Machine Learning: Project 1

Overview:

We started our process of building the architecture with 3 fully connected layers and moved towards our basic convolutional network known as SimpleNet. SimpleNet features include one convolutional layer, one pooling layer and one activation function [1]. Then we moved on to our best architecture with multiple convolutions, described below.

Sampling – The highest pitch of 108 is equivalent to the frequency of 4187 Hz. As per Nyquist theorem, we needed twice the frequency per second which gives us approximately 32000 samples for span of 4 seconds. We uniformly sampled on every two seconds to achieve this sampling rate. To improve the training time, we ignored the frequency of 8th highest pitch which is equivalent to 2489 Hz. Thus, approximating it to 16000 samples.

This input of 16000 samples is fed to the network. The configurations of network are shown below:

Basic Architecture –
Based on the idea of SimpleNet, our basic network includes a convolutional with ReLU activation function and max pooling layer followed by flattening convolutional layer, followed by 3 fully connected layers and finally a softmax classifier.

Layer	**Parameters**	**Activation**
Convolution (1D)	(Input, Output) = (1, 6), Kernel = 5	ReLU
Max Pooling	Stride = 2	
Fully Connected 1	(Input, Output) = (47976, 220)	ReLU
Fully Connected 2	(Input, Output) = (220, 60)	ReLU
Fully Connected 3	(Input, Output) = (60, 10)	Softmax
Hyperparameters	Learning rate = 0.03, Momentum = 0.3	

Reasons for choosing this design:

- Before designing the architecture, we focused on understanding the processing of the waveform in CNN. It is different from images as the audio does not have high, low level features.
- Stacking up multiple convolution layers does not help for audio dataset as proved in this paper [1]. We tried stacking the convolution layers, but it did not give better accuracy.
- One intuition from the computer vision field is that pooling layers with convolutional layers, extract different levels of features such as edges, shapes, and objects. However, pooling layers help very little with the prøcessing of the raw waveform and may hurt the performance if not properly applied.
- Based on this analysis, the pooling layers will lower the effectiveness of the following convolutional layers. Therefore, one possible way to use them is by placing them after the last convolutional layer. In this case, they work purely as a compressor.

Description of Best Architecture –

Layer	Input /Output channel	Activation
Convolution 1 (1D)	(1, 6), 3	ReLU
Convolution 2 (1D)	(6, 16)	ReLU
Batch Normalization (1D)	16	
Convolution 3 (1D)	(16, 20), 5	ReLU
Batch Normalization (1D)	20	
Max Pooling	5, 5	
Fully Connected 1	(12780, 9220)	ReLU
Batch Normalization (1D)	(9220)	
Fully Connected 2	(9220, 260)	ReLU
Batch Normalization (1D)	(260)	
Fully Connected 3	(260, 10)	Softmax
Hyperparameters	Learning rate - 0.003 and Momentum - 0.05	

Cited from [2]

Layers	Description
Conv1d	1D convolution is used where the input is a short, fixed length segment such as audio wave in our case. Another way is to transform the audio signal to an image using some pre-processing and use a 2D convolutional network, but we preferred to work with 1D convolutional network.
ReLU	We chose relu because it works better than other activation function because of its computational efficiency and its ability to alleviate the vanishing gradient problem.
Maxpool	Max pooling, which is a form of down-sampling is used to identify the most important features.
Softmax	This layer is added as the last layer in the network which acts as the output layer. As, we need multi-class classification, this function squeezes the output of each class between 0 and 1 which can be considered as the probability of each class.

Summary of main findings –

- We analyzed that stacking convolutional layers are unnecessary and pooling layers can lead to the aliasing effect in audio processing which are the potentially most significant contributors to the limited performance of existing solutions.
- Therefore, we propose a new network called SimpleNet, which features a very concise architecture and high model interpretability.
- Our experiments empirically prove our analysis and demonstrate the effectiveness of SimpleNet.
- The learning rate is most important hyper parameter**.**
- Batch normalization helps in optimization.

Training:

Criterion functions – It is an important parameter which determines the performance of the model and is helpful in training the neural network. We chose cross entropy because we are using softmax in the output layer.

Optimization model – The objective of using optimization was to minimize the generalization error while improving the accuracy.

We used Stochastic Gradient Descent (SGD) for following reasons:

- It performs parameter update for each training example [3].
- Due to frequent updates, a high variance is observed in the parameters and the loss fluctuates to different intensities which helps is finding better local minima [3].

We tuned the hyperparameters to control the effective capacity of the model.

Observations:

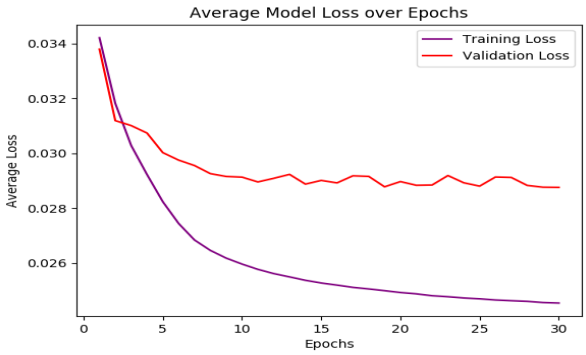
- When the learning rate is too small, training is not only slower buy may become permanently stuck with high training error [Ch. 11 of the Deep Learning book ('Practical Methodology')].
- When learning rate is too large, gradient descent increase rather than decreasing the training loss [Ch. 11 of the Deep Learning book ('Practical Methodology')].
- We got increasing validation loss on the learning rate of 0.05 and decreasing it to 0.003 gave us appropriate results.

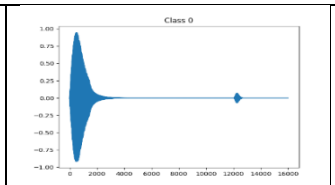
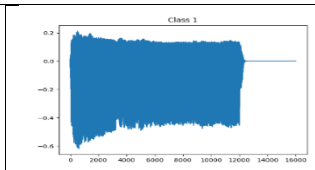
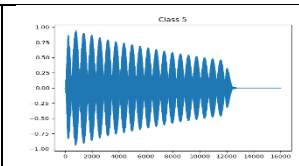
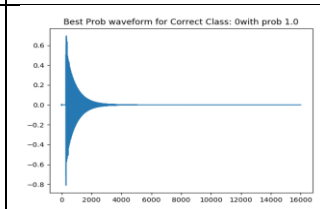
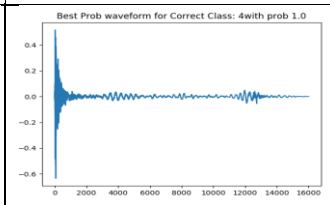
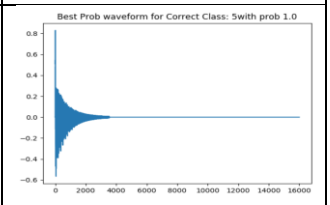
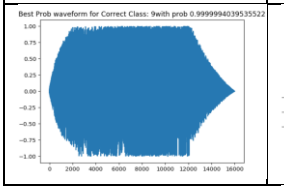
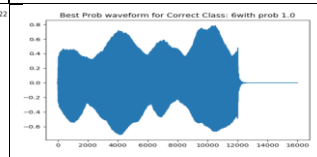
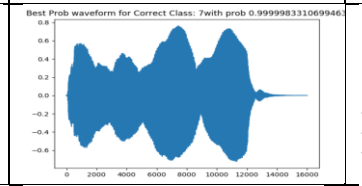
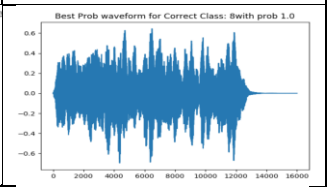
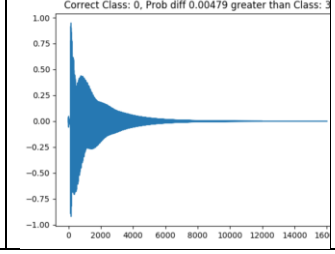
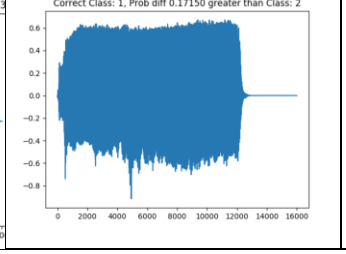
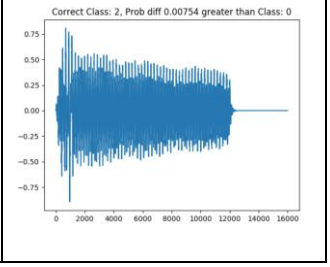
Training regimen – To train the network, following configuration changes were made:

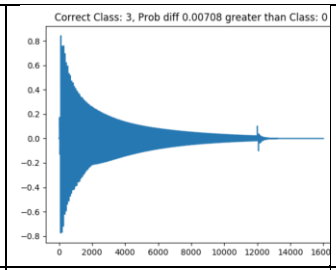
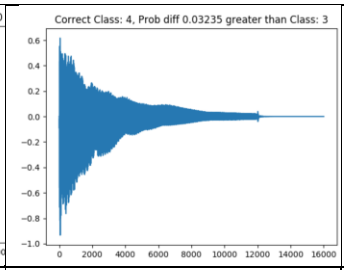
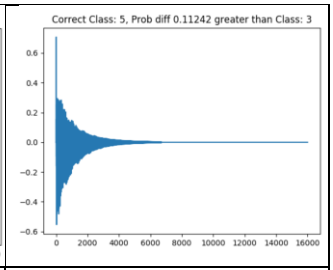
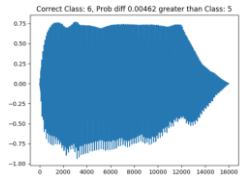
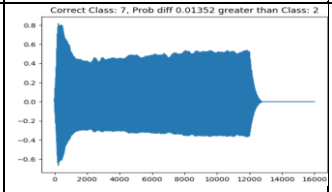
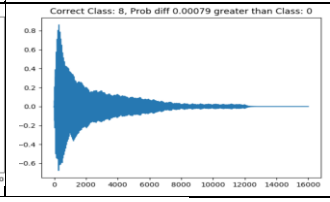
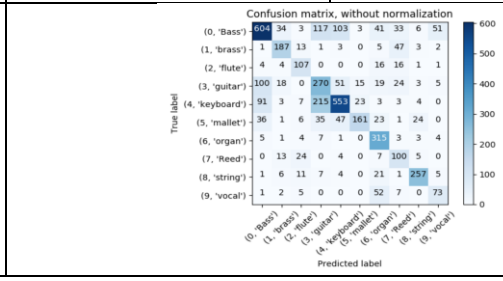
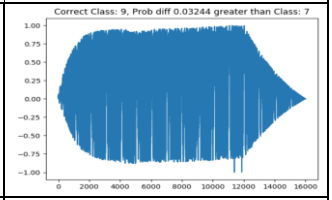
1. Sampling – We used Lambda Transformation function of pytorch for sampling the input.
2. Unsqueeze – The required input dimensions 1D Convolution and 1 input channel are [32*1*64000] where 32 is the batch size. The dimensions of the input tensor were [32*64000]. Thus, we used unsqueezing 1 to match the dimensions.
3. We Defined the network architecture, initialized random weights and performed SGD with learning rate of 0.03 and momentum of 0.3. We optimized the loss function using cross entropy and let it train for 50 epochs. For every epoch we are running the network on validation data as well. Over the training we keep noticing the training and validations loss to see how they perform. Decreasing training and validation losses are good sign. We save the best net here.
4. We run the best trained network on test data to see how well the network performs by looking at all the visualizations.
5. We looked at the plots to see how much the generalization error and accuracy is, and changed the hyper parameters accordingly.
6. Once we have a basic network up and running, we tried to improve the network by adding more convolution layers to extract more deep features, by changing activation functions, changing hyper parameters and optimized the model to get to best network which is explained in detail in "Discussion section of the report.
7. We started with simple network and by tuning the Hyperparameters we tried to reduce the generalization error. With an attempt to improve the accuracy, we performed batch normalization over all the layers. This is further explained in discussion.

Results:

Visualizations for Best network -

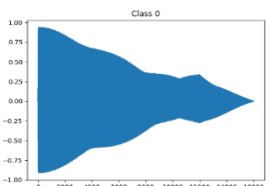
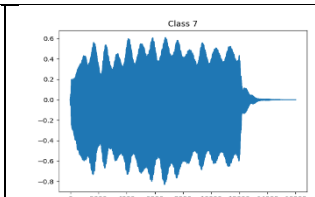
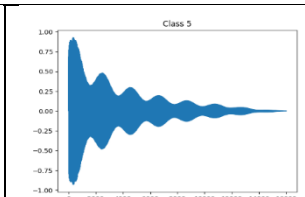
Accuracy of the network and individual classes on Test Data	Learning Curve
Accuracy of the network on test data is: 64 % Accuracy of Bass : 71 % Accuracy of brass : 69 % Accuracy of flute : 59 % Accuracy of guitar : 41 % Accuracy of keyboard : 72 % Accuracy of mallet : 79 % Accuracy of organ : 62 % Accuracy of Reed : 42 % Accuracy of string : 83 % Accuracy of vocal : 51 %	

1D audio waveform			
Waveform for samples where correct class probability is very high			
				
Waveform for samples near decision boundary where probability			

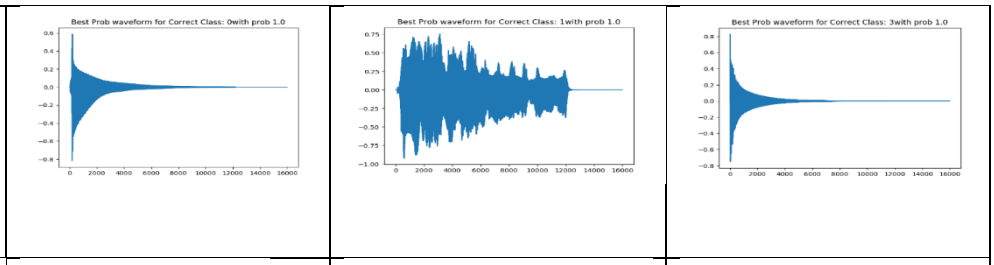
for correct class is slightly higher			
			
Confusion Matrix			

Visualizations for Basic network -

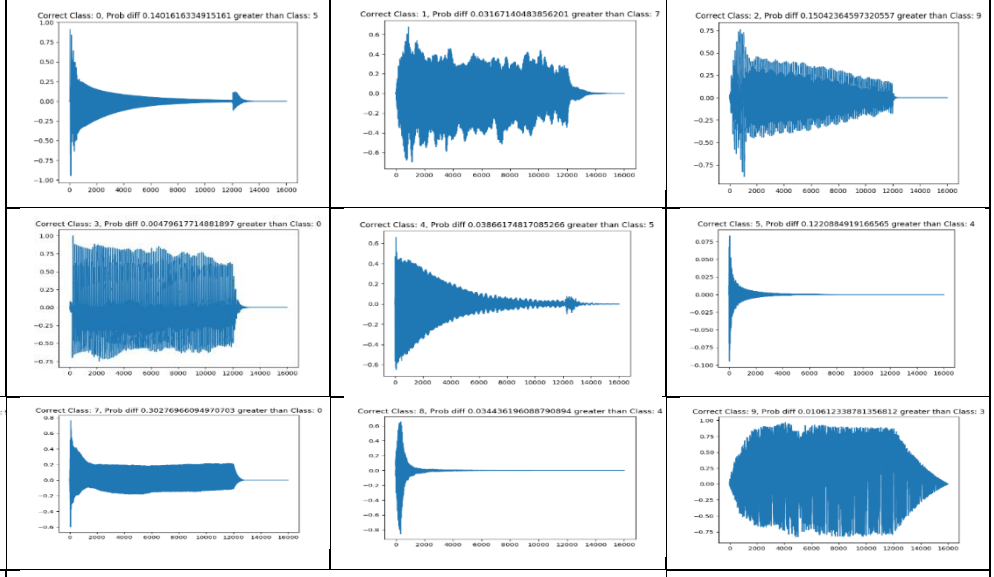
Accuracy of the network and individual classes on Test Data	Learning Curve
Accuracy of the network on test data is: 47 % Accuracy of Bass : 66 % Accuracy of brass : 73 % Accuracy of flute : 0 % Accuracy of guitar : 44 % Accuracy of keyboard : 72 % Accuracy of mallet : 48 % Accuracy of organ : 0 % Accuracy of Reed : 0 % Accuracy of string : 78 % Accuracy of vocal : 0 %	

1D audio waveform			

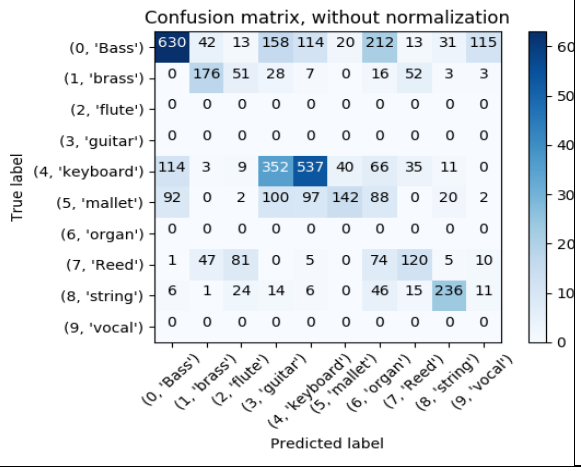
Waveform for samples where correct class probability is very high



Waveform for samples near decision boundary where probability for correct class is slightly higher



Confusion Matrix



Discussion:

Difference in results between your networks – An accuracy of 47% is achieved from the basic n/w whereas after improvisation, an accuracy of 64% is achieved.

Observations for basic Network – A single convolution layer with maxpool and relu activation function is used in the basic network.

- The class distribution in the training data is uneven. We can observe from the dataset that 'Bass', Keyboard and Guitar have large number of samples.
- Observing the best accuracy plots we can say that the network is confident in identifying the correct classes. But many samples show low accuracy as well.
- The network is most confused between classes Keyboard and Guitar as can be seen from confusion matrix. The reason being the sound frequency is low for these instruments. The network is also confused between Bass and Organ, and Keyboard and Bass. Organ has long sound, but bass acoustic and synthetic are different in length and waveform representation.
- Bass synthetic and bass acoustic have high variation in the frequencies over the span of 4 seconds which confuses the model for classification. Also, the number of training samples is more for Bass, thus, most classes are identified as Bass.
- As observed, decision boundary is confused with classes 0, 3, 4, 5, 7 and 9 as 0, 3, 4 have highest number of samples in training data.

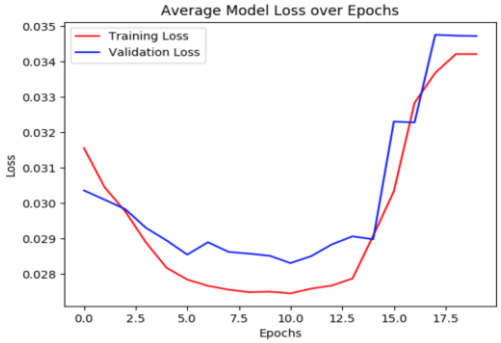
Attempts to improve the network - The network is taking too long to train. This could be due to the backpropagation of errors to the weights and biases we face undesired property of internal covariance shift.

- We applied Batch Normalization to the basic network and we were able to match its performance using only 7% of the training steps and we could improve its accuracy by 17%.
- To reduce the generalization error and improve the accuracy of the network, we tuned the hyper parameters by looking at the plot of learning curve to make it optimal (as discussed in the Optimization Model of the report).
- We tried Leaky ReLU to solve the problems of dying relu if it exists.
- We analyzed the normalization over different layers which were affecting the output and thus,we tried batch normalization.
- Implementing basic network without batch normalization led to some initial gains but then converged to a non-optimal local minimum.
- The Batch Normalization is applied after activation, and not before, which is derived from [6] as it gives better accuracy.
- The second layer changed the distribution of range 1 to -1, and the consecutive layers are not benefited. Thus, we added normalization to every layer.
- We applied Batch Normalization to our 1st network as well to see how well it performs but it gave 57% accuracy which is less as compared to the previous network. Thus, we conclude that the network with configuration of 2 convulation and 1 maxpool is performing better because of this configuration.
- 2 convolution layers with ReLU activation and batch normalization were added to analyze if the network can learn more features. This resulted in improved accuracy.

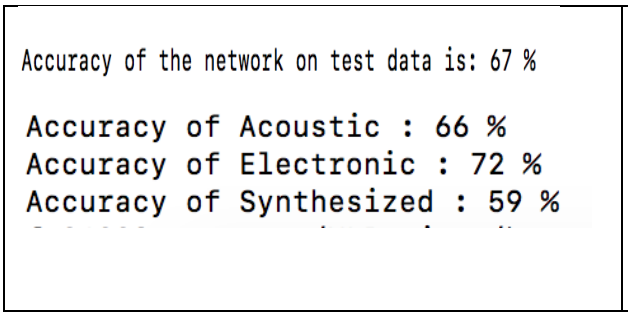
Observations:
From plots, we can observe that is able to differentiate between Bass and Organ and also the mis classification of Keyboard and Organ has been reduced.
Network is confident about its correct classes.
Now, in decision boundary we have only 0 and 3. This is because number of samples are high for these classes.
Now the network is able to identify every class which was not the case in previous

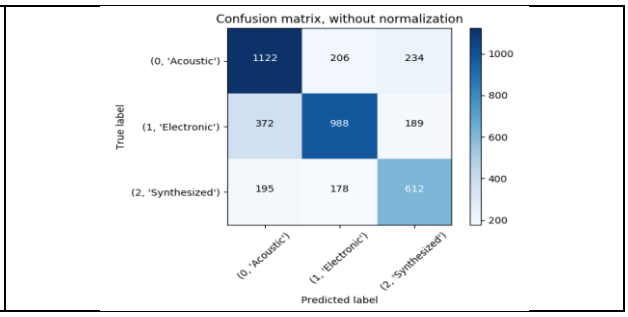
Bonus

We believe the problem of classifying between acoustic, electronic and synthesized is easier for network to classify over classifying 10 label for type of sound as these sounds are easily separable.
We were satisfied with our network design, which has 2 convolution layers with activation function applied on it and a final convolution layer with Maxpool and activation.
That was our base network to start with for this problem.
In terms of training regimen, its same as that of the best network but this time the output label is different. We got good accuracy in the beginning of the training , but after 15th Epoch, the model started to overfit.



We decided to simplify the network to one convolution layer, which is our second-best model. With batch normalization on it.
The architecture of the model is as follows:
Convolution1D → Relu → Batch Normalization1D → Maxpool1D → Fully Connected Layers → Sigmoid
This network gave us an accuracy of 67%.





Findings: The network is confident in classifying the correct class. Most of acoustic sounds generally ends in the first seconds, very few of the synthetic sounds behave this way. These are the kind of classes the network is confused for.

References:

[1] https://openreview.net/pdf?id=S1Ow_e-Rb
[2] https://engmrk.com/convolutional-neural-network-3/
[3] http://rohanvarma.me/Loss-Functions/
[4] https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/
[5] https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f
[6] https://arxiv.org/pdf/1502.03167.pdf
[7] https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf