Distributed DNN Serving in the Network Data Plane

Kamran Razavi†, George Karlos‡, Vinod Nigade‡, Max Mühlhäuser†, Lin Wang†‡

†University of Darmstadt
‡ Vrije Universiteit Amsterdam
Can we leverage a programmable network to perform DNN serving?
Deep Neural Network Inference

\[ W_{1121} \]

\[ W_{2131} \]
Deep Neural Network Inference

\[ N_{22} = N_{11} \times W_{1122} + N_{12} \times W_{1222} + N_{13} \times W_{1322} \]
Deep Neural Network Inference

\[ N_{22} = F(N_{11} \times W_{1122} + N_{12} \times W_{1222} + N_{13} \times W_{1322}) \]
DNN Serving in the Network
On-Switch Execution

No FPU: 1.1*2 is time consuming!

- Int-8 (first bit is the sign bit)
- Parallel multiplication

Multiplicand = 13  
\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1
\end{array}
\]

Multiplier = 5  
\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{array}
\]
On-Switch Execution

No FPU: $1.1 \times 2$ is time consuming!

- Int-8 (first bit is the sign bit)
- Parallel multiplication

Mul5picand = 13

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Multiplier = 5

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Result = $2^6 + 2^0 = 65 = 13 \times 5$

Integer Aggregation

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

8 ALUs per multiplication -> # of parallel multiplication $\sim= N/8$
On-Switch Execution

No FPU: 1.1*2 is time consuming!

- Int-8 (first bit is the sign bit)
- Parallel multiplication

Activation functions

For example, ReLU:

- If the sign bit == 1, put 0
- Else keep the number

Layer specific requirements: e.g. MaxPooling

- State storage
- Order of packets
Case Study: Mini-AlexNet

- **7 Layers**
- **15M Parameters**
- **91M Operations**
- **16MB Memory**
- **2ms Latency**

### mini-AlexNet (CIFAR-10)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 1</td>
<td>Input: 32 × 32 × 3</td>
</tr>
<tr>
<td>Layer 2</td>
<td>Conv1: 3 × 3 × 3 × 64</td>
</tr>
<tr>
<td>Layer 3</td>
<td>Conv2: 3 × 3 × 64 × 192</td>
</tr>
<tr>
<td>Layer 4</td>
<td>Conv3: 3 × 3 × 192 × 384</td>
</tr>
<tr>
<td>Layer 5</td>
<td>FC1: 4096</td>
</tr>
<tr>
<td>Layer 6</td>
<td>FC2: 2048</td>
</tr>
<tr>
<td>Layer 7</td>
<td>FC3 (Output): 10</td>
</tr>
</tbody>
</table>