Compact Residual Learning with Frequency-Based Non-Square Kernels for Small Footprint Keyword Spotting

Muhammad Abulaish, *SMIEEE*
Department of Computer Science
South Asian University
New Delhi, India
abulaish@sau.ac.in

Rahul Gulia
Department of Computer Science
South Asian University
New Delh, India
rahulgg619@gmail.com

Abstract

Enabling voice assistants on small embedded devices requires a keyword spotter with a smaller model size and adequate accuracy. It becomes difficult to achieve a reasonable trade-off between a small footprint and high accuracy. Recent studies have demonstrated that convolution neural networks are also effective in the audio domain. In this paper, taking into account the nature of spectrograms, we propose a compact `ResNet` architecture that uses frequency-based non-square kernels to extract the maximum number of timbral features for keyword spotting. The proposed architecture is approximately three-and-a-half times smaller than a comparable architecture with conventional square kernels. On the Google's speech command dataset v1, it outperforms both Google's convolution neural networks and the equivalent `ResNet` architecture with square kernels. By implementing non-square kernels for spectrogram-related data, we can achieve a significant increase in accuracy with relatively few parameters, as compared to the conventional square kernels that are the default choice for every problem.

1 Introduction

Keyword detection systems (KWS) are implemented on embedded or mobile devices to detect predefined keywords in an audio stream. These words can function as *wake words* or *trigger words* for intelligent voice assistants (e.g., *Hey Siri*, *Alexa*, or *Okay Google*) or as simple speech commands (e.g., *yes*, *no*, *on*, *stop*, etc.). Due to the nature of deployment, these systems must have a reasonable compromise between a small footprint and high accuracy. However, implementing a fast, compact, and highly accurate KWS model that can be deployed on embedded or mobile devices with limited hardware and computation is a significant challenge.

Recent studies have demonstrated that convolution neural networks (CNNs) perform well in the audio domain as well. CNNs are predominantly utilized for image-related tasks. CNN's lowest layers typically learn to detect edges. These edges can be oriented in any way. However, one cannot predict which kernel will acquire a given feature. Also, since filter dimensions have a spatial meaning, square kernels are widely used to preserve symmetry. It is common practice in the audio domain to transform an audio stream into a spectrogram in which the X-axis represents time and the Y-axis represents frequency. Then, these spectrograms are fed to two-dimensional CNNs as input.

Spectrograms are a visual representation of the audio intensity over time at various frequencies present in a specific waveform. On spectrograms, the X-axis represents time and the Y-axis represents frequency. The images are then sent to CNN, which performs the feature extractions. The composition of spectrograms is known in advance, i.e., time is represented on the X-axis and frequency on the Y-axis. In order to capture frequency-based and time-based characteristics, customized kernel designs can be implemented. A number of attempts have been made to use custom rectangular kernels for speech emotion detection and music rhythm classification (Pons et al., 2016; Badshah et al., 2019).

In this paper, we propose a compact `ResNet` architecture, i.e., CNNs with residual learning, that uses frequency-based non-square kernels to capture the maximum number of timbral features for keyword detection. Typically, the timbral features are taller than they are wide. Also, because a 3×1 matrix has fewer parameters than a 3×3 matrix, we experimented with non-square kernels for the KWS use-case, with the goal of reducing the number of parameters while maintaining decent accuracy. As a result, it is worthwhile to employ non-square kernels. Figure 1 depicts an example non-square frequency-based kernel. Our architecture is a modification of the res8 model (Tang and

Proceedings of the 19th International Conference on Natural Language Processing (ICON), pages 328 - 336

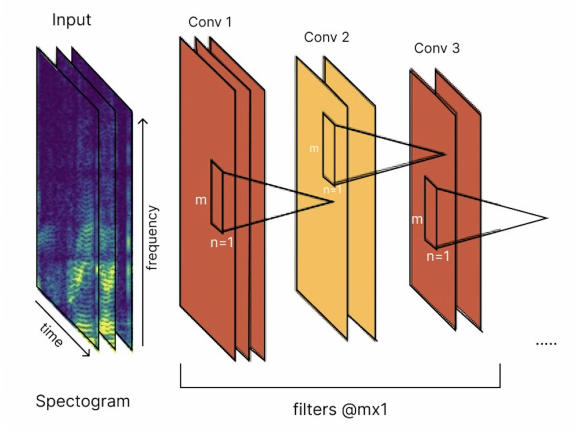

Figure 1: Non-square frequency-based $m \times 1$ kernels

Lin, 2018) with non-square kernels, and it is approximately 3.5 times more compact than the res8. In terms of number of parameters, our architecture also outperforms Google's best CNN (Sainath and Parada, 2015). Compared to the res8 model, the proposed architecture achieves higher evaluation performance and has a smaller footprint.

The remainder of the article is structured as follows. Brief summaries of recent and pertinent keyword spotting research are presented in Section 2. The functional details of our suggested approach for the keyword spotting use-case are presented in Section 3. The experimental setup and results are presented in Section 4. It also provides a comparative analysis of our proposed approach with respect to the two SOTA models, Tang and Lin (2018) and Sainath and Parada (2015), for keyword spotting use-case. Finally, the paper concludes with a discussion of future research directions in Section 5.

2 Related Works

Traditionally, hidden Markov models (HMMs) based approaches were used for KWS (Wilpon et al., 1990; Rose and Paul, 1990). These models were challenging to train, were computationally expensive, and had relatively long latency during inference. Some other techniques used recurrent neural networks (RNNs) like in (Fernández et al., 2007), but they suffered from high latency. After that, Chen et al. (2014) proposed deep neural networks (DNNs) with rectified linear unit (ReLU) activation functions that outperformed HMM-based models with low latency. But the drawback with DNNs is that they ignores the audio's local temporal and spectral correlations. To capture these correlations, variations of CNN-based KWS were introduced . They achieved better results with reduced footprints. Combining the strength of CNNs and RNNs, Arik et al. (2017) experimented with the convolution recurrent neural network-based KWS systems.

In August 2017, the Google's speech command dataset by Warden (2018) was released as a benchmarking dataset for evaluating KWS systems. Warden (2018) also released a baseline model based on the convolution architecture of Sainath and Parada (2015), achieving 85.4% accuracy in v1 version of the dataset. The related Kaggle competition was also organized, where the winner achieved 91% accuracy on the v1 version at that time. Publication of the Google's speech command dataset led to an acceleration in the research. Here, we will discuss the most relevant work as per the paper's objective.

He et al. (2016) (`ResNets`) significantly advance deep learning, and hence they have also been adopted in various audio tasks like automatic speech recognition (Xiong et al., 2018) and speaker identification (Yun et al., 2019). Tang and Lin (2018) further experimented with compact residual architecture using dialated convolution to enlarge the size of the receptive field exponentially with the depth of the network, resulting in improved accuracy. They also experimented with model width by decreasing number of filters. Szegedy et al. (2016) proposed several enhancements to the inception network (Szegedy et al., 2015). They replaced several convolutions with lower dimension convolutions to decrease the parameters. For example, a 7x7 convolution was replaced with four layers by using 1x7 and 7x1 convolutions twice. To reduce model footprint, a number of recent works like time delay neural network (TDNN), attention mechanism, and temporal convolutional network (TCN) are done (Sun et al., 2017; Shan et al., 2018; Choi et al., 2019).

There have been some attempts to use rectangular kernels for speech emotion detection and music rhythm classification (Pons et al., 2016; Badshah et al., 2019). The authors of (Hoogeboom et al., 2018) have used hexagonal kernels that utilized symmetry equivariance and in-variance of images. Our paper uses 2D convolution with non-square kernels ($m \times 1$) convoluted in the frequency domain to capture the maximum amount of timbral features, reducing the computation and number of operations.

This study focuses on the family of CNN mod-

els, as they continue to serve as the benchmark for KWS systems. For KWS systems, we desire an analysis between square and non-square kernels. Since CNNs have a simple architecture, we have conducted our experiments with CNN and residual blocks. In order to perform a more accurate benchmarking, we have referred to the results that were published in (Tang and Lin, 2018).

3 Proposed `ResNet` Architecture with non-square kernel

In this section, the functional details of our proposed `ResNet` architecture with non-square kernels are described. Figure 2 provides a visual representation of the architectural layout of the proposed `ResNet` with non-square kernel.

3.1 Feature Extraction and Preprocessing

Table 1: Parameters used for res8-3x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	3	1	45	36.4K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	43K

Table 2: Parameters used for res8-5x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	5	1	45	60.7K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	67.3K

Table 3: Parameters used for res8-7x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	7	1	45	85K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	91.6K

The input audio stream is converted into mel-scaled spectrograms with a length of $FFTwindow$ as 2048 and 512 samples between successive frames. The spectrogram is then converted to decibels (dB), with the highest dB being 80. We use the `Librosa` Python library to perform the conversion. Spectrogram data is then normalized, standardized, and converted to three dimensions by repeating the matrix along all three axes $[X, X, X]$. The spectrogram images are then scaled to 128×64 pixels. Figure 3 illustrates a sample of the generated spectrogram images.

Table 4: Parameters used for res8-9x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	9	1	45	109K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	115.9K

3.2 Proposed Architecture

Our network architecture contains residual blocks, as described in (He et al., 2016), in which it is proposed that it is simpler to learn residuals:

$$H(x) = F(x) + x$$

than the actual mapping $F(x)$ for a model with greater depth.

Instead of using small squared CNN kernels (e.g., 3×3 or 5×5), we use non-square kernels of $m \times 1$ with varying values of m because such kernels may be able to learn more timbral features than standard square kernels. From an audio standpoint, these kernels are expected to learn fundamental audio features such as frequency, pitch, and timbre, among others. Such kernels may be incapable of learning more about the time axis, i.e., rhythmic or tempo-related features. Taking into account the keyword spotting use-case, however, these kernels reduce the number of parameters and memory footprint, motivating us to move in this direction.

Our first layer is a bias-free $m \times n$ 2D convolution kernel, where m and n are, respectively, the height and width (m = 9, n = 5). Here, n is greater than 1 for the first layer to increase the receptive field, and in subsequent layers, the time axis also contributes. This layer has a stride of 2×2 which helps to reduce the size of the model. After the first layer, we added a 3×4 average pooling layer to reduce the input dimensions (3×4 kernel size yielded better results than 4×3 in our setup). Our residual block is comprised of a bias-free $m \times 1$ 2D convolution kernel, an activation function, and a batch normalization layer. Since the neuron cannot determine its own firing pattern, an activation function is used to determine whether or not a neuron should be activated. Before passing the input to the subsequent layer, the non-linear transformations

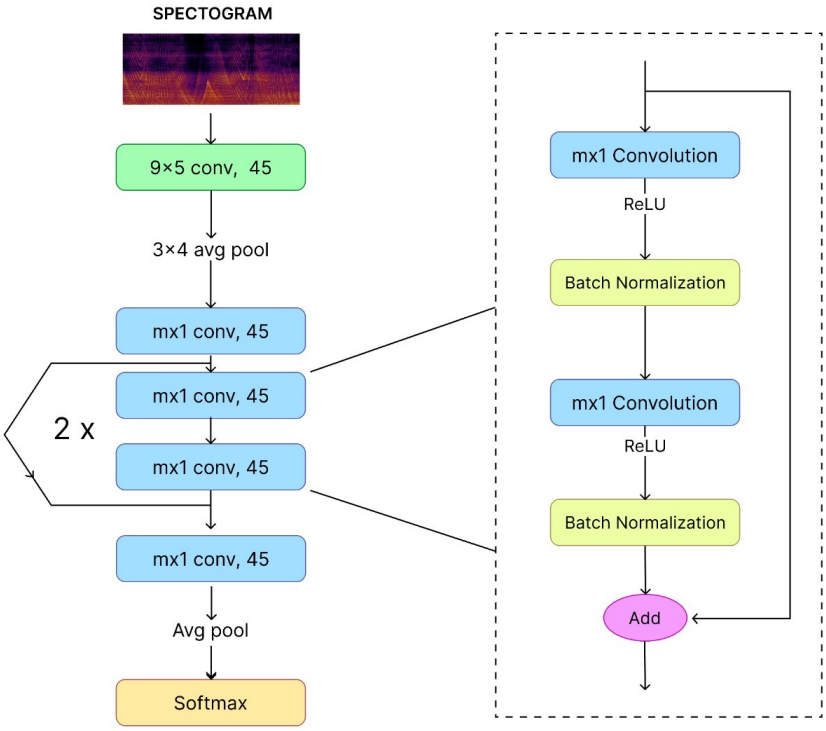

Figure 2: Architecture of the proposed `ResNet` with non-square kernel ($m \times 1$)

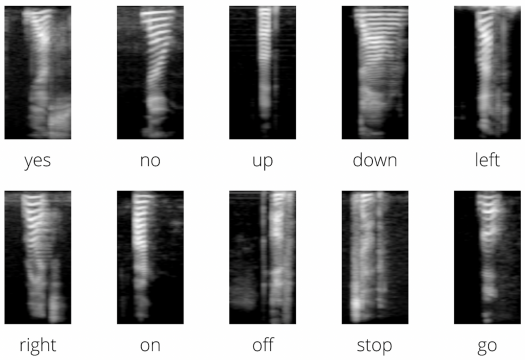

Figure 3: Spectrogram images of different classes

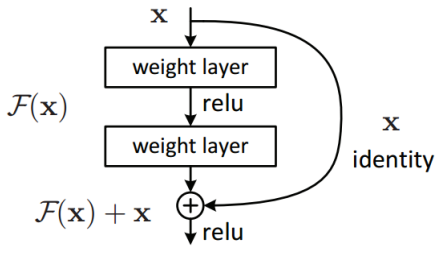

Figure 4: A residual block proposed in (He et al., 2016)

are applied. We have used the `ReLU` activation function, which outputs the input directly if it is positive and 0 if it is negative. Formally, it is defined as follows:

$$Relu(z) = max(0, z)$$

We added a chain of six residual blocks. In the end, we included batch normalization and a non-residual convolution layer. Every mini-batch's weights are normalized via batch normalization. Consequently, it stabilizes the network and drastically reduces the number of training epochs necessary to train deep neural networks. Following the addition of a linear layer, the output then passes through a `softmax` layer that generates a class probability distribution (see figure 2).

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K$$

$N = 45$ feature maps are utilized across all convolution layers. We tested four `ResNet` variants by varying m in the residual block: res8-3x1 with 43K parameters (Table 1), res8-5x1 with 68K parameters (Table 2), res8-7x1 with 92K parameters (Table 3), and res8-9x1 with 115K parameters (Table 4).

4 Experimental Setup and Results

This section describes the experimental design and results. It also provides a comparative analysis of our proposed method with two SOTA models, Tang and Lin (2018) and Sainath and Parada (2015), for keyword spotting use-case.

4.1 Dataset

We use Google's speech commands dataset v1 by Warden (2018) for training and benchmarking purposes for our proposed network that was released in August 2017 under a Creative Commons license. The dataset includes approximately 64727 one-second long utterances of 30 short words, sampled at 16k Hz and recorded by different individuals. The distribution of words in the dataset is depicted in Table 5.

In accordance with Google's implementation, we categorized the audios into 12 classes – *yes*, *no*, *up*, *down*, *left*, *right*, *on*, *off*, *stop*, *go*, *unknown* (remaining words), and *silence* (no speech detected). The blog post announcing the dataset mentions Google's `TensorFlow` implementation of Sainath and Parada (2015) models, which are used for comparison alongside Residual networks proposed by Tang and Lin (2018). Following the publication, we compared the results of the experiments to the v1 test data. Based on the Warden (2018), the dataset is divided into 80% training set, 10% validation set, and 10% test set. This resulted in approximately 23000 examples for training and 2700 for validation and testing.

Table 5: Word distribution in Google's speech command dataset v1

Word	#Utterances	Word	#Utterances	Word	#Utterances
bed	1,713	house	1,750	sheila	1,734
bird	1,731	left	2,353	six	2,369
cat	1,733	marvin	1,746	stop	2,380
dog	1,746	nine	2,364	three	2,356
down	2,359	no	2,375	tree	1,733
eight	2,352	off	2,357	two	2,373
five	2,357	on	2,367	up	2,375
four	2,372	one	2,370	wow	1,745
go	2,372	right	2,367	yes	2,377
happy	1,742	seven	2,377	zero	2,376

4.2 Evaluation Metrics

The proposed method is evaluated based on its accuracy, which is formally defined using True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values in the following equation.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP is defined in the preceding equation as the test result that correctly indicates the presence of a condition. FP is defined as a test result that incorrectly indicates the presence of a specific condition. The test result that correctly indicates the absence of a condition is defined as TN. Finally, FN is defined as a test result that incorrectly indicates the presence of a specific condition.

In addition to accuracy, we have considered the footprint size of the proposed method in terms of the number of parameters, which is calculated using the following formula:

$$Parameters = (fs * pf + 1) * N$$

In the above equation, *fs* represents the $mtimesn$ dimensions of the kernels used; *pf* represents the number of kernels used in the previous layer; and *N* represents the number of kernels used in the current layer. As the bias term, 1 is added to the previous equation.

4.3 Model Training

Using the `PyTorch` framework, we trained and evaluated various models. We used `AdamW` (Loshchilov and Hutter, 2017) as our optimizer with a learning rate of 3e-4 on a mini-batch of 64 samples with 0.001 weight decay at each layer except *LayerNorm* and *Bias*. For the LR scheduler, *ReduceLROnPlateau* is selected as the learning rate scheduler that reads the metric quantity and reduces the learning rate by a certain factor (0.8 in our case) if no improvement is observed for *patience* number of epochs (*patience* is set to 2 in our configuration). We utilized down-sampling to address class imbalance, if any. To prevent the occurrence of over-fitting, we also utilized early stopping on the validation loss with a *patience* of 5. As our loss function, we employ cross entropy, which is defined as follows, wherein K is the number of classes, y is a binary indicator to check whether the class label c is the correct classification for observation o, and p is the predicted probability of observation o for class c.

$$CrossEntropyLoss = -\sum_{c=1}^{K} y_{o,c} \log(p_{o,c})$$

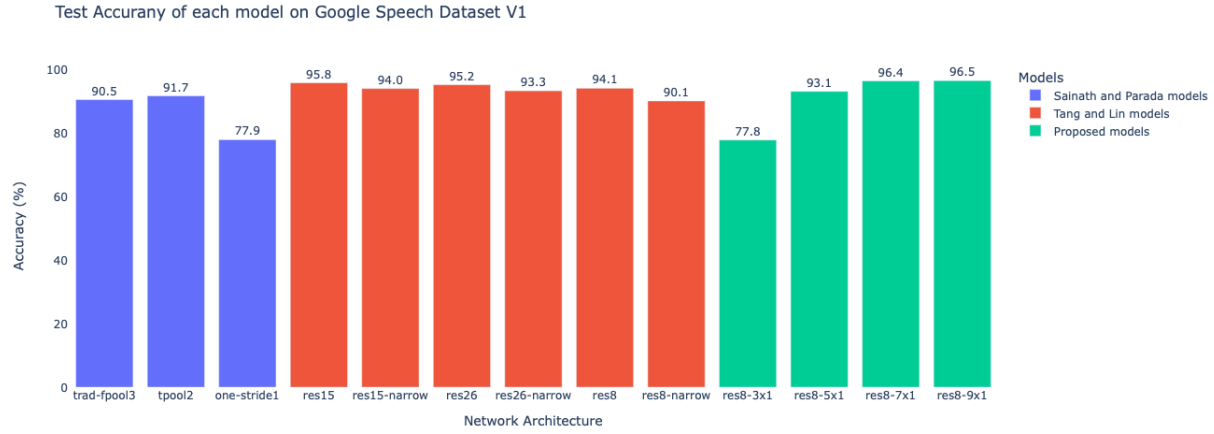

Figure 5: Visualization of accuracy of all models on Google's speech command dataset v1

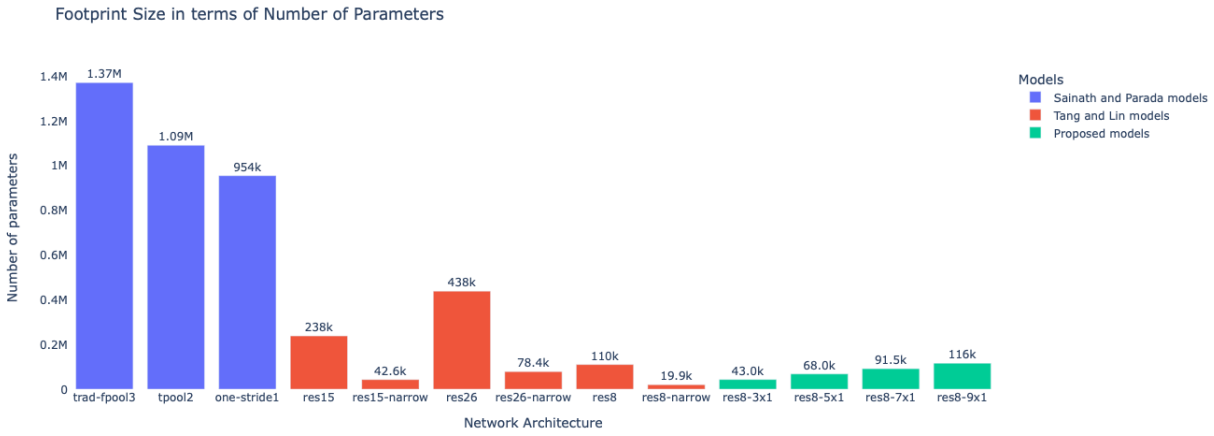

Figure 6: Visualization of footprint size of all models in terms of number of parameters

4.4 Results

For benchmarking, we employ the CNN variants proposed by Sainath and Parada (2015), namely *trad-fpool3*, *tpool2*, and *one-stride1*. In addition, we compared our findings to the compact `ResNet` models proposed in the (Tang and Lin, 2018). In consideration of the small footprint keyword spotting use-case, we employ *no. of parameters* and *model size (MB)* in addition to an evaluation metric for benchmarking purposes. Our proposed method yields comparable results with few parameters. As demonstrated in the table above, as m increases, the model is able to capture more timbral features, resulting in a certain degree of accuracy improvement.

Table 6: Test accuracy of our proposed models in terms of accuracy and number of parameters

Proposed Model	Accuracy	#Parameters
res8-3x1	77.8%	43K
res8-5x1	93.1%	68K
res8-7x1	96.4%	91.6K
res8-9x1	96.5%	115.9K

4.5 Comparative Analysis

In this section, we present the results of a comparative analysis between our proposed `ResNet` model and the Tang and Lin (2018) and Sainath and Parada (2015) models. These models are briefly described in the following paragraphs.

1. Sainath and Parada (2015) models
   - trad-fpool3: It is their base model. It consists of 2 convolution, 1 linear layer,

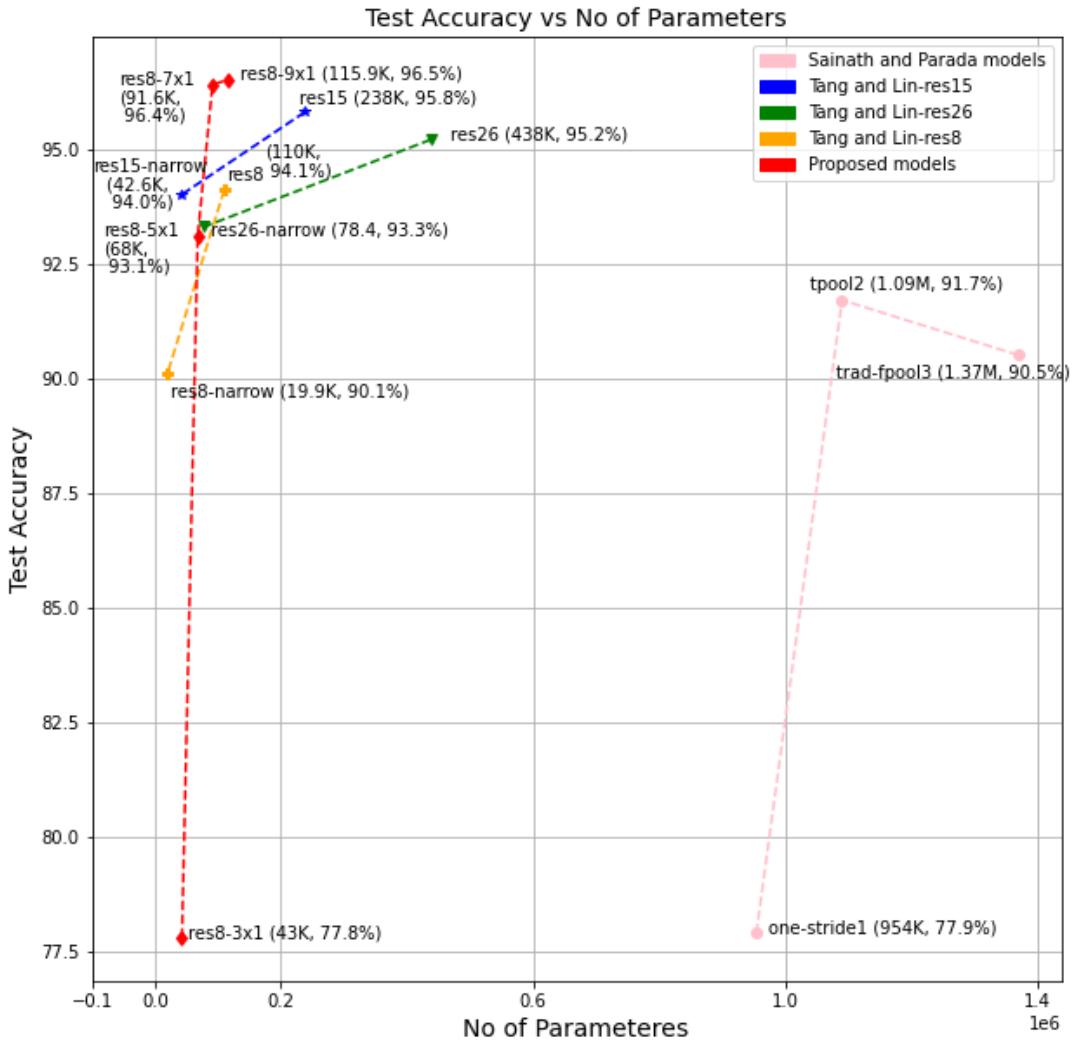

Figure 7: Number of parameters vs. accuracy over Google's speech command dataset v1 (see Table 7)

Table 7: Comparative analysis of our proposed model with other models

Models		Accuracy	#Parameters
Sainath and Parada (2015) models	trad-fpool3	90.5%	1.37M
	tpool2	91.7%	1.09M
	one-stride1	77.9%	954K
Tang and Lin (2018) models	res15	95.8%	238K
	res15-narrow	94.0%	42.6K
	res26	95.2%	438K
	res26-narrow	93.3%	78.4K
	res8	94.1%	110K
	res8-narrow	90.1%	19.9K
Proposed models	res8-3x1	77.8%	43K
	res8-5x1	93.1%	68K
	res8-7x1	96.4%	91.6K
	res8-9x1	96.5%	115.9K

hidden and softmax layer with poolng in frequency axis.

– tpool2: The most accurate variant they explored. It's the variant of the base model with pooling in time axis.

– one-stride1: Their best compact variant is the variant of the base model with stride in frequency axis.

2. Tang and Lin (2018) models

– res15: `ResNet` model with 15 layers and 45 kernels

– res15-narrow: `ResNet` model with 15 layers and 19 kernels

– res26: `ResNet` model with 26 layers and 45 kernels

– res26-narrow: `ResNet` model with 26 layers and 19 kernels

– res8: `ResNet` model with 8 layers and 45 kernels

– res8-narrow: `ResNet` model with 8 layers and 19 kernels

Except for res8-3x1, all of our proposed models (see Table 6) outperform Sainath and Parada (2015)

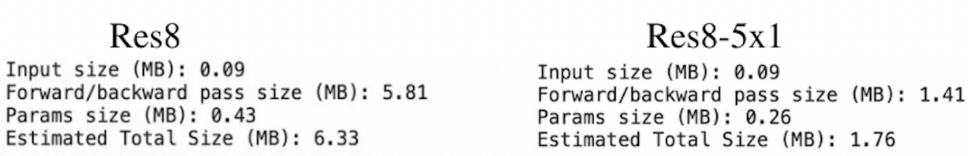

Figure 8: Model size (in terms of MB) comparison between models with square kernel (left) and non-square kernel (right)

in terms of accuracy (Figure 5) and number of parameters (Figure 6), as shown in Table 7. Our res8-5x1 provides slightly less accuracy than the compact `ResNet` models proposed in (Tang and Lin, 2018), but with a much smaller number of parameters.

With only 91.6K parameters, the res8-7x1 model that employs a 7×1 filter size achieves the optimal balance between accuracy and number of parameters, outperforming all models proposed in (Sainath and Parada, 2015) and (Tang and Lin, 2018). Using non-square kernels has been shown to provide adequate and, in some cases, superior accuracy with a small number of audio domain parameters.

Figure 6 illustrates the footprint size in terms of the number of parameters for each model considered for benchmarking. Except for the narrow models by Tang and Lin (2018), all of our proposed models have a smaller number of parameters and comparable accuracy when compared to the other models used for benchmarking. According to their paper, when comparing narrow versus wide models, the number of kernels has a greater effect on accuracy than model depth. Our emphasis is on employing non-square kernels for audio domain in order to reduce model footprint. We employ the same number of kernels ($N = 45$) as the res8 model from (Tang and Lin, 2018).

Compared to models with squared kernels, models with non-squared kernels have a tendency to have a smaller number of parameters with comparable or better accuracy. In addition, we use the same input size for models with a square (res8) kernel and a non-square (res8-5x1) kernel. Figure 8 demonstrates that res8-5x1 is approximately 3.5 times smaller than the res8 model.

5 Conclusion and Future Work

In this paper, we have presented a fast, small footprint model for real-time KWS with non-square kernels ($m \times 1$) that may be useful for small embedded devices. Depending on the characteristics of the spectrogram, non-square kernels may be a suitable alternative to square kernels, provided that non-square kernels have fewer trainable parameters than square kernels. Experiments indicate that non-square kernels reduce model size by reducing the number of required parameters without degrading performance. The frequency-based kernels have few parameters, and the proposed architecture demonstrates satisfactory performance during the evaluation phase. Our proposed work may inspire other researchers and developers to experiment with network architecture based on the dataset and use-case. After comprehending the structure of our data, i.e., spectrogram, we have utilized non-square kernels in the audio domain for KWS. It might be worthwhile to consider conducting additional research on benchmarking different architectures using non-square kernels in the future.

References

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

Abdul Malik Badshah, Nasir Rahim, Noor Ullah, Jamil Ahmad, Khan Muhammad, Mi Young Lee, Soonil Kwon, and Sung Wook Baik. 2019. Deep features-based speech emotion recognition for smart affective services. *Multimedia Tools and Applications*, 78(5):5571–5589.

Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091. IEEE.

Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha. 2019. Temporal convolution for real-time keyword spotting on mobile devices. *arXiv preprint arXiv:1904.03814*.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2007. An application of recurrent neural networks to discriminative keyword spotting. In *International Conference on Artificial Neural Networks*, pages 220–229. Springer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. 2018. Hexaconv. *arXiv preprint arXiv:1803.02108*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Jordi Pons, Thomas Lidy, and Xavier Serra. 2016. Experimenting with musically motivated convolutional neural networks. In *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6.

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

Tara Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Interspeech*.

Changhao Shan, Junbo Zhang, Yujun Wang, and Lei Xie. 2018. Attention-based end-to-end models for small-footprint keyword spotting. *arXiv preprint arXiv:1803.10916*.

Ming Sun, David Snyder, Yixin Gao, Varun K. Nagaraja, Mike Rodehorst, Sankaran Panchapagesan, Nikko Strom, Spyridon Matsoukas, and Shiv Naga Prasad Vitaladevuni. 2017. Compressed time delay neural network for small-footprint keyword spotting. In *INTERSPEECH*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Raphael Tang and Jimmy Lin. 2018. Deep residual learning for small-footprint keyword spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5484–5488. IEEE.

Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

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

Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. 2018. The microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE.

Sungrack Yun, Janghoon Cho, Jungyun Eum, Wonil Chang, and Kyuwoong Hwang. 2019. An end-to-end text-independent speaker verification framework with a keyword adversarial network. *arXiv preprint arXiv:1908.02612*.