PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Binary Neural Networks for Keyword Spotting Tasks
Auteur: Author:	Alexandre Riviello
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Riviello, A. (2020). Binary Neural Networks for Keyword Spotting Tasks [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5449/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/5449/
Directeurs de recherche: Advisors:	Jean Pierre David
Programme: Program:	génie électrique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Binary Neural Networks for Keyword Spotting Tasks

ALEXANDRE RIVIELLO
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie électrique

Août 2020

© Alexandre Riviello, 2020.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Binary Neural Networks for Keyword Spotting Tasks

présenté par **Alexandre RIVIELLO**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Yvon SAVARIA, président
Jean Pierre DAVID, membre et directeur de recherche
Pierre LANGLOIS, membre

DEDICATION

"I learned very early the difference between knowing the name of something and knowing something."

- Richard P. Feynman

ACKNOWLEDGEMENTS

First of all, I'd like to thank my Professor, Jean Pierre David, for his support and guidance. Throughout my studies, he offered valuable advice and recommendations that have helped me shape this thesis. He was quick to answer my questions and was always available to assist me.

I'd also like to thank Professors Pierre Langlois and Yvon Savaria for accepting to be part of my jury. Their classes on configurable processors and VLSI have made me develop a huge interest for processors and computer engineering.

I also want to thank Siva, Hossein and Moussa for making the lab a fun and dynamic place. I will miss our discussions and wish them all the best.

I would like to thank NSERC and IVADO for their financial support. Their help was much appreciated and was essential to the completion of my degree.

Finally, I'd like to thank my parents who have supported me throughout my academic journey. I hope I made them proud.

RÉSUMÉ

Le repérage de mots clés est une sous-tâche de la reconnaissance vocale qui a pour but de détecter des mots spécifiques dans un fil audio continu. Ce genre de tâche est généralement utilisé par des assistants vocaux ou des appareils de domotique afin d'effectuer des actions telles qu'allumer des lumières. Puisque cette tâche doit fonctionner en temps réel, elle roule généralement de façon locale sur des systèmes embarqués tels que des microcontrôleurs. Depuis quelques années, on a commencé à utiliser des Réseaux de Neurones (RdN) pour effectuer du repérage de mots. Toutefois, les RdN requièrent beaucoup de calculs et d'espace de stockage. Afin de réduire les coûts en calculs des RdN, beaucoup de systèmes emploient de la quantification. La binarisation, qui est une forme draconienne de la quantification, a déjà été utilisée pour des tâches de reconnaissance vocale complexes, mais pas pour du repérage de mots clés. Dans ce travail, nous évaluons l'utilisation des RdN binaires appliqués sur une tâche de repérage de 10 mots.

Les systèmes de reconnaissance vocale conventionnels utilisent le spectrogramme de Mel ou les MFCCs (Mel-Frequency Cepstral Coefficients) pour représenter le son. Afin de s'adapter au RdN binaires, nous proposons une nouvelle façon de modéliser l'audio qui mesure la variation de la puissance dans le temps. Cette représentation est composée de valeurs binaires et offre des résultats de classification supérieurs lorsque testée sur un RdN binaire (comparativement au spectrogramme de Mel). Le RdN binaire utilisé pour évaluer les performances de cette représentation était inspiré de l'architecture ResNet. Sur ce réseau, l'utilisation de la représentation binaire a amélioré la précision de la classification de 1.7%.

En considérant l'objectif de trouver une implémentation matérielle efficace et à basse puissance de RdN pour du repérage de mots clés, nous concluons que les microcontrôleurs ne sont pas de bons candidats pour de la binarisation. Ce type de système embarqué est optimisé pour des opérations s'effectuant sur des valeurs entières de un ou plusieurs octets. Puisque les FPGA peuvent facilement gérer des opérations binaires, une deuxième approche qui utilise ces systèmes a été proposée. Pour ce faire, une heuristique nommée PoET-BiN a été employée. Cette heuristique remplace les couches d'un RdN avec des opérations à base de tables de vérité (LUT). Toutefois, PoET-BiN n'a jamais été validé sur des couches convolutionnelles. Après avoir appliqué cette heuristique sur un réseau binaire convolutionnel, la précision de la classification a diminué de façon considérable. En somme, ces approches ne semblent pas être suffisantes pour exploiter le plein potentiel des réseaux binaires sur ce type de tâche. Il faudrait creuser d'avantage.

ABSTRACT

Keyword Spotting (KWS) is a machine learning task which attempts to detect a small set of words within an audio stream. This type of task is typically used for voice activation of virtual personal assistants or in domotic devices to perform simple actions like turning on the lights. Because of low-latency requirements, KWS usually runs on the edge (smartphones, microcontrollers, etc). In recent years, Deep Neural Networks (DNNs) have been used for this type of task. However, DNNs require many multiply-add operations and storage space. To reduce the computational costs of DNNs, quantization has often been used. Binarization, an extreme form of quantization, has previously been applied to larger speech recognition tasks, but not to KWS. In this work, we evaluate the use of Binarized Neural Networks (BNNs) on a 10-word KWS task.

Traditional voice recognition systems used log-Mel spectrograms or Mel-Frequency Cepstral Coefficients (MFCCs) to represent speech. A new way to extract features which uses binary features is proposed. The representation computes the variation of power over time and is better suited for BNNs. Overall, using this representation on a fully-binarized neural network offered a 1.7% higher classification accuracy compared to the traditional log-Mel spectrogram. The BNN used to test the representation was inspired by the ResNet architecture.

With the objective of finding a precise and hardware efficient implementation of BNNs for KWS, we conclude that binarization is not optimal for current microcontrollers since the hardware is optimized for byte-sized information. In contrast, with the currently existing hardware, FPGAs might be a better fit. To test this, PoET-BiN, a heuristic which maps neural network layers to FPGA Look-Up Tables (LUTs) was used. However, PoET-BiN was never validated on convolutional layers. Applying this heuristic to a binary convolutional network considerably affected the classification accuracy. Overall, the proposed approaches were not enough to exploit the full potential of BNNs for keyword spotting. Future works should seek to optimize these methods.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE OF CONTENTS vii

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF SYMBOLS AND ACRONYMS xiv

CHAPTER 1 INTRODUCTION 1
1.1 Research Objective 1
1.2 Contributions 2
1.3 Thesis Plan 2

CHAPTER 2 BACKGROUND 4
2.1 Speech Recognition 4
2.1.1 Feature Extraction for Speech 4
2.1.2 Evaluation Metrics 7
2.2 Deep Learning 9
2.3 Summary 13

CHAPTER 3 LITERATURE REVIEW 14
3.1 Keyword Spotting 14
3.1.1 Establishing a Common Benchmark: The Google Speech Commands 15
3.1.2 State-of-the-art Architectures 15
3.1.3 Binary Feature Extraction for Keyword Spotting 17
3.2 Binary Neural Networks 18
3.2.1 BinaryConnect: Neural Networks with Binary Weights 18
3.2.2 BNNs: Neural Networks with Binary Weights and Activations 19

3.2.3 BNN Architectures 20
3.2.4 Optimization of BNNs 21
3.2.5 BNNs using Look-Up Tables (LUTs) 24
3.3 Binary Nets and Speech Recognition 25

CHAPTER 4 BINARY SPEECH FEATURES FOR KWS TASKS 28
4.1 Introduction 28
4.2 Related works 29
4.3 Background 30
4.3.1 Traditional Speech Representations 30
4.3.2 Computing the Log-Mel Spectrogram and MFCCs 30
4.4 Proposed Speech Representations 31
4.4.1 Reducing the Precision of the Log-Mel Spectrogram 32
4.4.2 Power-Variation Representation 32
4.5 Experiments 34
4.6 Results 35
4.6.1 Low-precision Log-Mel Spectrogram 35
4.6.2 Power-variation Spectrogram 36
4.7 Conclusion 37

CHAPTER 5 BINARIZED NEURAL NETWORKS AND KWS 39
5.1 Expanding Binary Speech Features 39
5.2 Proposed Binary Neural Network 41
5.2.1 Network architecture 41
5.2.2 Tests and Ablation Study 43
5.3 Exploring Hardware Implementations : PoET-BiN for CNNs 45
5.3.1 Building the Data to Train PoET-BiN on CNNs 45
5.3.2 Replacing a Random Convolution Layer with PoET-BiN 47
5.3.3 Completely Replacing a Network with PoET-BiN using RINC-1 . . . 50
5.3.4 Completely Replacing a Network with PoET-BiN using RINC-2 . . . 51
5.3.5 Merging Convolution Layers 53

CHAPTER 6 GENERAL DISCUSSION 56
6.1 Using the ResNet Architecture 56
6.2 Choosing the Best Input Representation 56
6.3 Is Binarization Worth It (for KWS)? 57
6.4 FPGA Implementations : PoET-BiN for CNNs 58

CHAPTER 7 CONCLUSION 60
7.1 Limitations 61
7.2 Future Research 61

REFERENCES 63

LIST OF TABLES

3.1	Summary of Various Network Results on Google Speech Commands V1	17
3.2	Comparison of full Precision vs Binary Networks and the Effects of Knowledge Distillation [47].	27
4.1	Accuracy of Quantized Inputs	36
4.2	Accuracy of Power-Variation Spectrogram	37
5.1	Accuracy of the Power-Variation Spectrogram with Static Information on a Full-Precision ResNet	40
5.2	Hyperparameters and Calculations of the Proposed Binary Resnet	43
5.3	Accuracy of the Proposed BNN According to the Input Representation.	44
5.4	Ablation study: Accuracy of the Proposed BNN According to Modification.	45
5.5	Accuracy According to the Number of Layers Replaced with RINC-1.	50
5.6	Accuracy According to the Number of Layers Replaced with RINC-2.	52
5.7	Accuracy According to the Number of Layers Replaced with RINC-2 (Network-A).	53
5.8	Accuracy According to the Number of Layers Replaced with RINC-2 (Network-B).	55
5.9	Accuracy of Various Networks Before and After Using RINC Classifiers.	55
6.1	Proposed Binary Neural Network vs DS-CNN Networks.	58

LIST OF FIGURES

2.1 Extracting frames from an audio signal sampled at 16 kHz. 5
2.2 Filters spread out on the Mel Scale. 6
2.3 Example of the log-Mel Spectrogram with 40 filters. The vertical axis represents the various frequency bins (after applying filters). The horizontal axis represents time. 6
2.4 Example of Mel Frequency Cepstral Coefficients (MFCC)s using 13 coefficients. The vertical axis represents MFCCs for a specific time frame. The horizontal axis represents time. 7
2.5 Example of Receiver Operating Characteristics (ROC) curves illustrating False Rejects vs False Alarms. 9
2.6 General structure of a neuron. The inputs are represented with X and the weights with W. 10
2.7 Example of a Multi-Layer Perceptron with a single hidden layer. . . . 11
2.8 Example of the convolution operation performed in a CNN. 12
2.9 Structure of an LSTM cell. Each cell includes a forget, input and output gate. The C values represents the cell state which remembers the information across states. 13
3.1 Forward propagation of Binarized Neural Network (BNN)s. k represents the layer number, r, a full precision or real value and b, a binarized value. 20
3.2 a) Sign function used during forward propagation. b) Gradient of sign function (Dirac impulse). c) Hard tanh function used during backpropagation. d) Gradient of hard tanh (Rectangular function) 21
3.3 Example of a binary dot-product with xnor and popcount operations. The result of the popcount operation is multiplied by 2 and substracted by the number of bits in the input array. This is required since -1s are represented with 0s. 22
3.4 The PReLU activation function. 23
3.5 The Signswish function and its gradient. 23
3.6 PoET-BiN architecture using 8-input LUTs. 25
4.1 Steps to compute the log-Mel spectrogram of a speech signal or sound wave. 31

4.2 Architecture of EdgeSpeechNet-A. This ResNet is used to test the different input representations proposed in this paper. 34
4.3 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. 35
4.4 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. 36
4.5 ROC curves for various input representations. 37
5.1 Converting the log-Mel spectrogram (middle-left) to a 3-channel representation with static power in the first channel (top-right), increases in power in the second channel (middle-right) and decreases in power in the third channel (bottom-right). 40
5.2 Proposed Binary Neural Network for keyword spotting (left). The ConvBlock used (right). The first ConvBlock does not use batch normalization. The scaler is a learnable parameter which multiplies the activations and the bias is a learnable parameter which is added to the activations. Thresholding is performed using the sign function. k represents the layer number and the b indice represents a binarized value. 42
5.3 Original use-case of the RINC classifier. The classifier replaced the fully-connected layers at the end of a VGG/LeNet network. To accomodate binary decision trees, the outputs of the convolution layers were flattened and binarized. The sparsely connected layer was used to generate higher bit precision logits. 46
5.4 Replacing the convolution kernels of a CNN with a RINC classifier. For each kernel position, the data is flattened and injected into the RINC classifier. A single image can generate many training samples since every kernel position represents new data. C represents the number of input channels, W the width of the kernel, H the height of the kernel and N, the amount of filters used. 47
5.5 Network used to test PoET-BiN. The ConvBlocks are identical to the ones used in Section 5.2.1 with the exception of the weights which are full-precision. This version of the network contains a total of 12 ConvBlocks. The second ConvBlock was replaced with a RINC-1 classifier. 48
5.6 Output Feature Map of the second ConvBlock when extracted from the original network (top), when extracted from the RINC-1 kernel (bottom). 49

5.7 Degradation of the classification accuracy compared to the previous layer according to the number of layers replaced. A higher value means a larger degradation. The red circles represent the layers for which the inputs had more input channels (32 instead of 16). 51
5.8 Receptive field after applying 2 convolution layers with 3×3 kernels using strides of 1. If two layers were to be combined, a 5×5 kernel would be required to capture all the same features. 52
5.9 Networks that merge convolution blocks using RINC-2 classifiers. Network-A (left) contains 3 layers and was the first attempt at merging continuous layers. Network-B (right) contains 7 layers and uses 1×1 convolutions to expand the size of the intermediate feature maps. Blocks in light blue use strides of 1 and blocks in dark blue use strides of 2. All weights and activations used in intermediate calculations are in floating-point format. A sign function is applied to the outputs of the last ConvBlocks of each layer to obtain binary values. 54

LIST OF SYMBOLS AND ACRONYMS

ADC	Analog to Digital Converter
AUC	Area Under the Curve
BN	Batch-Normalization
BNN	Binarized Neural Network
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
DSP	Digital Signal Processing
FAR	False Alarm Rate
FFT	Fast Fourier Transform
FRR	False Reject Rate
FPGA	Field-Programmable Gate Array
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
ISA	Instruction Set Architecture
KD	Knowledge Distillation
KWS	Keyword Spotting
LSTM	Long Short-Term Memory
LUT	Look-Up Table
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel Frequency Cepstral Coefficients
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
ROM	Read-Only Memory
SIMD	Single Instruction Multiple Data
SRAM	Static Random Access Memory
STE	Straight Through Estimator
WER	Word Error Rate

CHAPTER 1 INTRODUCTION

Speech recognition is a technology which aims to transcribe spoken words into written text. This technology has gained lots of traction in recent years due to the arrival of Virtual Personal Assistants (VPAs) [1]. These devices respond to spoken requests and are capable of executing a set of diverse tasks such as turning the lights on or playing music. By popularizing the speech interface, VPAs have changed the way we interact with machines. The modern approach to building a voice recognition system used by a VPA is to employ Deep Neural Networks (DNNs) [2] [3] [4]. DNNs are algorithms capable of approximating non-linear functions when presented with lots of training data. Much like a human learns the differences between a cat and a dog through examples, a neural network is trained by seeing thousands of pictures of both animals along with their associated labels. Neural networks used to distinguish input data (such as images) into a set of finite categories (i.e. cats and dogs) are performing a classification task. This type of task is also used for speech recognition which takes audio signals as inputs and determines which category it belongs to amongst a set of words, characters or sounds. Unfortunately, neural networks are computationally demanding. A typical DNN requires millions of weights and multiplications/additions to compute its predictions. Considering that many IoT applications run on edge devices, power consumption and the storage of weights can become a problem.

1.1 Research Objective

There are many ways to reduce the computational costs of DNNs. One popular method is to quantize the weights and the activations. A typical neural network represents its data using a 32-bit floating-point format. Quantization simply means that these values are discretized to shorter bit lengths. If the values are only represented with a single bit, the resulting network is a Binarized Neural Network (BNN). BNNs are very efficient to implement in hardware since mutliply-add operations can be performed with rudimentary logic and do not require complex multipliers/adders. The memory requirements are also smaller since weights only require a single bit of storage.

The main objective of this thesis is to reduce the power consumption of keyword spotting (KWS) tasks. KWS is a speech recognition task which only seeks to detect a small set of words. It is typically used for voice activation of domotic devices and smartphones [4] [5]. By combining the ideas of BNNs and KWS, the underlying objective is to produce accurate speech recognition systems that are computationally efficient. An accurate system is one

capable of correctly classifying keywords with as few errors as possible. Ideally, such an accurate system should require less memory to store the weights and intermediate results compared to state-of-the-art algorithms. Overall, it should also require less operations and time to compute its predictions.

A typical keyword spotting system represents audio through spectrograms, resulting in images with time-frequency axes. Perhaps, this is not an optimal way to represent audio when working with a BNN. Therefore, this thesis explores new representations more suited for this type of network. Following tests on input representations, the binarization of DNNs for KWS is tested with the hope of obtaining a compromise between accuracy and computational complexity.

1.2 Contributions

The contributions of this thesis are as follows:

- A new speech representation is proposed which offers better results compared to traditional representations when tested on a binarized DNN for KWS. The representation computes the variation of power over time and is composed of binary features.
- Binarization is attempted for the first time on a neural network used for keyword spotting. The proposed network is CNN-based and the effects of each hyperparameter are analyzed.
- PoET-Bin, an algorithm used to map DNNs to FPGA look-up tables is applied to convolution operations for the first time on a KWS task.

1.3 Thesis Plan

The thesis is organized as follows:

- Chapter 2 offers background information on the way features are extracted in a conventional speech recognition system. This chapter also introduces neural networks and their various topologies.
- Chapter 3 summarizes the state-of-the-art networks used for keyword spotting. Binarized neural networks and the optimizations that followed the original idea are also presented.

- Chapter 4 presents a new type of spectrogram which uses binary values (i.e. 1 or -1) to represent features.
- Chapter 5 presents a binarized neural network for keyword spotting based on the ResNet architecture. The PoET-Bin algorithm is tested on the convolution layers of this network.
- Chapter 6 discusses the results obtained with the binarized network.
- Chapter 7 concludes this thesis and offers insights into future works.

CHAPTER 2 BACKGROUND

This chapter offers a brief introduction to topics such as speech recognition and deep learning. Feature extraction for speech, evaluation metrics of recognition systems and neural network architectures are discussed.

2.1 Speech Recognition

The main objective of automatic speech recognition is to convert an audio sequence of spoken words into a written format. This task includes many sub-tasks such as Keyword Spotting (KWS) which aims to detect specific words defined in a restricted vocabulary or Large Vocabulary Continuous Speech Recognition (LVCSR) which seeks to understand any word or phrase in a language. The former task (KWS) can be accomplished by a classifier that is trained directly on audio sequences of words. The latter (LVCSR) requires the use of smaller units of speech, such as phonemes. Phonemes can be defined as the fundamental unit of speech. The English language contains about fourty different phonemes [6].

2.1.1 Feature Extraction for Speech

Like many supervised machine learning applications, speech recognition requires lots of data and proper preprocessing to be effective. Data, in this case, are acquired through an Analog to Digital Converter (ADC), resulting in a series of amplitudes over time. However, most systems do not make direct use of this data. Speech is usually processed into some form of spectrogram. The most commonly used features are MFCCs [7].

The frequency components of speech are usually contained under 8 kHz. In order to respect the Nyquist theorem, audio is usually acquired with a sampling rate of 16 kHz. The resulting audio signal is then cut into short segments of 20-40 ms. This short time-frame lets us assume that the audio signal is stationary, which means that it is possible to apply a Discrete Fourier Transform (DFT) required to compute MFCCs. A new frame is generated for every 10 ms of audio. Since the frame shift is smaller than the actual frame size, there is a certain overlap which prevents the loss of information. For each frame, a windowing function such as a Hamming Window is applied to prevent spectral leakage. To compute the DFT, the Fast Fourier Transform (FFT) algorithm is used, which requires a signal with a quantity of points corresponding to a power of two. The signal is zero-padded to meet this requirement. Figure 2.1 illustrates the windowing process using a window size of 30 ms (or 480 points of a signal

sampled at 16 kHz) and a frame shift of 10 ms.

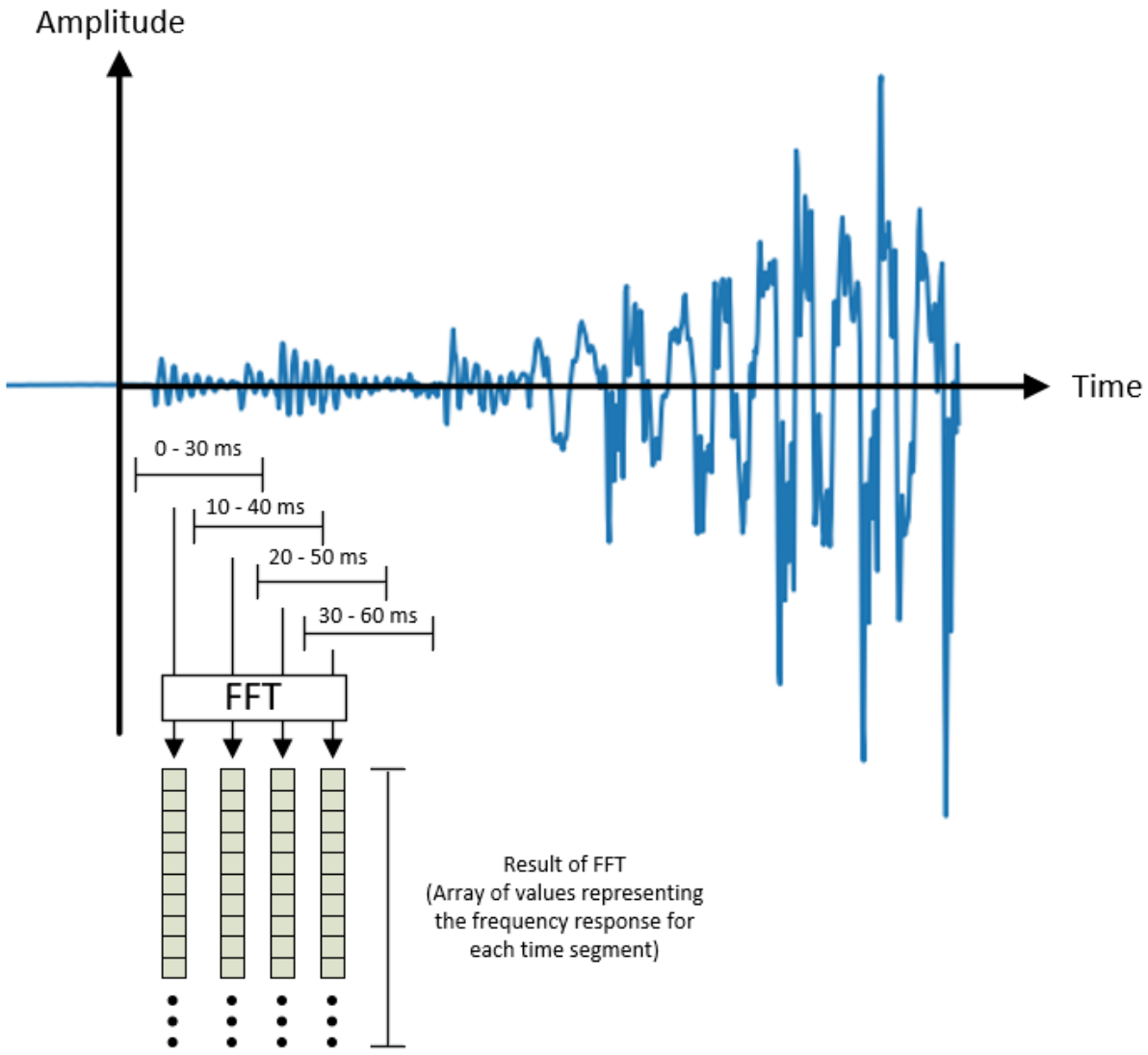

Figure 2.1 Extracting frames from an audio signal sampled at 16 kHz.

Once the FFT is calculated for each frame, the magnitude of the signal is computed over its respective frequencies. A series of filters called Mel Filters are then applied. These filters are disposed over a logarithmic scale and become wider and wider as the frequency increases. This behavior is meant to model the human auditory system which processes sound in a logarithmic manner [7]. Figure 2.2 illustrates a Mel Filterbank with 20 filters. Typically, systems may use a total of 40 filters.

To apply the filterbank, a dot product is performed between the power spectrum and each individual filter. The resulting spectrogram contains as many values per time frame as there

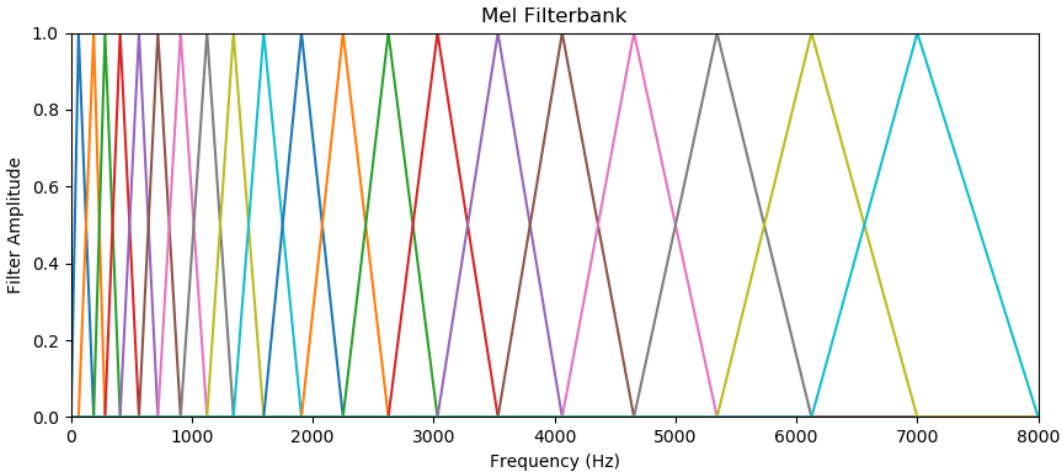

Figure 2.2 Filters spread out on the Mel Scale.

are filters. Since the perceived intensity of sound does not increase linearly with power, a logarithm is then applied to obtain a decibel scale. At this point, the calculated spectrogram corresponds to the log-Mel spectrogram. Figure 2.3 illustrates this representation. To obtain MFCCs, the last step is to compute the Discrete Cosine Transform (DCT) of the log-Mel spectrogram. The first coefficient is often ignored since it represents the average power of the signal. Typically, the 12 or 13 next coefficients are kept, as they represent most of the useful information relevant to speech. Figure 2.4 is an example of MFCCs.

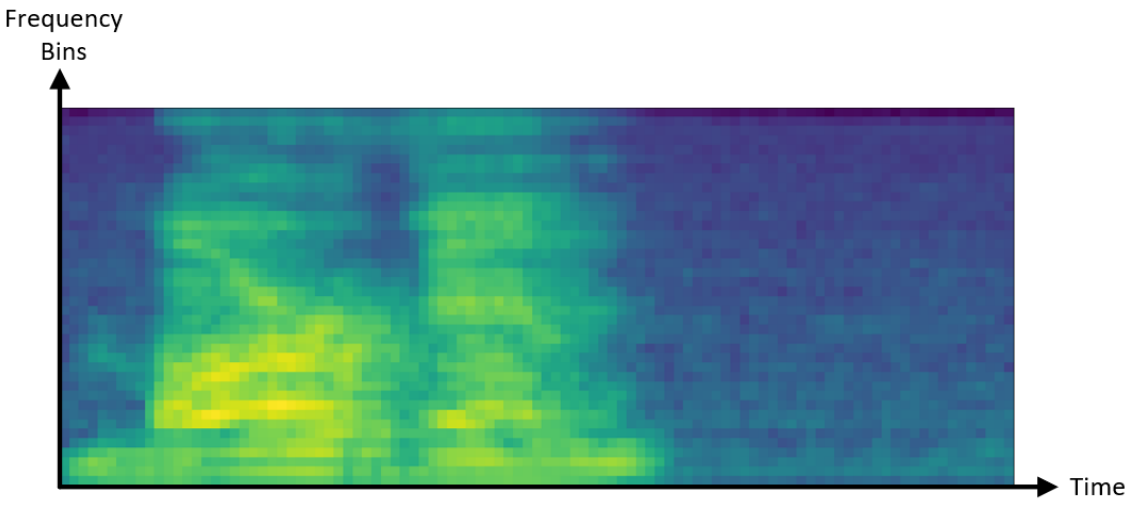

Figure 2.3 Example of the log-Mel Spectrogram with 40 filters. The vertical axis represents the various frequency bins (after applying filters). The horizontal axis represents time.

Traditional speech recognition systems (before neural networks) used Gaussian Mixture Mod-

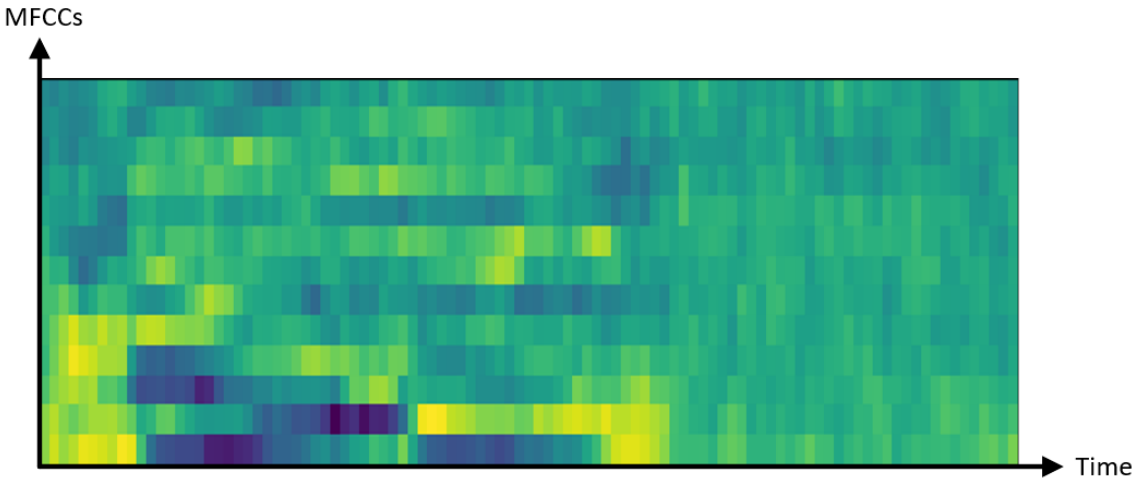

Figure 2.4 Example of MFCCs using 13 coefficients. The vertical axis represents MFCCs for a specific time frame. The horizontal axis represents time.

els (GMMs) to model the spectrum of each frame. To simplify training and inference, the gaussians that were used had diagonal covariance matrices. This meant that the spectrum had to contain coefficients that were decorrelated. The DCT operation does exactly that. However, neural networks are easily capable of understanding correlations between the input data and therefore do not need MFCCs [8]. To this day, many systems without GMMs still use MFCCs. However, for most of this thesis, the log-Mel spectrogram will be used.

Another common practice to improve the classification accuracy is to concatenate delta and delta-delta coefficients [9]. To compute these, past coefficients are substracted from future coefficients. Even if neural networks can easily learn weighted sums and substractions, this operation brings a slight improvement on recognition accuracy regardless of the type of spectrogram [8].

2.1.2 Evaluation Metrics

To evaluate and compare speech recognition systems, various metrics can be used. The common methods that will be discussed in this thesis are Word Error Rate (WER), Receiver Operating Characteristics (ROC) curves and classification accuracy.

Word Error Rate

WER is a metric often used in LVCSR tasks. WER measures three types of errors : substitutions, deletions and insertions. If any word is replaced with an incorrect word, a substitution

(N_S) error is detected. If a word is missing, it counts as a deletion (N_D). If an extra word is added to the transcription, then the error is an insertion (N_I). The WER is the sum of all errors divided by the total amount of expected words (N_T) (see eq. 2.1).

$$WER = \frac{N_S + N_D + N_I}{N_T} \tag{2.1}$$

For example, a speech recognition system needs to identify the following sentence: "Jason ordered a turkey sandwich." If the system detects: "Jay's son ordered a sandwich.", we have 1 substitution ("Jay's"), 1 insertion ("son") and 1 deletion ("turkey"). The WER can be calculated:

$$WER = \frac{1 + 1 + 1}{5} = 0.6$$

Receiver Operating Characteristics

A second type of metric used (for KWS) is based on ROC curves. These curves report the False Reject Rate (FRR) versus False Alarm Rate (FAR) for varying threshold values. For instance, if a classifier outputs probabilities for each potential keyword, a ROC curve can be calculated for each keyword versus all other categories. The curves are then averaged vertically (for fixed FAR values) to produce a single curve for the entire system. The threshold is initially set to 0% and is gradually incremented to 100%. If the expected word's probability exceeds the threshold, then the sample is a true positive. The FAR and FRR values are calculated as follows:

$$FAR = \frac{FP}{FP + TN} \tag{2.2}$$

$$FRR = \frac{FN}{FN + TP} \tag{2.3}$$

where TP = True Positive,
TN = True Negative,
FP = False Positive,
FN = False Negative

Figure 2.5 illustrates an example of ROC curves. The best system is the one with the smallest Area Under the Curve (AUC). System A has the lower AUC in this scenario.

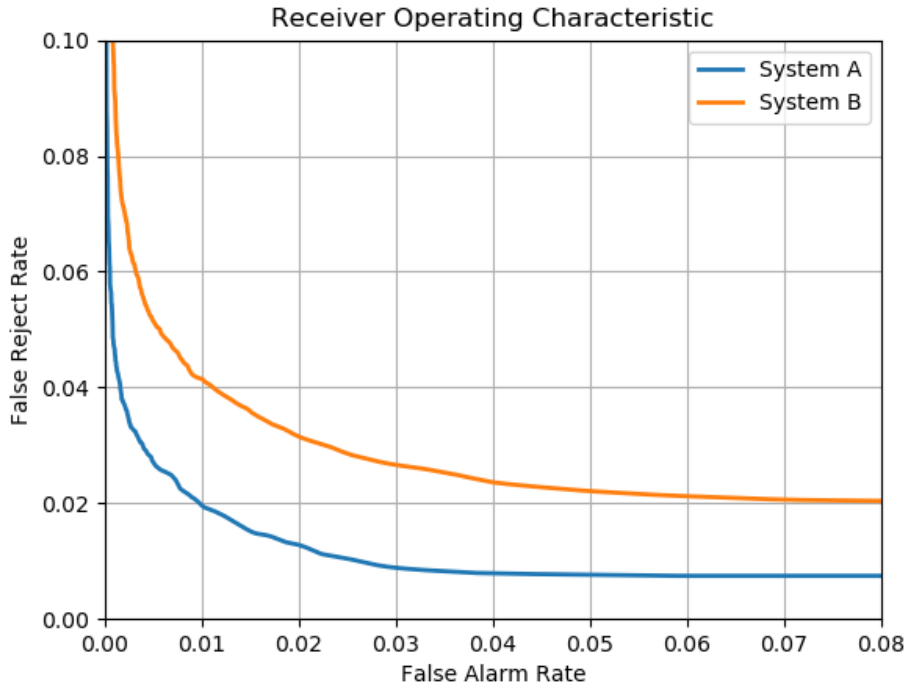

Figure 2.5 Example of ROC curves illustrating False Rejects vs False Alarms.

Top-One Classification Accuracy

The final metric used in this thesis is the classification accuracy. Deep Neural Networks (DNN) built for classification generally use a softmax layer at the end of the network. This normalizes the outputs so that the probabilities of all classes sum to one. Softmax can be calculated using equation 2.4. C represents the total number of output classes.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}} \tag{2.4}$$

The prediction of the classifier corresponds to the highest output probability. This will be the case for all reported accuracy figures in this thesis.

2.2 Deep Learning

In recent years, deep learning has found its way into many tasks, including speech recognition [4]. Deep learning simply means that deep neural networks are used. The most basic type

of neural network is the Multi-Layer Perceptron (MLP). The fundamental unit of an MLP is a neuron. From a computational perspective, a neuron is a weighted sum of inputs that is then subject to a non-linear function (see Figure 2.6).

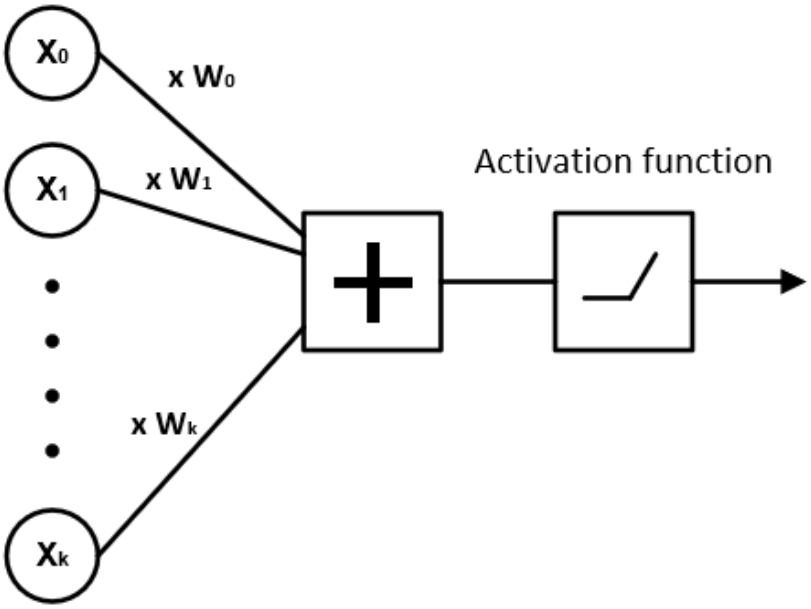

Figure 2.6 General structure of a neuron. The inputs are represented with X and the weights with W.

To obtain a neural network, neurons need to be stacked in layers. Each neuron in a specific layer also needs to be connected to all the neurons of the previous layer. A neural network with at least one hidden layer (1 middle layer between inputs and outputs) can approximate any continuous function [10]. The quality of this approximation depends on the dataset and network architecture. A DNN is a neural network which is deep (it has many layers). Figure 2.7 illustrates the general structure of a DNN.

In an MLP, the weights are learnable parameters. To correctly approximate a function, the neural network is trained using gradient descent. Input data is injected into the network and predictions are calculated. Labeled data is then presented to the classifier and weights are updated gradually by computing gradients derived from a loss function such as cross-entropy loss. If the network correctly classified the input data, the weights will not be updated because the loss is null. If the network made a mistake, the weights will be updated according to the intensity of the error (how close was the predicted probability).

A second type of neural network used is the Convolutional Neural Network (CNN). These networks exploit the concept of filtering to learn representations in images. The learnable

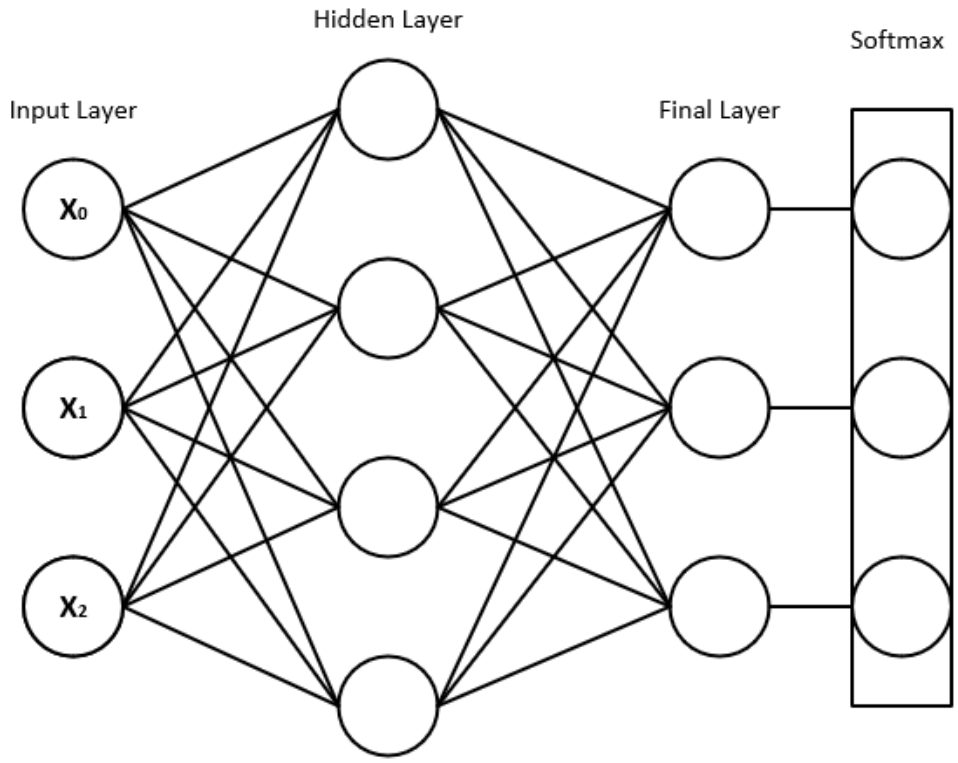

Figure 2.7 Example of a Multi-Layer Perceptron with a single hidden layer.

parameters in this type of network are contained in the filters. By reusing weights over the entire image, this type of network is capable of functioning with less weights compared to the MLP. It also has the advantage of extracting local features in an image. The early convolution layers learn basic geometric shapes, whereas later layers cover larger receptive fields, giving them the ability to learn more complex shapes and representations. CNNs typically use MLP layers as final layers. A graphical example of the convolution operation is illustrated in Figure 2.8. A sliding window or kernel computes a weighted sum and non-linearity for each position in the image it encounters generating a feature map.

The third type of neural network which is used in speech recognition is the Recurrent Neural Network (RNN). This type of network is powerful when used with time-dependant data. In the case of an RNN, a neuron has 2 inputs: the current input data and the output that was predicted for the previous time-step. If the input at time t is represented with x and the output with h, then equation 2.5 explains the relation.

$$h^t = f(h^{t-1}, x^t) \tag{2.5}$$

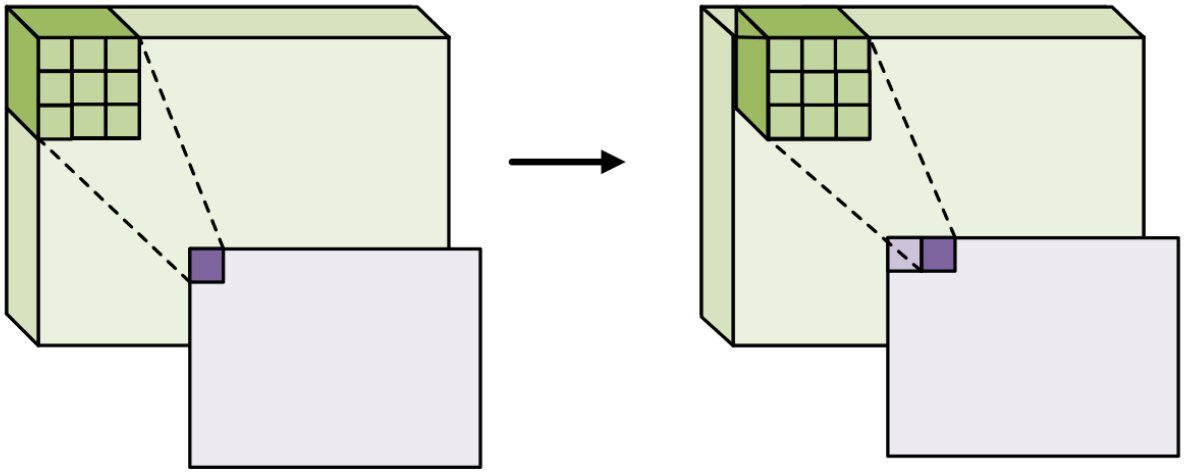

Figure 2.8 Example of the convolution operation performed in a CNN.

Vanilla RNNs suffer from vanishing gradients since the gradient has to flow through many states that evolve through time. One solution to this problem was the introduction of the Long Short-Term Memory (LSTM) [11]. This type of RNN uses gates in order to leverage the important information that flows through the network cells. The memory component of LSTMs are passed on using a cell state value (C). Using the output of the previous cell (h) along with the new input features (x), the cell state is updated by either forgetting information (equation 2.6a) or introducing new information (equation 2.6b). The output gate then computes the result of the cell using the current cell state, the previous output and the new input feature (equations 2.6c and 2.6d). Equations 2.6 summarize the intermediate calculations in an LSTM cell and Figure 2.9 illustrates the structure of a cell. LSTMs are very powerful when used in large vocabulary speech recognition tasks [12] or simpler keyword spotting tasks [13] [14].

$$f_t = \sigma(W^f * h_{t-1} + U^f * x_t + b^f) \tag{2.6a}$$

$$i_t = \sigma(W^i * h_{t-1} + U^i * x_t + b^i) \tag{2.6b}$$

$$o_t = \sigma(W^o * h_{t-1} + U^o * x_t + b^o) \tag{2.6c}$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * tanh(W^c h_{t-1} + U^c x_t + b^c)) \tag{2.6d}$$

$$h_t = tanh(C_t) * o_t \tag{2.6e}$$

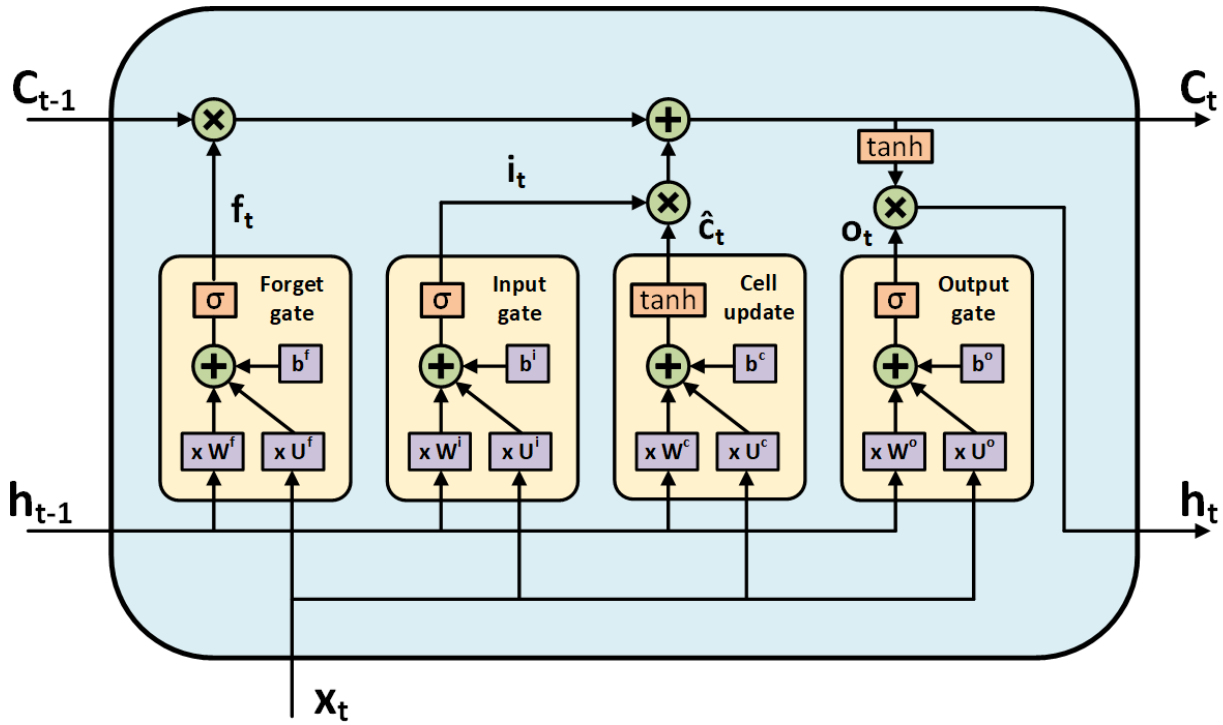

Figure 2.9 Structure of an LSTM cell. Each cell includes a forget, input and output gate. The C values represents the cell state which remembers the information across states.

2.3 Summary

This chapter presented the log-Mel spectrogram and MFCCs, two spectrograms used to represent speech. Various metrics were also presented to evaluate the quality of speech recognition systems. Finally, neural networks, which are used for speech recognition tasks, were explained. The following chapter will introduce works which achieved state-of-the-art results on a sub-task of speech recognition called keyword spotting. Works which managed to reduced the computational costs of neural networks are also discussed.

CHAPTER 3 LITERATURE REVIEW

This chapter will offer an introduction to the state-of-the-art in the sub-fields of keyword spotting and binarized neural networks. Section 3.1 will begin by presenting popular architectures that have achieved record performance. Section 3.2 will then present binary neural networks and the many optimizations that followed the original idea. Finally, section 3.3 will present works that attempted to combine the ideas of automatic speech recognition and binary neural nets.

3.1 Keyword Spotting

KWS is a common speech recognition task which typically runs on edge devices. The objective is to recognize a small-set of words when they are uttered. With the rise of Virtual Personal Assistants (VPAs), this task has become extremely popular in consumer electronics. Android phones, for example, are now equipped with the Google assistant which responds to "Ok, Google". Other examples include Apple's "Siri", Amazon's "Alexa" and Microsoft's "Cortana" [1].

In recent years, DNNs started to gain traction for KWS [4]. Chen et al. introduced MLP architectures with 3 and 6 hidden layers to classify specific key phrases such as "Ok, Google" and "Dismiss alarm". They compared their network to a system using Hidden Markov Models (HMM) that also used DNNs for acoustic modeling. When comparing ROC curves, the DNN architecture offered 45% relative improvement. This, however, was achieved using special techniques like transfer learning. This means the network was pretrained on another dataset. One interesting thing to note is that the DNN was trained on keywords directly, meaning that the total quantity of output neurons was greatly reduced compared to systems that use context-dependant phonemes which number in the thousands.

Following the works of Chen et al. on MLPs, Sainath et al. proceeded to apply CNNs to KWS tasks [5], since they have shown promising results in LVCSR tasks [3]. Compared to the DNN implementation proposed by Chen et al. [4], the CNN offered over 20% relative improvement. With the objective of limiting the number of multiplications and parameters, they tested various networks with different hyper-parameters. Notably, they tried varying the number of convolutional layers (1 or 2) and to increase pooling and strides. In these two works produced by Google, in-house datasets containing less than a dozen keywords were used.

3.1.1 Establishing a Common Benchmark: The Google Speech Commands

Speech recognition, like many other machine learning tasks, depends heavily on the availability of labeled training data. Traditionally, datasets in this field have been hiding behind a paywall or were proprietary. Some exceptions, such as Librispeech [15] were only aligned at the sentence level making it more relevant to LVCSR tasks. A dataset for the detection of simple keywords was required. This motivated Pete Warden from Google Brain to release the Google Speech Commands [16]. This dataset was crowdsourced via a web application. The volunteers were asked to record themselves prononcing specific words. The dataset comes in two versions. The first version contains 64,727 one-second speech utterances sampled at 16 kHz. Every sample represents one of thirty different words. The second version of the dataset has over 100 k utterances seperated into 35 classes. One of the proposed tasks for this dataset is to deduce the top-one classification accuracy of the labeled data. To do this, 10 words are chosen for classification: "yes", "no", "up", "down", "left", "right", "on", "off", "stop" and "go". Two extra output categories are added to any classifier trained on this dataset: "unknown" and "silence". The "unknown" category is a mix of samples from the remaining words and the "silence" category is extracted from audio files containing background noise.

With the introduction of the Google Speech Commands, the CNN proposed by Sainath et al. was adapted and tested on this dataset in order to establish a common benchmark [17]. Most works presented in this literature review made use of this dataset.

3.1.2 State-of-the-art Architectures

After proving the proficiency of deep learning in KWS tasks, many researchers proposed their own architectures to achieve better results on the Google Speech Commands dataset. A special type of CNN named ResNet was recently used to achieve state-of-the-art performance in computer vision tasks [18]. This type of network introduced skip connections where the feature maps or outputs of one layer were added to the inputs of future layers, bypassing some of the convolution layers. This had the property of reducing the impact of vanishing gradients which arise when a network is very deep. Given the success of ResNet, Tang et al. applied this technique to KWS tasks [19]. Unlike the original ResNet, their network required no pooling or strides greater than one. In order to achieve larger receptive fields, dilated convolutions were used. This means that filters expand and skip certain values within the input image. Their most accurate network ($res15$), had 14 layers. The input features were a spectrogram containing 101x 30 ms frames of 40 MFCCs. The resulting ResNet considerably improved the top-one classification accuracy of CNNs (from 91.7% to 95.8%). The ROC curves were also plotted and matched this improvement. Shortly after the release of this

work, Qiu Lin et al. optimized the hyper-parameters in order to produce a series of networks which had better accuracies and required less parameters and computations [20]. The best of these networks ($EdgeSpeechNet - A$) achieved 96.8% top-one accuracy.

Other works attempted to leverage the use of RNNs. In fact, RNNs have been quite popular on this type of task for over a decade [21]. When considering more recent developments, Coimbra et al. introduced a network which included convolution and LSTM layers [13]. An attention mechanism was added to improve the classification accuracy. This mechanism learned weights to associate greater value to the most relevant frames of the entire audio sample. This offered 95.6% classification accuracy when tested on the first version of the Google Speech Commands dataset and 96.9% when tested on the second version. Zeng et al. have also made use of RNNs in their architecture [14]. They proposed a 3-stage network containing dense blocks [22], bi-directional LSTMs with an attention mechanism and, finally, fully connected layers. Dense blocks are essentially a more draconian version of ResNet, where the feature maps of all previous layers act as residuals to the current layer. However, unlike ResNet, the residuals are concatenated instead of added to the output of the previous layer. Zang et al. reported 97.5% accuracy on the 12 commands classification task using this network.

Many of the previously mentioned networks were very accurate but this came at the cost of high computational complexity. Very low power edge devices, such as microcontrollers, cannot feasibly perform billions of multiplications in real-time. Also, there are severe memory constraints. When analyzing ARM Cortex-M based microcontrollers supplied by companies like STMicroelectronics, operating frequencies range from 50 MHz to over 200 MHz. On-chip memory is usually limited to Flash (under 2 MB) which acts as Read-Only Memory (ROM) and is used to store the program and Static Random Access Memory (SRAM) (under 500 kB) which is an active read/write memory. Only some ARM microcontrollers support floating-point instructions. To overcome this limitation, parameters are often represented in fixed-point format. Cortex-M4 and Cortex-M7 based microcontrollers also support Single Instruction Multiple Data (SIMD) operations. For example, if data is represented in an 8-bit fixed point format, 4 multiplications can be performed in parallel in a single clock cycle (since data is stored in a 32-bit register). CMSIS-NN, an open-source library produced by ARM for neural network inference on microcontrollers, exploits this parallelism [23]. With the help of this library, Zhang et al. proposed a series of networks for keyword spotting ($DS{-}CNN$) that could feasibly run on a microcontroller [24]. By working closer to the hardware, Zhang et al. considerered important information that is often overlooked such as the memory requirements of the intermediate computations and the storage of activations. Since 8-bit fixed-point representations were required, they also leveraged the concept of quantization and its effects

on classification accuracy. The most efficient networks proposed were directly inspired by MobileNet [25] which uses densely separated convolutions. This type of convolution applies 3×3 filters independently to each channel and then uses 1×1 convolutions on the resulting values to capture the information across channels. This reduces the amount of multiplications by almost $9\times$ with minimal loss in accuracy. Their largest and most accurate network offered a classification accuracy of 95.4%, required 500 kB of memory (for weights and activations) and 57 M multiply-add operations.

Despite the promising results of MobileNet, Choi et al. produced much more accurate and compact networks ($TC - ResNet14 - 1.5$) by modifying the dimensionality of the input features [26]. Instead of considering the spectrogram (or MFCCs in this situation) as an image with time on the x-axis and frequency on the y-axis, the frequency components were treated as channels. This created a simple 1-dimensional representation with multiple channels. Convolutions were performed over single dimensions, drastically reducing the amount of computations. The largest proposed network offered a classification accuracy of 96.6% on the Google Speech commands dataset, while only requiring 13.4 MFLOPs of calculations and 305 k parameters.

A summary of the results of these networks on the first version of the Google Speech Commands is reported in Table 3.1.

Table 3.1 Summary of Various Network Results on Google Speech Commands V1

Model	Param.	Mult.	Accuracy
trad-fpool3 [5] [19]	1.37 M	125 M	90.5%
cnn-tpool2 [5] [19]	1.09 M	103 M	91.7%
res15 [19]	238 k	894 M	95.8%
EdgeSpeechNet-A [20]	107 k	343 M	96.8%
Attention RNN [13]	202 k	-	95.6%
DenseNet-BiLSTM [14]	250 k	-	97.5%
DS-CNN-S [24] [26]	24 k	5.4 M	94.4%
DS-CNN-M [24] [26]	140 k	19.8 M	94.9%
DS-CNN-L [24] [26]	420 k	56.9 M	95.4%
TC-ResNet14-1.5 [26]	305 k	13.4 M	96.6%

3.1.3 Binary Feature Extraction for Keyword Spotting

As seen in Section 2.1.1, speech recognition systems make use of spectrograms as input features. An article explored the use of binary representations (black and white spectrograms) in Bangla speech [27]. Using Otsu's thresholding algorithm [28], the authors generated binary

representations using spectrograms of words. They argued that this binary representation offered a satisfactory classification accuracy. However, they used a very small in-house dataset and their objective was not clear. In the following chapters we also introduce a binary representation meant to be used with BNNs (which are explained in the following section). Given the time of publication of this article, the authors did not have knowledge of BNNs and were likely trying to improve existing feature extraction methods. Nevertheless, the similarities between this work and ours is noteworthy.

3.2 Binary Neural Networks

Neural networks are very computationally demanding. When considering low-power embedded systems, their memory requirements are also very large. Many solutions have been proposed to reduce their computational complexity. One of these solutions involves pruning unnecessary connections [29]. Han et al. retrained neural networks by gradually removing neurons. On popular CNN architectures, such as AlexNet [30] and VGG-16 [31], they managed to reduce the amount of parameters by an order of magnitude without affecting the classification accuracy on the ImageNet dataset.

Another useful way to reduce the number of computations in a neural network is to use quantization. This will be the main focus of this section. Quantization can be defined as the discretization of values to smaller precisions. Parameters in a conventional neural network are represented using a floating-point format. In terms of memory consumption, this means 32 bits are required per value. However, some works have demonstrated that it is possible to use smaller fixed-point representations without affecting accuracy [32]. The most draconian form of quantization is binarization. Binarization means that the parameters of the network can only hold the values {0, 1} or {-1, 1}. If both the weights and the activations in a neural network are binarized, then the network can be considered a BNN [33]. The following sections will introduce the works that led to this idea and the many optimizations that followed.

3.2.1 BinaryConnect: Neural Networks with Binary Weights

Most computations in neural networks are multiply-accumulate operations. If the values of the weights are restricted to {-1, 1}, then every multiply-add becomes a simple addition or substraction. With this idea in mind, Courbariaux et al. introduced BinaryConnect, a method to binarize the weights of a neural network with minimal loss in accuracy [34]. To do this, BinaryConnect trained networks using binary weights directly. During inference, the real valued weights were thresholded to {-1, 1}. Two methods were proposed for thresholding:

deterministic and stochastic binarization. In the case of deterministic binarization, positive weights were approximated by 1 and negative weights by -1. With stochastic binarization, higher valued weights were more likely to become 1 and smaller weights more likely to become -1.

During backpropagation, gradients were computed with respect to the binarized weights. Since the sign function is not differentiable, the Straight Through Estimator (STE) was used to let the gradient flow [35]. The full precision values of the weights were saved throughout the training as to permit small weight variations. Equation 3.1 explains how the weights were updated. The b index indicates a binarized value. C represents the cost function and η represents the learning rate. The final weights were clipped to remain in the [-1, 1] interval.

$$W_{(t)} = clip_{(-1,1)}(W_{(t-1)} - \eta \frac{\partial C}{\partial W^b}) \tag{3.1}$$

This method was applied to a DNN on the MNIST, CIFAR-10 and SVHN datasets. BinaryConnect produced slightly smaller test error rates when compared to the equivalent full precision network. It was concluded that binarization of weights acted as a regularizer, improving the generalization of the classifier.

3.2.2 BNNs: Neural Networks with Binary Weights and Activations

Extending on the ideas introduced in BinaryConnect, Hubara et al. released a paper reporting on Binarized Neural Networks (BNNs) [33]. This was an important contribution since most BNN implementations that followed were inspired by the techiques proposed in this paper. The main difference between BinaryConnect and BNN was that the activations in BNN were also binarized using the sign function. Activation functions, such as ReLU, were not used nor needed since the sign function was already non-linear. Prior to binarizing the activations, a regularization technique called Batch-Normalization (BN) was applied [36]. BN computes the average and the variance of mini-batches during training and adjusts the distribution of the activations to make them centered around an average of zero and a variance of one. In the case of binarization, this was important since the sign function could convert all activations to 1 or -1 if the values were biased in a certain direction. The general flow of computations is illustrated in Figure 3.1.

In BinaryConnect, the STE was used to let the gradient backpropagate through the sign function. This was also used in BNN. The activations followed a similar implementation where only the values contained within the [-1, 1] interval were backpropagated. Figure 3.2 summarizes the functions used for forward and backpropagation of activations. The sign

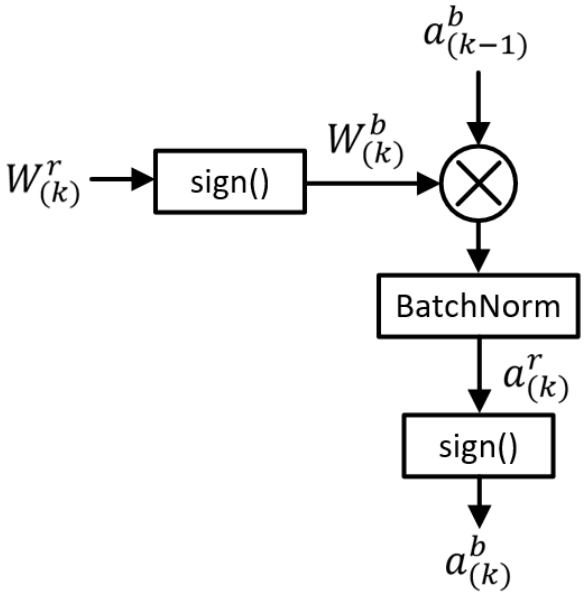

Figure 3.1 Forward propagation of BNNs. k represents the layer number, r, a full precision or real value and b, a binarized value.

function's gradient is nearly always zero (b). To prevent this, the sign function (a) is only used during forward propagation. When backpropagating, the sign function is approximated with the hard tanh function (c). This produces a usable gradient consisting of a rectangular function (d).

There are several advantages to have binary weights and activations. Hubara et al. pointed out that multiplications with values equal to {-1, 1} can be performed using an XNOR logic gate if all -1 are encoded as 0. If all activation and weight values are stored in a 32-bit register, then it is possible to perform 32 multiplications using a single logical instruction. The authors also suggested that the popcount instruction could be used to perform the accumulation part of the algorithm. Popcount is an operation which counts the number of 1s in an array of bits. In terms of memory consumption, BNNs require approximately 32× less space compared to their full precision counterpart. The XNOR and popcount operations are illustrated in Figure 3.3.

3.2.3 BNN Architectures

One of the most notable attempts at improving BNNs was XNOR-Net [37]. In this work, Rastegari et al. introduced a scaling factor for the weights and activations which attempted to reduce the L1 distance between the binarized weight/activation and their full precision coun-

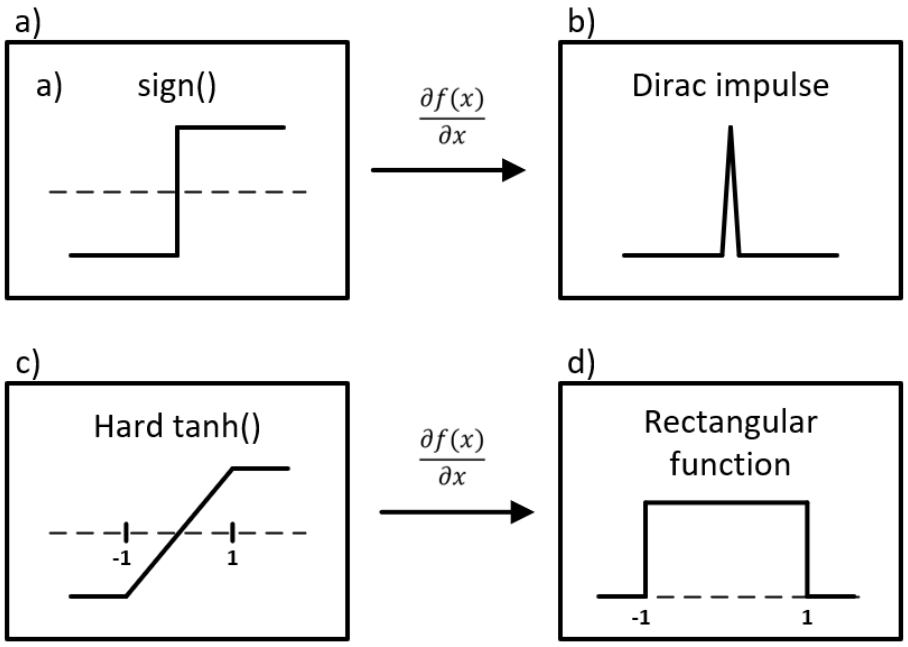

Figure 3.2 a) Sign function used during forward propagation. b) Gradient of sign function (Dirac impulse). c) Hard tanh function used during backpropagation. d) Gradient of hard tanh (Rectangular function)

terpart. XNOR-Nets claimed to have achieved better performance than traditional BNNs on the ImageNet dataset, scoring 44.2% top-1 accuracy, using Alex-Net. As pointed out by Simons et al., the use of these scaling terms required extra computations during inference which are often overlooked [38].

Another interesting development in BNNs was Bi-Real Net [39]. In this work, Liu et al. used a CNN, which, much like ResNet, contained full precision residuals. Full precision skip connections can be quite useful since information can still flow through the network without having to go through the sign function. Liu et al. also introduced other ideas such as a piecewise polynomial function to approximate the sign function applied to activations. When computing the gradients during backpropagation, the magnitude of the weights were also considered instead of using the STE. Combined, these small modifications achieved a 10% top-1 accuracy improvement over XNOR-Net applied to ResNet18.

3.2.4 Optimization of BNNs

Instead of proposing novel architectures, some works have focused on optimizing existing ones. Notably, Tang et al. have stated that the questionable accuracy of BNNs on larger datasets

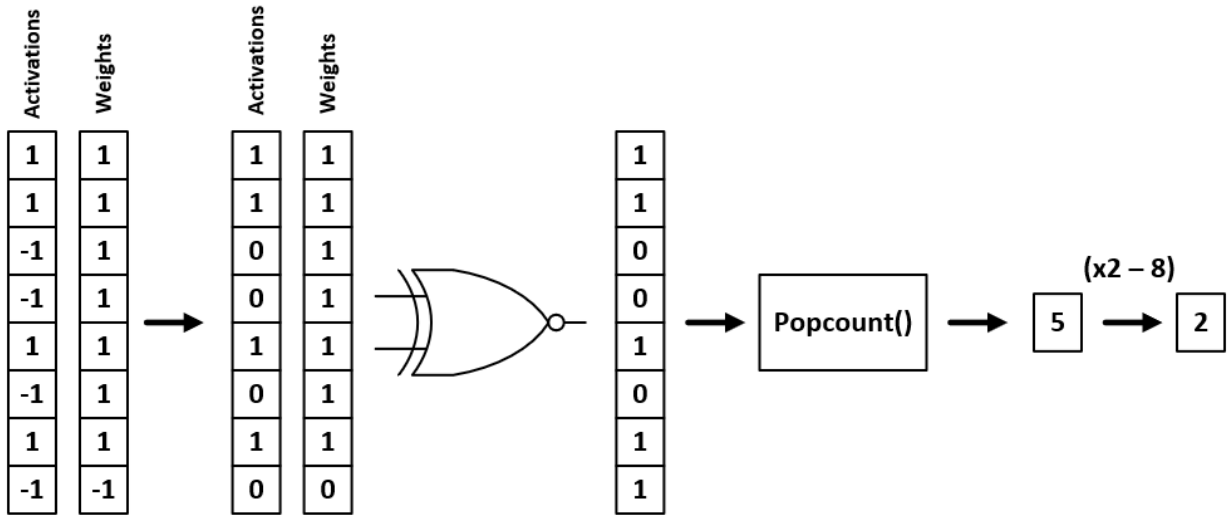

Figure 3.3 Example of a binary dot-product with xnor and popcount operations. The result of the popcount operation is multiplied by 2 and substracted by the number of bits in the input array. This is required since -1s are represented with 0s.

could be attributed to "improper training strategies" [40]. In their work, they proposed a few solutions including using a small learning rate, using the PReLU activation function, not using L2 regularization and adding a scale layer to the final network layer. As stated by Tang et al. the go-to learning rate for a typical CNN is 0.01. However, by reducing this to 0.0001, they obtained nearly 20% improvement in classification accuracy (top-1 and top-5). They argued that larger learning rates may flip the weight values very frequently making the overall training more instable. The PReLU activation function, which is nearly identical to ReLU, adds a scaling factor to negative values instead of ceiling them to zero. This scaling factor was said to help improve accuracy in the same way that XNOR-Net's scaling factors helped. However, unlike XNOR-Net, it did not require the real-time computation of the averages of activations. Figure 3.4 illustrates the PReLU function. Tang et al. also criticized the use of L2 regularization which is often used in neural network training. This regularizer typically penalizes large weights favoring values closer to zero. However, since the objective of a binary net is to push weights to {-1, 1}, this regularizer was at odds with the final objective. Instead, they proposed a regularizer which penalized values far from {-1, 1}, including values near zero. When looking at the training curves, this offered more stable accuracies (fewer oscillations during training). Tang et al. also noted that binarizing the final layer of a BNN degraded the overall accuracy. To counter this, they added a learnable scaling factor to the last layer. Overall, Tang et al. demonstrated that BNNs had lots of potential when implemented correctly.

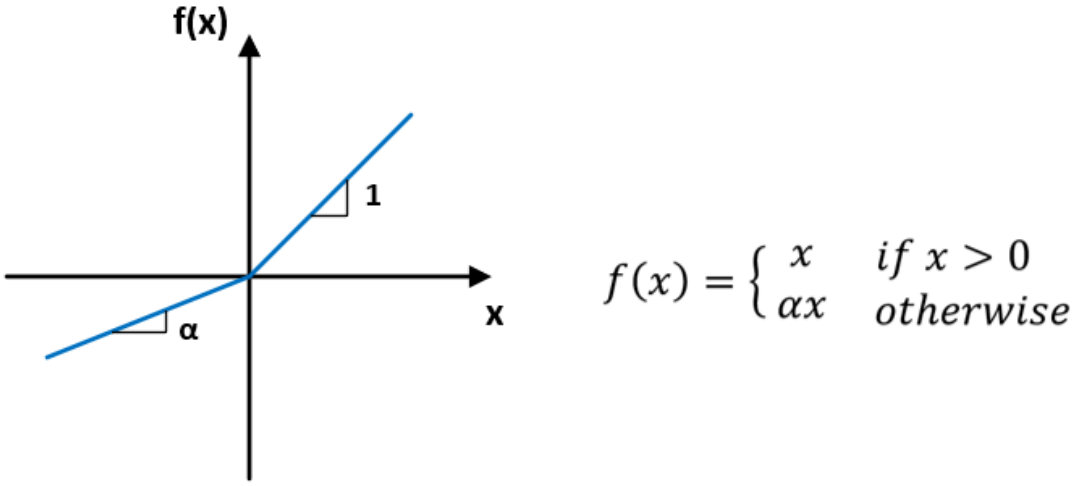

Figure 3.4 The PReLU activation function.

Recently, changes to the vanilla BNN were proposed by Darabi et al. in a work called BNN+ [41]. This work introduced a couple of ideas including an alternative to the STE and a regularizer function which penalized large weights, but also very small weights, much like Tang et al. proposed. When the weights in a BNN are near the limits of {-1, 1}, it is hard to train since the STE cuts off values that exceed these thresholds. To counter this, Darabi et al. proposed the Signswish function which is an approximation of the sign function that almost always has a non-zero gradient. The shape of the function is illustrated in Figure 3.5. These modifications have offered slight improvements over previous BNN state-of-the-art accuracies.

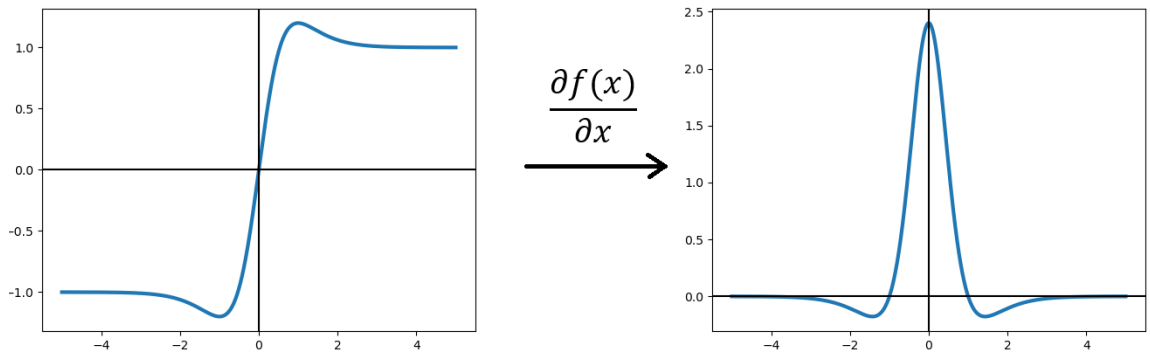

Figure 3.5 The Signswish function and its gradient.

3.2.5 BNNs using Look-Up Tables (LUTs)

Despite the encouraging results of the presented works, neural networks were not originally meant to be binarized. Training requires the calculation of gradients. A couple of tricks and approximations have to be used in order to let the gradients flow through the thresholding functions. Also, many hardcoded processors do not support popcount instructions, limiting the advantages of binarization. On the other hand, the low-level logical nature of Field-Programmable Gate Arrays (FPGAs) makes them a perfect candidate for BNN implementations. Logical elements in an FPGA are in the form of small memories named Look-Up Table (LUT). A typical 7-series Xilinx FPGA has 6-input LUTs. In order to obtain very efficient BNN implementations, some works have attempted to train classifiers using FPGA primitives directly [42] [43].

Notably, Abdelsalam et al. introduced PolyBiNN, a neural network engine built entirely of LUTs [42]. To do this, binary decision trees were used. Their network was tested on the MNIST and CIFAR-10 datasets. For MNIST, the input images were binarized. The trees were built by selecting the input features which offered the best split. Each decision tree contained as many splits as there were inputs to the target FPGA LUTs. Since a single decision tree would not act as a powerful classifier, the Adaboost algorithm was used to combine many decision trees together [44]. Each ensemble of decision trees was responsible for detecting one of the output classes in a one-versus-all fashion. A voting circuit was added at the end as to only obtain a single active output per input image. In the case of the CIFAR-10 dataset, convolution layers were applied and used as feature extractors. The outputs of the final convolution layer were binarized and served as inputs to the decision trees. Overall, PolyBiNN managed to produce comparable accuracies to FINN, a framework which builds inference engines for BNNs on FPGAs [45]. Since all the weights were incorporated inside the decision trees/LUTs, no block RAM was required. The biggest perks of this approach were high throughput and low-power consumption.

Influenced by PolyBiNN, Chidambaram et al. introduced PoET-BiN, an efficient alternative to BNNs which stacked multiple layers of LUTs [43]. To do so, PoET-BiN applied the Adaboost algorithm to learn several binary decision trees. Much like in PolyBiNN, every single one of these trees was to be interpreted as a LUT with as many inputs as the targetted FPGA technology. The Adaboost algorithm associated a weight to every LUT. The weighted sum of the results was then thresholded to obtain a binary value representing the decision of the ensemble of LUTs/decision trees. In turn, this entire ensemble of LUTs was considered to be a new independant classifier and the Adaboost algorithm was applied again. By reapplying Adaboost to larger and larger ensembles of classifiers, Chidambaram et al. were able to consider

more input features in their decisions. Moreover, the weighted sum/thresholding operations could be mapped to LUTs making the entire calculation a LUT-based operation. Figure 3.6 illustrates the PoET-BiN architecture. This heuristic was tested on the fully-connected layers of LeNet and VGG-11 networks on the MNIST, CIFAR10 and SVHN datasets. The resulting accuracies were comparable to PolyBiNN and BinaryNet. Compared to BinaryNet, the energy consumption at inference was reduced by a factor of 100× - 10000× depending on the dataset, making it very power-efficient.

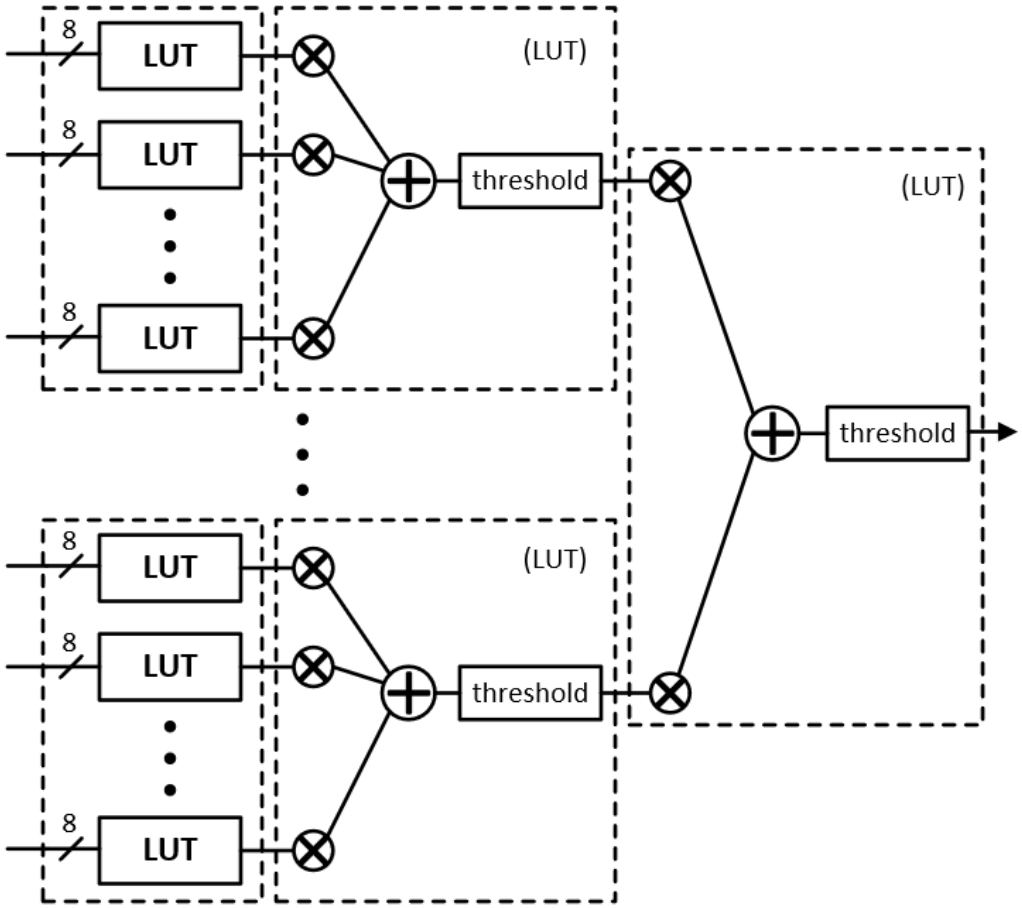

Figure 3.6 PoET-BiN architecture using 8-input LUTs.

3.3 Binary Nets and Speech Recognition

Zhang et al. have explored quantization of neural networks on keyword spotting with the hope of achieving inference in a reasonable amount of time and with reasonable accuracy on a resource-constrained device [24]. However, they only quantized weights and activations to 8

bits in fixed-point formats. To our knowledge, binarization was never attempted on keyword spotting tasks using neural networks.

The only known attempt at applying binarization to speech recognition was performed by Qian et al. [46] [47]. In this work, acoustic models were trained using binary weights and activations on a large vocabulary task. The objective of the proposed networks was to classify context-dependant phonemes when presented with conversational speech. The input of the networks was composed of 11 frames of 36 log-Mel spectrogram features along with their delta and delta-delta coefficients. In each case, the output was a softmax layer with 8876 classes representing the context-dependant phonemes. A total of three networks were proposed: a DNN, a CNN and a very deep CNN. The binarization was performed using the methods of Hubara et al. [33]. In other words, the weights and activations were binarized and the STE was used during backpropagation. Batch normalization was used, since this helped to recenter the distribution of activations around 0. To binarize the weights, a stochastic version of the the sign function was applied. The networks were tested on a few datasets including the NIST Hub5 Switchboard dataset, which is composed of telephone conversational speech. Notably, Qian et al. attempted to apply Knowledge Distillation (KD) to improve the performance of their binary networks [48]. KD is a technique used which leverages the information learned by a large accurate teacher network to train a smaller student network. When using cross-entropy loss, hard labels are used. In other words, the information only backpropagates according to the probability predicted on the class representing the ground truth. When using knowledge distillation, the information is not backpropagated according to the hard labels of the dataset, but rather with the output probabilities of the teacher network (soft labels). Instead of attempting to classify labels correctly, the student network attempts to replicate the predictions of the teacher network. When using this technique with a very deep full precision CNN as the teacher, they obtained improvements over training the same networks using hard labels. A summary of their results are illustrated in Table 3.2 [47]. Knowledge distillation reduced the WER by over 1%.

To our knowledge, binarization was never attempted on a keyword spotting task. The works presented in this literature review will hopefully serve as a good introduction to the proposed solutions.

Table 3.2 Comparison of full Precision vs Binary Networks and the Effects of Knowledge Distillation [47].

Model	Word Error Rate (%)
Full Precision DNN	15.6
Full Precision CNN	15.1
Full Precision Very Deep CNN	13.8
Binary DNN	20.8
Binary DNN with KD	19.7
Binary CNN	21.1
Binary CNN with KD	18.8

CHAPTER 4 BINARY SPEECH FEATURES FOR KWS TASKS

This chapter presents an article featured in a poster session during Interspeech 2019 [49], a peer-reviewed conference on speech processing and applications. The main objective of this work was to explore binary representations of speech which could work well with BNNs.

4.1 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) [50]. 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 [29]. 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 [33] [34] . 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 [51]. 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 [33], 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 [37] and improved BinaryNets [41] 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 4.2 presents some related works in the field of KWS. Section 4.3 explains traditional speech representations. Section 4.4 details the proposed algorithms to produce new representations. Section 4.5 describes our test setup. Results are reported in Section 4.6. Finally, Section 4.7 concludes this work and proposes future research directions.

4.2 Related works

Modern implementations of KWS algorithms either use sequence to sequence models such as Long Short-Term Memory (LSTM) based networks [14] because they can capture the temporal progression of utterances or Convolutional Neural Network (CNN) based models [5] 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 [18]. On a 10-word classification task, Tang et al. obtained over 95% accuracy using a ResNet [19]. Qiu Lin et al. then proceeded to improve the same ResNet by generating many models of

reduced sizes through hyper-parameter optimization [20]. 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 [16] 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 [33] [34] [37] [41]. 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 [24].

4.3 Background

4.3.1 Traditional Speech Representations

To represent speech data, traditional algorithms, such as Hidden Markov Models – Gaussian Mixture Models (HMM-GMMs) used Mel Frequency Cepstral Coefficients (MFCCs) [7]. 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.

4.3.2 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 4.1. In this paper, the Python library Librosa v0.6.2 was used to compute the Mel spectrogram [52]. 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.

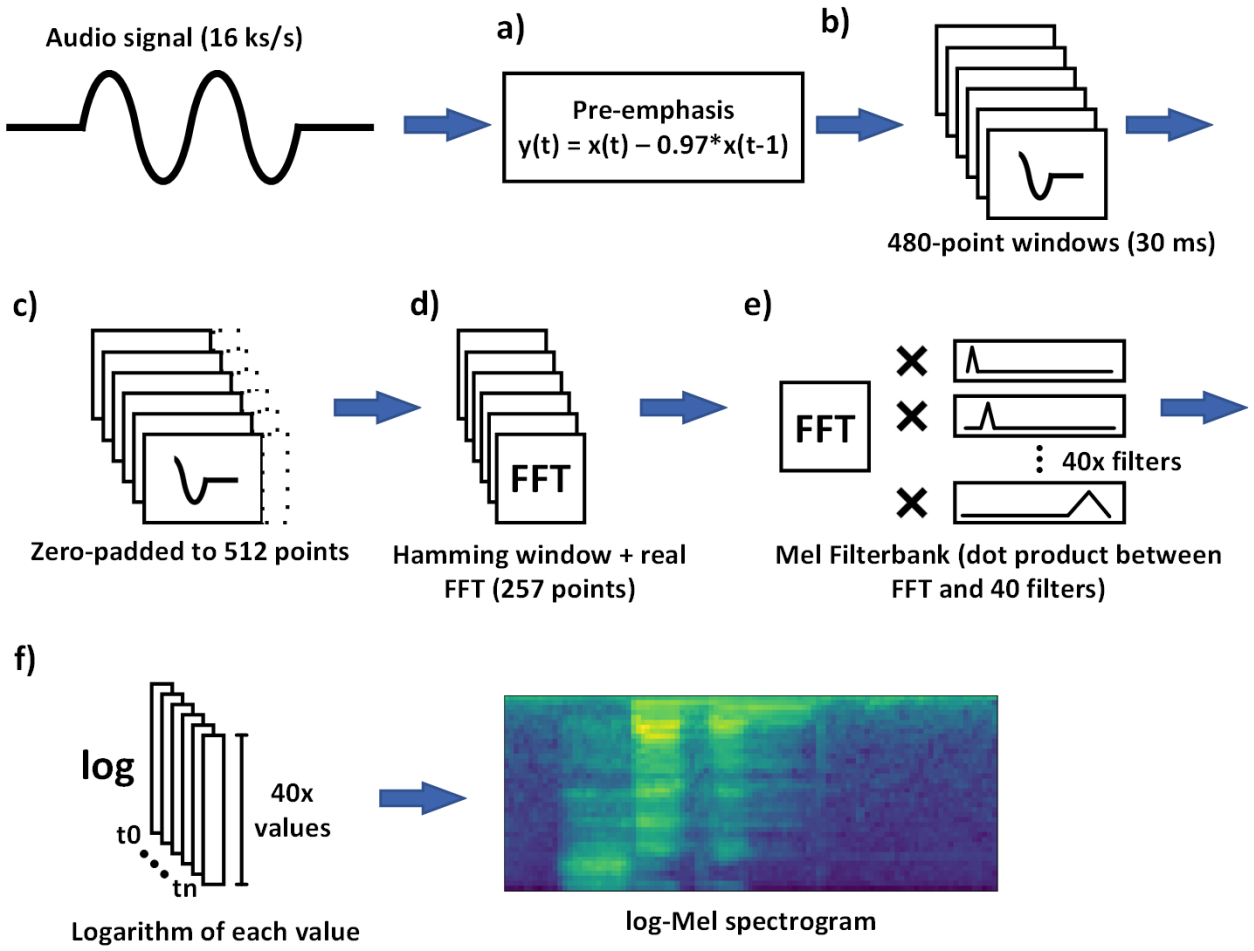

Figure 4.1 Steps to compute the log-Mel spectrogram of a speech signal or sound wave.

4.4 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 4.4.1 and the latter in Section 4.4.2. The power-variation spectrogram is also subdivided into two distinct sections introducing a 1-channel and 2-channel representation.

4.4.1 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 40×101 pixels. The code used to compute the log-Mel spectrogram can be found in a GitHub repository produced by Tang et al. [17] [19] . 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 ≫ (8-desired_bit_width)

4.4.2 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 3×3 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 [53]. In other words, measuring the change in power of sound may isolate important information.

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

```
1:  output_image ← array[40x101 of '0s']
2:  data ← compute log-Mel spectrogram() // 40x101 image
3:  data ← data - (max(data) - delta_power)
4:  data ← max(0, data)
5:  data ← floor(data*(255/delta_power))
6:  ref_value ← data[:, 0]
7:
8:  for i in range 0 to 100 do
9:      variation ← data[:, i+1] - ref_value
10:     for j in range 0 to 39 do
11:         if variation[j] > threshold then
12:             output_image[j, i] ← 1
13:             ref_value[j] ← data[j, i]
14:         else if variation[j] < -threshold then
15:             output_image[j, i] ← -1
16:             ref_value[j] ← data[j, i]
17:         else
18:             output_image[j, i] = 0
19:         end if
20:     end for
21: end for
```

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.

4.5 Experiments

Every input representation was tested using the ResNet architecture proposed by Tang et al. [19]. The quantity of feature maps for each respective layer has changed to match the EdgeSpeechNet-A model of Qiu Lin et al. [20]. The network is illustrated in Figure 4.2.

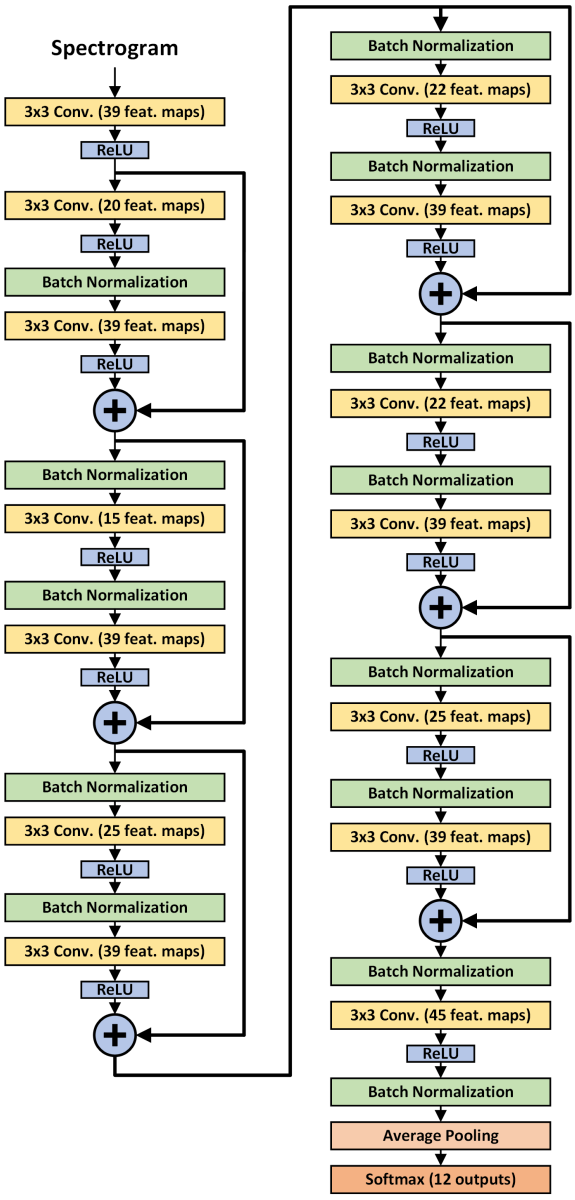

Figure 4.2 Architecture of EdgeSpeechNet-A. This ResNet is used to test the different input representations proposed in this paper.

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 [16]. 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 hyper-parameters (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. [19] [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).

4.6 Results

4.6.1 Low-precision Log-Mel Spectrogram

An example of the log-Mel spectrogram is illustrated in Figure 4.3 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.

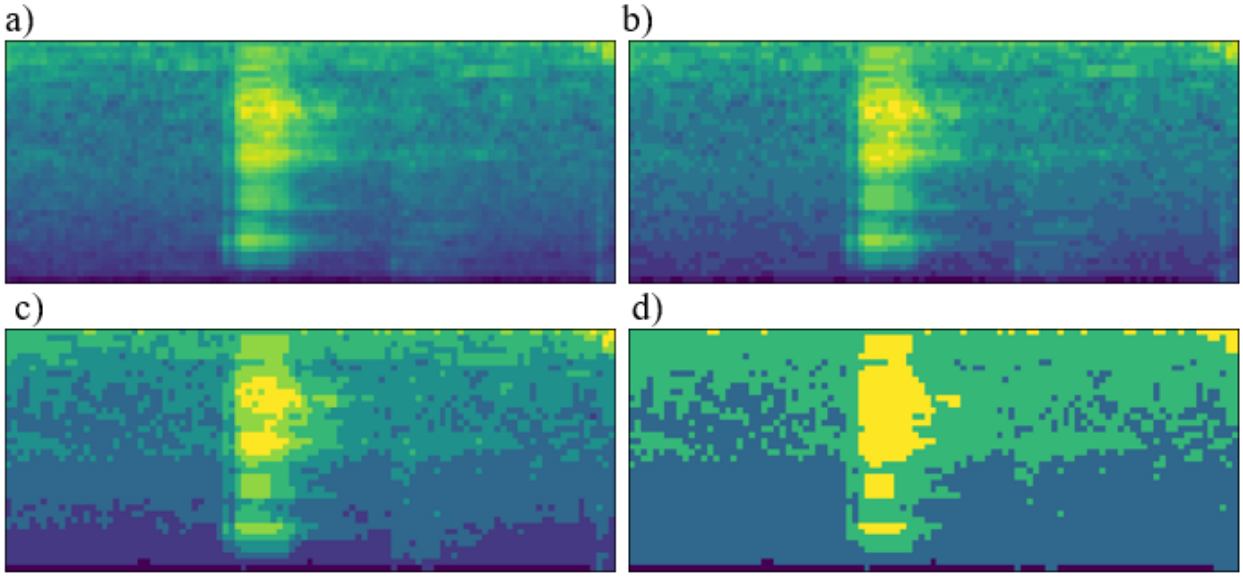

Figure 4.3 Quantized representations of the log-Mel spectrogram for the word “up”. a) 8-bit rep. b) 4-bit rep. c) 3-bit rep. d) 2-bit rep.

The results for the quantized inputs are summarized in Table 4.1. The accuracy claimed by

the EdgeSpeechNet-A model is approximately 96.8% using full-precision MFCCs as inputs [20]. 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 4.3. 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 4.1 Accuracy of Quantized Inputs

Precision	**Representation**	**Accuracy (%)**
Full-precision	MFCCS	96.8
8 bits	Log-Mel spectrogram	96.8 ± 0.1
4 bits	Log-Mel spectrogram	96.6 ± 0.1
3 bits	Log-Mel spectrogram	96.2 ± 0.5
2 bits	Log-Mel spectrogram	95.2 ± 0.3

4.6.2 Power-variation Spectrogram

An example of the 1-channel power-variation spectrogram is illustrated in Figure 4.4. 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 4.2. 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%).

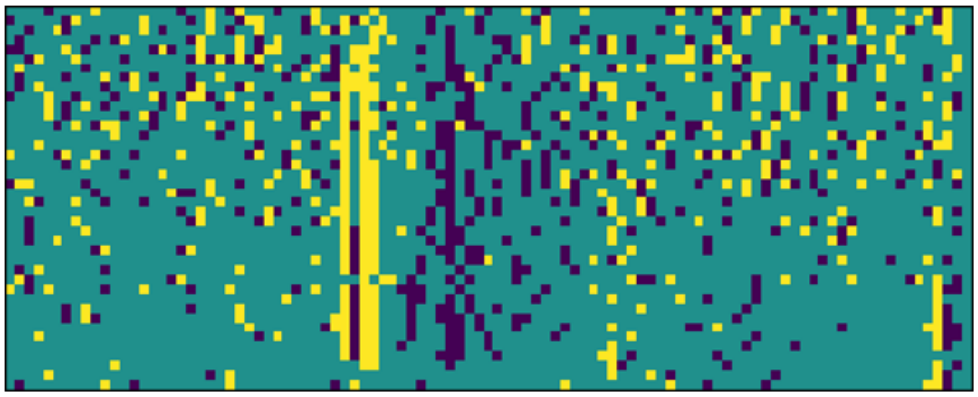

Figure 4.4 Ternary 1-channel power-variation spectrogram for the word "up". A yellow pixel is equal to 1, a purple pixel to -1 and a green pixel to 0.

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

Table 4.2 Accuracy of Power-Variation Spectrogram

of Channels	Precision (per channel)	Accuracy (%)
1	2 bits	95.8 ± 0.1
2	1 bit	95.6 ± 0.3

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 4.5.

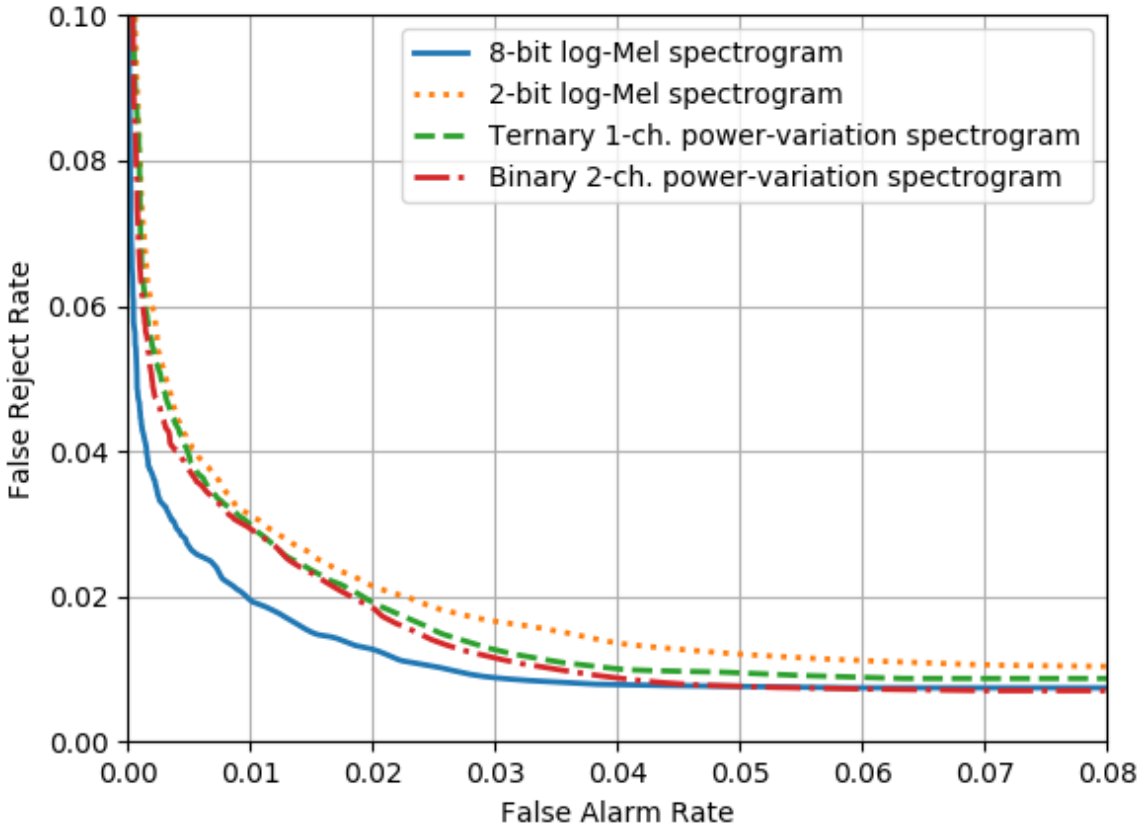

Figure 4.5 ROC curves for various input representations.

4.7 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 1× 2-bit channel or 2× 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.

CHAPTER 5 BINARIZED NEURAL NETWORKS AND KWS

This chapter explores the use of binary neural networks for keyword spotting. Building upon the binary speech representations featured in Chapter 4, subsection 5.1 proposes a novel 3-channel representation which offers better accuracy. Subsection 5.2 presents a binary neural network architecture capable of classifying keywords with reasonable accuracy. Finally, subsection 5.3 explores hardware implementations of the binary architecture using the ideas proposed in PoET-BiN [43].

5.1 Expanding Binary Speech Features

The power-variation spectrogram proposed in section 4.4 uses the dynamic information of the log-Mel spectrogram by comparing the power levels of current frames to the power levels of previous frames. This, however, ignores all the static information; i.e., the amplitude of the signal over time. To circumvent this lack of information, an additional channel is added to the representation using Algorithm 1 of Chapter 4 with a *desired_bit_width* of 1. In summary, the log-Mel spectrogram is normalized to fit within the 0 to 255 range. The MSB of this unsigned 8-bit representation is then extracted for every time/frequency component to generate an additional channel. The resulting channel is an indicator of whether the power is high or low.

The final binary representation of speech used is illustrated in Figure 5.1. This representation contains the static power in the first channel, the increases in power according to Algorithm 2 in the second channel and the decreases in power in the third channel.

In general, the log-Mel spectrogram is normalized; i.e., the average power is removed from every component. In this case, the values are normalized by substracting the maximum power to all components. This results in a very small change in computational costs. The dynamic features (the other 2 channels) can be calculated while the data is acquired by the ADC, resulting in a negligible amount of operations (40 values to compare and update for each new time frame).

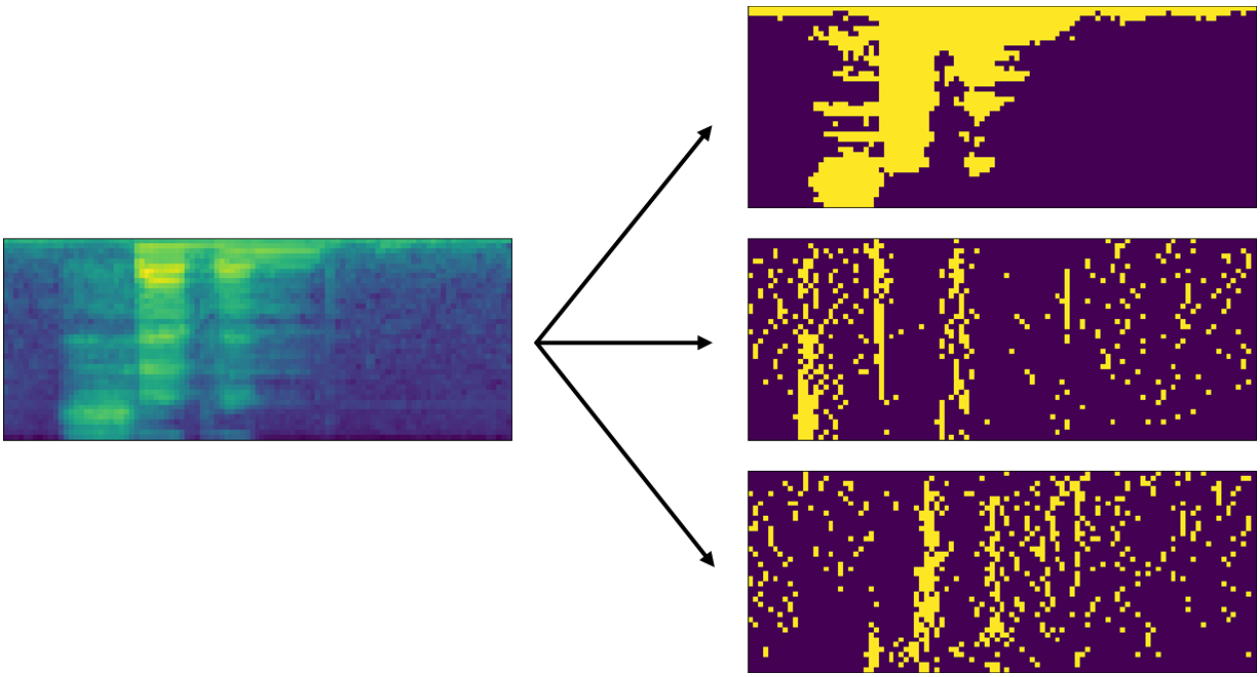

Figure 5.1 Converting the log-Mel spectrogram (middle-left) to a 3-channel representation with static power in the first channel (top-right), increases in power in the second channel (middle-right) and decreases in power in the third channel (bottom-right).

This new 3-channel representation was tested on the Google Speech Commands using the EdgeSpeechNet-A architecture produced by Lin et al [20] (see Figure 4.2). Using the same hyperparameters and test conditions as the power-variation spectrogram from Chapter 4, the accuracy of the 3-channel representation was evaluated. The results are reported in Table 5.1.

Table 5.1 Accuracy of the Power-Variation Spectrogram with Static Information on a Full-Precision ResNet

Representation	**Accuracy (%)**
log-Mel spectrogram (8 bits) [49]	96.8 ± 0.1
2-channel (dynamic info.) [49]	95.6 ± 0.3
3-channel (dynamic + static info.)	96.3 ± 0.2

Overall, adding a third channel with information on the static power increases the classification accuracy by 0.7% compared to only using the dynamic information (2-channels). However, introducing extra channels to the input does not considerably increase the amount of computations. For instance, a representation quantized to 8-bits could offer a classification accuracy of 96.8%. The advantage of this representation is in the binary nature of the data.

In other words, this representation is more likely to be compatible with a binarized neural network; i.e. offer higher classication accuracy when used with a BNN. This hypothesis is explored in Section 5.2.

5.2 Proposed Binary Neural Network

Section 5.2 serves two purposes: present a BNN capable of performing keyword spotting and compare the 3-channel power-variation spectrogram to the conventional log-Mel spectrogram when used on a BNN.

5.2.1 Network architecture

Inspired by the networks produced by Tang et al. [19], a binary version of ResNet15 is proposed. The network contains a total of 14 convolution blocks followed by an average pooling and full-precision fully-connected layer. Each convolution block performs a binary convolution; i.e. the weights and activations are either -1 or 1. The convolution is followed by the PReLU activation function which acts as a scaling factor to improve the expressivity of BNNs [40] [54]. Batch-normalization is then applied to recenter the distribution of activations. Finally the values are thresholded using the sign function after applying a scaling term and a bias to the activations. The network and the operations contained in a convolution block are illustrated in Figure 5.2.

Unlike a traditional ResNet, the activations are not added to the current layer's output. Instead the values are concatenated. If binary values (-1 or 1) were added to other binary values, then the resulting activations would be in the {-2, 0, 2} set. A second sign function would be required to binarize the values again in order to perform a binary convolution in the following layer. An alternative could be to pass full-precision residuals and only perform binarization after the addition. This technique was applied in Bi-Real Net produced by Liu et al. [39]. However, using a rather large spectrogram as an image (40 x 101 values) with many channels (up to 32), the storage of floating-point residuals during inference would be very demanding for devices with limited memory.

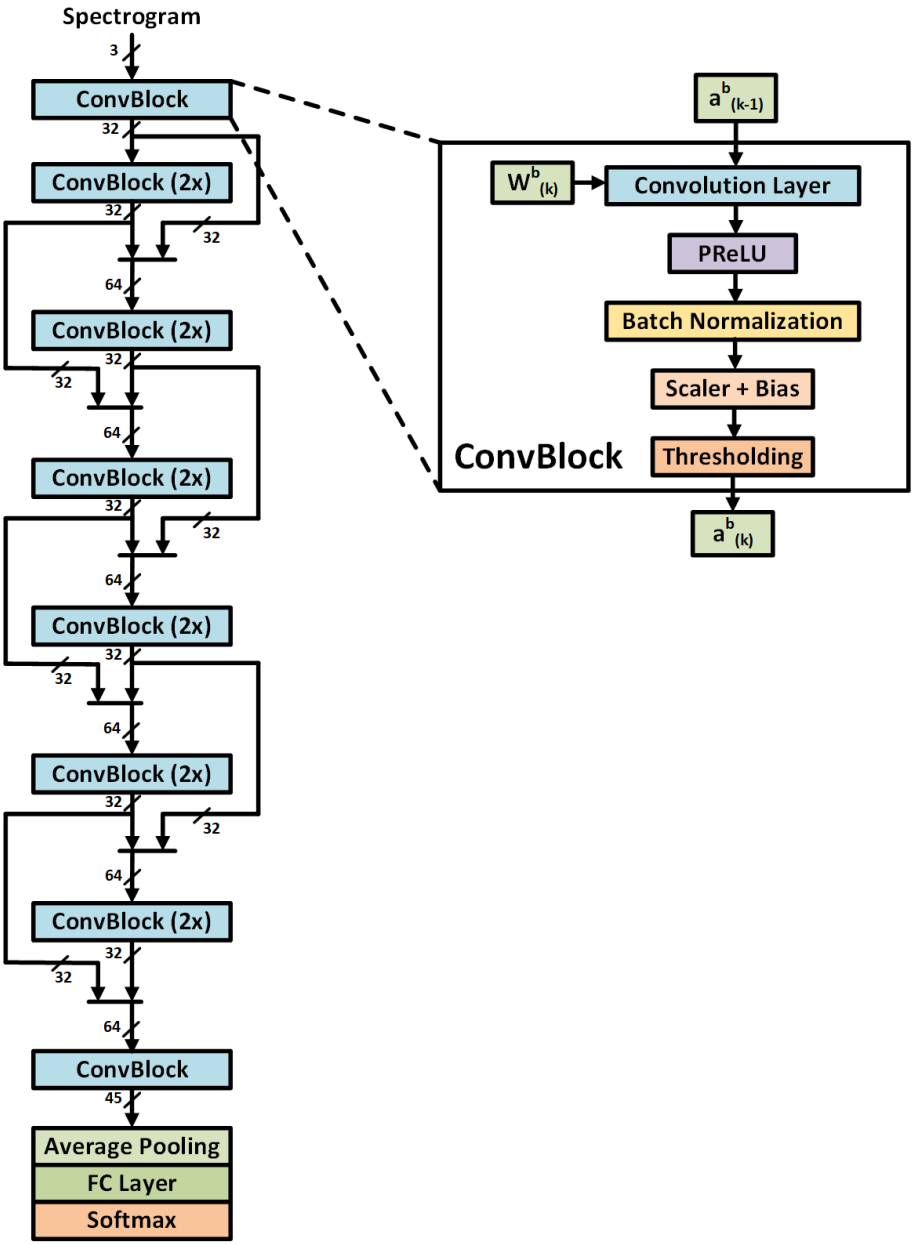

Figure 5.2 Proposed Binary Neural Network for keyword spotting (left). The ConvBlock used (right). The first ConvBlock does not use batch normalization. The scaler is a learnable parameter which multiplies the activations and the bias is a learnable parameter which is added to the activations. Thresholding is performed using the sign function. k represents the layer number and the b indice represents a binarized value.

The number of input channels to each layer was adapted to be 32 or 64. Many embedded processors make use of 32 bit registers and have byte addressable memory. For these reasons, it is much easier to work with sizes that are powers of 2. The detailed number of channels

and hyperparameters is summarized in Table 5.2. The convolutions in this type of ResNet use dilations (the kernel expands to obtain a larger receptive field). This prevents the use of pooling layers which may eliminate important information. The dilation values are identical to the ones proposed by Tang et al. [19]. In all cases, the images are padded with alternating channels of 1s and -1s. This was recommended by Guo et al. [38] [55]. The final fully connected layer of the network uses full-precision weights and activations. This is not a problem since it is only one dense layer with 45 inputs and 12 outputs resulting in approximately 540 floating point multiply-adds.

Table 5.2 Hyperparameters and Calculations of the Proposed Binary Resnet

ConvBlock Layer	**In ch.**	**Out ch.**	**Dilation**	**Multiply-adds.**	**Weights**
0	3	32	1	3,490,560	864
1	32	32	1	37,232,640	9216
2	32	32	1	37,232,640	9216
3	64	32	2	74,465,280	18,432
4	32	32	2	37,232,640	9216
5	64	32	2	74,465,280	18,432
6	32	32	4	37,232,640	9216
7	64	32	4	74,465,280	18,432
8	32	32	4	37,232,640	9216
9	64	32	8	74,465,280	18,432
10	32	32	8	37,232,640	9216
11	64	32	8	74,465,280	18,432
12	32	32	16	37,232,640	9216
13	64	45	16	104,716,800	25,920
Total	-	-	-	741,162,240	183,456

This network was tested on the 10-keyword objective of the Google Speech Commands V1 [16]. The learning rate was initially set to 0.01 and was reduced by a factor of 10 approximately every 9 epochs for a total of 30 epochs (with early stopping). Batches of 64 were used throughout the training. Binarization was performed using the recommendations of the original BNN paper [33]. The Adam optimizer was used during training [56]. All the reported figures in subsection 5.2.2 are the average of 5 tests using this proposed network with these configurations.

5.2.2 Tests and Ablation Study

The first objective was to evaluate the proposed network when used on the 3-channel power-variation spectrogram. The resulting accuracy was evaluated at 94.3%. In contrast, when

this representation was tested on a similar full-precision network, the accuracy was 96.3%. The second objective was to compare the 3-channel power-variation spectrogram with the log-Mel spectrogram when used on the proposed binary network.

Power-variation Spectrogram vs the log-Mel Spectrogram

To this point, the 3-channel representation has only been tested using a full-precision network, which was not its intended use-case. When tested on a full-precision ResNet (EdgeSpeechNet-A [20]), the log-Mel spectrogram quantized to 8 bits was better in terms of accuracy (96.8% vs 96.3%). However, when tested on the proposed binarized neural network, the 3-channel power-variation spectrogram offered better results (92.6% vs 94.3%). The results are summarized in Table 5.3.

Table 5.3 Accuracy of the Proposed BNN According to the Input Representation.

Representation	**Accuracy (%)**
Power-variation spectrogram	94.3 ± 0.3
Log-Mel spectrogram	92.6 ± 0.2

Ablation Study

This section presents the first known attempt at binarizing a neural network for keyword spotting. For this particular use-case, binarization seems to produce, at best, networks which offer accuracies which are 2-3% inferior to the state-of-the-art. To have a better understanding on the decisions that led to this particular architecture, ablation studies were conducted. In other words, a single hyper-parameter or function was removed/modified at a time, and the accuracy was measured. Notably, the network was tested without the PReLU activation function and without the scaling and bias factors. In another scenario, the number of output filters was reduced by a factor of 2. A smaller equivalent version of the ResNet with fewer layers was also tested (8 ConvBlocks instead of 14). Finally, knowledge distillation was tested using the EdgeSpeechNet-A model [20] as the teacher network. The hard labels were given as much importance as the soft labels ($\alpha = 0.5$), since this offered the best results in the work produced by Qian et al [47]. The 3-channel power-variation spectrogram was used in all cases. The results of these modifications are summarized in Table 5.4.

The network's accuracy seems to depend mostly on the quantity of feature maps. Since every individual weight and activation contains less information compared to a network using full-precision values, more feature maps may be needed to represent features adequately. The

Table 5.4 Ablation study: Accuracy of the Proposed BNN According to Modification.

Representation	Accuracy (%)	Variation (%)
Base network (reference)	94.3 ± 0.3	-
Without PReLU	93.4 ± 0.8	- 0.9
Without Scaling/Bias	94.2 ± 0.3	- 0.1
With $2x$ less feature maps	91.1 ± 0.5	- 3.2
With $\approx 2x$ less layers	92.9 ± 0.5	- 1.4
With Knowledge Distillation (α=0.5)	94.0 ± 0.3	- 0.3

use of the PReLU activation also seems to positively affect the classification accuracy. This coincides with the findings of Tang et al. [40] and Bulat et al. [54], which claim that PReLU adds a scaling factor comparable to XNOR-Net [37]. The scaling and bias values added were meant to shift the distribution and not solely rely on batch-normalization to threshold at a proper value. However, they had little influence on the overall classification accuracy. Knowledge distillation slightly degraded classification accuracy. Since the results are within the standard deviation, it likely had close to no effect.

5.3 Exploring Hardware Implementations : PoET-BiN for CNNs

As seen in section 5.2, binarization may degrade the classification accuracy to a point where smaller networks quantized to 8-bits, such as Mobile-Net, offer comparable accuracies [24] when used on embedded devices such as microcontrollers. Considering the fact that most MCUs do not support popcount instructions and other operations relevant to BNNs, they may not be the best fit for this type of task. Perhaps, the use of specialized hardware, such as FPGAs, may offer a competitive edge for BNNs used on KWS.

In chapter 3, the PoET-BiN algorithm was introduced [43]. This algorithm replaced the MLP layers of VGG and LeNet networks with stacked layers of decision trees which mapped to the LUTs of an FPGA. Since FPGA primitives were used directly, the power consumption was reduced by over 3 orders of magnitude compared to FPGA implementations offering similar accuracies. This algorithm, however, was only applied to the fully-connected layers of the network. Section 5.3 will attempt to expand this idea to the convolution layers of the proposed Binary ResNet for KWS.

5.3.1 Building the Data to Train PoET-BiN on CNNs

PoET-BiN can be interpreted as a black box which maps binary inputs to binary outputs. In the original paper, the black box's inputs were the flattened binarized outputs of the

convolution layers. The networks had a sparsely connected output layer which generated logits (the outputs of a neural network before going into the softmax layer) using 6 or 8 bits of precision. The output of the black box was the input of this sparsely connected layer. Figure 5.3 illustrates the flow of calculations proposed in the original PoET-BiN paper. These black boxes were referred to as Reduced Input Neural Circuits (RINC classifiers.)

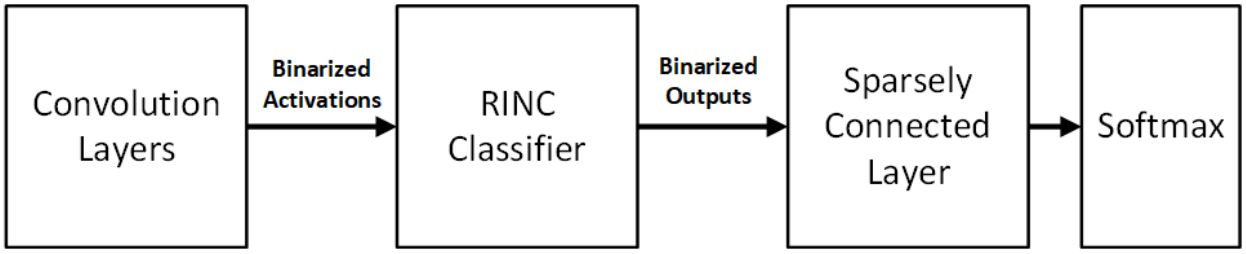

Figure 5.3 Original use-case of the RINC classifier. The classifier replaced the fully-connected layers at the end of a VGG/LeNet network. To accomodate binary decision trees, the outputs of the convolution layers were flattened and binarized. The sparsely connected layer was used to generate higher bit precision logits.

In the case of a CNN, the black boxes (RINC classifiers) can be translated to convolution kernels. For every position in the image, the kernel's inputs represents new input data and the resulting activations (each pixel in the resulting feature map) represents the output data. For an image with dimensions of 40×101 and a kernel that performs strides of 1, 4040 data samples are generated. Considering that each image produces over 4000 data samples, only a fraction of the dataset is required to train the RINC classifiers. During the tests described in the following section, 32 images were used since increasing this number had very little effect on the accuracy. This corresponds to 129,280 samples per kernel (32×4040). Figure 5.4 illustrates how the data samples were extracted from the input image.

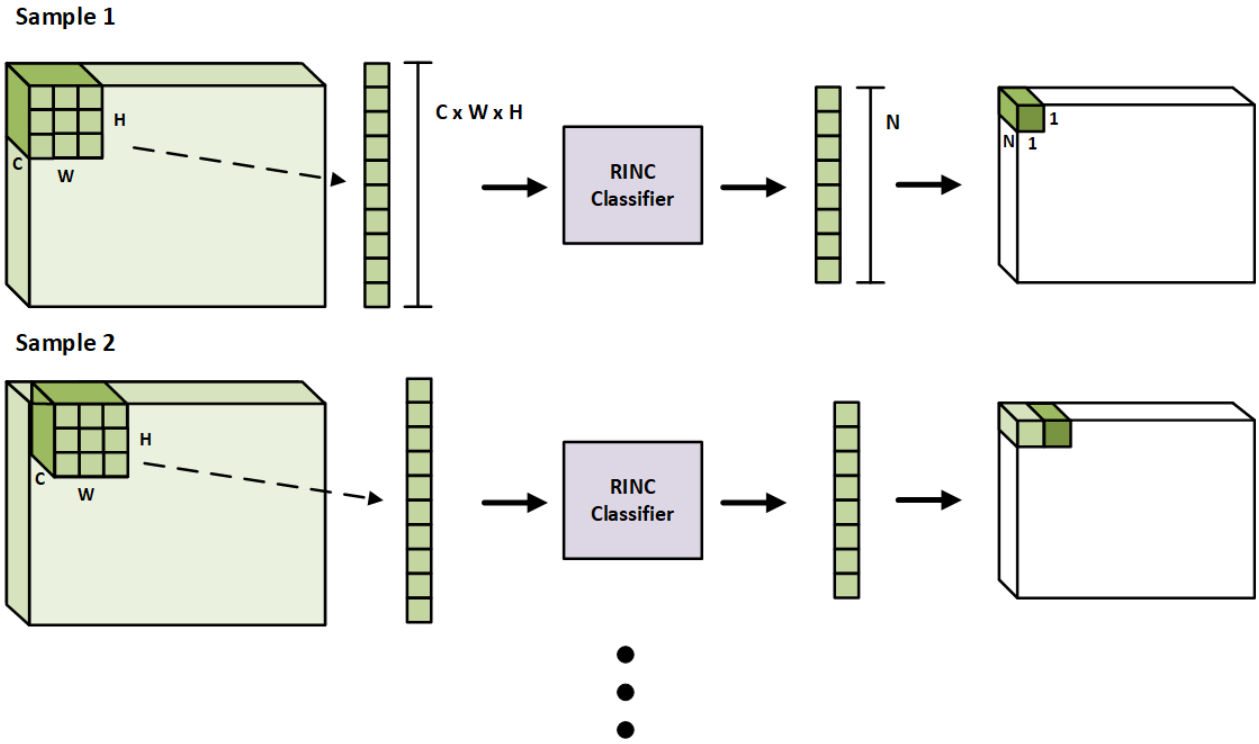

Figure 5.4 Replacing the convolution kernels of a CNN with a RINC classifier. For each kernel position, the data is flattened and injected into the RINC classifier. A single image can generate many training samples since every kernel position represents new data. C represents the number of input channels, W the width of the kernel, H the height of the kernel and N, the amount of filters used.

5.3.2 Replacing a Random Convolution Layer with PoET-BiN

To test PoET-BiN applied to CNNs, the network proposed in Section 5.2.1 was used with a few modifications. One of the residual blocks was removed, leaving a total of 12 convolution blocks and the quantity of filters was reduced by a factor of 2. This was done to reduce the training time since reading LUTs and constructing addresses are not operations supported with CUDA backends by frameworks like PyTorch. Since PoET-BiN only requires binary values for the inputs and outputs of the filters, the weights and intermediate operations used full-precision values. This was done to improve the classification accuracy. The network architecture is illustrated in Figure 5.5.

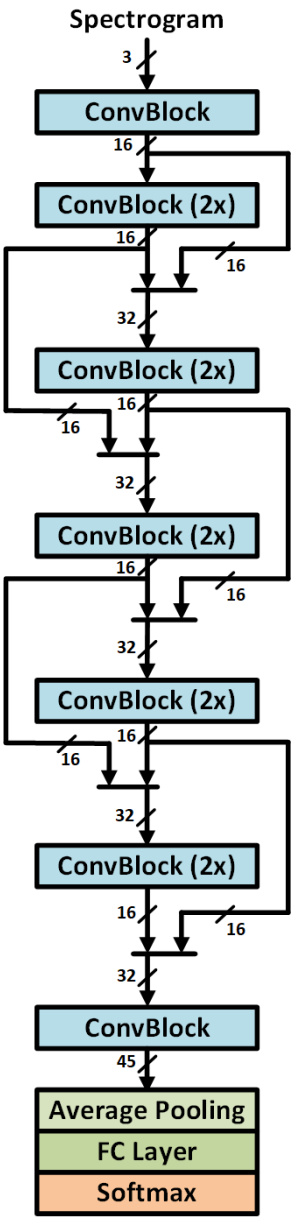

Figure 5.5 Network used to test PoET-BiN. The ConvBlocks are identical to the ones used in Section 5.2.1 with the exception of the weights which are full-precision. This version of the network contains a total of 12 ConvBlocks. The second ConvBlock was replaced with a RINC-1 classifier.

As a first test, a single convolution layer was replaced using PoET-BiN with RINC-1. RINC-1 simply means there are 2 layers of LUTs. The first layer of LUTs is composed of binary decision trees and the second layer of LUTs is a weighted sum of the predictions produced using the Adaboost algorithm. In all cases, the LUTs had 8 inputs. The largest filters used had 32 channels, a width of 3 and a height of 3. In other words, for every position within the

image, the filter had 288 inputs ($32 \times 3 \times 3$). Since the LUTs had 8 inputs and RINC-1 was used, 64 (8×8) out of 288 features were being used by the PoET-BiN algorithm. Out of the entire network, only the second convolution block was replaced. Using these configurations, the RINC-1 classifier was able to predict the correct features for all channels 95% of the time (the RINC-1 student could replicate the teacher's outputs 95% of the time for that specific layer). The dataset used was once again the Google Speech Commands. After replacing the second layer of the network with PoET-BiN, the classification accuracy on the 10 keyword task did not change (93.6% in all cases). Figure 5.6 illustrates one channel of the output of the second layer before and after using the RINC-1 classifier.

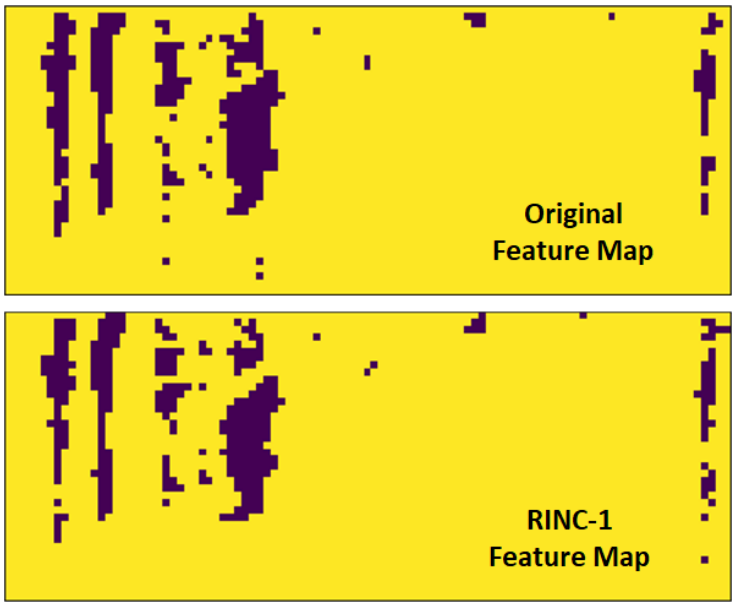

Figure 5.6 Output Feature Map of the second ConvBlock when extracted from the original network (top), when extracted from the RINC-1 kernel (bottom).

Applying PoET-BiN to a convolution layer is equivalent to introducing noise/bias by approximating the actual convolution and non-linear functions. Replacing a single layer does not seem to have an effect. However, this is not surprising since not all layers of a neural network have as much importance [57]. Zhang et al. have noticed that some layers of a CNN can be re-initialized or randomized with little loss to the classification accuracy. This isn't true for all layers. However, this may explain why the classification accuracy is unaffected. The next step is to apply PoET-BiN to all the convolution layers successively.

5.3.3 Completely Replacing a Network with PoET-BiN using RINC-1

A naive approach to implementing PoET-BiN on a CNN would be to directly replace every convolution layer with its RINC-1 equivalent. However, everytime a ConvBlock is replaced, the resulting layer is an approximation of what the teacher network learned. A better approach would be to gradually replace the layers and to retrain the rest of the network for a few epochs. This process was used on the network featured in Figure 5.5. The retraining of the subsequent layers was done using a learning rate of 10^{-5} for 10 epochs.

After replacing all convolution kernels with LUT-based RINC-1 classifiers, the resulting accuracy of the entire network was 76.9% (down from 93.6%). The results are summarized in Table 5.5. These results include the network accuracy according to the number of layers replaced with RINC-1 classifiers. The input/output channel dimensions are also included. Overall, this network would require 1,989x 8-input LUTs for the convolution kernels alone.

Table 5.5 Accuracy According to the Number of Layers Replaced with RINC-1.

Layers replaced	Replaced Layer Dimensions	Accuracy (%)	Variation (%)
0	-	93.6	-
1	$3 \rightarrow 16$	93.0	- 0.6
2	$16 \rightarrow 16$	92.3	- 0.7
3	$16 \rightarrow 16$	92.9	+ 0.6
4	$32 \rightarrow 16$	91.2	- 1.7
5	$16 \rightarrow 16$	89.9	- 1.3
6	$32 \rightarrow 16$	88.6	- 1.3
7	$16 \rightarrow 16$	88.4	- 0.2
8	$32 \rightarrow 16$	86.1	- 2.3
9	$16 \rightarrow 16$	86.1	- 0.0
10	$32 \rightarrow 16$	81.7	- 4.4
11	$16 \rightarrow 16$	80.6	- 1.1
12	$32 \rightarrow 45$	76.9	- 3.7

The biggest drops in accuracy seem to occur when the layers have more inputs (32 channels instead of 16). Figure 5.7 illustrates this effect. RINC-1 classifiers with 8-input LUTs only consider 64 input features. This may become a problem for layers with more input channels since these layers hold more features. The PoET-BiN algorithm is capable of increasing the quantity of inputs by re-applying the Adaboost algorithm to larger ensembles of classifiers. If RINC-2 classifiers are used (Adaboost is applied twice), then the total number of input features considered rises to 512 ($8 \times 8 \times 8$). The same experiment using RINC-2 classifiers is explored in section 5.3.4.

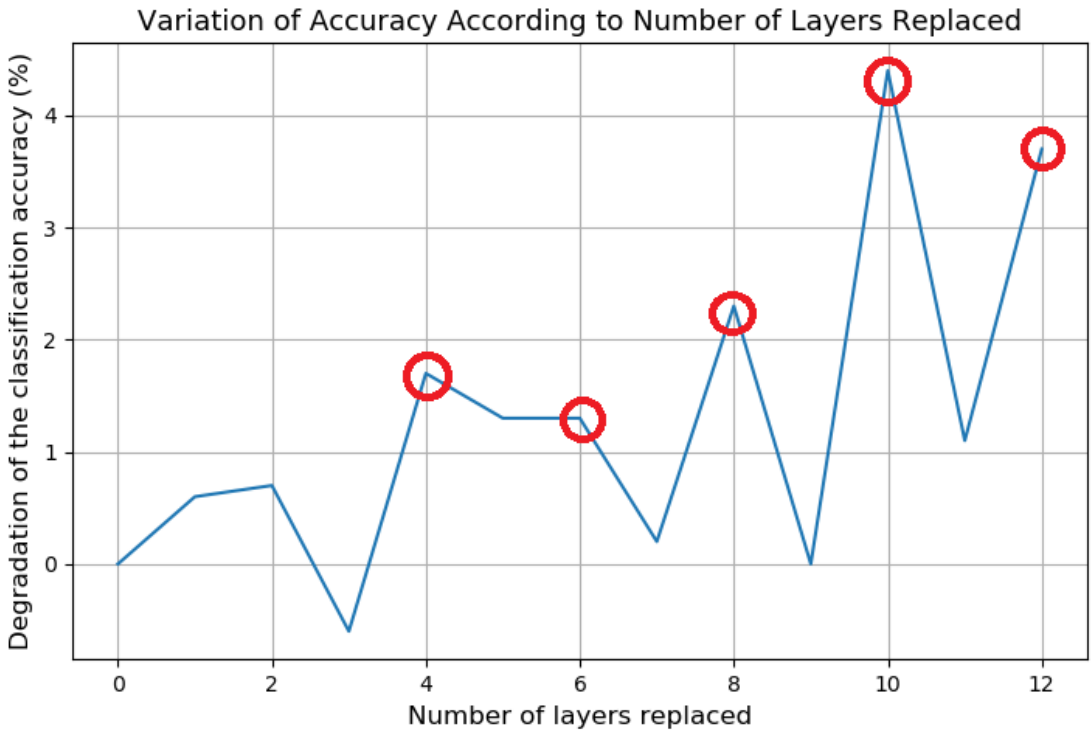

Figure 5.7 Degradation of the classification accuracy compared to the previous layer according to the number of layers replaced. A higher value means a larger degradation. The red circles represent the layers for which the inputs had more input channels (32 instead of 16).

5.3.4 Completely Replacing a Network with PoET-BiN using RINC-2

The same procedure was applied using RINC-2 classifiers to the 12 layer ResNet. The results are illustrated in Table 5.6. The final classification accuracy of the network with ConvBlocks replaced with RINC-2 classifiers is 89.7%. This is a 12.8% improvement in terms of accuracy compared to using RINC-1 classifiers. However, the use of RINC-2 classifiers raises the 8-input LUT count to 16,133.

Increasing the quantity of inputs considerably improved performance (89.7% vs 76.9% accuracy using RINC-2 and RINC-1 classifiers respectively). However, the LUT count increased considerably. Perhaps, it is possible to reduce the amount of total layers by merging subsequent convolution layers. If layers are to be combined, the receptive field of the first layer will become larger as illustrated in Figure 5.8. Also, the network used thus far makes use of dilations. The convolution kernels are not simply 3×3. Combining many layers would require very large kernels to match the correct input sizes. To test out the idea of merging convolution layers, a different network needs to be proposed.

Table 5.6 Accuracy According to the Number of Layers Replaced with RINC-2.

Layers replaced	Replaced Layer Dimensions	Accuracy (%)	Variation (%)
0	-	93.6	-
1	$3 \rightarrow 16$	93.2	- 0.4
2	$16 \rightarrow 16$	92.9	- 0.3
3	$16 \rightarrow 16$	93.2	+ 0.3
4	$32 \rightarrow 16$	92.8	- 0.4
5	$16 \rightarrow 16$	92.0	- 0.8
6	$32 \rightarrow 16$	92.8	- 0.2
7	$16 \rightarrow 16$	92.1	- 0.7
8	$32 \rightarrow 16$	91.7	- 0.4
9	$16 \rightarrow 16$	91.8	+ 0.1
10	$32 \rightarrow 16$	90.6	- 1.2
11	$16 \rightarrow 16$	90.2	- 0.4
12	$32 \rightarrow 45$	89.7	- 0.5

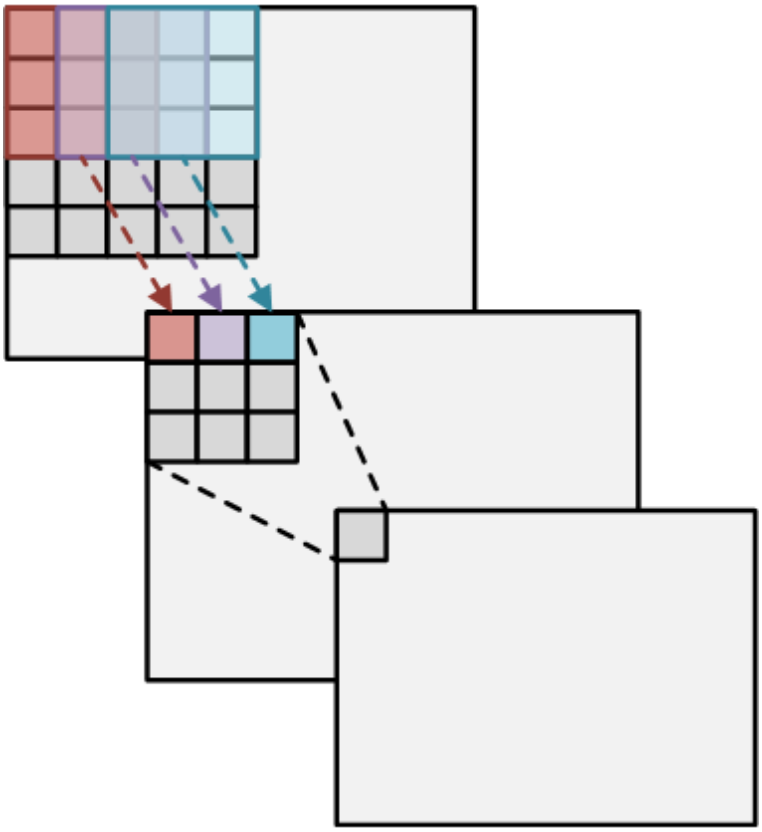

Figure 5.8 Receptive field after applying 2 convolution layers with 3×3 kernels using strides of 1. If two layers were to be combined, a 5×5 kernel would be required to capture all the same features.

5.3.5 Merging Convolution Layers

To test the idea of merging convolution layers, a network with 7 convolution layers was implemented. Instead of using dilations, the convolution kernels used strides of 2 to reduce the feature map size. The network was cut into 3 blocks or layers (L0, L1, L2). Each block contained binary inputs and outputs and was designed to be replaced with a RINC-2 classifier. Since the intermediate calculations are not known to the RINC-2 classifier, the number of intermediate feature maps was set to 128 with the objective of increasing the classification accuracy. All intermediate operations used full-precision values. Figure 5.9 illustrates this network (Network-A). Without using RINC-classifiers, the classification accuracy of this network was measured at 94.5%.

Table 5.7 Accuracy According to the Number of Layers Replaced with RINC-2 (Network-A).

Layers replaced	**Replaced Layer Dimensions**	**Accuracy (%)**	**Variation (%)**
0	-	94.5	-
1	$3 \rightarrow 64$	94.3	- 0.2
2	$64 \rightarrow 64$	92.1	- 2.2
3	$64 \rightarrow 64$	44.3	- 47.8

After replacing all layers with the RINC-2 classifier, the accuracy dropped by over 50%. The degradation seems to come from the last layer which combines 3 subsequent convolutions with 3×3 kernels. The input feature maps for all layers contained 3136 ($7 \times 7 \times 64$) values. Perhaps the 512 input features used by the RINC-2 classifiers did not cover enough information. Also, one advantage of conventional CNNs is their ability to be space invariant. By combining many layers together, the RINC classifiers start losing this property and resemble more an MLP.

A second network is proposed (Network-B) in Figure 5.9. This network does not combine convolution layers using 3×3 filters, but rather, combines 1×1 convolutions with 3×3 convolutions. 1×1 convolutions do not enlarge the receptive field of the convolution kernel. They can be used to expand and downsize feature maps. This idea was exploited in a neural network proposed by Chen et al. [58]. The classification accuracy of Network-B without any RINC classifiers was estimated at 94.6%. The results of the network after removing each individual layer is summarized in Table 5.8.

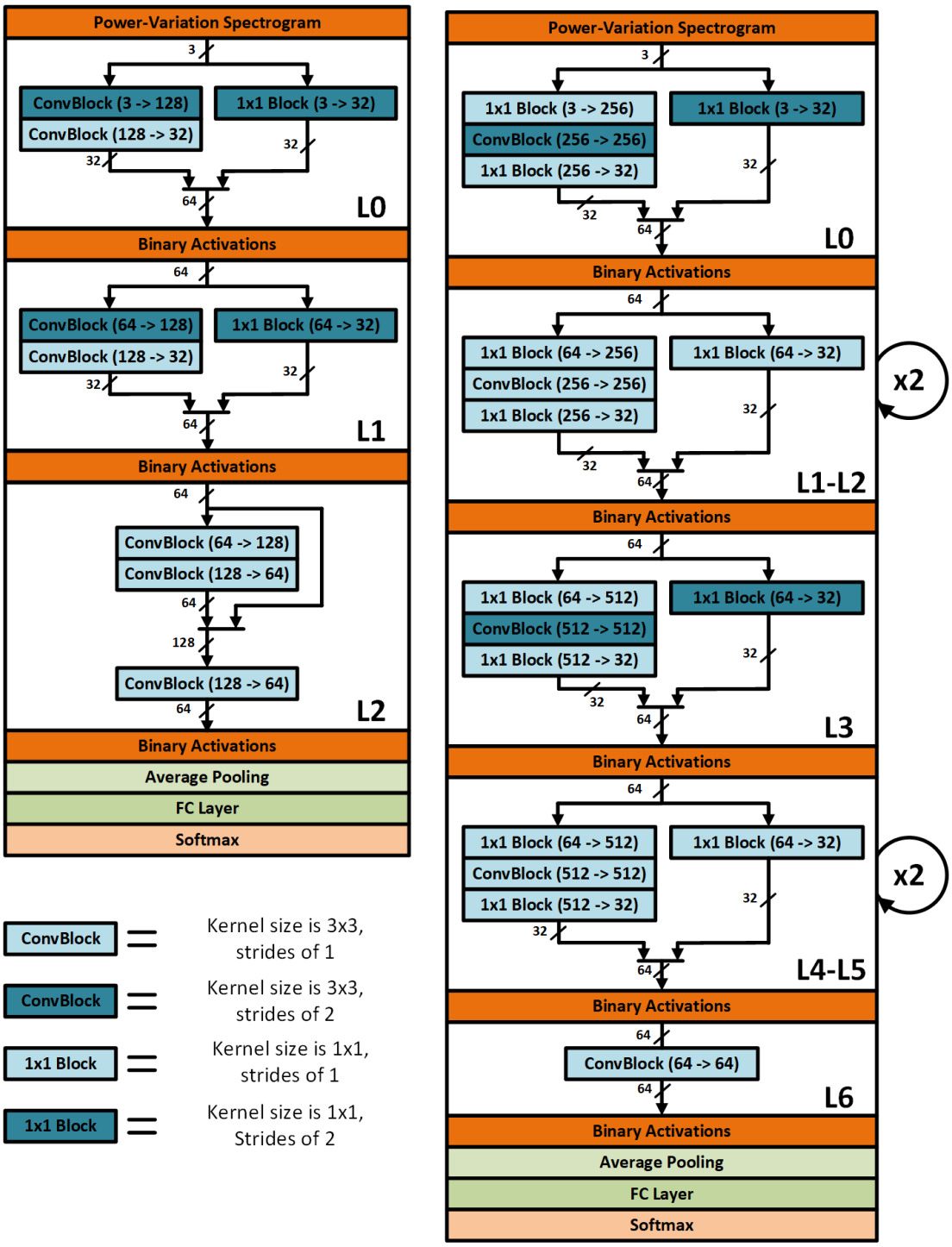

Figure 5.9 Networks that merge convolution blocks using RINC-2 classifiers. Network-A (left) contains 3 layers and was the first attempt at merging continuous layers. Network-B (right) contains 7 layers and uses 1×1 convolutions to expand the size of the intermediate feature maps. Blocks in light blue use strides of 1 and blocks in dark blue use strides of 2. All weights and activations used in intermediate calculations are in floating-point format. A sign function is applied to the outputs of the last ConvBlocks of each layer to obtain binary values.

Table 5.8 Accuracy According to the Number of Layers Replaced with RINC-2 (Network-B).

Layers replaced	Replaced Layer Dimensions	Accuracy (%)	Variation (%)
0	-	94.6	-
1	$3 \rightarrow 64$	94.4	- 0.2
2	$64 \rightarrow 64$	93.9	- 0.5
3	$64 \rightarrow 64$	93.7	- 0.2
4	$64 \rightarrow 64$	93.7	- 0.0
5	$64 \rightarrow 64$	93.2	- 0.5
6	$64 \rightarrow 64$	85.1	- 8.1
7	$64 \rightarrow 64$	84.4	- 0.7

The Network-B architecture offered a final classification accuracy of 84.4%, which is considerably better than Network-A, but not as good as the 12-layer ResNet. The results for all the networks tested with PoET-BiN are summarized in table 5.9. The large amount of LUTs for Network-A and B are due to the large quantity of output channels for each layer.

Table 5.9 Accuracy of Various Networks Before and After Using RINC Classifiers.

Network	Original Accuracy	Final Accuracy	LUT count (8-input)
ResNet (RINC-1)	93.6%	76.9%	1,989
ResNet (RINC-2)	93.6%	89.7%	16,133
Network-A	94.5%	44.3%	14,016
Network-B	94.6%	84.4%	32,704

Extending PoET-BiN to CNNs is still a work in progress. This section was meant to offer some insights and preliminary results on this application.

CHAPTER 6 GENERAL DISCUSSION

Chapter 4 presented a new way to represent speech more adapted to a BNN for KWS. Chapter 5 extended this idea further by complementing this representation. A BNN architecture was also proposed for KWS. Finally, PoET-BiN, an algorithm used for mapping binary operations to LUTs was tested on KWS with the hopes of obtaining a hardware-friendly algorithm. This section will discuss the idea of binarization in the context of KWS, in light of the presented results.

6.1 Using the ResNet Architecture

As seen in Chapter 3, many architectures are capable of performing keyword spotting. We attempted to binarize multiple architectures such as Temporal Convolutions [26] and DS-CNN (MobileNet) [24] since these networks offered the best compromise between classification accuracy and number of computations. However, preliminary results showed that binarization was very destructive to these networks. When tested on the Google Speech Commands, the classification accuracy of the binarized versions of these networks dropped considerably. In many cases, the training did not converge despite using very small learning rates. ResNet was the only one to retain reasonable accuracy (over 94%).

6.2 Choosing the Best Input Representation

In Chapter 4, a 2-channel power-variation spectrogram was proposed to produce binarized input features. In the following section, static information was added to this, producing a 3-channel spectrogram. These features were meant to be used with a BNN. The original BNN implementation [33] used input representations quantized to 8 bits. Binary convolutions were performed over each bit treating them as channels while attributing higher importance to the MSBs (the result of the convolution was shifted left according to its bit position). When applied to keyword spotting, this method offered worst results compared to the proposed 3-channel power-variation spectrogram. This may be due to the fact that a single convolution with binary weights is incapable of capturing full-precision features as well as representations that are inherently in a binary format. Overall, our hypothesis is confirmed; the 3-channel power-variation spectrogram is better for the BNN architectures tested on keyword spotting tasks.

Rahman et al. proposed a similar idea (binarizing speech spectrograms for keyword spotting)

[27]. Instead of measuring the variation of power over time and thresholding power levels at the 50% mark, they used Otsu's threshold. Their work had very few training examples (100) and their objective was not clear. However, since their classification accuracy was relatively good, we decided to test their representation with a BNN. To do so, we replaced the static information (one of the channels) of our 3-channel representation with a 1-bit log-Mel spectrogram thresholded using Otsu's method. We discovered this work after composing most of this thesis, so we did not conduct extensive tests. We simply trained our proposed BNN once on the Google Speech Commands using this representation. The classification accuracy did not considerably change (compared to our approach). However, thresholding at 50% power is less computationally demanding compared to applying Otsu's method. The latter requires the calculation of the *Between Class Variance* for every threshold value. This comes down to computing hundreds of averages for every single spectrogram.

The only known work which attempted binarization on speech recognition was produced by Qian et al. [47]. The input representation used was the log-Mel spectrogram (with 36 coefficients) along with the delta and delta-delta coefficients. In their DNN model, they compared using full precision weights to using binary weights in the first layer (all other layers had binary weights). The use of binary weights slightly increased the WER (about 1% on all datasets). For instance, the WER went from 20.8% to 21.8% on the SwitchBoard telephone speech recognition task. They argued that binarization of the weights on the first layer was only useful if the input representation was quantized. However, they feared that quantizing the input representation would severely affect the classification accuracy. When the power-variation spectrogram was compared to the log-Mel spectrogram in Chapter 5, binary weights were used for the first layer since we wanted a fully binarized neural network. The power-variation spectrogram offered better results (94.3% compared to 92.6%).

Another advantage to using the power-variation spectrogram is that some hardware implementations require binary values for efficient inference. Notably, PolyCiNN, a decision tree-based inference engine required binary inputs in an image classification task [59]. In this work, Abdelsalam et al. computed local binary patterns to generate 4 bit-strings representing input features. Perhaps, this wouldn't be necessary if the input features were already in an efficient binary format.

6.3 Is Binarization Worth It (for KWS)?

KWS can be found in many domotic/IoT devices. The calculations often have to be real-time and generally low-power. For this reason, it is quite common to see this task implemented on a microcontroller. In Chapter 3, a paper by Zhang et al. was introduced which implemented

a KWS system using the Google Speech Commands on an ARM-based microcontroller [24]. They produced a series of networks inspired by Mobile-Net that used weights and activations quantized to 8 bits. ARM Cortex-M4 and Cortex-M7 microcontrollers are equipped with DSP operations making 8-bit fixed-point operations very efficient. Yu et al. attempted to evaluate the use of sub-byte quantization of neural networks on MCUs [60]. They claimed that binary neural networks degrade the accuracy of the network to the point where they are not worth pursuing. Efficient microcontroller implementations, such as the ones produced by Zhang et al. exploit the already available resources and offer reasonable accuracy at a low-power cost. When comparing the proposed binary neural network with the DS-CNN proposed by Zhang et al., the BNN loses on all metrics. Table 6.1 compares these networks. The memory requirements listed include the memory required for weights and activations during inference. The proposed BNN requires over 700 M binary operations. On a microcontroller, having binary operations (compared to 8-bit operations) is not a huge benefit (other than for saving memory). The ARMv7m ISA doesn't have XNOR or popcount instructions, making these operations very slow. Also, increasing the BNN's size (feature maps and layers) does very little to improve its accuracy.

Table 6.1 Proposed Binary Neural Network vs DS-CNN Networks.

Network	Precision	Accuracy (%)	Multiply-adds	Memory Req. (kB)
DS-CNN (S)	8-bit	94.4	5.4 M	38.6
DS-CNN (M)	8-bit	94.9	19.8 M	189.2
DS-CNN (L)	8-bit	95.4	56.9 M	497.6
Proposed BNN	1-bit	94.3	741 M	71.4

Cypress microcontrollers were also considered since their PSoC devices are very low power and contain UDBs (Universal Digital Blocks). These blocks are essentially programmable logic, much like in an FPGA. They also contain many small Arithmetic Logic Units (ALUs) capable of performing 8-bit operations. The PSoC6 products, for example, contain enough logic to implement a 32-bit popcount circuit. This solution was explored. However, slow data-transfers between the UDBs and the processor created an unacceptable bottleneck.

The microcontroller does not seem to be the ideal candidate for BNNs used on KWS tasks. Programmable devices, such as FPGAs, may be a better alternative.

6.4 FPGA Implementations : PoET-BiN for CNNs

In Chapter 5, PoET-BiN was applied for the first time on CNNs for KWS. The objective was to explore hardware solutions and obtain an efficient implementation for KWS. Unfortunately,

applying PoET-BiN to CNNs degraded the classification accuracy of an already (partially) binarized neural network by over 3% in the most optimistic scenario. In contrast, the original PoET-BiN implementation applied to the classifier layers of CNN models produced slightly worst results compared to the teacher network on the MNIST and SVHN datasets (about 1% less accuracy) and improved the accuracy on the CIFAR-10 dataset by over 3% [43].

The 8-input LUT count of the 12 layer network with RINC-2 classifiers was evaluated at 16,133. Xilinx FPGAs make use of 6-input LUTs. To construct an 8-input LUT, at least 4x 6-input LUTs are required. Therefore, this design would need approximately 64,532 logic cells for the convolution kernels alone. This could fit in a higher end Artix-7 or Spartan-7 FPGA which contain over 100k logic cells with the caveat that the hardware required for the sliding windows and storage of intermediate results respects the limit. One disadvantage of this approach is the fact that each convolution layer requires its own set of LUTs. An implementation which would store the binary weights in block RAM and perform XNOR/popcount could be reused for each convolution layer. It would be hard to compare this implementation to any other design since the resulting accuracy is too low to justify an actual implementation.

In conclusion, mapping CNNs to LUTs is still a work in progress. This was attempted on a keyword spotting dataset which is not the typical way to test out CNNs. New CNN-based algorithms are usually tested on image recognition tasks such as MNIST, CIFAR-10, SVHN or ImageNet. Perhaps, applying PoET-BiN on a LeNet-5 network, which only contains 2 convolution layers would produce interesting results.

CHAPTER 7 CONCLUSION

This work was an attempt at applying BNNs to Keyword Spotting (KWS) with the objective of obtaining a low-power system with reasonable accuracy. KWS was defined as a sub-task of speech recognition which aims to detect specific keywords in a continous stream of audio. To do this, a new speech representation was proposed which is compatible with BNNs. A BNN architecture capable of performing KWS was also introduced. Finally, PoET-BiN, an algorithm used to learn the layers of a teacher neural network using stacked layers of decision trees was tested for the first time on CNNs.

Most speech recognition systems use log-Mel spectrograms or MFCCs as input features. These spectrograms are essentially images representing power over time and frequency axes. Since these representations use full-precision values to represent power, they may not be the best fit for BNNs which contain binary weights. To overcome this, a speech representation which used binary values exclusively was proposed. This representation computes the variation of power between two subsequent time frames in the log-Mel spectrogram for each individual frequency bin. If the variation of power is superior to a threshold, a 1 is set in the first channel. If the variation of power is inferior to a second threshold, a 1 is set in the second channel. Using 2 channels, the resulting representation contained the dynamic information of the log-Mel spectrogram, i.e. how the power varies over time. When tested on a state-of-the-art ResNet using the Google Speech Commands (a dataset with the objective of detecting one of ten keywords), the proposed representation offered a classification accuracy that was only 1.2% inferior to the log-Mel spectrogram. To reduce this gap, a third channel with binary information was added to take into account the power level. With this change, the gap between the log-Mel spectrogram and the power-variation spectrogram was reduced to 0.5%. When tested on a fully binarized neural network, the power-variation spectrogram offered 1.7% higher classification accuracy compared to the log-Mel spectrogram. Overall, the representation proposed offered better results on a BNN but the log-Mel spectrogram offered better results on a network with full-precision weights and activations.

Despite having an efficient binary representation of speech, BNNs do not produce optimal results on microcontrollers for KWS. This was explained by the fact that this type of system is optimized for operations on bytes or larger data types. Perhaps, future ISA extensions can include binary operations that may reduce the difficulty of working with sub-byte arrays. FPGAs, on the other hand, are good candidates for BNNs since they can perform all sorts of logical operations. In this work we proposed to use the PoET-BiN heuristic, which maps

neural network layers to FPGA LUTs [43]. When applied to the fully-connected section of CNN-based models, this method reduced the power consumption (of those fully-connected layers) by many-folds [43]. The original paper [43] did not test this algorithm on the CNN portion of the network. Extending this method to convolution layers could mean huge power savings. However, after applying PoET-BiN to a 12-layer ResNet on the Google Speech Commands, we did not obtain competitive results. In the best case scenario, the classification error was over 10%.

7.1 Limitations

The works produced come with several limitations. Notably, the binary representation was tested using 40 frequency bins (40 features per time frame). However, this is not set in stone. Systems that require higher precision tend to use more whereas rudimentary systems make use of fewer. Perhaps, with a different quantity, the representation would lose its competitive edge. A second consideration is the size of the BNN proposed. We were not able to achieve a compromise between size and accuracy. Even if we were to double the quantity of feature maps of the proposed ResNet, the resulting accuracy would barely match an efficient full-precision network like MobileNet which requires 10× fewer multiply-add operations. ARM microcontrollers, for example, can easily perform 8-bit fixed-point operations in parallel using DSP instructions. Binarization is not as efficient since popcount instructions don't exist. Quantizing to higher bit-lengths (8 bits) offers better results on all metrics (speed, accuracy, memory requirements).

PoET-BiN applied to CNNs comes with a considerable degradation to the classification accuracy. The idea of combining convolution layers also partially defeats the purpose of a convolution operation since the filters are no longer applied to a small fraction of the image, but rather to a large segment of it. Considering that each layer required its own set of LUTs, the overall LUT count was very large. A system that reused the same hardware for each layer but that stored weights in BRAM could make the resource count smaller and offer similar results.

7.2 Future Research

Future research should seek to extend the binary representation to larger vocabulary tasks of speech recognition. LVCSR tasks are more computationally demanding than KWS tasks and can gain from using BNNs. Qian et al. also demonstrated that binarization can work for networks applied to this task [47]. PoET-BiN applied to CNNs should also be tested on

more traditional image recognition datasets such as MNIST, CIFAR-10 or SVHN to offer a comparison with the original PoET-BiN paper [43]. Perhaps, a LeNet-5 network, which only contains 2 convolution layers, would produce better results compared to a ResNet with 12 convolution layers. Chidabaram et al. recommended looking into differentiable decision trees. This would enable the entire network to be trained on its own without relying on a teacher network. This could help reduce the gradual degradation seen when replacing convolution layers one at a time. Finally, it would be interesting to implement the proposed binary network using the FINN framework [45]. This could offer metrics to compare the BNN with a KWS implementation on another platform.

REFERENCES

[1] V. Kepuska and G. Bohouta, "Next-Generation of Virtual Personal Assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2018, pp. 99–103.

[2] G. Hinton *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] T. N. Sainath *et al.*, "Deep Convolutional Neural Networks for LVCSR," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8614–8618.

[4] 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)*. IEEE, 2014, pp. 4087–4091.

[5] T. N. Sainath and C. Parada, "Convolutional Neural Networks for Small-Footprint Keyword Spotting," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[6] M. Ostendorf and S. Roukos, "A Stochastic Segment Model for Phoneme-based Continuous Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 1857–1869, 1989.

[7] 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.

[8] A. R. Mohamed, "Deep Neural Network Acoustic Models for ASR," Ph.D. dissertation, University of Toronto, 2014.

[9] S. Furui, "Speaker-Independent Isolated Word Recognition using Dynamic Features of Speech Spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 1, pp. 52–59, 1986.

[10] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] W. Chan *et al.*, "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[13] D. C. de Andrade *et al.*, "A Neural Attention Model for Speech Command Recognition," *arXiv preprint arXiv:1808.08929*, 2018.

[14] M. Zeng and N. Xiao, "Effective Combination of DenseNet and BiLSTM for Keyword Spotting," *IEEE Access*, vol. 7, pp. 10 767–10 775, 2019.

[15] V. Panayotov *et al.*, "Librispeech: An ASR Corpus Based on Public Domain Audio Books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[16] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[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. He *et al.*, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[19] 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.

[20] 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.

[21] S. Fernández, A. Graves, and J. Schmidhuber, "An Application of Recurrent Neural Networks to Discriminative Keyword Spotting," in *International Conference on Artificial Neural Networks*. Springer, 2007, pp. 220–229.

[22] G. Huang *et al.*, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[23] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient Neural Network Kernels for ARM Cortex-M CPUs," *arXiv preprint arXiv:1801.06601*, 2018.

[24] Y. Zhang *et al.*, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[25] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

[26] S. Choi *et al.*, "Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices," *Proc. Interspeech 2019*, pp. 3372–3376, 2019.

[27] M. Rahman *et al.*, "Binary Features of Speech Signal for Recognition," *International Journal of Applied Research on Information Technology and Computing*, vol. 6, no. 1, pp. 18–25, 2015.

[28] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[29] S. Han *et al.*, "Learning Both Weights and Connections for Efficient Neural Network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[31] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] M. Courbariaux, Y. Bengio, and J.-P. David, "Training Deep Neural Networks with Low Precision Multiplications," *arXiv preprint arXiv:1412.7024*, 2014.

[33] I. Hubara *et al.*, "Binarized Neural Networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[34] 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.

[35] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *arXiv preprint arXiv:1308.3432*, 2013.

[36] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv preprint arXiv:1502.03167*, 2015.

[37] M. Rastegari *et al.*, "XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[38] T. Simons and D.-J. Lee, "A Review of Binarized Neural Networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

[39] Z. Liu *et al.*, "Bi-Real Net: Enhancing the Performance of 1-bit CNNs with Improved Representational Capability and Advanced Training Algorithm," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 722–737.

[40] W. Tang, G. Hua, and L. Wang, "How to Train a Compact Binary Neural Network with High Accuracy?" in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[41] S. Darabi *et al.*, "BNN+: Improved Binary Network Training," *arXiv preprint arXiv:1812.11800*, 2018.

[42] A. M. Abdelsalam *et al.*, "POLYBiNN: A Scalable and Efficient Combinatorial Inference Engine for Neural Networks on FPGA," in *2018 Conference on Design and Architectures for Signal and Image Processing (DASIP)*. IEEE, 2018, pp. 19–24.

[43] S. Chidambaram, P. Langlois, and J.-P. David, "PoET-BiN: Power Efficient Tiny Binary Neurons," in *Proceedings of Machine Learning and Systems 2020*, 2020, pp. 160–171.

[44] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[45] Y. Umuroglu *et al.*, "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 65–74.

[46] X. Xiang, Y. Qian, and K. Yu, "Binary Deep Neural Networks for Speech Recognition," in *Proc. Interspeech 2017*, 2017, pp. 533–537. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2017-1343

[47] Y.-m. Qian and X. Xiang, "Binary Neural Networks for Speech Recognition," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 5, pp. 701–715, 2019.

[48] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, 2015.

[49] A. Riviello and J.-P. David, "Binary Speech Features for Keyword Spotting Tasks," *Proc. Interspeech 2019*, pp. 3460–3464, 2019.

[50] G. Hinton *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[51] S. Srinivas and R. V. Babu, "Data-Free Parameter Pruning for Deep Neural Networks," *arXiv preprint arXiv:1507.06149*, 2015.

[52] B. McFee *et al.*, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[53] 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.

[54] A. Bulat *et al.*, "Improved Training of Binary Networks for Human Pose Estimation and Image Recognition," *arXiv preprint arXiv:1904.05868*, 2019.

[55] P. Guo *et al.*, "FBNA: A Fully Binarized Neural Network Accelerator," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 51–513.

[56] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[57] C. Zhang, S. Bengio, and Y. Singer, "Are All Layers Created Equal?" *arXiv preprint arXiv:1902.01996*, 2019.

[58] X. Chen *et al.*, "Small-Footprint Keyword Spotting with Graph Convolutional Network," *arXiv preprint arXiv:1912.05124*, 2019.

[59] A. M. Abdelsalam *et al.*, "POLYCiNN: Multiclass Binary Inference Engine using Convolutional Decision Forests," in *2019 Conference on Design and Architectures for Signal and Image Processing (DASIP)*. IEEE, 2019, pp. 13–18.

[60] J. Yu *et al.*, "TF-Net: Deploying Sub-Byte Deep Neural Networks on Microcontrollers," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–21, 2019.