Overview

Introduction

Research Overview
  PhD Research
  Collaborative Research

Motivation

Structural Priors
  Spatial Structural Priors
  Filter-wise Structural Priors

Summary/Future Work
  Collaborative Research
Introduction
About Me

- Ph.D. student in the Department of Engineering at the University of Cambridge.
- Funded by a Microsoft Research PhD Scholarship
- Supervised by Professor Roberto Cipolla, head of the Computer Vision and Robotics group in the Machine Intelligence Lab, and Dr. Antonio Criminisi, a principal researcher at Microsoft Research.
Research Overview
First Author Publications during PhD

- **Decision Forests, Convolutional Networks and the Models in-Between.**
  MSR Technical Report 2015

- **Training CNNs with Low-Rank Filters for Efficient Image Classification.**
  Y. Ioannou, D. Robertson, J. Shotton, R. Cipolla, A. Criminisi.
  ICLR 2016

- **Deep roots: Improving CNN efficiency with hierarchical filter groups.**
  Y. Ioannou, D. Robertson, R. Cipolla, A. Criminisi.
  CVPR 2017

*To be presented in this talk*
Collaborative Research

- **Medical Computer Vision**
    D. Zikic, Y. Ioannou, M. Brown, A. Criminisi. *MICCAI-BRATS 2014*
  - Using CNNs for Malaria Diagnosis.
    Intellectual Ventures/Gates Foundation

- **Adversarial Examples**
  Measuring Neural Net Robustness with Constraints.

- **Neural Network Design**
  Refining Architectures of Deep Convolutional Neural Networks.
  S. Shankar, D. Robertson, Y. Ioannou, A. Criminisi, R. Cipolla. *CVPR 2016*
Motivation
ILSVRC
Imagenet Large-Scale Visual Recognition Challenge

- Imagenet Large-Scale Visual Recognition Challenge\(^2\).
- 1.2 Million Training Images, 1000 classes.
- 50,000 image validation/test set.
  - In 2012 Alex Krizhevsky won challenge with CNN\(^3\).
  - ‘AlexNet’ was 26.2% better than second best, 15.3%.
- State-of-the-art beats human error (5%).

---

\(^2\)Russakovsky et al., “ImageNet Large Scale Visual Recognition Challenge”.

\(^3\)Krizhevsky, Sutskever, and Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”.
AlexNet Complexity

- \( \approx 61 \text{ million parameters} \)
- \( \approx 724 \text{ million FLOPS (per-sample)} \)
- Imagenet has 1.28 million training samples \((227 \times 227 \times 3)\)
- Images of dimensions \((227 \times 227 \times 3) \approx 200 \text{ billion pixels}\)

\(^4\)Krizhevsky, Sutskever, and Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”. 
AlexNet Complexity - Parameters

96% in fully connected layers
The Problem

- Creating a massively over-parameterized network, has consequences
- Training time: Translates into 2-3 weeks of training on 8 GPUs! (ResNet 200)
- Forward pass (ResNet 50): 12 ms GPU, 621 ms CPU
- Forward pass (GoogLeNet): 4.4 ms GPU, 300 ms CPU

But what about the practicalities of using deep learning:

- on embedded devices
- realtime applications
- backed by distributed/cloud computing
Isn’t that already being addressed?

- Approximation (compression/pruning) of neural networks
- Reduced representation (8-bit floats/binary!)

Allow us to have a trade off in compute v.s. accuracy.

*These methods will still apply to any network.* Instead, let’s try to address the fundamental problem of over-parameterization.
Generalization and Num. Parameters

Figure: Polynomial fits of samples from a 3rd order function.

Polynomials of high order, like neural networks of many parameters, easily overfit a small number of samples as compared to polynomials of a more suitable order for the sampled function. While generalization is helped by more data, the higher order polynomial still tends to overfit.
Generalization and Num. Parameters

- When fitting a curve, we often have little idea of what order polynomial would best fit the data!
- **Weak Prior - Regularization.**
  - Prior is knowing only that our model is over-parameterized
  - This restricts the model to effectively use only a small number of the parameters
- **Strong Priors - Structural.**
  - With more prior information on the task, e.g. from the convexity of the polynomial, we may imply that a certain order polynomial is more appropriate, and restrict learning to some particular orders.
Deep networks need many more parameters than data points because they aren’t just learning to model data, but also learning what \textit{not} to learn.

Idea: Why don’t we help the network, through structural priors, not to learn things it doesn’t need to?
Structural Priors
Typical Convolutional Layer

$H \times W \times c_1$

Number of filters $c_2$

ReLU

$H \times W \times c_2$
Convolutional Neural Networks (CNNs) are structural priors for natural images

- Local connectivity - local correlations are important in natural images, e.g. edges
- Shared parameters - we know we don’t need to re-learn filters for every pixel
Why are CNNs uniformly structured?

“The marvelous powers of the brain emerge not from any single, uniformly structured connectionist network but from highly evolved arrangements of smaller, specialized networks which are interconnected in very specific ways.”

Marvin Minsky
Perceptrons (1988 edition)

- Deep networks are largely monolithic (uniformly connected), with few exceptions
- Why don’t we try to structure our networks closer to the specialized components required for learning images?
Inception: Learning a Basis for Filter Size

- In\(^5\), linear combination of different sized filters is learned, i.e. a basis space for filters:

Motivation: expect most image correlations to be highly localized, i.e. many small filters. However, a few may require larger, more complex filters.

\(^5\)Szegedy et al., “Going Deeper with Convolutions”.
Spatial Structural Priors

Published as a conference paper at ICLR 2016

TRAINING CNNS WITH LOW-RANK FILTERS FOR EFFICIENT IMAGE CLASSIFICATION

Yani Ioannou¹, Duncan Robertson², Jamie Shotton², Roberto Cipolla¹ & Antonio Criminisi²
¹University of Cambridge, ²Microsoft Research
{yai20,rc10001}@cam.ac.uk, {a-durobe,jamiesho,antcrim}@microsoft.com

ABSTRACT

We propose a new method for creating computationally efficient convolutional neural networks (CNNs) by using low-rank representations of convolutional filters. Rather than approximating filters in previously-trained networks with more efficient versions, we learn a set of small basis filters from scratch; during training, the network learns to combine these basis filters into more complex filters that
A learned basis of vertical/horizontal rectangular filters and square filters!

- Shape of learned filters is a full $w \times h \times c$.
- But what can be effectively learned is limited by the number and complexity of the components.
VGG/Imagenet Results

- Baseline Networks
- Our Results

- Multiply-Accumulate Operations

- Parameters

- Top-5 Error
Imagenet Results

- VGG-11 (low-rank): 24% smaller, 41% fewer FLOPS
- VGG-11 (low-rank/full-rank mix): 16% fewer FLOPS with 1% lower error on ILSRVC val, but 16% larger.
- GoogLeNet: 41% smaller, 26% fewer FLOPS

Or better results if you tune it on GoogLeNet more…
1. Introduction

Since the 2012 ImageNet competition [16] winning entry by Krizhevsky et al. [9], their network “AlexNet” has represented a significant leap in performance, where their model achieved top-1 error of 15.3% and top-5 error of 6.7% on the ILSVRC 2012 validation set. These successes spurred a new line of research that focused on finding higher performing convolutional neural networks. Starting in 2014, the quality of network architecture has improved significantly, which has led to the emergence of efficient convolutional architecture as well, widening the efficiency gap again.

Still, the complexity of the Inception architecture makes training these methods a challenge for most benchmarks. Although increased model size and computational cost tend to translate to immediate quality gains in a wide variety of application domains, the added computation as efficiently as possible by suitably constraining the added operations via computational tricks [10]. However, these methods add extra complexity. Furthermore, these constraints on memory and computational budget. For example, GoogleNet employed only 5 million parameters, which represented a $12 \times$ reduction with respect to its predecessor AlexNet, which used 60 million parameters. Furthermore, VGGNet employed about 3x more parameters than AlexNet.

The computational cost of Inception is also much lower than VGGNet or its higher performing successors [6]. This has made it feasible to utilize Inception networks in big-data scenarios [17], [13], where huge amount of data needed to be processed at reasonable cost or scenarios where memory or computational capacity is inherently limited, for example, mobile vision and big-data scenarios [17], [13], where huge amount of data needed to be processed at reasonable cost or scenarios where memory is severely limited.

Although VGGNet [18] has the compelling feature of computational cost of 5 billion multiply-adds per inference and with using less than 25 million parameters. With an ensemble of 4 models and multi-crop evaluation, we report 3.5% top-5 error and 17.3% top-1 error.
Filter-wise Structural Priors

Deep Roots:
Improving CNN Efficiency with Hierarchical Filter Groups

Yani Ioannou\textsuperscript{1}  Duncan Robertson\textsuperscript{2}  Roberto Cipolla\textsuperscript{1}
Antonio Criminisi\textsuperscript{2}
\textsuperscript{1}University of Cambridge, \textsuperscript{2}Microsoft Research

Abstract

We propose a new method for creating computationally efficient and compact convolutional neural networks.
Typical Convolutional Layer

- \( H \) \( W \) \( c_1 \)
- \( c_2 \) filters
- \( h_1 \) \( w_1 \) \( C_1 \)
- ReLU
- image/feature map
- filter
- output featuremap

\( H \) \( W \) \( c_2 \)
Uses 2 filter groups in most of the convolutional layers
Allowed training across two GPUs (model parallelism)
Grouped Convolutional Layer

\[ H \times W \times c_1 \rightarrow c_2 \text{ filters} \times c_2/g \rightarrow \text{ReLU} \rightarrow H \times W \times c_2 \]

- \( H \times W \times c_1 \): image/feature map
- \( c_2 \) filters
- \( c_2/g \times \): convolution
- ReLU
- \( H \times W \times c_2 \): output feature map
Root Modules

Root-2 Module: \(d\) filters in \(g = 2\) filter groups, of shape \(h \times w \times c/2\).

Root-4 Module: \(d\) filters in \(g = 4\) filter groups, of shape \(h \times w \times c/4\).
Network-in-Network

input  conv1a  conv1b  conv1c

...  ...  ...

conv2a  conv2b  conv2c

...  ...  ...

conv3a  conv3b  conv3c  pool  output
NiN Root Architectures

root-4 module

root-2 module
**Network-in-Network.** Filter groups in each convolutional layer.

<table>
<thead>
<tr>
<th>Model</th>
<th>conv1</th>
<th></th>
<th></th>
<th>conv2</th>
<th></th>
<th></th>
<th>conv3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$a$</td>
</tr>
<tr>
<td>Orig.</td>
<td></td>
<td>5×5</td>
<td>1×1</td>
<td>1×1</td>
<td>5×5</td>
<td>1×1</td>
<td>1×1</td>
</tr>
<tr>
<td>root-2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>root-4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>root-8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>root-16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
CIFAR10: Model Parameters v.s. Error

NiN: mean and standard deviation (error bars) are shown over 5 different random initializations.
CIFAR10: FLOPS (Multiply-Add) v.s. Error.

NiN: mean and standard deviation (error bars) are shown over 5 different random initializations.
Figure: The block-diagonal sparsity learned by a root-module is visible in the correlation of filters on layers conv3a and conv2c in the NiN network.
Imagenet Results

Networks with root modules have similar or higher accuracy than the baseline architectures with much less computation.

- ResNet 50$^6$: 40% smaller, 45% fewer FLOPS
- ResNet 200$^7$: 44% smaller, 25% fewer FLOPS
- GoogLeNet: 7% smaller, 44% fewer FLOPS

But when you also increase the number of filters...
Figure 1. Left: A block of ResNet [13]. Right: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

“Moreover, increasing cardinality is more effective than going deeper or wider when we increase the capacity.”
Summary/Future Work
Summary

- Using structural priors:
  - Models are *less computationally complex*
  - They also use *less parameters*
  - They significantly help generalization in *deeper networks*
  - They significantly help generalization with *larger datasets*

- Are amenable to *model parallelization* (as with original AlexNet), for better parallelism across gpus/nodes
Future Work: Research

- We don’t always have enough knowledge of the domain to propose good structural priors
- Our results (and follow up work) do show however that current methods of training/regularization seem to have limited effectiveness in DNNs learning such priors themselves
- How can we otherwise learn structural priors?
Both of these methods apply to most deep learning applications:

- Smaller model state – easier storage and synchronization
- Faster training and test of models behind ML cloud services
- Embedded devices/Tensor processing units

And more specific to each method

- Low-rank filters
  - Even larger impact for volumetric imagery (Microsoft Radiomics)
- Root Modules
  - Model parallelization (Azure/Amazon Cloud)