arm

The Power of Speech: Supporting Voice-Driven Commands in Small, Low-Power Microcontrollers

By Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra, Arm

Borrowing from an approach used for computer vision, we created a compact keyword spotting algorithm that supports voice-driven commands in edge devices that use a very small, low-power microcontroller.

The Power of Speech: Supporting Voice-Driven Commands in Small, Low-Power Microcontrollers

With the widespread use of voice-activated virtual assistants, such as Apple's Siri, Amazon's Alexa, Microsoft's Cortana, and the Google Assistant, voice has become an everyday way to interact with electronics. We're talking to our devices more than ever, using speech to initiate searches, issue commands, and even make purchases.

There are a number of reasons why using your voice to control an electronic device is a good idea. It's almost always easier to talk than to type, and having voice activation means electronics can be controlled entirely hands-free. Devices can also have highly intuitive user interfaces, even if they don't have a display, and can operate in challenging environments where a keyboard would be problematic.

In developing the mechanisms for voice activation, designers have found that neural networks, which are modeled on brain function, are better at understanding us than traditional speech-processing networks. This is partly because neural networks, also known as "deep learning" algorithms, can be trained and can learn from experience, so they become more accurate over time.

Making Voice-Driven Commands Easier to implement

Neural networks are algorithms that improve over time, by learning from accumulated data. They're typically used in computationally-intensive applications that deal with very large amounts of data, including robot kinematics, statistical analysis, computer vision, and speech recognition. Neural networks can perform many millions of operations in a single second, and are usually executed on advanced, high-powered processors.

Performing full-on voice recognition means being able to recognize an entire dictionary's worth of words and phrases, and many of the devices we use every day don't have enough memory and onboard processing power to support full-on voice recognition with sufficient accuracy. So how do things like smartphones, smart speakers, smart home controllers, and other devices, operating at the edge of a network, support voice-initiated commands? By using processing power in the cloud along with a low-power, less processor-intensive algorithm, called KeyWord Spotting (KWS), which resides in the edge device. The edge device's keyword spotting algorithm listens for a specific set of keywords that then initiates a connection to the cloud.

Several such keywords have already become part of the daily lexicon. Saying "Hey, Siri" to an iPhone, "Ok, Google" to an Android device, "Alexa" to an Amazon Echo, or "Hey, Cortana" to a Microsoft device is putting keyword spotting to use. Having heard the keyword, the device connects to the cloud and starts transmitting whatever's said next to a backend processing system. The backend system uses its built-in, heavy-duty processing power to make sense of the sounds, create a response, and then send that response back to the device.

Keyword Spotting Needs Low-Power, Compact Operation

Keyword spotting (which is also called keyword detection) has to be very efficient. For completely hands-free operation, a voice-activated system needs to listen continuously for commands. A person may speak to the device at any time, and the device needs to be ready to respond immediately. Always-on operation reduces energy efficiency, so the implementation needs to consume as little power as possible.

Keyword spotting also has to work on its own, in the edge device, without interaction from the backend system. That's because transmitting continuous audio to the cloud introduces privacy concerns and bandwidth issues. There are already billions of devices accessing cloud services, and many billions more are predicted to come online soon. Even with only a portion of those devices supporting voice-initiated commands, the amount of network traffic required to support continuous listening would be overwhelming, and would take bandwidth away from other tasks that don't involve speech. Heavy network usage and lack of bandwidth could also impair performance, by introducing latency and causing frustrating delays in the system's response to a command.

Bringing Keyword Spotting to More Devices

Today's keyword spotting algorithms are designed to give quick responses while working in a variety of edge devices. But many of the devices that we would like to have support speech recognition – think small appliances, personal devices, wearables, and other portables – are so limited in their computing and power capacity that they either don't deliver enough accuracy with today's keyword spotting algorithms or simply can't support them at all.

To make keyword spotting viable in a wider range of edge devices, there's a need to optimize a neural network for keyword spotting such that it can perform at an acceptable level in a broader range of systems, including those that are battery operated, have very limited power budgets, and use a very small microcontroller with only limited memory and computing capability.

Our work shows that it is, indeed, possible to create a keyword spotting solution that meets these requirements. We found that a certain architecture, more commonly used for computer vision than for speech recognition, called a Depthwise Separable Convolutional Neural Network (DS-CNN), offers the performance and scalability we're looking for. In terms of accuracy, our DS-CNN model achieved an accuracy of 95.4% in our evaluation. This was 10% better than our Deep Neural Network (DNN) model, based on an architecture commonly used for speech recognition.

This paper summarizes how we got these results.

Choosing our Microcontroller Targets

To give ourselves a baseline hardware configuration to work with, we looked at small, off-the-shelf microcontrollers commonly used in edge devices, and chose a few as target examples. Looking at the characteristics of the processor core, the size of the SRAM block, and the amount of embedded flash memory, we selected from the Arm Cortex-M series a number of processor cores with different compute capabilities running at different frequencies. Table 1 gives a sample list of Cortex-M based microcontroller platforms and the hardware constraints they present.

Arm Mbed™ platform	Processor	Frequency	SRAM	Flash
Mbed LPC11 U24	Cortex-M0	48 MHz	8 KB	32 KB
Nordic nRF51-DK	Cortex-M0	16 MHz	32 KB	256 KB
Mbed LPC1768	Cortex-M3	96 MHz	32 KB	512 KB
Nucleo F103RB	Cortex-M3	72 MHz	20 KB	128 KB
Nucleo L476RG	Cortex-M4	80 MHz	128 KB	1 MB
Nucleo F411RE	Cortex-M4	100 MHz	128 KB	512 KB
FRDM-K64F	Cortex-M4	120 MHz	256 KB	1 MB
Nucleo F746ZG	Cortex-M7	216 MHz	320 KB	1 MB

Table 1. Typical off-the-shelf Arm Cortex-M based microcontroller development platforms
(Table 1 from the original paper)

For our purposes, the Cortex-M0 microcontrollers are the "smallest" options, since they have the lowest clock frequency and have the least amount of memory. The Cortex-M3 microcontrollers, with their somewhat higher frequency and larger memory, can be considered "medium" options, while the Cortex-M4 and Cortex-M7 can be considered "large" microcontroller options. It's important to note that the Cortex-M4 and Cortex-M7 microcontrollers have integrated functions for Digital Signal Processing (DSP), which can help the neural networks that support keyword spotting run more efficiently.

Our aim was to develop a solution for keyword spotting that could be scaled to work in any of these microcontroller platforms. To do this, we focused on two hardware constraints.

- **Limited memory footprint**
  As shown in Table 1, the microcontrollers have memories that offer from a few dozen to a few hundred kilobytes of space. The entire neural network model, including input/output, weights, and activations, had to fit within this very tight memory budget.

- **Limited computing resources**
  Since the keyword spotting function is always on, the algorithm places real-time constraints on the total number of operations for neural network functionality. That is, if most or all of the computing power is being used to support continuous listening, there's not much, if anything, left over for operating the

voice-recognition functions once they're triggered. Reducing computational complexity, so as to minimize execution time, is a way to limit the amount of computing resources needed to run keyword spotting.

Choosing our Neural Networks

Before selecting the neural networks we would train and evaluate, we did some background research. We found that, although there are many neural network models for keyword spotting presented in technical papers, it's difficult to make a fair comparison between them as they are all trained and evaluated on different proprietary datasets with different input speech features and audio duration. To create an "apples to apples" comparison, we made our own models, based on various neural network architectures, and trained them all on the same dataset. Testing neural networks that were taught the same words let us make valid comparisons for performance.

Here are the neural network architectures we explored most thoroughly, and tested on our sample configurations.

- DNN: Deep Neural Network
- CNN: Convolutional Neural Network
- Basic LSTM: Basic Long Short-Term Memory
- LSTM: Long Short-Term Memory
- GRU: Gated Recurrent Unit
- CRNN: Convolutional Recurrent Neural Network
- DS-CNN: Depthwise Separable Convolutional Neural Network

All these architectures are commonly used for speech recognition, except for DS-CNN. DS-CNN is an alternative method from computer vision. Google has developed a family of small, low-latency, low-power models for computer vision, called MobileNets, based on DS-CNN. MobileNets offer the kind of scalability, for computer vision, that we were aiming for with speech recognition, so we added it to our list. Figure 1 is a diagram of the DS-CNN architecture.

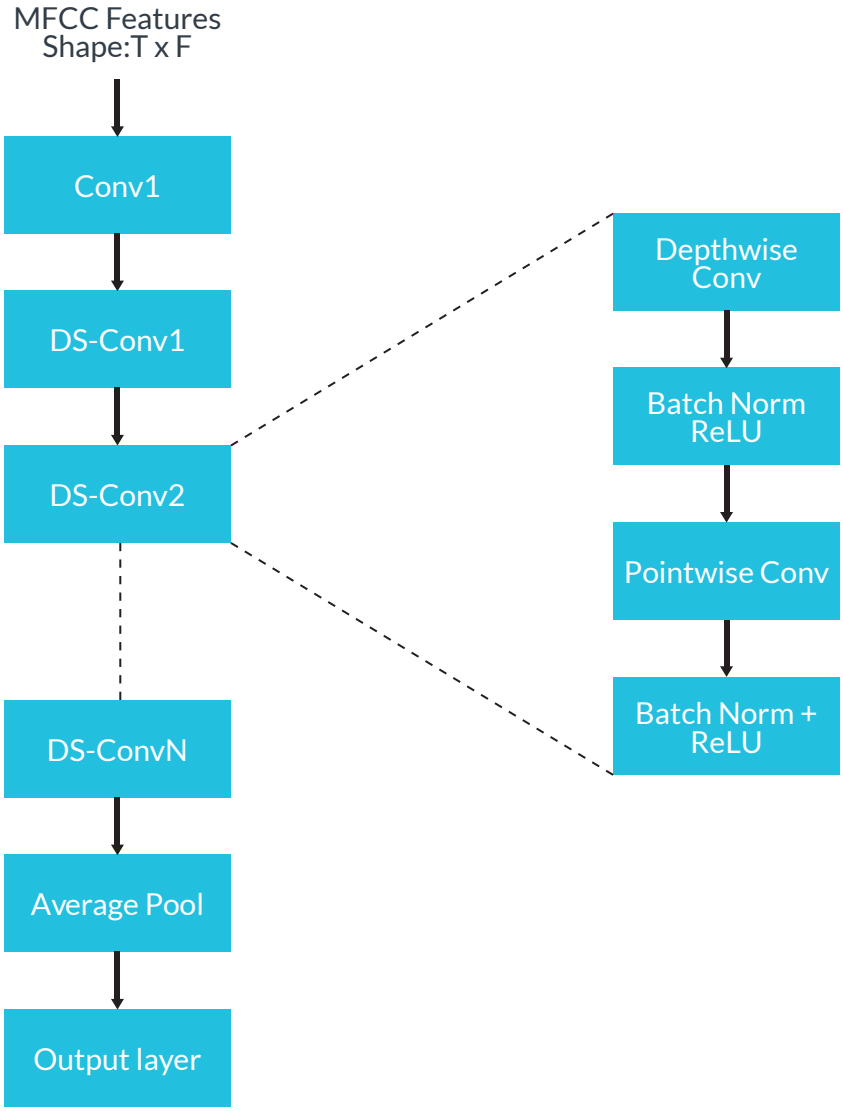

Figure 1. DS-CNN architecture frequently used in computer vision
(Figure 4 from the original paper)

Training our Models

Using our chosen neural network architectures as starting points, we created our own implementations of models for keyword spotting and trained each on the Google speech commands dataset, a limited dataset intended as a guide to help designers get familiar with speech-recognition systems.

The Google speech commands dataset includes 65,000 one-second-long audio clips of 30 essential words, as said by different people. Each audio clip includes just one word, such as yes, no, up, down, left, right, on, off, stop, go, or the numbers one through ten. To help sift out keywords, a series of "unknown" words, along with audio clips of silence, are also included. It's a good toolset for creating simple voice controls, using keyword spotting, without having to build everything from scratch.

Reducing Computational Complexity

When we started investigating neural networks, we found that the research emphasis for keyword spotting has generally been on maximizing accuracy within a small memory footprint, and not on reducing the number of operations per inference. In other words, present-day implementations only addressed one of our two key hardware constraints. As a result, we began experimenting with optimizations, to see if we could increase accuracy within a small computing footprint.

Trained model weights are usually represented by 32-bit floating point numbers, which can represent a wide range of values, including very small and very big numbers. Using a compression technique called quantization, we were able to convert the 32-bit floating point unit to an 8-bit fixed point unit. The 8-bit fixed point unit is more limited, able to represent a much narrow range of values, but we found that it still worked for keyword spotting without losing accuracy. The result is an algorithm for keyword spotting that is computationally less complex, so it needs fewer computing resources to run.

Putting Them to the Test

Having taught all our models to detect the same set of keywords and having quantized the models to 8-bit fixed point numbers, we compared how our models performed, in terms of accuracy, memory footprint, and number of operations per inference.

Based on our chosen microcontrollers, we derived three sets of memory/compute constraints for each of our neural networks. Table 2 specifies these constraints, targeting small, medium, and large microcontrollers. In deriving these categories, we assumed that some amount of resources will be allocated for running other tasks, such as the operating system, inputs/outputs, network communications, and so on. We also assumed that the system runs ten inferences per second with 8-bit weight/activations.

NN size	NN memory limit	Ops/inference limit
Small (S)	80 KB	6 MOps
Medium (M)	200 KB	20 MOps
Large (L)	500 KB	80 MOps

Table 2. Memory/Compute Constraints of our Small, Medium, and Large Neural Networks
(Table 3 of original paper)

Table 3 shows the results of our test cases. The DS-CNN model won out in all three sets of constraints.

NN model	S(80KB, 6MOps)			M(200KB, 20MOps)			L(500KB, 80MOps)		
	Acc.	Mem.	Ops	Acc.	Mem.	Ops	Acc.	Mem.	Ops
DNN	84.6%	80.0k	158.8K	86.4%	199.4K	397.0K	86.7%	496.6K	990.2K
CNN	91.6%	79.0k	5.0M	92.2%	199.4K	17.3M	92.7%	497.8K	25.3M
Basic LSTM	92.0%	63.3k	5.9M	93.0%	196.5K	18.9M	93.4%	494.5K	47.9M
LSTM	92.9%	79.5k	3.9M	93.9%	198.6K	19.2M	94.8%	498.8K	48.4M
GRU	93.5%	78.8k	3.8M	94.2%	200.0K	19.2M	94.7%	499.7K	48.4M
CRNN	94.0%	79.7k	3.0M	94.4%	199.8K	7.6M	95.0%	499.5K	19.3M
DS-CNN	94.4%	38.6k	5.4M	94.9%	189.2K	19.8M	95.4%	497.6K	56.9M

Table 3. Summary of Neural Network Performance
(Table 5 of original paper)

Conclusion

Our entire keyword spotting application occupies only about 70 KB of memory, so it fits within the 80 KB limit of even the smallest microcontrollers we evaluated. Our DS-CNN model, which delivers the highest accuracies across every size of microcontroller, shows that it's possible to produce an effective application for keyword spotting, with good scalability, even when targeting very small, very low-power microcontrollers.

Learn More

A more technical description of our work is available at

https://community.arm.com/processors/b/blog/posts/high-accuracy-keyword-spotting-on-cortex-m-processors.