From Edge Video Analytics to Federated Learning

Prof. Dr. Ling Liu
School of Computer Science
Georgia Institute of Technology
www.cc.gatech.edu/~lingliu/

Acknowledgement: This is a joint work with Ka Ho Chow, Yanzhao Wu, Wenqi Wei, Margaret Loper, Emre Gursoy, Stacey Truex
Partial funding support: NSF, CISCO, IBM
Video Analytics and Object Detection

• Video Cameras are everywhere
  - every cellphone, every vehicle, every house
  - every building, every street, every highway …

• Object Detection: a core perception for video analytics

http://www.firsttoyreviews.com/drones-taking-the-future/

https://kjzz.org/content/1318066/phoenix-red-light-and-speed-cameras-end-dec-31
• **Video analytics is typically done in the Cloud**

(1) **Overwhelming Demands for Bandwidth**
- Shipping all the videos to the Cloud is NOT scalable
  - Netflix: ~3GB/hr of HD video → 6.8 Mbps per stream (recom: 25 Mbps)
  - London is estimated to have >500,000 surveillance cameras

(2) **Privacy concerns**
Challenges of Edge Video Analytics

Unlike Cloud,

• Edge is resource limiting & little elasticity

• Edge is more exposed and more vulnerable to

  ➢ Systemic disruptions
    • contention induced delay, performance/accuracy degradation
    • poor input data induced inference errors (e.g., poor lighting, foggy weather, convoluted objects, network jitter, …)
    • Mismatch between incoming video stream rate and detection processing rate

  ➢ Adversarial disruptions (inference / training)
    • Security violation
    • Privacy violation
When incoming video streaming rate (FPS) is faster than the detection processing rate (FPS) at edge node → performance/accuracy degradation due to random frame dropping

→ A solution approach:

Parallel Detection Processing

• Leveraging AI hardware

• Leveraging fast network like 5G, 6G

Fast Edge Video Analytics by Exploiting Multi-model Detection Parallelism

Single Edge node attached with multiple AI-hardware devices, each runs one detection model

Incoming video rate $\lambda$ (FPS)
Detection processing rate $\mu$ (FPS)
Output streaming video rate $\sigma$ (FPS)

$\lambda \gg \mu \rightarrow \mu = \sigma$
Experimental Results (round robin scheduler)

**ADL-Rundle-6**

Input Video FPS ($\lambda$): 30

#Frames: 525

QoE $\rightarrow$ 12 FPS ($\sigma$)

Single NCS2: $\mu = 2.3$ FPS

$$\left\lceil \frac{\lambda}{\mu} \right\rceil \sim \left\lceil \frac{30}{2.3} \right\rceil \geq 13.04$$

$$\left\lceil \frac{\sigma}{\mu} \right\rceil \sim \left\lceil \frac{12}{2.3} \right\rceil \geq 4$$

$\Rightarrow n \in [4,14]$

Table 2: Experiments with Multiple NCS2 Sticks (ADL-Rundle-6)

<table>
<thead>
<tr>
<th>Processing</th>
<th>Offline</th>
<th></th>
<th></th>
<th></th>
<th>Online</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>#NCS2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>SSD300</td>
<td>Detection FPS</td>
<td>2.3</td>
<td>2.3</td>
<td>4.6</td>
<td>6.9</td>
<td>9.1</td>
<td>11.5</td>
<td>13.7</td>
<td>16.0</td>
</tr>
<tr>
<td></td>
<td>mAP (%)</td>
<td>54.4</td>
<td>46.7</td>
<td>56.2</td>
<td>55.8</td>
<td>55.4</td>
<td>55.7</td>
<td>55.7</td>
<td>54.7</td>
</tr>
<tr>
<td>YOLOv3</td>
<td>Detection FPS</td>
<td>2.5</td>
<td>2.5</td>
<td>5.1</td>
<td>7.5</td>
<td>10.0</td>
<td>12.5</td>
<td>14.8</td>
<td>17.3</td>
</tr>
<tr>
<td></td>
<td>mAP (%)</td>
<td>62.5</td>
<td>42.7</td>
<td>56.7</td>
<td>61.2</td>
<td>\textbf{62.7}</td>
<td>\textbf{62.7}</td>
<td>\textbf{62.7}</td>
<td>\textbf{62.7}</td>
</tr>
</tbody>
</table>

Experiment setup: 7 Intel NCS2 sticks, installed on an edge node with an Intel i7-10700K CPU, 24GB main memory and Ubuntu 20.04.
Original Video ($\lambda = 14$ FPS, $\sigma = 14$ FPS)

Detection on one NCS2 without dropping (set $\sigma = \mu$) $\sigma = 2.5$, slow detection processing rate (mAP=86.9%)

Detection on one NCS2 with dropping (set $\sigma = \lambda$) $\sigma = 14$, $\lambda = 14$, $\mu = 2.5$
cause large random frame dropping (mAP=66.1%)

Parallel Detection on six NCS2s with dropping (set $\sigma = \lambda$) $\sigma = 14$, $\lambda = 14$, $\mu = 14.8$
significantly reduce random frame dropping (mAP=86.9%)

https://github.com/git-disl/EVA/tree/main
Detection Parallelization for Fast Edge Detection Performance

Edge Devices

Live Videos (12~60 FPS)

Edge Parallel Model Detection Scheduler

Parallel Model Detection Edge Service

Sequence Synchronizer
- video frame sequence sync, time sync....

Processed Video Frames

Visualization of Detected Objects (video player)

<table>
<thead>
<tr>
<th>#NCS2</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB 2.0</td>
<td>2.0</td>
<td>3.9</td>
<td>5.9</td>
<td>7.8</td>
<td>9.7</td>
<td>11.6</td>
<td>13.2</td>
</tr>
<tr>
<td>USB 3.0</td>
<td>2.3</td>
<td>4.6</td>
<td>6.9</td>
<td>9.1</td>
<td>11.5</td>
<td>13.7</td>
<td>16.0</td>
</tr>
<tr>
<td>YOLOv3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB 2.0</td>
<td>1.9</td>
<td>3.7</td>
<td>5.5</td>
<td>7.2</td>
<td>8.1</td>
<td>8.0</td>
<td>8.1</td>
</tr>
<tr>
<td>USB 3.0</td>
<td>2.5</td>
<td>5.1</td>
<td>7.5</td>
<td>10.0</td>
<td>12.4</td>
<td>14.8</td>
<td>17.3</td>
</tr>
</tbody>
</table>

Impact of Connection Speed to AI Hardware

<table>
<thead>
<tr>
<th>Port</th>
<th>USB 2.0</th>
<th>USB 3.0</th>
<th>Ethernet</th>
<th>10 Gigabit Ethernet</th>
<th>WiFi 6</th>
<th>4G (peak)</th>
<th>5G (peak)</th>
</tr>
</thead>
</table>
Challenges of Edge Video Analytics

- **Systemic disruptions**
  - Contention induced delay, performance/accuracy degradation
  - Low-value data offloading induced inference errors (e.g., poor lighting, foggy weather, convoluted objects, network jitter, …)
  - Mismatch between incoming stream rate and the detection processing throughput (#frames per second – FPS)

- **Adversarial disruptions**
  - Adversarial Attacks in Model Inference Phase (Edge Inference)
  - Adversarial Attacks in Distributed Model Training Phase (Federated Learning)
ADVERSARIAL ATTACKS ON DEEP OBJECT DETECTORS

TOG-UNIVERSAL VANISHING ATTACKS

Full video available at https://youtu.be/_IZfQofiL9Q

TOG-universal (rescale for visibility)
Object Detector (Benign)  

Object Detector (under attack)
TOG-PATCH ATTACKS: A PHYSICAL THREAT


https://github.com/git-disl/TOG
Mitigating adversarial attacks to improve the robustness of DNN-based object detection systems while maintaining high benign performance.

**FUSE**: Robust model fusion with diverse object detection models.
Adversarial attacks can severely damage deep object detectors.

FUSE offers strong robustness against deception and high benign accuracy.

<table>
<thead>
<tr>
<th>Fusion Team Member</th>
<th>Benign mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRCNN (Victim)</td>
<td>67.37</td>
</tr>
<tr>
<td>YOLOv3-D</td>
<td>83.43</td>
</tr>
<tr>
<td>SSD512</td>
<td>79.83</td>
</tr>
</tbody>
</table>

Benign mAP: 85.95%
Adversarial attacks can severely damage deep object detectors
Adversarial Robustness of Object Detection

Object Detection on three images with the standard detector Faster RCNN
Object Detection on the same images with the robust fusion detector
Adversarial Robustness with FUSE

No Attack

TOG-v Patch

TOG-m Patch
person → chair

Standard

FUSE

person 1.00

person 1.00

chair 0.96

chair 0.90

person 1.00

person 0.92

person 1.00

person 0.92

person 1.00

person 0.92
Joint Training of a robust fusion of diverse detection models
**Key Idea:** Jointly optimize a team of object detectors with a loss optimizer, regulating and encouraging models to *learn differently*

\[ L(x, O; \Phi) = \frac{1}{n} \sum_{i=1}^{n} L(F_i(x), O) + \frac{2\lambda}{n(n-1)} \sum_{j=1}^{n} \sum_{k=1}^{j-1} R(x; F_j, F_k) \]

- **Aim to minimize detection errors**
- **Aim to promote diversity in detection performance between member models**

**How?**
**DIVERSITY JOINT TRAINING - REGULARIZERS**

**Diversified Detection Loss**

**Key Idea:** Minimize correlation of prediction errors on objectness, bounding boxes, and class labels among members

**Key Idea:** Minimize correlation of kernel filters to break the error amplification chain of adversarial perturbations by extracting different features

**Diversified Kernel Filters**
Conclusion and Challenges

• FUSE approach provides effective adversarial robustness
  • adversarial perturbed inputs
  • adverse detection conditions
    • Lighting, small objects, large overlapped objects,
    • Bad whether …
Adversarial Attacks to Object Detection at Edge

→ Countermeasure with Model Fusion

- Federated Learning
  - Adversarial Attacks in Federated Learning
    - Privacy Leakages
    - Data Poisoning Attacks
Federated Learning: A Brief Overview

Global Model Parameter Aggregation
Per round $K_t (<< N)$ participants Selection

Federated Learning Hosting Service

Download global parameter update

Upload local parameter update

finding $N$ subscribers
Federated Learning: Why Attractive

- Massive Data is generated at the edge
  - Billions of phones & IoT devices constantly generate data
  - Data enables better products and smarter models

- Federated learning allows data to live at the edge
  - Data processing is moving on device:
    - Improved latency
    - Works offline
    - Better battery life
    - Privacy advantages
• **Full Model v.s. Split Model FL**

✓ **Full Model:** Every client runs the same model in every iteration of federated learning.

✓ **Split-Learning:** split the execution of a model on a per-layer basis between the clients and the server

  • can be done at both training and prediction (inference).
• Horizontal v.s. Vertical FL

✓ **Horizontal FL:** data sets across $N$ clients (subscribers) share the same feature space but different in samples (row-wise partitioning)

✓ **Vertical FL:** data sets across $N$ clients (subscribers) share the same sample ID space but differ in feature space (column-wise partitioning)
Federated Learning: Categorization

• **Network Topology**
  - FL with Flat Client-Server Structure
  - FL with Hierarchical multi-tier Peer-to-Peer Structure

- **Homogeneous v.s. Heterogeneous Connectivity**
  - Small scale: Clients are always available and participate in every round
  - Large scale: Clients’ network connectivity are intermittent and only a small percentage of N clients needs to participate in each round of FL
Federated learning

- Traditional ML: one party with centralized dataset $\mathcal{D}$ trains model $M$.
- Federated Learning: $N$ parties with datasets $\mathcal{D}_1, \ldots, \mathcal{D}_n$ jointly train a global model $\mathcal{M}$.

Privacy Protection: All data stays local! Only required to periodically share model parameters.

New Threats in Federated Learning

- All client-server communications are encrypted
- New learning system leads to new threat landscape

1. FL Aggregator
2. FL Participants
3. FL Model consumers → Prediction Phase Threats

Training Phase Threats

\[ \theta = \frac{1}{n} \sum_{i=1}^{n} \theta_i \]

Local Training Module

\[ \theta_1 \]

\[ \theta_2 \]

\[ \theta_3 \]

Aggregation

Prediction Phase Threats

Local Training Module

Local Training Module

Local Training Module
Gradient leakage Attacks

- Adversaries steal local model updates (gradients) and infer private training data by reconstruction-based inference.

Data Poisoning Attacks

- Adversaries poison the local training data on the compromised client(s) by strategically flipping the ground truth data, aiming to mislead the global model with target manipulation.
Gradient Leakage Attacks

- Gradient Leakage at federated server before server-side SGD (type-0 leakage)
- Gradient Leakage at a client prior to encryption for upload (type-1 leakage)
- Gradient Leakage at a client during local model training (type-2 leakage)
Gradient Leakage Attacks

An adversary obtains the gradients by unauthorized read, then reconstruct the training data using a random dummy seed against the leaked gradient with no prior knowledge of the neural network structure used in FL.
Evaluation on Gradient Leakage Attacks (Four Datasets)

<table>
<thead>
<tr>
<th>Attack Iter</th>
<th>CPL</th>
<th>[34]</th>
<th>[9]</th>
<th>CPL</th>
<th>[34]</th>
<th>[9]</th>
<th>CPL</th>
<th>[34]</th>
<th>[9]</th>
<th>CPL</th>
<th>[34]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>28.3</td>
<td>114.5</td>
<td>6725</td>
<td>61.8</td>
<td>125</td>
<td>6813</td>
<td>25</td>
<td>69.2</td>
<td>4527</td>
<td>11.5</td>
<td>18.4</td>
<td>3265</td>
</tr>
<tr>
<td>ASRes</td>
<td>0.973</td>
<td>0.754</td>
<td>0.958</td>
<td>0.981</td>
<td>0.85</td>
<td>0.978</td>
<td>0.981</td>
<td>0.857</td>
<td>0.974</td>
<td>0.99</td>
<td>0.686</td>
<td>0.784</td>
</tr>
<tr>
<td>ASRI</td>
<td>1</td>
<td>0.965</td>
<td>1</td>
<td>1</td>
<td>0.94</td>
<td>1</td>
<td>1</td>
<td>0.951</td>
<td>1</td>
<td>1</td>
<td>0.951</td>
<td>1</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.9985</td>
<td>0.9982</td>
<td>0.9984</td>
<td>0.959</td>
<td>0.953</td>
<td>0.958</td>
<td>0.998</td>
<td>0.997</td>
<td>0.998</td>
<td>0.99</td>
<td>0.985</td>
<td>0.989</td>
</tr>
<tr>
<td>MSE</td>
<td>2.2E-04</td>
<td>2.5E-04</td>
<td>2.2E-04</td>
<td>2.2E-04</td>
<td>5.4E-04</td>
<td>6.5E-04</td>
<td>2.2E-04</td>
<td>2.9E-04</td>
<td>2.3E-04</td>
<td>1.5E-05</td>
<td>1.7E-05</td>
<td>1.6E-05</td>
</tr>
</tbody>
</table>


Defending Against Gradient Leakage Attacks

- **Idea:** Instead of sharing the raw local parameter update, perturb per-client local gradients before server aggregation or before sending gradients from client to the server.

- **Approaches:**
  - Gradient perturbation with random additive noise
  - **Gradient perturbation with differential privacy**
    - federated learning with conventional DP
    - federated learning with gradient leakage resilient DP
• **Fed-SDP approach:** add the DP-noise to the local gradient upon completion of the local training or prior to performing server-side SGD.

\[
\nabla \tilde{w}_i(t) = \nabla w_i(t) + \mathcal{N}(0, \sigma^2 S^2)
\]

\[
W(t + 1) = W(t) - \eta \sum_{i=1}^{n} \omega_i \nabla \tilde{w}_i(t)
\]

Noisy gradients

Local model training at each iteration (local SGD) is not differentially private

For presentation convenience, we call these approaches Fed-SDP
Federated Learning with Fed-CDP

- **Fed-CDP approach:** add the DP-noise to the per-example gradient during local training before performing local SGD at each client.

Both local model training at each iteration (local SGD) and server side SGD are differentially private.

<table>
<thead>
<tr>
<th>type 0&amp;1 leakage</th>
<th>DSSGD</th>
<th>Fed-SDP</th>
<th>Fed-CDP</th>
<th>Fed-CDP (decay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>recon. dist.</td>
<td>0.2027</td>
<td>0.7298</td>
<td>0.7831</td>
<td>0.956</td>
</tr>
<tr>
<td>att. iter.</td>
<td>32</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>type 2 leakage</th>
<th>DSSGD</th>
<th>Fed-SDP</th>
<th>Fed-CDP</th>
<th>Fed-CDP (decay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>recon. dist.</td>
<td>0.0009</td>
<td>0.0013</td>
<td>0.6233</td>
<td>0.951</td>
</tr>
<tr>
<td>att. iter.</td>
<td>28</td>
<td>27</td>
<td>300</td>
<td>300</td>
</tr>
</tbody>
</table>

**DSSGD:** R. Shokri, V. Shmatikov, Privacy-preserving deep learning, ACM CCS, 2015.


Gradient leakage Attacks

- Adversaries steal local model updates (gradients) and infer private training data by reconstruction-based inference.

Data Poisoning Attacks

- Adversaries poison the local training data on the compromised client(s) by strategically flipping the ground truth data, aiming to mislead the global model with target manipulation.
  - Perception poisoning
  - Classification poisoning
Perception Poisoning in FL

(a) Benign (No Poison)
- person 0.93
- car 0.86

(b) Class-Flip
- pottedplant 0.56
- car 0.72

(c) BBox-Flip
- person 0.62
- car 0.77

(d) Objn-Flip
- car 0.84

FL with no poisoning
mis-detecting person objects
mis-localizing person objects
making person objects vanishing

Malicious Participant availability

(a) CIFAR-10

(b) Fashion-MNIST

N=100, K=10%, T=200

https://github.com/git-disl/DataPoisoning_FL
Defending Against Data Poisoning Attacks

- **Idea:** distinguish poisoning gradients and malicious clients from non-poisoning gradients and benign clients and remove malicious/compromised clients upon detection;

- **Approach:** Federated Learning with forensic analysis for anomaly detection
Defending Data Poisoning in FL

• Two Phase Defense Methodology
  • Phase I: Identifying Poisoned Local Gradients
  • Phase II: Identifying Malicious Participants

Perception Poisoning Attacks

(a) No Poison  (b) Class-Flip  (c) BBox-Flip  (d) Objn-Flip

Attack Source: Person objects → Attack Target (pottedplant / wrong BBox / Vanishing)

Ka Ho Chow and Ling Liu. “Perception Poisoning Attacks in Federated Object Detection Learning”, May 2021
The learning curves of $\text{AP}_{\text{src}}$ without or with our defense compared with the benign scenario.

The AP$_{src}$ of the benign model (2nd column), the poisoned model by BBox-Flip (3rd column), and the BBox-Flip poisoned model protected under our defense (4th column).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source AP - AP$_{src}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Benign</td>
</tr>
<tr>
<td>VOC</td>
<td>52.54</td>
</tr>
<tr>
<td>INRIA</td>
<td>63.94</td>
</tr>
</tbody>
</table>

Ka Ho Chow and Ling Liu. “Perception Poisoning Attacks in Federated Object Detection Learning”, May 2021
Concluding Remarks

• Device-edge-cloud computing will be a dominating model for next generation intelligent systems.

• Edge Analytics and Federated Learning will be widely deployed in many fields of business, science and engineering.

• End-to-end resilience against systematic disruptions and adversarial disruptions should be the first principle for next generation intelligent systems.
Thank You

Contact: Prof. Dr. Ling Liu
lingliu AT cc.gatech.edu