Outline

• Introduction

• Deep Neural Networks

• Sparse Data Sets

• Challenge

• Summary
GraphChallenge encourages community approaches to developing new solutions for analyzing graphs and sparse data.

Goal: to provide a well-defined community venue for stimulating research and highlighting innovations in graph analysis software, hardware, algorithms, and systems.

Audience: any individual or team that seeks to highlight their contributions to graph analysis software, hardware, algorithms, and/or systems.

Graph Challenge contributions should be submitted to the IEEE HPEC conference submission website (see ieee-hpec.org).
Current Graph Challenges

• Static Graph Challenge: Sub-Graph Isomorphism
  – This challenge seeks to identify a given sub-graph in a larger graph
  – Triangle counting and K-Truss

• Streaming Graph Challenge: Stochastic Block Parition
  – This challenge seeks to identify optimal blocks (or clusters) in a larger graph

**Graph H**

```
+---+---+---+---+
| a | g | b | h |
+---+---+---+---+
| c | i | d | j |
```

**Graph G**

```
+---+---+---+---+
| 1 | 2 | 3 | 4 |
|---+---+---+---|
| 5 | 6 | 7 | 8 |
+---+---+---+---+

**Isomorphism**

- $f(a) = 1$
- $f(b) = 6$
- $f(c) = 8$
- $f(d) = 3$
- $f(g) = 5$
- $f(h) = 2$
- $f(i) = 4$
- $f(j) = 7$

**Streaming unstructured graph**

**Block optimized graph**
Motivation

Challenges such as YOHO, MNIST, HPC Challenge, ImageNet, and VAST have played important roles in driving progress in fields as diverse as machine learning, high performance computing, and visual analytics.

**GraphChallenge** encourages community approaches to developing new solutions for analyzing graphs and sparse data derived from social media, sensor feeds, and scientific data to enable relationships between events to be discovered as they unfold in the field.

**Graph Challenge Details**

**IEEE HPEC 2018 Graph Challenge Submitter Guidelines**

- May 18: regular HPEC paper deadline
- July 15 (anywhere on Earth): Graph Challenge regular and update submission deadline

**Analysis of all 2017 Triangle Counting Submissions**
Graph Datasets

- Public data available from a variety of sources
- Data formats are typically not standardized (we have normalized!)
Graph Datasets Hosted on Amazon


---

**Graph Datasets Hosted on Amazon**

**SNAP Datasets (click to expand)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon0302</td>
<td>Amazon product co-purchasing network from March 2 2003</td>
</tr>
<tr>
<td></td>
<td>Adjacency TSV</td>
</tr>
<tr>
<td></td>
<td>Incidence TSV</td>
</tr>
<tr>
<td></td>
<td>Adjacency MMIO</td>
</tr>
<tr>
<td></td>
<td>Incidence MMIO</td>
</tr>
<tr>
<td>amazon0312</td>
<td>Amazon product co-purchasing network from March 12 2003</td>
</tr>
<tr>
<td>amazon0512</td>
<td>Amazon product co-purchasing network from May 5 2003</td>
</tr>
<tr>
<td>amazon0612</td>
<td>Amazon product co-purchasing network from June 1 2003</td>
</tr>
</tbody>
</table>

**Data available for download in multiple formats**

- Adjacency and Incidence matrix for each graph
- File format: ASCII Tab separated file and MMIO format

**Partition Challenge Datasets with Known Truth Partitions (click to expand)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Graphs (small)</td>
<td>Small Static Graphs with known truth for the Stochastic Block Partitioning Challenge</td>
</tr>
<tr>
<td>50 nodes</td>
<td></td>
</tr>
<tr>
<td>100 nodes</td>
<td></td>
</tr>
<tr>
<td>500 nodes</td>
<td></td>
</tr>
<tr>
<td>1000 nodes</td>
<td></td>
</tr>
<tr>
<td>5000 nodes</td>
<td></td>
</tr>
<tr>
<td>Static Graphs (large)</td>
<td>Large Static Graphs with known truth for the Stochastic Block Partitioning Challenge</td>
</tr>
<tr>
<td>50 nodes</td>
<td></td>
</tr>
<tr>
<td>100 nodes</td>
<td></td>
</tr>
<tr>
<td>500 nodes</td>
<td></td>
</tr>
<tr>
<td>1000 nodes</td>
<td></td>
</tr>
<tr>
<td>5000 nodes</td>
<td></td>
</tr>
</tbody>
</table>

**Small graphs with known partitions available for block partitioning algorithm**
New Challenge Goal: Enable Large Sparse Deep Neural Networks

- Deep neural networks (DNNs) are at the heart of modern AI miracles
- Larger neural networks often perform better
  - Larger number of layers/features allow more non-linear boundaries
  - Problem: limited by expensive high-speed memory size
  - Solution: sparse (pruned) neural networks deliver comparable performance with less memory

Dense Training

Dense Training, then Pruning
LeCun et al., Hassibi et al.

Dense Training while Pruning
Srinivas et al., Han et al.

Sparse Training
Prabhu et al. (X-Net)

Input Features
$y_0$

Output Classification
$y_4$

Dense Training

LeCun et al., Hassibi et al.

Srinivas et al., Han et al.

Prabhu et al. (X-Net)
Outline

- Introduction
- Deep Neural Networks
- Sparse Data Sets
- Challenge
- Summary
Before 2010

Machine Learning
After 2015

Deep Neural Networks

Machine Learning
Lexington, Mass. B. G. Farley and W. A. Clark, "Simulation of self-organizing systems by digital computer," incoming connections at any element, each having its succeeding element, the appropriate weight is added to the potential still taking place even if the input patterns are subject to considerable random variation. The second experiment indicates that, when patterns are used, the behavior of the network of non-linear elements, each element having a different input pattern, was accomplished. The basis of this is the idea of treating the system as a network of simple units, each unit having a threshold below which firing does not occur. When an element fires, its threshold immediately rises to a definite threshold for incoming excitation, below which firing does not occur. The effect of the connection to which it is connected. The effectiveness of the connection to which it is connected. Tend arguments to the subjects of the other three speakers here today. Consider Fig. 1. The horizontal lines on the left differ from those on the right in having vertical spikes mostly from top to bottom. The vertical spikes indicate the presence of excitation. By pattern recognition we mean the extraction of features from a background of irrelevant data. The work in this paper was sponsored jointly by the U.S. Army, the U.S. Air Force, and the National Science Foundation. The first part received input patterns and the second part classified the output patterns. When a pattern is recognized, it is used to update the weights of the connections to which it is connected. Finally, it is worth mentioning that simulation experiments in general.
Deep Neural Network

\[ y_{l+1} = h(y_l W_l + b_l) \]
Graph Challenge Math Conventions

\[ Y_{l+1} = h(Y_l W_l + b_l) \]

- \( Y_0 \) is (# inputs) x (# features); each row is a feature vector
  - # features = # neurons/layer (constant for all layers)
- \( W_l \) is (# neurons) x (# neurons); \( W_l(i,j) > 0 \) means a connection from \( i \) to \( j \)
- \( b_l \) is a bias row vector applied to each input
- \( h(x) \) is the ReLU function with a max cutoff
  
  \[
  0 < x < 32, \text{ x is unchanged; } x < 0, \text{ x is changed to 0; } x > 32, \text{ x is changed to 32}
  \]

- Sparse DNN Challenge uses standard graph community terminology
  - Row vectors for features and left matrix multiply to progress through network
- Standard AI definitions can be used (transpose all matrices and multiply on right)
Next Frontier: Sparse Neural Networks

- Larger neural networks often perform better
  - Larger number of layers/features allow more non-linear boundaries
  - Challenge: limited by GPU memory size
  - Solution: sparse neural networks

- Active research area in AI community

**Efficient Sparse-Winograd Convolutional Neural Networks**

Xingyu Liu†, Jeff Pool†, Song Han†, William J. Dally†
*Stanford University, †NVIDIA, ‡Massachusetts Institute of Technology, ‡Google Brain
{xyl, dally}@stanford.edu

**SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks**

Angshuman Parashar† Minsoo Rhu†
Anurag Makkara‡ Antonio Puglielli‡ Rangharajan Venkatesan‡ Brucek Khailany‡
Joel Emer† Stephen W. Keckler† William J. Dally†
NVIDIA† Massachusetts Institute of Technology‡ UC-Berkeley* Stanford University‡

Exploring the Regularity of Sparse Structure in Convolutional Neural Networks

Huizi Mao†, Song Han†, Jeff Pool‡, Wenshuo Li‡, Xingyu Liu‡,
Yu Wang‡, William J. Dally†,†
†Stanford University
‡NVIDIA
Tsinghua University
(huizi,songhan,dally)@stanford.edu

Published as a conference paper at ICLR 2018

GPU = Graphics Processing Unit
Outline

• Introduction

• Deep Neural Networks

• Sparse Data Sets

• Challenge

• Summary
Sparse DNN Data

- Large sparse DNNs are difficult to obtain from real data
  - Similar to early days of Graph500.org
  - First step is to simulate data with desired topological properties
  - Emphasis on the “hard” part of the problem: large sparse DNNs

- Approach: RadiX-Net\(^1\) sparse DNN generator
  - Efficiently generates a wide range of pre-determined DNNs
  - Specific number of connections per neuron
  - Equal number of paths between all inputs/outputs and intermediate layers

- DNN generation algorithm
  - Mixed radices generate base DNNs of specified connectedness
  - Kronecker product base DNNs to create larger/sparser DNNs
  - Randomly permute and chain DNNs to create very large DNNs

\(^1\)RadiX-Net: Structured Sparse Matrices for Deep Neural Networks, Robinett & Kepner, IEEE IPDPS GrAPL Workshop 2, 2019
Sparsity, path-connectedness, and symmetry using mixed-radix numeral systems

\[ N = (N_1, \ldots, N_5) \]

Input Layer

\[ N' = \prod_{N \in \mathcal{N}} N = \prod_{N \in \mathcal{N}} \prod_{N \in \mathcal{N}} N \]

\[ \prod_{N \in \mathcal{N}} N \text{ divides } N' \]

\((W_1, \ldots, W_M) = (W_1, \ldots, W_1, \ldots, W_1, \ldots, W_1, \ldots)\)
RadiX-Net topologies: diversity in layer sizes allowed by Kronecker product
Graph Challenge Base DNNs

• Base DNNs (all with 32 connections per neuron) generated using RadiX-Net

  • 6 layers, 1024 neurons/layer
    – Using radix = [[2,2,2,2,2,2]]
    – kron = [16,16,16,16,16,16,16]

  • 8 layers, 4096 neurons/layer
    – Using radix = [[2,2,2,2,2,2,2,2]]
    – kron = [16,16,16,16,16,16,16,16,16]

  • 10 layers, neurons/layer = 16384
    – Using radix = [[2,2,2,2,2,2,2,2,2,2]]
    – kron = [16,16,16,16,16,16,16,16,16,16,16]

  • 12 layers, neurons/layer = 65536
    – Using radix = [[2,2,2,2,2,2,2,2,2,2,2,2]]
    – kron = [16,16,16,16,16,16,16,16,16,16,16,16,16]

• All non-zero weights set to 1/16 (weights into any neuron sum to 2)
  – Observed to produce layer count independent properties (theory in progress)
Example RadiX-Net Graph

- 6 layer, 64 neurons per layer, 2 connections per neuron RadiX-Net DNN produced from Radix set $[[2,2,2,2,2,2]]$

- Kronecker product of this DNN with $[16,16,16,16,16,16,16]$ produces a 6 layer, 1024 neurons per layer DNN with 32 connections per neuron
Graph Challenge Full DNNs

- Base DNNs are randomly permuted and replicated to generate 12 DNNs
- Each layer is written out as a .tsv file: row, column, weight

<table>
<thead>
<tr>
<th>Layers</th>
<th>Neurons/Layer</th>
<th>Total Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>1024</td>
<td>3,932,160</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>15,728,640</td>
</tr>
<tr>
<td></td>
<td>16384</td>
<td>62,914,560</td>
</tr>
<tr>
<td></td>
<td>65536</td>
<td>251,658,240</td>
</tr>
<tr>
<td>480</td>
<td>15,728,640</td>
<td>62,914,560</td>
</tr>
<tr>
<td></td>
<td>251,658,240</td>
<td>1,006,632,960</td>
</tr>
<tr>
<td>1920</td>
<td>62,914,560</td>
<td>1,006,632,960</td>
</tr>
<tr>
<td></td>
<td>4,026,531,840</td>
<td></td>
</tr>
</tbody>
</table>

Total Edges = 32 x layers x neurons/layer

- Bias values selected to produce reasonable number of output for MNIST input

<table>
<thead>
<tr>
<th>Neurons/Layer</th>
<th>1024</th>
<th>4096</th>
<th>16384</th>
<th>65536</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>-0.3</td>
<td>-0.35</td>
<td>-0.4</td>
<td>-0.45</td>
</tr>
</tbody>
</table>
MNIST input data

- **MNIST corpus** of 60,000 28x28 pixel images of handwritten numbers is a staple of AI

- Sparse DNN Graph Challenge uses interpolated sparse versions of this entire corpus as input
  - Each image is resized to 32x32 (1024 neurons), 64x64 (4096 neurons), 128x128 (16384 neurons), and 256x256 (65536 neurons)
  - Each sparse resized corpus is saved as .tsv file: row, column, 1

Outline

- Introduction
- Deep Neural Networks
- Sparse Data Sets
- Challenge
- Summary
Challenge Instructions

- Download from GraphChallenge.org: DNNs weight matrices $W_l$, sparse MNIST input data $Y_0$, and truth categories
- Load a DNN and its correspond input
- Create and set the appropriate sized bias vectors $b_l$ from the table
- Evaluate the ReLU DNN equation for all layers (timed)
  \[ Y_{l+1} = h(Y_l W_l + b_l) \]
- Identify the categories (rows) in final matrix with entries > 0 (timed)
- Compare computed categories with truth categories to check correctness
- Compute the rate connection rate for the DNN: (# inputs) x (# connections) / time
- Report time and rate for each DNN measured
function Y = inferenceReLUvec(W,bias,Y0);
% Performs ReLU inference using input feature vector(s) Y0, DNN weights W, and bias vectors.
YMAX = 32;               % Set max value.
Y = Y0;                   % Initialize feature vectors.
for i=1:length(W)        % Loop through each weight layer W{i}
    % Propagate through layer.
    % Note: using graph convention of A(i,j) means connection from i *to* j,
    % that requires *left* multiplication of feature *row* vectors.
    Z = Y*W{i};
    b = bias{i};
    Y = Z + (double(logical(Z)) .* b);  % Apply bias to non-zero entries.
    Y(Y < 0) = 0;                        % Threshold negative values.
    Y(Y > YMAX) = YMAX;                  % Threshold maximum values.
end
return
end
Recommendations

**Avoid**

- Exploiting repetitive structure of weight matrices, weight values, bias values
- Exploiting layer independence of results
- Using optimizations that would not work on real-world data

**Do**

- Use an implementation that could work on real-world data
- Create compressed binary versions of inputs to accelerate reading the data
- Split inputs and run in data parallel mode to achieve higher performance
  - Requires replicating weight matrices on every processor (can require a lot of memory)
- Split up layers and run in pipeline parallel mode to achieve higher performance
  - Saves memory, but requires communicating results after each group of layers
- Use other reasonable optimizations that would work on real-world data
### Example Serial Performance

<table>
<thead>
<tr>
<th>Neurons per Layer</th>
<th>Layers</th>
<th>Connections (edges)</th>
<th>Inference Time (second)</th>
<th>Inference Rate (input(^1) x edge/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>120</td>
<td>3,932,160</td>
<td>626</td>
<td>376\times10^6</td>
</tr>
<tr>
<td>1024</td>
<td>480</td>
<td>15,728,640</td>
<td>2440</td>
<td>386\times10^6</td>
</tr>
<tr>
<td>1024</td>
<td>1920</td>
<td>62,914,560</td>
<td>9760</td>
<td>386\times10^6</td>
</tr>
<tr>
<td>4096</td>
<td>120</td>
<td>15,728,640</td>
<td>2446</td>
<td>385\times10^6</td>
</tr>
<tr>
<td>4096</td>
<td>480</td>
<td>62,914,560</td>
<td>10229</td>
<td>369\times10^6</td>
</tr>
<tr>
<td>4096</td>
<td>1920</td>
<td>251,658,240</td>
<td>40245</td>
<td>375\times10^6</td>
</tr>
<tr>
<td>16384</td>
<td>120</td>
<td>62,914,560</td>
<td>10956</td>
<td>344\times10^6</td>
</tr>
<tr>
<td>16384</td>
<td>480</td>
<td>251,658,240</td>
<td>45268</td>
<td>333\times10^6</td>
</tr>
<tr>
<td>16384</td>
<td>1920</td>
<td>1,006,632,960</td>
<td>179401</td>
<td>336\times10^6</td>
</tr>
<tr>
<td>65536</td>
<td>120</td>
<td>251,658,240</td>
<td>45813</td>
<td>329\times10^6</td>
</tr>
<tr>
<td>65536</td>
<td>480</td>
<td>1,006,632,960</td>
<td>202393</td>
<td>298\times10^6</td>
</tr>
<tr>
<td>65536</td>
<td>1920</td>
<td>4,026,531,840</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\)Number of MNIST inputs = 60,000
• Serial reference implementation run in parallel by splitting feature vectors
• DNN replicated
• Run on MIT SuperCloud
• Intel KNL processors with 192 GB RAM
• MIT/IEEE/Amazon Graph Challenge has provides a valuable venue for innovators in graph and sparse network analysis system

• Sparse DNN Challenge seeks stimulate new work in the area of sparse AI

• Proposed challenge provides a wide range of simulated sparse DNNs to evaluate

Deep Neural Network (DNN)

Input Features

\[ y_0 \]

\[ W_0 \]
\[ b_0 \]

\[ W_1 \]
\[ b_1 \]

\[ W_2 \]
\[ b_2 \]

\[ W_3 \]
\[ b_3 \]

Output Classification

\[ y_4 \]

\[ y_1 \]

\[ y_2 \]

\[ y_3 \]
Acknowledgements

• David Martinez
• Jeremy Kepner
• Albert Reuther
• Vijay Gadepally
• Michael Jones
• Lauren Milechin
• Bill Arcand
• Bill Bergeron
• David Bestor

• Chansup Byun
• Matt Hubbell
• Pete Michaleas
• Julie Mullen
• Andy Prout
• Tony Rosa
• Jason Williams
• Ryan Whelan
• Charles Yee

• Paul Monticciolo
• Sanjeev Mohindra
• Steven Smith
• Edward Kao
• Michael Hurley
• William Song