The 35th Conference on Computational Linguistics and Speech Processing (ROCLING 2023)
Taipei City, Taiwan, October 20-21, 2023. The Association for Computational Linguistics and Chinese Language Processing

Compact CNNs for End-to-End Keyword Spotting on Resource-Constrained Edge AI Devices

Joseph Lin
Hsinchu County American School
Hsinchu, Taiwan
lintinghaojoseph@gmail.com

Ren-yuan Lyu
Department of Computer Science
and Information Engineering
Chang Gung University
Taoyuan, Taiwan
renyuan.lyu@gmail.com

Abstract

In this paper, we explore compact convolutional neural networks (CNNs) for end-to-end keyword spotting from raw audio to final recognition results, without using traditional feature extraction based on spectrogram. Such fully CNN models reach 90.5% accuracy, an improvement of 12.15% over traditional methods with similar structures, which only achieve 78.35% accuracy, on the Speech Commands dataset. This shows that learned CNN features outperform predefined FFT-based transforms. The results show that compact end-to-end CNNs enable efficient, accurate small vocabulary keyword spotting that is well-suited for resource-constrained edge devices. All code will be released on the GitHub of the authors [Lin and Lyu, 2023].

Keywords: End-to-end models, raw audio processing, keyword spotting

1 Introduction

Keyword spotting is a critical technology for edge applications such as voice-activated devices, smart speakers, industrial automation, security surveillance, and virtual assistants [Hoy, 2018]. It allows users to interact with devices hands-free, control devices without remote controls, operate machinery efficiently and safely, detect and respond to emergencies quickly, and get help and information from virtual assistants without opening dedicated apps. This technology has seen significant advances with the rise of deep learning.

Recently, an open-sourcing neural net called Whisper from OpenAI, which approaches human-level robustness and accuracy in English and multilingual speech recognition, has been released. [Radford, 2021] The Whisper model has proven to be very successful in large vocabulary speech recognition tasks. However, when it comes to keyword spotting on resource-constrained devices, the Whisper model has some disadvantages compared to a CNN-based model [Dai, 2016] due to its model complexity, inference speed/latency, and large memory footprint. This can be problematic on devices with limited computational and storage capacity. Recent research [Petrov, 2023] further shows that language model tokenizers favor English over other languages, exacerbating the resource constraints of non-English edge applications.

Convolutional neural networks (CNNs), on the other hand, have been proven particularly effective for speech processing tasks compared to older statistical models [LeCun, 2015]. However, CNNs may quickly lose their advantages if the network is too deep [He, 2016].

In this paper, we explore various CNN architectures for an end-to-end keyword-spotting application. Specifically, we investigate replacing the commonly used Fourier transform preprocessing with learned convolution layers for directly processing raw audio input. We also examine the tradeoffs between smaller and larger-scale CNN models in terms of accuracy and overfitting.

Our contributions are three-fold. First, we demonstrate competitive accuracy with an end-to-end CNN model operating directly on raw audio data, removing the need for engineered feature extraction. Second, we show that smaller CNN models can approach the accuracy of larger counterparts, reducing overfitting concerns given the limited training data size. Finally, we identify critical design choices regarding convolution window size and stride, model depth, and training procedures that impact performance.

Overall, this work provides good insights into the practical application of convolutional neural networks for small-vocabulary keyword spotting. Our findings on end-to-end learning from raw audio and model sizing considerations could help guide the design of accurate and compact CNN architectures suitable for embedded speech recognition applications. The tradeoffs identified also

suggest promising directions for further improving the accuracy of this audio classification task.

This paper is organized as follows. Section 2 presents the model architecture, which consists of several convolutional layers, a pooling layer, 2 fully connected layers, and a softmax layer. Section 3 describes the experimental setup, including the SPEECHCOMMAND dataset used in this study. Section 4 reports the primary experimental results. Finally, Section 5 concludes the paper and discusses future work.

2 Model Architectures

Convolutional Neural Networks

Convolutional neural network layers offer some critical advantages over Fourier transforms for processing raw audio data in speech recognition systems. The convolution kernels are optimized during training to extract the most useful representations of the input audio for the specific task. They learn data-driven features tailored to the dataset, rather than relying on a predefined transformation like the Fourier transform. Therefore, we hypothesize that CNNs can discover optimal ways to transform the raw waveforms to best feed into later network layers. Additionally, convolutional layers provide more flexibility compared to Fourier transforms in how the audio is processed. Parameters such as kernel size, stride, padding, and the number of filters can be tuned to appropriately transform the audio. With a Fourier transform, you get a fixed transformation with less ability to configure it to the data. Taking inspiration from the paper [Dai, 2016], we constructed a set of pure convolutional layers aiming to perform speech recognition of 35 English words.

In Figure 1, we described the actual CNN (left side) and FFT-based Mel-Spectrogram model (right side). First of all, the input is a 1x16000 tensor, which represents a 1-second audio with a sample rate of 16000. Traditionally, the input tensor is then transformed into a 2D tensor with a window size of 320, a hop length of 160, and 64 output channels. This is done by using the MelSpectrogram function from torchaudio, a python package in PyTorch platform and we will see a 2-dimensional tensor with a shape of 64x99, which is precisely a mel-scaled spectrogram shown as a 2D image on the right side of Figure 1.

Several convolution layers with a window length of 4, a stride of 2, and doubled output channels are then applied to the input tensor. The model reaches a maximum of 256 channels and then keeps the number of channels at 256 until the second-to-last convolution layer. Then, a convolutional layer reduces the number of channels from 256 to 128. The following tensor is then passed through a pooling layer, which averages out all the values in each feature map, resulting in a tensor with a shape of 128 x 1. The resulting tensor is then passed through two fully connected layers with shapes of 128x1 and 64x1 and then reaches the number of output classes, 35, the number of possible word options. The tensor is also passed through a softmax layer, which outputs the probability of each predicted word. The word with the highest probability is then chosen as the model's output.

The Python code for the prototype structure of the model is also shown in Figure 2, as a Python class asrCNN. This model is trained with the Adam optimizer and a learning rate of 0.001. The loss function is the cross-entropy loss function. The model is trained for about 30 epochs, and the model with the highest accuracy on the validation dataset is chosen as the final model. The model is then tested on a completely new dataset, the testing dataset, and the accuracy is reported as the final accuracy of the model.

3 Experiment Setup

The Dataset

Speech Commands is an open-source dataset consisting of 105,829 one-second English utterances of 35 words from 2,618 speakers [Warden, 2018]. It includes common words like digits and directions, as well as background noise clips. The files are in 16kHz WAV format, and the uncompressed waveforms total 3.8GB. The dataset's author suggested using specific files for training, testing, and validation, resulting in 84,843, 11,005, and 9,981 files, respectively. During our development, we utilized the validation set and only used the testing dataset once, as our model had already determined its "best performance" using the validation set. Therefore, we required an additional dataset that had not been used during our development process. We report the results of testing on this new dataset as the final results. In each model (Table 1: asrCNN1, asrCNN2, asrCNN3, and asrCNN4), we retained the model's parameters if it achieved the highest validation dataset accuracy. Otherwise, we replaced the parameters with those

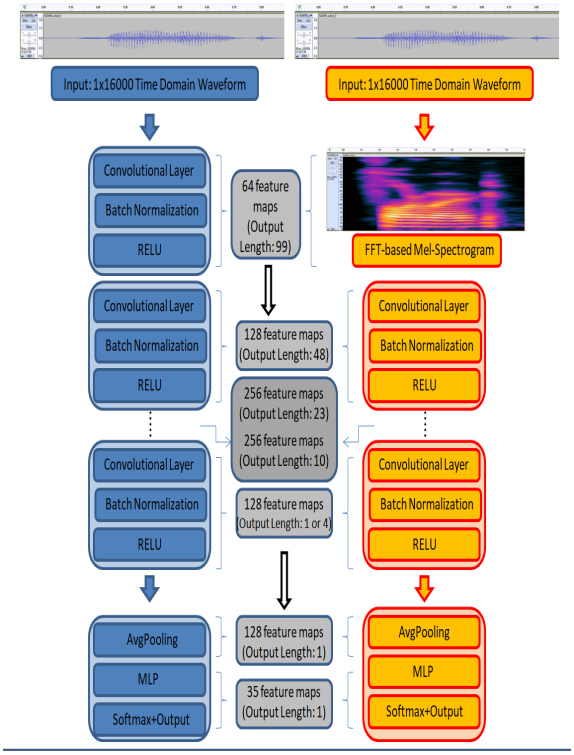

Figure 1: Diagram which describes the CNN (left) and FFT-based Mel-Spectrogram (right) models.

```python
import torch
import collections
import torch.nn as nn

class asrCNN(nn.Module):
    def __init__(self, in_chs= 1, out_cls= 35):
        super(asrCNN, self).__init__()
        layers= collections.OrderedDict([
            ('c1', nn.Conv1d(in_chs,  64, 320, 160)),
            ('b1', nn.BatchNorm1d(64)),
            ('r1', nn.ReLU()),
            '''
            mark the above 3 lines
            unmark the following 1 line
            when using mel-spectrogram model
            '''
            #('mels', ryMelsgram1d()),
            ('c2', nn.Conv1d(64,  128, 4, 2)),
            ('b2', nn.BatchNorm1d(128)),
            ('r2', nn.ReLU()),
            ('c3', nn.Conv1d(128, 256, 4, 2)),
            ('b3', nn.BatchNorm1d(256)),
            ('r3', nn.ReLU()),
            ('c4', nn.Conv1d(256, 256, 4, 2)),
            ('b4', nn.BatchNorm1d(256)),
            ('r4', nn.ReLU()),
            ('c5', nn.Conv1d(256, 256, 4, 2)),
            ('b5', nn.BatchNorm1d(256)),
            ('r5', nn.ReLU()),
            ('p1', ryAvgPool1d()),
            ('l1', nn.Linear(256, 128)),
            ('t1', nn.Tanh()),
            ('l2', nn.Linear(128, out_cls)),
            ('out',nn.LogSoftmax(dim=-1))
            ])
        self.model= nn.Sequential(layers)
    def forward(self, x):
        x= self.model(x)
        return x
```

Figure 2: the python code for the proto-type asrCNN model.

of the new model if it achieved higher validation dataset accuracy. Finally, we tested each model on a completely new dataset, the testing dataset.

	asrCNN1	asrCNN2	asrCNN3	asrCNN4
1	Input: 1x16000 time domain waveform			
2	MelSpectrogram (1,64,320,160) ⇒ [64,99]		Conv1d (1,64,320,160) +BatchNorm1d +Relu ⇒ [64x99]	
3	(64,128,4,2) Batch-Norm1d + Relu ⇒ [128x48]		(64,128,4,2) Batch-Norm1d + Relu ⇒ [128x48]	
4	(128,256,4,2) + BatchNorm1d + Relu ⇒ [256,23]			
5	(256,256,4,2) + BatchNorm1d + Relu ⇒ [256,10]			
6	(256,128,4,2) +Batch-Norm1d +Relu ⇒ [128,4]	(256,256,4,2) +Batch-Norm1d +Relu ⇒ [256,4]	(256,128,4,2) +Batch-Norm1d +Relu ⇒ [128,4]	(256,256,4,2) +Batch-Norm1d +Relu ⇒ [256,4]
7		(256,128,4,2) +Batch-Norm1d +Relu ⇒ [128,1]		(256,128,4,2) +Batch-Norm1d +Relu ⇒ [128,1]
8	Average Pooling Layer [128,1]			
9	MLP (128,64) ⇒ MLP (64,35) [35,1]			
10	Output [35,1]			

Table 1: Parameter sets for the structures of 4 different CNN models (asrCNN1 , asrCNN2, asrCNN3, and asrCNN4).

Traditional Input Transformations.

For all the models, we initially converted the input audio signal into a 1D tensor, with each value representing the magnitude of the audio at a specific time period. Table 1 shows the block diagram of the proto-type CNN models used in this paper. However, for the first two models (asrCNN1 and asrCNN2), we added an additional Fourier transform layer with 64 intervals evenly distributed in mel-scale frequency and a hop length of 160. This trans-

formation converted the input audio data into a two-dimensional spectrogram. We chose these numbers because we believed that, for smaller-scale models, 64 distinct frequencies would suffice to capture the features of the 35 words.

End-to-End Speech Recognition. A Fourier Transform-based Mel-Scale Spectrogram is theoretically just another way of extracting features from data, albeit with fixed parameters. Hence, we utilized a 1D convolution layer to introduce more flexibility to the network. This approach enabled the entire recognition system to be based purely on neural networks. Another advantage of using CNNs for data transformation is the ability to rapidly reduce the data size. The convolution layer had a window length of 320 and a stride of 160, significantly reducing the output length from 16,000 to around 100 (with possible zero padding). This reduction is especially useful for smaller models, as they do not need to extract as many high-level features as larger models. We hypothesized that a convolution with learnable parameters would yield better results than traditional Fourier transform-based signal processing. This paper evaluates four different models, asrCNN1, asrCNN2, asrCNN3, and asrCNN4, as shown in Table 1. Among these models, asrCNN1 and asrCNN2 are processed with torchaudio's MelSpectrogram transformation, which converts a 1D input waveform into a 2D spectrogram output on a mel-scale. The structures of the four different models (asrCNN1, asrCNN2, asrCNN3, and asrCNN4) involve transforming a 1-second audio (sample rate = 16,000) into a 1x16,000 tensor. MelSpectrogram[1,64,320,160] indicates a transformation function with a window size of 320, a hop length of 160, and an output of 64 channels from an input of 1 channel. The notation (128,256,4,2) represents a 1D convolution layer that takes 128 channels as input and outputs 256 channels. For each input channel, the window size is 4, and the hop length is 2. The notation [256,10] denotes an output shape with 256 channels, each with a length of 10. Similarly, [128,1] represents the use of an average pooling layer, resulting in a final output of 128 channels with a length of 1. Finally, MLP(128,64) and MLP(64,35) indicate that the model passes through two layers of MLP to reduce its size from 128 to 64 and finally to 35, which represents the output channels corresponding to the probabilities of the 35 words. asrCNN1 is a smaller-sized model (2.22MB), while asrCNN2 is a larger-sized model (2.79MB). The same applies to asrCNN3 and asrCNN4, with asrCNN3 (2.26MB) being a smaller-sized model and asrCNN4 (2.87MB) being a larger-sized model (Figure 3).

Larger Scale-Models. The larger models, asrCNN2 and asrCNN4, were used to compare their accuracy with the smaller models. These larger-scale models are similar to the smaller-scale models, with the addition of an extra convolution layer beneath the layer that has the maximum number of feature maps (256). This addition aims to maximize the analysis of the extracted features.

Validation and Testing. We trained our models for 30 epochs, with a validation dataset test conducted every 2 epochs. The final models for asrCNN1, asrCNN2, asrCNN3, and asrCNN4 were selected based on the highest accuracy achieved on the validation datasets throughout the 30 epochs of training.

4 Results and Analysis

Figure 3 presents a comparison of the accuracies achieved on the Testing, Validation, and Training Datasets. It is expected that the training dataset would exhibit higher accuracy, typically ranging from 5% to 10%, compared to the testing and validation datasets. However, all four models demonstrated consistent final accuracies ranging from 75% to 90% when evaluated on a separate one-time testing dataset. This indicates that convolutional neural networks (CNNs) remain a viable approach for speech recognition, possibly due to their ability to capture interdependencies among different segments of the audio within specific window sizes.

An intriguing finding in the results was that both asrCNN3 and asrCNN4, which employed fully convolutional layers, outperformed the traditional Fourier-transformed models asrCNN1 and asrCNN2. The former models exhibited an approximate 10% increase in accuracy compared to the latter (see Figure 3). This outcome further supports the hypothesis that convolutional layers offer greater flexibility than traditional transformations, enabling them to effectively learn from the input data. Another plausible explanation is that traditional Fourier transforms convert sound signals into spectrograms, which are more challenging for humans to interpret and match with specific words, unlike the raw sound wave representation.

Notably, there were no significant differences observed between larger and smaller CNN models.

asrCNN1 achieved a similar accuracy to asrCNN2, while asrCNN3 performed on par with asrCNN4. This suggests an intriguing notion that excessively deep CNN architectures may not necessarily lead to improved accuracy. In fact, such models could potentially introduce long-term dependencies that are error-prone for speech recognition tasks, which primarily require short-term attention. Therefore, for smaller-scale speech recognition programs, a simpler CNN structure may indeed yield better results.

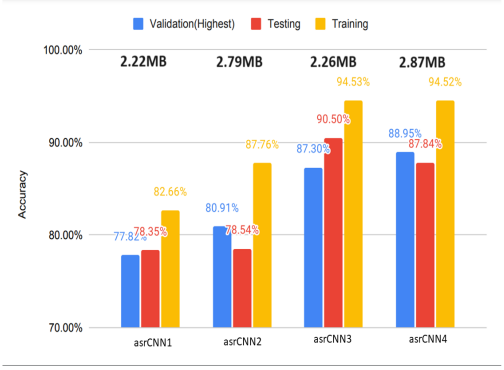

Figure 3: Comparing the highest accuracy in validation datasets with the accuracy in the one-time testing dataset and the accuracy of the training dataset. The sizes of the four experimental models are also shown above.

5 Conclusions

In this paper, we have demonstrated that convolutional neural networks (CNNs) can achieve competitive accuracy for small-vocabulary keyword spotting (KWS) when applied directly to raw audio data. We found that smaller CNN models with only a few convolutional layers were able to match the performance of larger and deeper counterparts. This suggests that excessive model complexity is not required for this audio classification task, reducing concerns about overfitting given the limited training data.

Our key results show that end-to-end CNNs operating on raw waveforms can outperform traditional Fourier transform preprocessing by learning optimal representations tailored to the speech data. This confirms the value of data-driven feature extraction with convolutional layers versus relying on predefined transformations. Additionally, we identified important architecture considerations, including kernel size, stride, model depth, and training procedures, that impact accuracy.

While further improvements to CNN-based KWS are possible, this research provides a strong foundation. Our proposed smaller CNN models offer accurate and efficient speech recognition suitable for embedded applications. Follow-on work could investigate techniques like transfer learning or data augmentation to improve accuracy given limited training data constraints. Overall, CNNs show promise for advancing speech processing capabilities on resource-constrained devices.

References

Lin, Joseph & Lyu, Ren-yuan: https://github.com/JosephtheUnbelievable/Rocling2023

Hoy, Matthew B. “Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants.” Medical Reference Services Quarterly 37 (2018): 81 - 88.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2021). Robust speech recognition via large-scale weak supervision. arXiv preprint arXiv:2106.10947.

Dai, Wei, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. "Very deep convolutional neural networks for raw waveforms." IEEE Signal Processing Letters 23.10 (2016): 1407-1411.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep Learning. Nature, 521, 436-444. http://dx.doi.org/10.1038/nature14539

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Petrov, A., La Malfa, E., Torr, P. H. S., & Bibi, A. (2023). Language model tokenizers introduce unfairness between languages. arXiv preprint arXiv:2305.15425.

Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. arXiv preprint arXiv:1804.03209 (2018).