Challenges in Applied DL

Yangqing Jia

Facebook

Rich feature hierarchies for accurate object detection and semantic segmentation

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

Abstract

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method R-CNN: *Regions with CNN features. Source code for the complete system is available at* http://www.cs.berkeley.edu/~rbg/rcnn.

1. Introduction

Features matter. The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT [27] and HOG [7]. But if we look at performance on the canonical visual recognition task, PASCAL VOC object detection [13], it is generally acknowledged that progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods.

SIFT and HOG are blockwise orientation histograms, a representation we could associate roughly with complex cells in V1, the first cortical area in the primate visual pathway. But we also know that recognition occurs several stages downstream, which suggests that there might be hierarchical, multi-stage processes for computing features that are even more informative for visual recognition.

Fukushima's "neocognitron" [17], a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process. The neocognitron, however, lacked a supervised training algorithm. Building on Rumelhart *et al.* [30], LeCun *et al.* [24] showed that stochastic gradient descent via backpropagation was effective for training convolutional neural networks (CNNs), a class of models that extend the neocognitron.

CNNs saw heavy use in the 1990s (*e.g.*, [25]), but then fell out of fashion with the rise of support vector machines. In 2012, Krizhevsky *et al.* [23] rekindled interest in CNNs by showing substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9, 10]. Their success resulted from training a large CNN on 1.2 million labeled images, together with a few twists on LeCun's CNN (*e.g.*, $\max(x, 0)$ rectifying non-linearities and "dropout" regularization).

The significance of the ImageNet result was vigorously debated during the ILSVRC 2012 workshop. The central issue can be distilled to the following: To what extent do the CNN classification results on ImageNet generalize to object detection results on the PASCAL VOC Challenge?

We answer this question by bridging the gap between image classification and object detection. This paper is the first to show that a CNN can lead to dramatically higher object detection performance on PASCAL VOC as compared to systems based on simpler HOG-like features. To achieve this result, we focused on two problems: localizing objects

Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [34] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%.

Source: Girshick et al. R-CNN, PhD Comic

The Needs

Two sides of the same coin

- Researchers

  ″I need to design flexible ML algorithms.″

  ″I need to reproduce that ResNet paper.″

- Companies

  ″Hey, I need to apply DL to my applications.″

Industry:

Stability

Scale & speed

Data Integration

Relatively Fixed

Research:

Flexible

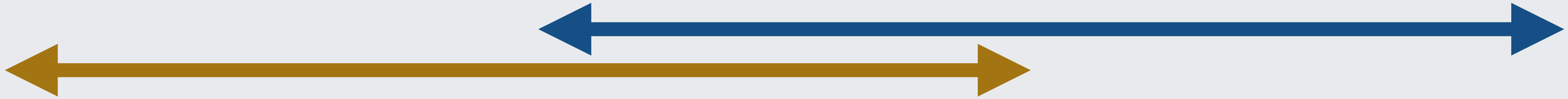
Fast Iteration

Debuggable

Relatively barbone

Caffe2

PyTorch



Industry:

Stability

Scale & speed

Data Integration

Relatively Fixed

Research:

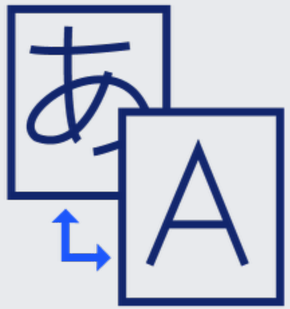
Flexible

Fast Iteration

Debuggable

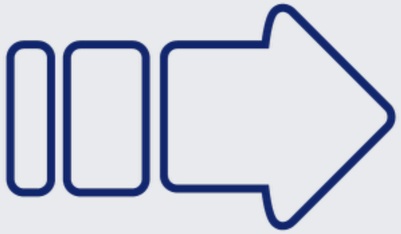
Relatively barbone

VISION

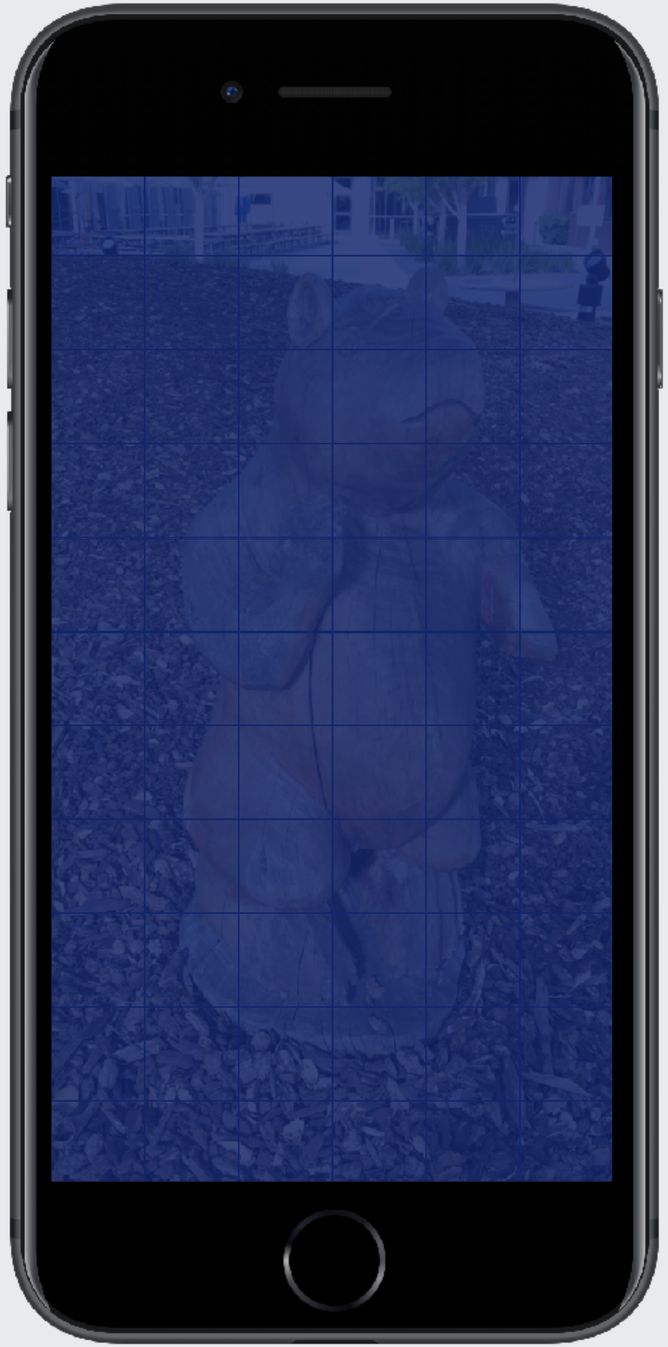


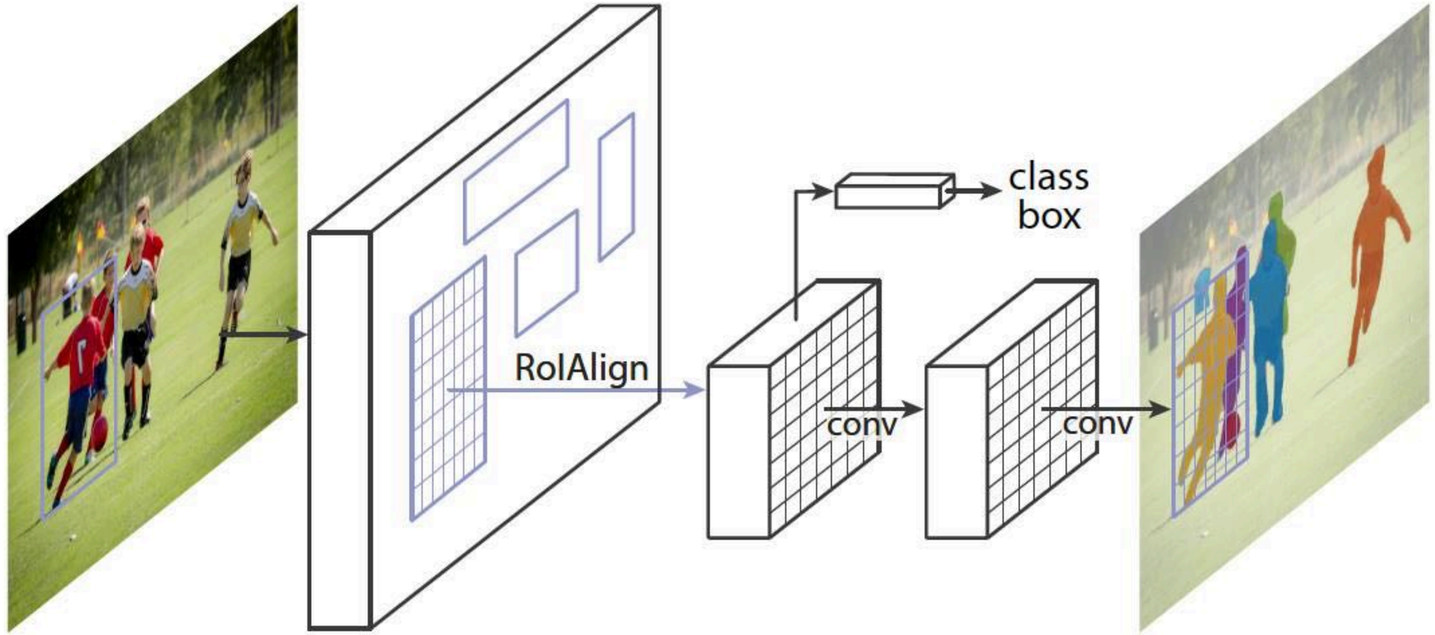
TRANSLATION

SPEECH

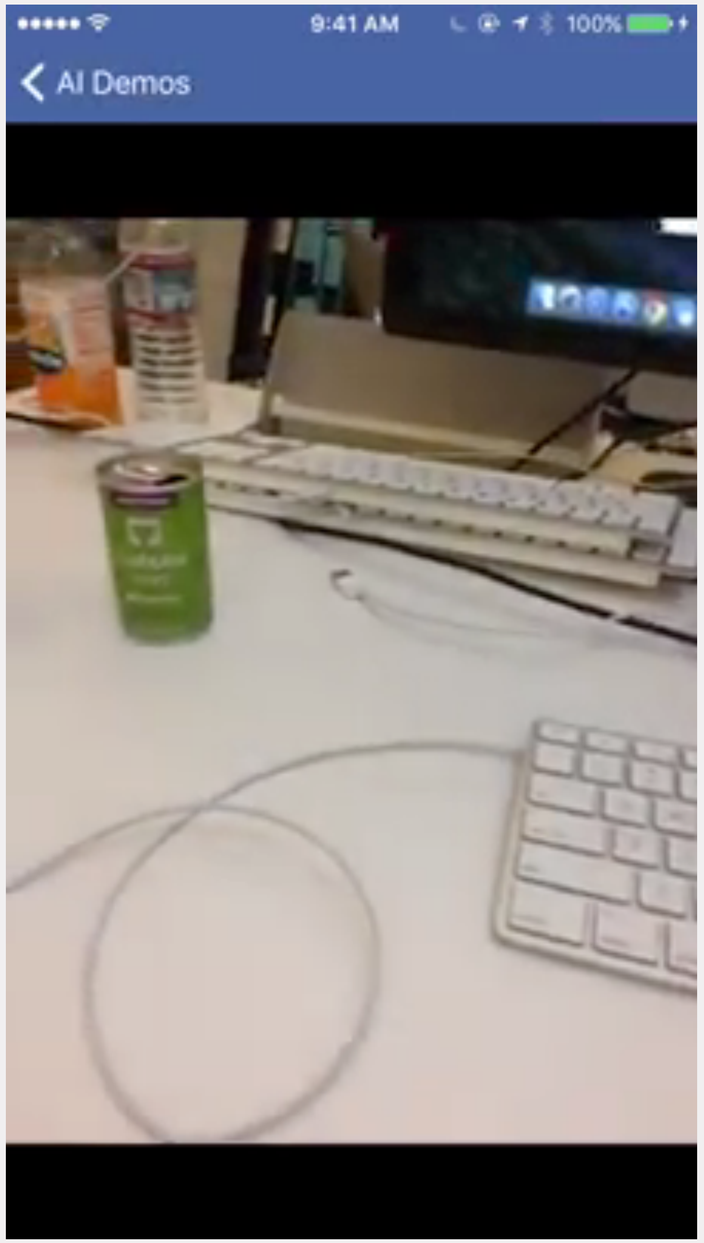


RANKING




RoIAlign
class
box
conv
conv

Things that Matters


9:41 AM
100%
AI Demos

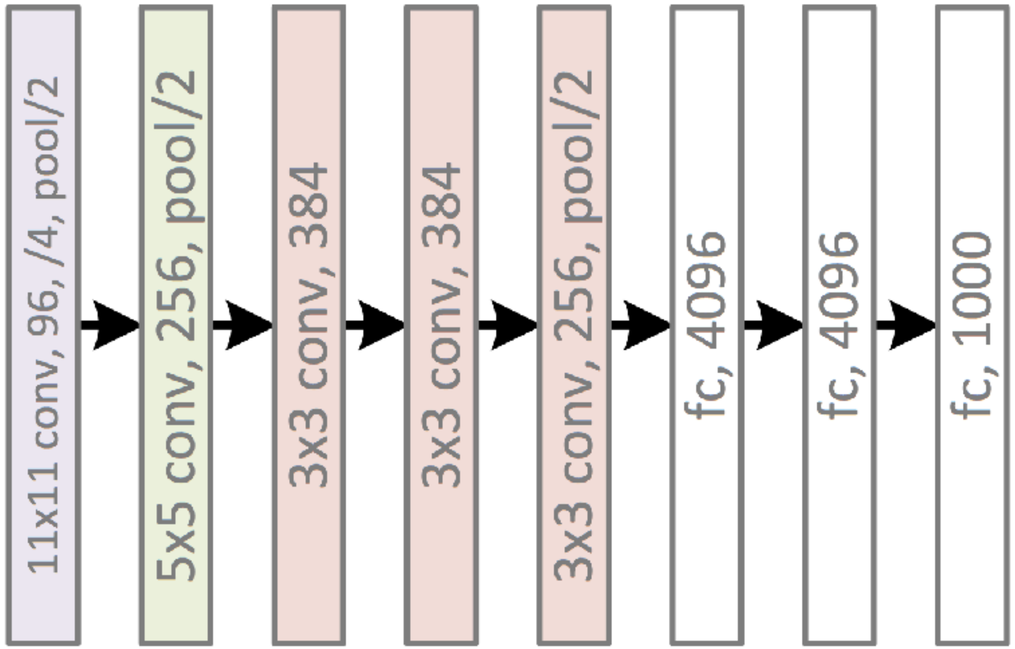
Finding better model architecture

Numerical optimization

Model Compression

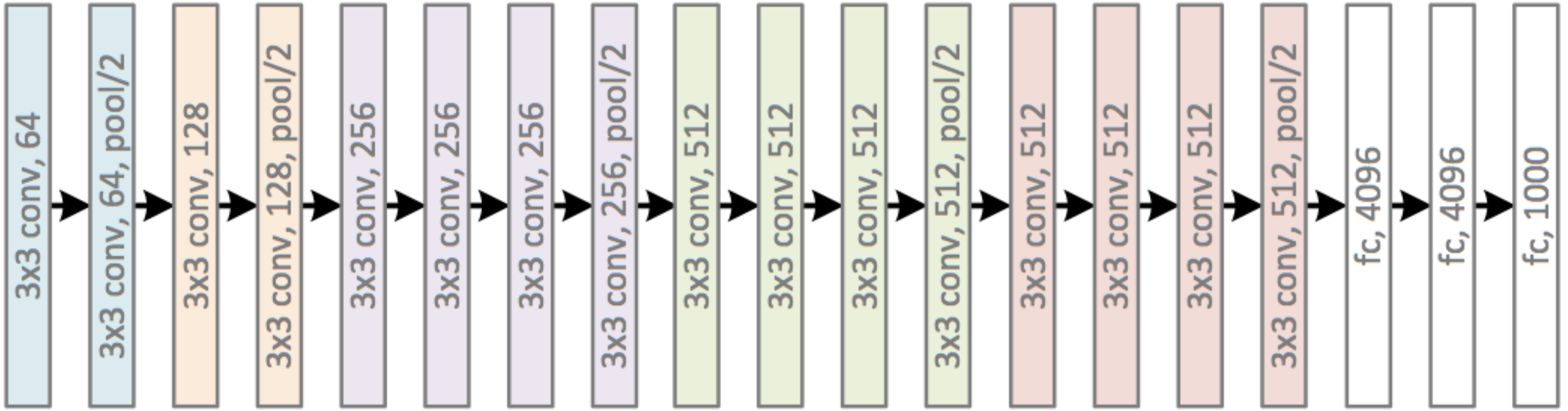
AlexNet

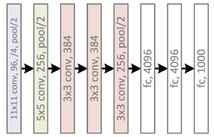
So it begins.



VGGNet

Punch it.

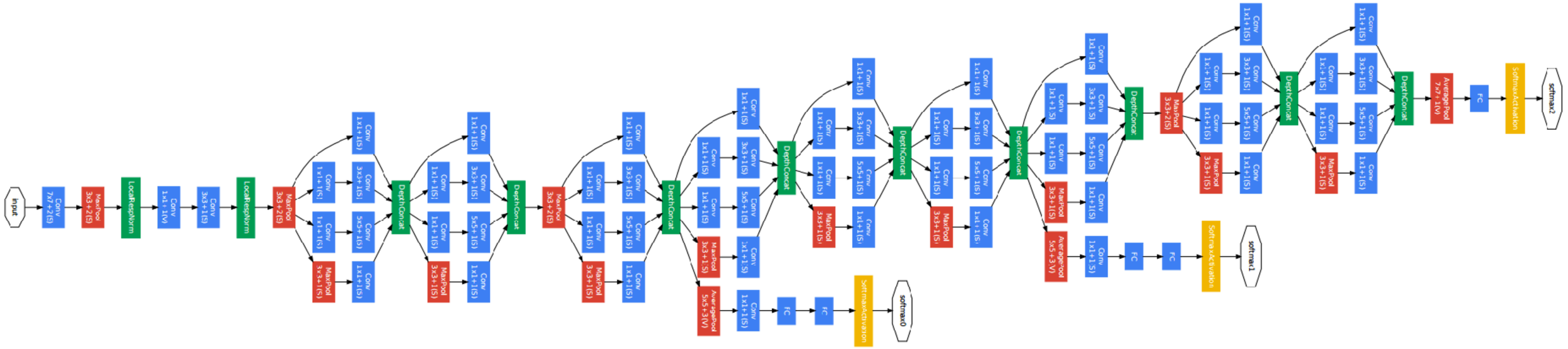


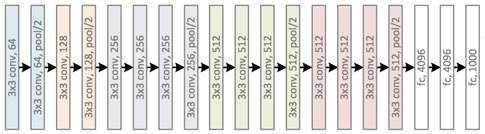


<- AlexNet

GoogLeNet

We must go deeper.

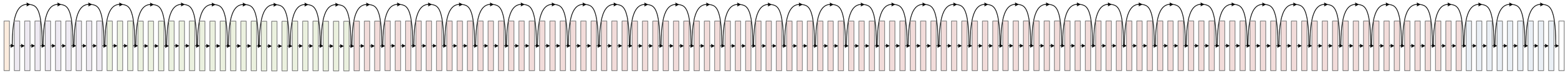


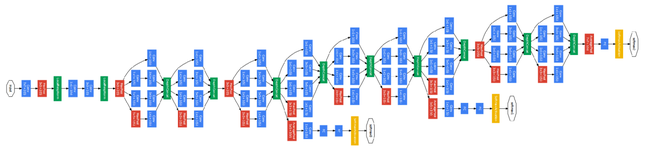


<- VGG

ResNet

And we took the word seriously

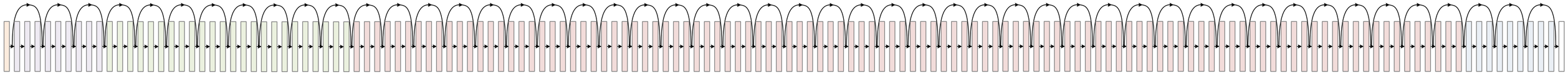


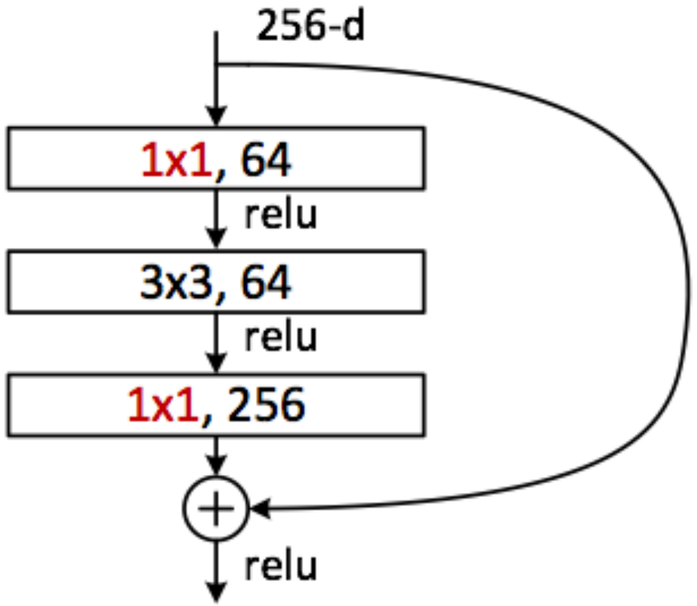


<- GoogLeNet

ResNet

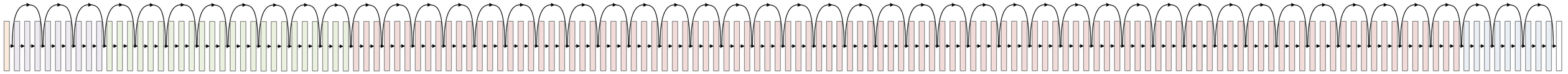
And we took the word seriously

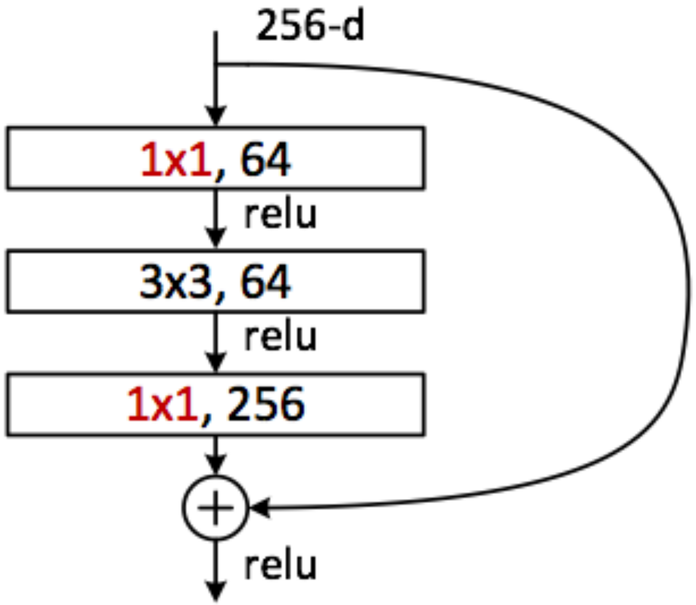


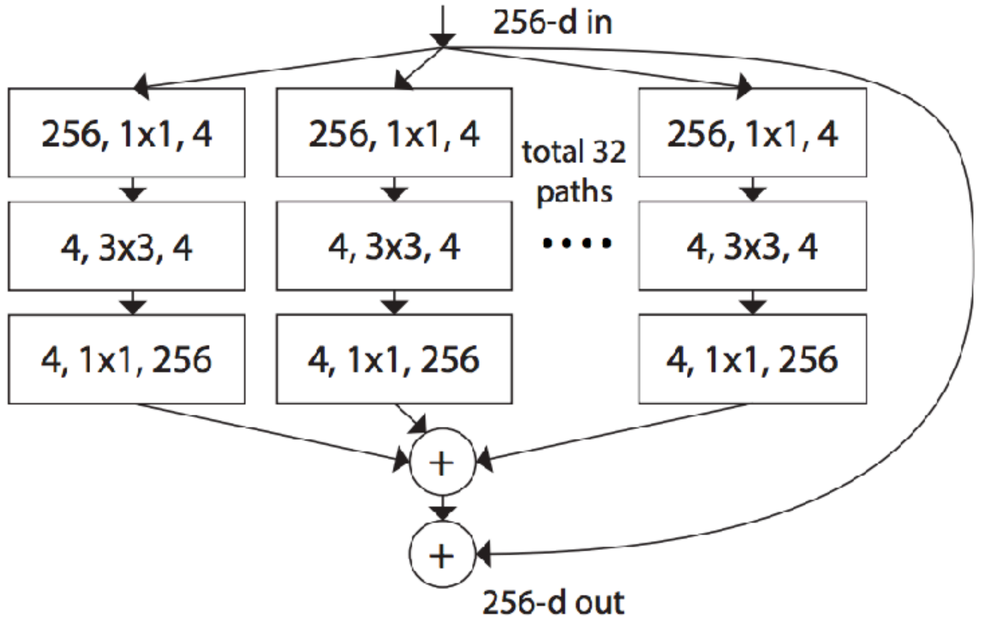


ResNeXT

We totally see it coming





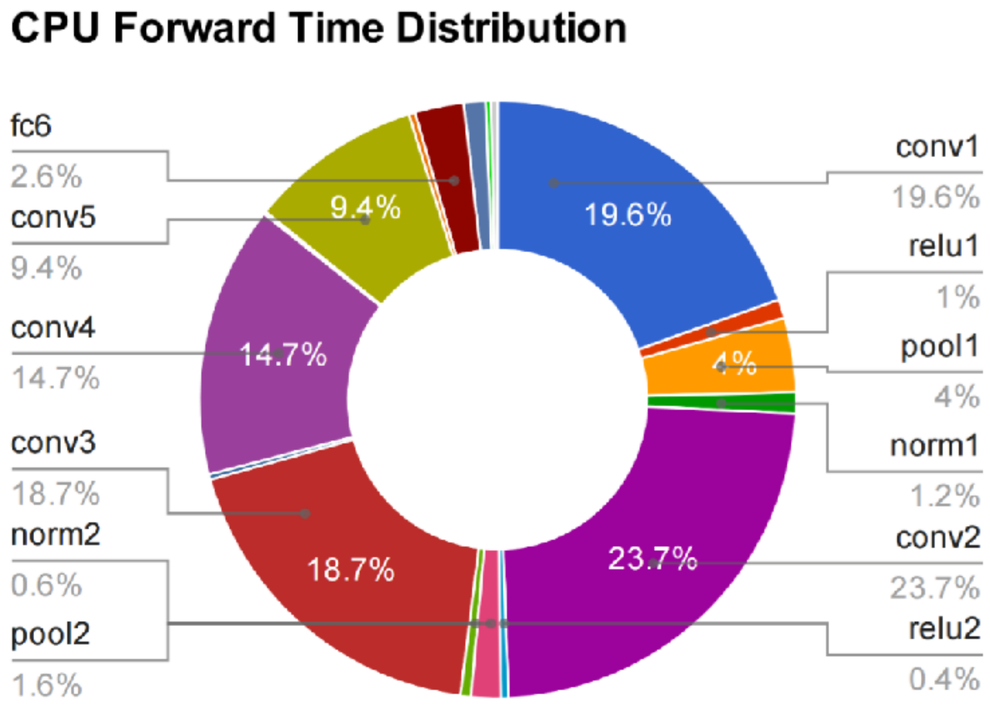


Practical Model Selections

- Go deeper & go structured
- No need to go very large layers
  - It only takes 8-16 channels for style transfer
- Model depth matters
  - (almost) never need to do more than 3x3 filters
- Conscious model search for performance balances
  - but, with ″grad student decay″

Numeric Optimizations

GEMM/Conv is usually at the core of CNNs



The Conv Math

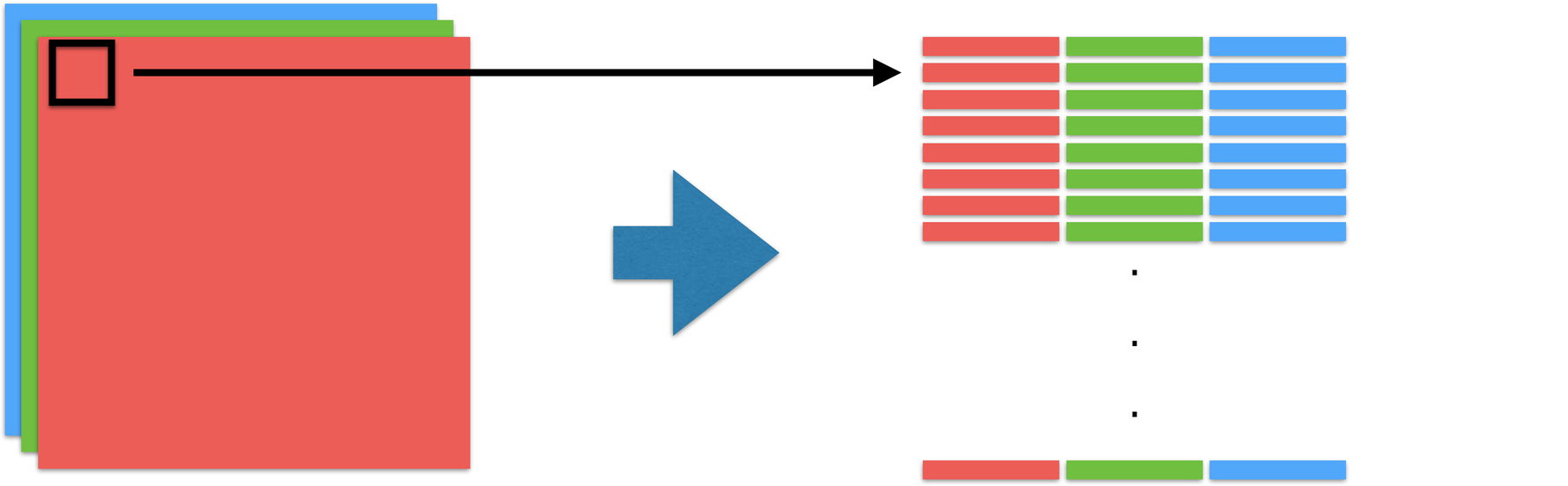
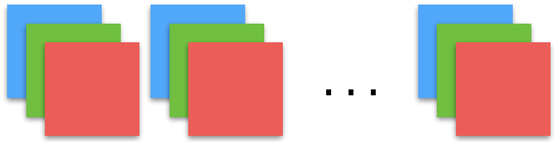
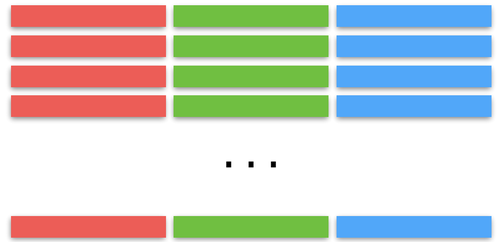


Image: CxHxW

Feature Matrix: (HxW) x (CxKxK)

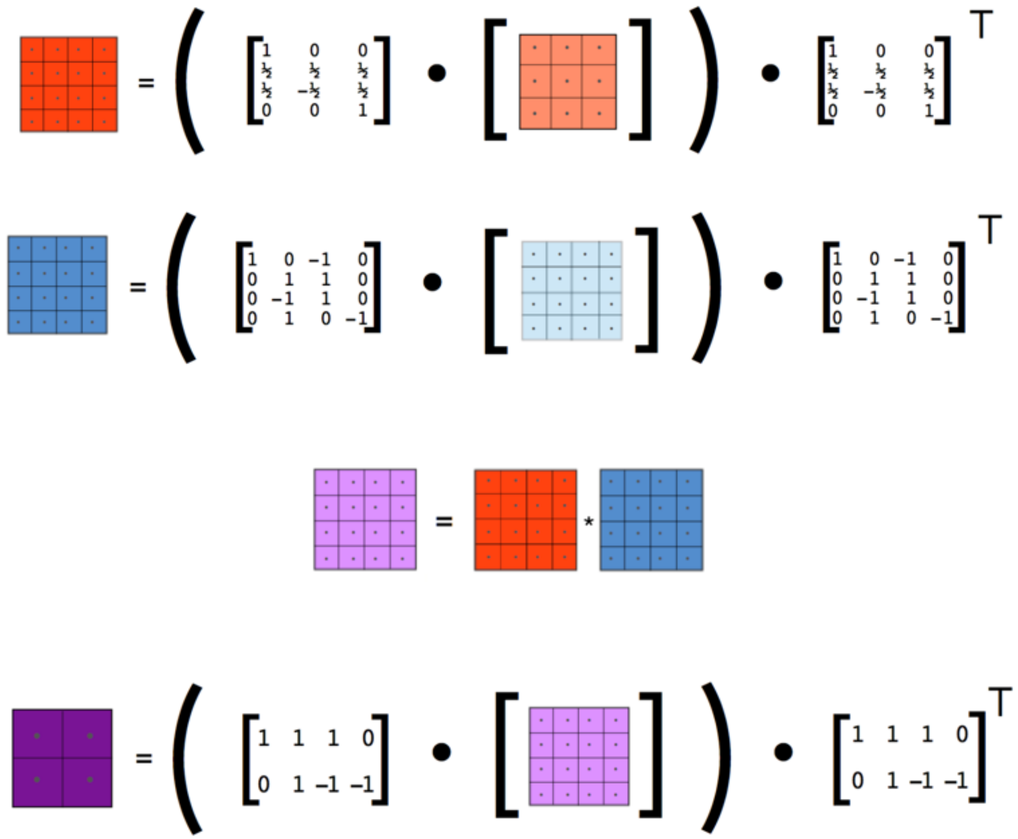


Filters: C_{out}xCxKxK



Filter Matrix: C_{out} x (CxKxK)

The Conv Math - Winograd

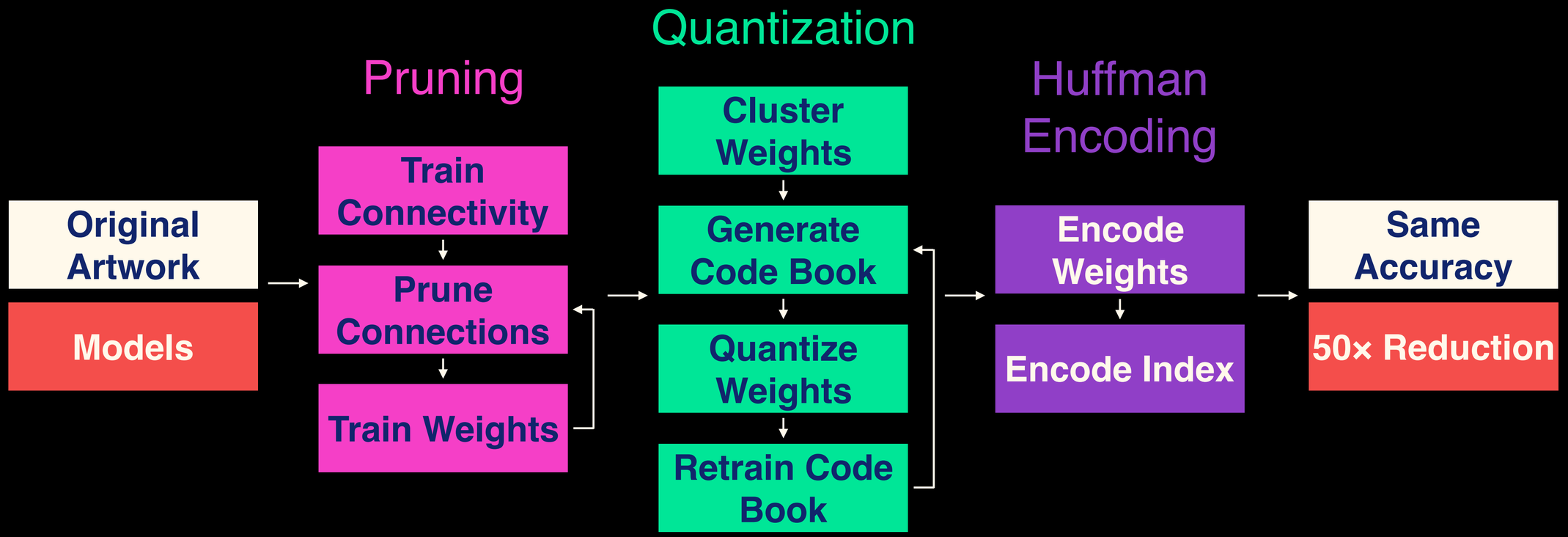


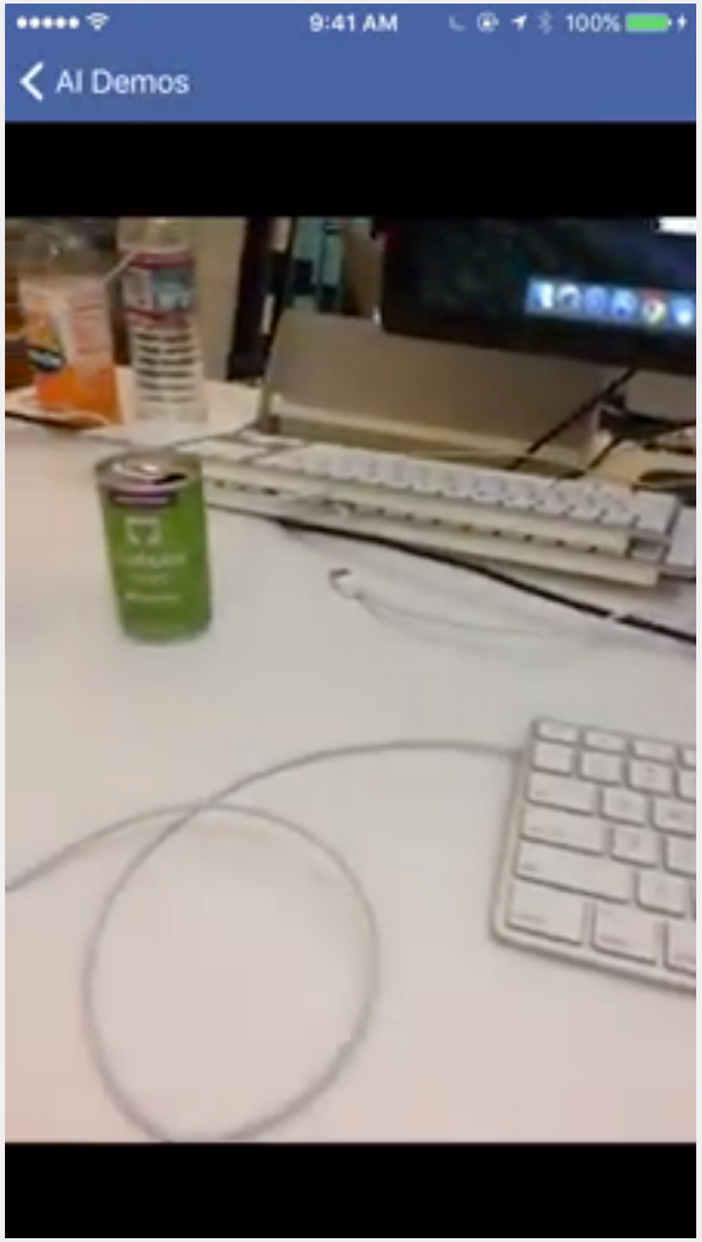
Layer	im2col	Winograd
AlexNet:conv2	315	86
AlexNet:conv3	182	44
AlexNet:conv4	264	56
AlexNet:conv5	177	40

(with NNPACK)

Source: https://www.nervanasys.com/winograd-2/, https://github.com/Maratyszcza/NNPACK

Model Compression




9:41 AM
100%
AI Demos

Food for Thought

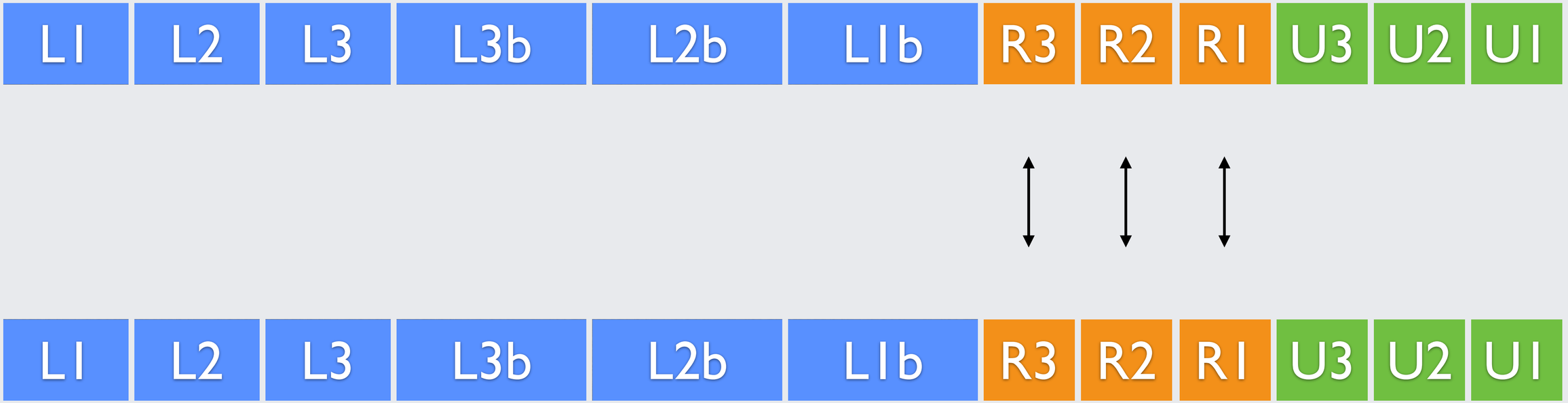
The Return of ~~Jedi~~ MPI

- MPI is actually a very good abstraction for ML
- Especially from a sync SGD perspective
- Most existing training algorithms adopt an MPI-like fashion

The Return of ~~Jedi~~ MPI

The Return of ~~Jedi~~ MPI

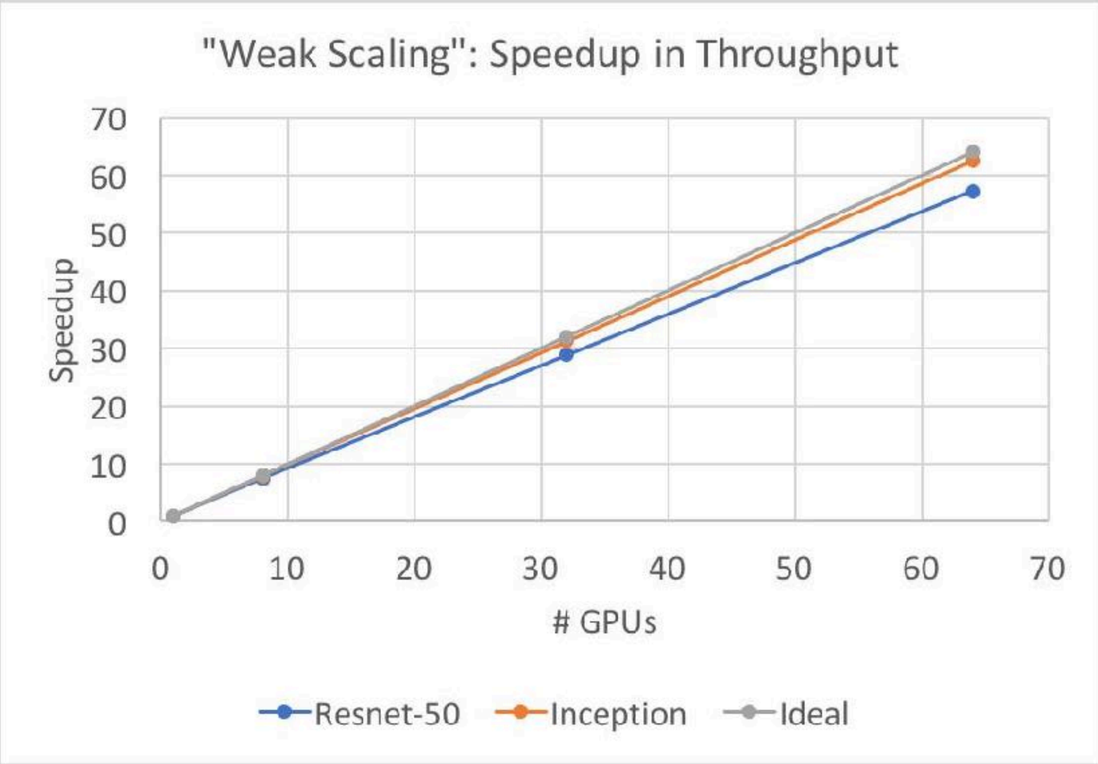
The Return of ~~Jedi~~ MPI



The Return of ~~Jedi~~ MPI

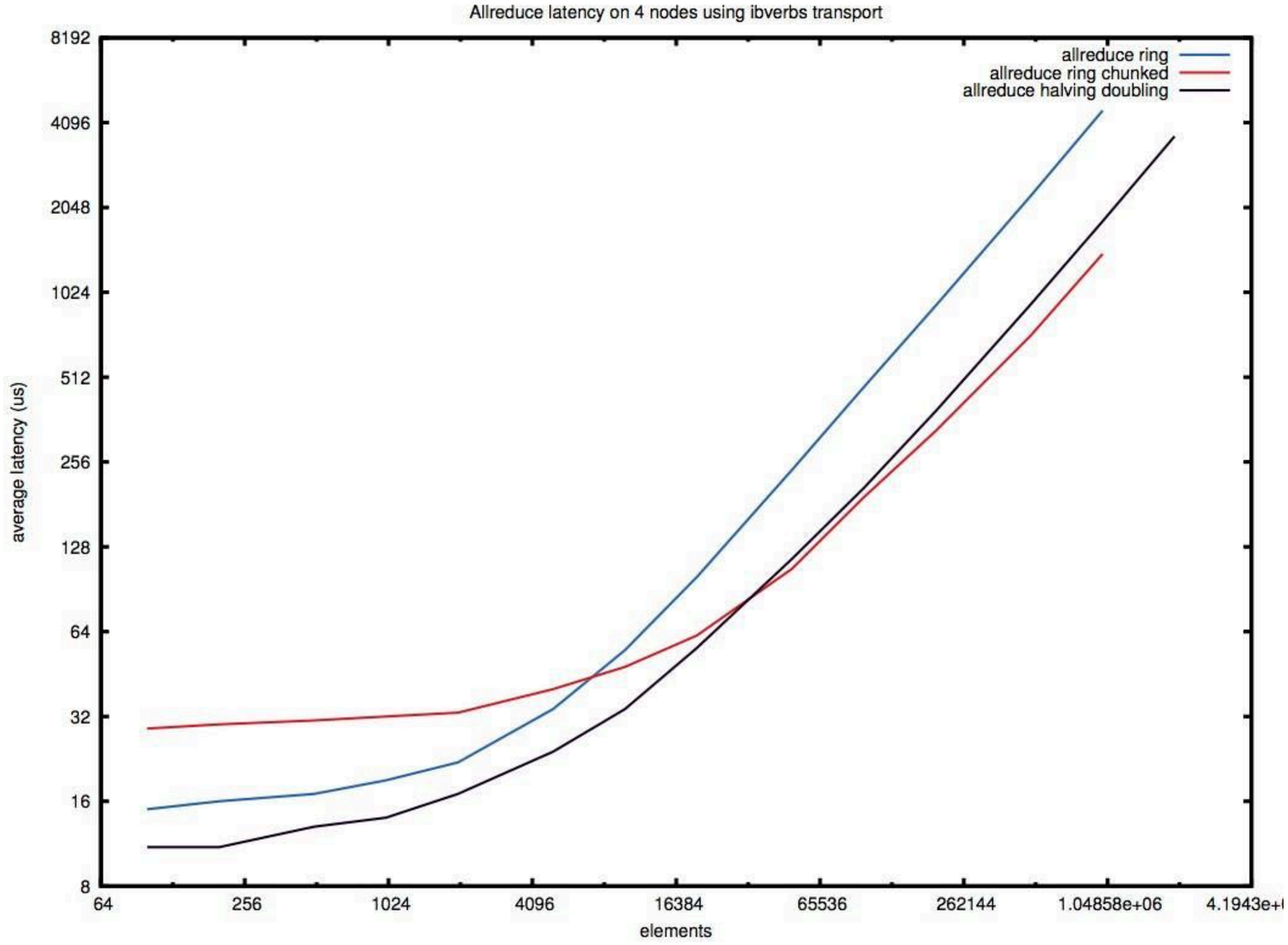
Gloo

MPI like library without the trouble of MPI



Pieter Noordhuis
(Also creator of hiredis)

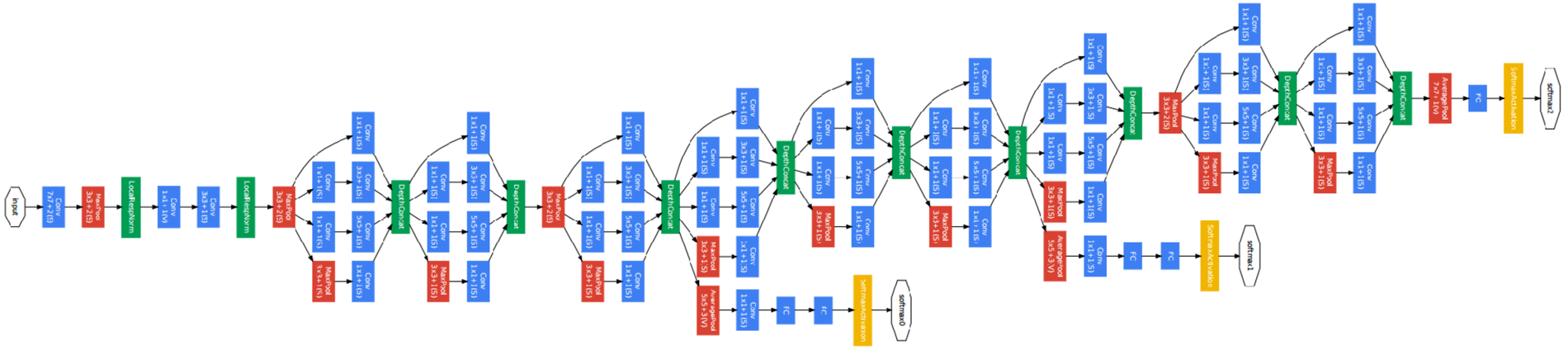
Source: https://github.com/facebookincubator/gloo



More: http://www.mcs.anl.gov/~thakur/papers/ijhpca-coll.pdf

Compiler?

- We are running an interpreter these days.

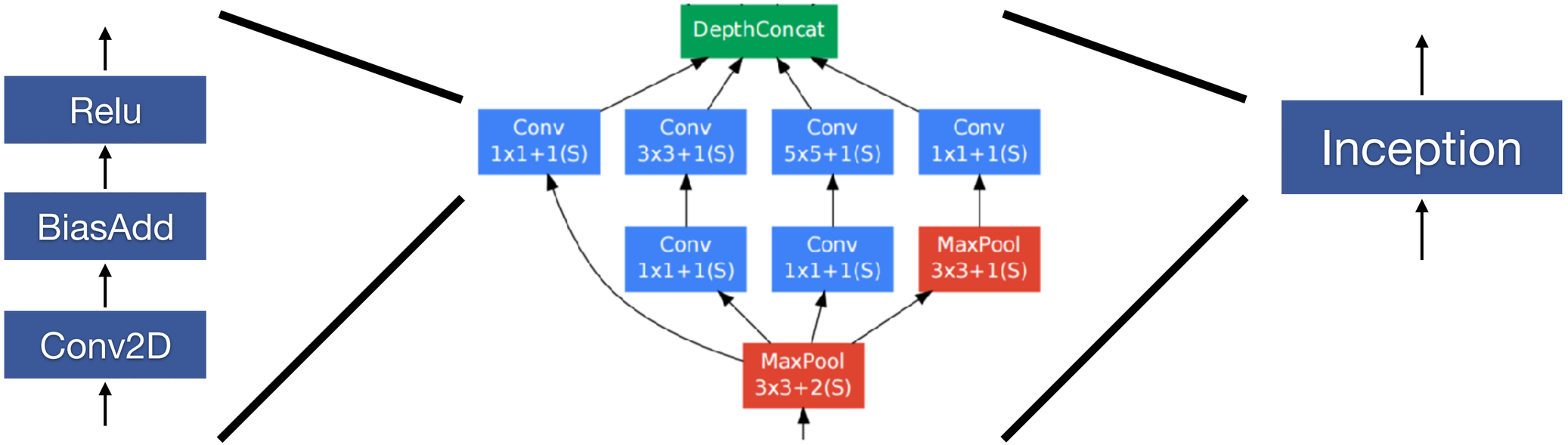


Interpreters are good and bad

- They allow easy, imperative computations
- But, they miss a lot of optimization opportunities
  - Inlining computation of multiple operators
  - Optimizing away common expressions
  - Memory optimizations

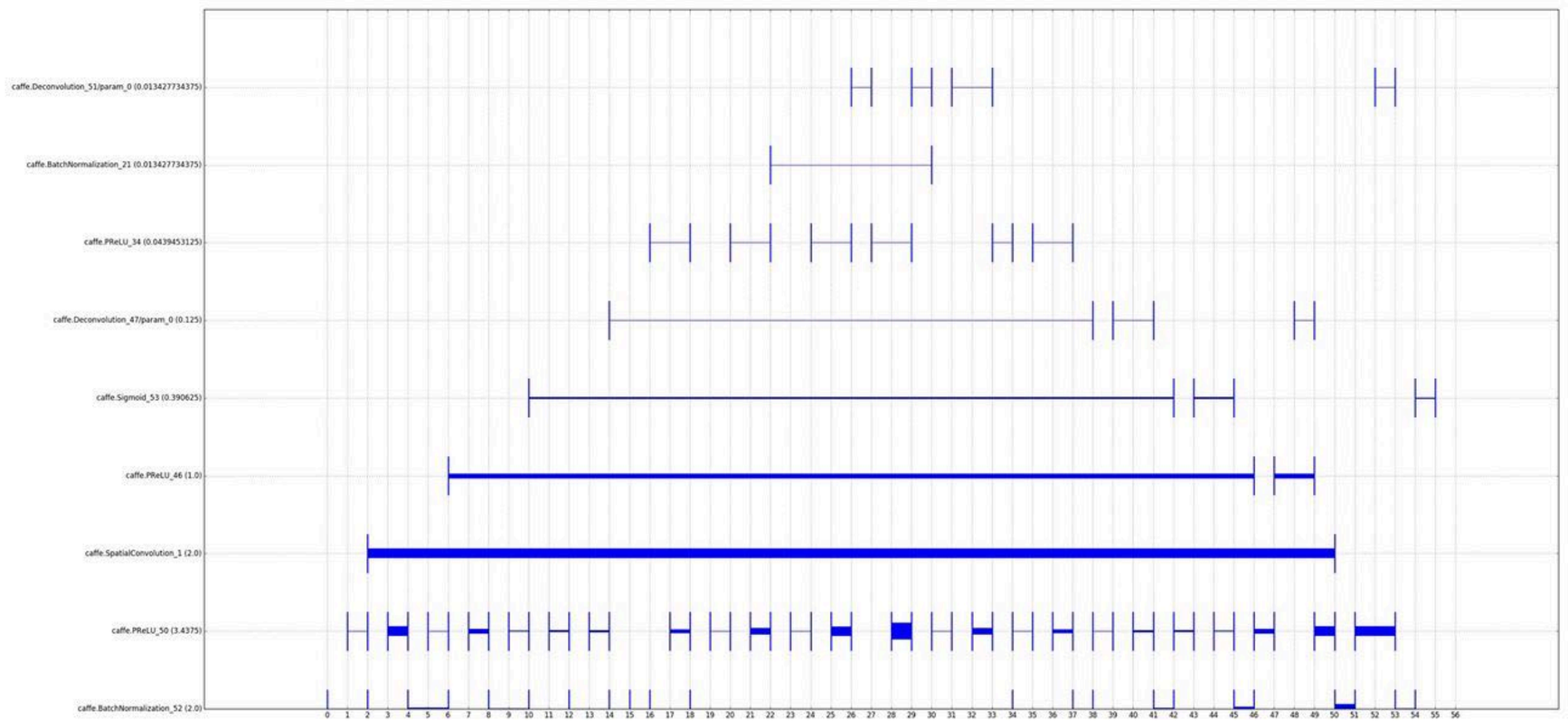
Compiler Usage #1

Graph parsing & rewriting



Compiler Usage #2

Memory optimizations



More: https://github.com/caffe2/caffe2/blob/master/caffe2/python/memonger.py

Compiler Usage #3

Code generator

- Current sparsification do NOT bring performance boosts
  - The uncanny valley of "semi-sparse"
- Model **and** parameters are "frozen" once trained
  - Generate code based on such frozen models

An overly simplified example

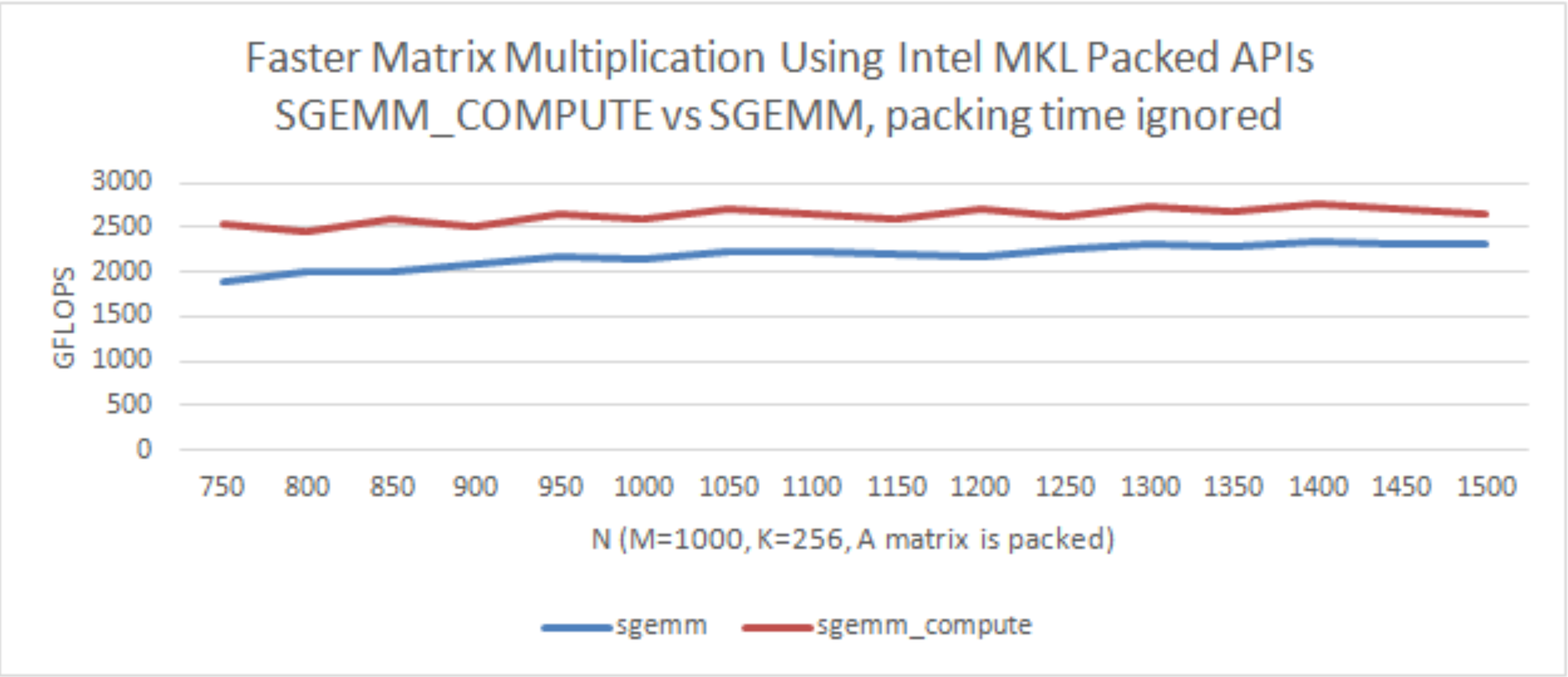
0	0.2	0.5
0.1	0.9	0
0	0.4	0

```
# naive
for i in 0..3
  for j in 0..3
    apply(i,j,f(i,j))

# zero aware
# but not efficient
for i in 0..3
  for j in 0..3
    if (f(i,j))
      apply(i,j,f(i,j))
```

```
# zero aware
# autogen code
apply(0, 1, 0.2)
apply(0, 2, 0.5)
apply(1, 0, 0.1)
apply(1, 1, 0.9)
apply(2, 1, 0.4)
```

A working example: sgemm_pack



More: https://github.com/caffe2/caffe2/blob/master/caffe2/mkl/operators/packed_fc_op.cc

Conclusion

Model Architecture

Optimization

Model Compression

MPI

Compilers

How to use the Computer science conventional wisdom to do better machine learning?

Thank you!

Yangqing Jia, jiayq@eecs.berkeley.edu