MAIS 202 – Project Deliverable 3

Xinyi Zhang

1. Final training results

As discussed in Deliverable 2, I attempted to apply transfer learning and data augmentation to my model. I had trouble getting a suitable pre-trained speech recognition model that was actually usable with my data, so in the end, after a lot of research, I decided to only go forward with data augmentation.

To augment my dataset, I first generated speed-pitched variants of each of my existing valid data (waveforms). Each input waveform was transformed using the *resize* function from OpenCV (cv2 in Python) using a randomly distributed stretch/squeeze factor between 0.7 (70% speed) and 1.3 (130% speed). This was repeated 10 times for each input, resulting in a multiplication of the original dataset size by 11 (original + 10 speed variants). The result of this transformation, when seen on a spectrogram (which is what the model actually uses to train), is a "squeezed" (narrower) or "stretched" (wider) image. This emulates the variation in pitch and tempo of a person's voice.

Then, I generated variants of my input waveforms with added white noise. The same principle as the speed augmentation was used (10 randomly augmented variants for each input waveform). This augmentation was applied to the already speed-augmented dataset, in order to get a final dataset that was 111 times larger than the original (1 + (10 * 1 + 1) * 10).

With this augmented dataset, I ran the same model as I had in Deliverable 2. I achieved ~81.82% accuracy on my original non-augmented dataset, and approximately 77% validation accuracy during training. The resulting confusion matrix for this augmented dataset is the following (see next page):

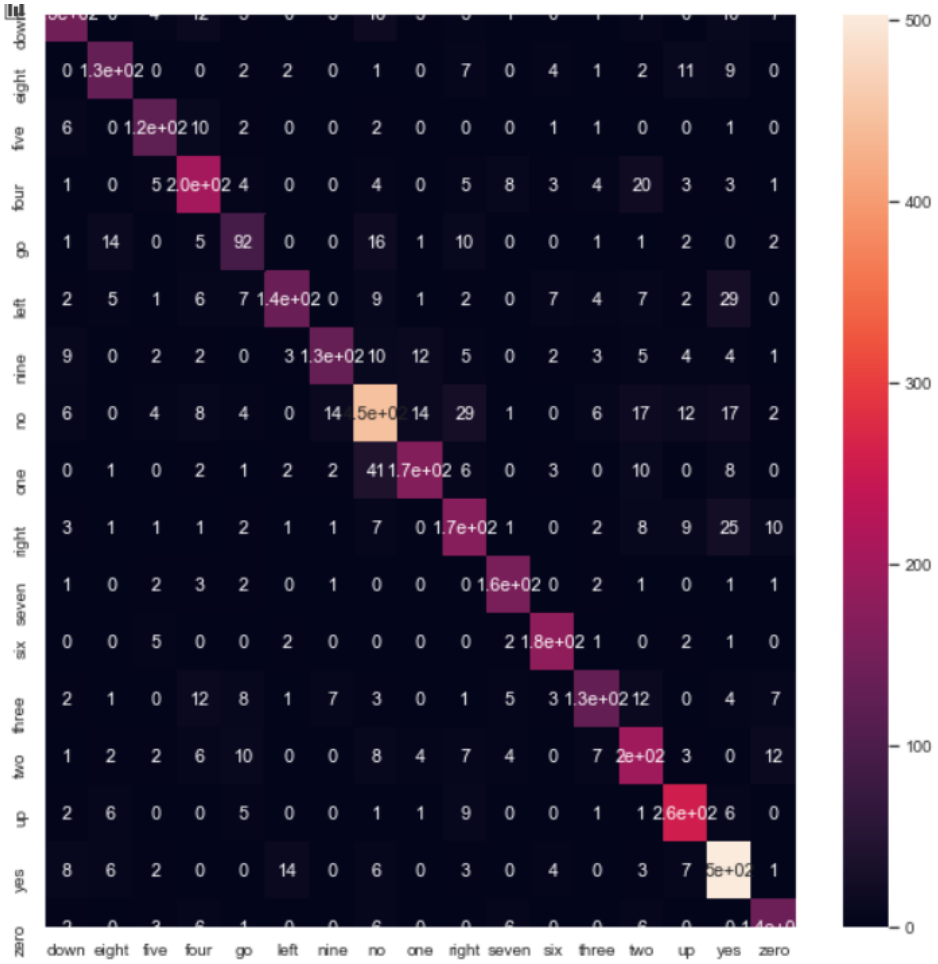

Figure 1: Confusion matrix for the model trained on the augmented dataset.

From this confusion matrix, we can see that the results are much better than for the non-augmented dataset.

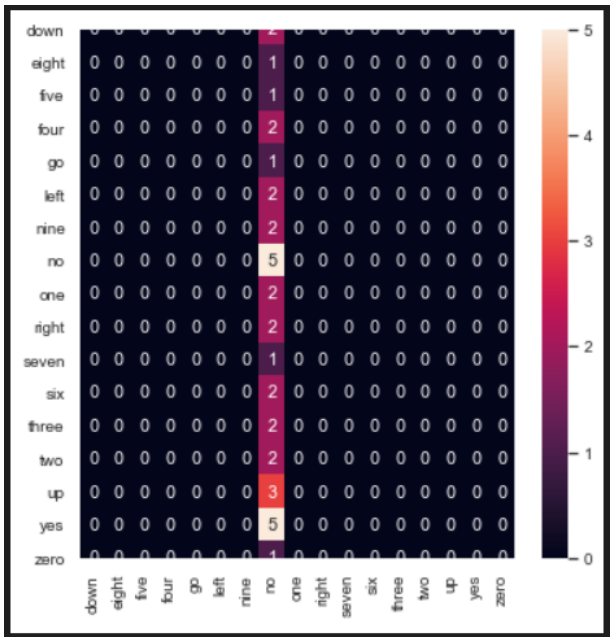

Figure 2: Confusion matrix for the model trained on the non-augmented dataset.

I believe that the non-augmented dataset was insufficient in size to properly affect the number of weights that the model contains, which was clearly alleviated by using the much larger, augmented dataset. However, from the confusion matrix of both datasets, we see that the distribution of data is far from uniform. This can negatively affect the performance of the model in real-life scenarios (receiving completely different, unseen data), as was proven by running it with new, self-recorded sound clips for a few of the commands ('six', 'eight', 'up', 'down', 'three'); I recorded my boyfriend saying those commands. Two of the five sounds were correctly predicted ('six' and 'three'), while the others had varying levels of similarity to the actual sound ('eight' got predicted as 'right', which is fairly close sounding, but 'down' was predicted as 'zero', which is very far and is highly unlikely according to the confusion matrix). It's interesting that some of the results were correctly or closely predicted, because they were done using a typical young male voice, while the input data from TORGO was made with elderly and atypical (dysarthria) speech. The less well predicted sounds probably lack enough representation in the original dataset ('zero', for instance). A possible explanation for the performance of the model on typical speech, despite being trained *only* on atypical speech, would be that:

1. Some of the original atypical speech inputs were not that distorted compared to typical speech;
2. The category defining features were kept even in the distorted (atypical speech) data, and the model was able to identify those features, which are present in typical speech;

Even considering those points, it is quite likely that the model does not actually perform as well as the accuracy and confusion matrix suggest; the reason being that a small set of data was re-used multiple times with small, simple manipulations to artificially increase the size of the dataset. This does not necessarily indicate that the model is able to generalize well, in fact, it could be overfitting the tiny, original dataset and performing "well enough" on the simple variations of itself. To truly test the "real-life" performance of the model, a person with dysarthria should be recorded to obtain new sound clips, that could then be fed into the model and more closely analyzed and standardized. One of the big challenges of using the TORGO dataset was really the non-standardized nature of the data (variation in volume, microphone used, phoneme format, etc.).

2. Final demonstration proposal

For the final demonstration, I would like to implement a webpage showcasing some sound clips from dysarthria patients (coming from the TORGO dataset) that are hard to understand through simply listening. I could then integrate my model on the webpage so that the user can select one of the soundclips and feed it as an input to the model, which will then predict what the command was (and hopefully give the correct answer). Interesting metrics could be displayed, for example the 3 most probable answers for the

given input, as well as the confidence level of the model for those answers (categories/labels). Another page or section could illustrate how it's achieved (basically, how the model "sees" the sound features on a spectrogram using a CNN). An annotated spectrogram could be provided to show the phonemic features of the sound. Another possibly interesting idea could be to allow the user to record a one-second sound clip (from the accepted speech commands) and have the model predict the command, although this might prove disappointing if the model is actually not that good at recognizing typical speech.

The technologies used would be a simple web server made with Python (either Flask or Django) that is able to run the model underneath, and the webpage itself could be basic HTML + CSS + some JavaScript (to send requests to the server), or a more complicated framework if I have the time to learn some of it. I have very limited web development knowledge, so I will need to investigate more and follow some tutorials to achieve what I want to implement. I do not exactly know how I would (or could, depending on the knowledge I will gain) present the information that I have suggested in the previous paragraph, thus I don't really have any diagrams or things of the sort to present yet. I guess I will see over the next few weeks!