Hello Edge: Keyword Spotting on Microcontrollers

Yundong Zhang[1,2], Naveen Suda[1], Liangzhen Lai[1] and Vikas Chandra[1]

[1]Arm, San Jose, CA
[2]Stanford University, Stanford, CA

Abstract

Keyword spotting (KWS) is a critical component for enabling speech based user interactions on smart devices. It requires real-time response and high accuracy for good user experience. Recently, neural networks have become an attractive choice for KWS architecture because of their superior accuracy compared to traditional speech processing algorithms. Due to its always-on nature, KWS application has highly constrained power budget and typically runs on tiny microcontrollers with limited memory and compute capability. The design of neural network architecture for KWS must consider these constraints. In this work, we perform neural network architecture evaluation and exploration for running KWS on resource-constrained microcontrollers. We train various neural network architectures for keyword spotting published in literature to compare their accuracy and memory/compute requirements. We show that it is possible to optimize these neural network architectures to fit within the memory and compute constraints of microcontrollers without sacrificing accuracy. We further explore the depthwise separable convolutional neural network (DS-CNN) and compare it against other neural network architectures. DS-CNN achieves an accuracy of 95.4%, which is ~10% higher than the DNN model with similar number of parameters.

1.关键词检测，设备语音应用
2 用户体验需要实时响应和高精度
3.NN相比传统的吸引力
4.资源受限
5本文就是探索资源受限的KWS的NN
6.DS-CNN的精度

1 Introduction

1.深度学习在很多识别任务中都超过了人的精度，图像分类，语音识别等

Deep learning algorithms have evolved to a stage where they have surpassed human accuracies in a variety of cognitive tasks including image classification [1] and conversational speech recognition [2]. Motivated by the recent breakthroughs in deep learning based speech recognition technologies, speech is increasingly becoming a more natural way to interact with consumer electronic devices, for example, Amazon Echo, Google Home and smart phones. However, always-on speech recognition is not energy-efficient and may also cause network congestion to transmit continuous audio stream from billions of these devices to the cloud. Furthermore, such a cloud based solution adds latency to the application, which hurts user experience. There are also privacy concerns when audio is continuously transmitted to the cloud. To mitigate these concerns, the devices first detect predefined keyword(s) such as "Alexa", "Ok Google", "Hey Siri", etc., which is commonly known as keyword spotting (KWS). Detection of keyword wakes up the device and then activates the full scale speech recognition either on device [3] or in the cloud. In some applications, the sequence of keywords can be used as voice commands to a smart device such as a voice-enabled light bulb. Since KWS system is always-on, it should have very low power consumption to maximize battery life. On the other hand, the KWS system should detect the keywords with high accuracy and low latency, for best user experience. These conflicting system requirements make KWS an active area of research ever since its inception over 50 years ago [4]. Recently, with the renaissance of artificial neural networks in the form of deep learning algorithms, neural network (NN) based KWS has become very popular [5, 6, 7, 8].

2深度学习的语音识别技术成为一个很自然的人机交互方法
3.但是语音识别并不能耗有效，数以百万计的设备上传audio数据可能会带来延迟
4.基于云的应用带来了延迟。
5.声音不断传到云中，也涉及到隐私问题
6 KWS是为了解决以上问题的
6 关键词识别唤醒设备和激活全功能设备或者云端的语音识别
7 语音序列作为声音控制命令
8 KWS应该低功耗和KWS应该高精度和低延迟的矛盾关系

[2]Work was done while the author was an intern at Arm.

9矛盾的系统需求使得KWS几十年以来都是一个活跃的研究领域，特别是AI的复兴，基于深度学习的KWS开始受到欢迎

Low power consumption requirement for keyword spotting systems make microcontrollers an obvious choice for deploying KWS in an always-on system. Microcontrollers are low-cost energy-efficient processors that are ubiquitous in our everyday life with their presence in a variety of devices ranging from home appliances, automobiles and consumer electronics to wearables. However, deployment of neural network based KWS on microcontrollers comes with following challenges:

Limited memory footprint: Typical microcontroller systems have only tens to few hundred KB of memory available. The entire neural network model, including input/output, weights and activations, has to fit within this small memory budget.

Limited compute resources: Since KWS is always-on, the real-time requirement limits the total number of operations per neural network inference.

These microcontroller resource constraints in conjunction with the high accuracy and low latency requirements of KWS call for a resource-constrained neural network architecture exploration to find *lean* neural network structures suitable for KWS, which is the primary focus of our work. The main contributions of this work are as follows:

- We first train the popular KWS neural net models from the literature [5, 6, 7, 8] on Google speech commands dataset [9] and compare them in terms of accuracy, memory footprint and number of operations per inference.

1.第一个训练KWS nn模型的，基于谷歌语音数据集；

- In addition, we implement a new KWS model using depth-wise separable convolutions and point-wise convolutions, inspired by the success of resource-efficient MobileNet [10] in computer vision. This model outperforms the other prior models in all aspects of accuracy, model size and number of operations.

2.实现一种新型的KWS模型

- Finally, we perform resource-constrained neural network architecture exploration and present comprehensive comparison of different network architectures within a set of compute and memory constraints of typical microcontrollers. The code, model definitions and pretrained models are available at https://github.com/ARM-software/ML-KWS-for-MCU.

3.资源首先的NN结构探索。代码，模型定义和预先训练的模型。

2 Background

2.1 Keyword Spotting (KWS) System

A typical KWS system consists of a feature extractor and a neural network based classifier as shown in Fig. 1. First, the input speech signal of length L is framed into overlapping frames of length l with a stride s, giving a total of $T = \frac{L-l}{s} + 1$ frames. From each frame, F speech features are extracted, generating a total of $T \times F$ features for the entire input speech signal of length L. Log-mel filter bank energies (LFBE) and Mel-frequency cepstral coefficients (MFCC) are the commonly used human-engineered speech features in deep learning based speech-recognition, that are adapted from traditional speech processing techniques. Feature extraction using LFBE or MFCC involves translating the time-domain speech signal into a set of frequency-domain spectral coefficients, which enables dimensionality compression of the input signal. The extracted speech feature matrix is fed into a classifier module, which generates the probabilities for the output classes. In a real-world scenario where keywords need to be identified from a continuous audio stream, a posterior handling module averages the output probabilities of each output class over a period of time, improving the overall confidence of the prediction.

l,重叠的帧长度
s,条幅stride的宽度
L，输入的语音信号长度
T，计算总帧数；
每一帧计算F语音特征

特征提取 NN用于分类

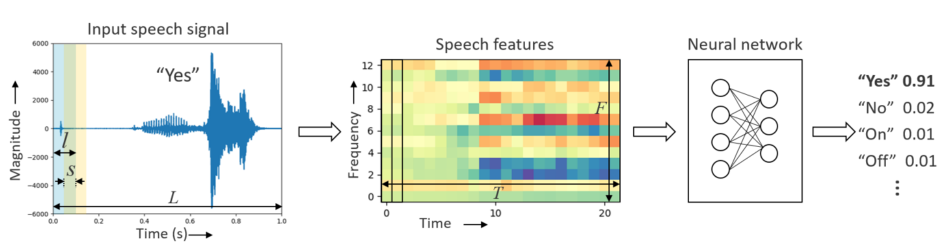


Figure 1: Keyword spotting pipeline.

1.典型的KWS语音识别技术采用的隐马尔科夫模型和viterbi解码

2.这些方法可以达到比较合理的精度，但是难于训练和在推理的时候运算成本很高

3.discriminative model判别模型，large margin problem formulation，RNN

Traditional speech recognition technologies for KWS use Hidden Markov Models (HMMs) and Viterbi decoding [11, 12]. While these approaches achieve reasonable accuracies, they are hard to train and are computationally expensive during inference. Other techniques explored for KWS include discriminative models adopting a large-margin problem formulation [13] or recurrent neural networks (RNN) [14]. Although these methods significantly outperform HMM based KWS in terms of accuracy, they suffer from large detection latency. KWS models using deep neural networks (DNN) based on fully-connected layers with rectified linear unit (ReLU) activation functions are introduced in [5], which outperforms the HMM models with a very small detection latency. Furthermore, low-rank approximation techniques are used to compress the DNN model weights achieving similar accuracy with less hardware resources [15, 16]. The main drawback of DNNs is that they ignore the local temporal and spectral correlation in the input speech features. In order to exploit these correlations, different variants of convolutional neural network (CNN) based KWS are explored in [6], which demonstrate higher accuracy than DNNs. The drawback of CNNs in modeling time varying signals (e.g. speech) is that they ignore long term temporal dependencies. Combining the strengths of CNNs and RNNs, convolutional recurrent neural network based KWS is investigated in [7] and demonstrate the robustness of the model to noise. While all the prior KWS neural networks are trained with cross entropy loss function, a max-pooling based loss function for training KWS model with long short-term memory (LSTM) is proposed in [8], which achieves better accuracy than the DNNs and LSTMs trained with cross entropy loss.

4.这些方法明显比HMM方法精度高，但是有比较大的检测延迟。

5.使用DNN的KWS模型比HMM效果好，延时还小

6，低级近似技术被用来压缩DNN权重，达到类似的精度，硬件资源也少

7.DNN的缺点，忽略输入语音特征的本地时间和频率相关性。

8，为了探索这个相关性各种CNN被研究

9.CNN的确定是忽略长时依赖

10.结合CNN和RNN的优势，CRNN的KWS被研究，对噪音的鲁棒性很好

10.KWS NN使用cross entropy loss function来训练，LSTM的KWS被提出，比DNN和LSTM的精度更高。

Although many neural network models for KWS are presented in literature, it is difficult to make a fair comparison between them as they are all trained and evaluated on different proprietary datasets (e.g. "TalkType" dataset in [7], "Alexa" dataset in [8], etc.) with different input speech features and audio duration. Also, the primary focus of prior research has been to maximize the accuracy with a small memory footprint model, without explicit constraints of underlying hardware, such as limits on number of operations per inference. In contrast, this work is more hardware-centric and targeted towards neural network architectures that maximize accuracy on microcontroller devices. The constraints on memory and compute significantly limit the neural network parameters and the number of operations. 14 内存和计算的限制，限制了NN参数和运行数量

11.KWS的不同NN模型，难以比较，因为他们使用不同的数据集。

12之前的工作主要聚焦于在小的内存上最大化精度。

13.相比之下，这个工作更偏硬件，目标是针对MCU设备的精度最大化。

2.2 Microcontroller Systems

1.典型的MCU系统包括处理器内核，片上SRAM块，片上嵌入式flash

A typical microcontroller system consists of a processor core, an on-chip SRAM block and an on-chip embedded flash. Table 1 shows some commercially available microcontroller development boards with Arm Cortex-M processor cores with different compute capabilities running at different frequencies (16 MHz to 216 MHz), consisting of a wide range of on-chip memory (SRAM: 8 KB to 320 KB; Flash: 128 KB to 1 MB). The program binary, usually preloaded into the non-volatile flash, is loaded into the SRAM at startup and the processor runs the program with the SRAM as the main data memory. Therefore, the size of the SRAM limits the size of memory that the software can use.

2.表1展示一些商用的mcu开发板，Cortex-M处理器核，不同计算能力，运行不同的频率，不同的片上内存。

3二进制代码预装入flash，在启动阶段导入SRAM，处理器运行SRAM。

Other than the memory footprint, performance (i.e., operations per second) is also a constraining factor for running neural networks on microcontrollers. Most microcontrollers are designed for embedded applications with low cost and high energy-efficiency as the primary targets, and do not have high throughput for compute-intensive workloads such as neural networks. Some microcontrollers have integrated DSP instructions that can be useful for running neural network workloads. For example, Cortex-M4 and Cortex-M7 have integrated SIMD and MAC instructions that can be used to accelerate low-precision computation in neural networks.

4，性能也是一个现在因素

4.MCU设计用于嵌入式应用，具有低成本和高能耗，没有很高的计算能力，如NN

5.单片机具有集成的DSP指令，可以用于运行NN

SIMD，single instruction multiple data，同时取得数据，一次执行
MAC，乘法和加法 multiply accumulate

Arm Mbed™ platform	Processor	Frequency	SRAM	Flash
Mbed LPC11U24	Cortex-M0	48 MHz	8 KB	32 KB
Nordic nRF51-DK	Cortex-M0	16 MHz	32 KB	256 KB
Mbed LPC1768	Cortex-M3	96 MHz	32 KB	512 KB
Nucleo F103RB	Cortex-M3	72 MHz	20 KB	128 KB
Nucleo L476RG	Cortex-M4	80 MHz	128 KB	1 MB
Nucleo F411RE	Cortex-M4	100 MHz	128 KB	512 KB
FRDM-K64F	Cortex-M4	120 MHz	256 KB	1 MB
Nucleo F746ZG	Cortex M7	216 MHz	320 KB	1 MB

Table 1: Typical off the shelf Arm Cortex-M based microcontroller development platforms.

现成的

KWS的NN神经网络架构

3 Neural Network Architectures for KWS

This section gives an overview of all the different neural network architectures explored in this work including the deep neural network (DNN), convolutional neural network (CNN), recurrent neural network (RNN), convolutional recurrent neural network (CRNN) and depthwise separable convolutional neural network (DS-CNN).

3.1 Deep Neural Network (DNN)

1.DNN是一个前馈NN，包括全连接，非线性激活层。

The DNN is a standard feed-forward neural network made of a stack of fully-connected layers and non-linear activation layers. The input to the DNN is the flattened feature matrix, which feeds into a stack of d hidden fully-connected layers each with n neurons. Typically, each fully-connected layer is followed by a rectified linear unit (ReLU) based activation function. At the output is a linear layer followed by a softmax layer generating the output probabilities of the k keywords, which are used for further posterior handling.

2.DNN的输入是特征矩阵，d个隐藏全连接层，每层都有n个神经元

3.每个全连接层都紧跟一个基于ReLU的激活函数。 4.输出是一个线性层，softmax层，产生k个关键词的可能，用于进一步的后验处理

3.2 Convolutional Neural Network (CNN)

1.基于DNN的KWS的缺点是，不能对语音特征的局部时间和频率相关性进行建模

2.CNN探索这种关系，对待输入时域和频域特征作为一个图像，2D卷积操作

One main drawback of DNN based KWS is that they fail to efficiently model the local temporal and spectral correlation in the speech features. CNNs exploit this correlation by treating the input time-domain and spectral-domain features as an image and performing 2-D convolution operations over it. The convolution layers are typically followed by batch normalization [17], ReLU based activation functions and optional max/average pooling layers, which reduce the dimensionality of the features. During inference, the parameters of batch normalization can be folded into the weights of the convolution layers. In some cases, a linear low-rank layer, which is simply a fully-connected layer without non-linear activation, is added in between the convolution layers and dense layers for the purpose of reducing parameters and accelerating training [18, 19].

3.卷积之后有batch normalization,基于ReLU的激活函数最大和平均池化层，减少特征的维度。

5.线性低级层，是一个没有非线性激活层的全连接层，在卷积层和密度层中增加，用于减少参数和加速训练

4.批归一化的参数在推理中折叠为卷积层的权重

3.3 Recurrent Neural Network (RNN)

2.RNN不仅探索输入信号的时间关系，也抓取长期依赖，使用gate机制

1.RNN在很多序列模型任务中现实很好的性能，特别是语音识别，语言建模，翻译等

RNNs have shown superior performance in many sequence modeling tasks, especially speech recognition [20], language modeling [21], translation [22], etc. RNNs not only exploit the temporal relation between the input signal, but also capture the long-term dependencies using "gating" mechanism. Unlike CNNs where input features are treated as 2-D image, RNNs operate for T time steps, where at each time step t the corresponding spectral feature vector $f_t \in R^F$ concatenated with the previous time step output h_{t-1} is used as input to the RNN. Figure 2 shows the model architecture of a typical RNN model, where the RNN cell could be an LSTM cell [23, 24] or a gated recurrent unit (GRU) cell [25, 26]. Since the weights are reused across all the T time steps, the RNN models tend to have less number of parameters compared to the CNNs. Similar to batch normalization in CNNs, research show that applying layer normalization can be beneficial for training RNNs [27], in which the hidden states are normalized during each time step.

3.不像CNN那样输入特征作为2D图像，RNN作为T步运行。每步都有相应的频率特征向量，和之前的作为RNN的输入。

4.图2是典型的RNN模型，RNN cell应该是LSTM cell。或者GRU的cell。

5.跨越T时间的权重重用，RNN模型往往具有比较少的参数，相比CNNs。

6.相比CNN中的批归一化，研究显示，应用层归一化可以对训练RNN有左右，每一步的隐藏状态归一化

3.4 Convolutional Recurrent Neural Network (CRNN)

1.CNN和RNN的混合模式称为CRNN，充分利用两者的优点。

2.循环层使用语音特征的卷积层和全句时间依赖来探索时间空间关系。

Convolution recurrent neural network [7] is a hybrid of CNN and RNN, which takes advantages of both. It exploits the local temporal/spatial correlation using convolution layers and global temporal dependencies in the speech features using recurrent layers. As shown in Fig. 3, a CRNN model

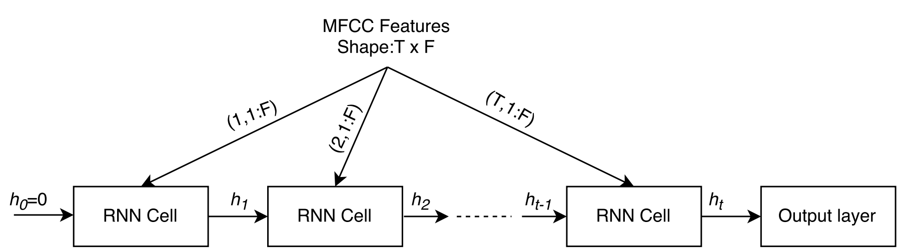


Figure 2: Model architecture of RNN.

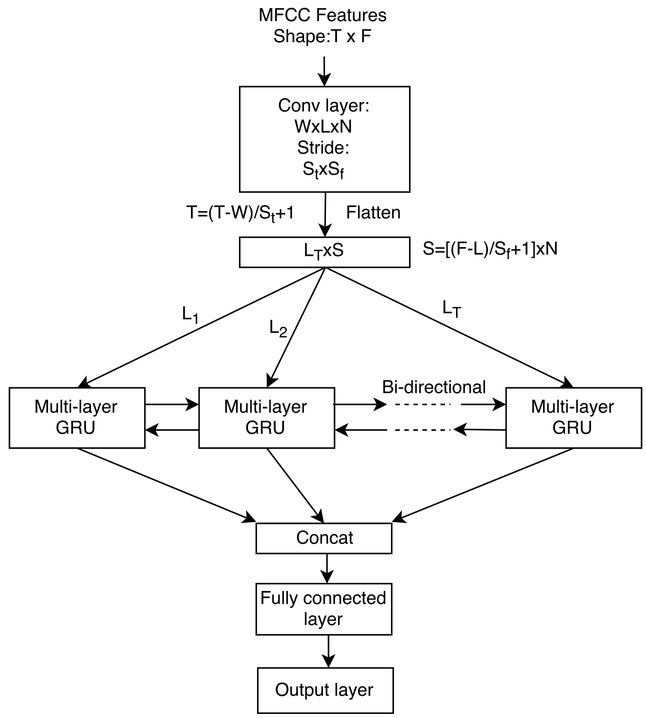


Figure 3: Model Architecture of CRNN.

3，图3，CRNN模型开始于卷积层，RNN对信号和稠密的全连接层进行译码，来映射信息。

starts with a convolution layer, followed by an RNN to encode the signal and a dense fully-connected layer to map the information. Here, the recurrent layer is bi-directional [28] and has multiple stages, increasing the network learning capability. Gated recurrent units (GRU) [25] is used as the base cell for recurrent layers, as it uses fewer parameters than LSTMs and gave better convergence in our experiments.

4.循环层是双向的，具有多个阶段，增加网络学习能力。

5.GRU被用于循环层的base cell,比LSTM具有比较少的参数，更好的聚合能力。

3.5 Depthwise Separable Convolutional Neural Network (DS-CNN)

1.深度分立卷积作为标准3D卷积操作的有效代替，被用于在计算机视觉领域达到压缩的网络结构

Recently, depthwise separable convolution has been proposed as an efficient alternative to the standard 3-D convolution operation [29] and has been used to achieve compact network architectures in the area of computer vision [10, 30]. DS-CNN first convolves each channel in the input feature map with a separate 2-D filter and then uses pointwise convolutions (i.e. $1x1$) to combine the outputs in the depth dimension. By decomposing the standard 3-D convolutions into 2-D convolutions followed by 1-D convolutions, depthwise separable convolutions are more efficient both in number of parameters and operations, which makes deeper and wider architecture possible even in the resource-constrained microcontroller devices. In this work, we adopt a depthwise separable CNN based on the implementation of MobileNet [10] as shown in Fig. 4. An average pooling followed by a fully-connected layer is used at the end to provide global interaction and reduce the total number of parameters in the final layer.

2.DS-CNN在每个输入特征映射图，具有一个2D滤波器，使用逐点卷积，来联合输出。在深度维度中。

3.压缩3-D到2D卷积，通过1D卷积，深度分立卷积能更加有效，在参数数量和运作上。在资源受限的微控制器上

4.采用深度分立CNN实现的MobileNet 5.平均池化和全连接层，被用于提供全局的交互和减少最后层的参数总数。

4 Experiments and Results

1.google语音命令数据集，NN结构探索实验。 2 数据集包括65k 1秒的长时audio片段，30关键词，上千人，每个片段一个关键词

We use the Google speech commands dataset [9] for the neural network architecture exploration experiments. The dataset consists of 65K 1-second long audio clips of 30 keywords, by thousands of different people, with each clip consisting of only one keyword. The neural network models are trained to classify the incoming audio into one of the 10 keywords - "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", along with "silence" (i.e. no word spoken) and "unknown" word, which is the remaining 20 keywords from the dataset. The dataset is split into training, validation and test set in the ratio of 80:10:10 while making sure that the audio clips from the same person stays in the same set. All models are trained in Google Tensorflow framework [31] using

3 NN模型被训练来分类输入的audio信号成10个关键词

4.数据集分为训练，验证，测试，他们的比例是80:10:10

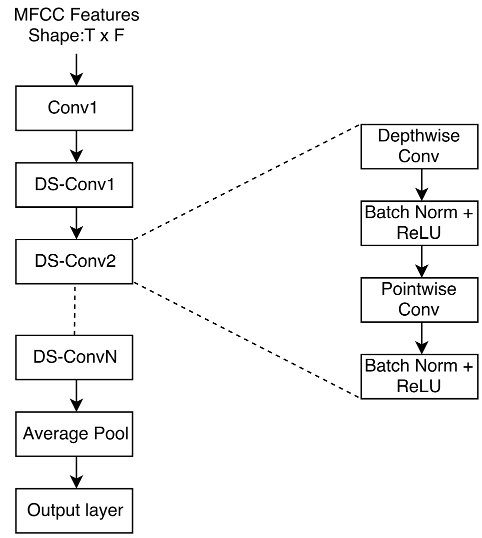


Figure 4: Depthwise separable CNN architecture.

6 batch的大小是100,模型训练为20k迭代　　5.adam源自adaptive moments，学习率自适应的优化算法

the standard cross-entropy loss and Adam optimizer [32]. With a batch size of 100, the models are trained for 20K iterations with initial learning rate of 5×10^{-4}, and reduced to 10^{-4} after first 10K iterations. The training data is augmented with background noise and random time shift of up to 100ms. The trained models are evaluated based on the classification accuracy on the test set.

7 训练数据被背景噪音增强，随机时间达到100ms　　8.在测试数据集上的训练模型被评估用于分类精度

4.1 Training Results

8 从文献中可以看出，精度，内存需求和每个推理的运算次数。

Table 2 summarizes the accuracy, memory requirement and operations per inference for the network architectures for KWS from literature [5, 6, 7, 8] trained on Google speech commands dataset [9]. For all the models, we use 40 MFCC features extracted from a speech frame of length 40ms with a stride of 20ms, which gives 1960 (49×40) features for 1 second of audio. The accuracy shown in the table is the accuracy on test set. The memory shown in the table assumes 8-bit weights and activations, which is sufficient to achieve same accuracy as that from a full-precision network.

4.在表中的内存，假定有8位权重和激活，获取相同精度。

2.对于所有模型，使用了MFCC特征，长度为40ms的语音数据帧

3.表中的精度是测试集的精度。

NN Architecture	Accuracy	Memory	Operations
DNN [5]	84.3%	288 KB	0.57 MOps
CNN-1 [6]	90.7%	556 KB	76.02 MOps
CNN-2 [6]	84.6%	149 KB	1.46 MOps
LSTM [8]	88.8%	26 KB	2.06 MOps
CRNN [7]	87.8%	298 KB	5.85 MOps

Table 2: Neural network model accuracy. CNN-1, CNN-2 are (*cnn-trad-fpool3, cnn-one-fstride4*) architectures from [6].

1假定激活的内存是跨层重用的，因此内存需求使用两个连续层的最大值　　3表中的操作计算乘法和加法，在网络中的矩阵乘法操作中，是整个网络中执行时间的代表。

Also, we assume that the memory for activations is reused across different layers and hence memory requirement for the activations uses the maximum of two consecutive layers. The operations in the table counts the total number of multiplications and additions in the matrix-multiplication operations in each layer in the network, which is representative of the execution time of the entire network. The models from the existing literature are optimized for different datasets and use different memory/compute resources, hence a direct comparison of accuracy is unfair. That said, these results still provide useful insights on the different neural network architectures for KWS:

3.现存文献中的模型，用于不同数据集的优化，使用不同的内存计算资源，因此直接比较精度是不公平的。

4 这些结果仍然提供有用的观点

- Although DNNs do not achieve the best accuracy and tend to be memory intensive, they have less number of operations/inference and hence suit well to systems that have limited compute capability (e.g. systems running at low operating frequencies for energy-efficiency).
- CNNs, on the other hand, achieve higher accuracy than DNNs but at the cost of large number of operations and/or memory requirement.

5，虽然DNN没有达到最好的精度和往往内存敏感，具有比较少的运算推理比较适合计算资源有限的，系统运行在低运行频率下用于能量有效性。

6.另外，CNN具有比DNN更高的精度，但是比较大的运行数量和内存需要

7.LSTM和CRNN在内存和运算之间达到平衡，同样达到好的精度。

- LSTMs and CRNNs achieve a balance between memory and operations while still achieving good accuracy.

4.2 Classifying Neural Networks for KWS based on Resource Requirements

关键词检测的资源需求来分类NN

1.内存消耗和执行时间是KWS在单片机上运行的两个重要考虑因素。

As discussed in section 2.2, memory footprint and execution time are the two important considerations in being able to run keyword spotting on microcontrollers. These should be considered when designing and optimizing neural networks for running keyword spotting. Based on typical microcontroller system configurations (as described in Table 1), we derive three sets of constraints for the neural networks in Table 3, targeting small, medium and large microcontroller systems. Both memory and compute limit are derived with assumptions that some amount of resources will be allocated for running other tasks such as OS, I/O, network communication, etc. The operations per inference limit assumes that the system is running 10 inferences per second.

2.设计和优化NN用于KWS是必须要考虑的。

3.基于典型的微控制器系统配置，我们在表3中延伸出三类神经网络

4.内存和计算现在，还有资源，比如操作，IO，网络通信等任务

5.每个推理的操作或运算限制，假定了系统运行，

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

Table 3: Neural network (NN) classes for KWS models considered in this work, assuming 10 inferences per second and 8-bit weights/activations.

资源受限的NN网络结构探索

4.3 Resource Constrained Neural Network Architecture Exploration

Figure 5 shows the number of operations per inference, memory requirement and test accuracy of neural network models from prior work [5, 6, 7, 8] trained on Google speech commands dataset overlayed with the memory and compute bounding boxes for the neural network classes from section 4.2. An ideal model would have high accuracy, small memory footprint and lower number of computations, i.e., close to the origin in Fig. 5. Apart from the LSTM model, the other models are too memory/compute resource heavy and do not fit into the bounding box *S* with 80KB/6MOps memory/compute limits. CNN-2, CRNN and DNN models fit in the *M* and *L* bounding boxes, but have lower accuracies as compared to the CNN-1 model, which does not fit in any of the boxes at all. The rest of this section discusses different hyperparameters of the feature extraction and neural network architectures that can be tuned in order to bring the models close to the origin and still achieve high accuracy.

3.除了LSTM模型，其他模型都是太需要内存和计算资源的，不是和80KB/6MOps内存和计算限制。

2.理想的模型应该有高的精度，小的内存占用和低计算数。

4.比起CNN-1，CNN-2，CRNN和DNN模型适合M/L边界的盒子。虽然有比较低的精度。

5.本节的讨论不同特征提取的超参数和NN架构，可以调谐，使模型更接近真实，仍然可以得到高精度。

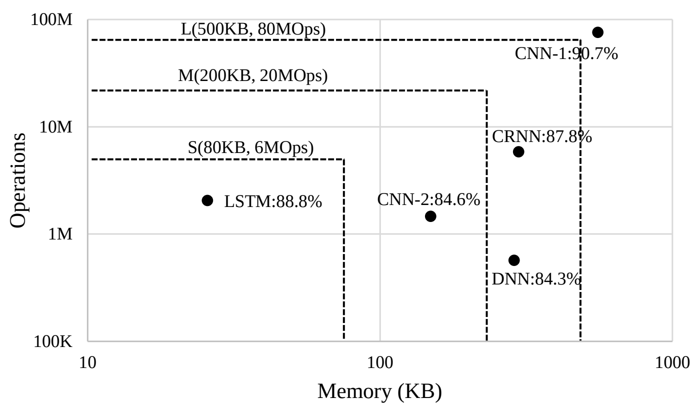


Figure 5: Number of operations vs. memory vs. test accuracy of NN models from prior work [5, 6, 7, 8] trained on the speech commands dataset [9].

1.每个输入信号，TXF的特征被提取出来，特征数量影响到模型大小，运算操作数量和精度。

As shown in Fig. 1, from each input speech signal, $T \times F$ features are extracted and the number of these features impact the model size, number of operations and accuracy. The key parameters in the feature extraction step that impact the model size, number of operations and accuracy are (1) number of MFCC features per frame (F) and (2) the frame stride (S). The number of MFCC features per audio frame (F) impacts the number of weights in fully-connected and recurrent layers,

2.特征提取步骤的关键参数，影响模型大小运算和精度，1）每帧的MFCC特征，2）帧的宽带

3.MFCC的数量影响到全连接和循环层的权重。

5.帧的带宽S，决定了每个推理的帧数，影响到全连接的权重数量。不会影响到循环和卷积层，因为这些权重是需要重用的。

4.卷积层，权重被重用到卷积层。

but not in convolution layers as weights are reused in convolution layers. The frame stride (S), which determines the number of frames to be processed per inference (i.e. T), impacts the number of weights in fully-connected layers but not in recurrent and convolution layers because of the weight reuse. Both F and S impact the number of operations per inference. An efficient model would maximize accuracy using small $T \times F$, i.e., small F and/or large S.

6 F和S影响到每帧的操作数量。

7.一个有效的模型可以使用TXF最大化精度。

1 NN结构和相应的超参数，如表4。

The neural network architectures and the corresponding hyperparameters explored in this work are summarized in Table 4. The LSTM model mentioned in the table includes peephole connections and output projection layer similar to that in [8], whereas basic LSTM model does not include those. CRNN uses one convolution layer followed by multi-layer GRU for the recurrent layers. We also use batch normalization for convolutional/fully-connected layers and layer normalization for recurrent layers. During inference, the parameters of batch normalization and layer normalization can be folded into the weights of the convolution or recurrent layers and hence these layers are ignored in memory/Ops computation.

2.LSTM模型包括窥视孔链接和输出投射层，而基本LSTM不包括这些。

3.CRNN使用一个卷积层和多个GRU

4.使用批归一化来给卷积层和全连接层，循环层的层归一化。

5在推理中，批归一化的参数和层归一化可以被折叠到卷积循环层的权重中，因此这些层可以在内存和运行计算中被忽视

NN model	Model hyperparameters
DNN	Number of fully-connected (FC) layers and size of each FC layer
CNN	Number of Conv layers: features/kernel size/stride, linear layer dim., FC layer size
Basic LSTM	Number of memory cells
LSTM	Number of memory cells, projection layer size
GRU	Number of memory cells
CRNN	Conv features/kernel size/stride, Number of GRU and memory cells, FC layer size
DS-CNN	Number of DS-Conv layers, DS-Conv features/kernel size/stride

Table 4: Neural network hyperparameters used in this study.

1.我们反复地对特征提前超参数进行深入的研究，NN模型超参数和手动选择，来缩小搜索空间。

We iteratively perform exhaustive search of feature extraction hyperparameters and NN model hyperparameters followed by manual selection to narrow down the search space. The final best performing models for each neural network architecture along with their memory requirements and operations are summarized in Table 5 and Fig. 6. The hyperparameters of these networks are summarized in Appendix A. From the results we can see that DNNs are memory-bound and achieve less accuracies and saturate at ~87% even when the model is scaled up. CNNs achieve better accuracies than DNN, but are limited by the weights in the final fully-connected layers. RNN models (i.e. Basic LSTM, LSTM and GRU) achieve better accuracies than CNNs and yield even smaller models with less Ops in some cases, demonstrating that exploiting temporal dependencies maximizes accuracy within the same resource budget. CRNN models, which combine the best properties of CNNs and RNNs, achieve better accuracies than both CNNs and RNNs, even with less Ops. CRNN architecture also scales up well when more memory/compute resources are available. DS-CNN achieves the best accuracies and demonstrate good scalability owing to their deeper architecture enabled by depthwise separable convolution layers, which are less compute/memory intensive.

2.最后最好的性能模型和内存需求总结在表5.6中。

3.这些网络的超参数如附录A

4.从结果中我们看出DNN是内存边界的，达到比较少的精度，当模型搜索的时候，在87%的时候饱和。

5.CNN比DNN的精度高，但是会收全连接层的权值影响

6.RNN模型比CNN的精度更高，模型更小，操作更少，探索时间依赖最大化精度，在同样的资源开销的情况下。

7.CRNN模型，联合CNN和RNN，比RNN和CNN都有更好的精度，更少的操作。

7.DS-CNN达到最好的精度，具有最好伸缩性，因为他们的深度结构，由深度可分离卷积层，比较少的计算和内存敏感。

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

Table 5: Summary of best neural networks from the hyperparameter search. The memory required for storing the 8-bit weights and activations is shown in the table.

1.为了研究小型单片机具有8kb的内存的模型的扩展性，我们扩展了DS-CNN模型的搜索空间

To study the scalability of the models for smaller microcontroller systems with memory as low as 8KB, we expand the search space for DS-CNN models. Figure 7 shows the accuracy, memory/Ops requirements of the DS-CNN models targeted for such constrained devices. It shows that scaled-down DS-CNN models achieve better accuracies than DNN models with similar number of Ops, but with >10x reduction in memory requirement.

2.DS-CNN的精度，内存/操作需求针对某类资源限制的设备。

3.按比例缩小的DS-CNN模型，比DNN模型获得更好的精度，具有类似的操作，内存需求减少了10x

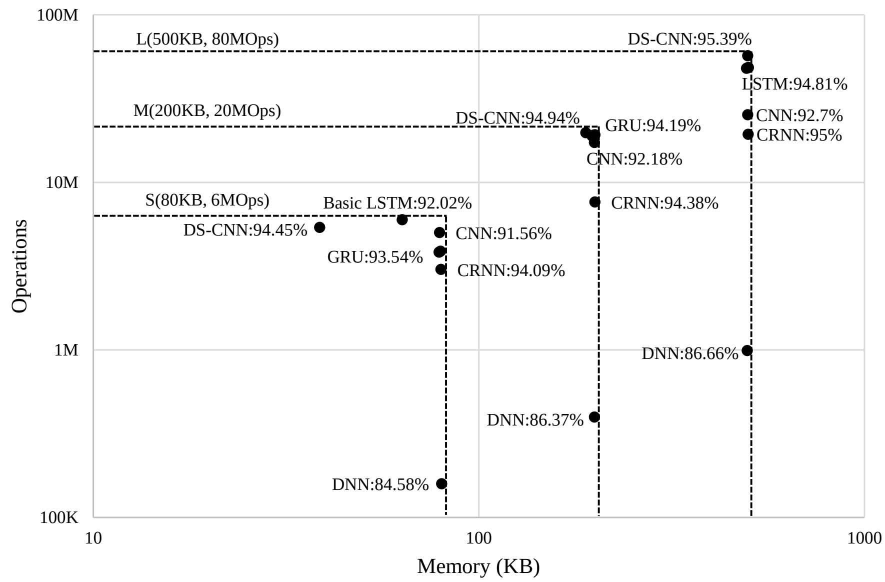


Figure 6: Memory vs. Ops/inference of the best models described in Table 5.

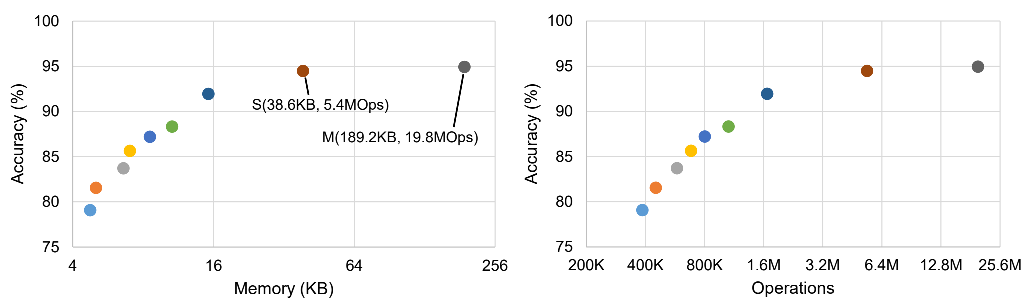


Figure 7: Accuracy vs. memory and Ops of different DS-CNN models demonstrating the scalability of DS-CNN models down to <8KB memory footprint and <500K operations.

7.不同的DS-CNN模型的精度，内存，操作，展示模型在小于8KB内存和小于500k操作

4.4 Neural Network Quantization

1.NN典型地是在浮点权值和激活函数上训练的。

Neural networks are typically trained with floating point weights and activations. Previous research [33, 34, 35] have shown that fixed-point weights is sufficient to run neural networks with minimal loss in accuracy. Microcontroller systems have limited memory, which motivates the quantization of 32-bit floating point weights to 8-bit fixed point weights for deployment, thus reducing the model size by 4×. Moreover, fixed-point integer operations run much faster than floating point operations in typical microcontrollers, which is another reason for executing quantized model during deployment.

2.之前的研究都是浮点权值，可以充分地用于运行NN，精度可以最小的损失。

3.微控制器系统具有有限的内存，使得32位的浮点权值到8位的定点权值，用于部署，从而减少模型尺寸。

4.进一步来说，定点整型运算，比浮点型操作运行得快，这是在部署的时候执行量化模型的一个原因

In this work, we use the quantization flow described in [34] using 8-bits to represent all the weights and activations. For a given signed 2's complement 8-bit fixed-point number, its value (v) can be expressed as $v = -B_7.2^{7-N} + \sum_{i=0}^{6} B_i.2^{i-N}$, where N is the fractional length, which can also be negative. N is fixed for a given layer, but can be different in other layers. For example, $N = 0$ can represent the range $[-128, 127]$ with a step of 1, $N = 7$ can represent the range $[-1, 1-(1/2^7)]$ with a step of $1/2^7$ and $N = -2$ can represent the range $[-512, 508]$ with a step of 2^2.

1.本文中，使用8位来代表权值和激活

2.值的表达，n，

3.N是固定给定层的，但是其他层不同。

The weights are quantized to 8-bits progressively one layer at a time by finding the optimal N for each layer that minimizes the loss in accuracy because of quantization. After all the weights are quantized, the activations are also quantized in a similar way to find the appropriate fractional length N for each layer. Table 6 shows the accuracies of representative 8-bit networks quantized using this method and compared with those of the original full-precision networks. The table shows that the

1.权值被量化为8位每层，通过找到最优的N，减少因为量化所带来的精度损失。

2.在所有的权值被量化之后，激活被以类似的方式量化，用来找到fractional长度N

3.表6显示8位网络量化的精度，使用了这个方法，比起原始全精度网络。

4.表中显示量化网络的精确性，要不相同要不比全精度网络好，因为量化带来的更好的正则化

accuracy of the quantized network is either same or marginally better than the full-precision network, possibly due to better regularization because of quantization. We believe that the same conclusion will hold for the other neural network models explored in this work.

5 相同的结论被用于其他神经网络模型中

NN model	32-bit floating point model accuracy			8-bit quantized model accuracy		
	Train	Val.	Test	Train	Val.	Test
DNN	97.77%	88.04%	86.66%	97.99%	88.91%	87.60%
Basic LSTM	98.38%	92.69%	93.41%	98.21%	92.53%	93.51%
GRU	99.23%	93.92%	94.68%	99.21%	93.66%	94.68%
CRNN	98.34%	93.99%	95.00%	98.43%	94.08%	95.03%

Table 6: Accuracy comparison of representative 8-bit quantized networks with full-precision networks.

4.5 KWS Deployment on Microcontroller

1.使用CMSIS-NN内核部署KWS到CORTEX M7的开发板STM32F746G-DISCO开发板上。

We deployed the KWS application on Cortex-M7 based STM32F746G-DISCO development board using CMSIS-NN kernels [36]. A picture of the board performing KWS is shown in Fig. 8. The deployed model is a DNN model with 8-bit weights and 8-bit activations and KWS is running at 10 inferences per second. Each inference, including memory copying, MFCC feature extraction and DNN execution, takes about 12 ms. The microcontroller can be put into Wait-for-Interrupt (WFI) mode for the remaining time for power saving. The entire KWS application occupies ~70 KB memory, including ~66 KB for weights, ~1 KB for activations and ~ 2 KB for audio I/O and MFCC features.

2.KWS的开发板如图8所示

3.部署模型是一个DNN模型，具有8位权重和8位激活，每秒有10个推理。

4.每个推理包括内存复制，MFCC特征提取，DNN执行，需要12ms。

5.微控制器可以进入WFI模式来节约用电。

6.整个的KWS应用占用70KB内存，包括66KB权重，1k激活函数，2k用于audio IO和MFCC特征

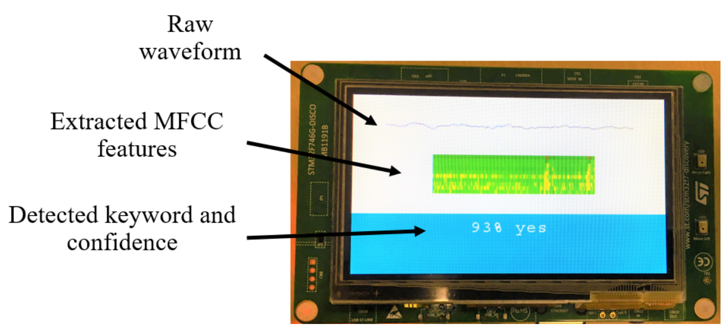


Figure 8: Deployment of KWS on Cortex-M7 development board.

5 Conclusions

1硬件优化的NN结构是在内存和计算受限的微控制器上的得到有效的结果的关键

Hardware optimized neural network architecture is key to get efficient results on memory and compute constrained microcontrollers. We trained various neural network architectures for keyword spotting published in literature on Google speech commands dataset to compare their accuracy and memory requirements vs. operations per inference, from the perspective of deployment on microcontroller systems. We quantized representative trained 32-bit floating-point KWS models into 8-bit fixed-point versions demonstrating that these models can easily be quantized for deployment without any loss in accuracy, even without retraining. Furthermore, we trained a new KWS model using depthwise separable convolution layers, inspired from MobileNet. Based on typical microcontroller systems, we derived three sets of memory/compute constraints for the neural networks and performed resource constrained neural network architecture exploration to find the best networks achieving maximum accuracy within these constraints. In all three sets of memory/compute constraints, depthwise separable CNN model (DS-CNN) achieves the best accuracies of 94.4%, 94.9% and 95.4% compared to the other model architectures within those constraints, which shows good scalability of the DS-CNN model. The code, model definitions and pretrained models are available at https://github.com/ARM-software/ML-KWS-for-MCU.

2训练了在文献中各类NN模型用于KWS，基于GOOgle的语音数据集来比较他们的精度和内存需求，每条推理的操作书，从微控制器的部署方面。

3.量化训练32位的浮点KWS模型到8位浮点版本，容易部署，不会损失精度，不会重新训练。

4.进一步训练新的KWS模型，使用深度分类卷积层，跟MobileNet

5.基于典型的微控制器系统，我们延伸了三种NN内存计算限制的三个集，探索NN发现最好的网络，达到最大精度。

Acknowledgements

We would like to thank Matt Mattina from Arm Research and Ian Bratt from Arm ML technology group for their help and support. We would also like to thank Pete Warden from Google's TensorFlow team for his valuable inputs and feedback on this project.

References

[1] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017. 已经下载

[2] W. Xiong, L. Wu, F. Alleva, Jasha Droppo, X. Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. *CoRR*, abs/1708.06073, 2017. 已经下载

[3] Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Arenas, Kanishka Rao, David Rybach, Ouais Alsharif, Haşim Sak, Alexander Gruenstein, Françoise Beaufays, et al. Personalized speech recognition on mobile devices. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5955–5959. IEEE, 2016.

[4] C Teacher, H Kellett, and L Focht. Experimental, limited vocabulary, speech recognizer. *IEEE Transactions on Audio and Electroacoustics*, 15(3):127–130, 1967.

此人在百度dueros，KITT.AI CTO

[5] Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4087–4091. IEEE, 2014. google引用量达到109 这个感觉相关性好

[6] Tara N Sainath and Carolina Parada. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[7] Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. Convolutional recurrent neural networks for small-footprint keyword spotting. *arXiv preprint arXiv:1703.05390*, 2017.

[8] Ming Sun, Anirudh Raju, George Tucker, Sankaran Panchapagesan, Gengshen Fu, Arindam Mandal, Spyros Matsoukas, Nikko Strom, and Shiv Vitaladevuni. Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 474–480. IEEE, 2016.

[9] Pete Warden. Speech commands: A public dataset for single-word speech recognition. *Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz*, 2017.

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] Jay G Wilpon, Lawrence R Rabiner, C-H Lee, and ER Goldman. Automatic recognition of keywords in unconstrained speech using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(11):1870–1878, 1990.

[12] Richard C Rose and Douglas B Paul. A hidden markov model based keyword recognition system. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 129–132. IEEE, 1990.

[13] Joseph Keshet, David Grangier, and Samy Bengio. Discriminative keyword spotting. *Speech Communication*, 51(4):317–329, 2009.

[14] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. *Artificial Neural Networks–ICANN 2007*, pages 220–229, 2007.

[15] George Tucker, Minhua Wu, Ming Sun, Sankaran Panchapagesan, Gengshen Fu, and Shiv Vitaladevuni. Model compression applied to small-footprint keyword spotting. In *INTERSPEECH*, pages 1878–1882, 2016.

[16] Preetum Nakkiran, Raziel Alvarez, Rohit Prabhavalkar, and Carolina Parada. Compressing deep neural networks using a rank-constrained topology. In *INTERSPEECH*, pages 1473–1477, 2015.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[18] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[19] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6655–6659. IEEE, 2013.

[20] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[21] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[25] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. arX

[26] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. arX

[27] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. arXiv

[28] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[29] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016. arXiv

[30] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017. arXiv

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. arXiv

[32] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. arXiv

[33] Naveen Suda, Vikas Chandra, Ganesh Dasika, Abinash Mohanty, Yufei Ma, Sarma Vrudhula, Jae-sun Seo, and Yu Cao. Throughput-optimized opencl-based fpga accelerator for large-scale convolutional neural networks. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 16–25. ACM, 2016.

[34] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, et al. Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 26–35. ACM, 2016.

[35] Liangzhen Lai, Naveen Suda, and Vikas Chandra. Deep convolutional neural network inference with floating-point weights and fixed-point activations. *arXiv preprint arXiv:1703.03073*, 2017. arXiv

[36] Liangzhen Lai, Naveen Suda, and Vikas Chandra. CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs. *arXiv preprint arXiv:1801.06601*, 2018. arXiv

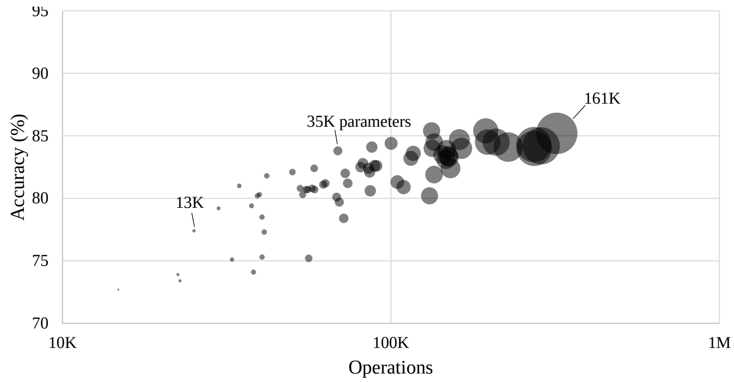
A Appendix: Neural Network Hyperparameters

Table 7 shows the summary of the hyperparameters of the best neural networks described in Table 5, along with their memory, number of operations and accuracy on training, validation and test sets. All the models use 10 MFCC features, with a frame length (L) of 40ms, where as the frame stride (S) is shown in the table. FC stands for fully-connected layer and the number in the parentheses shows the number of neurons in the fully-connected layer. C stands for convolution layer and the numbers in parentheses correspond to the number of convolution features, kernel sizes in time and frequency axes, strides in time and frequency axes. Although not shown, all the convolution and fully connected layers have a ReLU as activation function. L stands for low-rank linear layer with the number of elements shown in parentheses. The number in the parentheses for $LSTM$ and GRU models correspond to the number of memory elements in those models. DSC is depthwise separable convolution layer (DSConv in Fig. 4) and the number in the parentheses correspond to the number of features, kernel size and stride in both time and frequency axes.

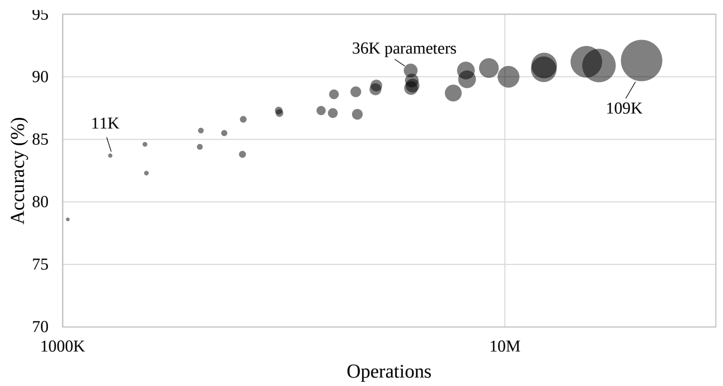
Model	S	NN model hyperparameters	Memory	Ops	Train	Val.	Test
DNN	40	FC(144)-FC(144)-FC(144)	80.0KB	158.8K	91.5%	85.6%	84.6%
DNN	40	FC(256)-FC(256)-FC(256)	199.4KB	397.1K	95.4%	86.7%	86.4%
DNN	40	FC(436)-FC(436)-FC(436)	496.6KB	990.2K	97.8%	88.0%	86.7%
CNN	20	C(28,10,4,1,1)-C(30,10,4,2,1)-L(16)-FC(128)	79.0KB	5.0M	96.9%	91.1%	91.6%
CNN	20	C(64,10,4,1,1)-C(48,10,4,2,1)-L(16)-FC(128)	199.4KB	17.3M	98.6%	92.2%	92.2%
CNN	20	C(60,10,4,1,1)-C(76,10,4,2,1)-L(58)-FC(128)	497.8KB	25.3M	99.0%	92.4%	92.7%
Basic LSTM	20	LSTM(118)	63.3KB	5.9M	98.2%	91.5%	92.0%
Basic LSTM	20	LSTM(214)	196.5KB	18.9M	98.9%	92.0%	93.0%
Basic LSTM	20	LSTM(344)	494.5KB	47.9M	99.1%	93.0%	93.4%
LSTM	40	LSTM(144), Projection(98)	79.5KB	3.9M	98.5%	92.3%	92.9%
LSTM	20	LSTM(280), Projection(130)	198.6KB	19.2M	98.8%	92.9%	93.9%
LSTM	20	LSTM(500), Projection(188)	498.8KB	4.8M	98.9%	93.5%	94.8%
GRU	40	GRU(154)	78.8KB	3.8M	98.4%	92.7%	93.5%
GRU	20	GRU(250)	200.0KB	19.2M	98.9%	93.6%	94.2%
GRU	20	GRU(400)	499.7KB	48.4M	99.2%	93.9%	93.7%
CRNN	20	C(48,10,4,2,2)-GRU(60)-GRU(60)-FC(84)	79.8KB	3.0M	98.4%	93.6%	94.1%
CRNN	20	C(128,10,4,2,2)-GRU(76)-GRU(76)-FC(164)	199.8KB	7.6M	98.7%	93.2%	94.4%
CRNN	20	C(100,10,4,2,1)-GRU(136)-GRU(136)-FC(188)	499.5KB	19.3M	99.1%	94.4%	95.0%
DS-CNN	20	C(64,10,4,2,2)-DSC(64,3,1)-DSC(64,3,1)-DSC(64,3,1)-DSC(64,3,1)-AvgPool	38.6KB	5.4M	98.2%	93.6%	94.4%
DS-CNN	20	C(172,10,4,2,1)-DSC(172,3,2)-DSC(172,3,1)-DSC(172,3,1)-DSC(172,3,1)-AvgPool	189.2KB	19.8M	99.3%	94.2%	94.9%
DS-CNN	20	C(276,10,4,2,1)-DSC(276,3,2)-DSC(276,3,1)-DSC(276,3,1)-DSC(276,3,1)-DSC(276,3,1)-AvgPool	497.6KB	56.9M	99.3%	94.3%	95.4%

Table 7: Summary of hyperparameters of the best models described in Table 5.

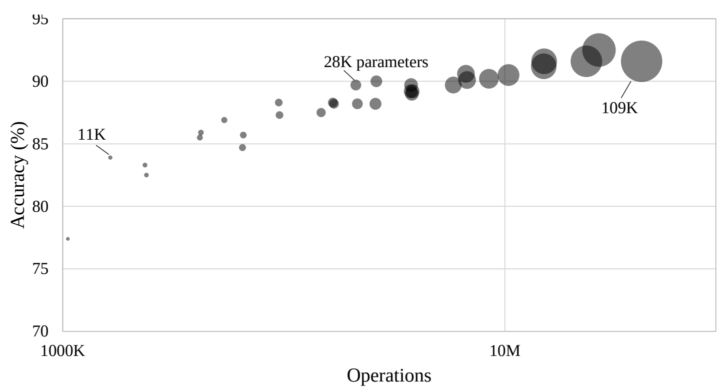
Figures 9(a), 9(b), 9(c), 9(d) show the hyperparameter search of DNN, basic LSTM, LSTM and CRNN architectures depicting the model accuracy vs. number of operations. The model size is depicted by the size of the circle.



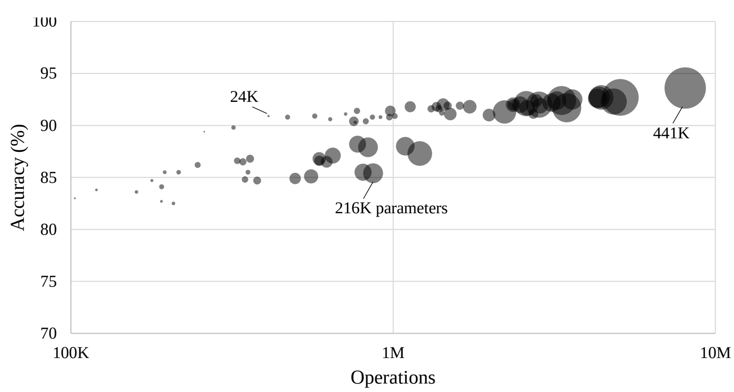
(a) DNN



(b) Basic LSTM



(c) LSTM



(d) CRNN

Figure 9: Hyperparameter search for (a) DNN, (b) basic LSTM, (c) LSTM and (d) CRNN showing the model accuracy vs. operations, with the number of parameters depicted by the size of the circle.