Structural Compression of ResNet-like Convolutional Neural Networks

Nicholas Kastanos (nk569), Queens' College
Department of Computer Science and Technology
University of Cambridge
Cambridge, CB3 0FD
nk569@cam.ac.uk

Abstract

abstract

1 Introduction

2 Related Work

2.1 ResNet

The residual block first postulated for use in ResNet has become a common-place feature in many subsequent networks (DenseNet, Inception).

2.2 Separable convolutions

Convolution Layers contain a vast majority of the parameters in modern CNNs. By targeting parameter reductions to these layers, the compression can be spread throughout the network. Separable convolutions reduce the number of parameters by separating the convolution into multiple stages through spatially and depthwise separable convolutions. While these convolutions reduces the memory and computation requirements of the system, the reduction in parameters reduces the number of possible kernels explored in training, and the resulting network may be suboptimal.

2.2.1 Spatially separable convolutions

A convolution kernel can be decomposed on its 2D spatial axis, i.e. height and width. Conceptually, the $n \times n$ kernel can be separated into two smaller kernels, a $n \times 1$ followed by a $1 \times n$ kernel. These kernels can be applied in sequential convolutions to obtain the same output shape as the single convolution. These decomposed kernels scale the parameters required by the convolution by a factor $P_s(n)$ (see Equation 1).

Similarly, the multiplication operations of a spatially separated convolution are reduced. For a $M \times M$ input convolved with a $n \times n$ kernel, the number of multipications are reduced by a factor of $M_s(n)$ (see Equation 2).

Equations 1) and 2 show that spatially separable convolutions show computational benefits when $n > 2$.

$$P_s(n) = \frac{2}{n} \tag{1}$$

$$
\begin{aligned}
M_s(M,n) =& \frac{2}{n} + \frac{2}{n(M-2)} \\
\Rightarrow M_s(n) =& \frac{2}{n}, \text{ where } M >> n
\end{aligned}
\tag{2}
$$

2.2.2 Depthwise separable convolutions

Depthwise separable convolutions separate the spatial convolution from the depth of the filters. This is accomplished by an initial depthwise convolution, followed by a pointwise convolution. The initial depthwise convolution separates the channels of the input and kernel, and convolves them independently. The pointwise convolution is a $N_F \times 1 \times 1 \times N_C$ convolution where N_F and N_C are the number of filters and channels respectively.

The number of parameters $P_d(n)$ and multiplications $M_d(n)$ are reduced by the same factor, which can be seen in Equation 3. Many CNNs have $N_F >> 1$, therefore depthwise convolutions show compression kernel sizes greater than 1.

$$
P_d(n) = M_d(n) = \frac{1}{n^2} + \frac{1}{N_F} \approx \frac{1}{n^2} \tag{3}
$$

2.3 Quantization and datatype compression

Many resource-constrained devices do not have sufficient memory to use large neural networks, or may not have access to floating-point arithmetic units. Both of these factors can be mitigated by using low-precision integer datatypes, such as 8-bit integers. This effectively reduces the memory required for each parameter by ¼.

However, by reducing the precision of the datatype, the learned parameters are not represeted fully, reducing the performance of the network. This has been shown to have significant effect on the network when reducing 32-bit floating point numbers to 8-bit integers, however quantization aware training can limit the impact.

TensorFlow, by default, uses 32-bit floating point precision for its network layers and training. TensorFlow Lite provides functionality to convert trained models to quantized, 8-bit integer models, however it does not have native 8-bit implementations for all layer types. In the event an incompatible layer is used, the engine upscales the activations to the full precision for that layer.

3 Methodology

In order to asses the benefits and costs of the compression, a baseline architecture is established and used as a comparison reference point. The ResNet50V2 image classification CNN-based neural network is used as the baseline architecture. The baseline architecture and subsequent networks are trained using the same parameters using TensorFlow.

3.1 Training

The CNNs are trained using the CIFAR-10 dataset, with random horizontal flips, rotations, and cropping. The Adam optimizer is used with a learning rate of 0.001, and categorical crossentropy loss. The networks are trained for 30 epochs with a batch size of 32.

3.2 Modified residual blocks

In order to modify the ResNetV2 network, the convolutions in the residual blocks are chosen to be substituted with separable convolutions. Additional batch normalization and activation layers are not added between these new layers. An example of this expansion can be seen in Figure 1. Only convolutions which would provide parameter and compute reduction are replaced with separable convolutions. For ResNet50V2, these are any convolutions where the kernel size $n > 1$.

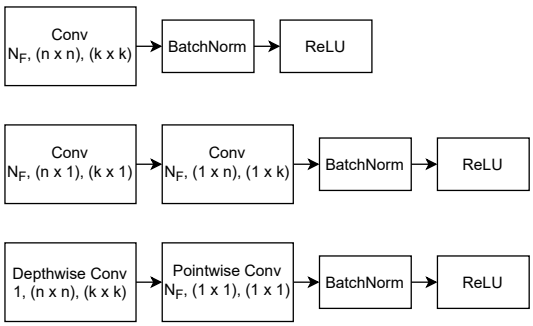

Figure 1: Example of replaced convolutions with separable convolutions with N_F filters, kernel size n, and stride k. From top to bottom: Traditional Convolution; Spatially Separable Convolution; Depthwise Separable Convolution

Table 1: Results

Name	Loss	Accuracy (%)	Size (MB)
ResNet50V2	0	100	20000
ResNet50V2-Spatial	0	100	2000
ResNet50V2-Depthwise	0	100	2000
Compressed ResNet50V2	0	100	20000
Compressed ResNet50V2-Spatial	0	100	2000
Compressed ResNet50V2-Depthwise	0	100	2000

These two additional CNNs are called ResNet50V2-Spatial and ResNet50V2-Depthwise, where convolutional layers are replaced with spatial and depthwise convolutions respectively.

3.3 Post-training quantization

The fully trained models are compressed to 8-bit precision using TensorFlow Lite. The framework recommends the provision of the representative dataset to enhance the optimization process. This allows for quantization of both weights and activations for quantizable operations. The representative dataset is compiled using 1000 samples from the dataset.

4 Evaluation and results

5 Conclusion