Algorithm 1 The Algorithm for Optimal Separable Convolution

Input: Input channel $C_1 = C_{in}$, output channel $C_{N+1} = C_{out}$, kernel size (K^H, K^W), number of separated convolutions N

Optional Input: internal kernel sizes (optional, preset), internal number of groups (optional, masked values), spatial separable (True or False)

Output: internal channel sizes $C_2, \cdots, C_N$, internal kernel sizes $K_1^{H|W}, \cdots, K_N^{H|W}$, internal number of groups $g_1, \cdots, g_N$

Calculate internal channel sizes $C_2, \cdots, C_N$ as $\min(C_{in}, C_{out})$, $\max(C_{in}, C_{out})/4$, or $4\min(C_{in}, C_{out})$, etc. according to a preset policy.

if internal kernel sizes $K_1^{H|W}, \cdots, K_N^{H|W}$ are not given **then**
  if spatial separable **then**
    Set $K_{\lfloor N/2 \rfloor}^H = K^H$, $K_{\lfloor N/2+1 \rfloor}^W = K^W$ and all other internal kernel sizes to 1.
  else
    Set $K_{\lfloor N/2 \rfloor}^{H|W} = K^{H|W}$ and all other internal kernel sizes to 1.
  end if
end if

Calculate internal channels per group n_l according to $n_l = \frac{\sqrt[N]{\Pi_{i=1}^{N+1} C_i \Pi_{i=1}^{N} K_i^H \Pi_{i=1}^{N} K_i^W}}{C_{l+1} K_l^H K_l^W}$.

Let $g_l = \min(\lceil C_l/n_l \rceil, C_l, C_{l+1})$. If $C_l/n_l < 1$ or $C_l/n_l > \min(C_l, C_{l+1})$ for certain l, re-optimize g_l with a masked number of groups by pre-setting $g_l = 1$ for $l \in \{l : C_l/n_l < 1\}$, $g_l = \min(C_l, C_{l+1})$ for $l \in \{l : C_l/n_l > \min(C_l, C_{l+1})\}$.

▷ Because $n_l \sim \sqrt[N]{C}$, for large channel sizes, we rarely need to re-optimize.

Return $C_2, \cdots, C_N$; $K_1^{H|W}, \cdots, K_N^{H|W}$; $g_1, \cdots, g_N$

A Algorithmic Details of the Proposed Optimal Separable Convolution

For the proposed optimal separable convolution, oneof the internal kernel sizes can take $K^{H|W}$, while the rest takes 1. In this research, we simply select the middle kernel size as (K^H, K^W). In a spatial separable configuration, we select the middle two to have their kernel sizes as $(K^H, 1)$ and $(1, K^W)$. It is worth noting that all these configurations have the same FLOPs. Unlike the spatial separable convolution where the spatial separable configuration is able to reduce the complexity from K^2 to $2K$. This is because the complexity has already been reduced to $O(K)$ for the proposed optimal separable convolution. Another interesting property of the proposed optimal separable convolution is that it prefers large kernel sizes over small ones.

For the proposed optimal separable convolution, we are able to preset the internal convolutional kernel sizes according to a custom policy, and optimize the internal number of groups only. Furthermore, we are able to preset a portion of the internal number of groups to certain values, and optimize only the remaining internal number of groups. Suppose that the internal channel and kernel sizes are given. Without loss of generality, we assume that $g_{M+1}, \cdots, g_N$ are preset. The proposed optimal separable problem will be an M-separable convolution sub-problem ($M < N$):

$$f(\{n_*\}, \{K_*^{H|W}\}) = C_2 n_1 K_1^H K_1^W HW + \cdots + C_{M+1} n_M K_M^H K_M^W HW + const \tag{20}$$

satisfying the volumetric RF condition

$$K_1^H + \cdots + K_M^H = K^H + const \qquad (Receptive\ Field\ Condition) \tag{21}$$

$$K_1^W + \cdots + K_M^W = K^W + const \tag{22}$$

$$n_1 \cdots n_M \geq \frac{C_1}{n_{M+1} \cdots n_N} \Leftrightarrow g_1 \cdots g_M \leq \frac{C_2 \cdots C_N}{g_{M+1} \cdots g_N} \qquad (Channel\ Condition) \tag{23}$$

$$n_l \geq \max(1, \frac{C_{l+1}}{C_l}) \Leftrightarrow g_l \leq \min(C_l, C_{l+1}) \quad (Group\ Convolution\ Condition) \tag{24}$$

This M-separable sub-problem can be solved by the same algorithm. A detailed implementation of the proposed optimal separable convolution is described by Algorithm 1.

Table 5: Experimental results on CIFAR10 for the ResNet architecture with ablation studies of internal BN and non-linearity and spatial separable configuration.

Net Arch	Channel Multiplier	FLOPs (billion)	#Params (million)	Accuracy (%)	Internal BN and Non-linearity (%)	Spatial Separable (%)
ResNet20	-	0.04055	0.270	91.25	-	-
o-ResNet20	3.875	0.04054	**0.206**	**92.89**	**92.74**	**92.32**
ResNet32	-	0.06886	0.464	92.49	-	-
o-ResNet32	3.875	0.06760	**0.352**	**93.18**	**93.22**	**92.88**
ResNet56	-	0.12548	0.853	93.03	-	-
o-ResNet56	3.875	0.12180	**0.643**	**93.32**	**93.42**	**92.93**
ResNet110	-	0.25289	1.728	93.39	-	-
o-ResNet110	3.875	0.24370	**1.298**	**94.35**	**94.21**	**93.96**

B TRAINING SETTINGS

Experiments on CIFAR10 for the ResNet architecture The images are padded with 4 pixels and randomly cropped into 32×32 to feed into the network. A random horizontal flip with a probability of 0.5 is also applied. All the networks are trained with a standard SGD optimizer for 200 epochs. The initial learning rate is set to 0.1, with a decay of 0.1 at the 100 and 150 epochs. The batch size is 128. A weight decay of 0.0001 and a momentum of 0.9 are used.

Experiments on CIFAR10 for the DARTS architecture We follow the same training settings in (Liu et al., 2018): the network is trained with a standard SGD optimizer for 600 epochs with a batch size of 96. The initial learning rate is set to 0.025 with a cosine learning rate scheduler. A weight decay of 0.0003 and a momentum of 0.9 are used. Additional enhancements include cutout, path dropout of probability 0.2, and auxiliary towers with weight 0.4.

Experiments on ImageNet40 for the ResNet architecture Each network is trained with a standard SGD optimizer for 20 epochs with the initial learning rate set to 0.1, and a decay of 0.1 at the 10 and 15 epochs. The batch size is 256, the weight decay is 0.0001 and the momentum is 0.9.

Experiments on full ImageNet for the DARTS architecture We follow the training setting in (Chen et al., 2019): the images are random resized crop into 224×224 patches with a random scale in [0.08, 1.0] and a random aspect ratio in [0.75, 1.33]. Random horizontal flip and color jitter are also applied. The network is trained from scratch for 250 epochs with batch size 1024 on 8 GPUs. An SGD optimizer with an initial learning rate of 0.5, a momentum of 0.9, and a weight decay of 3e-5. The learning rate is decayed linearly after each epoch. Additional enhancements include label smoothing with weight 0.1 and auxiliary towers with weight 0.4.

C ABLATION STUDIES

Internal BatchNorm and Non-linearity For a DCNN, it is generally a good practice to add a BatchNorm (BN) (Ioffe, 2017) and a non-linearity after each convolution. For the proposed optimal separable convolution, we wonder if it is still necessary to add such a BN and a non-linearity after each of the internal separated convolutions. Experimental results are illustrated in Table 5. Comparing the "Internal BN and Non-linearity" column against the "Accuracy" column, we are able to conclude that with or without internal BN and non-linearity, similar results with only statistical variances can be generated. This is reasonable because the network has already been regularized by outer BN and non-linearity layers from the macro architecture. Internal ones shall offer little to no additional improvements. Because internal BN and non-linearity could introduce extra computation and parameters, in the proposed research, we shall not use internal BN and non-linearity.

Spatial Separable Another variation of the proposed optimal separable convolution scheme is the spatial separable configuration. For Equation (16), the optimal solution is achieved when one of

the internal kernel sizes takes $K^{H|W}$and all the rest takes 1. It does not matter which one of the internal kernel sizes takes $K^{H|W}$. Hence, we have this spatial separable variant: a single kernel takes (K^H, K^W) or two kernels take $(K^H, 1)$ and $(1, K^W)$. The detailed implementation is illustrated in Algorithm 1. While spatial separable or not affects neither the FLOPs nor the number of parameters for the proposed optimal separable convolution, the results could be slightly different. As illustrated by the column "Spatial Separable" in Table 5, the spatial separable configuration leads to slightly worse performances. The reason might be that spatial separation fuses horizontal and vertical features separately, which could be less efficient than fusing them simultaneously.

D Related Work

There have been many previous works aiming at reducing the amount of computation in convolution. Historically, researchers apply Fast Fourier Transform (FFT) (Nussbaumer, 1981; Quarteroni et al., 2010) to implement convolution. For 1D convolution, FFT reduces the number of computations for H points from $O(H^2)$ to $O(H \log H)$. For 2D convolution, FFT-2D reduces the computational complexity from $O(HW \cdot K^2)$ to $O(HW \cdot (\log H + \log W))$ (Podlozhnyuk, 2007). Hence, it can be easily concluded that FFT gains great speed up for large convolutional kernels. For small convolutional kernels ($K << H$ or W), a direct application is often still cheaper. Researchers also explore low rank approximation (Jaderberg et al., 2014; Ioannou et al., 2015) to implement convolutions. However, most of the existing methods obtain moderate efficiency improvements, and they usually require a pre-trained model and mainly focus on network pruning and compression. In recent state-of-the-art deep CNN models, several heuristics are adopted to reduce the heavy computation in convolution. For example, in (He et al., 2016), the authors use a bottleneck structure. Yet in (Sandler et al., 2018), the authors adopt an inverted bottleneck structure. Such heuristics may require further ad hoc design to work in practice, however, they are not solid and shall become less convincing.

Among various implementations of convolution, separable convolution has been proven to be more efficient in reducing the computational demand. Depth separable convolution is explored extensively in modern DCNNs (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019; Liu et al., 2018; Tan & Le, 2019). It reduces the computational cost of a conventional convolution from $O(C^2K^2HW)$ to $O(CHW \cdot (C + K^2))$. However, the proposed optimal separable convolution is even more efficient than depth separable convolution. It can be calculated at $O(C^{\frac{3}{2}}KHW)$ and has the full potential to replace the usage of depth separable convolutions. A second advantage of the proposed optimal separable convolution over depth separable convolution is that it can be applied to fully connect layers if we view them as 1×1 convolutional layers, whereas depth separable convolution cannot. Spatial separable convolution was originally developed to speed up image processing operations. For example, a Sobel kernel is a 3×3 kernel and can be written as $(1, 2, 1)^T \cdot (-1, 0, 1)$. Spatial separable will require 6 instead of 9 parameters while doing the same operation. Spatial separable convolution is also adopted in the design of modern DCNNs. For example, in (Szegedy et al., 2016), the authors introduce spatial separation to the GoogLeNet (Szegedy et al., 2015) architecture. For the proposed optimal separable convolution, there is also a spatial separable configuration.

In the body of literature, separable convolution is also referred to *factorized convolution* or *convolution decomposition*. In this research, the proposed scheme is called optimal separable convolution following the naming conventions of depth and spatial separable convolutions.