Title: Fast Training of Convolutional Networks through FFTs

JX-Ma

2024/10/19

ABSTRACT

This article proposes a simple algorithm that can significantly accelerate training and inference, and can achieve orders of magnitude improvements compared to existing state-of-the-art implementations. This is achieved by computing the convolution as a pointwise product in the Fourier domain, while repeatedly using the same transformation feature map multiple times.

INTRODUCE

Normally, it takes several weeks to fully train a neural network. This article proposes a simple algorithm that utilizes convolutional networks to accelerate training and inference. The idea is to treat convolution as a product in the Fourier domain and repeatedly use the transformed feature map. The important operations for training convolutional networks can be seen as convolutions between two pairs of 2-D matrices

THEORY

Introduced some background and symbols of Fourier convolution

When the input tensor is nxn and the convolution kernel tensor is kxk, the number of floating-point operations for FFT is $6Cn^2logn + 4n^2$.

Assuming the input tensor batch is f, the output tensor channel is f ', the input tensor channel is s, the input tensor width and height are nxn, and the convolution kernel width and height are kxk, the number of floating-point operations required for direct convolution is $S \cdot f' \cdot f \cdot n'^2 \cdot k^2$ n' equal (n-k+1),The computational complexity of FFT used in this article is $2Cn^2logn[f' \cdot S + f \cdot S + f' \cdot f] + 4S \cdot f' \cdot f \cdot n^2$

The method proposed in this article also requires a lot of additional memory, and the specific required memory formula is as follows:

$4n(n+1)(S \cdot f + S \cdot f' + f \cdot f')$

Experience

This article compares the method implemented in this article with the custom implementation of CudaConv GPU and Torch 7 machine learning environment, and it can be seen that the method proposed in this article is significantly better than the other two methods in almost all cases. The improvement is particularly evident in the accGradParameters operation, which is the most computationally intensive operation. This may be because the convolution we computed has a larger kernel, and for such cases, FFT is more suitable.