



Deep Learning For IoT Fingerprinting

Maxwel Bar-on

Supervisor: Indrakshi Ray

Fall 2025

Outline

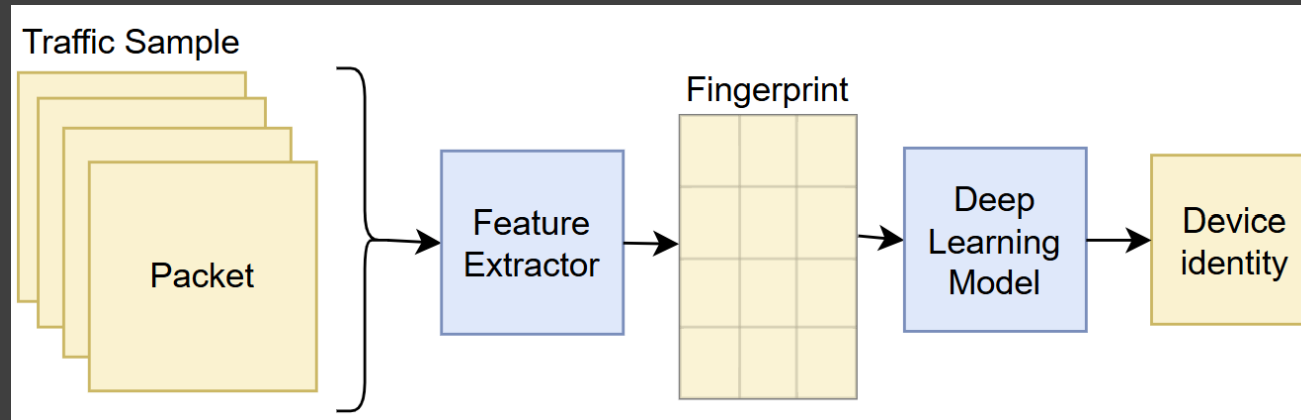
- Introduction
 - Motivation
 - Challenges
 - Proposed Approach
- Part 1: Federated learning and Mixture-of-Experts
- Part 2: Relative encoding of communication endpoints
- Part 3: Bi-component architecture
- Part 4: Fixed-time sampling
- Conclusion and future work

IoT Security Problem

- IoT devices have become nearly ubiquitous
- Unfortunately, many devices have weak security
- Attackers can compromise vulnerable devices
- Consequences can be devastating
 - Massive DDoS attacks from IoT botnets
 - Theft of highly sensitive information
 - Sabotage

Our approach: IoT fingerprinting

- IoT fingerprinting: process of identifying an unknown IoT device from its network behavior
 - Collect sample of traffic
 - Extract fingerprint from sample
 - Identify corresponding device by applying deep learning model to fingerprint
- IoT fingerprinting can be integrated with access control systems to secure networks
 - Enforcing device-specific access controls



Challenges of deep learning for IoT fingerprinting

- **Limited data availability**
 - Training an accurate fingerprinting model requires large and diverse datasets
 - Difficult for a single observer to collect sufficient data
 - Combining data from multiple observers raises privacy concerns
- **Imbalanced datasets**
 - Communication rates vary among different devices
 - Models will be biased against underrepresented devices
- **Capturing relationships in multi-flow traffic**
 - Devices may generate several separate network flows for a single task
 - Including multiple flows can confuse the model
 - Flow-based filtering ignores cross-flow relationships

More challenges

- **Adaptation of existing fingerprinting models**
 - New types of devices are regularly introduced
 - Models can only identify the devices they are trained on
 - existing models have to be adapted to identify new devices
 - It is inefficient to adapt a model when it is not reused
 - Reusing a model can lead to poor performance
 - Catastrophic forgetting
 - New device behaviors outside of learned distribution
- **Unbounded fingerprinting time**
 - Fixing the number of packets per sample requires unbounded collection time
 - Older packets may not be relevant to the current behavior of a device

Proposed solutions

- **Federated learning**
 - Addresses challenge of limited data availability
 - Separate observers collectively train a fingerprinting model without sharing data
- **Communication rate-based hierarchical Mixture-of-Experts (MoE)**
 - Addresses challenge of imbalanced training data
 - Partitions data based on device communication rate
 - Expert fingerprinting model trained on each subset, data in subset is more balanced
- **Relative encoding of communication endpoints**
 - Addresses challenge of capturing relationships in multi-flow traffic
 - Encodings enable fingerprinting models to understand packet relationships
- **Bi-component fingerprinting architecture**
 - Addresses challenge of model adaptability
 - Part of model is re-used without loss of performance
- **Fingerprinting with fixed-time sampling**
 - Fixed time sampling addresses challenge of unbounded fingerprinting time

Part 1: Federated Learning and Mixture-of-Experts

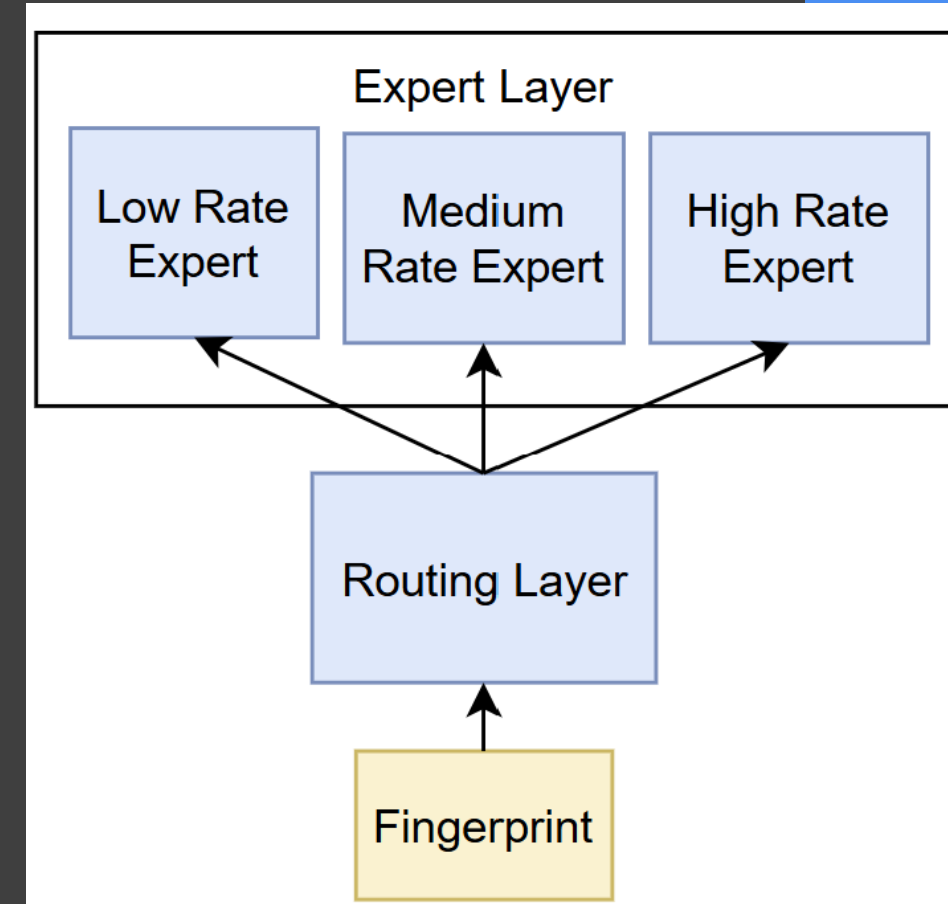
Bar-on, M., Bezawada, B., Ray, I., Ray, I. (2024). A Small World–Privacy Preserving IoT Device-Type Fingerprinting with Small Datasets. In: Mosbah, M., Sèdes, F., Tawbi, N., Ahmed, T., Boulahia-Cuppens, N., Garcia-Alfaro, J. (eds) Foundations and Practice of Security. FPS 2023. Lecture Notes in Computer Science, vol14551. Springer, Cham.
https://doi.org/10.1007/978-3-031-57537-2_7

Approach: federated learning

- Separate observers collaboratively train a global fingerprinting model
 - Each observer has a private dataset that is not directly shared
 - All datasets contribute to global model
- Observers use private data to update local fingerprinting models and share updates with a server
- Server updates global model by aggregating observers' updates
- We evaluate two aggregation techniques:
 - Selective parameter swapping:
 - Observer sends updated parameters
 - Server swaps portion of global parameters for observer's parameters
 - FedAvg:
 - Observers send loss gradients to server
 - Server averages observers' gradients and applies them to global model

Approach: Hierarchical MoE

- Separates devices into groups based on communication rate
 - Three groups: low, medium, and high rate
 - Devices in a group have similar communication rates, similar number of samples
- Each group is associated with an expert fingerprinting model
 - Trained on data from devices in its group
 - Data in groups are more balanced than the overall dataset
- Hierarchy has two layers:
 - Routing layer
 - Selects experts for fingerprints based on communication rate
 - Uses a pre-defined routing policy
 - Expert layer
 - Collection of expert fingerprinting models, one for each group of devices
 - When a fingerprint is routed to an expert, it identifies the device from the devices in its group



Experiment: federated learning and MoE

- Architecture: Multilayer Perceptron (MLP)
- Fingerprint representation: flat 183-dimensional feature vector
 - 3 window features
 - 20 packets, 9 features per packet

	Feature	Explanation
Packet	TCP	1 if transport protocol is TCP else 0
	UDP	1 if transport protocol is UDP else 0
	HTTP	1 if src or dst port is 80 else 0
	TLS	1 if src or dst port is 443 else 0
	Header Length	Length of packet header in bytes
	Payload Length	Length of payload in bytes
	Payload Entropy	Byte entropy of payload
	Common Src	1 if src port is ≤ 1023 else 0
	Common Dst	1 if dst port is ≤ 1023 else 0
Window	Max flow	Packet count of largest flow in window
	Unique IPs	Count of unique IP addresses in window
	Local IPs	Count of unique IP addresses from local subnet

Features

	Device	Packet Count
High Rate	Arlo Camera	25000
	Somfy	25000
	Home Eye Camera	25000
	Philips Hue Bridge	17840
	Taxi Cab WiFi AP	17514
	Amazon Echo	12195
Moderate Rate	Eufy Homebase	11431
	Netatmo Camera	10625
	NestCam	7490
	Sonos One Speaker	6567
	Borun Camera	4970
	Google NestMini	3269
Low Rate	Roomba Vacuum	2292
	LG Smart TV	1785
	Ring	1460
	Globe Lamp	1253
	Standard WiFi APs	1203
	Amcrest Camera	682
	Yutron Plug	473
	SimCam	452
	Atomi Coffee Maker	413
	D-Link Omna	321

Devices and groupings

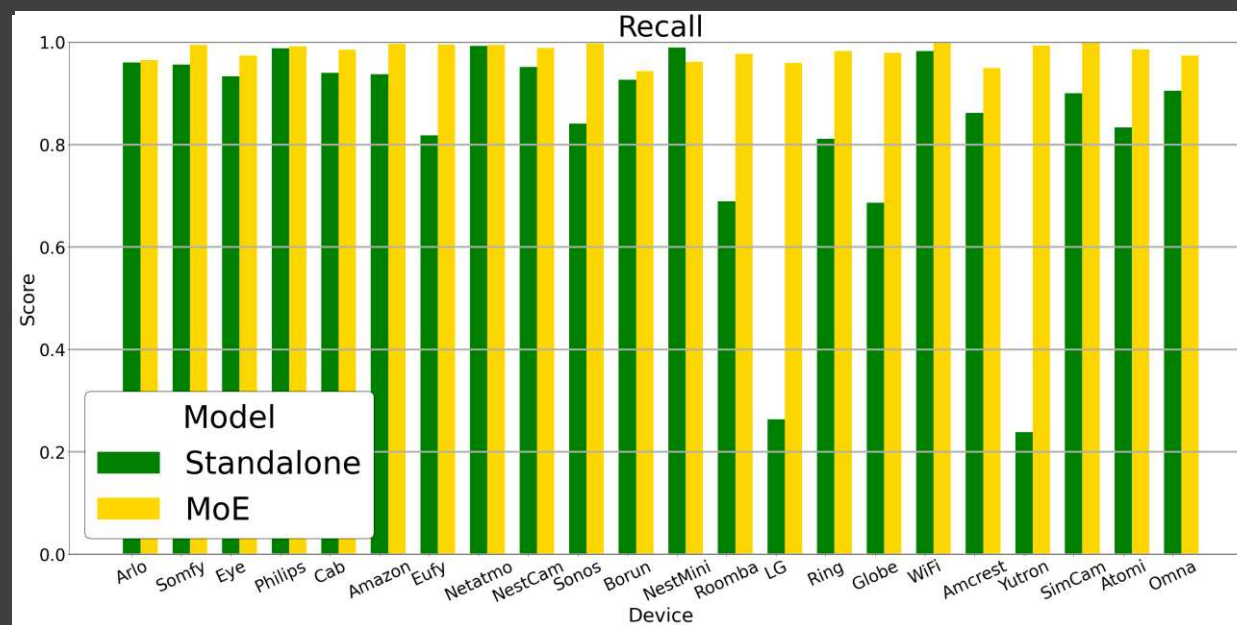
Results: federated learning

- Baseline: centralized learning, each observer trains an individual model on its private dataset without aggregating updates
- Both aggregation techniques perform better than baseline
- FedAvg performs slightly better than parameter swapping
- 2.9% - 8.4% improvement over centralized for Parameter
- 5% - 8.9% improvement for FedAvg

Learning Mode	# Observers	Recall	Precision	F1
Centralized	2	93.9%	94.3%	94.1%
	4	90.1%	90.8%	90.4%
FedAvg	2	98.6%	98.5%	98.6%
	4	98.1%	98.2%	98.2%
Parameter	2	96.6%	97%	96.8%
	4	97.7%	97.6%	97.6%

Results: MoE

- Baseline: standalone model trained on all devices
- MoE increases training data balance by 15.6%
- Improves recall by 17.2% on average
- Minimum 90% recall across devices



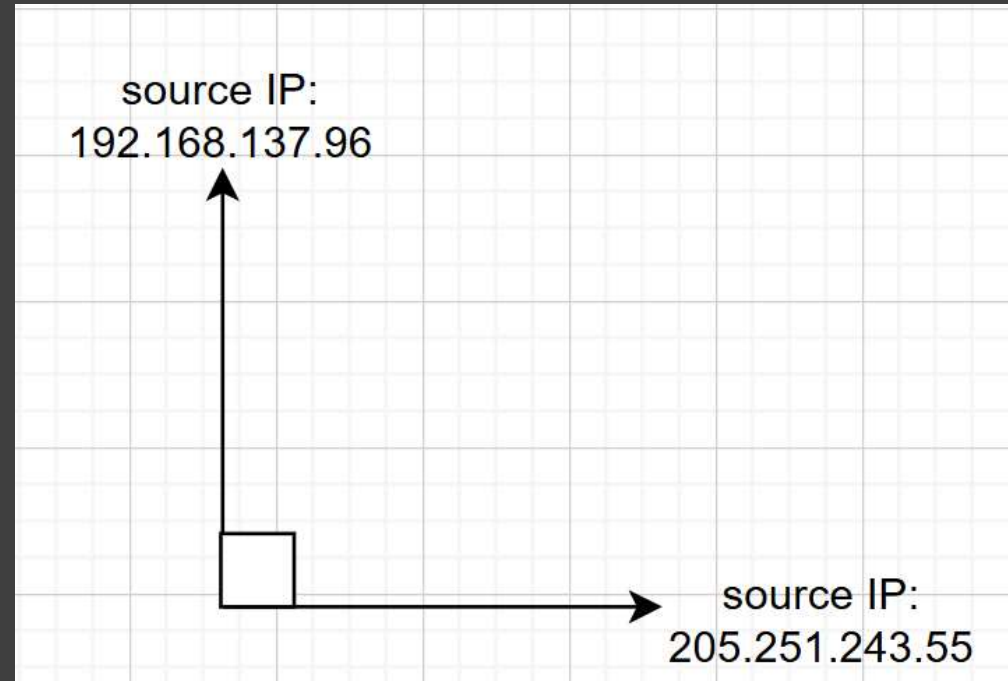
Model	Recall	Precision	F1	Balance
Hierarchical MoE	98.1%	98.2%	98.2%	94.8%
Standalone	83.7%	84.7%	83.8%	82%

Part 2: Relative Encoding of Communication Endpoints

Bar-on, M., Krosky, K., Larrieu, F., Bezawada, B., Ray, I., & Ray, I. (2025, June). Jibber-Jabber!: Encoding the (Un-) Natural Language of Network Devices and Applications. In IFIP Annual Conference on Data and Applications Security and Privacy (pp. 3-22). Cham: Springer Nature Switzerland

Approach: relative encoding of communication endpoints

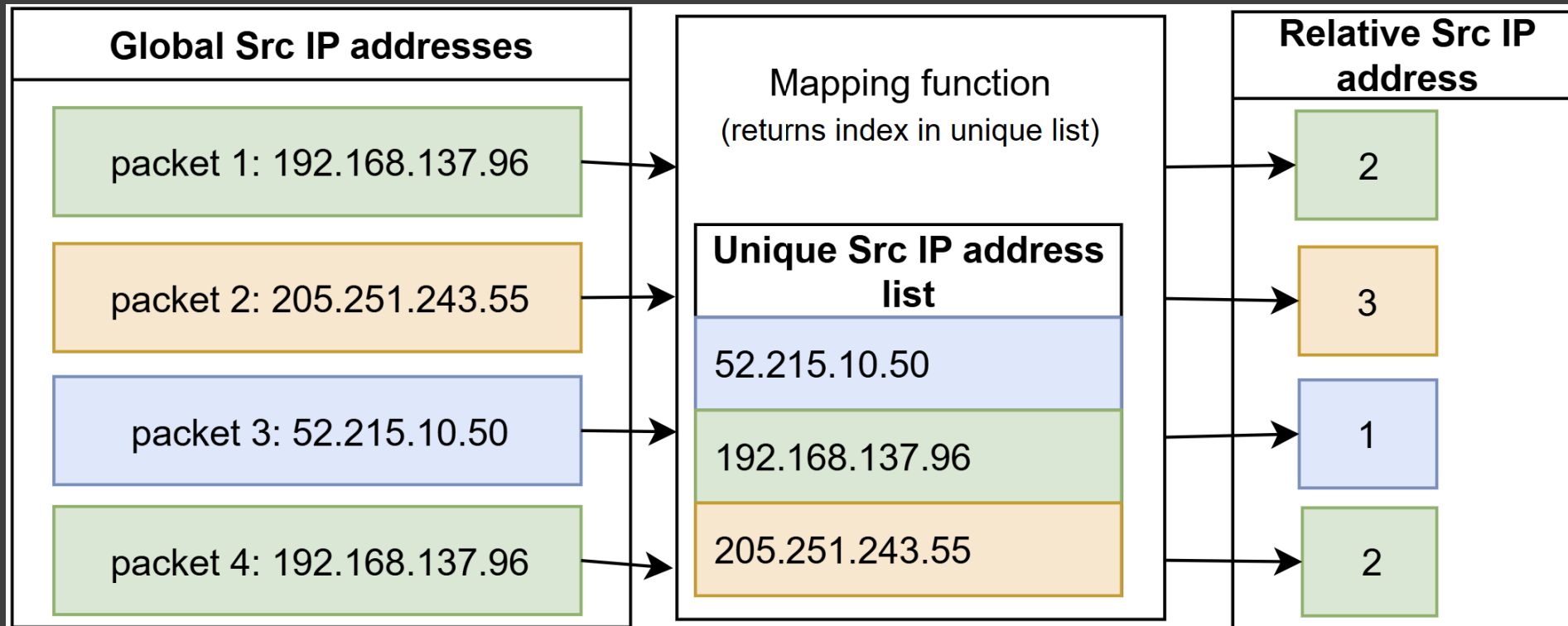
- Relative features: encode packet endpoints relative to other packets in a sample
 - Source IP address, destination IP address, source port, destination port
- Combined with transport-protocol, they act as a flow identifier
 - Useful for identifying relationships between packets in multi-flow traffic
- Encoded in low dimensional vector space to avoid overfitting
 - Minimum dimensionality to represent unique symbols in a traffic sample
 - Preserves relationship information through vector orientation



Visualization of relative encodings for two IP addresses

Relative features: encoding process

- Endpoint identifiers mapped from global to local scale
- Global scale contains all possible values, local scale contains all possible values in the traffic sample



Relative features: example

$$\rho^{(1)} : (192.168.137.96, 205.251.243.55, 42000, 443) \rightarrow [1, 2, 3, 2]$$

$$\rho^{(2)} : (205.251.243.55, 192.168.137.96, 443, 42000) \rightarrow [2, 1, 1, 3]$$

$$\rho^{(3)} : (192.168.137.96, 205.251.243.55, 45000, 137) \rightarrow [1, 2, 4, 1]$$

$$\rho^{(4)} : (192.168.137.96, 205.251.243.55, 42000, 443) \rightarrow [1, 2, 3, 2]$$

$$\rho^{(5)} : (192.168.137.96, 205.255.10.123, 40000, 137) \rightarrow [1, 3, 2, 1]$$

Example relative features
for sample of 5 packets

	$\rho^{(1)}$	$\rho^{(2)}$	$\rho^{(3)}$	$\rho^{(4)}$	$\rho^{(5)}$
$\rho^{(1)}$	0.4538	0.0083	0.0614	0.4538	0.0226
$\rho^{(2)}$	0.0171	0.9317	0.0171	0.0171	0.0171
$\rho^{(3)}$	0.0950	0.0129	0.7021	0.0950	0.0950
$\rho^{(4)}$	0.4538	0.0083	0.0614	0.4538	0.0226
$\rho^{(5)}$	0.0397	0.0146	0.1080	0.0397	0.7979

Attention matrix of
relative features for
sample

Experiment: relative features

- Architecture: Transformer encoder + MLP with average-pooling
 - Attention layers can use relative features to find relationships
- Fingerprint representation: feature matrix
 - Each row represents one packet in a sample

Feature	Explanation
Subnet Src	1 if source IP is in subnet else 0
Subnet Dst	1 if destination IP is in subnet else 0
Broadcast Dst	1 if destination IP is broadcast else 0
TLS Handshake	1 if packet includes TLS handshake header else 0
TCP	1 if packet includes TCP header else 0
UDP	1 if packet includes UDP header else 0
HTTP	1 if packet uses HTTP port else 0
SSL/TLS	1 if packet uses SSL port else 0
Common Src	1 if packet uses common source port else 0
Common Dst	1 if packet uses common dst port else 0
Header Length	length of transport-layer header (bytes)
Payload Length	length of payload (bytes)
Payload Entropy	information entropy of payload

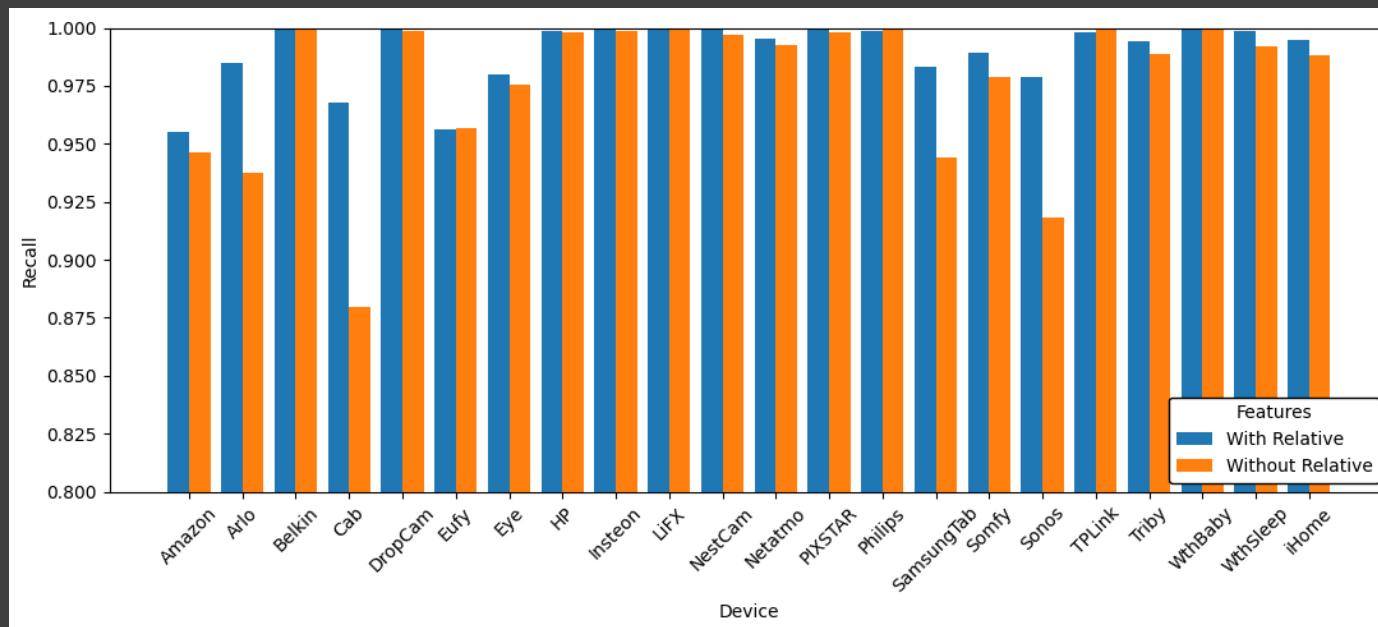
Non-relative features

Device	Packet Count
Somfy	25000
WthSleep	25000
DropCam	25000
WthBaby	25000
Belkin	25000
SamsungTab	25000
HP	25000
PIXSTAR	25000
Insteon	25000
Tribby	25000
Arlo	25000
Eye	25000
LiFX	24823
Cab	17513
Philips	16920
iHome	13872
TPLink	12465
Eufy	11207
Netatmo	10576
Amazon	8982
NestCam	7484
Sonos	6382

Devices

Results: relative features

- 1.3% improvement in recall
- 1.7% improvement in precision on average
- Large improvement for “Cab” and “Sonos”



Features	Recall	Precision	F1
With Relative Features	99.0%	98.9%	98.9%
Without Relative Features	97.7%	97.2%	97.4%

Part 3: Bi-component Architecture

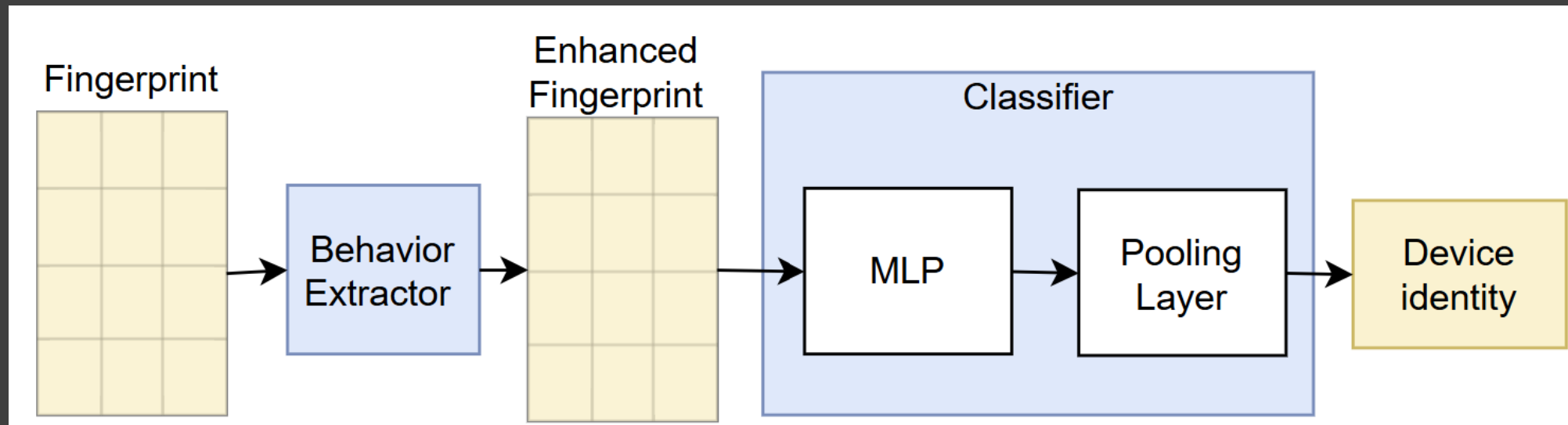
Bar-on, M., Patterson, K., Bezawada, B., Ray, I., & Ray, I. (2025, July). "Bring your own device!": Adaptive IoT Device-type Fingerprinting using Automatic Behavior Extraction [Work In Progress Paper]. In Proceedings of the 30th ACM Symposium on Access Control Models and Technologies (pp. 201-206).

Approach: adaptable bi-component architecture

- Bi-component architecture separates training into two stages
 - Extraction of network behavior patterns from traffic samples
 - Device identification from extracted patterns
- Each stage operates on a different component
 - **Behavior Extractor (BE)** trained in first stage
 - **Classifier** trained in second stage
- Enables efficient adaption through reuse of BE
 - When new devices are introduced, BE is reused with no additional training
 - Avoids catastrophic forgetting
- Generalized training technique for BE allows adapted model to accurately identify new IoT devices
 - Behavior extraction is learned as a separate task from device identification

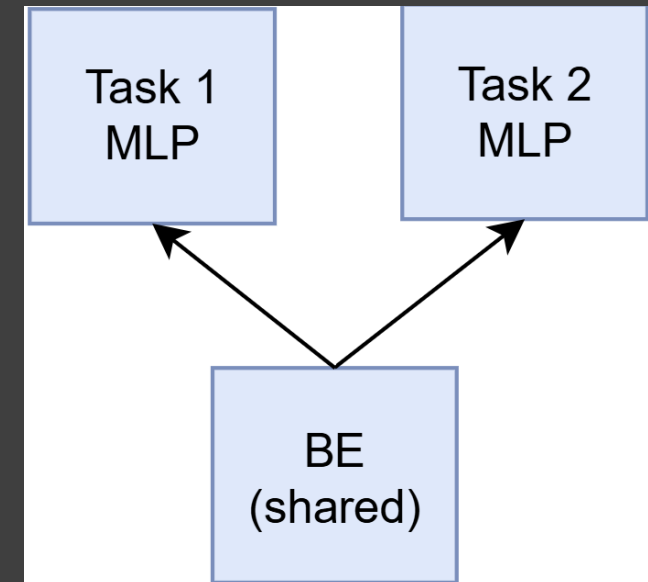
Bi-component architecture

- BE: Transformer that extracts behaviors and produces enhanced fingerprint
- Classifier: MLP with average-pooling layer that identifies devices using enhanced fingerprint



BE training

- BE trained on self-supervised machine learning tasks
 - Allows it to extract behaviors from new devices without additional training
- Train temporary model consisting of the BE followed by a task-specific MLP
 - BE updated by back-propagating through MLP
- BE training procedures consist of one or more self-supervised task
 - For each task, initialize separate MLPs
 - Simultaneously train the MLPs and BE until all tasks have converged



BE training: self-supervised tasks

- **Masked autoencoding**
 - Apply mask to BE input features, randomly remove 40% of values
 - Train model to reconstruct input features
- **Denoising**
 - Add random noise to BE input features by adding values in the range $(-0.15, 0.15)$
 - Train model to reconstruct original values
- **Anomaly locating**
 - Randomly insert anomalous packet into each sample
 - Train model to predict relative position of anomaly for each packet
 - For each packet X, the model chooses from the options:
 - X is before the anomaly in the sample
 - X is the anomaly
 - X is after the anomaly in the sample

Classifier training

- Classifier trained to identify devices from enhanced fingerprints produced by the BE
 - After training, BE and Classifier are combined to create a fingerprinting model
- An existing model can be adapted by training a new Classifier
 - Use BE to produce enhanced fingerprints from new device traffic
 - Combine enhanced fingerprints from new devices with enhanced fingerprints from original devices
 - Initialize new Classifier and train it on the combined data
 - Replace original Classifier with the new Classifier
- This is efficient because the Classifier is lightweight
 - Less than 4% of model parameters
 - No attention: computation cost is linear in the sample size

Experiments: adaptable fingerprinting

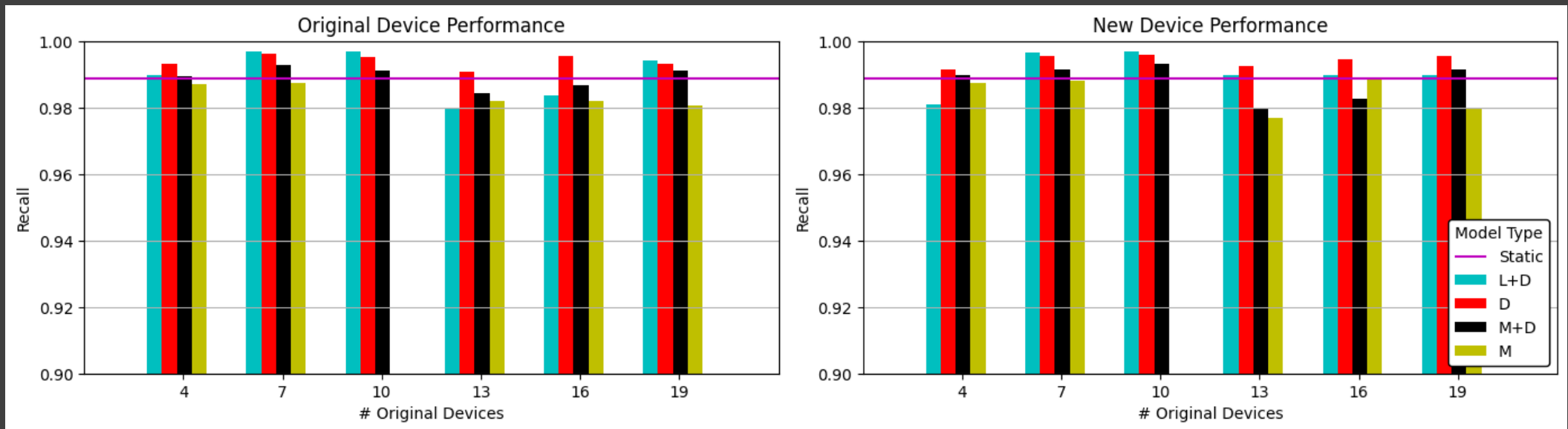
- Baseline: static model, same size but both components are merged and trained together in a single stage
- BE training procedures:
 - L + M: anomaly locating and mask autoencoding
 - D: only denoising
 - L + D: anomaly locating and denoising
 - M + D: masked autoencoding + denoising
 - M: only masked autoencoding
- Same devices as FL experiment
- Same features as relative feature experiment

Results: adaptable fingerprinting

- Bi-component require less time to adapt
- Model with D-procedure has the fastest adapt time
 - 78.3x faster than static model
- Adapted models perform well on new and original devices
- Model with D-procedure consistently outperforms static model

	Bi-component models					Static Model
	L+M	D	L+D	M+D	M	
GPU	384	60	67	303	391	4,698
CPU	22,921	3,619	4,021	18,095	23,323	78,513

Time to adapt in seconds



Performance on new and original devices with varying numbers of original devices

Part 4: Fixed-Time Sampling

Bar-on, M., Alqobaisi, A., Bezawada, B., Ray, I., & Ray, I. (2025, November). It's about time!: Exploiting Timing Variance for IoT Device-type Fingerprinting. Accepted to 7th IEEE International Conference on Trust, Privacy, and Security in Intelligent Systems, and Applications

Approach: fixed-time sampling

- IoT devices are identified using samples of traffic containing all packets captured from a device over a fixed window of time
- Ensures that packets within a sample will be relevant to the most recent behavior of a device
- Improves quality-of-service (QoS) guarantees for fingerprinting time
 - The fingerprinting time will be the same for devices with varying communication rates

Fixed-time sampling: architecture

- Traffic sampling:
 - Traffic samples include packets observed within a time-window
 - Maximum of 50 packets per sample for efficient vectorized training
- Two types of features:
 - Packet features
 - Represents attributes of individual packets in a time-window sample
 - Feature matrix with padding rows for samples with fewer than 50 packets
 - Window features
 - Aggregated statistics representing frequency of important communication activities in a time-window
 - Feature vector
- Modified architecture for handling variable-length input and multiple feature types
 - Transformer encoder
 - Encodes dependencies in packet feature matrix
 - Uses attention mask for ignoring padding rows
 - MLP classifier
 - Input layer integrates information from window features with encodings from Transformer
 - Pooling layer ignores outputs corresponding to padding

Window Features

- Number of packets
- Number of DNS queries
- Number of DNS responses
- Number of unique source IP addresses
- Number of unique destination IP addresses
- Number of unique source ports
- Number of unique destination ports

Experiment: fixed-time sampling

- Baseline: fixed-length packet sampling, where every traffic sample has a fixed number of packets and the packet capture time is unbounded
- We evaluate four time-window durations:
 - 0.1 seconds
 - 10 ms
 - 1 ms
 - 0.1 ms

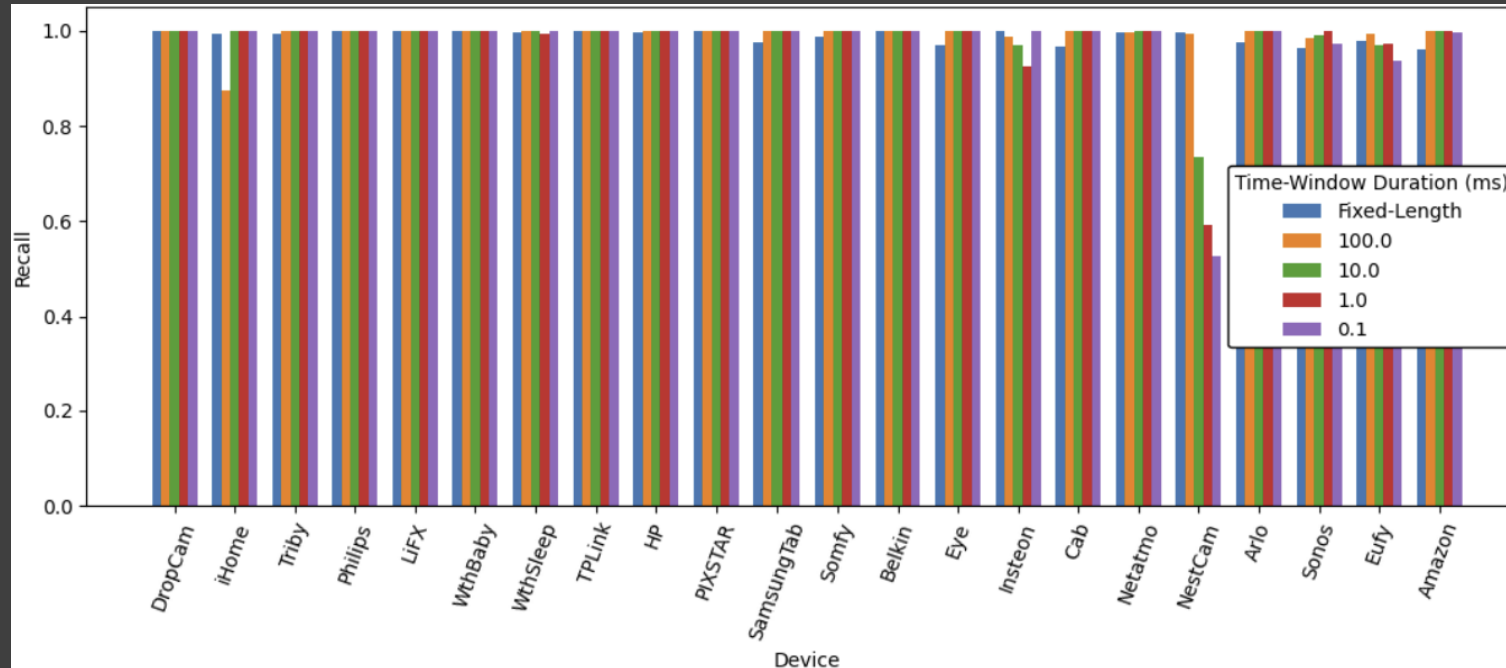
Device	# Packets	Average Packets Per Sample			
		Time-Window			
		0.1s	10ms	1ms	0.1ms
PIXSTAR	25000	3.059	2.56	1.397	1.095
Insteon	25000	47.785	10.867	1.678	1.01
WthBaby	25000	1.391	1.333	1.274	1.005
Triby	25000	1.401	1.146	1.105	1.003
DropCam	25000	1.046	1.021	1.0	1.0
Arlo	25000	39.568	36.335	35.4	34.028
Somfy	25000	17.321	6.376	2.888	1.981
SamsungTab	25000	4.559	2.938	1.441	1.045
Eye	25000	27.264	10.482	7.831	6.091
Belkin	25000	25.073	6.755	1.806	1.271
HP	25000	2.864	1.638	1.096	1.023
WthSleep	25000	1.559	1.428	1.023	1.001
LiFX	24823	1.737	1.326	1.059	1.0
Cab	17513	22.355	12.474	4.402	3.327
Philips	16920	2.515	1.304	1.137	1.028
iHome	13872	1.245	1.098	1.047	1.001
TPLink	12465	1.604	1.464	1.106	1.015
Eufy	11207	42.521	39.113	34.399	32.173
Netatmo	10576	44.97	24.054	18.157	17.336
Amazon	8982	49.577	49.387	49.361	49.355
NestCam	7484	28.736	26.488	25.696	25.546
Sonos	6382	36.661	36.481	32.903	31.897

Devices and average packet counts per sample

Results: fixed-time sampling

- Higher precision than fixed-length sampling
- Longer time-window durations result in higher average recall
- Reducing capture time by 1,000x reduces recall by less than 2%
- 0.1 second time-window results in higher average performance than fixed-length
- Fixed-time performs on-par with fixed-length for nearly all devices

Sampling		Recall	Precision	F1
Fixed-length		98.9%	98.8%	98.8%
Fixed-time	0.1s	99.2%	99.5%	99.4%
	10ms	98.5%	99.3%	98.9%
	1ms	97.6%	99.0%	98.3%
	0.1ms	97.4%	99.0%	98.2%



Conclusion and Future Work

Conclusion

- Addressed 5 challenges of deep learning for IoT fingerprinting
 - Limited training data availability from individual observers → federated learning
 - Imbalanced training datasets → hierarchical MoE
 - Capturing relationships in multi-flow traffic → relative features
 - Adaptability of existing models → bi-component architecture with partial reuse
 - Unbounded fingerprinting time → fixed-time sampling
- Evaluation:
 - Federated learning results in 3% to 9% better performance than centralized learning on average
 - MoE increases recall by 17.2% on average
 - Relative features improve performance by over 1%
 - D (denoiser only) procedure results in fastest adapt time and highest performance on new and original devices
 - Traffic capture time can be constrained without reducing performance

Future work

- Evaluate on larger datasets with more IoT devices
- Explore alternative deep learning architectures
- Enhance federated learning for non-i.i.d. data
- Alternative encoding techniques for packet endpoints to better capture bi-directional flow relationships
- Knowledge-retrieval-based approach for more efficient adaptability

Thank you

- Questions?