Activate or Not: Learning Customized Activation

Ningning Ma[1] Xiangyu Zhang[2] Ming Liu[1] Jian Sun[2]
[1] The Hong Kong University of Science and Technology
[2]MEGVII Technology
{nmaac,eelium}@ust.hk {zhangxiangyu,sunjian}@megvii.com

Abstract

We present a simple, effective, and general activation function we term `ACON` which learns to activate the neurons or not. Interestingly, we find Swish, the recent popular NAS-searched activation, can be interpreted as a smooth approximation to ReLU. Intuitively, in the same way, we approximate the more general Maxout family to our novel `ACON` family, which remarkably improves the performance and makes Swish a special case of `ACON`. Next, we present `meta-ACON`, which ***explicitly*** *learns to optimize the parameter switching between non-linear (activate) and linear (inactivate) and provides a new design space. By simply changing the activation function, we show its effectiveness on both small models and highly optimized large models (e.g. it improves the ImageNet top-1 accuracy rate by 6.7% and 1.8% on MobileNet-0.25 and ResNet-152, respectively). Moreover, our novel `ACON` can be naturally transferred to object detection and semantic segmentation, showing that `ACON` is an effective alternative in a variety of tasks. Code is available at https://github.com/nmaac/acon.*

1. Introduction

The Rectified Linear Unit (ReLU) [13, 24, 39] has become an effective component in neural networks and a foundation of many state-of-the-art computer vision algorithms. Through a sequence of advances, the Swish activation [41] searched by the Neural Architecture Search (NAS) technique achieves top accuracy on the challenging ImageNet benchmark [9, 42]. It has been shown by many practices to ease optimization and achieve better performance [18, 49]. Our goal is to interpret the mechanism behind this searched result and investigate more effective activation functions.

Despite the success of NAS on modern activations, a natural question to ask is: *how does the NAS-searched Swish actually work?* Despite its widespread use, this activation function is still poorly understood. We show that Swish can be surprisingly represented as a smooth approximation to ReLU, by a simple and general *approximation formula* (Equ. 2).

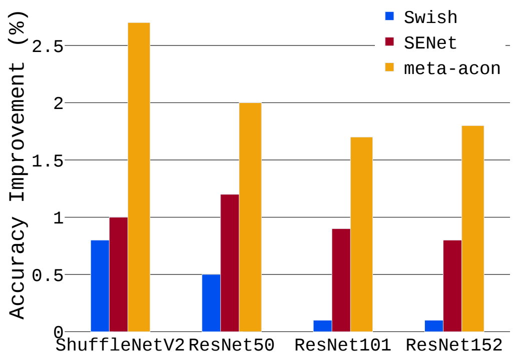

Figure 1. ImageNet top-1 accuracy relative improvements compared with the ReLU baselines. As the models go larger, Swish and SENet gain smaller, but **`Meta-ACON`** still improves very stably and remarkably even on the substantially deep and highly optimized ResNet152. The relative improvements of **`Meta-ACON`** are about twice as much as SENet.

This paper pushes the envelop further: our method, called `ACON`, follows the spirit of the ReLU-Swish conversion and approximates the general Maxout [12] family to our novel `ACON` family by the general *approximation formula*. We show the converted functions (`ACON-A`, `ACON-B`, `ACON-C`) are smooth and differentiable, where Swish is merely a case of them (`ACON-A`). `ACON` is conceptually simple and does not add any computational overhead, however, it improves accuracy remarkably. To achieve this result, we identify the fixed upper/lower bounds in the gradient as the main obstacle impeding from improving accuracy and present the `ACON` with learnable upper/lower bounds (see Fig. 3).

In principle, `ACON` is an extension of Swish and has a dynamic non-linear degree, where a switching factor decays to zero as the non-linear function becomes linear.

[1]This work is supported by The National Key Research and Development Program of China (No. 2017YFA0700800) and Beijing Academy of Artificial Intelligence (BAAI).

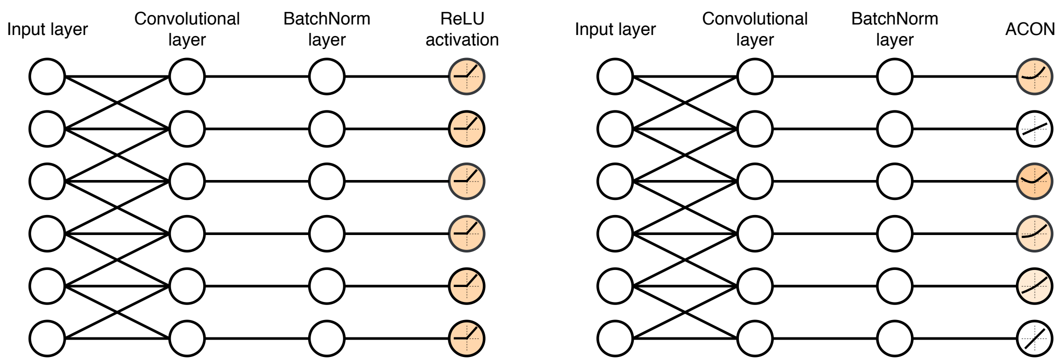

Figure 2. We propose a novel activation function we term the `ACON` that *explicitly* learns to activate the neurons or not. **Left:** A ReLU network; **Right:** An `ACON` network that learns to activate (orange) or not (white).

Intuitively, this switching factor enables `ACON` to switch between activating or not. However, evidence [41] has shown that optimizing the factor simply by using the gradient descent cannot learn to switch between linear and non-linear well. Therefore, we optimize the switching factor *explicitly* for fast learning and present `meta-ACON` that learns to learn whether to activate or not (see Fig. 2). Despite it seems a minor change, `meta-ACON` has a large impact: it has significant improvements on various tasks very stably (even the highly-optimized and extremely deep SENet-154) and provides a new architecture design space in the meta learner, which could be layer-wise, channel-wise, or pixel-wise. The design in the provided space is beyond the focus of this paper, but it is suggestive for future research.

`ACON` transfers well on a wide range of tasks. No matter for small models or large models, our approach surpasses the ReLU counterpart significantly: it improves the ImageNet top-1 accuracy rate by 6.7% and 1.8% on MobileNet-0.25 and ResNet-152, respectively. We show its generality on object detection and semantic segmentation tasks.

We summarize our contributions as follows: (1) We present a novel perspective to understand Swish as a smoothed ReLU; (2) from this valuable perspective, we connect the two seemingly unrelated forms (ReLU and Swish), and smooth ReLU's general Maxout family to Swish's general `ACON` family; (3) we present `meta-ACON` that explicitly learns to activate the neurons or not, improves the performance remarkably.

2. Related Work

Activation functions The Rectified Linear Unit (ReLU) [13, 24, 39] and its variants [37, 15, 7, 35] are the most widely used activation functions in the past few years. ReLU is non-differentiable at zero and is differentiable anywhere else. Many advances followed [27, 45, 1, 17, 40, 11, 55], and softplus [10] is a smooth approximation to the maximum function ReLU based on the LogSumExp function.

The Maxout [12] can be a piecewise linear approximation for arbitrary convex activation functions. It generalizes the leaky ReLU and ReLU and can approximate linear activations. The Maxout is a general formulation of many current activation functions. In this work we present a new family of activation which is a smooth approximation to the Maxout family. For example, the recent searching technique contributes to a new searched scalar activation called Swish [41] by combing a comprehensive set of unary functions and binary functions. Firstly, the searched results show that the form of SiLU [11, 17], $y = x \cdot Sigmoid(x)$, achieves good performance and outperforms other scalar activations on many vision tasks. Secondly, the method also indicates that the form of Swish, $y = x \cdot Sigmoid(\beta x)$, shows great potential. However, there is a lack of proper explanation of the searched Swish formulation. In this paper, we generalize Swish to the `ACON` family, showing that it is a smooth approximation of ReLU based on the well-known smooth conversion called α-$softmax$, which is frequently applied in optimization and neural computation [28, 3, 14].

A recent activation called DY-ReLU [5] encodes the global context into a hyperfunction and adapts the piecewise linear activation function accordingly. The method increases the number of parameters remarkably and improves the performance significantly especially on light-weight models. However, the improvements become smaller when the models go larger and deeper. For example, though DY-ReLU generalizes SENet conceptually, ResNet50-DY-ReLU (only 1.0% improvement, 76.2->77.2) cannot outperform ResNet50-SENet, the improvements on larger models become smaller. Different from DY-ReLU, firstly, `ACON` learns to determine the activation to be linear or non-linear. Secondly, our method has a comparable amount of parameters with

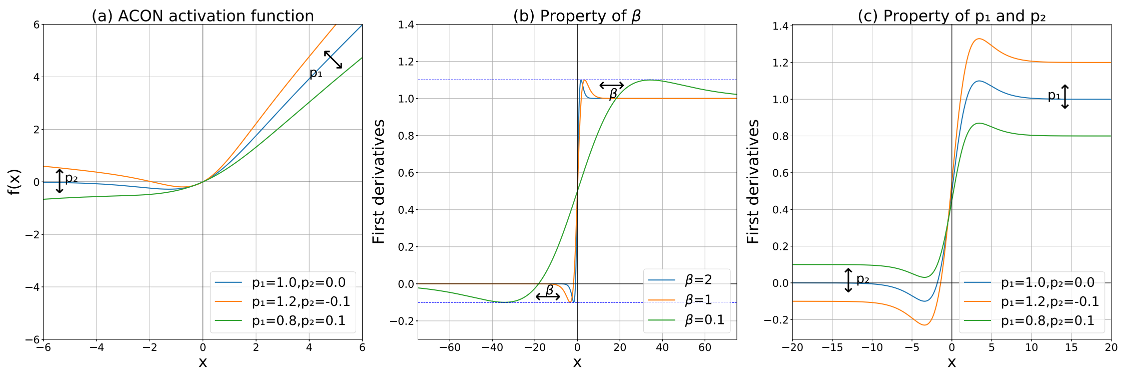

Figure 3. The `ACON` activation function and its first derivatives. (a) The `ACON-C` activation function with fixed β (see Fig. 4 for the influence of β); (b-c) The first derivatives with fixed p_1&p_2 (b) and fixed β (c). β controls how fast the first derivative asymptotes to the upper/lower bounds, which are determined by p_1 and p_2.

ReLU networks. Thirdly, the performance improvement is still very significant even on the very deep and highly optimized ResNet-152, which is more than twice as much as SENet (Δ accuracy = 1.8 v.s. 0.8, see Fig. 1).

Dynamic network Standard CNNs [44, 16, 47, 59, 6, 19, 43, 18] share the same network structure and convolution kernels for all the samples, while conditional (or dynamic) CNNs [29, 32, 54, 58, 26, 34] use dynamic kernels, widths, or depths conditioned on the input samples, obtaining remarkably gains in accuracy.

Some dynamic networks learn the dynamic kernels [56, 34], some use attention-based approaches [50, 33, 2, 51, 53, 20] to change the network structures, another series of work [52, 21] focus on dynamic depths of the convolutional networks, that skip some layers for different samples. In our work, we learn the non-linear degree in the activation function dynamically, which controls to what degree the non-linear layer is.

Neural network design space The design of the neural network architecture mainly includes the kernel level space and the feature level space. The most common feature design space aims to optimize the performance via channel dimension [47, 48, 43, 36, 20], spatial dimension [49, 4, 23] and feature combination [16, 22]. On the recent popular kernel design space we can optimize the kernel shape [46, 57, 25] and kernel computation [56, 34]. In this work, we provide a new design space on the non-linear degree level by customizing the non-linear degree in each layer.

3. ACON

We present **Act**ivate**O**r**N**ot (`ACON`) as a way of learning to activate the neurons or not. In this paper, we first show how we use the general approximation formula: *smooth maximum* [28, 3, 14] to perform the ReLU-Swish [41] conversion. Next, we convert other cases in the general Maxout [12] family, which is thus a natural and intuitive idea and makes Swish a case of `ACON`. At last, `ACON` learns to activate (non-linear) or not (linear) by simply maintaining a switching factor, we introduce `meta-ACON` that learns to optimize the factor explicitly and shows significant improvements.

Smooth maximum We begin by briefly reviewing the smooth maximum function. Consider a standard maximum function $max(x_1, ..., x_n)$ of n values, we have its smooth and differentiable approximation:

$$S_\beta(x_1, ..., x_n) = \frac{\sum_{i=1}^{n} x_i e^{\beta x_i}}{\sum_{i=1}^{n} e^{\beta x_i}} \tag{1}$$

where β is the switching factor: when $\beta \to \infty$, $S_\beta \to$ *max*; when $\beta \to 0$, $S_\beta \to$ *arithmetic mean*.

In neural networks, many common activation functions are in the form of $max(\eta_a(x), \eta_b(x))$ function (e.g. ReLU $max(x, 0)$ and its variants) where $\eta_a(x)$ and $\eta_b(x)$ denote linear functions. Our goal is to approximate the activation functions by this formula. Therefore we consider the case when $n = 2$, we denote σ as the Sigmoid function and the approximation becomes:

$$\begin{aligned}
&S_\beta(\eta_a(x), \eta_b(x)) \\
&= \eta_a(x) \cdot \frac{e^{\beta\eta_a(x)}}{e^{\beta\eta_a(x)} + e^{\beta\eta_b(x)}} + \eta_b(x) \cdot \frac{e^{\beta\eta_b(x)}}{e^{\beta\eta_a(x)} + e^{\beta\eta_b(x)}} \\
&= \eta_a(x) \cdot \frac{1}{1 + e^{-\beta(\eta_a(x) - \eta_b(x))}} + \eta_b(x) \cdot \frac{1}{1 + e^{-\beta(\eta_b(x) - \eta_a(x))}} \\
&= \eta_a(x) \cdot \sigma[\beta(\eta_a(x) - \eta_b(x))] + \eta_b(x) \cdot \sigma[\beta(\eta_b(x) - \eta_a(x))] \\
&= (\eta_a(x) - \eta_b(x)) \cdot \sigma[\beta(\eta_a(x) - \eta_b(x))] + \eta_b(x)
\end{aligned} \tag{2}$$

Table 1. Summary of the Maxout family and ACON family. σ denotes Sigmoid.

		Maxout family	ACON family
$\eta_a(x)$	$\eta_b(x)$	$max(\eta_a(x), \eta_b(x))$	$(\eta_a(x) - \eta_b(x)) \cdot \sigma(\beta(\eta_a(x) - \eta_b(x))) + \eta_b(x)$
x	0	$max(x, 0) : ReLU$	ACON-A$(Swish) : x \cdot \sigma(\beta x)$
x	px	$max(x, px) : PReLU$	ACON-B $: (1-p)x \cdot \sigma(\beta(1-p)x) + px$
$p_1 x$	$p_2 x$	$max(p_1 x, p_2 x)$	ACON-C $: (p_1 - p_2)x \cdot \sigma(\beta(p_1 - p_2)x) + p_2 x$

ACON-A We consider the case of ReLU when $\eta_a(x) = x, \eta_b(x) = 0$, then $f_{\texttt{ACON-A}}(x) = S_\beta(x, 0) = x \cdot \sigma(\beta x)$, which we call ACON-A and is exactly the formulation of Swish [41]. Swish is a recent new activation which is a NAS-searched result, although it is widely used recently, there is a lack of reasonable explanations about why it improves the performance. From the perspective above, we observe that Swish is a smooth approximation to ReLU.

ACON-B Intuitively, based on the approximation we could convert other maximum-based activations in the Maxout family (e.g. Leaky ReLU [37], PReLU [15], etc) to the ACON family. Next, we show the approximation of PReLU. It has an original form $f(x) = max(x, 0) + p \cdot min(x, 0)$, where p is a learnable parameter and initialized as 0.25. However, in most case $p < 1$, under this assumption, we rewrite it to the form: $f(x) = max(x, px)(p < 1)$. Therefore we consider the case when $\eta_a(x) = x, \eta_b(x) = px$ in Equ. 2 and get the following new activation we call ACON-B:

$$f_{\texttt{ACON-B}}(x) = S_\beta(x, px) = (1-p)x \cdot \sigma[\beta(1-p)x] + px \tag{3}$$

ACON-C Intuitively, we present a simple and more general case we term ACON-C. We adopt the same two-argument function, with an additional hyper-parameter. ACON-C follows the spirit of ACON-B that simply uses hyper-parameters scaling on the feature. Formally, let $\eta_a(x) = p_1 x, \eta_b(x) = p_2 x (p_1 \neq p_2)$:

$$f_{\texttt{ACON-C}}(x) = S_\beta(p_1 x, p_2 x) = (p_1 - p_2)x \cdot \sigma[\beta(p_1 - p_2)x] + p_2 x \tag{4}$$

As with PReLU, β, p_1, and p_2 are channel-wise. We init β=p_1=1, p_2=0. Our definition of ACON-C is a very simple and general case (see Fig. 3, Fig. 4). Moreover, there could be many more complicated cases in the Maxout family (e.g. more complicated formulations of $\eta_a(x)$ and $\eta_b(x)$), which are beyond the scope of this paper. We focus on the property of the conversion on this simple form.

Upper/lower bounds in the first derivative. We show that Swish has fixed upper/lower bounds (Fig. 3 b) but our definition of ACON-C allows the gradient has learnable upper/lower bounds (Fig. 3 c). Formally, we compute the first derivative of ACON-C and its limits as follows:

$$\begin{aligned} &\frac{d}{dx}[f_{\texttt{ACON-C}}(x)] \\ =&\frac{(p_1 - p_2)(1 + e^{-\beta(p_1 x - p_2 x)})}{(1 + e^{-\beta(p_1 x - p_2 x)})^2} \\ +&\frac{\beta(p_1 - p_2)^2 e^{-\beta(p_1 x - p_2 x))} x}{(1 + e^{-\beta(p_1 x - p_2 x)})^2} + p_2 \end{aligned} \tag{5}$$

$$\lim_{x \to \infty} \frac{df_{\texttt{ACON-C}}(x)}{dx} = p_1, \quad \lim_{x \to -\infty} \frac{df_{\texttt{ACON-C}}(x)}{dx} = p_2 \;\; (\beta > 0) \tag{6}$$

To compute the upper/lower bounds, which are the maxma/minima values, we compute the second derivative:

$$\begin{aligned} &\frac{d^2}{dx^2}[f_{\texttt{ACON-C}}(x)] \\ =&\beta\left(p_2 - p_1\right)^2 \mathrm{e}^{\beta(p_1 - p_2)x} \cdot \\ &\frac{\left(\left(\beta\left(p_2 - p_1\right)x + 2\right)\mathrm{e}^{\beta(p_1 - p_2)x} + \beta\left(p_1 - p_2\right)x + 2\right)}{\left(\mathrm{e}^{\beta(p_1 - p_2)x} + 1\right)^3} \end{aligned} \tag{7}$$

We set $\frac{d^2}{dx^2}[f_{\texttt{ACON-C}}(x)] = 0$, simplify it, and get $(y - 2)e^y = y + 2$, where $y = (p_1 - p_2)\beta x$. Solving the equation we get $y \approx \pm 2.39936$. Then we get maxima and minima of Equ. 5 when $\beta > 0$:

$$\begin{aligned} maxima(\frac{d}{dx}[f_{\texttt{ACON-C}}(x)]) &\approx 1.0998 p_1 - 0.0998 p_2, \\ minima(\frac{d}{dx}[f_{\texttt{ACON-C}}(x)]) &\approx 1.0998 p_2 - 0.0998 p_1 \end{aligned} \tag{8}$$

This is different from Swish with the fixed upper/lower bounds (1.0998, -0.0998) in the first derivative. In Swish, the hyper-parameter β only determines how fast the first derivative asymptotes to the upper bound and the lower bound, however, the bounds are learnable and determined by p_1 and p_2 in ACON-C (see Fig. 3 c). The learnable boundaries are essential to ease optimization and we show by experiments that these learnable upper/lower bounds are the key for improved results.

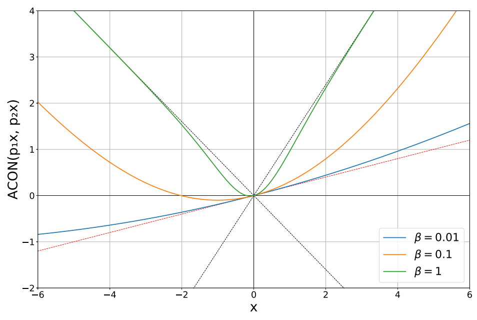

Figure 4. `ACON-C` applied on '$1.2x$' and '$-0.8x$' functions with various values of β, which switches between non-linear and linear (arithmetic mean). Very large β can approximate the *maximum* function (non-linear) and β near zero can approximate the *mean* function (linear) (the red dashed line).

3.1. Meta-ACON

`ACON` switches the activation to activate or not as the switching factor β controls it to be non-linear or linear. Specifically, when $\beta \to \infty, f_{\texttt{ACON-C}}(x) \to max(p_1x, p_2x)$; when $\beta \to 0, f_{\texttt{ACON-C}}(x) \to mean(p_1x, p_2x)$. Thus, unlike the traditional activations such as the ReLU, `ACON` allows each neuron to adaptively activate or not (see Fig. 2). This customized activating behavior helps to improve generalization and transfer performance. This motivated us to develop the following `meta-ACON` that plays a key role in the customized activation.

Our proposed concept is simple: learning the switching factor β explicitly conditioned on the input sample $x \in \mathbb{R}^{C\times H\times W}$: $\beta = G(x)$. We are not aiming to propose a specific structure, we provide a design space in the generating function $G(x)$.

Design space The concept is more important than the specific architecture which can be layer-wise, channel-wise of pixel-wise structure. Our goal is to present some simple designing examples, which manage to obtain significantly improved accuracy and show the importance of this new design space.

We briefly use a routing function to compute β conditioned on input features and present some simple structures. First, the structure can be layer-wise, which means the elements in a layer share the same switching factor. Formally, we have: $\beta = \sigma \sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{c,h,w}$.

Second, we present a simple channel-wise structure, meaning the elements in a channel share the same switching factor. Formally, we show it by: $\beta_c = \sigma W_1 W_2 \sum_{h=1}^{H}\sum_{w=1}^{W} x_{c,h,w}$. We use $W_1 \in \mathbb{R}^{C\times C/r}$, $W_2 \in \mathbb{R}^{C/r\times C}$ to save parameters ($r = 16$ by default).

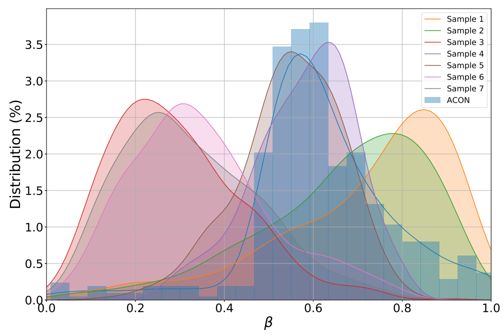

Figure 5. `ACON` v.s. `meta-ACON`. β-distribution of an activation layer in the last bottleneck of the trained ResNet-50 models. We randomly select 7 samples, for the `ACON` network, they share the same β distribution (the blue histogram); while for the `meta-ACON` network, they have 7 distinct β distributions. Lower β values mean more linear, lager β values mean more non-linear.

To further reduce the amount of parameters for large models (Res101, Res152), we find the depth-wise fully-connected layers also achieve good results.

Third, for the pixel-wise structure, all the elements use unique factors. Although there could be many structure designing methods, we simply present an extremely simple structure aiming to present a pixel-wise example. Formally, we have: $\beta_{c,h,w} = \sigma x_{c,h,w}$.

We note our `meta-ACON` has a straightforward structure. For the following `meta-ACON` experiments, we use the channel-wise structure and `ACON-C` unless otherwise noted. More complex designs have the potential to improve performance but are not the focus of this work.

4. Experiment

4.1. Image Classification

We present a thorough experimental comparison on the challenging ImageNet 2012 classification dataset [9, 42] along with comprehensive ablations. For training, we follow the common practice and train all the models using the same input size of 224x224 and report the standard top-1 error rate.

`ACON` We first evaluate our `ACON` method on the light-weight CNNs (MobileNets [19, 43] and ShuffleNetV2 [36]) and deep CNNs (ResNets [16]). For light-weight CNNs we follow the training configure in [36]; for the larger model ResNet, we use a linear decay learning rate schedule from 0.1, a weight decay of 1e-4, a batch size of 256, and 600k iterations. We run numerous experiments to analyze the behavior of the `ACON` activation function, by simply changing all the activations on various network structures and various model sizes.

Table 2. **Comparison of the `meta-ACON` on MobileNets, ShuffleNetV2, and ResNets.** We report the top-1 error on the ImageNet dataset (train and test on 224x224 input size).

	ReLU			`meta-ACON`		
	FLOPs	# Params.	Top-1 err.	FLOPs	# Params.	Top-1 err.
MobileNetV1 0.25	41M	0.5M	47.6	41M	0.6M	**40.9**$_{(+\mathbf{6.7})}$
MobileNetV2 0.17	42M	1.4M	52.6	42M	1.9M	**46.2**$_{(+\mathbf{6.4})}$
ShuffleNetV2 0.5x	41M	1.4M	39.4	41M	1.7M	**34.8**$_{(+\mathbf{4.6})}$
MobileNetV1 0.75	325M	2.6M	30.2	326M	3.1M	**26.4**$_{(+\mathbf{3.8})}$
MobileNetV2 1.0	299M	3.5M	27.9	299M	3.9M	**25.0**$_{(+\mathbf{2.9})}$
ShuffleNetV2 1.5x	301M	3.4M	27.4	304M	6.0M	**24.7**$_{(+\mathbf{2.7})}$
ResNet-18	1.8G	11.7M	30.3	1.8G	11.9M	**28.4**$_{(+\mathbf{1.9})}$
ResNet-50	3.9G	25.5M	24.0	3.9G	25.7M	**22.0**$_{(+\mathbf{2.0})}$
ResNet-101	7.3G	44.1M	22.8	7.3G	44.1M	**21.1**$_{(+\mathbf{1.7})}$
ResNet-152	11.3G	60.0M	22.3	11.3G	60.1M	**20.5**$_{(+\mathbf{1.8})}$

Table 3. **Comprehensive comparison of `ACON` on MobileNetV2, ShuffleNetV2 and ResNets.** We report the top-1 error on the ImageNet dataset (train and test on 224x224 input size).

Model	MobileNetV2		ShuffleNetV2		ResNets		
Channel / depth	0.17	1.0	0.5×	1.5×	ResNet-18	ResNet-50	ResNet-101
FLOPs	42M	300M	41M	300M	1.8G	3.9G	11.3G
ReLU	52.6	27.9	39.4	27.4	30.3	24.0	22.8
`ACON-A` (Swish)	51.4	27.3	38.3	26.8	30.3	23.5	22.7
`ACON-B`	50.8	26.4	38.0	26.8	29.4	23.3	22.3
`ACON-C`	**48.9**	**26.4**	**37.0**	**26.5**	**29.1**	**23.2**	**22.1**

The baseline networks are the ReLU networks and the extra parameters in `ACON` networks are negligible.

We have two major observations from Table 3 and Fig. 3. (i), `ACON-A`, `ACON-B`, and `ACON-C` all improve the accuracy remarkably comparing with their *max-based* functions. This shows the benefits of the differentiable and smooth conversion. (ii), `ACON-C` outperforms `ACON-A` (Swish) and `ACON-B`, benefitting from the adaptive upper/lower bounds in the `ACON-C`'s first derivatives. (iii), Although `ACON-A` (Swish) shows minor improvements as the models go deeper and larger (0.1% on ResNet-101), we still obtain continuous accuracy gains from `ACON-C` (0.7% on ResNet-101).

`Meta-ACON` Next, we evaluate the `meta-ACON` function. For light-weight CNNs we change all the ReLU activations to `meta-ACON`, for deep CNN (ResNet-50, ResNet-101) we change one ReLU (after the 3×3 convolution) in each building block to `meta-ACON` to avoid the overfitting problem.

The results in Table 2 show that we manage to obtain a remarkably performance boost in all the network structures. For light-weight CNNs, `meta-ACON` improves 6.7% on MobileNetV1 0.25 and still has around 3% accuracy gains on 300M level models. For deeper ResNets, `meta-ACON` still shows significant improvements, which are 2.0% and 1.7% on ResNet-50 and ResNet-101.

To reveal the reasons, in Fig. 5 we select a layer in the last bottleneck and compare the learned β distribution in ResNet-50. `ACON` shares the same β distribution for all the different samples across the dataset, however, in `meta-ACON` different samples have distinct non-linear degrees instead of sharing the same non-linear degree in `ACON`. Specifically, some samples tend to have more values close to zero, which means for such samples the network tends to have a lower non-linear degree. While some samples tend to have more values far from zero, meaning the network adaptively learns a higher non-linearity for such samples. This is an intuitively reasonable result as different samples usually have quite different characteristics and properties.

4.2. Ablation Study

We run several ablations to analyze the proposed `ACON` and `meta-ACON` activations.

Comparison with other activations Table 4 show the comparison with more activations besides ReLU and

Table 4. **Comparison with other activations.** We report the ImageNet top-1 error on ShuffleNetV2 0.5x.

Activation	Top-1 err.
ReLU	39.4
Swish [41]	38.3
Mish [38]	39.5
ELU [7]	39.5
SoftPlus[10]	39.6
`ACON-C`	37.0
`meta-ACON`	**34.8**

Table 5. **Design space in `meta-ACON`.** We report the top-1 error on the ImageNet dataset. Comparison on 3 different levels of design space. We give 3 most simple examples on ShuffleNetV2 0.5×. fc denotes fully-connected, σ denotes sigmoid, GAP denotes global average pooling.

	manner	Top-1 err.
baseline	-	39.4
pixel-wise	$\sigma(x)$	37.2
channel-wise	$\sigma(fc[fc[GAP(x)]])$	34.8
layer-wise	$\sigma[\sum_c GAP(x)]$	36.3

Swish, including Mish [38], ELU [7], SoftPlus [10]. We note that recent advances show comparable results comparing with ReLU, except that Swish shows greater improvement (1.1% top-1 error rate). `ACON` and `meta-ACON` manage to improve the accuracy remarkably (2.4% and 4.6%) comparing with the previous activations.

Design space in `meta-ACON` We provide a new architecture design space in `meta-ACON` ($G(x)$ in Sec. 3.1). As the switching factor determines the non-linearity in the activation, we generate β values for each sample on different levels, which could be pixel-wise, channel-wise, and layer-wise. Our goal is to present a wide design space that provides more possibilities in the future neural network design, we are not aiming to propose the most effective specific module in this paper, which is worth more future studies. We investigate the most simple module for each level that is described in Section 3.1. Table 5 shows the comparison on ShuffleNetV2 0.5×. The results show that all three levels could improve accuracy significantly, with more careful design, there could be more effective modules.

Switching factor distribution In `meta-ACON` we adopt a meta-learning module to learn the switching factor β explicitly. Figure 5 shows the distribution of the learned factor in the last activation layer of ResNet-50, we compare `meta-ACON` with `ACON`, and randomly select 7 samples to show the result. The distributions

Table 6. **`Meta-ACON` v.s. SENet** [20]. We report the ImageNet top-1 error rates of ShuffleNetV2 and ResNet. For fair comparison, we only replace one ReLU with `Meta-ACON` in each building block.

	Baseline	SE	`meta-ACON`
ShuffleNetV2 0.5x	39.4	37.5	**34.8**
ShuffleNetV2 1.5x	27.4	26.4	**24.7**
ResNet-50	24.0	22.8	**22.0**
ResNet-101	22.8	21.9	**21.1**
ResNet-152	22.3	21.5	**20.5**

Table 7. **Comparison on the extremely deep SENet-154** [20]. We report the ImageNet top-1 error rates. We implement all the models by ourselves.

Activation	Top-1 err.
ReLU	18.95
`ACON-A` (Swish)	19.02
`ACON-C`	**18.40**

indicate three conclusions: (1) `meta-ACON` learns a more widespread distribution than `ACON`; (2) each sample has its own switching factor instead of sharing the same one; (3) some samples have more values close to zero, meaning some neurons tend not to activate in this layer.

Comparison with SENet We have shown that `ACON` helps improving accuracy remarkably. Next, we compare our channel-wise `meta-ACON` with the effective module SENet [20] on various structures. We conduct a comprehensive comparison of both light-weight CNNs and deep CNNs. Table 6 shows that `meta-ACON` outperforms SENet significantly on all the network structures. We note that it is more difficult to improve accuracy on larger networks because of highly optimized, but we find that even in the extreme deep ResNet-152, `meta-ACON` still improves accuracy by 1.8%, which gains 1% comparing with SENet.

Moreover, we conduct experiments on the highly optimized and extremely large network SENet-154 [20], which is challenging to further improve the accuracy. We re-implement SENet-154 and change the activations to `ACON` under the same experimental environment for fairness comparison. We note that SE together with `ACON-A` or `ACON-C` is a case of channel-wise `meta-ACON` structure, the differences between them are the learnable upper/lower bounds (see Sec.3). Table 7 shows two results: first, simply combing `ACON-A` (Swish) with SENet performs comparable or even worse result comparing to ReLU activation; second, `ACON-C` achieves 18.40 top-1 error rate on the challenging ImageNet dataset, improving the performance remarkably.

Table 8. Evaluation of `ACON-FReLU`, which outperforms existing static networks.

	FLOPs	# Params.	Top-1 err.
0.17 MobileNetV2	42M	1.4M	52.6
ShuffleNetV2 0.5	41M	1.4M	39.4
TFNet 0.5	43M	1.3M	**36.6**
0.6 MobileNetV2	141M	2.2M	33.3
ShuffleNetV2 1.0	146M	2.3M	30.6
TFNet 1.0	135M	1.9M	**29.7**
1.0 MobileNetV2	300M	3.4M	28.0
ShuffleNetV2 1.5	299M	3.5M	27.4
TFNet 1.5	279M	2.7M	**26.0**
1.4 MobileNetV2	585M	5.5M	25.3
ShuffleNetV2 2.0	591M	7.4M	25.0
TFNet 2.0	**474M**	**3.8M**	**24.3**

More complicated activation In the previous sections we show the { `ACON-A`, `ACON-B`, `ACON-C` } activations converted from the general Maxout family. Recently, a more powerful activation FReLU shows its potential for vision tasks. FReLU also belongs to the Maxout family. Next, we evaluate the `ACON-FReLU` by simply modifying $\eta_a(x)$ and $\eta_b(x)$ according to the form of FReLU. FReLU boosts accuracy for light-weight networks. However, since both the FReLU layer and the original blocks contain depth-wise convolutions, directly changing the ReLU functions to FReLU is not optimal because of the redundancy utilization of the depth-wise convolution. Therefore, to evaluate the performance, we design and train a simple toy funnel network (TFNet) made only by *pointwise convolution* and `ACON-FReLU` operators. The simple block is shown in Appendix Fig. 2, the Appendix Table 1 shows the whole network structure. We train the models with a cosine learning rate schedule, the other settings follow the work in [36].

Table 8 shows the comparison to the state-of-the-art static light-weight networks. Although the structure is very simple, the TFNet shows great improvements. Since this structure does not have dynamic modules such as the SE [20] module, we category the TFNet to static networks according to the WeightNet [34]. By carefully adding dynamic modules to the structure we could get an optimal dynamic network [18], which is beyond the focus of this work.

4.3. Generalization

Our novel activation can easily be extended to other tasks, we show its generalization performance by experiments on object detection and semantic segmentation.

COCO object detection We report the standard COCO [31] metrics including AP (averaged over IoU thresholds), AP_{50}, AP_{75}, AP_S, AP_M , AP_L (AP at different scales). We train using the union of 80k train images and a 35k subset of validation images ($trainval35k$) and report results on the remaining 5k validation images ($minival$). We choose the RetinaNet [30] as the detector and use ResNet-50 as the backbone. As a common practice, we use a batch size of 2, a weight decay of 1e-4, and a momentum of 0.9. We use anchors for 3 scales and 3 aspect ratios and use a 600-pixel train and test image scale. To evaluate the results of different activations, we use the ImageNet pre-trained ResNet-50 with different activations as backbones. Table 9 shows the significant improvements comparing with other activations.

Table 9. **Comparison of different activations on the COCO object detection [31] task.** We report results on RetinaNet [30] with ResNet-50 backbones.

	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ReLU	35.2	53.7	37.5	18.8	39.7	48.8
Swish	35.8	54.1	38.7	18.6	40.0	49.4
`meta-ACON`	**36.5**	**55.9**	**38.9**	**19.9**	**40.7**	**50.6**

Table 10. **Comparison of different activations on the CityScape [8] semantic segmentation task.** We report results on PSPNet [60] with ResNet-50 backbones.

Activation	FLOPs	# Params.	mean_IU
ReLU	3.9G	25.5M	77.2
Swish	3.9G	25.5M	77.5
`meta-ACON`	3.9G	25.7M	**78.3**

Semantic segmentation We further present the semantic segmentation results on the CityScape dataset [8]. We use the PSPNet [60] as the segmentation framework and ResNet-50 as the backbone. As a common practice, we use the poly learning rate policy where the base is 0.01 and the power is 0.9, a weight decay of 1e-4, and a batch size of 16. Table 10 shows that our result (78.3) is 1.1 points higher than the ReLU baseline, showing larger improvement than Swish. Given the effectiveness of our method on various tasks, we expect it to be a robust and effective activation for other tasks.

5. Conclusion

In this work, we present the novel `ACON` as a simple but effective activation that learns to activate or not. We show `ACON` family that approximates to the general Maxout family, exploring more functions in the Maxout family is a promising future direction yet beyond the focus of this work. We expect this robust and effective activation applied to a wide range of applications.

References

[1] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014. 2

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 3

[3] Stephen P. Boyd and Lieven Vandenberghe. Convex optimization. *IEEE Transactions on Automatic Control*, 51:1859–1859, 2006. 2, 3

[4] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3

[5] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. *arXiv preprint arXiv:2003.10027*, 2020. 2

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 3

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 2, 7

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 8

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 5

[10] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *NIPS*, 2000. 2, 7

[11] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 2

[12] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013. 1, 2, 3

[13] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000. 1, 2

[14] Simon Haykin. Neural networks: A comprehensive foundation. 1998. 2, 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 2, 4

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 5

[17] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. 2016. 2

[18] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 1, 3, 8

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3, 5

[20] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 3, 7, 8

[21] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 3

[22] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2016. 3

[23] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 3

[24] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009. 1, 2

[25] Yunho Jeon and Junmo Kim. Active convolution: Learning the shape of convolution for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4201–4209, 2017. 3

[26] Cem Keskin and Shahram Izadi. Splinenets: continuous neural decision graphs. In *Advances in Neural Information Processing Systems*, pages 1994–2004, 2018. 3

[27] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural

networks. In *Advances in neural information processing systems*, pages 971–980, 2017. 2

[28] Mandy Lange, Dietlind Zühlke, Olaf Holz, and Thomas Villmann. Applications of lp-norms and their smooth approximations for gradient based learning vector quantization. In *ESANN*, 2014. 2, 3

[29] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pages 2181–2191, 2017. 3

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 8

[32] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3

[33] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. 3

[34] Ningning Ma, X. Zhang, J. Huang, and J. Sun. Weightnet: Revisiting the design space of weight networks. 2020. 3, 8

[35] Ningning Ma, X. Zhang, and J. Sun. Funnel activation for visual recognition. 2020. 2

[36] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018. 3, 5, 8

[37] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013. 2, 4

[38] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *ArXiv*, abs/1908.08681, 2019. 7

[39] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 1, 2

[40] Suo Qiu, Xiangmin Xu, and Bolun Cai. Frelu: Flexible rectified linear units for improving convolutional neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1223–1228. IEEE, 2018. 2

[41] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 1, 2, 3, 4, 7

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 1, 5

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 3, 5

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[45] Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. *arXiv preprint arXiv:1911.09737*, 2019. 2

[46] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 3

[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 3

[48] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 3

[49] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1, 3

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[51] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017. 3

[52] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018. 3

[53] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 3

[54] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and

Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018. 3

[55] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 2

[56] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems*, pages 1305–1316, 2019. 3

[57] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 3

[58] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 3

[59] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 3

[60] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 8