INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Multi-scale Convolution for Robust Keyword Spotting

Chen Yang[1], *Xue Wen*[1], *Liming Song*[1]

[1]Samsung Research China-Beijing(SRC-B)

c95.yang@samsung.com, xue.wen@samsung.com, lm0518.song@samsung.com

Abstract

We propose a robust small-footprint keyword spotting system for resource-constrained devices. Small footprint is achieved by the use of depthwise-separable convolutions in a ResNet framework. Noise robustness is achieved with a multi-scale ensemble of classifiers: each classifier is specialized for a different view of the input, while the whole ensemble remains compact in size by heavy parameter sharing. Extensive experiments on public Google Command dataset demonstrate the effectiveness of our proposed method.

Index Terms: keyword spotting, small footprint, residual neural network, multi-scale classifier

1. Introduction

In speech technology, keyword spotting (KWS) refers to the detection of pre-defined keywords from incoming audio. Recent market boom of voice-enabled smart phones and speakers, etc., has brought renewed interest in KWS in the role of the wake-up engine: that switches on the full-scale voice interface using a spoken keyword. Given its special position in the system, weak performance of KWS can have strong negative impact on user experience of the voice interface.

Designing an on-device KWS system is always a trade-off between performance and resource allowance. Small resource footprint is desired as KWS must remain operational at all times, which in turn limits what performance can be achieved. Besides, mobile devices are often associated with diverse environments and speaking styles, bringing additional robustness challenges.

Traditionally KWS systems are often setup under the keyword/filler model [1,2]. These train time series models, like the hidden Markov model (HMM), to characterize keywords and fillers, then decode audio inputs on the learned model to make decisions. In the deep learning era, deep neural networks (DNN) have been applied to KWS and achieved state-of-the-art performance [3,4,5,6,7,8,9,10]. In [3], DNNs were treated as word classifiers, outputting posteriors over pre-defined keyword set. [4] applied convolutional neural networks (CNN) to replace DNNs. [5] and [6] improved CNN using residual connections and depthwise-separable designs. [7] proposed an end-to-end solution with joint CNNs and RNNs. [8] and [9] replaced 2-D convolutions in CNNs with 1-D convolutions. [8] and [10] used attention mechanism and graph convolution to capture global features. [10] further used 1x1 convolutions for model compression. Many recent works feature compact designs for on-device scenarios.

In this work we focus on designing robust, small-footprint KWS. Our contributions are in two folds: first we add yet another small-footprint design to the portfolio by applying depthwise-separable convolution ([6][10]) to a recent state-of-the-art keyword-spotting ResNet, reducing the network size by 4x~20x at comparable accuracy; second we follow the idea of multi-scaling classification [11][12] to design an ensemble of classifier "heads" that pool features at different time scales, intervals and reception-field widths, improving robustness of KWS against noise and speech rate variations. Effectiveness of these methods is verified by extensive experiments on the Google Command dataset [13].

Section 2 describes details of our proposed methods. Section 3 gives our experiment setup, results and analyses. Conclusion and future work are discussed in Section 4.

2. Method

We consider a setup of spotting multiple short keywords from a small pre-defined set. Outputs are produced over one-second segments. Each segment contains at most one keyword. The task is to output the correct keyword ID if the segment contains a keyword, or "non_keyword" if not.

2.1. Preprocessing and feature extraction

Fixed-length audio segments are extracted from continuous audio stream with 1-sec windows at 0.5sec intervals. Each segment goes through a 20Hz/4kHz band-pass filter, followed by voice activity detection (VAD) to test voice presence. Those passing VAD as speech material are split into 30ms frames at 10ms frame shift. 40-dimension MFCC features (MFCC13+Δ+ΔΔ+energy) are computed for all frames and stacked into a 2-D image. This is send to our KWS system as input.

2.2. System overview

Our KWS system is shown in Figure 1. At the centre is a deep residual CNN (ResNet) following [5], which we call depthwise-separable residual neural network (DRN). It takes the 2-D feature image from the front-end as input, and computes a distribution over the set of keywords plus two non-keywords *silence* and *filler*, understood as the probability of the observed input containing each keyword, is silence, or otherwise. Post handling summarizes the outputs (e.g. over segments) to make a final decision.

The ResNet is made up of a stack of residue units (layers). Shallow layers (close to the input) use small receptive fields and compute localized features. Deep layers use large receptive fields and compute complex, far-reaching features. Residue connections help passing information between layers and ease training of very deep networks. Convolution kernels make up the bulk of the ResNet. In [5] the network uses 3×3 conv kernels and has ~250k parameters in total, which is above our preferred budget.

2.2.1. Depthwise-separable convolutions

Depthwise-separable convolution [14] has become a staple design for slimming down conv layers. The idea is to replace a

http://dx.doi.org/10.21437/Interspeech.2020-2185

large cross-channel convolution with a small (typically 1×1) cross-channel convolution followed by large per-channel convolutions. The parameter saving comes from ignoring synapses between neurons not aligned by channel or position. Further reduction is available by low-rank projection: we use 1×1 layers to project the number of channels down or up, so that large per-channel convolutions are computed on fewer channels.

In this paper we replace conv layers in 3×3 residual units by depthwise-separable convolution with low-rank projection, named as depthwise-separable residual unit (DRU) in Figure 2. Incoming n-channel feature map is down-projected to $n/2$ channels by 1×1 convolution, processed per-channel by 3×3, up-projected to n channels by another 1×1, then summed with original input to complete the residual unit. Each of the $n/2$ channels has its own $3 \times 3 \times 1 \times 1$ per-channel kernel, which is $\Omega(n)$ saving compared to one $3 \times 3 \times n \times n$ cross-channel kernel. For a concrete example, one of our DRUs takes 16-channel input ($n = 16$) so the DRU has $n^2/2 \times 2 + 3^2 \times n/2 = 328$ parameters, compared to original residue unit with $3^2 n^2 = 2304$.

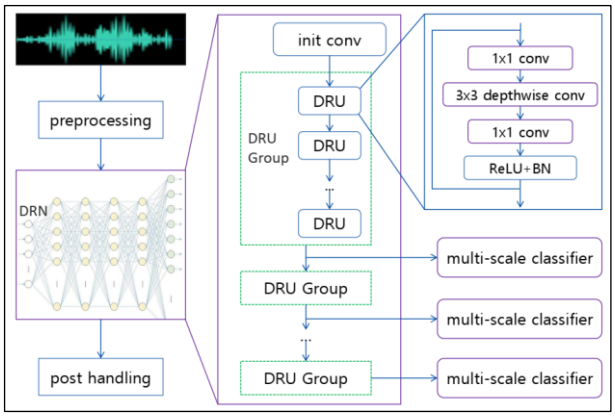

Figure 1: *System Overview.*

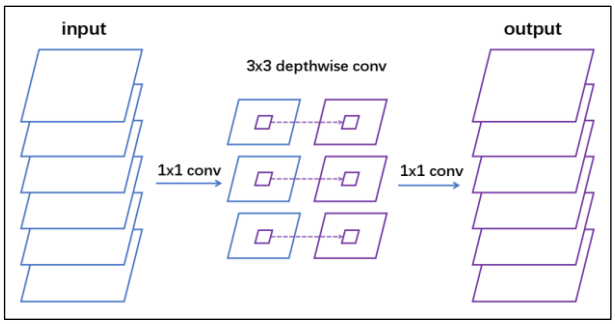

Figure 2: *Principle of model compression in DRU.*

2.2.2. *Multi-scale ensemble*

DRN computes a large set of activations with different localities and depths. Any subset of activations constitutes a *view* of the input. While most DNNs take the *top* view for final prediction, in this paper we consider an ensemble of multiple views. In particular, we define a view as a tuple (d, t, w), where d is a feature depth in the DRN, t is a position in time, w is a duration at t. In short, each view spans a specific time interval of a specific network layer.

Our motivation is illustrated in Figure 3. While the DRN always takes a one-second segment as input, the actual keyword is probably shorter and appears over an unknown interval within that second. Capturing this "oracle" interval can help minimize interference from non-overlapping noise, as well as improve feature consistency between different speech rates. We also conjecture that different noise conditions and speech rates may also have different preferred feature depths (=receptive field sizes), hence the entry d. Since our primary motivation is the actual keyword duration, we call it a multi-scale ensemble.

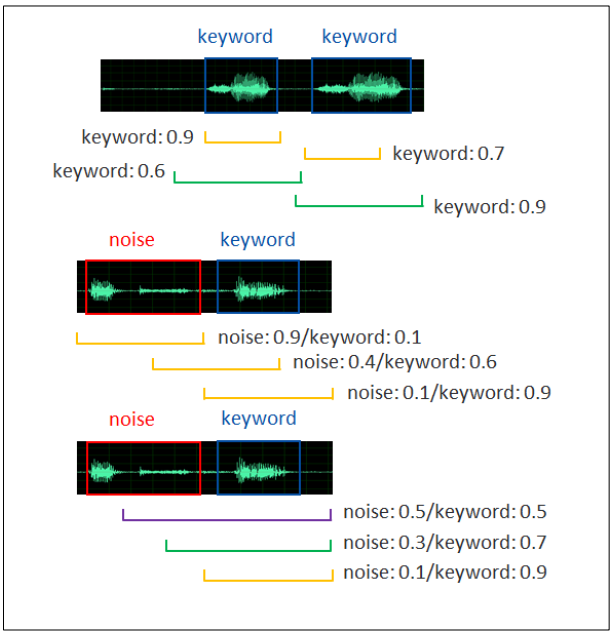

Figure 3: *Multi-scale ensemble under different speaking rates and noisy environments*

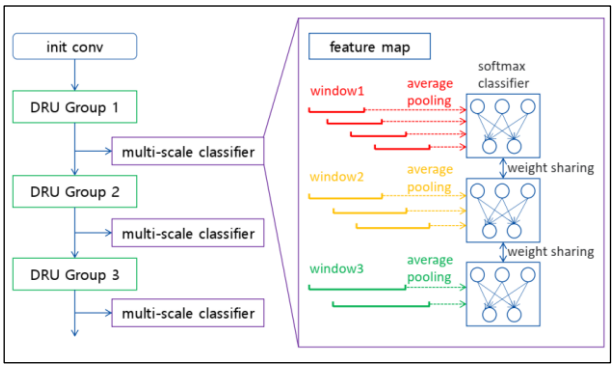

Figure 4: *Multi-scale ensemble*

Architecture of our ensemble is shown in Figure 4. For each (d, t, w), we use a classification head that average-pools feature map from layer d over interval $I(t, w)$, then predicts the keyword ID from the pooled feature via softmax. Parameters of this softmax classifier is shared by all heads pooling the same layer. This keeps the ensemble compact, makes the computation parallelizable, and encourages learning features that are robust against speech rate changes. The output of head (d, t, w) is a distribution $p_{d,t,w}(v)$ over keywords (including non-keywords). The DRN and ensemble of classification heads are trained end-to-end under a naïve cross-entropy objective

$$naiveCE := \sum_{d,t,w} \log p_{d,t,w}(y), \quad (1)$$

so called because it ignores dependency between views. y is the ground truth label independent of the view, even if the pooled receptive fields have no overlap with actual keyword. Sharing a learnable underlying network differs from the traditional concept of independently-trained ensembles, but is crucial for keeping the model size in check.

2.3. Implementation details

Our base implementation of DRN is denoted as DRN10, as shown in Table 1. It has 10 conv layers in total, including one initial convolution and 9 stacked DRUs. Initial convolution reduces input size and projects features into multiple channels. The 9 DRUs are arranged in three groups. DRUs in the same group have the same number of channels. Deeper groups are allocated more channels to learn more complex features. Inside each DRU, the number of channels is halved by a 1x1 convolution and later restored by another. The size of softmax output is task-specific. In Table 1, k_w and k_h refers to width and height of convolution kernel, respectively. c and c' refers to channel number before and after 1x1 low-rank projection.

Table 1: *Architecture of DRN10.*

type	k_w	k_h	c	c'	#*param*	#*mul*
init-conv	9	4	16	-	576	86.4K
DRU×3	3	3	16	8	1112	147.6K
DRU×3	3	3	32	16	3760	525.6K
DRU×3	3	3	48	24	7560	1.1M
avg-pool	-	-	48	-	-	-
Softmax	-	-	12	-	576	576
Total	-	-	-	-	13.6K	1.95M

Apart from DRN10, we also design and test two variants of DRN which we call DRN7 and DRN13. Their structures are given in Table 2 and Table 3, respectively.

Table 2: *Architecture of DRN7.*

type	k_w	k_h	c	c'	#*param*	#*mul*
init-conv	9	4	16	-	576	86.4K
DRU×2	3	3	16	8	784	117.6K
DRU×2	3	3	32	16	2592	388.8K
DRU×2	3	3	48	24	5040	756.0K
avg-pool	-	-	48	-	-	-
Softmax	-	-	12	-	576	576
Total	-	-	-	-	9.6K	1.35M

Table 3: *Architecture of DRN13.*

type	k_w	k_h	c	c'	#*param*	#*mul*
init-conv	9	4	32	-	1152	172.8K
DRU×4	3	3	32	16	5184	777.6K
DRU×4	3	3	64	32	18.6K	2.8M
DRU×4	3	3	96	48	38.6K	5.8M
avg-pool	-	-	96	-	-	-
Softmax	-	-	12	-	576	576
Total	-	-	-	-	64.0K	9.52M

We choose the multi-scale views (d, t, w) as follows. d is chosen as the last layer of each DRU group (e.g. layers 4, 7, 10 in DRN10). Let the time dimension of the feature map at depth d be T. Time scale w is chosen from $T/3, T/2, 2T/3$. For each w we choose $t = 0, T/6, \ldots, T - w$ and average-pool the feature map between t and $t + w$. This yields 36 different views in total. The whole ensemble produces $36(|\mathcal{V}| + 2)$ scalar outputs, where $|\mathcal{V}|$ is the set of the keyword set.

Table 4 summarizes the model size and time complexity of the three DRN variants we consider. It's easy to see that the multi-scale ensemble brings only marginal parameter and computation overheads.

Table 4: *Model size and complexity of DRN variants with multi-scale ensemble (msc).*

model	#*param*	#*mul*
DRN7 + msc	10.7K	1.35M
DRN10 + msc	14.7K	1.95M
DRN13 + msc	66.4K	9.53M

At test time we pick the highest of all $36|\mathcal{V}|$ ensemble outputs corresponding to keywords. This is understood as choosing the view that produces the most confident top-1 keyword prediction. If the value is above a pre-fixed threshold then we output the corresponding keyword; otherwise we output "non_keyword":

$$result = \begin{cases} \arg\max_{v \in \mathcal{V}} \left(\max_{d,t,w} p_{d,t,w}(v) \right), & \text{if } \exists p_{d,t,w}(v) > th, \\ \text{non\_keyword}, & \text{otherwise} \end{cases}, \quad (2)$$

3. Experiments and results

3.1. Experimental setup

We use publicly available Google Command dataset [13] for model training and testing. This dataset consists of utterances of 30 different commands, spoken by hundreds of different speakers. Duration of a majority of the utterances is one second. We select all one-second utterances and extract 40-dimensional feature frames with the HTK toolkit. We choose 10 commands "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go" as keywords and all other 20 as *filler*. Background noise is associated with the *silence* class. We split the chosen data into training set, evaluation set and testing set, at percentages of 80%, 10% and 10%, respectively. The split follows subset lists in the original dataset to avoid overlap of speakers.

Training and testing are implemented in TensorFlow. All parameters are randomly initialized, and trained with ADAM using initial learning rate 0.04 and batch size 64. The whole training process runs on a K80 GPU with 4GB memory and converges after 100 epochs.

3.2. Results

We benchmark our multi-scale DRN against CNN [4], ResNet [5], DS-CNN [6] and latest state-of-the-art CENet [10]. We treat all misclassifications as keywords as false positives and all misclassifications as *silence* or *filler* as false negatives. For metric we use the false positive rates (FAR, A for *alarm*) and false negative rate (FRR, R for *rejection*), which are two metrics commonly used for performance evaluation.

Here, we set FAR to fixed 1% and compare FRRs of all models. Detailed results are given in Table 5, where DRNs are fitted with the usual top-view classifier (without multi-scale ensemble).

Table 5: *Performance of DRN variants and baselines.*

model	#*param*	#*mul*	FRR
trad-fpool3 [4]	1.73M	125M	9.5%
tpool2 [4]	1.09M	103M	8.3%
one-stride1 [4]	954K	5.76M	22.1%
res15 [5]	238K	894M	4.2%
res8 [5]	110K	30M	5.9%
res15-narrow [5]	42.6K	160M	6.0%
res8-narrow [5]	19.9K	5.65M	9.9%
DS-CNN-S [6]	38.6K	5.4M	5.6%
DS-CNN-M [6]	189.2K	19.8M	5.1%
DS-CNN-L [6]	497.6K	56.9M	4.6%
CENet-6 [10]	16.2K	1.95M	6.1%
CENet-24 [10]	44.3K	8.51M	4.4%
CENet-40 [10]	61K	16.18M	3.6%
DRN7	9.6K	1.35M	9.8%
DRN10	13.6K	1.95M	5.9%
DRN13	64.0K	9.52M	3.2%

DRN7 achieves comparable performance to res8-narrow, which uses nearly 2x more parameters and 3x more multiplications. DRN10 outperforms CENet-6 which has 19% more parameters, and is comparable to DS-CNN-S which is 4x as expensive in both size and multiplications. DRN13 outperforms all baseline models with similar size and 2x fewer multiplications than CENet-40. These show the effectiveness of depthwise-separable convolution on ResNets for KWS.

Table 6: *Effectiveness of multi-scale ensemble (msc).*

model	FRR	FRR(noise)	FRR(fast)
DRN7	9.8%	21.1%	15.8%
DRN7 + msc	7.2%	19.3%	12.5%
DRN10	5.9%	18.5%	13.5%
DRN10 + msc	4.4%	16.2%	10.8%
DRN13	3.2%	10.8%	8.7%
DRN13 + msc	2.8%	9.9%	6.1%

The effectiveness of the multi-scale ensemble is evaluated in Table 6. Comparisons are made between different DRN variants with and without multi-scale ensemble classification. In addition to the original test set, we also include an artificial noisy dataset and an artificial fast-speaking test set. The noisy test set is built by adding 3 types of noise (babble, office, car) to the original test set, with random signal-to-noise ratio (SNR) between 5dB to 15dB. The fast-speaking test set is built by 1.2x time stretching.

The results show that the multi-scale ensemble can help decrease FRR for all variants on all test sets. Surprisingly, improvements on the noisy test sets fall behind those on the original dataset. We speculate this were related to domain shift as noisy data had not been seen during training. Improvements achieved on fast speech are comparable to those seen on the original test set, in line with our conjecture that multiscaling helps grabbing the actual time span of keywords.

3.3. A visualization

Figure 5 visualizes feature maps at specific layers for 2 noisy utterances. We observe features of keyword (highlight regions) surrounded by noise features (light-color regions), at both time and feature dimension. Pooling with appropriate intervals can help to avoid interference of features of non-overlapping noise at time dimension, thus conduct to better noise robustness. However, features of overlapping noise cannot be eliminated by our multi-scale ensemble. That's why FRR improvements are not so promising on noisy test sets.

Figure 6 visualizes feature maps for a fast-speaking keyword at different layers pooled by the multi-scale ensemble: the one on top from a shallow layer, the other from a deep layer. We observe compact, localized features from the shallow layer and a more diffused map from the deep layer, as expected. Pooling from the shallow layer can therefore provide more meaningful localization of the keyword, even though both layers pool at the same set of intervals.

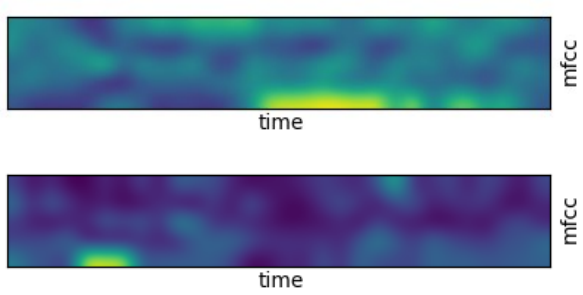

Figure 5: *Input feature maps for noisy utterances.*

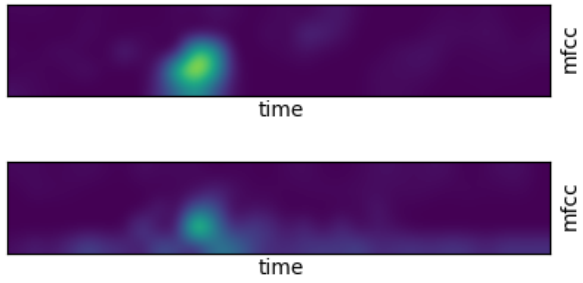

Figure 6: *Input feature maps at different layers for a fast-speaking utterance.*

4. Conclusions

In this paper, we propose a depthwise-separable residual neural network and a multi-scale classifier ensemble for KWS: the former for reducing model size and computation, the latter improves robustness of performance. Some of our models are among the smallest to achieve state-of-the-art performance on the Google Command dataset.

For future work, we plan to explore other DRN variants and multi-view setups, including adaptive and lightweight multi-view designs. In-depth investigation is needed to explain the behavior of multi-view approach on unseen noise, as well as the extent to which it can capture the "right" keyword interval before being distracted towards alternatives. Understanding such mechanisms can help design better-principled yet still compact KWS architectures and training recipes.

5. References

[1] M.Wu, S.Panchapagesan, M.Sun, J.Gu, R.Thomas, S.N.P.Vitaladevuni, B.Hoffmeister and A.Mandal, "Monophone-based background for two-stage on-device wake word detection,"

in ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 5494-5498.
[2] F.Ge and Y.Yan, "Deep neural network based wake-up-word speech recognition with two-stage detection," in ICASSP 2017 - 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2761-2765.
[3] G.Chen, C.Parada and G.Heigold, "Small-footprint keyword spotting using deep neural networks," in ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 4087-4091.
[4] T.N.Sainath and C.Parada, "Convolutional neural networks for small-footprint keyword spotting," in Proc.InterSpeech 2015, 2015, pp. 1478-1482.
[5] R.Tang and J.Lin, "Deep residual learning for small-footprint keyword spotting" in ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 5484-5488.
[6] Y.Zhang, N.Suda, L.Lai and V.Chandra, "Hello edge: Keyword spotting on microcontrollers," arXiv preprint arXiv: 1711.07128, 2017.
[7] 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.
[8] Y.Bai, J.Yi, J.Tao, Z.Wen, Z.Tian, C.Zhao and C.Fan, "A time delay neural network with shared weight self-attention for small-footprint keyword spotting," in INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association, September 15-19, Graz, Austria, Proceedings, 2019, pp. 2190-2194.
[9] S.Choi, S.Seo, B.Shin, H.Byun, M.Kersner, B.Kim, D.Kim and S.Ha, "Temporal convolution for real-time keyword spotting on mobile devices," in INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association, September 15-19, Graz, Austria, Proceedings, 2019, pp. 3372-3376.
[10] X.Chen, S.Yin, D.Song, P.Ouyang, L.Liu and S.Wei, "Small-footprint keyword spotting with graph convolutional network," arXiv preprint arXiv: 1912.05124, 2019.
[11] S.Ren, K.He, R.Grishick and J.Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, " arXiv preprint arXiv: 1506.01497, 2015.
[12] W.Liu, D.Anguelov, D.Erhan, C.Szegedy, S.Reed, C.Fu and A.C.Berg, "SSD: Single Shot MultiBox Detector," arXiv preprint arXiv: 1512.02325, 2015.
[13] P.Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," arXiv preprint arXiv: 1804.03209, 2018.
[14] F. Chollet, “XCeption: deep learning with depthwise separable convolutions,” in *Proc. CVPR'17*, Honolulu, 2017