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 lens, 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, which are the forward pass, backward pass and gradient update:

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

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

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

Letting $\mathcal{F}$ denote the 2-D Fourier transform operator and $\odot$ denote pointwise multiplication, these updates can be reformulated as follows:

$$y_{sf'} = \mathcal{F}^{-1} \sum_f \mathcal{F} x_{sf} \odot \mathcal{F} w_{f'f} \tag{4}$$

$$\frac{\partial L}{\partial x_{sf}} = \mathcal{F}^{-1} \sum_{f'} \mathcal{F} \frac{\partial L}{\partial y_{sf'}} \odot \overline{\mathcal{F} w^T_{f'f}} \tag{5}$$

$$\frac{\partial L}{\partial w_{f'f}} = \mathcal{F}^{-1} \sum_s \mathcal{F} \frac{\partial L}{\partial y_{sf'}} \odot \mathcal{F} x_{sf} \tag{6}$$

Here we assume that the weight kernels are zero-padded to be the same size as the input images, hence their Fourier transforms of both are the same size.

2.2. Spectral Networks on Images

We now consider the question of learning the weights directly in the Fourier domain. First, we fix some terminology. We call *free weights* the parameters of the model which are learned. We call the *expanded weights*, the output of some function which maps the free weights to a frequency representation which is the same size as the input

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

image. We will denote the free weights of one $k \times k$ feature map by W and its expanded weights of size $n \times n$ by $\tilde{W}$.

In the case of classical convolutional networks, the free weights and expanded weights are related as follows:

$$\tilde{w} = FWF^T \tag{7}$$

where F denotes the first k columns of the Discrete Fourier Transform (DFT) matrix. Note that this operation corresponds to zero-padding the $k \times k$ kernel to be of size $n \times n$ and applying a 2-D Fourier transform.

We can generalize this idea by letting

$$\tilde{w} = KWK^T \tag{8}$$

for different choices of K. Specifically, setting K to be an interpolation kernel we can encourage the expanded weights $\tilde{W}$ to be smooth, which corresponds to localized weights in the spatial domain.

A justification for this is due to a statement of the Uncertainty Principle in the context of harmonic analysis, which restricts how much spatial and frequency energy can be simultaneously concentrated. Specifically, the spatial and frequency variance of a function satisfy

$$\sigma_x^2 \sigma_\omega^2 \geq \frac{1}{4} \tag{9}$$

which means that the more one concentrates the spatial energy of a function, the more its energy in the frequency domain will be spread out and vice versa. For certain families of functions (such as Gaussians), the inequality in equation (9) becomes an equality, which indicates that enforcing smoothness in one domain translates into concentration in the other.

In the case of spectral networks, by learning a small number of weights and expanding them via an interpolation kernel, we obtain weights with a smooth representation in the frequency domain, and hence a well-localized support in the spatial domain. However, the smoothness in the frequency domain is not fixed, which means that in the spatial domain our convolutional kernels can have variable size. In principle, our approach should allow the learned kernels to adapt to the optimal size.

3. Experiments

References