Abstract

One difficult problem of keyword spotting is how to miniaturize its memory footprint while maintaining high precision. Although conventional neural networks have shown to be effective for the small-footprint keyword spotting problem, they still need hundreds of thousands of parameters to achieve good performance. In this paper, we propose an efficient model based on depthwise separable convolution layers and squeeze-and-excitation blocks. Specifically, we replace the standard convolution by the depthwise separable convolution, which reduces the number of parameters of the standard convolution without significant performance degradation. We further improve the performance of the depthwise separable convolution by reweighting the output feature maps of the first convolution layer with a so-called squeeze-and-excitation block. We compared the proposed method with five representative models on two experimental settings of the Google Speech Commands dataset. Experimental results show that the proposed method achieves the state-of-the-art performance. For example, it achieves a classification error rate of 3.29% with a number of parameters of 72K in the first experiment, which significantly outperforms the comparison methods given a similar model size. It achieves an error rate of 3.97% with a number of parameters of 10K, which is also slightly better than the state-of-the-art comparison method given a similar model size.

Index Terms: keyword spotting, depthwise separable convolution, squeeze-and-excitation block

1. Introduction

Keyword spotting (KWS) aims at detecting predefined keywords in an audio stream. A common approach for KWS is based on large vocabulary continuous speech recognition [1, 2]. It costs huge memory footprint and has a high latency, so that it is often used for the keyword search in large databases. Another approach is based on keyword/filler hidden Markov models (HMMs) [3], which requires a high computational cost and is therefore difficult to be applied to on-device applications. This paper focuses on small-footprint KWS, which requires small memory footprint and low computational power. This kind of technology is able to run on low-resource devices. It provides a fully hands-free way for users to control intelligent devices.

Recently, deep neural network (DNN) based approaches yield significant improvement over the conventional methods in small-footprint KWS. DeepKWS [4] regards keyword spotting as a classification problem and trains a DNN to directly predict the subword units of keywords. It achieves significant improvement over the HMM-based methods, e.g. [3], with small footprint and low computational cost. Because DNN does not consider the local temporal and spectral correlation of speech, Sainath and Parada [5] proposed to replace DNN by convolutional neural network (CNN), which results in better performance with smaller memory footprint than DNN. However, the size of the receptive field of CNN is usually limited, which cannot grasp enough temporal correlation of speech. To overcome this problem, Tang and Lin [6] proposed a residual network (ResNet) based KWS system where they used dilated convolution to enlarge the size of the receptive field exponentially with the depth of the network. However, the ResNet based method still needs several hundreds of thousands of parameters to achieve the state-of-the-art performance. To further reduce the memory footprint, a number of recent works applied time delay neural network (TDNN), attention mechanism, and temporal convolutional network (TCN) to KWS, see e.g. [7, 8, 9]. In [10], Zhang et al. adopted MobileNet [11] that was originally designed for image classification to KWS, where MobileNet reduces the number of parameters and computational costs by a so-called depthwise separable convolution structure [12]. However, if the method uses numerous ReLU activation functions after the convolution operations, the representational ability of the model may be hurt [13], and moreover, the method adopts a conventional convolution structure, which is inefficient in propagating gradients across layers. To summarize, although a number of new architectures have been proposed, they still need a lot of parameters, which does not fully meet the requirement of modern low-resource devices.

Motivated by [6, 10, 13], we propose the depthwise separable convolution based ResNet (DS-ResNet), which is a stack of depthwise separable convolution layers with residual connections. This structure not only improves the representation ability over [10] but also results in smaller memory footprint than [6]. To further improve the performance of the proposed method, we add a squeeze-and-excitation block [14] over the output of the bottom convolutional layer of DS-ResNet. We compared DS-ResNet with ResNet [6], TC-ResNet [9], DS-CNN [10], DenseNet-BiLSTM [15] and tdnn-swsa [16]. Experimental results on the Google speech commands dataset demonstrate that DS-ResNet outperforms the comparison methods in terms of classification errors with fewer parameters than the latter.

The remainder of the paper is organized as follows. Section 2 introduces the proposed DS-ResNet. Section 3 presents the experimental setup and results. Section 4 concludes the paper.

2. Algorithm description

2.1. Network structure

As shown in Figure 1, our entire architecture starts with a standard bias-free convolution layer (Conv) with weights \( W \in \mathbb{R}^{m \times \times n} \), where \( m \) and \( r \) are the height and width of the convolution kernel respectively, and \( n \) is the number of filters (i.e.,
the number of the output channels). The model takes the output of the first convolution layer as the input of a squeeze-and-excitation layer (SE) which is used to reweight the output feature maps. Then, the output of the squeeze-and-excitation layer is the input of a chain of residual blocks, followed by a separate non-residual depthwise separable convolution layer (DS-Conv) which consists of a depthwise convolution layer (depth-Conv) and a pointwise convolution layer (point-Conv). Finally, the output of the model is composed of an average-pooling layer (Avg-pool) followed by a fully-connected softmax layer (Softmax). Additionally, a $(d_h \times d_w)$ convolution dilation was used to increase the receptive field of the depthwise separable convolution layers.

### 2.2. Depthwise separable convolution

Depthwise separable convolution considers the channel realm and space realm separately. It factorizes a standard convolution into two simplified steps. The first step is a spatial feature learning step, named depthwise convolution. The second step is a channel combination step, named pointwise convolution. The most attractive property of the depthwise separable convolution is its low computational cost and small amount of parameters. Before describing the depthwise separable convolution, we first take a look at the computational cost of a standard convolution. As shown in Figure 2(a), given a $C_{in} \times H_{in} \times W_{in}$ input feature maps of a certain convolution layer where $C_{in}$ is the number of the input channels, and $H_{in}$ and $W_{in}$ are the spatial height and width of the input feature maps, a standard convolution operation operates over a joint “space-cross-channels realm” which applies $C_{out}$ filters of size $D_K \times D_K \times C_{in}$ to compute the output feature maps, where $D_K$ is the spatial dimension of the convolution filters, $C_{in}$ is number of the input channels, and $C_{out}$ is the predefined number of the filters (i.e., channels of the output feature maps). The computational cost $C$ and amount of parameters $S$ of the standard convolution are:

$$C_{(Conv)} = C_{in} \times D_K \times D_K \times H_{in} \times W_{in} \times C_{out}$$

and

$$S_{(Conv)} = C_{in} \times D_K \times D_K \times C_{out}$$

respectively, where we have made the same assumption as the standard convolution.

To show the advantage of the depthwise separable convolution over the standard convolution apparently, we give an example as follows. When we set $C_{out} = 64$ and $D_K = 3$, the computational cost and model size of the depthwise separable convolution are only about 1/8 of those of the standard convolution.

### 2.3. Squeeze-and-excitation block

Squeeze-and-excitation block is a new architectural that aims to recalibrate the channel-wise feature responses adaptively by modeling the interdependency between the channels[14]. As illustrated in Figure 3, it consists of two successive operations—squeeze and excitation. The squeeze operation compresses the...
feature maps along the spatial dimension which extracts the mean of the feature maps for each channel. The excitation operation models the correlation between the channels, and then generates a weight for each channel. Finally, the output of the squeeze-and-excitation block is generated by multiplying the input feature maps of the block with the output weights of the excitation operation.

In our implementation, the squeeze operation is an average-pooling layer. The excitation operation consists of two fully-connected layers that take the rectified linear units and sigmoid activation units as the hidden units respectively. The dimension between the two fully-connected layers can be adjusted by a hyperparameter $\alpha$. We set $\alpha = 2^{-3}$ in this paper as [14] did.

### 2.4. Model implementation

In this subsection, we configure the models for the experiment. The first model, named DS-ResNet18, achieves the highest accuracy with a small model size. It consists of 7 residual blocks, each of which contains 2 depthwise separable convolution layers and 64 input and output channels. Because there is an independent depthwise separable convolution layer before the average-pooling layer, the total number of the depthwise separable convolution layers is 15. The dilation in the $i$th layer was set to $2^{\lceil i/2 \rceil}$. DS-ResNet18 has roughly 72K parameters. It needs 285M multiplication operations to generate an output from an input time-frequency spectrum. The details of the model are listed in Table 1.

To reduce the model footprint, the most efficient way is to use fewer filters in each convolution layer. Here we reduced the number of the input and output channels to $n = 32$ for each depthwise separable convolution layer. To further reduce the model footprint, we further reduced the number of the depthwise separable convolution layers to 11. When the number of the depthwise separable convolution layers is less than 13, the receptive field of the entire network cannot cover the entire input. To cover the entire input with a large enough receptive field, we added a $(2 \times 2)$ average-pooling layer after the squeeze-and-excitation layer. The model, named DS-ResNet14, has roughly 15.2K parameters, and needs 15.7M multiplies to generate an output. The details of the model are listed in Table 2.

The smallest model we have implemented contains 8 convolution layers. When the number of the convolution layer is less than 10, the residual connections seem unnecessary anymore. Therefore, we removed the residual connections. To keep the receptive field wide enough, we added a $(4 \times 2)$ average-pooling layer after the squeeze-and-excitation layer. This compact model, named DS-ResNet10, consists of 7 separable convolution layers without residual connections. It has about 10K parameters, and needs 5.8M multiplies to generate an output. The details of the model are listed in Table 3.

### 3. Experiments

#### 3.1. Experimental setup

We evaluated the proposed models using Google's Speech Commands Dataset version 1 [17]. The dataset consists of 64721 one-second-long recordings of 30 words by thousands of different speakers, as well as background noise samples such as pink noise, white noise, and human-made sounds. Among the 30 words, 10 words (including “yes”, “no”, “up”, “down”, “left”, “right”, “on”, “off”, “stop”, “go”) were used as keywords, and the rest 20 words were used as fillers which were labeled as “unknown”.

We followed the way in [6] to add noise and random shift to each segment. Then, we extracted 40 dimensional Mel-frequency cepstrum coefficient features from each frame with a frame length of 25ms and a frame shift of 10ms. We used the stochastic gradient descent with a momentum of 0.9 as the optimizer of the proposed networks, and added the $L_2$ weight decay of $10^{-3}$ as the regularization. The batch size was set to 100. All models were trained from scratch for roughly 30000 steps. The initial learning rate was set to 0.1, and multiplied by 0.1 for every 10000 steps. The network was evaluated on the validation set for every 1000 steps. The model that achieved the highest accuracy on the validation set was used as the final model. We ran all experiments for five independent times with different random seeds, and reported the average performance.

#### 3.2. Results of the first experiment

In the first experiment, we followed the experimental setup of [6]. Specifically, we additionally added some silent segments which are random noise only. We assigned the silent segments a keyword “silence”. We randomly selected a number of segments from the keyword “unknown”, which keeps the ratio of "silence" and "unknown" to about 10% of the total segments. According to the SHA1-hashed name, the audio files was split into three parts for training, validation, and test, which contain roughly 22000, 2700, and 2700 segments, respectively. We compared the proposed DS-ResNet with ResNet [6], TC-ResNet [9], and DS-CNN [10]. The three models of ResNets are denoted as res15, res15-narrow, and res8-narrow. The three models of TC-ResNets are denoted as TC-ResNet14-1.5, TC-ResNet14, and TC-ResNet8. The three models of DS-CNNs are denoted as...
DS-CNN (L), DS-CNN (M), and DS-CNN (S). See [6, 9, 10] for the description of the aforementioned comparison models. All comparison methods followed the same settings as in [6, 9, 10].

Figure 4 shows the comparison results in terms of the error rate (in the vertical ordinate) and model parameters (in the horizontal ordinate). From the figure, we see that the proposed DS-ResNet yields better performance curve than the comparison methods.

Table 4 lists the comparison results in terms of the error rate, model parameters, and the number of multiplication operations per inference pass. From the table, we see that DS-ResNet18 achieves a relative 21.7% error rate reduction over res15, with its number of parameters being only 1/3 of that of the latter. When the number of the model parameters dramatically. We have compared DS-ResNet with DenesNet-BiLSTM, achieving similar performance with DenesNet-BiLSTM. Finally, DS-ResNet18 achieves similar performance with DenesNet-BiLSTM, with the number of parameters being only 1/3 of the latter.

3.3. Results of the second experiment
To further investigate the effectiveness of the proposed method, we used the standard configuration of the dataset [15, 16], where we used 51088 utterances for training, 6798 utterances for validation, and 6835 utterances for testing. We compared with DenesNet-BiLSTM [15] and tdnn-swsa [16]. All comparison methods followed the same settings as in [15, 16].

Table 6 lists the comparison results. From the table, we see that DS-ResNet10 is competitive with tdnn-swsa in the low-resource condition. If we slightly relaxed the restrictions on the model size, DS-ResNet14 achieves a relative 32.2% error rate reduction over tdnn-swsa. Finally, DS-ResNet18 achieves similar performance with DenesNet-BiLSTM, and the number of parameters being only 1/3 of the latter.

4. Conclusions
In this paper, we have proposed the depthwise separable convolution based ResNet for the small-footprint keyword spotting problem, which contains two novel components. The first component concatenates the depthwise separable convolution with ResNet. This concatenation significantly reduces the number of parameters without suffering performance degradation. The second component applies the squeeze-and-excitation block to the output of the first convolution layer, which is able to further improve the performance without increasing the number of parameters dramatically. We have compared DS-ResNet with 5 referenced methods on two settings of the public available Google Speech Commands dataset. Experimental results show that the proposed DS-ResNet achieves the state-of-the-art performance in various experimental settings.

5. Acknowledgements
This work was supported in part by the National Key Research and Development Program of China under Grant No. 2018AAA00102200, in part by National Science Foundation of China under Grant No. 6176114001, 61831019, 61671381, in part by the Project of the Science, Technology, and Innovation Commission of Shenzhen Municipality under grant No. JCYJ20170815161820095, and in part by the Open Research Project of the State Key Laboratory of Media Convergence and Communication, Communication University of China, China under Grant No. SKLMCC2020K009.

Table 4: Comparison between DS-ResNet and ResNet. The number after "*" is the 95% confidence interval.

<table>
<thead>
<tr>
<th>Error rate</th>
<th>Parameters</th>
<th>Multiplies</th>
</tr>
</thead>
<tbody>
<tr>
<td>res15*[6]</td>
<td>4.20% ± 0.484</td>
<td>238K</td>
</tr>
<tr>
<td>res15-narrow*[6]</td>
<td>6.0% ± 0.516</td>
<td>42K</td>
</tr>
<tr>
<td>res10-narrow*[6]</td>
<td>9.90% ± 0.976</td>
<td>19.9K</td>
</tr>
<tr>
<td>DS-ResNet18</td>
<td>3.29% ± 0.159</td>
<td>72K</td>
</tr>
<tr>
<td>DS-ResNet14</td>
<td>4.12% ± 0.178</td>
<td>15.2K</td>
</tr>
<tr>
<td>DS-ResNet10</td>
<td>4.76% ± 0.365</td>
<td>10K</td>
</tr>
</tbody>
</table>

Table 5: Effect of the squeeze-and-excitation block with different settings.

<table>
<thead>
<tr>
<th>Error rate</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS-ResNet18</td>
<td>3.29% ± 0.195</td>
</tr>
<tr>
<td>DS-ResNet18-n</td>
<td>3.45% ± 0.152</td>
</tr>
<tr>
<td>DS-ResNet18-d</td>
<td>3.54% ± 0.191</td>
</tr>
<tr>
<td>DS-ResNet18-p</td>
<td>3.67% ± 0.147</td>
</tr>
</tbody>
</table>

Table 6: Comparison between DS-ResNet, DenesNet-BiLSTM, and tdnn-swsa.

<table>
<thead>
<tr>
<th>Error rate</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DenesNet-BiLSTM*[15]</td>
<td>2.7%</td>
</tr>
<tr>
<td>tdnn-swsa*[16]</td>
<td>4.19% ± 0.191</td>
</tr>
<tr>
<td>DS-ResNet18</td>
<td>3.22% ± 0.109</td>
</tr>
<tr>
<td>DS-ResNet14</td>
<td>2.84% ± 0.257</td>
</tr>
<tr>
<td>DS-ResNet10</td>
<td>3.97% ± 0.154</td>
</tr>
</tbody>
</table>
6. References


