Real-time gesture recognition using microphone data
Stanford EE292D

Abhipray Sahoo (abhipray@stanford.edu)

April 2020

1 Motivation

Common methods for gesture recognition on embedded devices often use accelerometer or other dedicated hardware like radar [1]. Gestures that they can recognize range from finger taps with accelerometer to hand waving with radar. The gestures can be used as user input. A microphone could be a low-cost alternative sensor that can detect similar gestures.

The NeoSensory Buzz (Figure 1) is a wristband for the hard of hearing. It maps sound information in real-time to vibration patterns on the wrist. It has a microphone and three buttons for inputs. The three buttons have several functions assigned to them and are not sufficient for the the full range of device features. An example interaction is to display battery level with the LEDs when a gesture like double tap is performed. In order to add more inputs, we can use the microphone to detect "double tap", "drumming", "slide up", "slide down" and "finger snap".

Figure 1: The Neosensory buzz wristband

A neural network is well suited to the problem since it can learn to be robust to a large variety of noises. In this project, we build a model that can detect these five gestures in real-time on-device. We train the model to be robust to different kinds of noise.

2 Data collection

To collect examples of the different gestures, we built on top of an open-source project called Open-Speech recording [2]. It is a web-app with two screens. First screen presents instructions on how to perform each of the gestures. Second screen starts presenting a new prompt randomly every 2 seconds. The Neosensory buzz device exposes its microphone as a USB microphone which can be accessed by the web-app. The user is requested to perform 25 gestures in one session, five per gesture. They can review their recordings by listening and when ready, upload their recordings to a storage bucket.

In total, we collected roughly 400 examples per gesture type. 360 of them were collected by a single person on a single device. Additionally, 100 negative examples were recorded that included sounds of single taps, rubbing, speech, music etc. We also collected long background examples recorded through the Buzz microphone and played out of loudspeakers. These included white noise, pink noise, dish-washing, riding an exercise bike and motor noise. Motor noise is the noise produced by the motors on the Buzz wristband. The last of these is important because motors can be very loud.

2.1 Quality control

We extracted the loudest 1 second audio segment from each of the gesture recording using the algorithm described here [3]. We then visually inspected each recording spectrogram and removed ”bad” examples.

3 Tiny-conv experiments

Using the training scripts for tensorflow’s speech commands tutorial, we trained a ”tiny-conv” model on the gesture data. The model uses spectrograms on the mel-scale with 40 filterbank energies. Results of the training are shown below.

Train	Validation	Test
76%	92.6% (N=149)	95.4% (N=151)

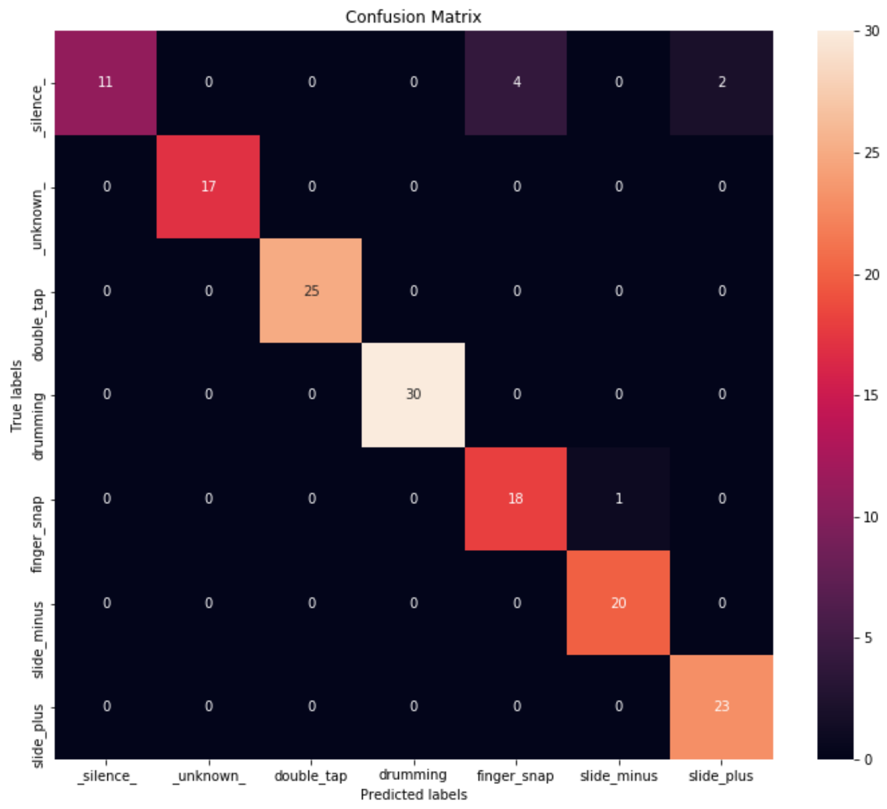

Figure 2: Confusion matrix of tiny conv test results

The model however does not work well in a streaming setting on real-time microphone audio. To measure streaming accuracy, we created a 10 minute audio clip that overlays gesture clips randomly on to a background track. We used the 10 minute clip to measure streaming accuracy.

Only 26.0% of the examples were predicted correctly, 45.5% incorrectly with 8.0% false positive rate. As shown by this test, the tiny-conv model does poorly in a streaming setting. One speculation for why this might be is that the tiny-conv model front-end might be optimized for speech rather than the gestures which are short impulses.

4 RNN experiments

We designed a recurrent neural network model trained on synthesized data to improve performance during streaming.

4.1 Data synthesis

Since we only have 400 examples per gesture type recorded in quiet conditions, we artificially synthesized a large number of training examples. We do this by overlaying gesture clips on top of background tracks along with negative example clips. The steps are listed below:

1. Trim any silence at the end of each gesture clip.
2. Choose a random background track and randomly select a 14 second segment.
3. Choose between 1-5 gesture examples and 1-5 negative examples. Overlay them randomly on the background track.
4. For the target labels, assign gesture label for 16ms at the end of the clip and 250ms after the end of each inserted clip.
5. For the training features, extract 40 log mel-filterbank energies with 16ms hop length and 32ms frame length.
6. Low-pass filter the mel-filterbank energies with exponential moving average for 2 seconds. Subtract the mean estimate from the mel-filterbank energies.
7. Drop the first 2 seconds of training features and labels since the moving average estimate has not converged

4.2 Model

The model uses a linear transformation to reduce dimensionality followed by a stack of three recurrent layers. A final fully connected layer with softmax activations server as the output layer. Batch normalization layers are placed after the fully connected and recurrent layers to help increase training convergence speed.

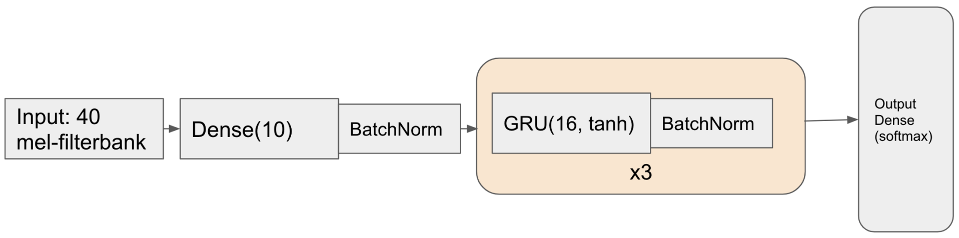

Figure 3: Model architecture

The model has a total of 5,369 parameters which takes up 33,444 bytes as a tflite micro model. For contrast, tiny-conv model takes 30,720 bytes.

4.3 Training results

The model is trained on the synthesized data with 32,768 examples for training, 4096 for validation and test each using Adam optimizer. We also found that training with L2 regularization ($\lambda = 1e^{-4}$) on all the layers helped the network generalize better during streaming evaluation.

The frame-wise performance is presented below. The overall accuracy is 94.29%.

	precision	recall	f1-score	support
background	0.964774	0.982396	0.973505	3.203928e+06
other	0.792443	0.612425	0.690900	1.871250e+05
double_tap	0.803048	0.930214	0.861966	3.795900e+04
drumming	0.839213	0.831624	0.835401	3.704200e+04
finger_snap	0.847889	0.747467	0.794517	3.691400e+04

Continued on next page

	precision	recall	f1-score	support
slide_minus	0.686032	0.515526	0.588681	3.919300e+04
slide_plus	0.731130	0.584373	0.649566	3.774300e+04
accuracy	0.949214	0.949214	0.949214	9.492137e-01
macro avg	0.809218	0.743432	0.770648	3.579904e+06
weighted avg	0.946032	0.949214	0.946647	3.579904e+06

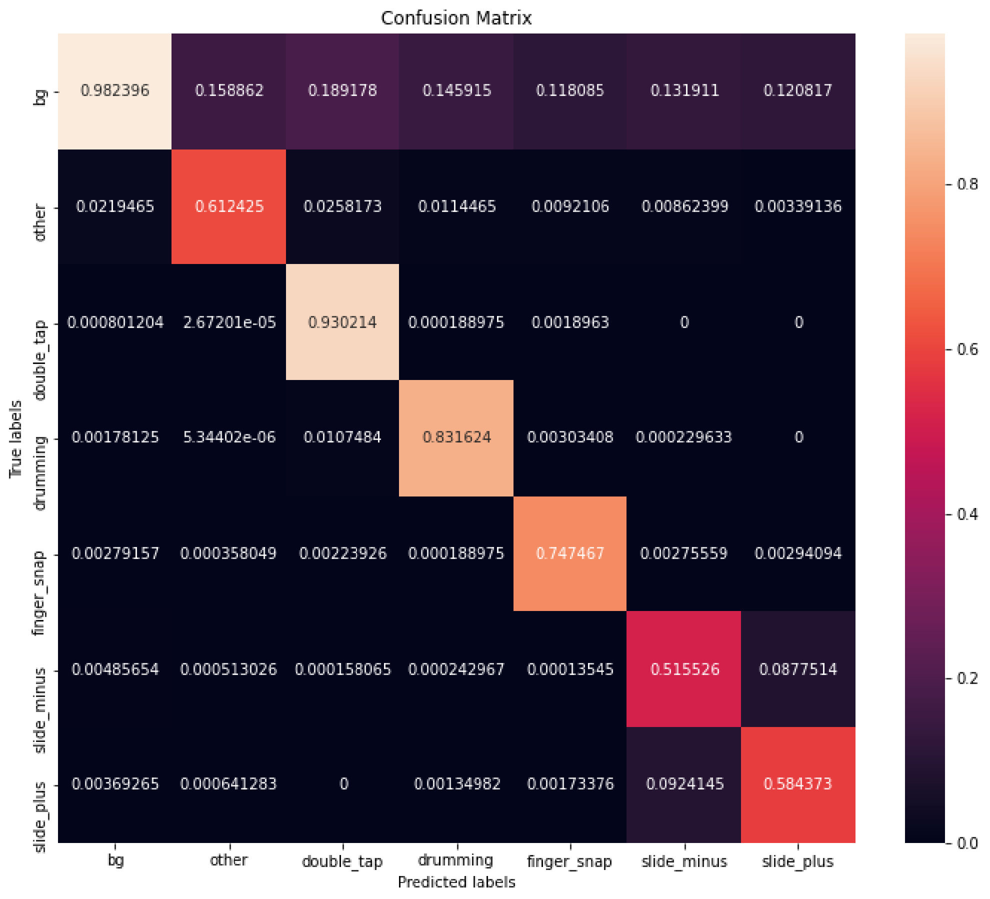

Figure 4: Confusion matrix normalized to show recall

4.4 RNN model conversion using tf lite micro

RNNs have limited support in tf lite framework. Keras RNN models that maintain state do not convert to a tf lite model directly. As a work around, we re-wrote the model to expose the states externally as inputs and outputs. At each timestep, the state inputs are filled with the outputs from the previous time-step.

We do not perform any quantization of weights or activations. All data types are 32-bit floating point.

4.5 Post-processing

The model emits a probability distribution over the different classes at each time step. We pick the most likely gesture with argmax. Since we have labeled our dataset by assigning every frame for 266ms at the end of a gesture clip as the output label, we can use the number of consecutive frames as another confidence measure. If the most likely gesture was detected for the last 10 consecutive frames, then we emit the detected gesture. We suppress subsequent outputs for 500ms since we do not expect another gesture to take place.

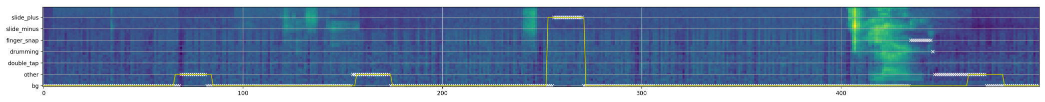

Figure 5: Example 12 second clip with model predictions. White crosses are model predictions and yellow line is the ground truth.

4.6 Evaluation

For evaluation in silent conditions, we ran the tf-lite micro model on a Macbook pro with audio coming directly from the Neosensory Buzz wristband over USB. In this scenario, we tested real-time performance by performing each gesture 20 times and recording how many times the gesture was detected correctly.

Double tap	Drumming	Finger snap	Slide minus	Slide plus
19/20	18/20	19/20	5/20	14/20

When the model was run on device directly, it performed poorly because of motor noise. In normal operation, the motors produce noise and their firing is correlated to the sounds in the environment. The noise affected the recall for all the different gestures as well as false positive rate.

5 Conclusion

We demonstrated that an RNN model can be used effectively as a audio based gesture recognition engine on an embedded device. The model was trained on synthesized data and achieved a recognition accuracy of 95% on test data. We saw good streaming performance in noise-free conditions. However more work needs to be done to make recognition robust to different kinds of noise especially motor noise.

References

[1] Soli: Home. https://atap.google.com/soli/. (Accessed on 04/28/2020).

[2] petewarden/open-speech-recording: Web application to record speech for an open data set. https://github.com/petewarden/open-speech-recording. (Accessed on 04/28/2020).

[3] petewarden/extract_loudest_section: Trims .wav audio files to the loudest section of a given length. https://github.com/petewarden/extract$_{l}oudest_{s}ection.(Accessedon06/08/2020)$.