Professor Marvin: Efficient Keyword Spotting via Knowledge Distillation and Quantization

Arman Dave
armdave@mit.edu

Julian Hamelberg
jshx@mit.edu

Abstract

We present Professor Marvin, a proposed system that uses a low-memory, energy-efficient device that runs a compressed keyword spotting algorithm, which has no ability to connect to the cloud. When the device detects the keyword, it sends a signal to the home assistant device to awaken, parse the user's commands, and connect to the cloud. In this way, the home assistant is only awake when the user explicitly commands. Professor Marvin has been implemented by compressing a keyword spotting Convolutional Neural Network (CNN) system through the use of quantization and knowledge distillation to reduce memory footprint and FLOPs. The compressed model has been tested, reaching accuracy nearly as high as the baseline, and works on a Raspberry Pi architecture and can be deployed to edge devices.

1 Introduction

1.1 Motivation

The use of home assistant devices, such as Amazon's Alexa and Google Home, has become increasingly popular in recent years. These devices use keyword spotting algorithms to identify and respond to voice commands given by the user - in particular, the wake word that triggers the device e.g. "Alexa" or "Hey Google". However, the use of these algorithms has raised privacy concerns, since the devices are constantly listening to the user for the wake word and may be collecting sensitive information without their knowledge or consent in the process.

One possible solution to the privacy issue is to run all Natural Language Processing (NLP) algorithms on-device and never send user data to the cloud; however, for the time being, Large Language Models such as BERT Devlin et al. [2018] that can effectively understand and serve complicated user requests (e.g. "What's the weather in Los Angeles tomorrow?") are far too large to run on a local device. For effective usage, home assistant devices need to touch the cloud - it is a question of limiting how often they reach out to the cloud.

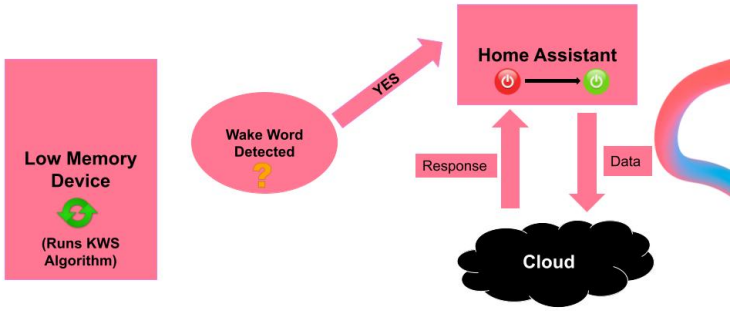
1.2 Contribution

To our knowledge, the system that we propose has not been explored in the literature. In this system, a low-memory, energy-efficient device runs a compressed keyword spotting algorithm and has no ability to connect to the cloud. When this device detects the keyword, it sends a signal to the home assistant device to awaken, parse the user's commands, and connect to the cloud. In this manner, the home assistant that touches the cloud is only on when the user requires.

In order to implement this system, we compress a standard keyword spotting Convolutional Neural Network system based on the work of Saha: Our compression techniques include quantization and knowledge distillation. In this process, we shrink a model with size 10.7 MB and 1 percent false positive rate to a model of size 439 KB and minimal loss in accuracy. The compressed model has

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Figure 1: Baseline Architecture



been tested and works on a Raspberry Pi architecture and can be deployed to edge devices at this time.

2 Related Works

Keyword spotting is a rich field that has benefited from the advances in machine learning over the past decade. In 2014, researchers made great strides in replacing highly complex Filler Hidden Markov Models with relatively simple Deep Neural Net architectures Chen et al. [2014]. This model architecture was lightweight enough to run on mobile devices and could perform inference with latency of 10 ms with high precision. Researchers made improvements the following year by creating a similarly sized Convolutional Neural Network that improved performance by forty percent.

However, in recent years the brunt of research has shifted to to bulkier transformer-based models such as Wav2Vec Schneider et al. [2019]. While models like Wav2Vec have the advantage of not needing to preprocess sound wave data with log filters such as log Mel filterbank, these models pay the price with large and complex architectures. Wav2Vec is a 1.2 GB model and compression techniques that maintain good accuracy, to our knowledge, have only decreased the model to 200-300 MB.

We draw inspiration from Saha and Zhang et al. [2017]. Saha creates a 10 MB CNN in tensorflow, while Zhang's Hello Edge system finds optimal architectures via NAS and compresses them using quantization. In our approach, we seek to compress Saha's model using knowledge distillation and quantization.

3 Professor Marvin System Overview

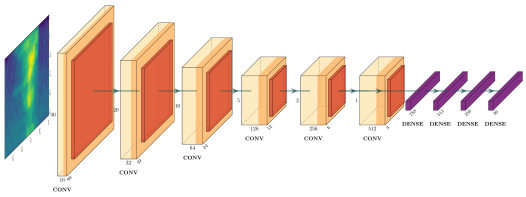
3.1 Baseline

For Professor Marvin's baseline architecture, we chose a relatively simple Convolutional Neural Network that would lend itself well to knowledge distillation. The heart of the architecture is a sequence of progressively larger Conv2D layers, starting from 16 channels and doubling at each layer over 5 layers to a final 256 output channel layer. The outputs of each convolutional layer pass through a BatchNormalization, ReLU, and Dropout layer. We then pass these outputs through Dense layers of size 512 and 256 and finally to 31 (representing the number of classes), at which point they pass through the softmax function. We take the max node from the final 31 nodes as the prediction.

3.2 Compression through Quantization

We use the TFLite library to run Integer quantization with float fallback. This means the weights are 8 bit integers, but inputs and outputs remain floating points, and floating point multiplication is performed between weights and inputs. Since floating point multiplication is still performed, we use knowledge distillation to reduce the number of FLOPs in addition to model size.

Figure 2: Baseline Architecture



3.3 Compression through Knowledge Distillation

More aggressive forms of integer quantization, such as 4 bit and 2 bit, often lead to poorly performing models. As such, we opted to use knowledge distillation to build a smaller model with the same parameters. In the baseline model, our convolutional layers increase in size from 16 output channels to 256 over 5 layers, doubling the number of channels at each step. We tested reducing each layer by 1/2, 1/4, and finally 1/8 (e.g. for 1/8 we would have 5 layers ranging from 2 output channels to 32) in the student model. We found that reducing the channels by 1/4 kept results nearly the same while reducing further to 1/8 caused a drastic drop in results - a jump from 3 false positives to 14. As such, in the full end-to-end Professor Marvin pipeline, we distilled our teacher model to a student model with 1/4 the channel outputs in each convolutional layer.

4 Evaluation

4.1 Experiment Setup

For the purposes of the experiment, we use the Google Speech Commands Dataset Warden [2018], which consists of 65,000 one second long utterances divided amongst 30 classes. We divide the dataset into a 80/10/10 training/validation/test split. For our wake word, we choose "Marvin," since "Marvin" is one of two names in the dataset. While we recognize that the dataset imbalance would cause issues in a production system, we instead choose to focus on methods that can effectively compress and speed up a neural net. There exist several methods to address the issue of dataset imbalance, such as using an SVM classifier on the output of the neural net Saha or generally augmenting the dataset Kotsiantis et al. [2006], when the system is ready to be productionized.

Our primary metrics are accuracy and model size: We measure accuracy primarily in terms of a confusion matrix (the number of true positives, false positives, false negatives, and true negatives). Many real-world home assistants prioritize minimizing the number of false positives in order to minimize user privacy breaches Combs et al. [2022]. For model size, we measure both the size of the model file and the number of FLOPs. As a secondary metric, we focus on inference speed. For Natural Language Processing systems, the metric often used is called Real-Time Factor (RTF), which is defined as the length of the time taken to process an utterance divided by the length of the utterance.

Our experiment is divided into three phases. In the first, we measure the performance of our baseline architecture on these metrics. In the second, we measure the performance of the quantized model, and in the third, we measure the end-to-end performance - the full Professor Marvin architecture.

4.2 Baseline Model

The baseline architecture is measured at 10.7 MB and has around 34.9M flops. RTF is measured at 1.08. The confusion matrix is presented below in Figure 3.

The false positive rate is very low, which is highly desirable, but the model itself is over 10 MB and so compression could allow the KWS system to run on a smaller more energy efficient device.

4.3 Quantized Model

The quantized model is measured at 929 KB and has around 34.9M flops. RTF is measured at 1.04. The confusion matrix is presented below in Figure 4.

Figure 3: Baseline Results

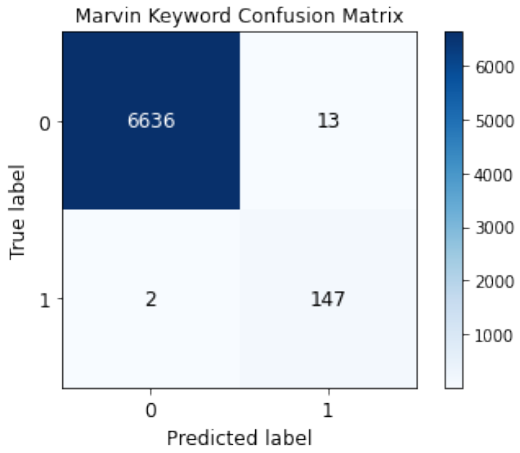
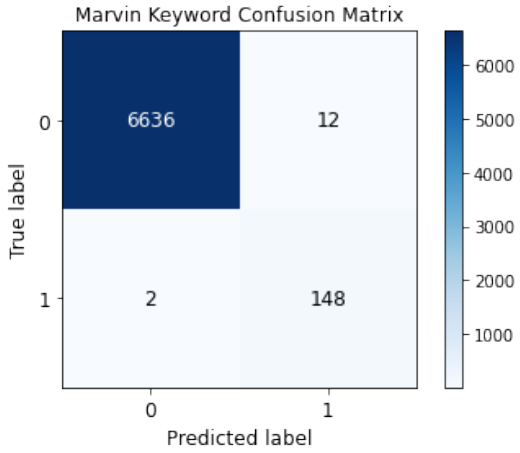


Figure 4: 8-bit Quantized Results



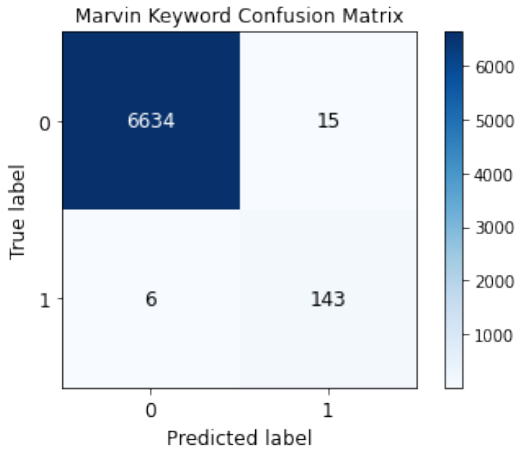
The results of the quantized are nearly identical to those of the baseline architecture, which is promising. However, this also means that we can afford to compress the model even further before we begin to see meaningful decline in accuracy, which we do in the next section.

4.4 Knowledge Distillation and Quantization

The end to end model is first trained through knowledge distillation and then quantized to 8-bit weights. The end-to-end model is measured at 439 KB and around 9.0M flops. RTF is measured at 1.02. The confusion matrix is presented below in Figure 5.

We believe the extremely low false positive and false negative rate shown by Professor Marvin demonstrate that more effort should be placed into developing simple neural net architectures rather than trying to quantize gigabyte-sized transformer based models. These simple NN architectures can be further compressed to run on very low-memory devices, which can be very useful in preserving privacy while still getting use out of home assistants.

Figure 5: End-to-end Results



5 Conclusion

5.1 Limitations

While we were still able to produce some interesting results, we were slowed down when Colab updated their default Python to version 3.8 (https://github.com/googlecolab/colabtools/issues/3246issue-1462476281) on Nov 30. Some issues with this update no longer allowed us to use our Tensorflow dataloader (tf.data.Dataset) on Colab even when reverting to Python 3.7. As a result, we were forced to set up an environment on our local computers to run our models. These machines were far slower than Colab's and so we couldn't do as many experiments as planned.

5.2 Future Works

For future works, we would like to look further into the effects of quantization-aware training. From our results, we did not see a decrease in performance after quantization which may indicate that we can quantize further using different techniques. We also hope to implement our entire Professor Marvin pipeline by integrating it with Mycroft for the full voice assistant experience.

References

G. Chen, C. Parada, and G. Heigold. Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091. IEEE, 2014.

M. Combs, C. Hazelwood, and R. Joyce. Are you listening?–an observational wake word privacy study. *Organizational Cybersecurity Journal: Practice, Process and People*, (ahead-of-print), 2022.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

S. Kotsiantis, D. Kanellopoulos, P. Pintelas, et al. Handling imbalanced datasets: A review. *GESTS international transactions on computer science and engineering*, 30(1):25–36, 2006.

V. Saha. Spoken keyword spotting. URL `https://github.com/vineeths96/Spoken-Keyword-Spotting/blob/master/docs/report.pdf`.

S. Schneider, A. Baevski, R. Collobert, and M. Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Y. Zhang, N. Suda, L. Lai, and V. Chandra. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017.