Training Structured Efficient Convolutional Layers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Typical recent neural network designs are primarily convolutional layers, but the tricks enabling structured efficient linear layers (SELLs) have not yet been adapted to the convolutional setting. We present a method to express the weight tensor in a convolutional layer using diagonal matrices, discrete cosine transforms (DCTs) and permutations that can be optimised using standard stochastic gradient methods. A network composed of such structured efficient convolutional layers (SECL) outperforms existing low-rank networks and demonstrates competitive computational efficiency.

1 Introduction

Deep neural networks have evolved, and no longer contain the gigantic linear layers seen in Simonyan & Zisserman (2015), instead opting for large convolutional feature maps and increased depth (Springenberg et al., 2014; He et al., 2016; Hu et al., 2018). This means that most of the network's parameters and multiply-add (Mult-Add) operations are used in these convolutions. We present a method to reduce both, while keeping the network structure the same.

SELLs provide a framework to approximate linear layers that we adapt to make convolutions more efficient. A convolution can be viewed as a matrix multiplication between the extracted patches from the input tensor and the weights of the filters passed over the image. In this work, we show that this matrix multiplication can be replaced with a structured efficient transformation based on the ACDC layer (Moczulski et al., 2016). In practice, this gives us a specific parameterisation at training time, which we describe in Section 3.

This structured efficient parameterisation of a convolutional layer consumes far fewer parameters than a full convolutional layer, scaling as $O(N)$ versus $O(N^2)$ in the number of channels N. At the same time, it can be implemented efficiently at test time using a fast DCT. In our experiments, detailed in Section 4, we show that, using distillation (Crowley et al., 2018), a network composed of primarily this type of convolution can learn to classify CIFAR-10 to 91.43% accuracy, while only using 46,442 parameters. For comparison, this is a greater accuracy and fifty times fewer parameters than the network presented in Hinton et al. (2016).

2 Background

SELLs are aimed at compressing the linear layers of convolutional networks. Typically, they are composed as an operator $\mathbf{\Phi}$:

$$\mathbf{y} = \mathbf{x}\mathbf{\Phi}\left(\mathbf{D}, \mathbf{P}, \mathbf{S}, \mathbf{B}\right) \tag{1}$$

In which, $\mathbf{D}$ are diagonal matrices, $\mathbf{P}$ are permutations, $\mathbf{S}$ are sparse matrices and $\mathbf{B}$ are bases, such as Fourier, Cosine (Moczulski et al., 2016) or Hadamard (Yang et al., 2015; Ailon & Chazelle, 2009)

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

transforms. Using these component transformations, the resulting random projections can approximate random matrices used in deep learning. In this work, we build on the ACDC SELL (Moczulski et al., 2016).

Our approach can be viewed as yielding a low-rank tensor to use in the convolutional layers. Previous efforts on low-rank convolutional networks have focused on transforming pre-trained networks (Jaderberg et al., 2014; Alvarez & Petersson, 2016; Denton et al., 2014; Lebedev et al., 2014) or training networks with appropriate regularisers (Alvarez & Salzmann, 2017; Wen et al., 2017).

Networks with low-rank constraints are markedly more difficult to train. In Garipov et al. (2016) the authors train such a network on CIFAR-10, but only achieve a $2\times$ compression rate over a convolutional network, and attain less than 90% accuracy. Other papers have focused on similar tensor decompositions; Su et al. (2018) obtain 91.28% accuracy compressing a ResNet-34. However, neither decomposition affects the number of Mult-Adds used at test time, whereas our method achieves a substantial reduction.

3 Structured Efficient Convolutional Layers

If we define a function to map the patches over which a convolution passes on an input tensor to rows of a matrix as $\mathbf{F}$, we can express convolution using a *kernel matrix* $\mathbf{W}$ as $\mathbf{y} = \mathbf{F}^{-1}(\mathbf{F}(x)\mathbf{W})$. This is the common algorithm known as `im2col-gemm` (Chetlur et al., 2014). From this, we define a Structured Efficient Convolutional Layer (SECL) with the following parameterisation for $\mathbf{W}$:

$$\mathbf{W} = \prod_{l=1}^{L} \mathbf{A}_l \mathbf{C} \mathbf{D}_l \mathbf{C}^{-1} \mathbf{P} \tag{2}$$

Where $\mathbf{A}$ and $\mathbf{D}$ are diagonal matrices, $\mathbf{C}$ and $\mathbf{C}^{-1}$ are the forward and inverse DCTs and $\mathbf{P}$ is a riffle shuffle.

Using the weight matrix in Equation 2 is equivalent to a stack of ACDC layers (Moczulski et al., 2016); the permutation being implemented by a riffle shuffle. A *riffle shuffle* is a fixed permutation, splitting the input in half and then interleaving the two halves; equivalent to a perfect riffle shuffle with a deck of cards (Gilbert, 1955). As described in Section 4, this was found to work as well as a fixed random permutation and can be evaluated much faster (Zhang et al., 2018).

Substituting this parameterisation into convolutional layers presents a problem: most kernel matrices are not square, with one exception. Kernel matrices in pointwise convolutions are square when the number of input channels matches the output. To increase the number of channels, we repeat the input along the channel dimension. As channels commonly increase in integer steps, this allows us to implement almost all pointwise convolutions.

Given a pointwise convolution, we can now implement a convolution with any kernel size by preceding the pointwise with a grouped convolution. This is known as a depthwise separable convolution and has been demonstrated as a substitute for convolution (Chollet, 2016). This can also be implemented using an ACDC parameterisation for each filter, but there is not much benefit, as shown in Section 4.

The motivation underlying ACDC layers comes from their complex equivalent; using Fourier transforms, $\mathbf{F}$, it is possible to show (Moczulski et al., 2016; Huhtanen, 2008) almost all matrices $\mathbf{M}$ can be factored as:

$$\mathbf{M} = \left[\prod_{i=1}^{N-1} \mathbf{D}_{2i-1} R_{2i} \right] \mathbf{D}_{2N-1} \tag{3}$$

Where $\mathbf{D}_{2i-1}$ is a diagonal and $\mathbf{R}_{2i}$ is composed of $\mathbf{F}\mathbf{D}\mathbf{F}^{-1}$.

However, machine learning systems typically operate using real numbers, leading to the decision to use the DCT. Despite the break in theory, the ACDC layer was found to work well as a replacement for the fully connected layers in CaffeNet (Moczulski et al., 2016).

Table 1: Results of training a Wide ResNet with SECL substituting the convolutional layers. WRN-SECL refers to a Wide ResNet using full rank grouped 3x3 convolutions, while WRN-SECL-LR refers to using ACDC-parameterised grouped 3x3 convolutions. Networks trained without distillation are reported under *Scratch*, while those trained using any form of distillation are under *Distilled*.

			Scratch	Distilled
Model	Params	Mult-Adds	Top 1	Top 1
WRN(40,2)	2.24M	328M	4.8%	–
WRN-SECL(40,2)	46.4K	33.2M	10.42%	8.57%
WRN-SECL-LR(40,2)	38.7K	33.2M	13.4%	10.85%
Belagiannis et al. (2018)	0.27M	–	–	8.08%
Hinton et al. (2016)	0.27M	–	–	8.88%
Su et al. (2018)	40K	–	–	8.72%

4 Experiments

We demonstrate the effectiveness of this compression strategy in Section 4.1 in experiments on CIFAR-10. PyTorch (Paszke et al.) was used to implement experiments. All code will be made available following the reviewing process.

Linear Approximation We compared the riffle shuffle to a fixed random permutation on the toy synthetic regression problem described in Section 6.1 of Moczulski et al. (2016). Both permutations converged to a final mean squared error of 0.02.

Importance of Weight Decay A small network based on the All Convolutional network (Springenberg et al., 2014) was trained using Hyperband (Li et al., 2017) to tune the weight decay term, learning rate and minibatch size on the CIFAR-10 dataset (Krizhevsky, 2009). The networks converging well had weight decay around 10^{-5}, prompting us to use 8.8×10^{-6} in all experiments.

4.1 Convolutional Networks

To compare with contemporary works on compressed networks, we focus on a recent network architecture, but SECLs could be effectively substituted into any convolutional network. The network used in experiments was a Wide ResNet with depth 40 and width factor 2 (Zagoruyko & Komodakis, 2016), and we train it on the CIFAR-10 dataset (Krizhevsky, 2009). We use 12 ACDC layers in each convolutional layer, to match the original paper (Moczulski et al., 2016).

The results are shown in Table 1. The results are most comparable to those of Su et al. (2018), in which the authors compress a network using a low-rank approximation. We performed distillation using attention-transfer (Zagoruyko & Komodakis, 2017) with the method described in Crowley et al. (2018). Each other paper reported as *distilled* in Table 1 implements its own form of distillation, detailed in each paper.

Each ACDC layer is expected to cost $4N + 5N\log_2(N)$ Mult-Adds at test time. However, we found that the early layers, where the dimension of the effective matrix multiplication is low, the Mult-Adds used by the unrolled ACDC layers is greater than just applying the parameterised filter tensor in a convolution. In this case, we assume that at test time the implementation of layers would depend on which would be cheaper.

5 Conclusion

It is well known that deep networks are overparameterised (Denil et al., 2013), but networks with low-rank weight matrices have failed to gain traction as they are typically difficult to train. Fortunately, for networks with structured efficient parameterisations we can use the same tools we would use to train a standard deep network, requiring a minimum of hyperparameter tuning. At test time, they can be implemented extremely efficiently using FFT-like algorithms or, as noted in Moczulski et al. (2016), by an optical processor (Saade et al., 2015). In this work, we have shown that these parameterisations aren't restricted to linear layers, and can be applied to convolutional layers too; resulting in small, efficient neural networks.

References

Ailon, N. and Chazelle, B. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

Alvarez, J. M. and Salzmann, M. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, 2017.

Alvarez, J. M. and Petersson, L. DecomposeMe: Simplifying convnets for end-to-end learning. *arXiv:1606.05426*, 2016.

Belagiannis, V., Farshad, A., and Galasso, F. Adversarial network compression. *CoRR*, abs/1803.10750, 2018. URL `http://arxiv.org/abs/1803.10750`.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. cuDNN: Efficient primitives for deep learning. *arXiv:1410.0759*, 2014.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL `http://arxiv.org/abs/1610.02357`.

Crowley, E. J., Gray, G., and Storkey, A. Moonshine: Distilling with cheap convolutions. In *Advances in Neural Information Processing Systems*, 2018.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, 2013.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, 2014.

Garipov, T., Podoprikhin, D., Novikov, A., and Vetrov, D. P. Ultimate tensorization: compressing convolutional and FC layers alike. *arXiv:1611.03214*, 2016.

Gilbert, E. Theory of shuffling. Technical report, Bell Labs, Murray Hill, New Jersey, U.S., 1955.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2016. URL `http://arxiv.org/abs/1503.02531`.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Huhtanen, M. Approximating ideal diffractive optical systems. *Journal of Mathematical Analysis and Applications*, 345(1):53–62, 2008.

Jaderberg, M., Vedaldi, A., and Zisserman, A. Speeding up Convolutional Neural Networks with Low Rank Expansions. In *British Machine Vision Conference*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. *arXiv:1412.6553*, 2014.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18 (1):6765–6816, 2017.

Moczulski, M., Denil, M., Appleyard, J., and de Freitas, N. ACDC: a structured efficient linear layer. In *International Conference on Learning Representations*, 2016.

Paszke, A., Gross, S., Chintala, S., and Chanan, G. PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration. `https://github.com/pytorch/pytorch`. Accessed: 31st October 2017.

Saade, A., Caltagirone, F., Carron, I., Daudet, L., Drémeau, A., Gigan, S., and Krzakala, F. Random projections through multiple optical scattering: Approximating kernels at the speed of light. *CoRR*, abs/1510.06664, 2015. URL `http://arxiv.org/abs/1510.06664`.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. Striving for simplicity: The all convolutional net. *arXiv:1412.6806*, 2014.

Su, J., Li, J., Bhattacharjee, B., and Huang, F. Tensorized Spectrum Preserving Compression for Neural Networks. *arXiv:1805.10352*, 2018.

Wen, W., Xu, C., Wu, C., Wang, Y., Chen, Y., and Li, H. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L., and Wang, Z. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference*, 2016.

Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.

Zhang, X., Zhou, X., Lin, M., and Sun, J. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.