit the training data. The first improved model was developed empirically. And we detected distinct overfitting issue occurred during the training process. As we discussed in section 2.3., Dropout was introduced. We tried different combinations of Dropout layers and rates, and an updated model (the model showing in Figure 2) has been developed. From Figure 9, we can see that adding Dropout layers can effectively improving the model's performance on overfitting issues.

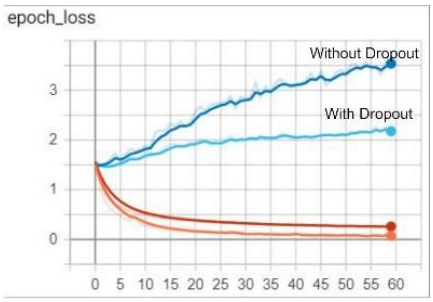

Figure 9. Improved CNN model learning curve on loss w/o Dropout layers.

Even though adding Dropout layers helps preventing overfitting issues, the learning curve of the improved CNN (Figure 4) still indicates strong overfitting. We tried to add more methods to prevent overfitting, but the test accuracy dropped dramatically. More experiments and studies need to be done facing this issue.

From the results of the first CNN, we can see that it has the highest classification rate on instrument class 2 (flute). Besides, it has a higher than normal classification rate on instrument classes 0 and 6 (bass and organ). In addition, it has the lowest classification rate on instrument class 7 (reed).

From the results of the improved CNN, we can see that it has the highest classification rate on instrument class 6 (organ). Besides, it has a higher than normal classification rate on instrument classes 1, 2 and 8 (brass, flute and string). In addition, it has the lowest classification rate on instrument class 5 (mallet).

5. References

[1] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," arXiv:1704.01279, 2017.

[2] G. Hinton, "Dropout : A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014.

[3] N. Ackermann, "Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences," Medium, 14-Sep-2018. [Online]. Available: https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf.

[4] "Categorical crossentropy loss function: Peltarion Platform," Peltarion.com. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy.

[5] D. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980