Speeding up Convolutional Neural Networks with Low Rank Expansions

Max Jaderberg
max@robots.ox.ac.uk
Andrea Vedaldi
vedaldi@robots.ox.ac.uk
Andrew Zisserman
az@robots.ox.ac.uk

Visual Geometry Group
University of Oxford
Oxford, UK

Abstract

The focus of this paper is speeding up the evaluation of convolutional neural networks. While delivering impressive results across a range of computer vision and machine learning tasks, these networks are computationally demanding, limiting their deployability. Convolutional layers generally consume the bulk of the processing time, and so in this work we present two simple schemes for drastically speeding up these layers. This is achieved by exploiting cross-channel or filter redundancy to construct a low rank basis of filters that are rank-1 in the spatial domain. Our methods are architecture agnostic, and can be easily applied to existing CPU and GPU convolutional frameworks for tuneable speedup performance. We demonstrate this with a real world network designed for scene text character recognition, showing a possible 2.5× speedup with no loss in accuracy, and 4.5× speedup with less than 1% drop in accuracy, still achieving state-of-the-art on standard benchmarks.

1 Introduction

Many applications of machine learning, and most recently computer vision, have been disrupted by the use of convolutional neural networks (CNNs). The combination of an end-to-end learning system with minimal need for human design decisions, and the ability to efficiently train large and complex models, have allowed them to achieve state-of-the-art performance in a number of benchmarks [3, 10, 17, 29, 32, 36, 37]. However, these high performing CNNs come with a large computational cost due to the use of chains of several convolutional layers, often requiring implementations on GPUs [14, 17] or highly optimized distributed CPU architectures [39] to process large datasets. The increasing use of these networks for detection in sliding window approaches [9, 26, 32] and the desire to apply CNNs in real-world systems means the speed of inference becomes an important factor for applications. In this paper we introduce an easy-to-implement method for significantly speeding up pre-trained CNNs requiring minimal modifications to existing frameworks. There can be a small associated loss in performance, but this is tunable to a desired accuracy level. For example, we show that a 4.5× speedup can still give state-of-the-art performance in our example application of character recognition.

Layer name	Filter size	In channels	Out channels	Filters	Maxout groups	Time
Conv1	9×9	1	48	96	2	0.473ms (8.3%)
Conv2	9×9	48	64	128	2	3.008ms (52.9%)
Conv3	8×8	64	128	512	4	2.160ms (38.0%)
Conv4	1×1	128	37	148	4	0.041ms (0.7%)
Softmax	-	37	37	-	-	0.004ms (0.1%)

Table 1: The details of the layers in the CNN used with the forward pass timings of each layer.

dropout [12] lead to under-fitting, most likely due to the fact that we are already trying to heavily approximate our original filters. However, this is an area that needs to be investigated in more detail.

3 Experiments & Results

In this section we demonstrate the application of both proposed filter approximation schemes and show that we can achieve large speedups with a very small drop in accuracy. We use a pre-trained CNN that performs case-insensitive character classification of scene text. Character classification is an essential part of many text spotting pipelines such as [3, 4, 22, 23, 24, 25, 27, 28, 40, 42].

We first give the details of the base CNN model used for character classification which will be subject to speedup approximations. The optimization processes and how we attain the approximations of Scheme 1 & 2 to this model are given, and finally we discuss the results of the separable basis approximation methods on accuracy and inference time of the model.

Test Model. For scene character classification, we use a four layer CNN with a softmax output. The CNN outputs a probability distribution $p(c|x)$ over an alphabet C which includes all 26 letters and 10 digits, as well as a noise/background (no-text) class, with x being a grey input image patch of size 24×24 pixels, which has been zero-centred and normalized by subtracting the patch mean and dividing by the standard deviation. The non-linearity used between convolutional layers is maxout [11] which amounts to taking the maximum response over a number of linear models *e.g.* the maxout of two feature channels z_i^1 and z_i^2 is simply their pointwise maximum: $h_i(z_i(u,v)) = \max\{z_i^1(u,v), z_i^2(u,v)\}$. Table 1 gives the details of the layers for the model used, which is connected in the linear arrangement Conv1-Conv2-Conv3-Conv4-Softmax.

Datasets & Evaluation. The training dataset consists of 163,222 collected character samples from a number of scene text and synthesized character datasets [1, 2, 5, 15, 19, 33, 41]. The test set is the collection of 5379 cropped characters from the ICDAR 2003 training set after removing all non-alphanumeric characters as in [3, 40]. We evaluate the case-insensitive accuracy of the classifier, ignoring the background class. The Test Model achieves state-of-the-art results of 91.3% accuracy compared to the next best result of 89.8% [3].

Implementation Details. The CNN framework we use is the CPU implementation of `Caffe` [14], where convolutions are performed by constructing a matrix of filter windows of the input, `im2col`, and using BLAS for the matrix-matrix multiplication between the filters and data windows. We found this to be the fastest CPU CNN implementation attainable. CNN training is done with SGD with momentum of 0.9 and weight decay of 0.0005. Dropout of 0.5 is used on all layers except Conv1 to regularize the weights, and the learning rate is adaptively reduced during the course of training.