Challenges in Applied DL

Yangqing Jia
Facebook
Rich feature hierarchies for accurate object detection and semantic segmentation

Implement it!
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
Industry:
- Stability
- Scale & speed
- Data Integration
- Relatively Fixed

Research:
- Flexible
- Fast Iteration
- Debuggable
- Relatively barebone
Things that Matters
Finding better model architecture
Numerical optimization
Model Compression
AlexNet

So it begins.
VGGNet

Punch it.

<- AlexNet
GoogLeNet
We must go deeper.
ResNet
And we took the word seriously
ResNet

And we took the word seriously
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: \( C \times H \times W \)

Feature Matrix: \( (H \times W) \times (C \times K \times K) \)

Filters: \( C_{out} \times C \times K \times K \)

Filter Matrix: \( C_{out} \times (C \times K \times K) \)
The Conv Math - Winograd

\[
\begin{align*}
\text{Layer} & \quad \text{im2col} & \quad \text{Winograd} \\
\text{AlexNet:conv2} & \quad 315 & \quad 86 \\
\text{AlexNet:conv3} & \quad 182 & \quad 44 \\
\text{AlexNet:conv4} & \quad 264 & \quad 56 \\
\text{AlexNet:conv5} & \quad 177 & \quad 40 \\
\end{align*}
\]

(with NNPACK)

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

Original Artwork
- Train Connectivity
  - Prune Connections
    - Train Weights

Pruning
- Train Connectivity
- Prune Connections
- Train Weights

Quantization
- Cluster Weights
  - Generate Code Book
    - Quantize Weights
      - Retrain Code Book

Huffman Encoding
- Encode Weights
  - Encode Index

Same Accuracy
- 50x Reduction
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
• 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

Diagram showing graph parsing & rewriting with nodes labeled Conv2D, BiasAdd, Relu, DepthConcat, etc., leading to an Inception node.
Compiler Usage #2
Memory optimizations

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

# 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
# autogenerated 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

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

Yangqing Jia, jiayq@eecs.berkeley.edu