INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Binary Speech Features for Keyword Spotting Tasks

Alexandre Riviello, Jean-Pierre David

Department of Electrical Engineering
Polytechnique Montréal, Canada
{alexandre.riviello, jpdavid}@polymtl.ca

Abstract

Keyword spotting is a classification task which aims to detect a specific set of spoken words. In general, this type of task runs on a power-constrained device such as a smartphone. One method to reduce the power consumption of a keyword spotting algorithm (typically a neural network) is to reduce the precision of the network weights and activations. In this paper, we propose a new representation of speech features which is more adapted to low-precision networks and compatible with binary/ternary neural networks. The new representation is based on the log-Mel spectrogram and models the variation of power over time. Tested on a ResNet, this representation produces results nearly as accurate as full-precision MFCCs, which are traditionally used in speech recognition applications.

Index Terms: speech recognition, keyword spotting, deep learning, quantization, speech preprocessing, log-Mel spectrogram, convolutional neural networks.

1. Introduction

Speech recognition, as a consumer electronics feature, has gained much popularity in recent years. Many devices, such as the Google Home or the Amazon Echo, make use of it as a central feature. The algorithms used to detect utterances can vary a lot depending on the type of task at hand. For instance, in a Large Vocabulary Continuous Speech Recognition (LVCSR) task, speech is decomposed to fundamental units such as phonemes or characters. A linguistic and acoustic model are then combined to predict the pronounced word. In contrast, simpler tasks, such as Keyword Spotting (KWS), only detect a specific set of words. The utterances don't need to be decomposed into phonemes and the algorithm can be trained using entire words as inputs. A system like the Amazon Echo will perform a KWS task when attempting to detect a keyword like "Alexa" and will then likely perform a LVCSR task when trying to understand spoken requests.

The rise of speech recognition has been possible due to advances in deep learning which developed and popularized the use of neural networks. In fact, most modern implementations of speech recognition algorithms use Deep Neural Networks (DNNs) [1]. However, improving the accuracy of speech detection has a cost: DNNs are computationally expensive and energy intensive. A typical DNN can contain millions of parameters which must be stored in some kind of memory. Fast memory access and millions of multiplications are the main factors behind their computational cost [2]. The power requirements are often so high that these algorithms often need to run on remote servers. Considering the finite battery-life and the computational inefficiency of smartphones, this design choice is justified. Nevertheless, the use of cloud computing can raise privacy concerns because data is continuously collected and sent to private servers. Reliability concerns must also be considered since remote processing depends on the reliability of the connection. Speech recognition algorithms would gain to run locally on a smartphone or a domotic device. This can be done by reducing their computational requirements, which is one of the main objectives of this paper.

Many techniques have been proposed to reduce computational requirements. A common method used is to quantize the weights and activations of the networks. By modifying these values to shorter bit representations, memory access costs can be greatly reduced. When the weights and activations are binary {0,1} or ternary {-1,0,1}, multiplication operations are no longer necessary since convolutions can be performed with combinatorial logic. Courbariaux et al. demonstrated that it was possible to obtain comparable results to full-precision networks when training with binarized weights and activations for datasets such as MNIST and CIFAR-10 [3] [4]. Other techniques include weight pruning, which eliminates redundant neurons. With this technique, Srinivas et al. managed to eliminate 85% of the parameters before obtaining a drop of approximately 1% in accuracy on the MNIST classification task [5]. This paper will focus on quantized networks.

One of the challenges of such networks is to find an appropriate representation for the input data. In the case of BinaryNets [4], the input image is represented with 8-bit fixed point values. Therefore, the first convolution layer has 8-bit inputs. Only the following layers receive binary values. XNOR-Nets [6] and improved BinaryNets [7] also ignore quantization for the input data. However, a single convolution layer with binary weights might not be able to extract all the potential features contained in the input image. Directly reducing the precision of the input image may also destroy useful information. With the intent of addressing this issue, this paper will introduce a low-precision data representation that can be directly used by a quantized network for KWS.

At first, we will reduce the precision of the input image representing speech features and see how the accuracy is affected. Afterwards, we will present our new representation which models the variation of power over time. The following sections will be divided as follows: Section 2 presents some related works in the field of KWS. Section 3 explains traditional speech representations. Section 4 details the proposed algorithms to produce new representations. Section 5 describes our test setup. Results are reported in Section 6. Finally, Section 7 concludes this work and proposes future research directions.

2. Related works

Modern implementations of KWS algorithms either use sequence to sequence models such as Long Short-Term Memory (LSTM) based networks [8] because they can capture the temporal progression of utterances or Convolutional Neural Net-

http://dx.doi.org/10.21437/Interspeech.2019-1877

work (CNN) based models [9] since the preprocessed input can be considered an image representing sound over time-frequency axes. Other variants include ResNets which are CNNs with skip connections [10]. On a 10-word classification task, Tang et al. obtained over 95% accuracy using a ResNet [11]. Qiu Lin et al. then proceeded to improve the same ResNet by generating many models of reduced sizes through hyper-parameter optimization [12]. The most precise model obtained an accuracy of 96.8% and required less than half of the parameters of the previous model. In both experiments, the Google Speech Commands Dataset [13] was used to train and test the networks. This dataset includes 65000 sound files containing 30 distinct 1-second utterances of words such as "yes", "no", "up" or "down". 10 words were selected, and 2 extra categories were added to the softmax output of the network: silence and random noise. This ResNet and dataset will be used in this paper to test out different preprocessing approaches to speech.

As previously mentioned, most implementations of quantized networks use 8-bit precision inputs as input images [3] [4] [6] [7]. The first convolution uses binary weights and applies bit-wise operations to the input image. To our knowledge, no other paper has attempted to reduce the precision of the input image for speech to binary or ternary values. KWS hardware implementations, such as the one proposed by Zhang et al., use 8 bits to represent the input image [14].

3. Background

3.1. Traditional speech representations

To represent speech data, traditional algorithms, such as Hidden Markov Models Gaussian Mixture Models (HMM-GMMs) used Mel Frequency Cepstrum Coefficients (MFCCs) [15]. GMMs required decorrelated inputs to model the data correctly. Hence, Discrete Cosine Transforms (DCTs) are capable of decorrelating inputs and are used to compute MFCCs. However, with the introduction of DNNs, which can extract complicated correlations found in input data, the DCT operation is no longer necessary. Simply using the log-Mel spectrum coefficients should not deteriorate results. For this reason, this paper will focus on the log-Mel spectrogram.

3.2. Computing the log-Mel spectrogram and MFCCs

The steps to compute the log-Mel spectrum coefficients are as follows: a) Apply a pre-emphasis filter to a sound wave (typically sampled at 16 kHz); b) Divide the wave into 25-30 ms frames with 10 ms shifts between each frame; c) Zero-pad each frame to obtain a sample count equal to a power of 2 (typically 512); d) Apply a Hamming Window and a real FFT to each frame; e) Perform a dot-product between the resulting frame and each Mel filter (typically 40 filters); f) Compute the logarithm of each value, resulting in 40 coefficients. The steps are illustrated in detail in Figure 1. In this paper, the Python library Librosa v0.6.2 was used to compute the Mel spectrogram [16]. The pre-emphasis step was not part of the preprocessing. The sampling rate, window size and quantity of filters can vary from one implementation to another. To compute MFCCs, one must simply apply a DCT operation on each frame of the log-Mel spectrogram.

4. Proposed speech representations

In this paper, two distinct speech representations are presented. The first is a quantized version of the log-Mel spectrogram and the second is a power-variation spectrogram. The former is explained in detail in Section 4.1 and the latter in Section 4.2. The power-variation spectrogram is also subdivided into two distinct sections introducing a 1-channel and 2-channel representation.

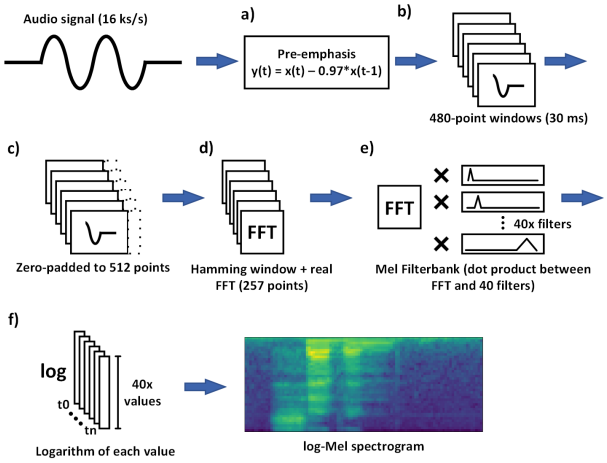

Figure 1: *Steps to compute the log-Mel spectrogram of a speech signal or sound wave.*

4.1. Reducing the precision of the log-Mel spectrogram

The first attempt at finding a low-precision representation was to simply reduce the precision of the log-Mel spectrogram. Since the power of the signal is already represented on a logarithmic scale, a linear quantizer is used to separate different levels of intensity.

Each utterance in the Google Speech Commands Dataset is 1-second long. Using 40 Mel filters with 10 ms frame shifts, the resulting spectrogram image is of size 40x101 pixels. The code used to compute the log-Mel spectrogram can be found in a GitHub repository produced by Tang et al. [11] [17]. Unlike the original code, we used a wider bandpass filter covering frequencies from 0 to 8 kHz. This offered better results. The maximum value over all the images often revolved around the same value delta_power. For this reason, we added an offset to the data so that every pixel value ranged from 0 to delta_power (20 in this case). The image was then rescaled to cover values between 0 and 255 and floored to the closest integer, creating an unsigned 8-bit representation. Depending on the desired bit width, a given number of Least Significant Bits (LSBs) was ignored. The computations are summarized in Algorithm 1.

Algorithm 1 Quantized log-Mel spectrogram

1: *data* ← compute log-Mel spectrogram() // 40x101 image
2: *data* ← data - (max(data) - delta_power)
3: *data* ← max(0, data)
4: *data* ← floor(data*(255/delta_power))
5: *data* ← data $\gg$ (8-desired_bit_width)

4.2. Power-variation representation

Reducing the precision of the log-Mel spectrogram reveals that 8-bit representations hold more information than what is needed. However, using a 2-bit representation is not an optimal solution. Since the value plateaus for a good section of the image, most of the information is repetitive (in the context of a convolution with 3x3 filters). A more interesting representation

would seek to describe the contours of the plateaus or simply the variation of power between two time frames. The idea of calculating variations is also inspired by the use of delta-coefficients in many speech recognition systems. Kumar et al. offer examples showing that noise tends to be stationary compared to speech [18]. In other words, measuring the change in power of sound may isolate important information.

4.2.1. *Ternary 1-channel representation*

Variations between two time frames can easily be represented using ternary logic. In our case, we equate an increase in power to 1, a decrease to -1 and constant power to 0. We start off by remapping the data to values between 0 and 255, exactly like in Algorithm 1. We then set all reference values (40 values representing every Mel filter output) to the first data time frame. If the values of the second frame are greater than the reference values by a certain threshold, then the output is 1 and the reference value is updated. If the values are smaller than -1*threshold, then the output is -1 and the reference value is also updated. This process is repeated for all subsequent frames. The algorithm is explained in detail in Algorithm 2. For our tests, the threshold was set to 12. This value was determined by performing an exhaustive search.

Algorithm 2 Ternary 1-channel representation

```
output_image ← array[40x101 of '0s']
data ← compute log-Mel spectrogram() // 40x101 image
data ← data - (max(data) - delta_power)
data ← max(0, data)
data ← floor(data*(255/delta_power))
ref_value ← data[:, 0]

for i in range 0 to 100 do
    variation ← data[:, i+1] - ref_value
    for j in range 0 to 39 do
        if variation[j] >threshold then
            output_image[j, i] ← 1
            ref_value[j] ← data[j, i]
        else if variation[j] <-threshold then
            output_image[j, i] ← -1
            ref_value[j] ← data[j, i]
        else
            output_image[j, i] = 0
        end if
    end for
end for
```

4.2.2. *Binary 2-channel representation*

A second representation was introduced which divided the previous image into 2 channels. Essentially, the increases in power are represented by {0,1} in the first channel and the decreases by {0,1} in the second channel. This gives a purely binary representation, which is interesting from a hardware perspective since every multiplication operation can be performed with a single logic gate.

5. Experiments

Every input representation was tested using the ResNet architecture proposed by Tang et al. [11]. The quantity of feature maps for each respective layer has changed to match the EdgeSpeechNet-A model of Qiu Lin et al. [12]. The network is illustrated in Figure 2.

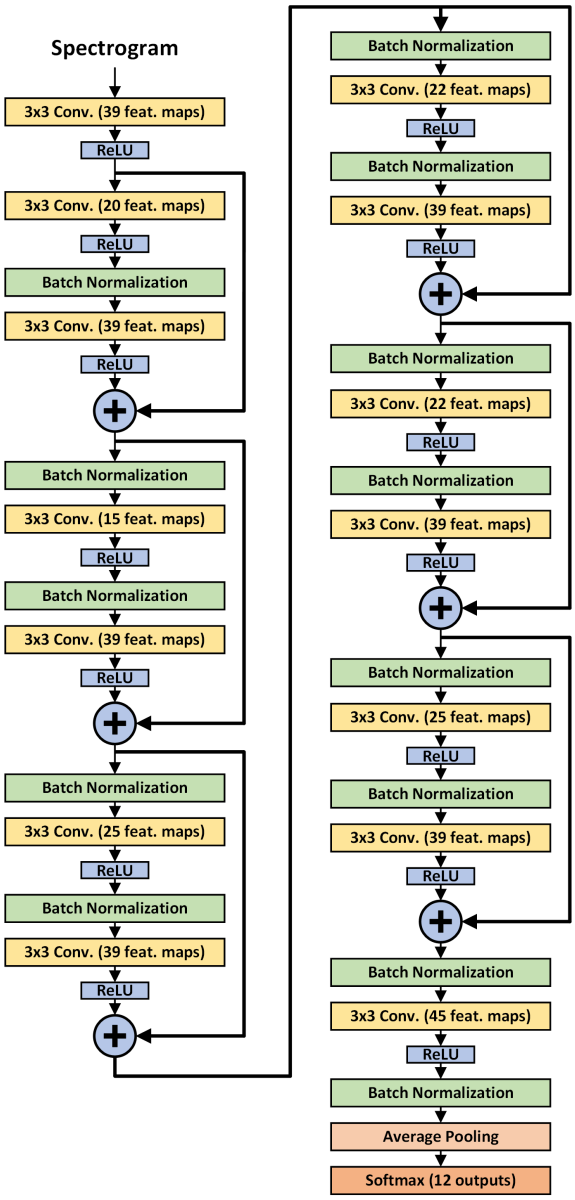

Figure 2: *Architecture of EdgeSpeechNet-A [12]. This ResNet is used to test the different input representations proposed in this paper.*

The classification accuracy of the four quantized log-Mel spectrograms and the ternary power-variation spectrograms were tested using the ResNet as is. The Google speech commands dataset was used for training and testing [13]. All weights and activations were represented with floating-point values. In each case, the network was trained for 26 epochs. The learning rate was set to 0.1 for approximately 12 epochs. It then decreased by a factor of 10 every 6 subsequent epochs. Noise and random shifts were added to the input data. Training was done with mini-batch sizes of 64. All other network hyperparameters (other than the quantity of feature maps) remained unchanged compared to what was found in the res15 model of the GitHub repository of Tang et al. [11] [17]. Each result is

an average of 5 training runs. The preprocessing and network were implemented entirely in NumPy and PyTorch. The network was trained on a GeForce GTX 1060 Graphics Processing Unit (GPU).

6. Results

6.1. Low-precision log-Mel spectrogram

An example of the log-Mel spectrogram is illustrated in Figure 3 for representations quantized to 8, 4, 3 and 2 bits respectively. Even when using a 2-bit representation, the image seems to have retained most of its characteristics.

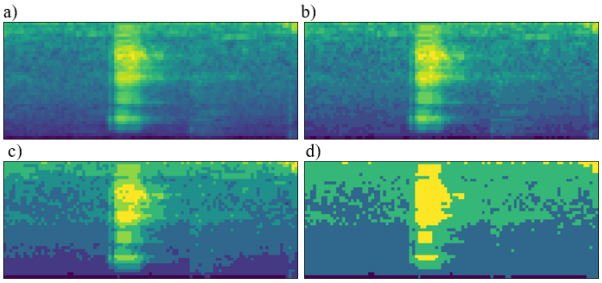

Figure 3: *Quantized representations of the log-Mel spectrogram for the word "up". a) 8-bit rep. b) 4-bit rep. c) 3-bit rep. d) 2-bit rep.*

The results for the quantized inputs are summarized in Table 1. The accuracy claimed by the EdgeSpeechNet-A model is approximately 96.8% using full-precision MFCCs as inputs [12]. The 8-bit log-Mel spectrogram representation gives roughly the same results. Also, using 8-bit pixels offers a very detailed representation, as can be seen in Figure 3. In fact, the accuracy doesn't change considerably until we reach 3 bits. This may indicate that much of the information contained in these images is unnecessary, at least, when attempting to detect a set of keywords.

Table 1: *Accuracy of quantized inputs*

Precision	Representation	Accuracy
Full-precision	MFCCS	96.8%
8 bits	Log-Mel spectrogram	96.8% $\pm$ 0.13
4 bits	Log-Mel spectrogram	96.6% $\pm$ 0.14
3 bits	Log-Mel spectrogram	96.2% $\pm$ 0.51
2 bits	Log-Mel spectrogram	95.2% $\pm$ 0.33

6.2. Power-variation spectrogram

An example of the 1-channel power-variation spectrogram is illustrated in Figure 4. The 2-channel equivalent (not illustrated) has power increases in one channel and power decreases in another. The accuracy of the power-variation spectrogram is reported in Table 2. The representations proposed offer an accuracy drop of only 1% compared to the 8-bit log-Mel spectrogram or MFCCs (95.8% vs 96.8%).

The receiver operating characteristic (ROC) curves were plotted using the False Alarm Rate (FAR) and False Reject Rate (FRR) for the most important cases. For each keyword, the FAR and FRR were computed over sensitivity thresholds ranging from 0 to 1. The curves were then averaged over fixed FAR values. Some key representations are illustrated in Figure 5.

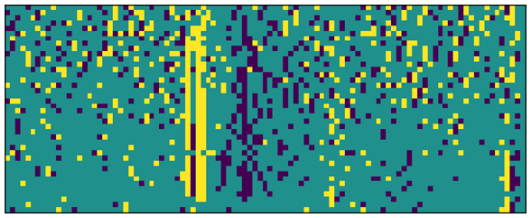

Figure 4: *Ternary 1-channel power-variation spectrogram for the word "up". A yellow pixel is equal to 1, a purple pixel to -1 and a green pixel to 0.*

Table 2: *Accuracy of power-variation spectrogram*

of Channels	Precision (per channel)	Accuracy
1	2 bits	95.8% $\pm$ 0.11
2	1 bit	95.6% $\pm$ 0.32

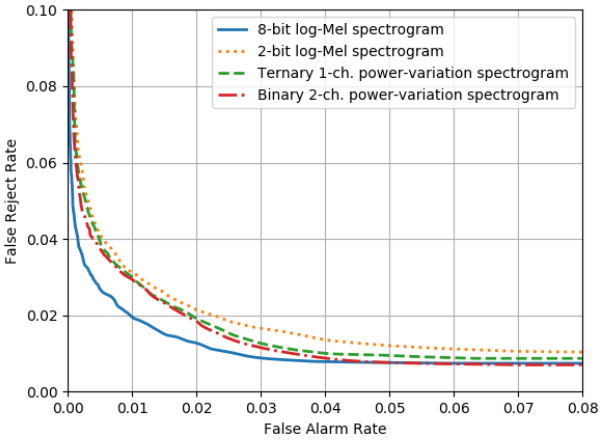

Figure 5: *ROC curves for various input representations.*

7. Conclusion

In this paper, we evaluated the impact of the audio features' precision on the quality of the detection for keyword spotting applications. These features are meant to be used with neural networks that have quantized weights and activations to reduce the energy requirements. A power-variation spectrogram, based on the log-Mel spectrogram, has been proposed. Compared to an 8-bit image representing the traditional log-Mel spectrogram, our representation leads to nearly the same precision when tested on a ResNet with full-precision weights and activations (95.8% vs 96.8%). However, it only requires 1x 2-bit channel or 2x 1-bit channels, which are easily interfaceable with binary or ternary neural networks. Future works will attempt to test this representation using a low-precision network in various noise conditions. Also, it would be interesting to test this representation in an actual KWS task instead of a multi-class classification task.

8. References

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[2] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 1135–1143.

[3] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[4] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[5] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *arXiv preprint arXiv:1507.06149*, 2015.

[6] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, 2016, pp. 525–542.

[7] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "Bnn+: Improved binary network training," *arXiv preprint arXiv:1812.11800*, 2018.

[8] M. Zeng and N. Xiao, "Effective combination of densenet and bilstm for keyword spotting," *IEEE Access*, vol. 7, pp. 10 767–10 775, 2019.

[9] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Interspeech*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.

[12] Z. Q. Lin, A. G. Chung, and A. Wong, "Edgespeechnets: Highly efficient deep neural networks for speech recognition on the edge," *arXiv preprint arXiv:1810.08559*, 2018.

[13] P. Warden, "Launching the speech commands dataset," *Google Research Blog*, 2017.

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

[15] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[16] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[17] R. Tang and J. Lin, "Honk: A pytorch reimplementation of convolutional neural networks for keyword spotting," *arXiv preprint arXiv:1710.06554*, 2017.

[18] K. Kumar, C. Kim, and R. M. Stern, "Delta-spectral cepstral coefficients for robust speech recognition," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4784–4787.