Internet of Things Devices

- Internet of Things (IoT) devices
  - Have access to an abundance of raw data
  - In home, work, or vehicle
IoT: Raw Data & Processing

- IoT is gaining ground with the widespread of
  - Embedded processors
  - Ubiquitous wireless networks
- Access to raw data
  - Understand it!
  - Real-time constraints
  - Limited resources
    - Power
    - Compute
With deep neural networks (DNNs):

- With DNNs IoTs can
  - Process several new data types and
  - Understand behaviors
- Speech, vision, video, and text

But, DNNs are resource hungry

- Cannot met real-time constraints on IoT devices
- Several DNNs cannot be executed on IoTs
Approach 1: Offload to Cloud

- Send the request to cloud services

- AWS
- Google Cloud
- Microsoft
Why Cloud is not Always a Solution

- Unreliable connections to the cloud
  - Plus low bandwidth and high latency
- Disconnected Devices
- Privacy
  - Privacy preserving learning (e.g., differential privacy)
  - Privacy preserving inference (e.g., homomorphic encryption)
- Personalization
- Federated learning
Approach 2: IoT Collaboration

- Distribute computations with collaboration
  - To meet demands of DNNs
  - On top of common DNN techniques for constrained devices (e.g., pruning)
### IoT Collaboration Pros & Cons

- Assuming DNN performance barrier is solved with collaboration among IoT devices

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Dependent on Cloud</td>
<td>Unreliable Latencies</td>
</tr>
<tr>
<td>Privacy Preserving</td>
<td>Accuracy Drop due to Data Loss &amp; Device Failure</td>
</tr>
<tr>
<td>Enables Personalized Insight</td>
<td></td>
</tr>
</tbody>
</table>
Challenges Impact: Unreliable Latencies

- Histogram of arrival times in 4-node system performing AlexNet (model parallelism).

### Compute Time vs. Tail Latency

<table>
<thead>
<tr>
<th>Latency</th>
<th>Data Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>50ms</td>
<td>Computation</td>
</tr>
<tr>
<td>&lt;100ms</td>
<td>34%</td>
</tr>
<tr>
<td>&lt;150ms</td>
<td>42%</td>
</tr>
</tbody>
</table>

- Long Tail and Max Latency -> Straggler Problem
Challenges Impact: Accuracy Drop

- Common to lose data parts due to

(a) 10-class digit recognition (LeNet-5)

(b) 1000-class image recognition (Inception v3)

High Accuracy Drop
Computation of DNNs

- Each layer’s computations can be represented as matrix-matrix multiplication (GEMM kernels).

**Fully-connected layer:**

**Conv. layer:**

\[
W_{k \times F^2 C} \times I_{F^2 C \times WH} = O_{K \times WH}
\]
Computation Distribution of DNNs

Methods distributing computation of a model*

- **Output splitting:**
  - Fully-connected Layers
  - Input splitting:

```
\begin{pmatrix}
  w_{11} & w_{12} & \cdots & w_{1k} \\
  w_{21} & w_{22} & \cdots & w_{2k} \\
  w_{31} & w_{32} & \cdots & w_{3k} \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{m1} & w_{m2} & \cdots & w_{mk}
\end{pmatrix}
\times
\begin{pmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
  \vdots \\
  a_k
\end{pmatrix}
= 
\begin{pmatrix}
  \delta a_1 \\
  \delta a_2 \\
  \delta a_3 \\
  \vdots \\
  \delta a_k
\end{pmatrix}
```

- **Input splitting (divided among nodes)**

```
\begin{pmatrix}
  w_{11} & w_{12} & \cdots & w_{1k} \\
  w_{21} & w_{22} & \cdots & w_{2k} \\
  w_{31} & w_{32} & \cdots & w_{3k} \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{m1} & w_{m2} & \cdots & w_{mk}
\end{pmatrix}
\times
\begin{pmatrix}
  a_1' \\
  a_2' \\
  a_3' \\
  \vdots \\
  a_k'
\end{pmatrix}
= 
\begin{pmatrix}
  \delta a_1' \\
  \delta a_2' \\
  \delta a_3' \\
  \vdots \\
  \delta a_k'
\end{pmatrix}
```

- Inputs (every node needs a copy)
- Outputs (each node independently)

- Same can be applied on conv. layers*
  - Channel, spatial, and filter splitting

---

Coded Distributed Computing (CDC)

- Designed for MapReduce workloads (2018)*
- Performing redundant or coded computer per node to reduce communication.

This work: **DNNs on IoT**

More Compute / Node = More Reliability

---

Using CDC for Robustness

- Add column-wise summation of the weights:

\[
\begin{bmatrix}
  w_{11} & w_{12} \\
  w_{21} & w_{22} \\
  w_{11} + w_{21} & w_{12} + w_{22}
\end{bmatrix}
\times
\begin{bmatrix}
  a'_1 \\
  a'_2
\end{bmatrix}
=
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_1 + a_2
\end{bmatrix}
\]

- The new weights are constant, so done in offline

\[
\begin{bmatrix}
  w_{11} & w_{12} \\
  w_{21} & w_{22} \\
  w_{11}^{cdc} & w_{22}^{cdc}
\end{bmatrix}
\times
\begin{bmatrix}
  a'_1 \\
  a'_2
\end{bmatrix}
=
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a^{cdc}
\end{bmatrix}
\]

- Distribute outputs among nodes
  - Thus, applicable only to output-splitting methods
How to Distribute CDC and Recover?

- Add column-wise summation of the weights:
  - Simple example
    (one output/device)
  - Recovery
    - **Subtraction vs. Multiplication**
    - You also need the weights, that you would not have in the final node
  - Multiple out/device: Just create a new weight matrix
Straggler Mitigation & Failure Coverage

Do not need to wait for all devices to send data: (AlexNet)

Better Coverage versus with 2-modular redundancy (2MR):

- CDC + 2MR
- 2MR

Graphs showing performance improvement and coverage with different numbers of devices.