**Why Deep Learning is the Best Approach for Speech Recognition**

[Automatic speech recognition](https://blog.deepgram.com/what-is-asr/) isn’t new. It has its origins in Cold War-era research with narrow military implementations, which was followed in the 1960s, 70s, and 80s by developments from leaders like Marvin Minsky and research funded by [DARPA](https://en.wikipedia.org/wiki/DARPA). However, it wasn’t until the 1990s that researchers saw real progress thanks to government-funded projects like the Wall Street Journal Speech Dataset. Even then, these small datasets of around 30 hours of audio only yielded accuracies of about 30-50% in a research setting. Continued developments in speech technology have led to a variety of improvements and consumer use cases that we’re all familiar with today-Alexa, Siri, telling the automated bank system that you need a PIN, etc. But if you’ve ever used any of these speech recognition tools, you know that they’re far from perfect. That’s because they rely on an old-fashioned way of doing speech recognition that has its roots back in those original experiments in the 1960s.

In this blog post, we’ll walk through the old-fashioned way of doing speech recognition-because it’s the one that’s still used by most companies today-and then show why the new way, which relies on end-to-end deep learning to process speech, is far superior.

## **The Old Way: An Acoustic Model, a Pronunciation Model, and a Language Model-Oh my!**

The smallest units of sound in spoken language are called phonemes. For example, “cat” has three phonemes: an initial “k” sound, a middle “a” vowel like in “apple”, and a final “t” sound. In the old way of doing ASR, you start by identifying the phonemes in a recording, and then trying to assemble clumps of phonemes into possible words. Next, you look for how those possible words might fit together to make grammatical sense. Finally, you hack all those possibilities down to one ‘transcript’. The components of this system are called the acoustic model, pronunciation model, and language model with beam search.

* The **acoustic model** takes a representation of the audio signal-usually as a waveform or spectrogram-and tries to guess a phoneme probability distribution function over timeboxed windows of 10-80 ms throughout the entire recording. Essentially, the output is a huge lattice of possible phonemes as a function of time rather than simply a phonemic transcription.
* The **pronunciation model** then takes the phoneme lattice as its input and tries to guess a *word* probability distribution function over time windows. The output of this step is a huge lattice of possible words as a function of time.
* A **language model** is then used in conjunction with a beam search. The model takes the word lattice as its input and cuts down all the possibilities it thinks are less likely until it arrives at the final transcription. In addition, it uses a beam search: at every time step, the search throws away all possibilities below its cutoff (called the beam width), never to be seen or thought of again.

Although this old way of building speech recognition models is intuitive to humans, and is motivated to some extent by how linguists think about language, it’s highly lossy to a computer. At each step in this process, your models have to make simplifying assumptions to fit the computations in memory or finish within the lifetime of the universe-not kidding. There are just too many combinations and permutations for the models to return results if they consider all of the possibilities. This is why, for instance, the language model portions are typically very limited trigram language models. The *tri-* in trigram means “three” and indicates that the model only looks back two words to see if the current word makes sense in context. That might only be half of a sentence-or less! These simplifications are rampant and result in a performance-limited, pipelined approach for optimizing sub-problems at each step of the process, rather than an end-to-end approach that can simultaneously optimize across the entire problem domain. This creates three major problems with traditional methods.

### **Problems with the Traditional Approach**

There are three big problems with the old-fashioned method of speech recognition: it’s slow, it’s inaccurate, and it’s brittle. The slowness makes it expensive and time consuming. The inaccuracy makes the traditional methods ineffective and frustrating to use, especially for enterprises and domains that require high levels of accuracy, like the health and legal fields. And the brittleness makes engineers afraid to change any code for fear that the house of cards will come crashing down.

#### Slow

The traditional methods are slow because they rely on unoptimized heuristic approaches, which use compute and memory resources inefficiently. These approaches can only process around 0.5-2 streams per CPU core. This can lead to long turnaround times when it comes to providing results-often so long that some applications, like real-time chatbots, are simply not possible using these methods.

#### Inaccurate

The old method is inaccurate because the models lack expressiveness and capacity. Expressiveness is a measure of how complicated of a world the systems can model while still maintaining accuracy. Capacity is a similar measure of how much knowledge a model can retain. Traditional systems are shallow in this sense. They have no hope of covering everything extremely well, so they either cover most areas with meager success or a narrow domain with some success.

#### Inflexible

The old method is brittle because the systems are extremely complicated and inflexible. It takes a team of 20 engineers a year to set up a system that only starts to get adequate performance. So, they leave it alone and hope for the best. Attempting to modify or improve the system will only end in defeat as the surface area of the problem overcomes the team. This is why traditional speech recognition providers only serve one model (maybe two or three, but certainly not hundreds or thousands) and refuse to customize for their customers. The cost is too high with the old hydra methods (cut off one problem head and three grow back).

## **The best way: End-to-end deep learning for speech recognition**

The good news, if you’re looking for a speech recognition solution, is that it doesn’t have to be this way! Although the old way of doing things is still used by most providers, there is an alternative that’s fast, accurate, and flexible-an end-to-end deep learning (E2EDL) model.

### **Fast**

An end-to-end model can be better optimized for run-time execution. Deep learning, specifically, utilizes the same set of mathematical operations ([tensor math](https://en.wikipedia.org/wiki/Tensor)) as implemented on graphics cards (GPUs). This means that E2EDL models are the very fastest implementations available. On the other hand, traditional speech stacks are composed of multiple sub-problems (less surface area for optimization) and cannot use accelerating computing resources (thus forcing them onto the general-purpose CPU). E2EDL on GPUs achieves over 300 streams per GPU, meaning that results are returned much faster to customers-so much so that they are often surprised and delighted. Deepgram’s customers often think they must have done something wrong, but no, it’s just that fast.

### **Accurate**

E2EDL models have much greater capacity and enjoy a compression efficiency that allows all parts of the network to learn in cohesion, all as one organism. As a result, these models can optimize across the entire problem space at once-from the input audio features all the way through the transcript production. The result is an expert model that achieves much higher accuracy and continues to get better while training without “topping out.”