Machine Learning is Fun Part 6

Yuan An

In the last article, the machine translation is introduced. In the part six of Geitigey's article [1], it is talking about speech recognition. It's built into phones, game consoles and smart watches. It's even automating homes. *E.g*. TmallGenie (see Fig. 1)developed by Alibaba A.I.Labs is a smart speech terminal, like your personal assistant. It can allows you to order pizza, get a weather report or buy trash bags.

Figure 1. TmallGenie

The reason is that deep learning finally made speech recognition accurate enough to be useful outside of carefully controlled environments. And Andrew Ng has long predicted that as speech recognition goes from 95% accurate to 99% accurate, it will become a primary way that we interact with computer.

In the last article, we think machine learning as a black box. But machine learning isn't always a black box. What speech recognition would do is feed sound recordings into a neural network and train it to produce text, as shown in Fig. 2.

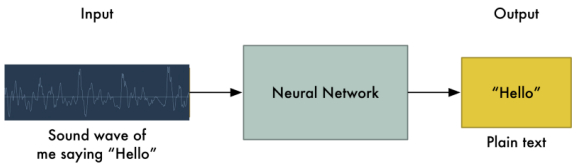

Figure 2. How speech recognition might be?

But there is a big problem that speech varies in speed. One person might say "hello" very quickly and another person might say "heeellllllooooo" very slowly. It produced a much longer sound file with much more data. However both sound files should be recognized as exactly the same text–"hello". But aligning audio files of various lengths to a fixed-length piece of text automatically turns out to be pretty hard. But we can use some special tricks and extra precessing in addition to a deep neural network.

The first step is turning sounds into bits. As we all know, sound is transmitted as waves (see Fig. 3).

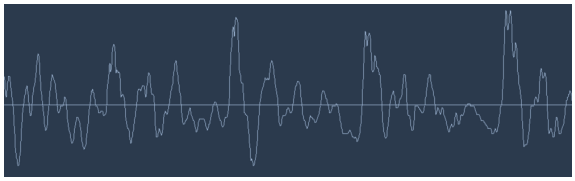

Figure 3. A waveform of "Hello"

Sound waves are two-dimensional. At every moment in time, they have a single value based on the height of the wave. To turn this sound into numbers, record of the height of the wave at equally-spaced points as shown in Fig. 4.

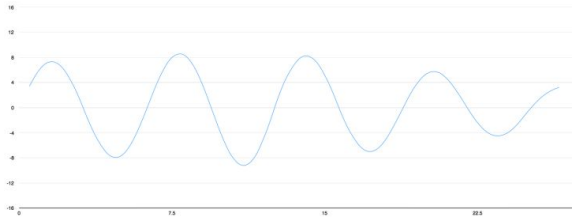

Figure 4. Sampling a sound wave

This is called sampling. Take a reading thousands of times a second and recording a number representing the height of the sound wave at that point in time.

Thanks to the Nyquist theorem [3], we can use math to perfectly reconstruct the original sound wave from the space-out samples–as long as we sample at least twice as fast as the highest frequency we want to record.

The next step is pre-processing sampled sound data. After sampling, we could feed these numbers right into a neural network. But trying to recognize speech patterns by processing these samples directly is difficult. So we can doing some pre-processing on the audio data.

First, group sampled audio into 20-millisecond-long chunks. Then plot those numbers as a simple line graph gives a rough approximation of the original sound wave for that 20 millisecond period of time (see Fig. 5).

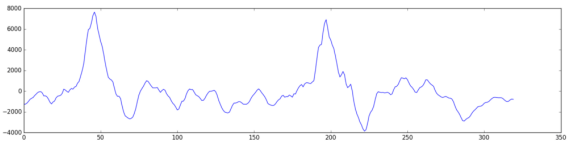

Figure 5. The simple line graph of sampled audio sound wave.

This recording is only 1/50 of a second long. But it is a complex mish-mash of different frequencies of sound. They contain some low sounds, some mid-range sounds, and even some high-pitched sounds sprinkled in.

To make this data easier for a neural network to process, we are going to break apart this complex sound wave into its component parts. Well break out the low-pitched parts, the next-lowest-pitched-parts, and so on. Then by adding up how much energy is in each of those frequency bands (from low to high), we create a fingerprint of sorts for this audio snippet.

We do this using a mathematic operation called a Fourier transform. It breaks apart the complex sound wave into the simple sound waves that make it up. Once we have those individual sound waves, we add up how much energy is contained in each one (see Fig. 6).

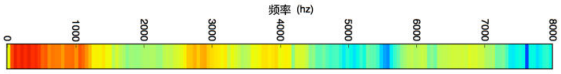

Figure 6. There is a lot of low-frequency energy and not much energy in the higher frequencies. Thats typical of male voices.

If we repeat this process on every 20 millisecond chunk of audio, we can get a spectrogram like Fig. 7.

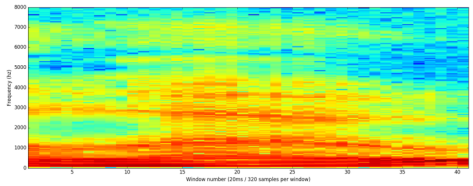

Figure 7. The total spectrogram.

In the spectrogram, we can see musical notes and other pitch pattern in audio data. A neural network can find patterns in this kind of data more easily than raw sound waves. So this is the data representation well actually feed into our neural network.

Now that we have an easy-to-process format audio, it can be fed into a deep neural network (see Fig. 8. The input of the neural network is 20 millisecond audio chunks. For every little audio slice, it will try to figure out the letter that correspond the sound currently being spoken. Well use a recurrent neural networkthat is, a neural network that has a memory that influences future predictions. Thats because

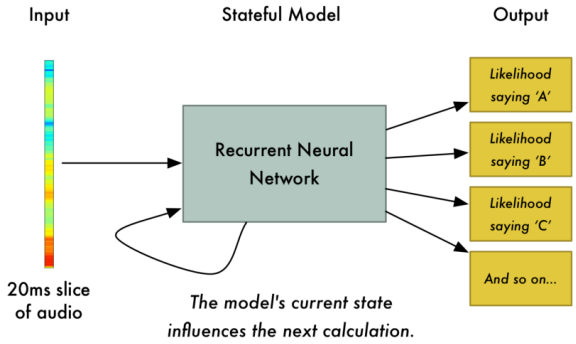

Figure 8. The neural network model of speech recognition

each letter it predicts should affect the likelihood of the next letter it will predict too.

Now we can roughly predict what people said. But there are many problem to be solved. We can learn more about it from the paper of Graves and Gomez [2]. This is about an algorithm to deal with variable-length audio which is called Connectionist Temporal Classification (CTC).

References

[1] A. Geitgey. Machine learning is fun. `https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471`.

[2] A. Graves and F. Gomez. Connectionist temporal classification:labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376, 2006.

[3] Wikipedia. Nyquist-shannon sampling theorem. `https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem`.