DEEP NEURAL NETWORK - FALL-2018
PROJECT3 (HW5) Classification of the Dog and Cat Voices and Voice Generation

Azad Almasov (1511417)

The University of Tulsa

December 13, 2018

1 STATEMENT OF THE PROBLEMS

We have WAV data of dog and cats. And our main purpose in this project to classify dog and cat voices properly. I have tried different NN methods to do so.
Furthermore, I have tried to do regression to be able to predict next 10 data points given 100 previous data sequence.
After all, having had model that can predict future data points with good accuracy, I tried to generate cat or dog voices.

2 SOLUTIONS

We have to know which kind of data we work with. WAV files we have are in int16 format. So we need to remember when we generate data we need to return its type into int16. Our rate is 16000 data point for 1 sec. I converted voice into numerical data using **scipy.io.wavfile** method where you can read and write WAV files. I plotted one of the data for the purpose of the illustration (Figure 1) **Preparing data** We have plenty of data with

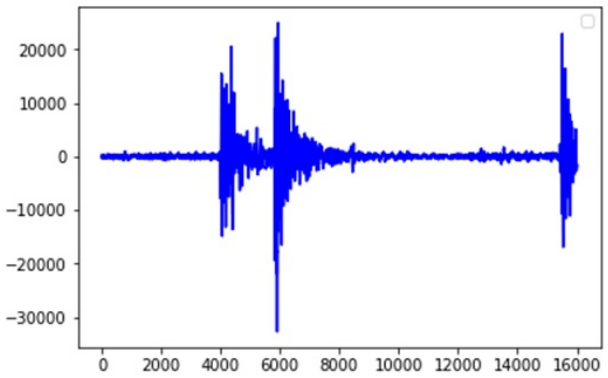

Figure 1: *wav data*

different length. First of all I need to pad them into the same length. After downloading them I have defined **max-len**=100000 for my data length. If data has more data it will be cut and if it doesn't have that long data we put them as zero. When one listen those audio voices you would see that we dont here cat or dog voices at

every part of audio part. Sometimes it is in the middle sometimes it is in another parts. Furthermore, we have irrelvant voices like door opening etc. Therefore, as Dr. Girgis suggested, I have defined **portion** to cut my data into that portion from different places evenly skipping some points so that I have good generalization about voice then combined them to get max-len data.
Then I normilized my data for classification process. First I have tried the following dense model:
Dense1: Dense model 1024-dp0.5-512-dp0.5 activation tanh. I have used 1024 as my first layer to get rid of bottleneck effect since I have 100000 descriptive feature. However, my accuracy was low and I couldn't fight with that even though I have tried different layers dropouts regularizations (Figure 2). test-acc: 0.54.

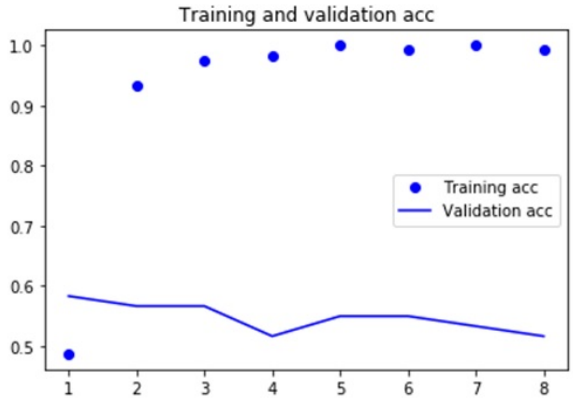

Figure 2: *Dense1*

Then I have tried GRU model. First I defined my data as sequence of 10 data and each have just one descriptive feature. Since I have one descriptive feature I dont see any meaning to put 32 output layer mathematically. Even though I have tried 32 output case and 4 output case and I got the same result: **GRU1: gru4-dense256-dp0.5-128-dp0.5-32-dp0.5**(Figure3) My accuracy was not good again.

I have tried bidirectional model, conv1d model but results did not improve more than 55 to 60 accuracy. The reason is that we have 100000 descriptive feature and we have just 179 trianing data. This of course will always result tremendous overfitting and accuracy will be low. I dealt with this problem as following:
Importance of Optimization algorithm If you look at previous plot you would see that loss is not affected that much. If we have this kind of situation we need to modify optimization algorithm since their search strategy are different. I have tried (optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True) as my optimizer. And this affected my results. However, I did

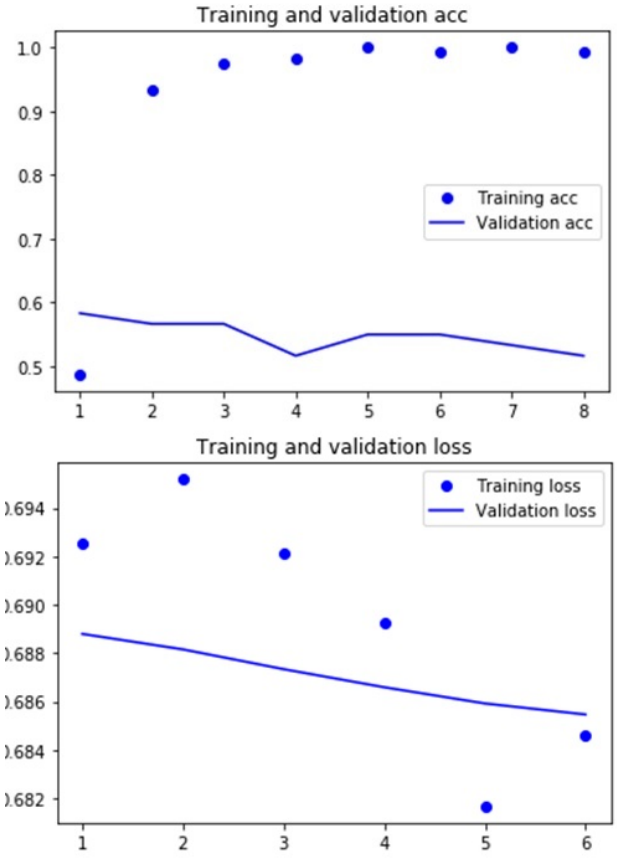

Figure 3: *GRU1*

another strategy to improve efficiency (not only accuracy. But this strategy didnt affect accuracy that much). It is data preparation strategy. When we see Figure 1 we can see that higher values we get when we hear dog barking and cat mewing voices appear. So what I did for each data is that I found std and mean for each data and took only those values are lower than mean -0.5std and higher than mean+0.5std. Then I generated wav files after this cut of data. I heard almost only cat mewing and dog barking extremely. So this strategy helped me to get rid of more than 60 percent of my data and thus I could use less max-less value 16000 which means using 1 sec of data I can predict dog and cat voices. Lets see how our accuracy improved:
Bidirectional1: redifined bidirectional (Figure4)
test-acc: 0.6836734693877551

CONV1d1: C1-32-mp-C1-32-gmp-dense256-128-dp0.5-32-dp0.5-SGDoptim-REDIFINED(Figure5)
test-acc: 0.6836734693877551

I improved accuracy to 70 percent by applying conv1d layer 2 times:

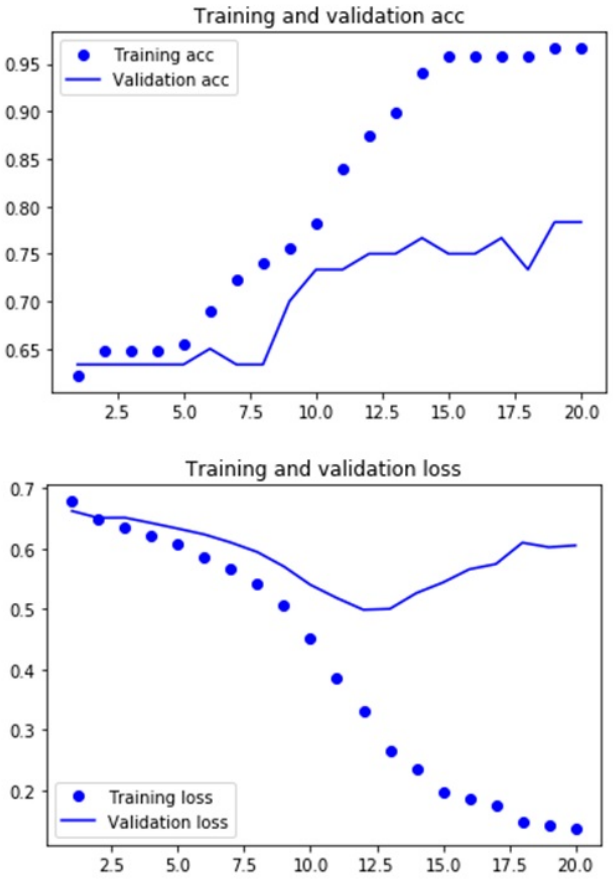

Figure 4: *redifinedBidirectionalSGD*

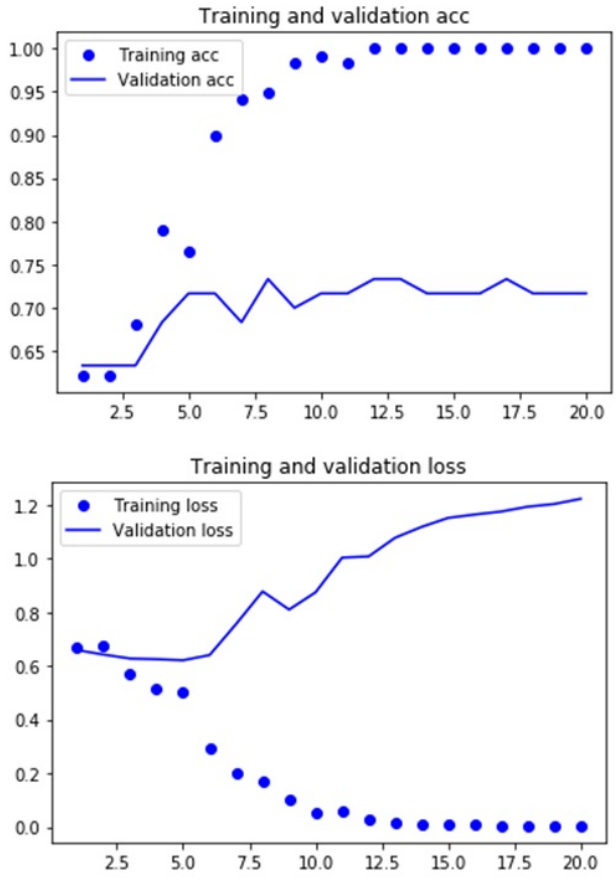

Figure 5: *con1d1*

'C1-32-mp-C1-32-gmp-dense256dp0.5-128dp0.5-32dp0.5-SGDoptim-REDIFINED(Figure6)
test-acc: 0.6938775510204082

However, I could not overcome overfitting issue, since still my descriptive feature size is much higher than my data size. In my opinion with this kind of noisy data 70 percent accuracy is fair.

Sequential Data Prediction -Regression

I did this regression for cats and dogs separately. Actually, I wrote code where you can use either only cat data or only dog data or mixed dog and cat data. Now lets look at predicting next 10 data given 100 previous data. I used cat data for the illustrative purposes. Actually the reason why i use cat data is because dog data is 4 times less than cat data, and when I did regression lack of data showed its side effects on my results. Again I got higher lost value 28 percent (Figure 7). I thought this might be because I modified my data. I tried it with original data (and, of course, z-score normalized). It little bit improved but not recognizable.

First of all I have to tell that I found conv1d-conv1d, which were my best choice in classification, model good for regression as well. Using that model I tried to found out my problem causing this kind of high loss. I had z-score normalized my data and used *tanh* in classification problem and here as well. In classification problem our ouput is probability space having elements between 0 and 1. Therefore, using z-normilized data, which map my data into space having bounds -10 and 10. During transformations these descriptive features will be converted into space having bounds -1 and 1 due to tanh function. This is ok when we do classification. However, in regression, I cannot constraint my output to be in the bound -1 and 1. Because my output is also z-scored data and it has to take value between -10 and 10 (these are just max and min of z-score normalized data). Therefore, I mapped my z-score normalized data into space having bound 0 to 1 using max-min scaling, and i used relu for my activations. This improved my data a lot. I got 2 percent loss (Figure 8). As optimizer RMSProp worked well in contrast to classification problem.

Using original data of time sequence data is important

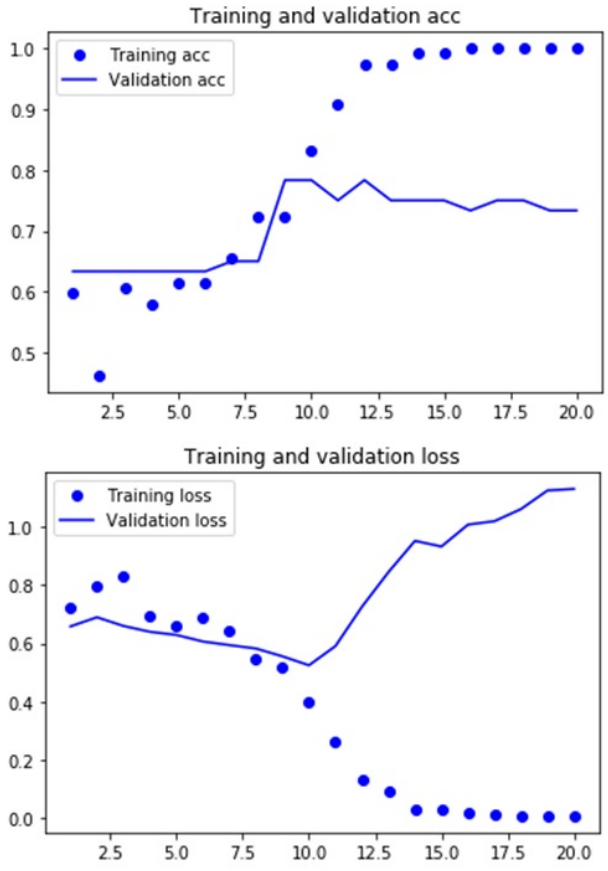

Figure 6: *con1d-con1d*

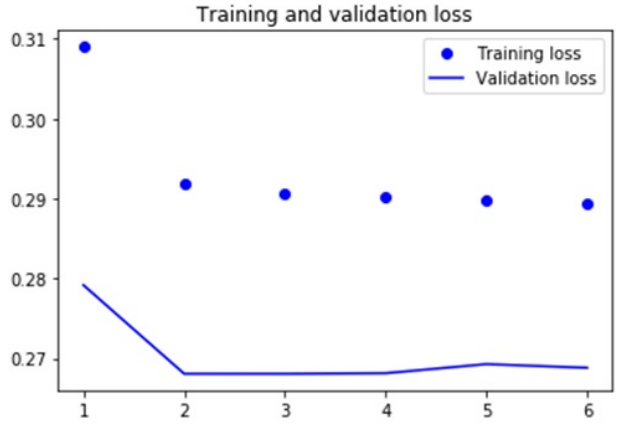

Figure 7: *regressionCatAttempt1*

in regression. I got 5 percent loss when used modified data (here when I say modified I mean when I modifed my data for classification - taking numbers only between some range). So using original data improves regression.

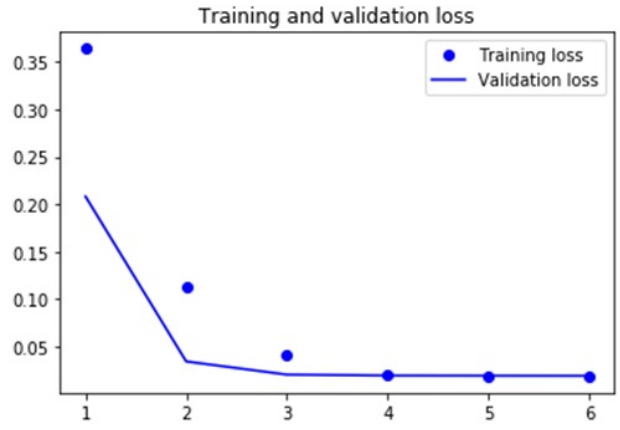

Figure 8: *regressionCatAttempt2-normilized and min-max scaled*

Furthermore, since I am going to generate data using this model original data is important for me. Since my original data size will be more than modified data, I will take higher max-len value = 50000 = 3 sec data cut from different parts of given audio files.

Generating Data
Using well trained model for cats I wrote code which takes output as its input and keeps generating data. Remember that we converted our data type into float32 format to be able to work with it. Now when we want write wav file first of all we need to scale it up to its original values (unscale then un z score). After doing that we need to convert datatype back to int16 so that we can write hearable audio wav file. So I did so but generated data was just signal - monoton voice (Figure **??**). I think this because I did not choose my initial data point well. That is very hard to choose good initial 100 data (which means 1/160 sec) among 1 million data where cat will start meowing. Even though regression model accuracy was 98 percent I couldn't catch best point to generate cat meowing. However, I generated the sound I created as an example in the project folder.

3 CONCLUSIONS

• Data preperation is very important and can be different depending on problem
• Optimization algorithm is very important depending on problem, specially where your loss value directly stabilize very fast and you see overfitting at very early time
• In regression depending on your output range, choosing appropriate scaling and normalizing can affect your results dramatically.