CondConv: Conditionally Parameterized Convolutions for Efficient Inference

Brandon Yang, Gabriel Bender, Quoc V. Le, Jiquan Ngiam
Google Brain

NIPS 2019

Presented by Eungyeup Kim

Vision Seminar
09 APR 2020

Motivation

Convolutional layer is a basic building block of modern deep learning architecture.

- *Same convolutional kernels are applied to every example in a dataset*

- *To increase the capacity of a model, we usually add more convolutional layers or increase the size of existing convolutions (kernel height/width, number of input/output channels)*

- *This increases the 1) computational cost proportionally to the size of the input to the convolution and 2) latency requirements at inference.*

- So, how do we manage this model capacity and latency constraints?

This work

1) Proposes conditionally parameterized convolutions (CondConv), which challenge the paradigm of static convolutional kernels by **computing convolutional kernels as a function of the input**.

2) Demonstrates that CondConv can increase model capacity and performance while maintaining efficient inference.

Methods

Conditionally Parameterized Convolutions

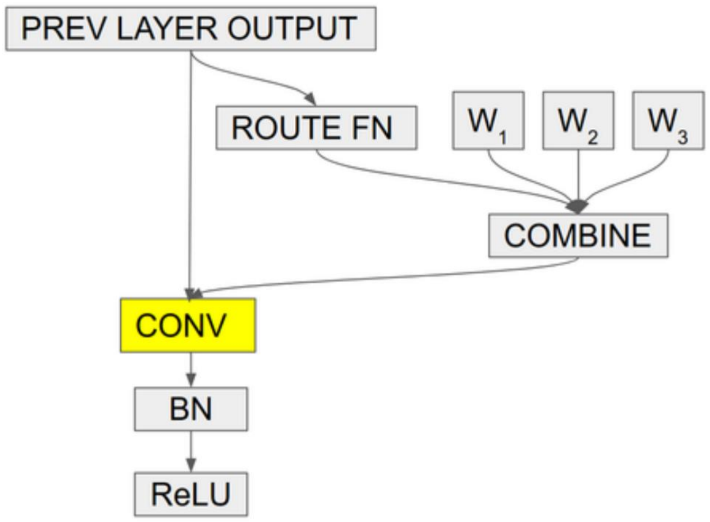
$$\text{output} = (\alpha_1 \cdot W_1 + \cdots \alpha_n \cdot W_n) * x,$$

where each $\alpha_i = r_i(x)$ is an example-dependent scalar weight computed a routing function r_i with learned parameters and n is the number of experts. Kernel W_i has the same dimensions as the kernel in the original convolution.

Routing function

$$r(x) = Sigmoid(GAP(x)R),$$

where R is a matrix of learned routing weights mapping the pooled inputs to n expert weights.



(a) CondConv: $(\alpha_1 W_1 + \ldots + \alpha_n W_n) * x$

Methods

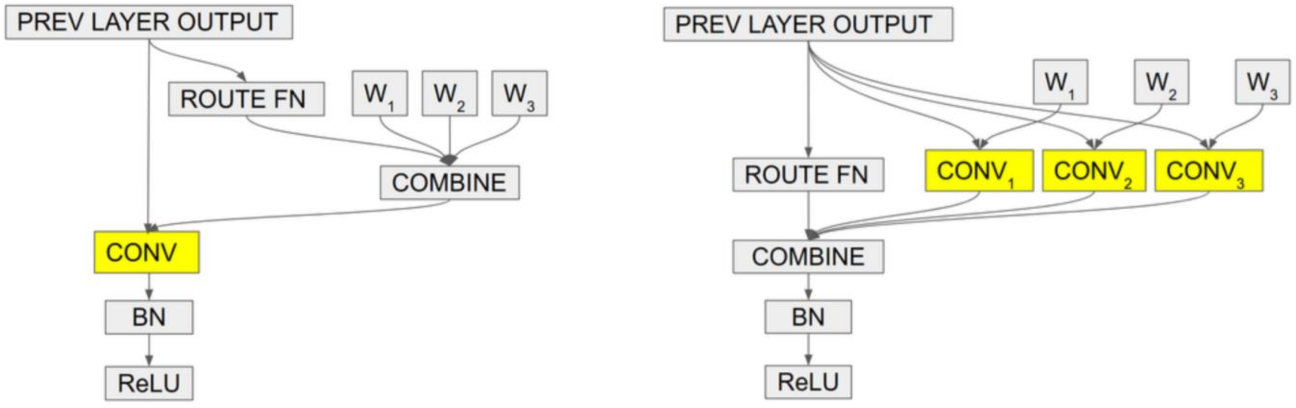
Strengths of CondConv

- In regular convolutional layer, we increase the kernel height/width or number of input/output channel, which requires multiple-add proportional to the number of pixels in input feature map.

- In CondConv, we compute a kernel for each example as a linear combination of n experts before applying the convolution. This requires only 1 additional multiply-add.

- A CondConv layer is mathematically equivalent to a more expensive linear mixture of experts formulation, where each expert corresponds to a static convolution.

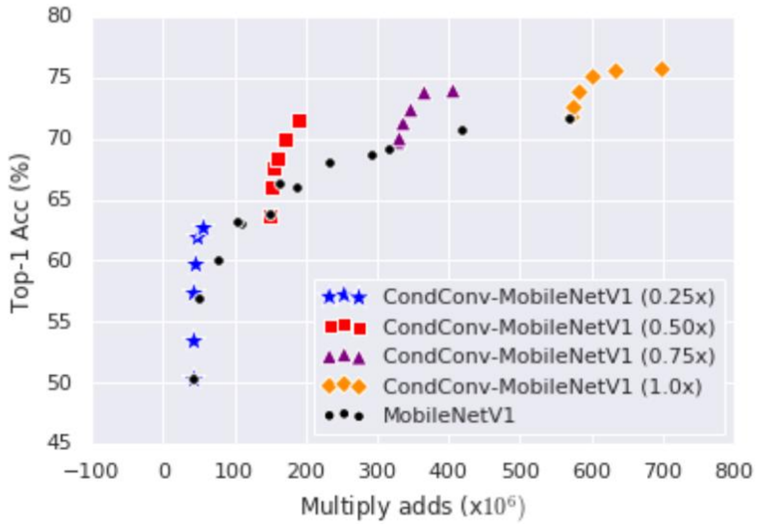
$$\text{output} = (\alpha_1 \cdot W_1 + \cdots \alpha_n \cdot W_n) * x = \alpha_1 \cdot (W_1 * x) + \cdots + \alpha_n \cdot (W_n * x)$$



(a) CondConv: $(\alpha_1 W_1 + \ldots + \alpha_n W_n) * x$ (b) Mixture of Experts: $\alpha_1(W_1 * x) + \ldots + \alpha_n(W_n * x)$

Experiments

[ImageNet Classification]



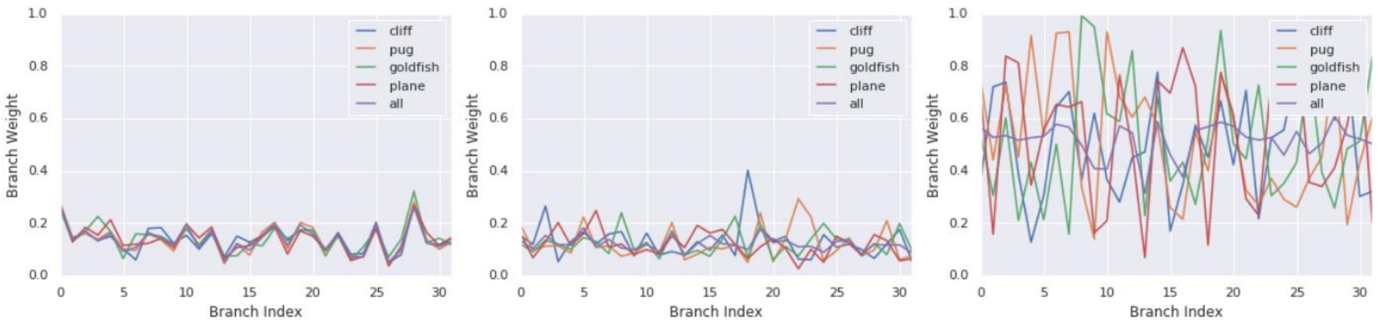
	Baseline[2]		CondConv	
	MADDs ($\times 10^6$)	Top-1 (%)	MADDs ($\times 10^6$)	Top-1 (%)
MobileNetV1 (1.0x)	567	71.9	600	73.7
MobileNetV2 (1.0x)	301	71.6	329	74.6
MnasNet-A1	312	74.9	325	76.2
ResNet-50	4093	77.7	4213	78.6
EfficientNet-B0	391	77.2	413	78.3

[Object Detection]

	Baseline[3]		CondConv	
	MADDs ($\times 10^6$)	mAP	MADDs ($\times 10^6$)	mAP
MobileNetV1 (0.5x)	352	14.4	363	18.0
MobileNetV1 (0.75x)	730	18.2	755	21.0
MobileNetV1(1.0x)	1230	20.3	1280	22.4

- Performance is improved as the number of experts is increased.
- CondConv improves performance relative to inference cost on a wide range of architectures.

Experiments



(a) Layer 12 (b) Layer 26 (c) Fully Connected (FC)

Figure 3: Mean routing weights for four classes averaged across the ImageNet validation set at three different depths in our CondConv-MobileNetV1 (0.5x) model. CondConv routing weights are more class-specific at greater depth.

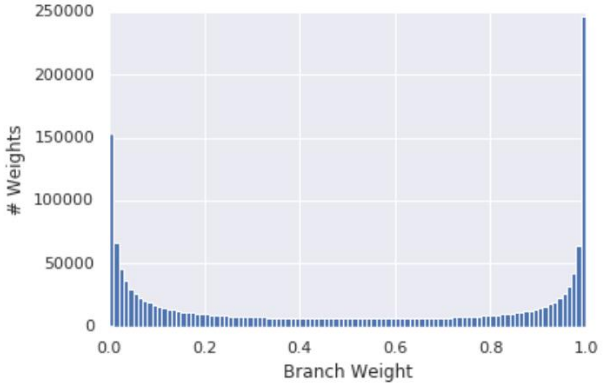


Figure 4: Distribution of routing weights in the final CondConv layer of our CondConv-MobileNetV1 (0.5x) model when evaluated on all images in the ImageNet validation set. Routing weights follow a bi-modal distribution.

- The distribution of the routing weights is very similar across classes at early layers, and become more class specific at layer layers.
- The routing weights of the final fully-connected layer follow a bi-modal distribution, with most experts receiving a routing weight close to 0 or 1.
- In other words, the experts are sparsely activated, even without regularization.

Experiments

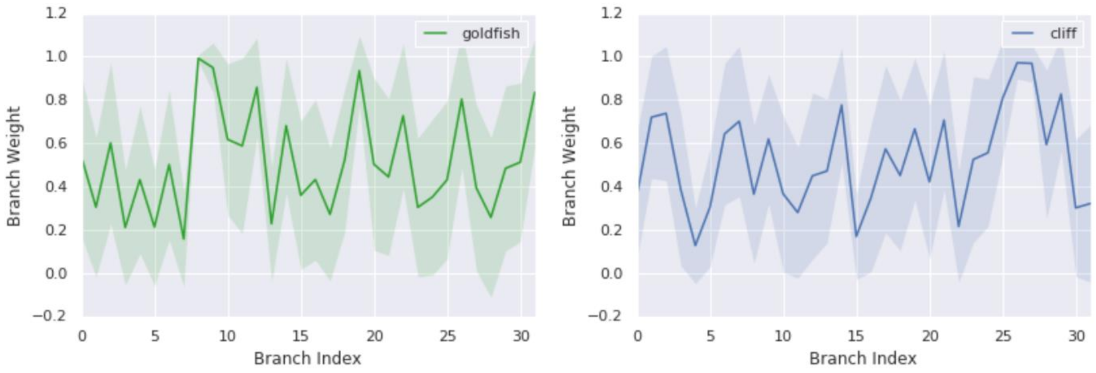
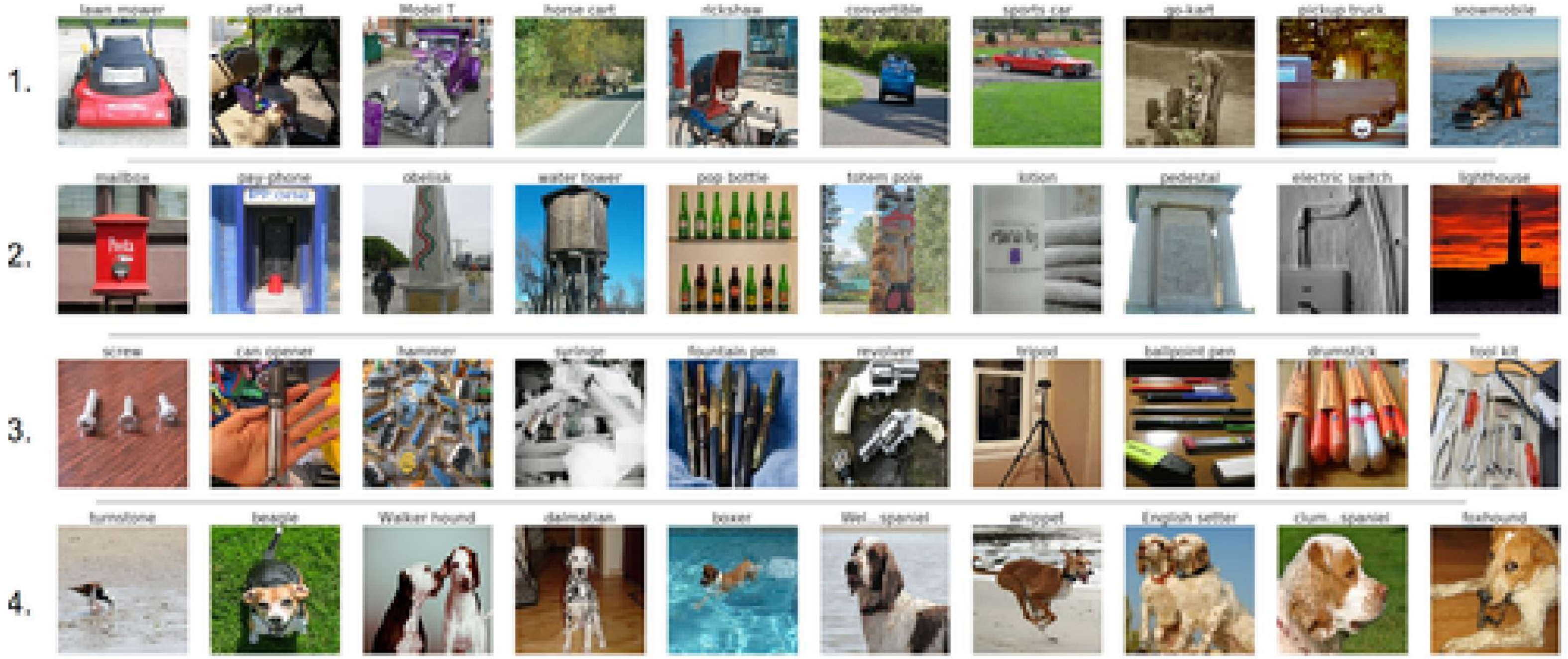


Figure 5: Routing weights in the final CondConv layer in our CondConv-MobileNetV1 (0.5x) model for 2 classes averaged across the ImageNet validation set. Error bars indicate one standard deviation.



- In the intra-class variation, some kernels are activated with high weight and small variance for all examples.
- However, there can be big variation in the routing weights between examples.
- Top 10 classes with highest mean routing weight for 4 different experts in the final CondConv layer.
- This indicates that CondConv layers learn to specialize in semantically and visually meaningful ways.

Thank you