2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin).

Studying the Effects of Feature Extraction Settings on the Accuracy and Memory Requirements of Neural Networks for Keyword Spotting

Muhammad Shahnawaz*†, Emanuele Plebani†, Ivana Guaneri‡, Danilo Pau†(*Senior Member IEEE*), Marco Marcon*
*Image and Sound Processing Lab, DEIB, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133, Milano, Italy
†STMicroelectronics, Via Camillo Olivetti, 2, 20864 Agrate Brianza, MB, Italy
‡STMicroelectronics, Str. Primosole, 50, 95121 Catania, Italy

Abstract

Due to the always-on nature of keyword spotting (KWS) systems, low power consumption micro-controller units (MCU) are the best choices as deployment devices. However, small computation power and memory budget of MCUs can harm the accuracy requirements. Although, many studies have been conducted to design small memory footprint neural networks to address this problem, the effects of different feature extraction settings are rarely studied. This work addresses this important question by first, comparing six of the most popular and state of the art neural network architectures for KWS on the Google Speech-Commands dataset. Then, keeping the network architectures unchanged it performs comprehensive investigations on the effects of different frequency transformation settings, such as number of used mel-frequency cepstrum coefficients (MFCCs) and length of the stride window, on the accuracy and memory footprint (RAM/ROM) of the models. The results show different preprocessing settings can change the accuracy and RAM/ROM requirements significantly of the models. Furthermore, it is shown that DS-CNN outperforms the other architectures in terms of accuracy with a value of 93.47% with least amount of ROM requirements, while the GRU outperforms all other networks with an accuracy of 91.02% with smallest RAM requirements.

I. Introduction

Advancements in the field of deep learning field have come to a stage where machines' cognition have overcome the human cognition in various tasks such as object detection, image classifications [1] and natural speech utterance and recognition [2]. Thanks to these advancements, speech based human-machine interaction is becoming more and more natural these days. Some of the most recent and popular examples include Amazon Echo, Google Home and Smart Phones, Siri for Apple and Cortana based solutions for MS Windows. To make the experience as natural as possible the speech interface needs to have an always-on nature and requires to transmit the speech to the cloud based speech recognition engines constantly. For an example: on average a normal human being speaks almost 16,000 words/day [3], with an average rate of 163 words/minute [4], this translates to continuous speech of 98 minutes. Considering the 128 kbps data rate, the speech data to be transmitted per person per device per day is 94 MBs, which results in a lot of power consumption in terms of wifi and network usage [5]. Also, considering the mass availability of such devices, the speech data to be processed can easily reach upto to 100s of PBs per day. This can easily create the congestion problems for the network and cloud services. Furthermore, the latency due to transmission can harm the real-time response requirements. Nonetheless, continuous speech transmission creates privacy concerns for the users.

To overcome these problems without compromising the utility of the speech recognition applications, the concept of "Hot/Keyword" based activation of speech recognition systems came into existence [6]. The idea is to run the small speech recognition algorithms locally on the device which can spot one or more predefined keyword(s) such as **Alexa**, **Cortana**, **Ok Google**, **Hey Siri**, etc in a continuous speech stream, which gives it its name "keyword spotting". Still keeping the application processor active all the time puts a heavy load on the battery. If this task fits on a dedicated tiny MCU which can work in isolation from the main application processor in the smart phones, it can potentially save a lot of power consumption. Then, once the keyword is spotted, the MCU can wake up the application processor, device and the speech recognition engine which can be either be offline [7], [8], cloud based [9] or a mixture [10] to perform the main speech recognition. An other use case for keyword spotting applications in these days is to use the sequence of small keywords as voice commands to control smart devices such as a voice-enabled smart home systems [11], remote controls for smart tvs [12], or to operate a small robot [13] etc.

Since KWS system has to be always-on and run on the MCUs, it would be desirable to have a very small memory footprint, small computation power and very low power con-

Corresponding author: Muhammad Shahnawaz, Danilo Pau
Emails: muhammad.shahnawaz@polimi.it, danilo.pau@st.com

978-1-5386-6095-9/18/$31.00 ©2018 IEEE

sumption. On the other hand, to provide a natural experience the KWS system should detect the keywords with high accuracy e.g. $> 90\%$ and low latency. These conflicting system requirements make KWS an active area of research ever since it was proposed over 50 years ago [7]. Recently, with the renaissance of deep learning, neural network have become a very popular choice for KWS systems such as: [14]–[18]. However, for neural network based KWS to be deployed on the MCUs, there are two main challenges e.g. **Limited memory footprint** and **Limited computation power**, that must be kept in mind while designing. The network architecture, should be so that it can fit including the activations and weights in the small budget of few hundreds KBs of RAM/ROM available on the MCUs. Also, the limited computation power in terms of number of operation per inference in MCU for the real-time requirements can harm the accuracy.

Although many studies have tried to develop the neural networks for KWS with small memory footprint yet with high accuracy [18], the effects of choice of feature extraction settings on the performance and memory footprint are rarely studied. This work investigates the effects of different settings for feature extraction process from the raw audio i.e. changing the number of used MFCCs and stride size of framing window can effect the accuracy and memory foot print of a neural nets. Thus the main contributions of this work are as follows:

1) In this work we train six of the state of the art KWS neural networks presented in [18], on Google speech commands dataset [19] and compare them in terms of accuracy and required memory (ROM & RAM).
2) Then keeping the network unchanged we investigate the effects of different preprocessing settings e.g. the numbers of the MFCCs used, or different stride sizes on the accuracy and size of the memory footprint.
3) Furthermore, networks are evaluated against three benchmarks.
   - **Best accuracy**, ignoring all other constraints.
   - Best accuracy for the **smallest RAM requirements**.
   - Best accuracy for the **smallest ROM requirements**.

II. Background

A. State of the art

The pipeline of the KWS systems is similar to any other classification system and is shown in Fig. 1. Like any deep learning system the first step is the preprocessing and feature extraction. The most commonly used human engineered speech recognition features features are Log-mel Filter Bank Energies (LFBE) and Mel-Frequency Cepstral coefficients (MFCC) [20], [21]. For the purpose of this study we used MFCCs. Speech is a non-stationary signal i.e. the frequency contents of the signal change over time, for this reason the first step involved in modeling and feature extraction of the speech signal is to perform the short time fourier transform. The speech signal is windowed into small overlapping frames assuming the framing duration is short enough to keep the signal stationary yet long enough to model the lowest spectrum components of human speech. Typically the length of the frame is between 20 to 40 msec. We used 40 msec long windows. If the length of the input speech signal is denoted by L samples, the length of the framing window is l samples and the stride between adjacent frames is s samples, the speech signal is divided into N small overlapping frames, where $N = \frac{L-l}{s} + 1$. The overlapping windows are used in order to exploit the local relationship between the consecutive frames. From each signal frame, D speech features are extracted, generating a total of $N \times D$ feature matrix, which contains the information of the entire input speech signal.

The aim of this feature extraction process transforms the raw speech signal into a compressed time-frequency domain representation, which reduces the dimensionality yet preserves the relevant information, which makes the learning process for neural networks easy. Once the features have been extracted, the neural networks are trained using these feature matrices to classify them into different classes. In case of a real-world implementations and additional step of posterior handling is required which averages the output probabilities of each class after every inference over a given period of time. This step increases the overall confidence of the classification task [18].

Traditionally, KWS systems are made using Hidden Markov Models (HMMs) and Viterbi decoding. Although, these solutions can provide descent performances the training process for these techniques is quite complex. Also, driving the inferences through these models is expensive computationally. Some other techniques explored for KWS are discriminative models, which formulates the problem using a large-margin problem formulation [22] and artificial neural networks (ANN) [23]. Its worth mentioning that comparing to HMM and Viterbi decodingm these methods result in very good performance. However, thes require a large number of operations which can introduced the latencies harming the real-time experience.

During the last decade many studies have been conducted in order to solve these problems, by to creating the KWS systems which have a small budget in terms of memory and computation power however still are capable to provide good performances. For example, in [14], author proposed a fully connected deep neural network (DNN) with rectified linear unit (ReLU) activation for KWS. This has a very small footprint and small computation requirements still surpasses the accuracy of the HMM. Furthermore, [24], [25] proposed some low-rank approximation techniques which can compress the DNN model weights achieving similar accuracy with less resources. However, DNNs is not able to model the local temporal and spectral correlation in the input speech features, it can not model the speech efficiently. In [15], proposed a convolutional neural network (CNN) which can capture the local temporal and spectral correlation in feature matrices to overcome these problems, hence having a better accuracy. However, CNN are unable to capture the global spectral and temporal features. In [16], proposed a small footprint recurrent neural network (RNN) which can model the long term temporal dependencies hence best for the sequence data resulting in a better performance in speech processing. However, this fails to

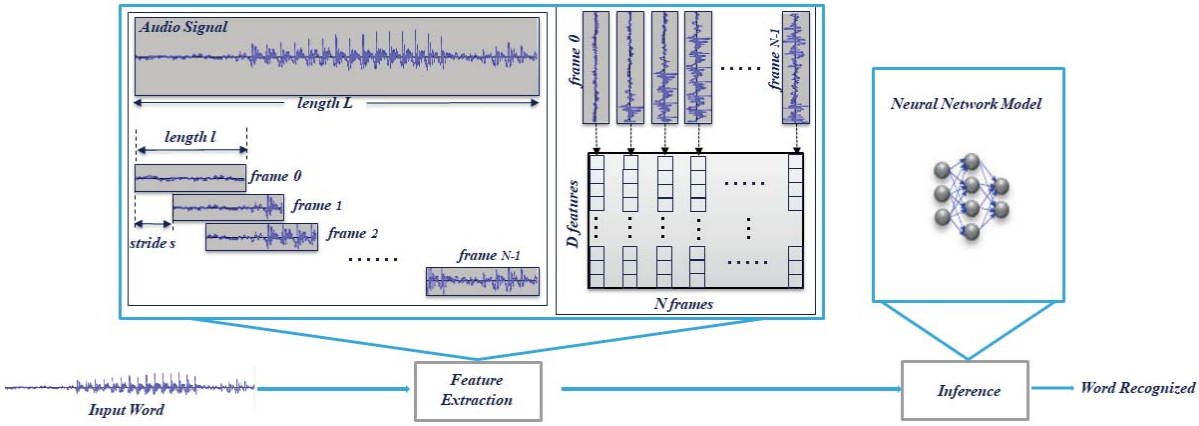


Fig. 1: Keyword spotting pipeline [18].

capture the spectral correlation. In [17] proposed a new method called convolutional recurrent neural network (CRNN), which combines the benefits of both CNN and RNN and is proven to be robust against noise in speech signal.

Despite the fact that many neural network based solutions were available and used for KWS, it was really hard to make a fair comparison because different architectures are trained on different datasets and with different settings. In [18], authors performed a detailed comparison between seven of the most popular neural networks for KWS. Rigorously exploring the hyper parameters of these networks they identify the architectures which can satisfy the memory and computational power budget of the three predefined specifications of MCU classes. Although, this study provides great insights about the architectures, however, it ignores one very important aspect, the effect of different preprocessing settings on the memory footprint and accuracy. To answer this question in this work we evaluate the six of the chosen neural network architectures from [18], against 24 different feature extraction settings.

B. Artificial Networks explored in this work

In this work we have explored the following six of seven artificial neural network architectures to meet the smallest constraints studied in [18].

1) Deep Neural Network (DNN): The Deep neural networks (DNN) models are the most basic and simple, Multi-Layer Perceptrons (MLP) with the same number of neurons in each layer as described in [18]. We used the simplest DNN models with a rectified linear unit (ReLu) as activation function. The DNN input is the flattened $N \times D$ feature matrix as described in section II-A. The outputs are converted to the probabilities through a softmax function at the last layer. For details of the hyper parameters refer to the appendix A of [18].

2) Convolutional Neural Network (CNN): Although DNN is very simple, it is inefficient both in terms of number of parameters and ability to capture the temporal and spectral correlation in the speech features. Convolutional neural networks (CNNs), on the other hand exploits those correlations by treating the whole feature matrix as a 2D image and performing $2D$ convolutions over it. The convolution layers are typically followed by batch normalization [26], ReLU activation functions and optionally max/average pooling layers to reduce the dimensionality of the features. During the inference, parameters of the batch normalization can be folded into the weights of the convolution layers. In some cases, an additional linear low-rank layer, which is simply a fully-connected layer without a non-linear activation, is added in between the convolution layers and dense layers to reduce the number of parameters and accelerate the training [27]. In this work we used the CNN from [18] for details refer to the appendix A of the paper.

3) Long Short Term Memory (LSTM): LSTM models are a form of recurrent neural networks (RNNs) family, which can capture the long-term dependencies through the use of a memory with controlled reads and writes. Compared to CNNs, they show superior performances for modeling the sequence data, especially speech and natural language modeling as shown by several studies [28]–[30]; moreover, tend to have a lower number of parameters since the weights are reused across time steps. LSTMs operate for T time steps, where at each time step t the corresponding spectral feature vector $f_t \in R^F$ is concatenated with the previous time step output h_{t-1} and is used as input to the LSTM layer. For details on the hyper parameters refer to the appendix A of the [18].

4) Gated Recurrent Unit (GRU): GRU models use a simplified form of the LSTM layer and they can similarly model long-term dependences but with fewer parameters and gives better convergence on this task. On audio recognition tasks, they performs on par with LSTM [**?**]. For details on the hyper parameters of the GRU architecture used in this study refer to the appendix A of [18].

5) CRNN: Convolution recurrent neural network [17] is a hybrid of CNN and RNN. Combining the advantages from both architectures it exploits the local temporal/spatial correlation using convolution layers and global temporal dependencies in the speech features using recurrent layers. In this study a combination of 2D CNN layers and GRU layer is used as it uses fewer parameters than LSTMs and gave better convergence in our experiments. For details of the hyper parameters refer to the appendix A of [18].

6) DS-CNN: Recently, a new and very efficient variant of CNN has been proposed in [31], which implements the traditional 3D convolution by separating it depthwise, hence called the depthwise separable convolution neural network (DS-CNN). DS-CNN first convolves each channel in the input feature map with a separate 2D filter and then uses pointwise convolutions (i.e.1x1) to combine the outputs in the depth dimension. By decomposing the standard 3-D convolutions, DS-CNN is more efficient both in number of parameters and operations, which makes deeper and wider architecture possible even in the resource-constrained conditions. 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. Though primarily designed for the vision problems, this architecture shows very promising results in the area of KWS. For details of the hyper parameters refer to the appendix A of [18].

III. Experiments

A. Dataset and experimental setup

All the experiments done in this study used Speech-Commands dataset [19] launched by Google and Tensorflow. The dataset contains 65,000, one second long audio clips with a sample rate of 16,000 samples/sec. Each file has utterance of only one command word. As this dataset is collected using crowd sourcing, it contains audio from different speakers in different accents and in different noisy scenarios. This ensures the trained KWS neural networks will be speaker and accent independent.

In this work we trained the neural networks to classify the incoming sound into 12 classes: “zero”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”, “unknown” (containing data from remaining 20 classes) and ”silence”. The ratio for unknown and silence samples are kept at 10% of all the other samples. The data is then divided in to “80:10:10” ratio for training, validation and testing, making sure that the audio clips from the same person are in the same set.

All the models are trained in the Keras framework [32] using Tensorflow back-end, with cross-entropy and “ADAM” [33] as the loss function and optimizers respectively. The batch-size is 100 and the learning rate starts from 5×10^{-4} and drops to one fifth every 10^4 iterations, with total 3×10^4 iterations. The motivation for using a decaying learning rate is to fine tune them to the very precise values.

B. Results and discussion

1) Basic evaluation: First of all we evaluated all the neural networks with a simple setting as proposed by [18] and calculated the feature matrices using a framing window of size of 40 msec with 50% overlap and using only 10 MFCCs, yielding a matrix of size (49×10) for each file. The neural networks are then trained using these feature matrices. Table I summarizes the results reporting the required memory, both RAM (for activations) and ROM (for weights), the classification accuracy on test set along with the required Millions of Operations (MOPS) assessed on an embedded platform [34]. The size of memory footprint shown in the table assumes that all the weights and activations are for a single inference and all are 32-bit floating point numbers.

TABLE I: Basic Evaluations

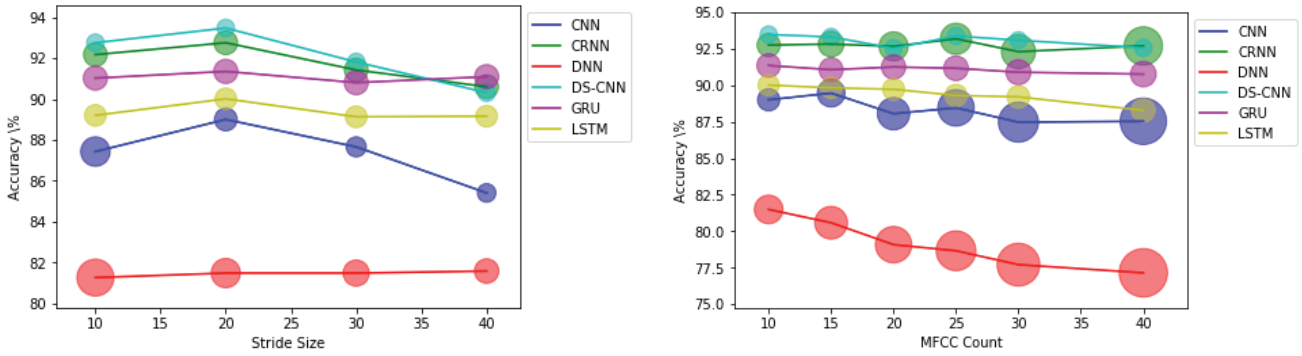
Model architecture	RAM (KBs)	ROM (KBs)	Accuracy %	MOPS*
DNN	1.13	446.11	81.49	0.23
CNN	38.13	271.37	89.00	5.06
LSTM	0.51	243.42	90.01	1.59
GRU	0.65	305.04	91.35	7.46
CRNN	19.69	294.66	92.75	3.02
DS-CNN	62.50	161.30	93.47	9.10

Another assumption worth mentioning is that the memory for activations is reused between different layers, hence the memory requirements are computed only for the maximum size of two consecutive layers activations.

2) Exploration of different stride and number of MFCCs: We performed an exhaustive study to understand the effects of different frame stride and the number of MFCCs used on the accuracy and memory requirements. Keeping the architecture and frame size fixed, we tried 24 combinations of different settings with modifying the stride size from 10 msec to 40 msecs with a step of 10 msecs and MFCC count to $\{10, 15, 20, 25, 30, 40\}$.

The results reported in Fig. 2, show the effects of stride size and number of features to be used on the accuracy and memory footprint size. It is evident from the graphs that, the performance of the all networks stays the same when changing the stride size, except the convolution based neural networks which decreases by increasing the stride size. The reason for this is that the convolution neural networks utilized the local temporal correlation which decreases while increasing the size of the stride. Also it is worth noticing that, increasing the stride size decreases the size of ROM for CNN and DNN while the ROM requirements for all other networks stay the same. In case of increasing the MFCC count, the performance of all networks except DNN stays the same. The reason is the over fitting phenomenon due to large capacity of the network. While the ROM for all networks except GRU and LSTM increase with increasing the number of MFCCs. Combining the analysis it can be said that the GRU and LSTM memory foot print sizes are independent of the MFCC count and window stride size.

Furthermore, we evaluate the performance of the networks evaluated against following three different bench marks we



(a) Accuracy and ROM vs Stride Size (MFCC count = 10)

(b) Accuracy and ROM vs MFCC count (stride size = 20msec)

Fig. 2: Effects of stride size and MFCC count on accuracy and memory footprint (RAM/ROM).

created.

Highest Accuracy

This benchmark is created to identify the combination of chosen stride size and the used MFCC count which results in highest accuracy performance for the tested architectures. The results are summarized in Table II. Table shows the accuracy on test instances, ROM and RAM footprint size, along with the selected stride size in msec and count of MFCCs used. It can be observed from the table that that among all the test architectures DS-CNN provides the best performance in terms of accuracy. Another thing to be noted is that although it provides the best performance it has the least number of parameters and can fit into a device with a ROM size as big as 170KBs.

TABLE II: Best test accuracies

Model architecture	RAM (KBs)	ROM (KBs)	Stride (msec)	Nr. of MFCC	Accuracy %
DNN	1.13	311.11	40	10	81.59
CNN	69.38	421.37	20	15	89.46
LSTM	0.51	243.42	20	10	90.01
GRU	0.65	314.06	40	15	92.00
CRNN	59.81	362.17	10	15	93.21
DS-CNN	62.50	161.30	20	10	93.47

Minimum RAM requirements

As RAM is a very expensive component, hence most of the MCUs have very small amount of RAM available. Also, the RAM access operations are very power hungry. So, keeping that in mind this study creates a second bench mark the smallest RAM footprint for best performance. In these experiments it identifies the feature settings for each architecture which provides best performances keeping the RAM requirements to minimum. The results are presented in Table III. It can be seen that the RAM requirements are smallest for the LSTM and GRU comparing to the other network architectures, in fact even less than a KB. Also, in terms of accuracy GRU outperforms all other architectures with an accuracy of 91.02%, while the LSTM is just 2% behind. CRNN and DS-CNN shows almost same performance as GRU but require 12 and 50 times more RAM respectively.

TABLE III: Minimum RAM

Model architecture	RAM (KBs)	ROM (KBs)	Stride (msec)	Nr. of MFCC	Accuracy %
DNN	1.13	311.11	40	10	81.59
CNN	14.13	181.37	40	10	85.41
LSTM	0.51	242.72	40	10	89.16
GRU	0.65	305.04	10	10	91.02
CRNN	7.88	294.66	40	10	90.60
DS-CNN	32.50	161.30	40	10	90.31

Lowest ROM requirements

Different MCUs have different configurations and different size of ROM. However the size of the ROM is a crucial part to be considered for neural network architects for KWS systems. As the access to ROM is slow and can really add the latency to the inference process which can harm the real time experience. Also, having small ROM requirements means more deployment choices. Keeping this in mind this study creates the third benchmark: the smallest ROM requirements. The results are summarized in Table IV. Its evident from the table that the DS-CNN outperforms all the other architectures with a smallest ROM requirements of just 161.3KBs and still provides the best accuracy of 93.47%. CRNN has almost same performance yet requires almost double amount of ROM. While for all the other architectures the accuracies are low even though these require higher amount of ROM.

TABLE IV: Minimum ROM

Model architecture	RAM (KBs)	ROM (KBs)	Stride (msec)	Nr. of MFCC	Accuracy %
DNN	1.13	311.11	40	10	81.59
CNN	14.13	181.37	40	10	85.41
LSTM	0.51	242.72	40	10	89.16
GRU	0.65	305.04	10	10	91.02
CRNN	43.31	294.66	10	10	92.17
DS-CNN	62.50	161.30	20	10	93.47

IV. Conclusion and Discussion

To satisfy the low power consumption, real-time response for the always on KWS systems MCUs are the best deploy-

ment devices. This work, starts from the work of [18] and investigated the minimum footprint neural network designs for different preprocessing settings. Keeping the network architecture unchanged we evaluate the neural networks against 24 different feature extraction settings. The results in Fig. 2 summarize how changing the stride window size and number of MFCCs can change the accuracy and (RAM/ROM) requirements for KWS systems. We also benchmarked the explored neural networks against three different criteria, highest accuracy without considering any other bound, best accuracy with the least amount of RAM requirements and best accuracy with least amount of ROM requirements. The results shows that both LSTM and GRU requires less than 1kB of RAM. Though LSTM requires the least amount of RAM it only proved 81.42% accuracy. On the other hand with a little bit more RAM requirement GRU architecture provides an improvement of 10% in accuracy. DS-CNN wins in both departments of least ROM requirements and best accuracy with $161Kbs$ and 93.45% respectively.

REFERENCES

[1] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 4470–4478.

[2] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "The microsoft 2016 conversational speech recognition system," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5255–5259.

[3] M. R. Mehl, S. Vazire, N. Ramírez-Esparza, R. B. Slatcher, and J. W. Pennebaker, "Are women really more talkative than men?" *Science*, vol. 317, no. 5834, pp. 82–82, 2007.

[4] A. Dlugan. What is the average speaking rate? [Online]. Available: http://sixminutes.dlugan.com/speaking-rate/

[5] M. Tawalbeh, A. Eardley *et al.*, "Studying the energy consumption in mobile devices," *Procedia Computer Science*, vol. 94, pp. 183–189, 2016.

[6] H. Christensen, I. Casanueva, S. Cunningham, P. Green, and T. Hain, "homeservice: Voice-enabled assistive technology in the home using cloud-based automatic speech recognition," in *Proceedings of the Fourth Workshop on Speech and Language Processing for Assistive Technologies*, 2013, pp. 29–34.

[7] C. Teacher, H. Kellett, and L. Focht, "Experimental, limited vocabulary, speech recognizer," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 3, pp. 127–130, 1967.

[8] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays *et al.*, "Personalized speech recognition on mobile devices," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5955–5959.

[9] J. Twiefel, T. Baumann, S. Heinrich, and S. Wermter, "Improving domain-independent cloud-based speech recognition with domain-dependent phonetic post-processing." in *AAAI*, 2014, pp. 1529–1536.

[10] T. M. Soemo, L. Soong, M. H. Kim, C. R. Heinemann, and D. H. Hawkins, "Integrated local and cloud based speech recognition," Feb. 25 2014, uS Patent 8,660,847.

[11] D. Grammenos, S. Kartakis, I. Adami, and C. Stephanidis, "Camile: controlling ami lights easily," in *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2008, p. 35.

[12] J.-S. Park, G.-J. Jang, J.-H. Kim, and S.-H. Kim, "Acoustic interference cancellation for a voice-driven interface in smart tvs," *IEEE Transactions on Consumer Electronics*, vol. 59, no. 1, pp. 244–249, 2013.

[13] P. J. Prodanov, A. Drygajlo, G. Ramel, M. Meisser, and R. Siegwart, "Voice enabled interface for interactive tour-guide robots," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2. IEEE, 2002, pp. 1332–1337.

[14] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Acoustics, speech and signal processing (icassp), 2014 ieee international conference on*. IEEE, 2014, pp. 4087–4091.

[15] 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.

[16] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 474–480.

[17] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," *arXiv preprint arXiv:1703.05390*, 2017.

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

[19] P. Warden, "Speech commands: A public dataset for single-word speech recognition," *Dataset available from http://download. tensorflow. org/data/speech_ commands_ v0*, vol. 1, 2017.

[20] I. Patel and Y. S. Rao, "Speech recognition using hmm with mfcc-an analysis using frequency specral decomposion technique," *Signal & Image Processing: An International Journal (SIPIJ)*, vol. 1, no. 2, pp. 101–110, 2010.

[21] R. Vergin, D. O'shaughnessy, and A. Farhat, "Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 5, pp. 525–532, 1999.

[22] J. Keshet, D. Grangier, and S. Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009.

[23] 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.

[24] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting." in *INTERSPEECH*, 2016, pp. 1878–1882.

[25] P. Nakkiran, R. Alvarez, R. Prabhavalkar, and C. Parada, "Compressing deep neural networks using a rank-constrained topology," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[27] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. 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*. IEEE, 2013, pp. 6655–6659.

[28] H. Sak, A. Senior, and F. 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.

[29] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[30] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[31] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *arXiv preprint arXiv:1707.01083*, 2017.

[32] F. Chollet, "keras," Available at https://github.com/fchollet/keras, 2015.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] STMicroelectronics, "Sensortile development kit," Available at https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html.