Ptolemy: Architecture Support for Robust Deep Learning

Yiming Gan
Department of Computer Science, University of Rochester

with
Yuxian Qiu, Shanghai Jiao Tong University
Jingwen Leng, Shanghai Jiao Tong University
Minyi Guo, Shanghai Jiao Tong University
Yuhao Zhu, University of Rochester

https://github.com/Ptolemy-dl/Ptolemy
Deep Learning: Not Robust
Deep Learning: Not Robust

Legitimate Example + Perturbation = Adversarial Example
Mission Critical System

ADAS

Security Cameras

Trajectory is the word I was searching for.
Robust Deep Learning Requirements

• Accurately detect adversarial examples
Robust Deep Learning Requirements

- Accurately detect adversarial examples
- Do not bring large overhead on system performance
Robust Deep Learning Requirements

- Accurately detect adversarial examples
- Do not bring large overhead on system performance

= Ptolemy
Hot Path

Traditional Software

[1] Thomas Ball, James R. Larus, Using Paths to Measure, Explain, and Enhance Program Behavior
Hot Path

Traditional Software

- Measure Program Behavior
- Optimizing Program
- Debugging

[1] Thomas Ball, James R. Larus, Using Paths to Measure, Explain, and Enhance Program Behavior
Defining Important Neuron

Input Feature Map

Weights

Output Feature Map

0.3 0.4 0.2 1.0 0.1

0.2 0.2 0.3
0.3 0.2 0.4
0.4 0.1 0.2
-0.1 0.09 0.1
-1.0 2.1 0.5

= 0.06 0.46 0.44
## Defining Important Neuron

**Input Feature Map**

| 0.3 | 0.4 | 0.2 | 1.0 | 0.1 |

**Weights**

| 0.2 | 0.2 | 0.3 |
| 0.3 | 0.2 | 0.4 |
| 0.4 | 0.1 | 0.2 |
| -0.1 | 0.09 | 0.1 |
| -1.0 | 2.1 | 0.5 |

**Output Feature Map**

| 0.06 |
| 0.46 |
| 0.44 |
Defining Important Neuron

Input Feature Map

0.3 0.4 0.2 1.0 0.1

Weights

<table>
<thead>
<tr>
<th>0.2</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>0.4</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>-0.1</td>
<td>0.09</td>
<td>0.1</td>
</tr>
<tr>
<td>-1.0</td>
<td>2.1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Output Feature Map

= 0.06

= 0.46

= 0.44
Defining Important Neuron

Input Feature Map

0.3 0.4 0.2 1.0 0.1

Weights

<table>
<thead>
<tr>
<th></th>
<th>0.2</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.2</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>-0.1</td>
<td>0.09</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>-1.0</td>
<td>2.1</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

Output Feature Map

0.06
0.46
0.44
Defining Important Neuron

### Important Neuron Extraction in Convolution Layer

<table>
<thead>
<tr>
<th>Input Feature Map</th>
<th>Kernel</th>
<th>Output Feature Map (OFMap)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 1.2 1.0</td>
<td>0.1 0.2 0.2</td>
<td>5.97 5.14 2.88</td>
</tr>
<tr>
<td>1.2 1.9 1.1 0.2</td>
<td>0.2 0.7 0.8</td>
<td>4.31 5.47 3.57</td>
</tr>
<tr>
<td>0.9 1.0 2.0 1.5</td>
<td>0.3 0.9 0.9</td>
<td>1.95 3.14 2.63</td>
</tr>
<tr>
<td>0.2 1.0 1.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
5.47 = 2.0 \times 0.7 + 1.4 \times 0.9 + 1.5 \times 0.8 + 1.0 \times 0.9 + \ldots
\]

\[
2.0 \times 0.7 + 1.4 \times 0.9 + 1.5 \times 0.8 > 0.6 \times 5.47, \text{ assuming } \theta = 0.6
\]

Important Neurons in the OFMap (identified before): 5.47

Important Neurons identified in the current layer: 2.0, 1.4, 1.5
From Neuron to Path

Output Layer

Hidden Layer

Input Layer
From Neuron to Path

Output Layer

Hidden Layer

Input Layer
From Neuron to Path

- **Input Layer**
- **Hidden Layer**
- **Output Layer**
Class Path
Class Path

Union
Class Path Similarity

AlexNet @ ImageNet
Class Path Similarity

AlexNet @ ImageNet
Ptolemy Overview

Neural Networks

“Cat”
Ptolemy Overview

Neural Networks

“Cat”

Extract
Ptolemy Overview

Neural Networks

“Cat”

Extract

Compare
Ptolemy Pipeline

Layer N

Layer N-1

Layer 2

Layer 1
Ptolemy Pipeline

Layer N

Layer N-1

Layer 2

Layer 1

Inference
Ptolemy Pipeline

Layer N

Layer N-1

Layer 2

Layer 1

Inference
Ptolemy Pipeline

Layer N

Layer N-1

Layer 2

Layer 1

Inference
Ptolemy Pipeline

Layer N

Layer N-1

Layer 2

Layer 1

Inference
Ptolemy Pipeline

Layer N
Layer N-1
Layer 2
Layer 1

Extraction
Ptolemy Pipeline

Layer N

Layer N-1

Layer 2

Layer 1

Extraction
Ptolemy Pipeline

Layer N

Layer N-1

Layer 2

Layer 1

Extraction
Algorithmic Variation

Layer N

Layer N-1

Layer 2

Layer 1

Inference
Algorithmic Variation

Layer N

Layer N-1

Layer 2

Layer 1

Inference

Extraction
Algorithmic Variation

Layer N

Layer N-1

Layer 2

Layer 1

Inference

Extraction
Algorithmic Variation

Layer N

Layer N-1

Layer 2

Layer 1

Inference

Extraction
Algorithmic Variation

Layer N

Layer N-1

Layer 2

Layer 1

Extraction
Algorithmic Variation

Layer N

Layer N-1

Layer 2

Layer 1

Extraction

IF 1  IF 2  ...  IF N

EX 1  EX 2

EX N  Det
Algorithmic Variation

Sorting

IF 1 | IF 2 | ... | IF N | EX N | EX N-1 | ... | EX 1 | Det

Threshold

IF 1 | IF 2 | ... | IF N | EX N | EX N-1 | ... | EX 1 | Det

IF: Inference, EX: Extraction, Det: Detection
Algorithmic Variation

Full Extraction

IF 1 IF 2 ... IF N EX N EX N-1 ... EX 1 Det

Partially Extraction

IF 1 IF 2 ... IF N EX N EX N-1 Det

IF: Inference, EX: Extraction, Det: Detection
Framework

Backward

Partial Extraction

Forward

Thresholding

= Backward + Fully Extraction + Sorting
Interface

- High-level: Python-based, user define input
Interface

- High-level: Python-based, user define input
- Low-level: Customized ISA
Compiler Optimization: Layer Level

for j = 1 to L {
    inf(j)
    <extraction on layer j>
}

Compiler Optimization: Layer Level

for j = 1 to L {
    inf(j)
    <extraction on layer j>
}

for j = 1 to L {
    inf (j+1)
    <extraction on layer j>
}

<extraction on layer L>
Compiler Optimization: Neuron Level

for j = 1 to N {
    sort(i)
    acum(i)
}
Compiler Optimization: Neuron Level

```c
for j = 1 to N {
    sort(i)
    acum(i)
}

\[\text{sort}(1)\]

for i = 1 to N-1{
    sort(i+1)
    acum(i)
}

acum(N)
```
Architecture Overview

- **DRAM**
  - Input/Output
  - Weights
  - Feature Maps
  - Partial Sums
  - Masks
  - Paths

- **DNN Accelerator**
  - SRAM (Weights, Feature Maps, Partial Sums, Masks)

- **Controller**
  - SRAM (Code, Paths)

- **Path Constructor**
  - Sort & Merge
  - Accumulate
  - Gen Masks
  - SRAM (Partial sums, Partial masks, Masks)
Enhanced MAC unit

\[ i \times w + \text{thd} \]

MUX

\[ \text{mode} \]

to SRAM

\[ \text{psum} \]

to SRAM
## Evaluation

<table>
<thead>
<tr>
<th>Network</th>
<th>AlexNet, ResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>Cifar10, Cifar100, ImageNet</td>
</tr>
<tr>
<td>Attacks</td>
<td>BIM, CWL2, DeepFool, FGSM, JSMA</td>
</tr>
<tr>
<td>Adaptive Attacks</td>
<td>Self constructed</td>
</tr>
<tr>
<td>Baselines</td>
<td>EP[1], CDRP[2]</td>
</tr>
</tbody>
</table>

# Hardware Setup

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DNN Accelerator</strong></td>
<td><strong>20 x 20</strong></td>
</tr>
<tr>
<td><strong>Technology</strong></td>
<td><strong>Silvaco 15nm</strong></td>
</tr>
<tr>
<td><strong>On-chip SRAM</strong></td>
<td><strong>1.5MB</strong></td>
</tr>
</tbody>
</table>
Evaluation

Accuracy

1

0

AlexNet on ImageNet

1 2 3 Hybrid EP CDRP
Evaluation

Accuracy

AlexNet on ImageNet
Evaluation

Accuracy

AlexNet on ImageNet
Evaluation

Accuracy decrease

<table>
<thead>
<tr>
<th>AlexNet on ImageNet</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.92</td>
</tr>
<tr>
<td>2</td>
<td>0.92</td>
</tr>
<tr>
<td>3</td>
<td>0.92</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.92</td>
</tr>
<tr>
<td>EP</td>
<td>0.92</td>
</tr>
<tr>
<td>CDRP</td>
<td>0.84</td>
</tr>
</tbody>
</table>
Evaluation

Latency

Overhead

Energy

Overhead

BwCU    BwAb    FwAb    Hybrid    EP

AlexNet on ImageNet
Evaluation

Latency Overhead

Energy Overhead

BwCU  BwAb  FwAb  Hybrid  EP

AlexNet on ImageNet
Evaluation

Latency Overhead

Energy Overhead

- Latency Overhead Decrease
- Energy Overhead Decrease

AlexNet on ImageNet
Conclusion

Ptolemy: Accurate, low overhead, adversarial attack detection

- Algorithm Framework
- Compiler Optimization
- Architecture Support
Collaborators

Yuxian Qiu  Jingwen Leng  Minyi Guo  Yuhao Zhu
Questions

https://github.com/Ptolemy-dl/Ptolemy
def AdversaryDetection(model, input, θ, φ):
    output = Inference(model, input)
    N = model.num_layers
    // Selective extraction only in the last three layers
    for L in range(N-3, N):
        if L != N-1:
            // Forward extraction using absolute thresholds
            ImptN[L] = ExtractImptNeurons(1, 1, φ, L)
        else:
            // Forward extraction using cumulative thresholds
            ImptN[L] = ExtractImptNeurons(1, 0, θ, L)
    dynPath.concat(GenMask(ImptN[L]))
    classPath = LoadClassPath(argmax(output))
    is_adversary = Classify(classPath, dynPath)
    return is_adversary
## Backup

<table>
<thead>
<tr>
<th>Class</th>
<th>Name</th>
<th>23-20</th>
<th>19-16</th>
<th>15-12</th>
<th>11-8</th>
<th>7-4</th>
<th>3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>inf</td>
<td>0000</td>
<td>Input addr.</td>
<td>Weight addr.</td>
<td>Output addr.</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td></td>
<td>infsp</td>
<td>0001</td>
<td>Input addr.</td>
<td>Weight addr.</td>
<td>Output addr.</td>
<td>First partial sum addr.</td>
<td>Unused</td>
</tr>
<tr>
<td></td>
<td>csps</td>
<td>0010</td>
<td>Output neuron ID</td>
<td>Layer ID</td>
<td>First partial sum addr.</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>Path Construction</td>
<td>sort</td>
<td>0011</td>
<td>Unsorted seq. start addr.</td>
<td>Seq. length</td>
<td>Sorted seq. start addr.</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td></td>
<td>acum</td>
<td>0100</td>
<td>Input addr.</td>
<td>Output addr.</td>
<td>Cumulative threshold</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td></td>
<td>genmask</td>
<td>0101</td>
<td>Input addr.</td>
<td>Output addr.</td>
<td>Unused</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findneuron</td>
<td>0110</td>
<td>Layer ID</td>
<td>Neuron position</td>
<td>Target neuron addr.</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findref</td>
<td>0111</td>
<td>Neuron addr.</td>
<td>Receptive field addr.</td>
<td>Unused</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>Classification</td>
<td>cls</td>
<td>1000</td>
<td>Class path addr.</td>
<td>Activation path addr.</td>
<td>Result</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Omitted for simplicity (mov, dec, jne, etc.)
(a) Overlapping inference with extraction across layers in forward extraction. $L$ is the total number of DNN layers.

(b) Neuron-level pipelining in important neuron extraction. $N$ denotes the number of important neurons in the current layer's output.
Backup
Backup
Backup