**TIMELINE:**

*2nd April - 3rd April:* Downloading the data files and then uploading them to google colab. Then cleaning of data and data visualization was carried out.

*4th April:* Deciding which kind of architecture to work with planning the architecture and deciding the input shape of the sound files. (5th April)

*6th April - 7th Apri*l*:* Writing the code for training the dataset and checking on validation data to determine the best possible arrangement of convolutional layers.

*7th April:* Finally testing on the test dataset and making required submissions.

*7th April:* I was planning to increase the efficiency by using resnet and tried to implement confusion matrices for better results.

*7th April:* I was thinking about more fine-tuning the hyperparameters so that I could get better results.

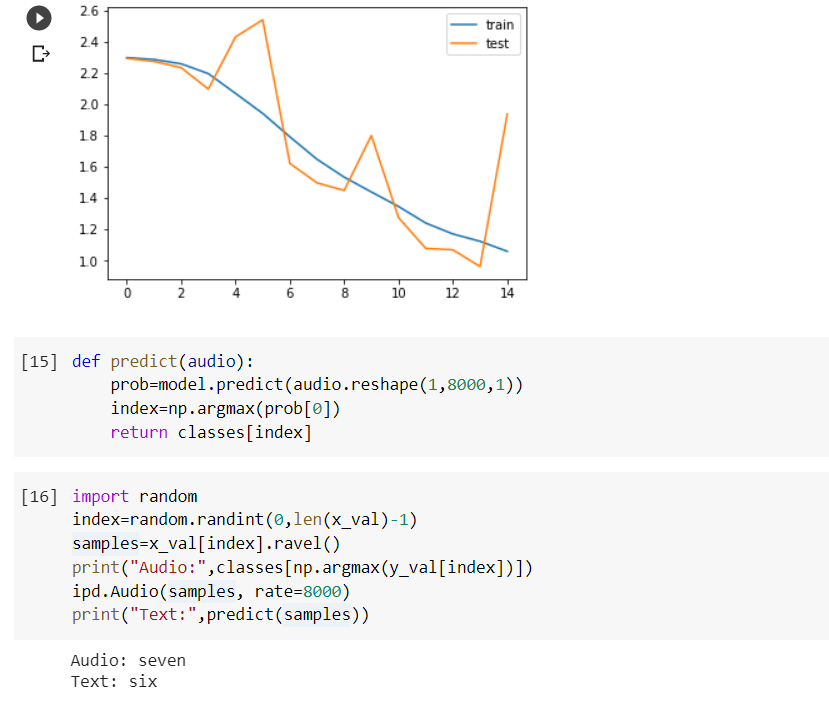
*8th April:* I was planning to use this model to construct sentences when a user is speaking, that is, I would give a user speech and my algorithm would recognize the speech.

*8th April-9th April:* I built a basic layout and wrote some code but faced repeated errors, but given time I would work more on it and fix it.

**The motivation behind using CNN:**

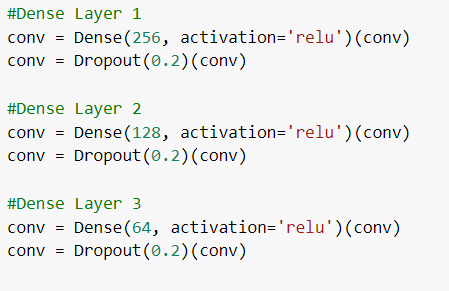
1. This is the most efficient and lightweight model I came across, with decent accuracy in the range of 85–87%. It is probably the best single model to use out of the box in production with a little tuning. I achieved an accuracy of around 85% using it.
2. Further, I have focused on fine-tuning the parameters to achieve the best accuracy out of this architecture. (Given below are graphs showing my efforts)
3. One of the reasons behind my mentioning about Resnets and not using it was my lack of hardware power.

1st Attempt:



1. Here, I have used 6 convolutional layers and 3 dense layers. This was a purely experimental setup. I started with this and further made modifications in the layers to get various results.
2. I have added the structure of convolutional and dense layers below for reference,
3. The above graph suggests there is definitely some overfitting of the data which could be inferred by a sudden abrupt rise in the curve near iteration 13. This motivated me to reduce the number of layers to prevent overfitting.

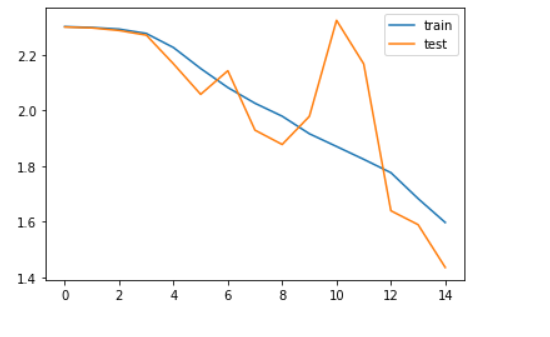




Here I showed my 3 major attempts in deciding on architecture. I had done several parameter tuning. Changes in dropout values, filter values were made by me and the best result was incorporated.

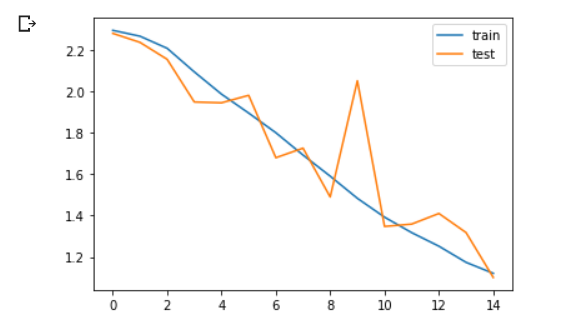
2nd Attempt:

1. Here I used 3 Conv-1D layers with 3 dense layers. Here I achieved the below graph. Comparing the previous graph with this one I could see this is better. Overfitting was eliminated.
2. Both test and validation error was reduced.
3. Now it had the risk of underfitting but the graph was hinting that I have not yet under-fitted because then the graph would have been flatter if I had under-fitted.
4. So I further reduced 1 layer to see the effects.

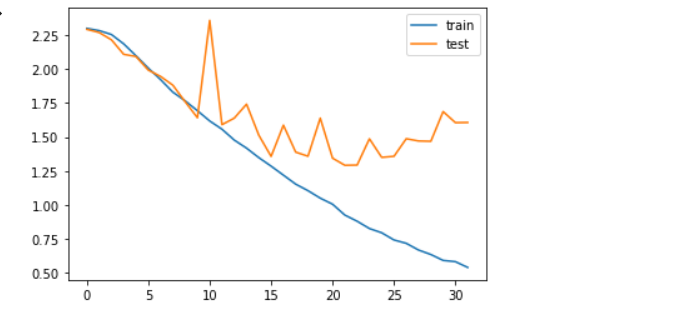


3rd Attempt:

1. Here I used 2 Conv-1D layers with 2 dense layers. Here I achieved the below graph. Comparing the previous graph with this one I could see this is better in terms of both test and validation error.
2. Here I can say both underfitting and overfitting problems are being taken care of as we can see there are not many steep rises in the curve. In the first attempt, there was a more steep rise as compared to this one.
3. Then I performed 100 iterations to better understand the overall effect and achieved the following graph using the architecture of my 3rd attempt.



The curve for 100 iterations stopped at 30 because it was giving the best results after 30 iterations:



1. Iterations stopped at iteration number 32 giving the best accuracy. From the plot, it is visible that validation, as well as test error, has also reduced a lot.
2. There are a lot of peaks but they are not very abrupt giving me confidence that this model would give comparatively good results. Small spikes suggest underfitting is not taking place.
3. Now I have saved the model and will use it to predict the test data.
4. Further, I have tested the given dataset on this model and achieved a score of 0.8019.

**ACKNOWLEDGEMENTS:**

1. I have read the following kaggle notebook and derived understanding to make my project.

<https://www.kaggle.com/himanshurawlani/a-cnn-lstm-model>

1. I have read the following “towardsdatascience” article on speech recognition which gave me insights on what kind of CNN architecture might help.

[<https://towardsdatascience.com/tensorflow-speech-recognition-challenge-solution-outline-9c42dbd219c9>]