E6893 Big Data Analytics: Lecture 13

Cognitive Mobile Analysis

Ching-Yung Lin, Ph.D.
IBM Chief Scientist, Graph Computing
Adjunct Professor, Dept. of Electrical Engineering and Computer Science

December 8th, 2016
Advanced Big Data Analytics Projects

If you are interested in doing advanced research in Spring 2017 and beyond in the following areas, please let me know (or contact the researchers in the next few pages) and probably take the Advanced Big Data Analytics class as your course project:

• Large-Scale Graph Analysis
• Machine Reasoning, including Bayesian Networks and Game Theory
• Mobile Vision
• Robots — Vision and Cognitive Interaction
• Large-Scale Visualization
• Deep Learning on Mobile Devices
• etc.. (As in the next few slides)
Contact: Toyo Suzumura, tsuzumrua@us.ibm.com

Project Description
This project focuses on how deep learning could be applied to the graph classification method or finding anomalous nodes in graphs – especially a real-world massive graph with properties and time-evolving characteristics.

Required Skills: Experience on any deep learning framework such as TensorFlow etc.

Big Graph (e.g. Transaction Graph in the Financial Domain)
Potential Project Directions

- Contact: Dr. Guangnan Ye, gye@us.ibm.com
- Projects:
  - Project 1: Robot Intelligence
    - Description: Improving computer vision and speech recognition capability on robots — Nao, Pepper, etc.
    - Prior experience required: Knowledge on computer vision and machine learning.
  - Project 2: Content analysis and verification from web resources
    - Description: Given the entity of a product\person\company, determine the reasonable content based on information sources available from the Internet.
    - Prior experience required: Knowledge on web crawler, NLP, and machine learning
  - Project 3: Recognition in computer vision
    - Description: OCR, Face Analysis, Object Recognition, Event Recognition, etc.
    - Prior experience required: Knowledge on computer vision and machine learning.
Potential Project Directions

- Contact: Dr. Conglei Shi, shiconglei@us.ibm.com
- Projects:
  - **Interactive Machine Learning**
    - Description: In this project, we want to build a visual analytic system to better understand the machine learning model, explore the training and testing process, and improve the performance.
    - Prior experience required: Experience on machine learning and familiar with at least one machine learning toolkit
Potential Project Directions

- Contact: Jason Crawford, ccjason@us.ibm.com
- Projects:
  - **Graph Query Language for SQL users**
    - Description: In this project, we want to build a novel graph query language that is applicable for all kinds of applications as well as friendly with traditional SQL users
    - Prior experience required: Relational Database, Graph Database,
Potential Project Directions

- Contact: Danny Yeh, dlyeh@us.ibm.com
- Projects:
  - **Robo-Advisor for Wealth Management**
    - Description: In this project, we want to test various strategies that are related to building a Robo-Advisor which involves 4 steps of wealth management: financial data gathering and processing, person / user understanding, personal portfolio management, and recommendation for changes
  - Prior experience required: Big Data Analytics
Network / Graph is the way we remember, we associate, and we reason.
“IBM System G”: a brand name approved by HQ — April 2014.

Based on 30+ graph-related projects; 150+ papers; ~40 patents; ~10 best paper awards; ~$25M Research funding

Accomplishments in 2013 ($100M+ contribution - SmallBlue) and 2014 (Scientific contribution — Social & Cognitive Network Science)

Connecting the Dots and Reasoning Big Data

http://systemg.research.ibm.com
System G Tools provide Building Blocks for Cognitive Solutions

- **Graph Middleware:**
  - Parallel Prog. Lib.
  - Power Optimization
  - GPU Optimization

- **Graph Analytics:**
  - Topological Analysis
  - Matching and Search
  - Path and Flow

- **Graph Database:**
  - Native Store
  - GBase

- **Graph Visualization:**
  - Multivariate Graph
  - Dynamic Graph
  - Big Graph

- **Machine Learning:**
  - Deep Learning Tools
  - Visual and Text Sentiment Tools
  - Anomaly Detection Tools

- **Mobile Cognition:**
  - iOS Cognition Tools
  - Robot Cognition Tools

- **Machine Reasoning:**
  - Bayesian Networks
  - Game Theory Tools
  - Multimodal Analysis Platform

- **Spatiotemporal Analytics:**
  - Spatiotemporal Mining
  - Spatiotemporal Indexing

- **Graph Database Technologies:**
  - Native Store
  - GBase

- **Graph Visualization Technologies:**
  - Multivariate Graph
  - Dynamic Graph
  - Big Graph

- **Network Analytics Technologies:**
  - Topological Analysis
  - Matching and Search
  - Path and Flow

- **Machine Learning Technologies:**
  - Deep Learning Tools
  - Visual and Text Sentiment Tools
  - Anomaly Detection Tools

- **Mobile Cognition Technologies:**
  - iOS Cognition Tools
  - Robot Cognition Tools

- **Spatiotemporal Analytics Technologies:**
  - Spatiotemporal Mining
  - Spatiotemporal Indexing

- **Machine Reasoning Technologies:**
  - Bayesian Networks
  - Game Theory Tools
  - Multimodal Analysis Platform

- **IBM System G**
IBM System G Application Use Cases

1. System G for Expertise Location
2. System G for Recommendation
3. System G for Commerce
4. System G for Financial Analysis
5. System G for Social Media Monitoring
6. System G for Telco Customer Analysis
7. System G for Watson
8. System G for Data Exploration and Visualization
9. System G for Personalized Search
10. System G for Anomaly Detection (Espionage, Sabotage, etc.)
11. System G for Fraud Detection
12. System G for Cybersecurity
13. System G for Sensor Monitoring (Smarter another Planet)
14. System G for Cellular Network Monitoring
15. System G for Cloud Monitoring
17. System G for Traffic Navigation
18. System G for Image and Video Semantic Understanding
19. System G for Genomic Medicine
20. System G for Brain Network Analysis
21. System G for Data Curation
22. System G for Near Earth Object Analysis
Deep Learning on Mobile Object Detection & Recognition
Challenges
How to do Machine Reasoning?

*Five Layers of Understanding.*

*Deterministic Classification + Learning Inference*

- **Cognition Layer**
- **Semantics Layer**
- **Concept Layer**
- **Feature Layer**
- **Sensor Layer**

HR records, Travel records, Badge/Location records, Phone records, Mobile records

Transmitted images, speech content, video content

- : observations
- : hidden states

© 2013 IBM Corporation
Multi-Scale Deep Convolutional Neural Network for Fast Object Detection
Where are Pedestrians? — Deep Learning + Graph Contextual Analysis
Demo: Detecting Cars and Pedestrians in complex scenario
Comparison to the State-of-the-Art on KITTI benchmark test set

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
<th>Cars</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Easy</td>
<td>Mod</td>
<td>Hard</td>
<td>Easy</td>
<td>Mod</td>
<td>Hard</td>
<td>Easy</td>
<td>Mod</td>
</tr>
<tr>
<td>LSVM-MDPM-sv [35]</td>
<td>10s</td>
<td>68.02</td>
<td>56.48</td>
<td>44.18</td>
<td>47.74</td>
<td>39.36</td>
<td>35.95</td>
<td>35.04</td>
<td>27.50</td>
</tr>
<tr>
<td>DPM-VOC-VP [36]</td>
<td>8s</td>
<td>74.95</td>
<td>64.71</td>
<td>48.76</td>
<td>59.48</td>
<td>44.86</td>
<td>40.37</td>
<td>42.43</td>
<td>31.08</td>
</tr>
<tr>
<td>SubCat [16]</td>
<td>0.7s</td>
<td>84.14</td>
<td>75.46</td>
<td>59.71</td>
<td>54.67</td>
<td>42.34</td>
<td>37.95</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3DVP [37]</td>
<td>40s</td>
<td>87.46</td>
<td>75.77</td>
<td>65.38</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AOG [38]</td>
<td>3s</td>
<td>84.80</td>
<td>75.94</td>
<td>60.70</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Faster-RCNN [4]</td>
<td>2s</td>
<td>86.71</td>
<td>81.84</td>
<td>71.12</td>
<td>78.86</td>
<td>65.90</td>
<td>61.18</td>
<td>72.26</td>
<td>63.35</td>
</tr>
<tr>
<td>CompACT-Deep [15]</td>
<td>1s</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>70.69</td>
<td>58.74</td>
<td>52.71</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DeepParts [39]</td>
<td>1s</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>70.49</td>
<td>58.67</td>
<td>52.78</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FilteredICF [40]</td>
<td>2s</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>67.65</td>
<td>56.75</td>
<td>51.12</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pAUCEnsT [41]</td>
<td>60s</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>65.26</td>
<td>54.49</td>
<td>48.60</td>
<td>51.62</td>
<td>38.03</td>
</tr>
<tr>
<td>Regionlets [20]</td>
<td>1s</td>
<td>84.75</td>
<td>76.45</td>
<td>59.70</td>
<td>73.14</td>
<td>61.15</td>
<td>55.21</td>
<td>70.41</td>
<td>58.72</td>
</tr>
<tr>
<td>3DOP [5]</td>
<td>3s</td>
<td><strong>93.04</strong></td>
<td><strong>88.64</strong></td>
<td><strong>79.10</strong></td>
<td><strong>81.78</strong></td>
<td><strong>67.47</strong></td>
<td><strong>64.70</strong></td>
<td><strong>78.39</strong></td>
<td><strong>68.94</strong></td>
</tr>
<tr>
<td>Ours</td>
<td>0.4s</td>
<td>89.62</td>
<td>87.05</td>
<td>70.76</td>
<td><strong>82.02</strong></td>
<td><strong>71.47</strong></td>
<td><strong>66.02</strong></td>
<td><strong>83.19</strong></td>
<td><strong>73.93</strong></td>
</tr>
</tbody>
</table>

Single CPU core (2.40GHz) of an Intel Xeon E5-2630 server with 64GB of RAM. An NVIDIA Titan GPU was used for CNN computations.
Demo: Multi-Scale Deep Convolutional Neural Network for Fast Object Detection
System G Mobile Cognition — Enabling AI right on the Edge

- Created novel graph computing and deep learning framework on iOS devices and NAOqi robots including:
  - generic object recognition, event recognition, face recognition, visual sentiment recognition, and document recognition
  - graph database

Novel Deep Learning works that Speed Up image computation utilizing the GPUs on iOS devices: 195x or 1657x faster

<table>
<thead>
<tr>
<th></th>
<th>iPad Pro</th>
<th>iPhone 6s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification rate (on ~1000 classes)</td>
<td>~13 frames/sec</td>
<td>~7 frames/sec</td>
</tr>
</tbody>
</table>
Example - Generic Object Recognition running natively on mobile device (iPhone)
Demo - Document Detection (Warning!! Potential Sensitive Info Leakage!)
IBM System G EventNet

— Generic Event Recognition on Videos
Demo — System G Video event recognition and search example
Event Detection Baseline

Training Videos
- Attempting board trick
- Feeding an animal
- Landing a fish

Feature Extractions
- Low-level feature
  - SIFT (Visual)
  - STIP (Motion)
  - MFCC (Audio)
- Mid-level Concept

Classifiers
- Deep Learning
- Decision Tree
- SVM

Fusion
- Late Fusion
- Early Fusion

Output
Mid-level Feature Representation

- Decompose an event into concepts

**Scene**
- park
- street

**Object**
- person
- board

**Event: Board Trick**
- running
- jumping

**Audio**
- speech
- sound

**Action**
- street
- park
Events Classification Framework

Feature Extracting

Event Classifier
- Embrace Classifier
- PeopleMeet Classifier
- PeopleSplitUp Classifier
- PersonRuns Classifier

Pair-Activity Event Classifier

Event Identifying
- Backwards Search
- Forwards Search

Detected Events
- Embrace
- PeopleMeet
- PeopleSplitUp
- PersonRuns

Event Merging

Post-processing

Preliminary Events
Automatic Video Event Tagging

4,490 trained event models and a library of 95K videos

Event detected: cut_a_watermelon

Reasoned Event

Reasoned Event Concepts

Timestamp: 00:00:10
Concepts detected:
knife
watermelon
melon

Timestamp: 00:00:20
Concepts detected:
watermelon
melon

Timestamp: 00:00:30
Concepts detected:
knife
melon
watermelon

Timestamp: 00:00:40
Concepts detected:
knife
watermelon
melon

Timestamp: 00:00:50
Concepts detected:
food
knife
fruit

Timestamp: 00:01:10
Concepts detected:
fruit
knife
home

Timestamp: 00:01:20
Concepts detected:
food
knife
watermelon
melon

Timestamp: 00:01:30
Concepts detected:
ingredient
knife
home
Demo: IBM System G EventNet

Large Video Event Ontology Browsing, Search and Tagging (EventNet Demo)
Examples of our Previous work on Abnormal Video Event Analysis

**Event: Abnormal Behavior**
(Surveillance Video)

*TRECVID Surveillance Event Detection (SED) Evaluation 2008-2016*

![Abnormal Behavior Image]

**Event: Making a bomb**
(Consumer Video)

*TRECVID Multimedia Event Detection (MED) Evaluation 2010-2016*

![Making a bomb Image]
Detection and Tracking of Head, Shoulder, and Body

- Fusion of Head-shoulder and Body detection
- Adjust the detector searching scales
Detection Results
Complex Scenario — how to predict people action?
Since 2009, U.S. Justice Department lawyers have pursued at least 19 cases of corporate espionage, including GM, Ford, Motorola, DuPont, ... “Impacted economic and jobs” – WSJ Feb 21, 2013
System G Reasoning and Predictive Pipeline is composed of hundreds of cognitive analyzers.
Game Theory may help decision making in complex scenario -- Forecasting what will happen based on our and others potential actions.
Acceleration of Neural network on Mobile Devices

Richard Chen, Ruichi Yu*, Larry Lai
IBM T. J. Watson Research Center
*Ruichi Yu is IBM summer intern this year.
Summary of Acceleration of Neural network on Mobile Devices

- Porting Deep Convolution Neural Network on iOS Device with near real-time computation
  - Reduce algorithmic complexity with ignorable performance degradation.
  - Utilize computational hardware to achieve better performance.

<table>
<thead>
<tr>
<th></th>
<th>iPhone 7</th>
<th>iPhone 7+</th>
<th>iPhone 6s</th>
<th>iPad Pro 12.9&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex Net</td>
<td>70 ms</td>
<td>70 ms</td>
<td>130 ms</td>
<td>69 ms</td>
</tr>
<tr>
<td>p-Alex Net</td>
<td>N/A</td>
<td>35 ms</td>
<td>N/A</td>
<td>28 ms</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>130 ms</td>
<td>128 ms</td>
<td>195 ms</td>
<td>110 ms</td>
</tr>
<tr>
<td>p-GoogLeNet</td>
<td>N/A</td>
<td>80 ms</td>
<td>N/A</td>
<td>70 ms</td>
</tr>
<tr>
<td>VGG16 Net</td>
<td>880 ms</td>
<td>883 ms</td>
<td>1450 ms</td>
<td>725 ms</td>
</tr>
</tbody>
</table>
Outline

• Background

• Full Network Acceleration and Compression
  • Kernel Importance Measurement
  • Algorithmic Performance Evaluation
  • Computation Complexity Assessment

• Acceleration on iOS Mobile Devices
  • Metal API for GPU Programming
  • Computation Speed Evaluation
Methods for Running CNNs on Mobile Devices

- How to trade off between algorithmic complexity and performance?
- How to utilize hardware effectively

Sending CNN jobs to cloud

Acceleration CNN on Local Device

Apple A9X SoC, 12-core GPU
Problems for Running CNNs on Mobile Devices

<table>
<thead>
<tr>
<th>Store</th>
<th>AlexNet</th>
<th>VGG-S</th>
<th>VGG-16</th>
<th>GoogLeNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Size</td>
<td>243MB</td>
<td>393MB</td>
<td>552MB</td>
<td>51MB</td>
</tr>
<tr>
<td>Weights</td>
<td>61M</td>
<td>103M</td>
<td>138M</td>
<td>6.9M</td>
</tr>
<tr>
<td>Mult.s</td>
<td>725M</td>
<td>2640M</td>
<td>15484M</td>
<td>1566M</td>
</tr>
</tbody>
</table>

Statistics of some popular CNNs

Reference:
Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications
## Computational Resource on iPhone and iPad

<table>
<thead>
<tr>
<th></th>
<th>iPhone 6S (Plus)</th>
<th>iPad Air 2</th>
<th>iPad Pro (12.9/9.7)</th>
<th>iPhone 7 (Plus)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SoC</strong></td>
<td>A9</td>
<td>A8X</td>
<td>A9X</td>
<td>A10 Fusion</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>2x Twister @ 1.85 GHz</td>
<td>3x Typhone @ 1.5 GHz</td>
<td>2x Twister @ 2.26 GHz</td>
<td>4-core</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>PVR GT7600 (6 cluster)</td>
<td>PVR GXA6850 (8 cluster)</td>
<td>PVR 12 Cluster Series 7</td>
<td>6 cluster GPU?</td>
</tr>
<tr>
<td><strong>RAM (shared memory)</strong></td>
<td>2GB LDDR4</td>
<td>2GB LDDR3</td>
<td>4GB LDDR4</td>
<td>3GB on Plus?</td>
</tr>
<tr>
<td><strong>Memory bus width</strong></td>
<td>64-bit</td>
<td>128-bit</td>
<td>128-bit</td>
<td>?</td>
</tr>
<tr>
<td><strong>Max # of threads per group</strong></td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>?</td>
</tr>
</tbody>
</table>
Methods for Running CNNs on Mobile Devices

- Sending CNN jobs to cloud
- Compression (pruning) of CNN
- Speeding up CNN
- Sending CNN jobs to cloud
Think Different

- All existing methods can be viewed as approximations of an overly-redundant CNN, but do we really need such a CNN as the starting point?
Good to be Slim

• Slim CNN leads to:
  • less storage space
  • less memory usage
  • less computation
  • less power consumption

• Compare with others:
  • Full-Network Acceleration and Compression
  • We think one step preceedingly
Be Slim is Hard...

Train a small model from scratch

Randomly Pruning
Feature Selection on CNN

- CNNs can be viewed as a set of "overly-redundant" feature extractors
A method for Pruning Redundant Neurons and Kernels of Deep Convolutional Neural Networks

A pre-trained CNN

Extract CNN Responses

Measure the Importance of Feature Extractors

Prune Model

Fine-tuning
A method for Pruning Redundant Neurons and Kernels of Deep Convolutional Neural Networks

- Intractable → tractable
- Inconsistent → consistent

A pre-trained CNN → Extract Responses of a High-level Layer → Measure the Importance of Feature Extractors → Back-propagate the Importance & Prune Model → Fine-tuning

Forward Propagation

Input layers → Response → Response → Response → Important Score → Back Propagation and Pruning

FC layers
Fine-tuning the Pruned Model

- The pruned model consists of important feature extractors, but will suffer loss of accuracy due to loss of redundant features.
- Good starting point on the learning curve due to feature selection.
- Fine-tuning the pruned model with a lower learning rate to recover the performance.

![Graphs showing accuracy loss over iterations for LeNet Half IS Random TS and Cifar10 Half IS Random TS](image-url)
Evaluation

<table>
<thead>
<tr>
<th>IS_C</th>
<th>IS_CF</th>
<th>High</th>
<th>Mid</th>
<th>Low</th>
<th>FC+C_1</th>
<th>FC+C_2</th>
<th>FC+C_3</th>
<th>FC+C_4</th>
<th>FC+C_5</th>
<th>FC+C_4C_5</th>
<th>C_2C_3F_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error ↑ (%)</td>
<td>2.0</td>
<td>3.6</td>
<td>1.5</td>
<td>2.5</td>
<td>1.3</td>
<td>1.5</td>
<td>1.2</td>
<td>1.0</td>
<td>1.4</td>
<td>1.3</td>
<td>0.8</td>
</tr>
<tr>
<td>Mult. ↓</td>
<td>3.11x</td>
<td>3.46x</td>
<td>2.40x</td>
<td>2.64x</td>
<td>1.55x</td>
<td>2.12x</td>
<td>2.22x</td>
<td>2.43x</td>
<td>2.59x</td>
<td>2.68x</td>
<td>2.16x</td>
</tr>
<tr>
<td>Params. ↓</td>
<td>1.51x</td>
<td>3.74x</td>
<td>2.34x</td>
<td>3.63x</td>
<td>3.64x</td>
<td>1.51x</td>
<td>1.50x</td>
<td>1.49x</td>
<td>1.49x</td>
<td>1.03x</td>
<td>1.02x</td>
</tr>
<tr>
<td>Mem. ↓</td>
<td>1.61x</td>
<td>1.77x</td>
<td>1.65x</td>
<td>1.72x</td>
<td>1.18x</td>
<td>1.25x</td>
<td>1.40x</td>
<td>1.58x</td>
<td>1.58x</td>
<td>1.58x</td>
<td>1.46x</td>
</tr>
<tr>
<td>CPU speed ↑</td>
<td>2.64x</td>
<td>2.81x</td>
<td>2.09x</td>
<td>2.81x</td>
<td>1.32x</td>
<td>1.62x</td>
<td>1.86x</td>
<td>2.10x</td>
<td>2.19x</td>
<td>2.29x</td>
<td>2.10x</td>
</tr>
<tr>
<td>GPU speed ↑</td>
<td>2.53x</td>
<td>2.80x</td>
<td>2.21x</td>
<td>2.40x</td>
<td>1.40x</td>
<td>1.70x</td>
<td>1.90x</td>
<td>2.16x</td>
<td>2.24x</td>
<td>2.19x</td>
<td>1.93x</td>
</tr>
</tbody>
</table>
Programming Model of Apple GPU

• Programming by Metal language
  • Most of syntaxes are compatible with C++14

• A unified programming language interface for **graphics** and **data-parallel computation** workloads.

• Single Instruction Multiple Threads (SIMT) programming fashion
  • Every thread performs **simple** and **identical** instruction.
Execution Model of Apple GPU

- It integrates the support for both **graphics** and **compute** operations.
  - Three command encoder:
    - Render Command Encoder: Graphics Rendering
    - Compute Command Encoder: Data-Parallel Compute Processing
    - Blitting Command Encoder: Transfer Data between Resource
  - Multi-threading in encoding command is supported

*Put identical type of commands together as possible.*
Diagram of iOS GPU Execution
Metal Programming, Kernal Function

• Compute command:
  • Two parameters, `threadsPerGroup` and `numThreadgroups`, determines number of threads. (They are all 3-D variable.)
  • The total of all threadgroup memory allocations must not exceed 16 KB (shared memory within one threadgroup.)
    • Shared memory used to sync data across different threads.

• Four attribute qualifiers are used to access data in kernel function:
  • `thread_position_in_grid`, `threadgroup_position_in_grid`
  • `thread_position_in_threadgroup`, `thread_index_in_threadgroup`

Max number of threads in a group is 512 (device-dependent).

For a 48x48 image

```
threadGroupCount = MTLSizeMake(16, 16, 1)
threadGroups = MTLSizeMake(3, 3, 1)
```
Sigmoid Function — Speedup

Perform sigmoid function on $2^{24}$ data points on iPad Pro:

| SpeedGPU: runtime : 0.01544429166666667 seconds |
|-----------------|-----------------------------------------------|
| CPU: runtime : 39.584522208333333 seconds      |
| Speedup Ratio: 2563.05197173713                 |

Syntax might be changed in swift3.
Image Recognition — Convolution Layer

- Computation-intensive layer.
- Explore a way to reuse data in cache to improve the speed.
- Lots of implementation trick to reduce overhead in GPU computations.
  - E.g., perform compilation optimization in your codes.

\[
Y(n, y, x) = \sum_{c} \sum_{j} \sum_{i} X(c, y - j, x - i) * W(n, c, j, i),
\]
Evaluation on iOS Mobile Devices

- p-prefix denotes pruned/compressed CNN.

<table>
<thead>
<tr>
<th></th>
<th>iPhone 7</th>
<th>iPhone 7+</th>
<th>iPhone 6s</th>
<th>iPad Pro 12.9&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex Net</td>
<td>70 ms</td>
<td>70 ms</td>
<td>130 ms</td>
<td>69 ms</td>
</tr>
<tr>
<td>p-Alex Net</td>
<td>N/A</td>
<td>35 ms</td>
<td>N/A</td>
<td>28 ms</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>130 ms</td>
<td>128 ms</td>
<td>195 ms</td>
<td>110 ms</td>
</tr>
<tr>
<td>p-GoogLeNet</td>
<td>N/A</td>
<td>80 ms</td>
<td>N/A</td>
<td>70 ms</td>
</tr>
<tr>
<td>VGG16 Net</td>
<td>880 ms</td>
<td>883 ms</td>
<td>1450 ms</td>
<td>725 ms</td>
</tr>
</tbody>
</table>