Intro to Machine learning - Project 1

Govindarajan Chandraketu, **Gokul Rajkumar** Lenin Pillai

28-Nov-2018

- **Initial network :**

  This network was intended to understand the data and classify between the 10 instrument classes. We took a sub-sample of first **9000** datapoints to perform the classification.

  - Overview :
    This network was built from scratch by experimenting and trying out different parameters to tune the network for better classification results. This network has 3 layers (1 Convolutional layer and 2 Fully connected layers). This network operates on the data sample by sub-sampling the datasamples and considering only the first 9000 datapoints for each data sample. This made quite a difference in terms of efficiency, as the network has less data to process, but enough to classify most of the data sample properly.
  - Training :
    We used **cross-entropy loss** as our criterion function, **S**tochastic **G**radient **D**escent as our optimizer and we used **0.01** as **learning rate** and **0.3** as **momentum** to train the network.
  - Results :
    The network trained itself to classify the dataset, and it ended up classifying the test dataset at about 41.65% accuracy. Below is the confusion matrix of the test dataset.

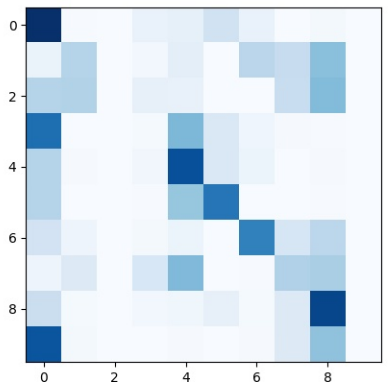


Figure 1: Confusion Matrix for the test dataset

  - Discussions :
    The above confusion matrix shows that the network did classify good amount of data samples properly. But, it didn't learn about every class. In fact it didn't learn anything about class(2)-brass and class(9)-vocal. The reason in our theory for the network not learning the classes(2 and 9) is that the probability of a data sample being either a class(2) or a class(9) is about 8%. As these are the most sparse classes in the given dataset, the network didn't bother to learn about them. Below is the learning rate for the Initial and primitive network. Below are 4 samples based on the classification correctness and network's prediction probability.
    For the figure top(a), the initial network classified a data sample from class(5) as a data sample in class(5) with really high confidence. But, for the figure top(b), the initial network wrongly classified a class(0) data sample as class 5, with high confidence. Meaning, the network was really sure about it's answer and got it wrong. But, quite convincingly, these two data sample visually looks like originated from the same class, (as we are considering only [0-9000] subsample), though the didn't. Similarly, for the bottom(a) figure, the initial network classified a data sample which belonged to class(5) as a data sample that belongs to class(5). But, this time, it classified it with very low confidence. Meaning, it wasn't really sure about it's answer, but somehow got

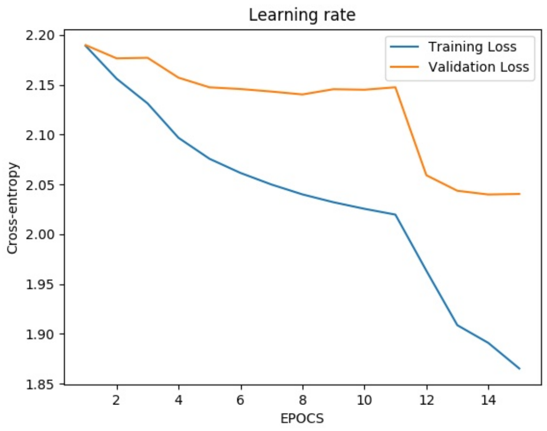
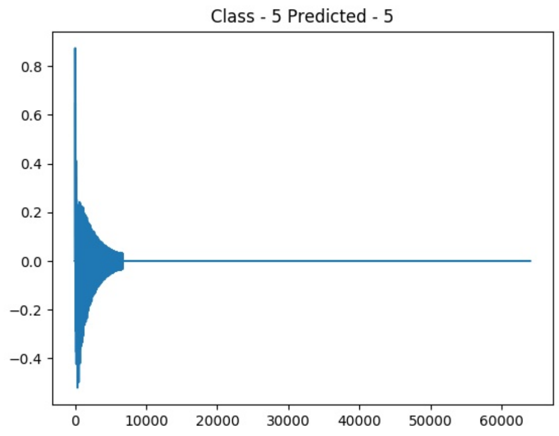
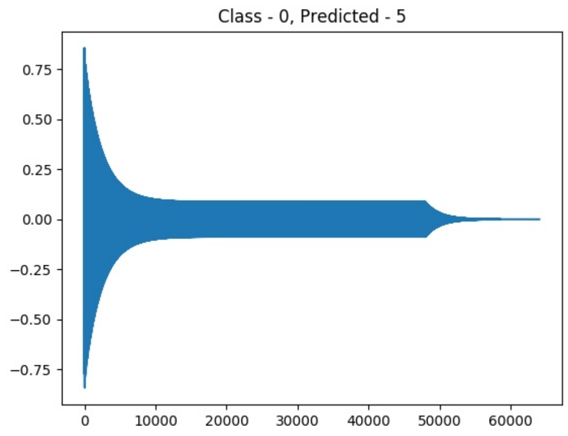


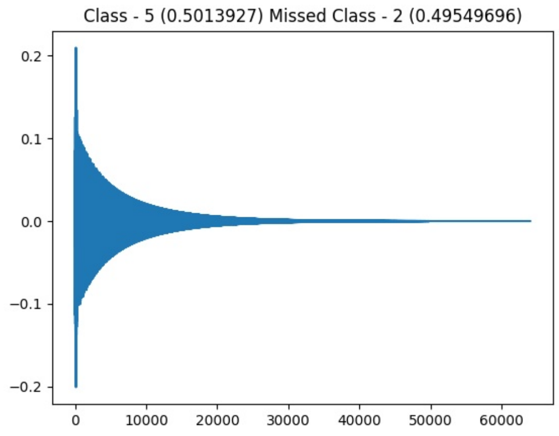
Figure 2: Learning rate



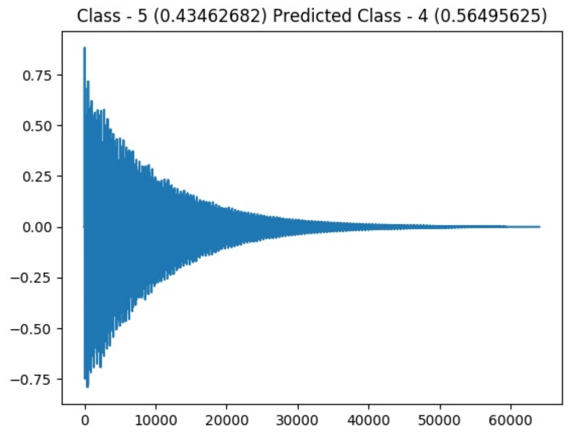
(a) Class 5 classified as 5



(b) Class 0 wrongly classified as 5



(a) Class 5 classified as 5 with low probability



(b) Class 5 wrongly classified as 4 with low probability

it right. Similarly, for the bottom(b) figure, the initial network classified a data sample from class(5) as class 4. But, it barely missed classifying the data sample into it's right class.

- **Advanced network :**

  - OVERVIEW :
    This network was inspired from **LeNet-5** as this data had 10 classes and was equally difficult as the Fashion-MNIST dataset. We just did a slight modification by changing the 2-D convolutional layers to 1-D as this data consisted of audio samples which are 1-D in nature. We also had to change the no of neurons in the fully-connected layer, due to larger data samples. But, we took a very structured approach and eventually reached close to LeNet-5 network structure. Below are the justifications for it.
  - TRAINING :
    We used **cross-entropy loss** as our criterion function, **S**tochastic **G**radient **D**escent as our optimizer and we used **0.01** as **learning rate** and **0.3** as **momentum** to train the network.
  - RESULTS :
    The network trained itself to classify the dataset, and it ended up classifying the test dataset at about 58.34% accuracy. Below is the confusion matrix of the test dataset. This shows that the

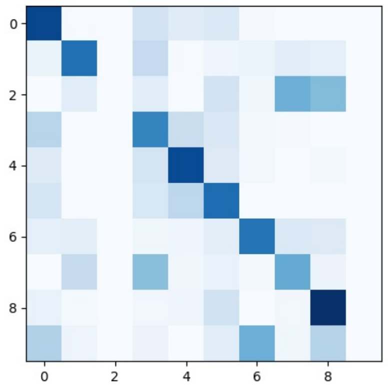


Figure 5: Confusion Matrix for the test dataset

network classified most of the datasamples correctly. But it wasn't perfect as it did not learn to classify the class-2 and class-9 at all. In a way, we can confidently say that the network was not overfitting, as the learning rate was always decreasing for the training and the validation dataset. Below is the learning rate curve.

  - DISCUSSIONS AND SCENARIOS TRIED :
    We slowly started to enhance the primitive Initial network. Below are the list of failure cases faced and our intuition towards it.
    - **More layers (Success) :**
      We initially thought that there are more features that the network is unable to learn. We also inspected the output of the first convolutional layer (after 50 EPOCS) by plotting, to understand if the feature extracted are distinct. But, we couldn't really decipher the audio waves and the feature that the neural network learned. So, we added one more convolutional layer. This lead to an increase in feature understanding of the network and ended up giving better classification result. But the network took more time to train for each EPOCH.
    - **Sampling range (Failed) :**
      We were using Relu as the activation function till now. So, we started thinking if Relu is

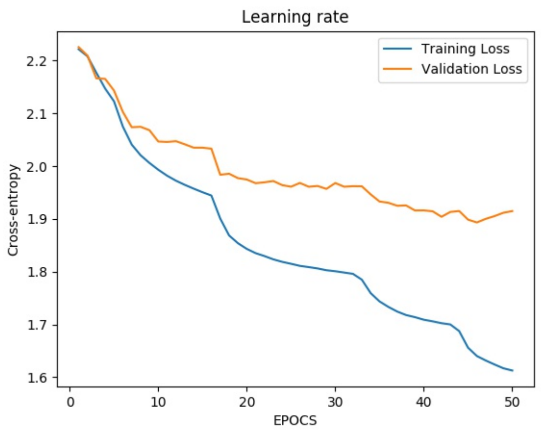
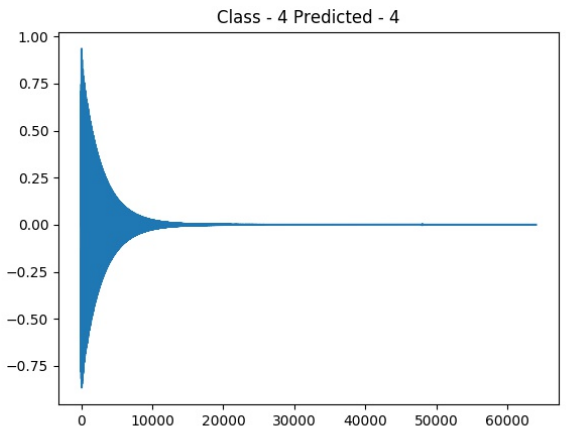


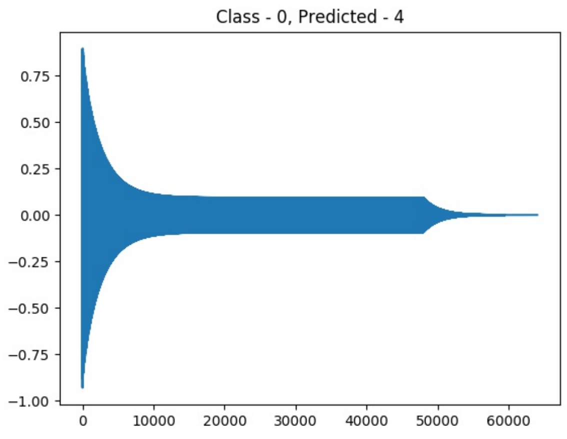
Figure 6: Learning Rate of Network 2

unable to extract the features to it's best. We then realized that Relu is defined as non-existent or 0 for the entire input range except [0 to 1]. But our sample data is defined for the entire range of [-1 to 1] (after normalization). So, we tried normalizing in the range of [0 to 1] hoping that relu will be able to capture more detail and features about the data samples, but in vain. The accuracy ended up falling quite a bit, making it a bad approach.

- **Changing the activation function (Failed) :**
  We then started thinking about other activation functions. What if Relu doesn't suit these convolutional layers ? So, we changed the activation functions to other functions like Sigmoid, tanh etc but in vain. The accuracy of the network again decreased and the network didn't learn for many epochs at all. Then we plotted the data samples of the initial network based on 4 cases. The case where the network successfully classified with high probability, classified incorrectly with high probability, successfully classified with low probability (boundary condition) and the network incorrectly classified with low probability. We plotted the best data samples for each of these 4 cases (if occurred) to observe the pattern in which the classification and mis-classification occurred. Below are some of these plots.



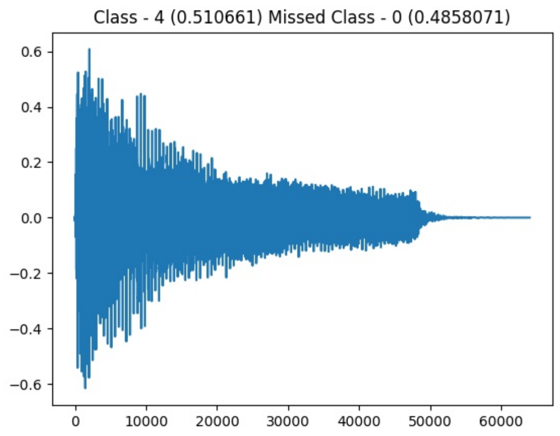
(a) Class 4 classified as 4



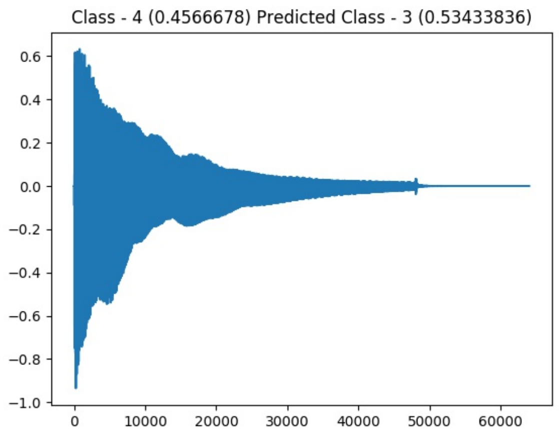
(b) Class 0 wrongly classified as 4

Visually speaking, the plots top(a) and top(b) look quite similar, especially in the range of

[0-9000]. Despite of the misclassification, it's quite convincing that the network is in the right direction, as it is picking up similar looking shapes. The bottom(b) shows that the



(a) 4 classified as 4 with low probability



(b) 4 classified as 3 with low probability

class 3 was predicted while it was actually class 4. But, the probabilities are quite close. The bottom(a) shows that the class 4 datasample was classified properly as class 4. But the network wasn't quite certain about the decision, as the probabilities in the brackets indicates so. For the range of [0-9000] visually too the dataset quite convinsingly looks both of class 0 and class 4 type.
All saved plots can be found in the folder along with this PDF file.

- **Improve sub sampling (Failed) :**
  We noticed the plots and the waveforms visually and realized that some of the classes can have better accuracy if the subsampling was done differently, as there were few other sub-sample areas where the data samples were clearly distinct. For instance, with class 9 (vocal), the later part of the audio samples were non-empty. Meaning that most of the vocal notes were sustained almost to the end of the recording. Just by considering this observation, the network should be able to classify data with a better accuracy for atleast class 9. And in many of our initial attempts, none of the data samples were classified as class 9. So, we tried adding more sub-samples to the input for the network. But again in vain, as this still didn't improve the accuracy. We tried subsampling 9000 datavalues at the beginning of the network, 9000 in the middle and 9000 towards the end of the network all combined. Even the combination of these data samples didn't improve the network's accuracy.

- **Custom Pooling layers (Failed) :**
  We then noticed that most of the jaggedness in the dataset aren't being classified properly. So, we tried adding more layers to check if it improved the accuracy, but it didn't. So, we started thinking if the pooling layers are the bottle neck. As we are doing maxpool, this always captures the max of the current range. What if there are more details that could be understood in the min sample values. So, we wanted to use other pooling layers like AVGPool, MinPool etc to check if improves. There was no minpool library available. So, we had to write it manually. So, before putting efforts to compute something, I wanted to determine which one works best. So, we took a more formal approach. We segmented each data sample into 8 equal chunks (half a second each) and extracted features (like average, min, max, standard deviation etc) for each of the chunk for each data sample in the test dataset. Then we performed a PCA and projected the data in two most significant components. This way, we can understand which features have most variation in a sub-sample. Hence, I can direct my network to learn those variations. The plan was to use those noticed variation in the pool layers, thereby helping the network to learn some particular features really good. But, this didn't work as PCA gave the below plot as output.

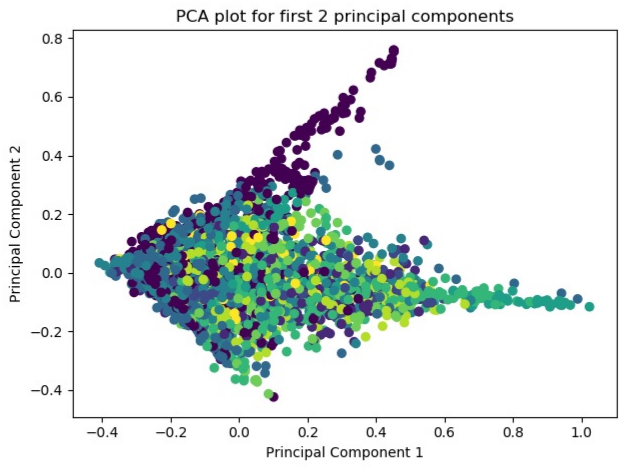


Figure 9: PCA in first 2 dimensions

  This shows that all data samples are equally spread across feature space thereby making the pooling layer's choice unintuitive and unproductive too.

- **Maxpool layer's range (Success) :**
  Our task that to identify the jaggedness was still not met. So, we concentrated on pooling more. We understood that more detail is being lost if we perform a 1D maxpool of 10 (which is what we had been doing initially). This is like loosing resolution in an image at a ratio of 10:1. Then we realized that it didn't help the network learn the features quite well. Then we reduced the network to perform a maxpool of 2. This ended up having a network that's quite a bit more heavy than usual. But, improved the accuracy too. Especially with the newly introduced Convolutional layers, the network was able to learn more features easily and thereby the overall accuracy improved.

- **Dropout layers (Failure) :**
  We also tried to us dropout layer in our CNN. This didn't improve the accuracy at all. Later did we both conceptually and logically realize that the drop out layers are usually used to prevent a network from overfitting. Our network was anyways not overfitting, which made this approach ineffective.

- **Instrument source Network Bonus:**

  We used both the Initial and advanced network of the Instrument family networks to train and determine for the instrument source, with small modifications to it.

  - **Overview :**

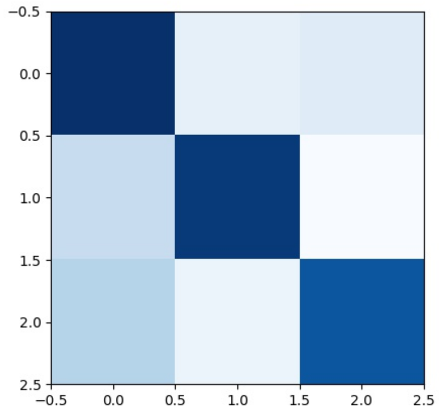
    The network was modified to give 3 outputs rather than 10. This makes the Network learn for the labels (which are Instrument source) for each EPOCH.

  - **Training :**

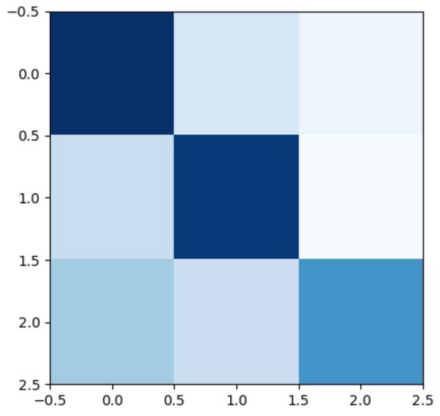
    We trained the network using the same training parameters as the previous 2 networks.

  - **Results :**

    Result of the 2 networks (Initial and advanced) were quite similar, though initial network is significantly simpler in terms of architecture compared to the complex advanced network. Below are the Confusion matrices of the initial and advance network after training.



(a) Initial Network Confusion matrix



(b) Advanced Network Confusion matrix

  - **Discussions :**

    The advanced network had slightly lesser success rate for class(2)-Synthetic, but had slightly higher success rate for class(0)-Acoustic and class(1)-Electronic. But, the apriori probability of the Acoustic and Electronic classes are slightly higher than the Synthetic class, making the Advanced network little more successful. But, the advanced network had 7 layers in the Convolutional Neural network where as the initial network had just 3 layers. So, looking at this value proposition, I would suggest that the initial network is good enough compared to this advanced network, as it did yield only little more accuracy for a high cost of performance.