Learning Multiscale Convolutional Kernels in the Fourier Domain

Abstract

1. Introduction

Convolutional Networks are one of the most widely-used models in computer vision. They are able to improve generalization performance by exploiting stationarity and local statistics to greatly reduce the number of parameters in the model while maintaining the ability to model the data.

The well-known Convolution Theorem states that the convolution operator is diagonalized in the Fourier basis, i.e. a convolution in the spatial domain is equivalent to a pointwise multiplication in the Fourier domain. This result has been exploited to accelerate training and inference using ConvNets by reusing the same Fast Fourier Transform (FFT) multiple times during each stage of the backpropagation algorithm. However, the passing of the weights to the frequency domain has thus far been a purely computational trick which leverages the low complexity of the FFT algorithm to compute the same weight updates as the traditional method in a more efficient manner. An alternate viewpoint is to learn the weights directly in the frequency domain. Seen through this lense, spatial convolutions with small kernels are simply a parameterization of a low-dimensional subspace corresponding to high frequency filters. We propose several alternate parameterizations which are capable of automatically adjusting the kernel size in the spatial domain by modulating the smoothness of the kernel weights in the frequency domain. We show that our method is capable of outperforming classic ConvNets for large image sizes.

2. Theory

2.1. Backpropagation Algorithm

We briefly recall the three steps performed by the backpropagation algorithm, which is the standard method to train Convolutional networks. First we fix some notation: for a given layer, we have a set of inputs x_{sf} indexed by s and f, each of size $n \times n$. Here s indicates the sample in the minibatch and f indicates the feature map. The output of the layer is represented by $y_{sf'}$, where f' represents the output feature map. The layer's trainable parameters consist of a set of weights $w_{f'f}$, each of size $k \times k$.

The backpropagation algorithm consists of three steps: the forward pass

$$y_{sf'} = \sum_f x_{sf} * w_{f'f} \tag{1}$$

the backward pass

$$\frac{\partial L}{\partial x_{sf}} = \sum_{f'} \frac{\partial L}{\partial y_{sf'}} * w^T_{f'f} \tag{2}$$

and the gradient weight update

$$\frac{\partial L}{\partial w_{f'f}} = \sum_s \frac{\partial L}{\partial y_{sf'}} * x_{sf} \tag{3}$$

3. Experiments

References

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.