CBG

Cyber@Ben-Gurion
University of the Negev

NDSS '18 SYMPOSIUM

Kitsune

AN ENSEMBLE OF AUTOENCODERS FOR ONLINE NETWORK INTRUSION DETECTION

Yisroel Mirsky, Tomer Doitshman,
Yuval Elovici, and Asaf Shabtai

Introduction

- **Neural Networks (NN) are great at detecting malicious packets**
  - Great results in literature
    (NNs can learn nonlinear complex patterns and behaviors)
  - But, not so common in practice (*where is my SNORT plugin?*)
- **Existing NN solutions use supervised learning** (e.g., classification):

  1. Collect packets
  2. Label packets: malicious or normal
  3. Train deep NN on labeled data
  4. Deploy the NN model to the device
  5. Execute the model on each packet
  6. When a new attack is discovered, go to #1

Introduction

- **Neural Networks (NN) are great at detecting malicious packets**
  - Great results in literature
    (NNs can learn nonlinear complex patterns and behaviors)
  - But, not so common in practice (*where is my SNORT plugin?*)
- **Existing NN solutions use supervised learning** (e.g., classification):

1. Collect packets
   - *Large storage, many samples of every kind of malicious packet*
2. Label packets: malicious or normal
   - *Expert with a lot of time*
3. Train deep NN on labeled data.
   - *Large GPU server, and time…*
4. Deploy the NN model to the device
5. Execute the model on each packet
   - *Handle thousands of packets a second (e.g., a simple router)*
6. When a new attack is discovered, go to #1

Kitsune Overview

*A **Kitsune**, in Japanese folklore, is a mythical fox-like creature that has a number of tails, can mimic different forms, and whose strength increases with experience.*

*So too, **Kitsune** has an ensemble of small neural networks (autoencoders), which are trained to mimic (reconstruct) network traffic patterns, and whose performance incrementally improves overtime.*

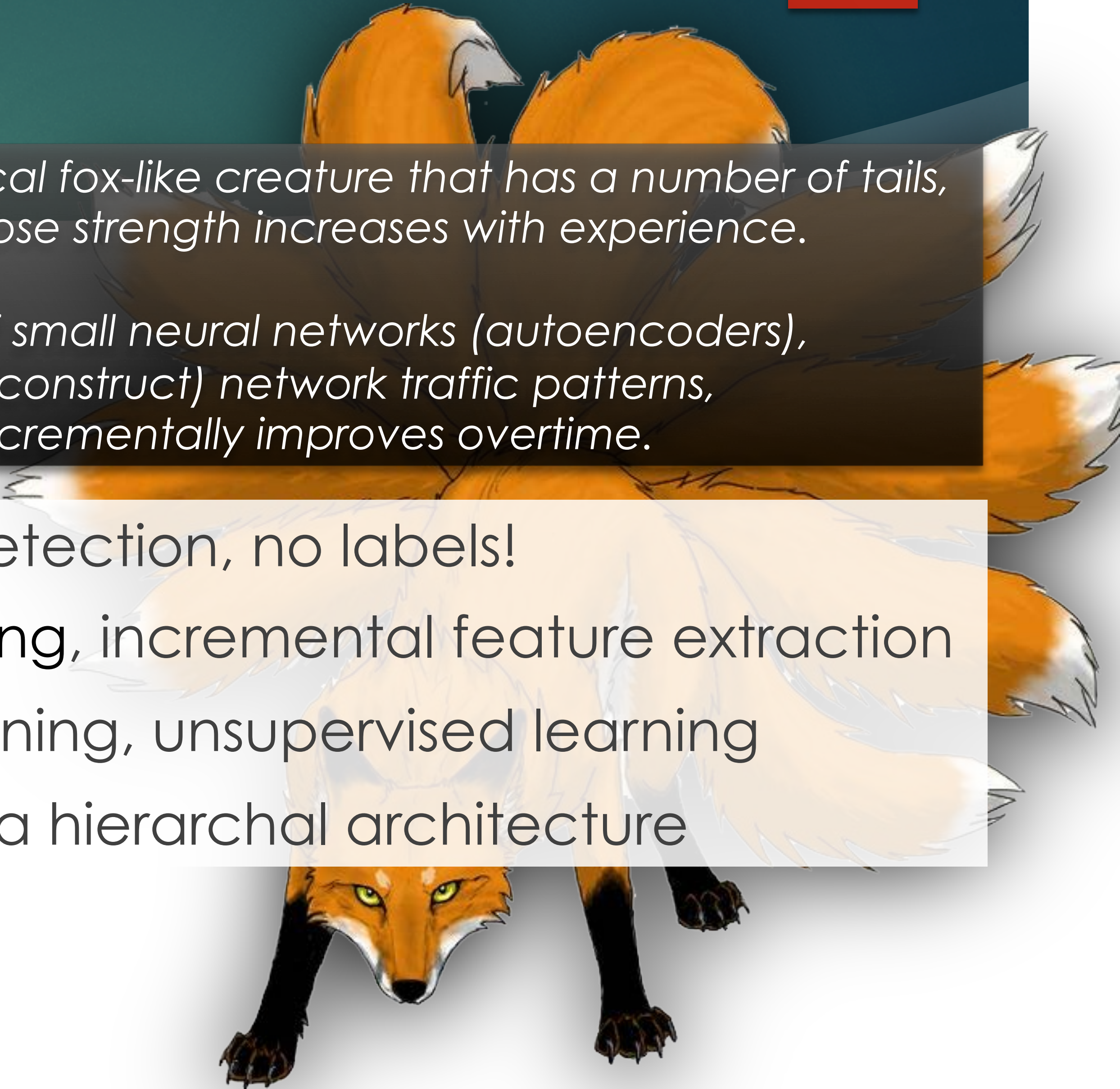
Enables NN on network traffic

- **Unsupervised:** Anomaly detection, no labels!
- **Online**: Incremental learning, incremental feature extraction

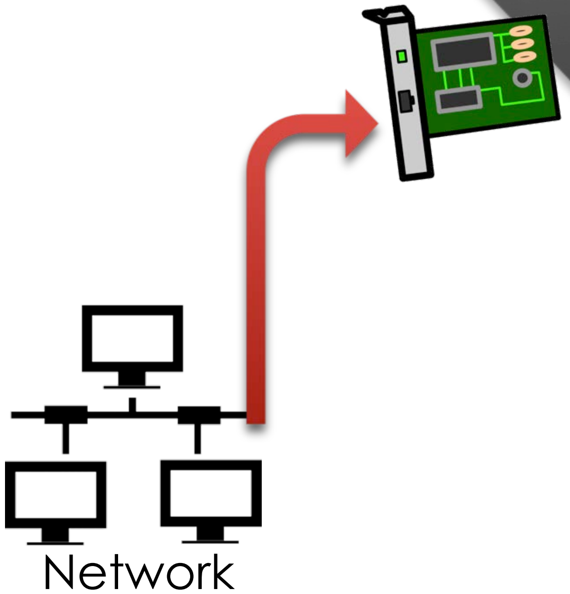
Enables realistic deployments

e.g., routers

- **Plug-and-Play**: On-site training, unsupervised learning
- **Light-weight**: The NN uses a hierarchal architecture



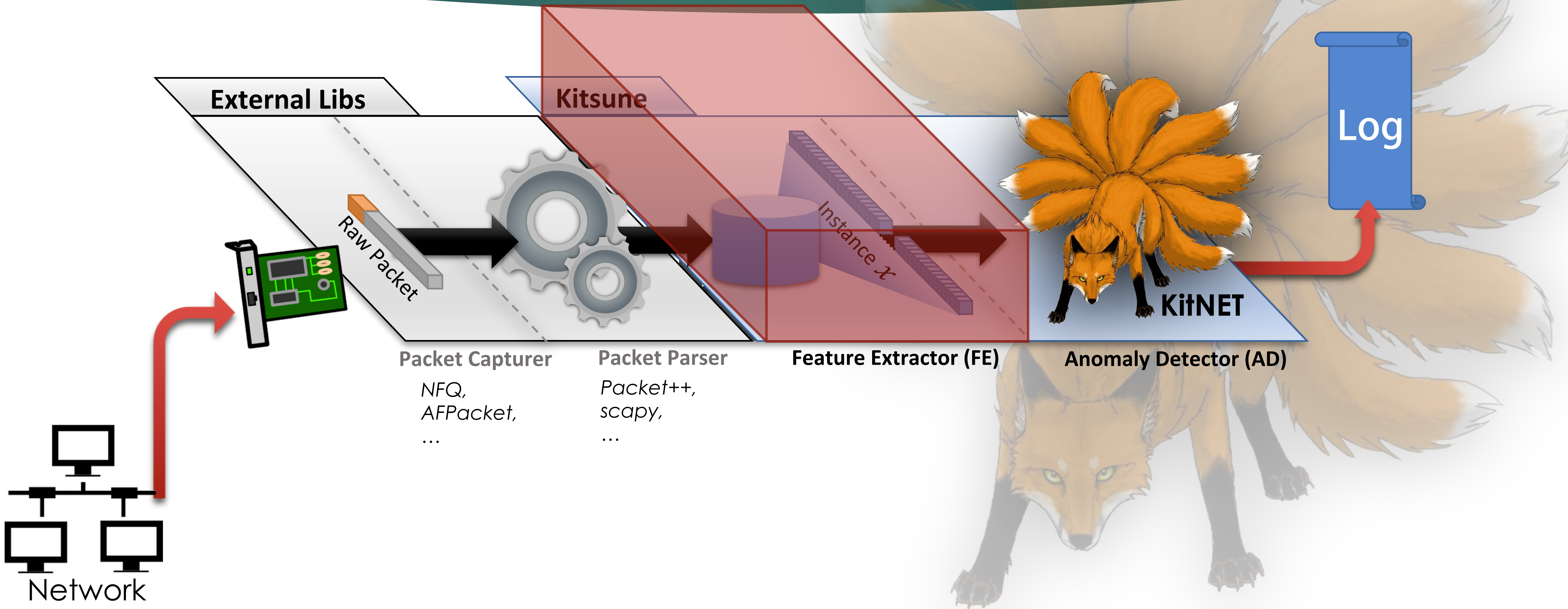
Kitsune Framework



NIDS are Located on:

- Gateways/Routers
- Servers
- Dedicated Devices
  (e.g., PI attached to a mirror port)

Kitsune Framework



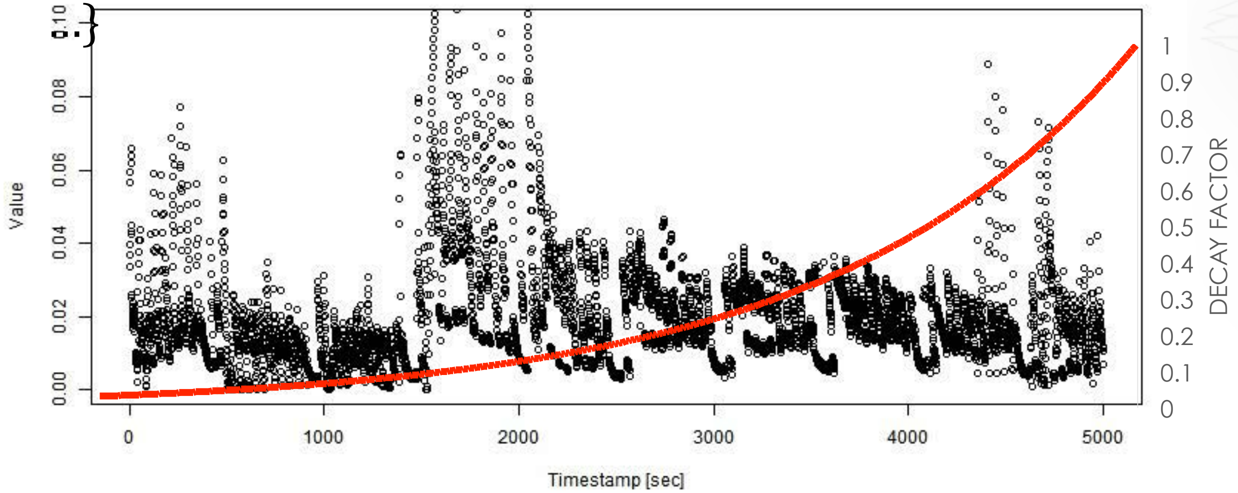
Kitsune Feature Extractor (FE)

- FE uses **damped incremental statistics** to *efficiently* measure *recent* traffic patterns

Decay Factor:

$d_\lambda(t) = 2^{-\lambda t}$

An unbounded stream of values $S=\{x_1, x_2, \dots\}$



Objective: Compute the stats ($\mu, \sigma, \dots$) over the recent history of S, given limited memory and non-uniform sample rates (timestamps)

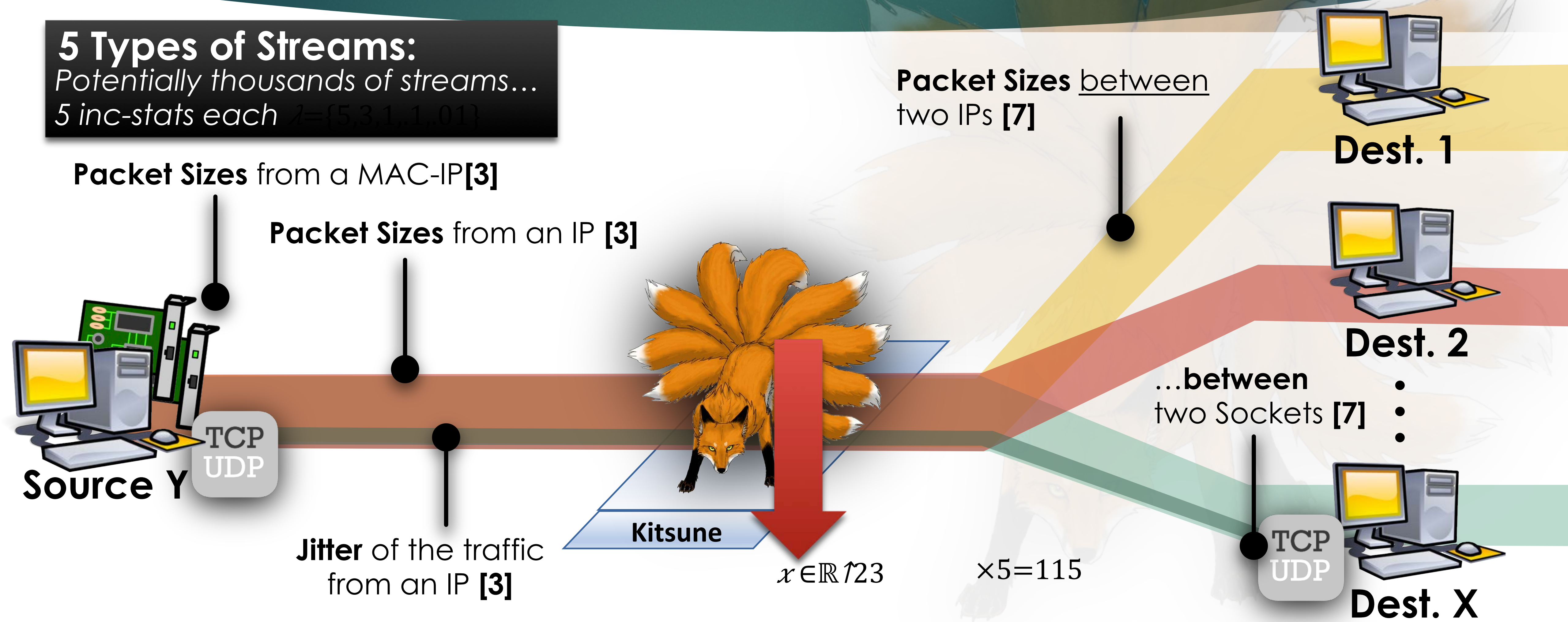
$\gamma \leftarrow$

$IS \leftarrow ($

Type	Statistic	Notation	Calculation
1D	Weight	w	w
	Mean	μ_{S_i}	LS/w
	Std.	σ_{S_i}	$\sqrt{\lvert SS/w - (LS/w)^2 \rvert}$
2D	Magnitude	$\lVert S_i, S_j \rVert$	$\sqrt{\mu_{S_i}^2 + \mu_{S_j}^2}$
	Radius	R_{S_i,S_j}	$\sqrt{\left(\sigma_{S_i}^2\right)^2 + \left(\sigma_{S_j}^2\right)^2}$
	Approx. Covariance	Cov_{S_i,S_j}	$\frac{SR_{ij}}{w_i + w_j}$
	Correlation Coefficient	P_{S_i,S_j}	$\frac{Cov_{S_i,S_j}}{\sigma_{S_i}\sigma_{S_j}}$

Kitsune Feature Extractor (FE)

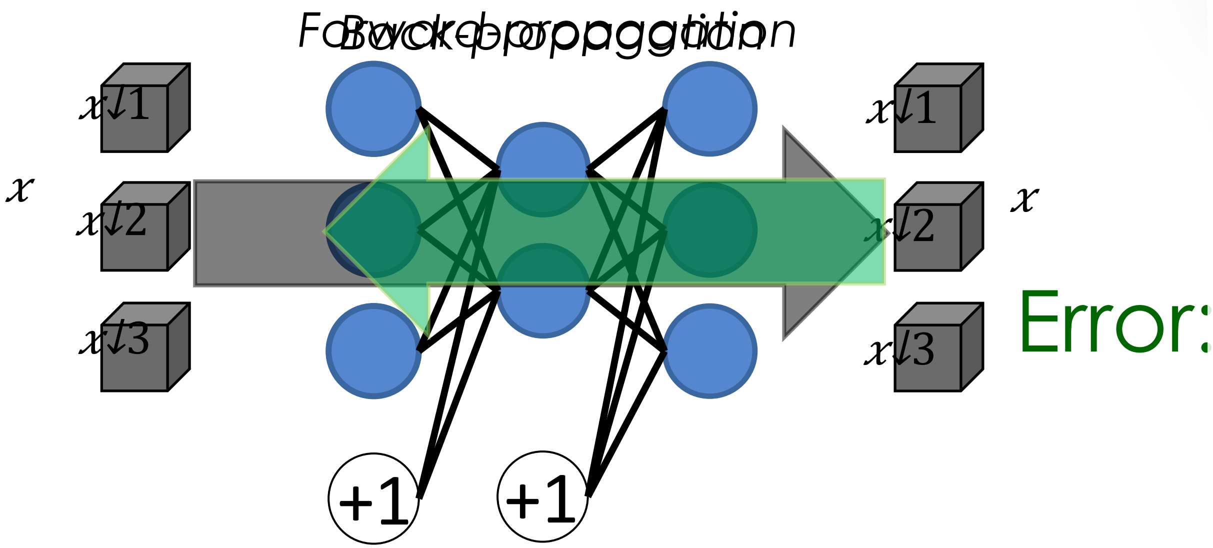
5 Types of Streams:
Potentially thousands of streams…
5 inc-stats each $\lambda=\{5,3,1,.1,.01\}$



The **KitNET** Anomaly Detector

Anomaly Detection with an Autoencoder

- An Autoencoder is a NN which is trained to reproduce its input after compression
- There are two phases: **Train** **Execute**

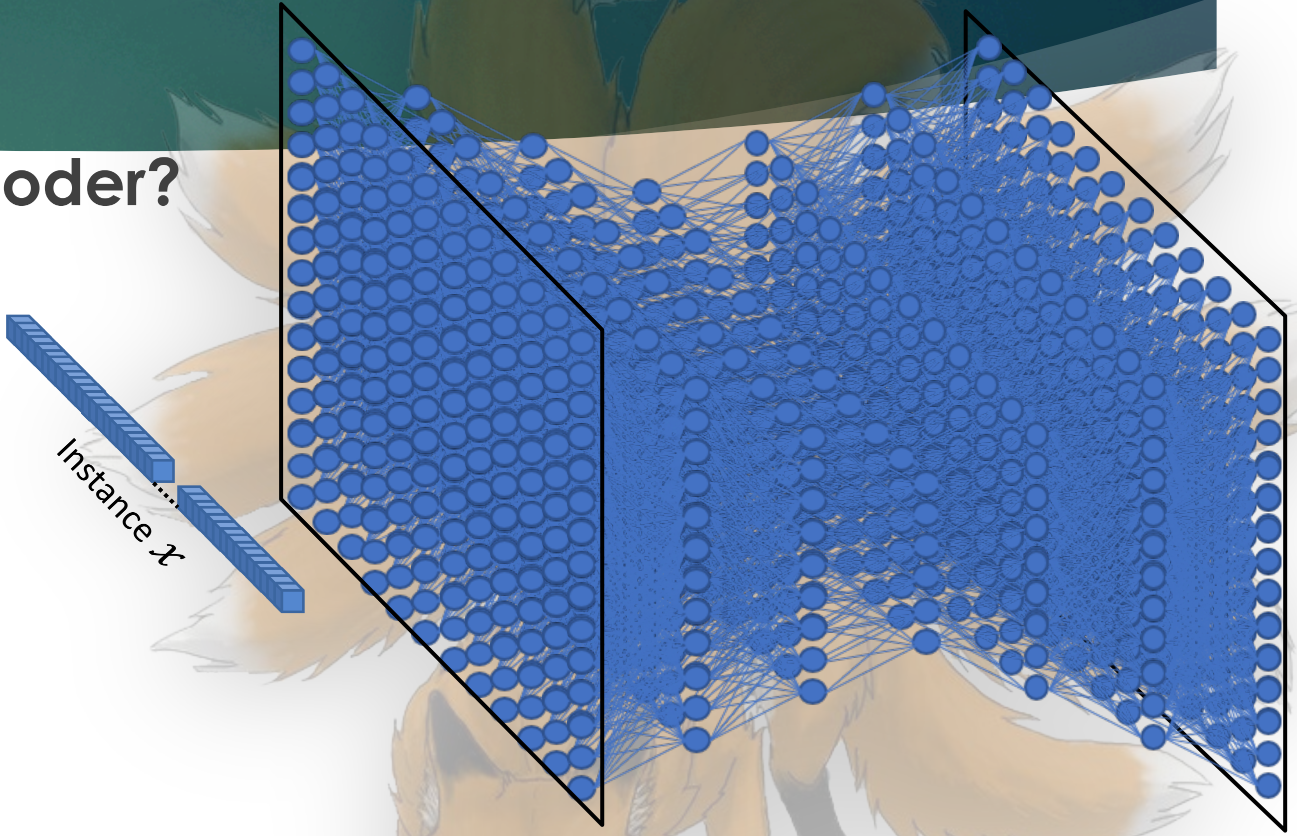


Reconstruction Error

$$\mathrm{RMSE}(\vec{x}, \vec{y}) = \sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n}}$$

Low value: x is normal
High value: x is abnormal
(does not fit known concepts)

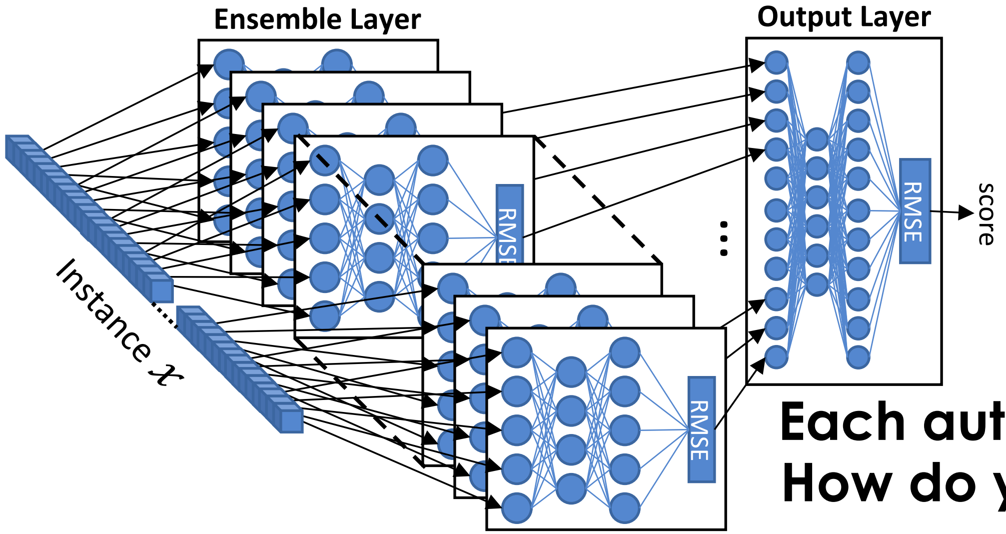
The **KitNET** Anomaly Detector



Why not one massive deep autoencoder?

- Curse of dimensionality!
- Train/Execute Complexity

Our Solution:

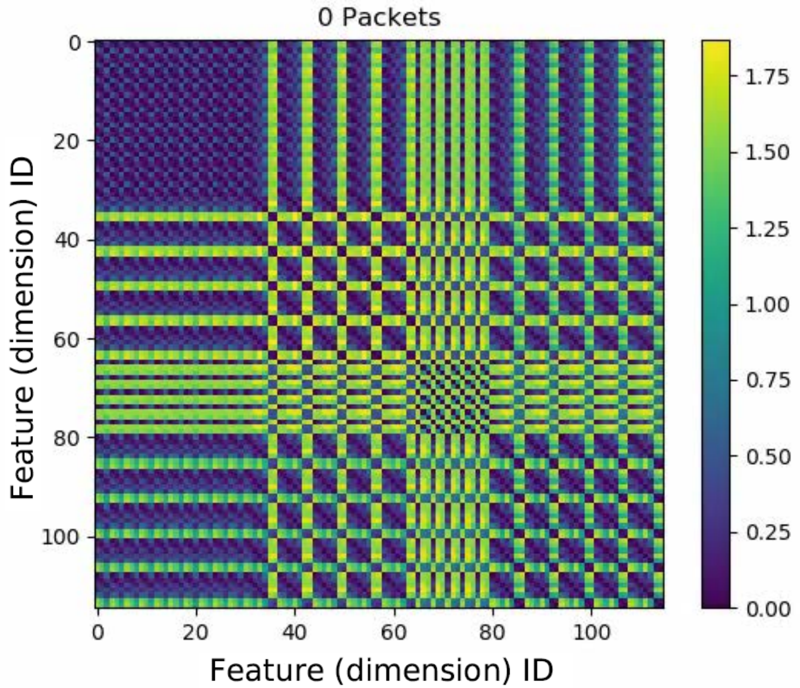


Each autoencoder receives a group of correlated features
How do you find the groupings online?

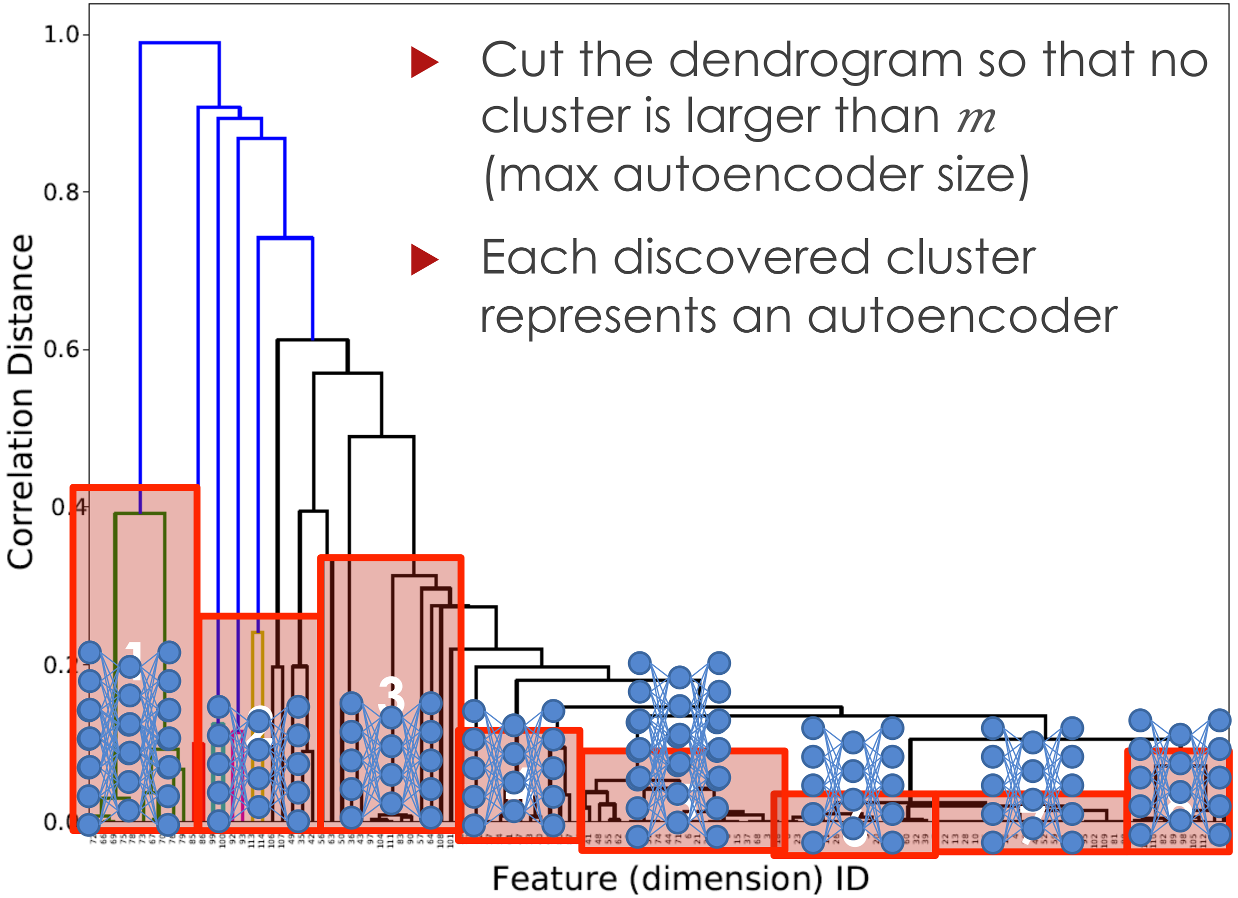
The **KitNET** Anomaly Detector

- For the first N observations (x), **incrementally** update a correlation distance matrix

$$D = [D_{ij}] = 1 - \frac{(x_i - \bar{x}_i) \cdot (x_j - \bar{x}_j)}{\|(x_i - \bar{x}_i)\|_2 \, \|(x_j - \bar{x}_j)\|_2}$$

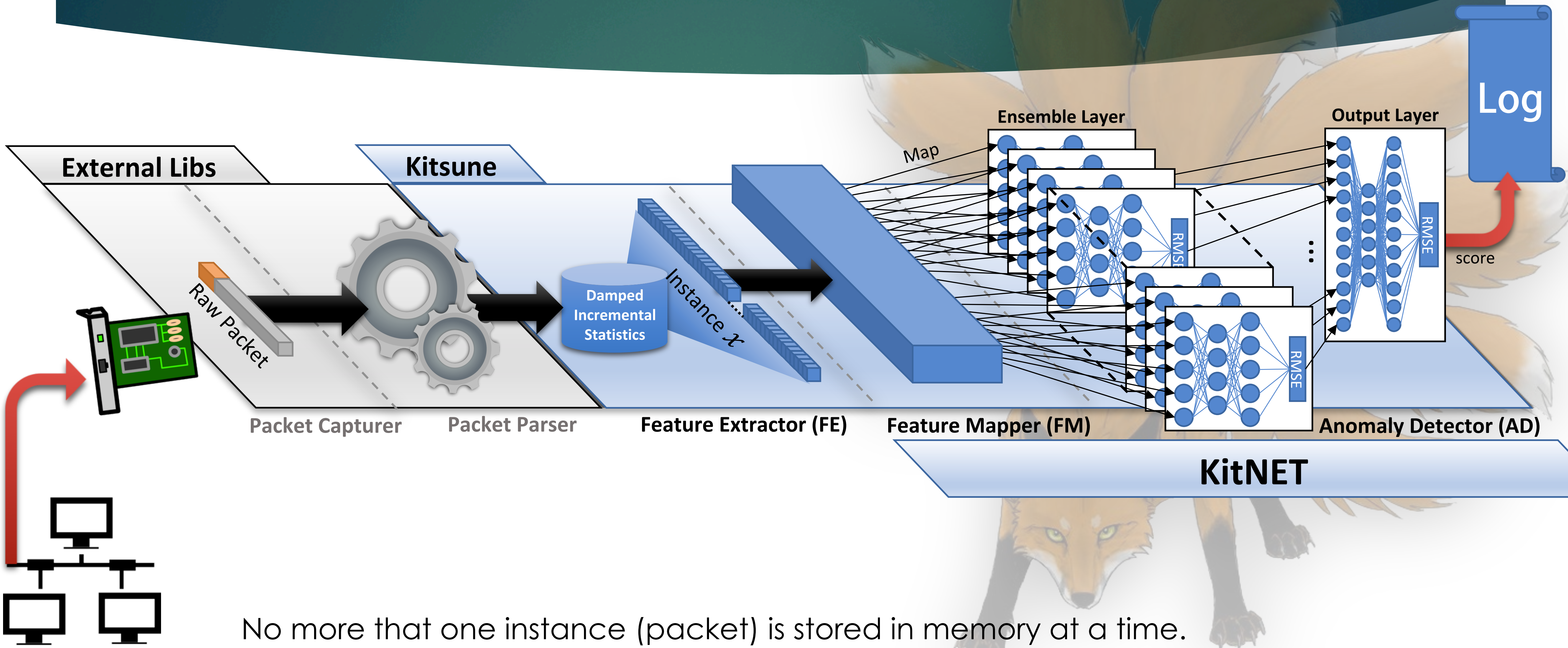


- Perform **one-time** agglomerative hierarchal clustering on D (fast)



- Cut the dendrogram so that no cluster is larger than m (max autoencoder size)
- Each discovered cluster represents an autoencoder

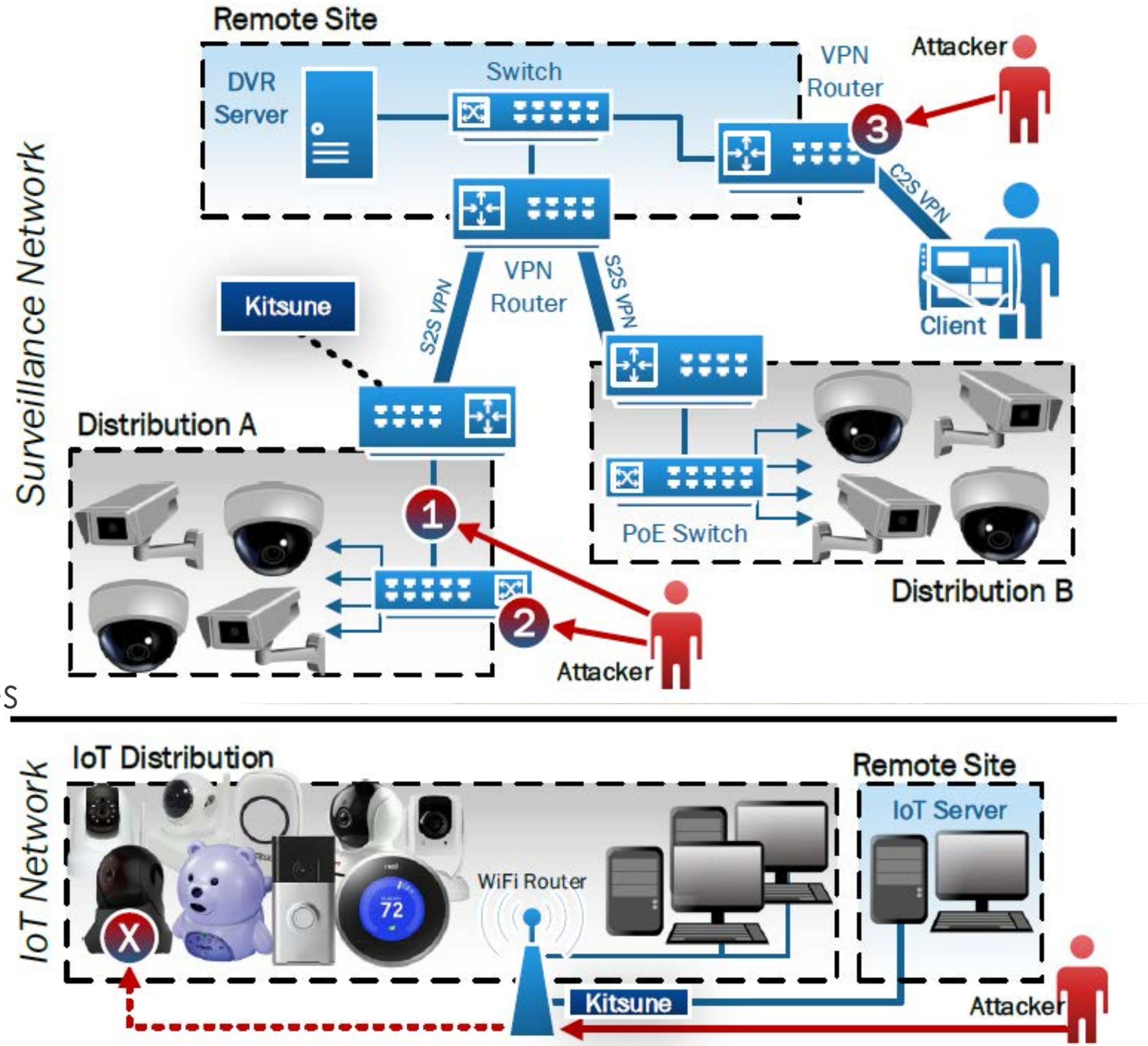
Kitsune NIDS



No more that one instance (packet) is stored in memory at a time.

Experimental Results

- **Networks**:
  - Surveillance
  - IoT
- **Algorithms**:
  - **Signature-based**: Suricata with over 13,465 emerging threat rules
  - **Anomaly-based**:
    - **Batch**: GMM, Isolation Forest
    - **Online**: pcStream & iGMM



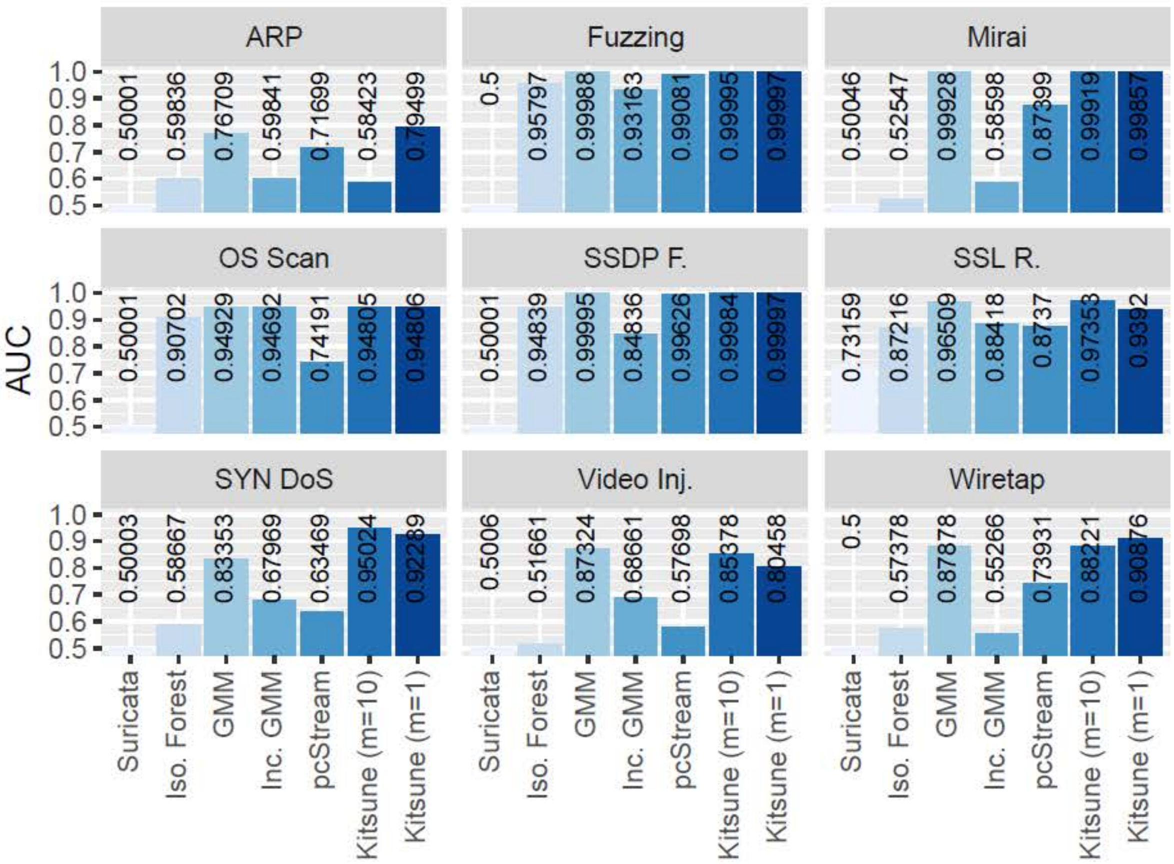
Experimental Results

Attacks

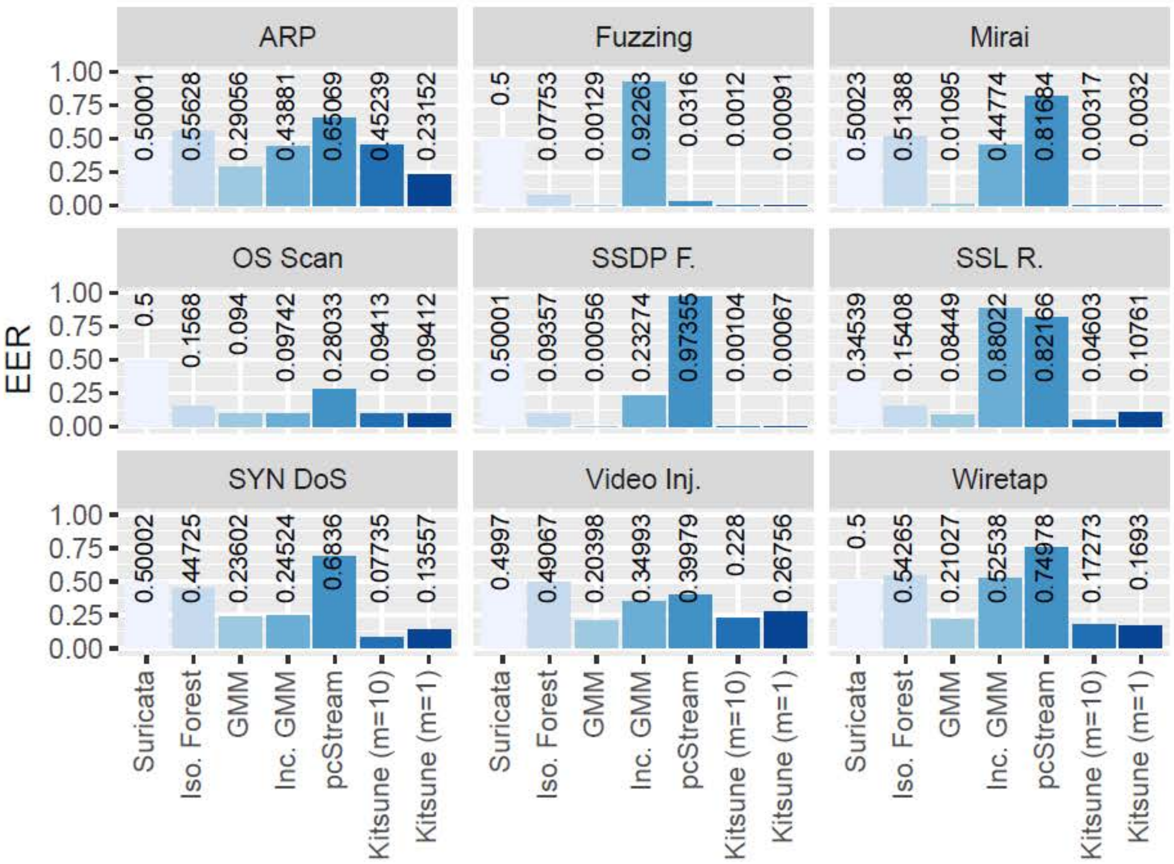
Attack Type	Attack Name	Tool	Description: *The attacker…*	Violation	Vector	# Packets	Train [min.]	Execute [min.]
Recon.	**OS Scan**	Nmap	*…scans the network for hosts, and their operating systems, to reveal possible vulnerabilities.*	**C**	1	1,697,851	33.3	18.9
	Fuzzing	SFuzz	*…searches for vulnerabilities in the camera's web servers by sending random commands to their cgis.*	**C**	3	2,244,139	33.3	52.2
Man in the Middle	**Video Injection**	Video Jack	*…injects a recorded video clip into a live video stream.*	**C, I**	1	2,472,401	14.2	19.2
	ARP MitM	Ettercap	*…intercepts all LAN traffic via an ARP poisoning attack.*	**C**	1	2,504,267	8.05	20.1
	Active Wiretap	Raspberry PI 3B	*…intercepts all LAN traffic via active wiretap (network bridge) covertly installed on an exposed cable.*	**C**	2	4,554,925	20.8	74.8
Denial of Service	**SSDP Flood**	Saddam	*…overloads the DVR by causing cameras to spam the server with UPnP advertisements.*	**A**	1	4,077,266	14.4	26.4
	SYN DoS	Hping3	*…disables a camera's video stream by overloading its web server.*	**A**	1	2,771,276	18.7	34.1
	SSL Renegotiation	THC	*…disables a camera's video stream by sending many SSL renegotiation packets to the camera.*	**A**	1	6,084,492	10.7	54.9
Botnet Malware	**Mirai**	Telnet	*…infects IoT with the Mirai malware by exploiting default credentials, and then scans for new vulnerable victims network.*	**C, I**	X	764,137	52.0	66.9

Experimental Results

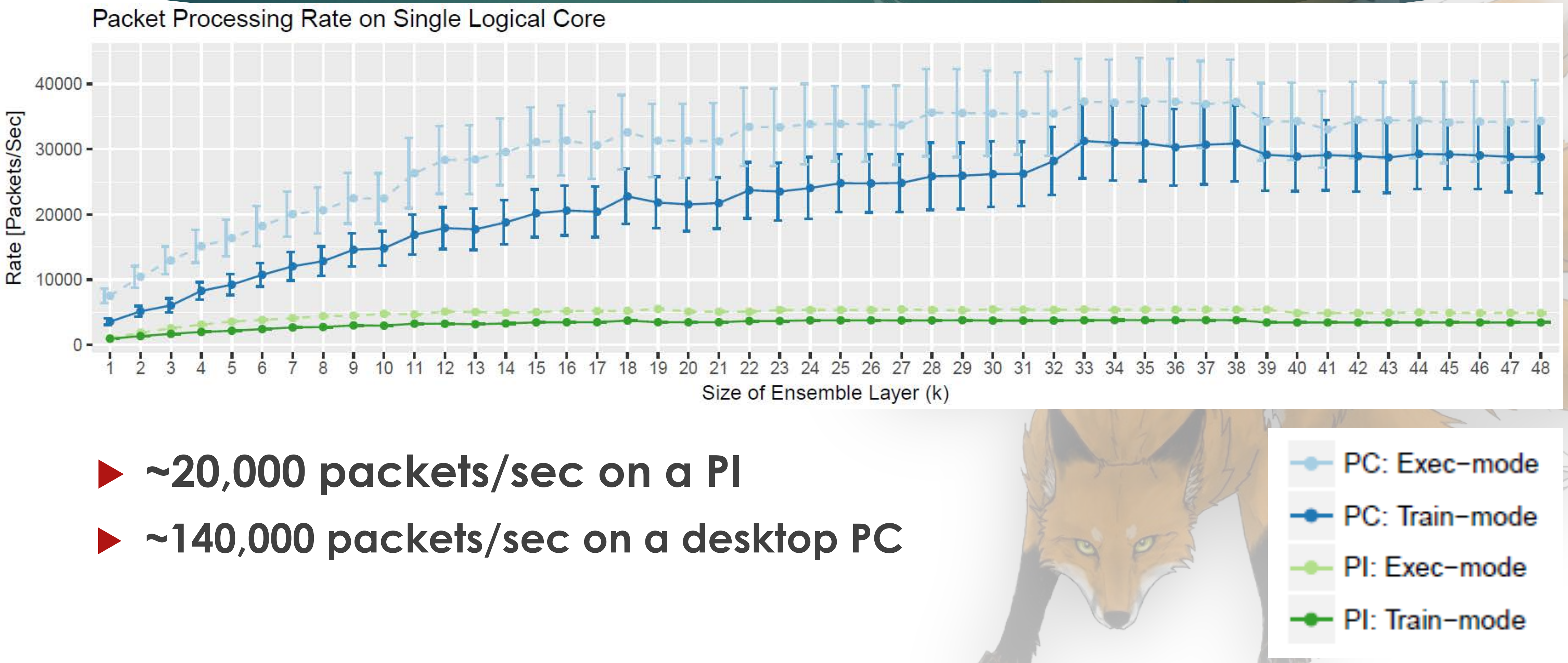
Area Under the Curve (AUC) *-Higher is better*



Equal Error Rate (EER) *-Lower is better*



Experimental Results



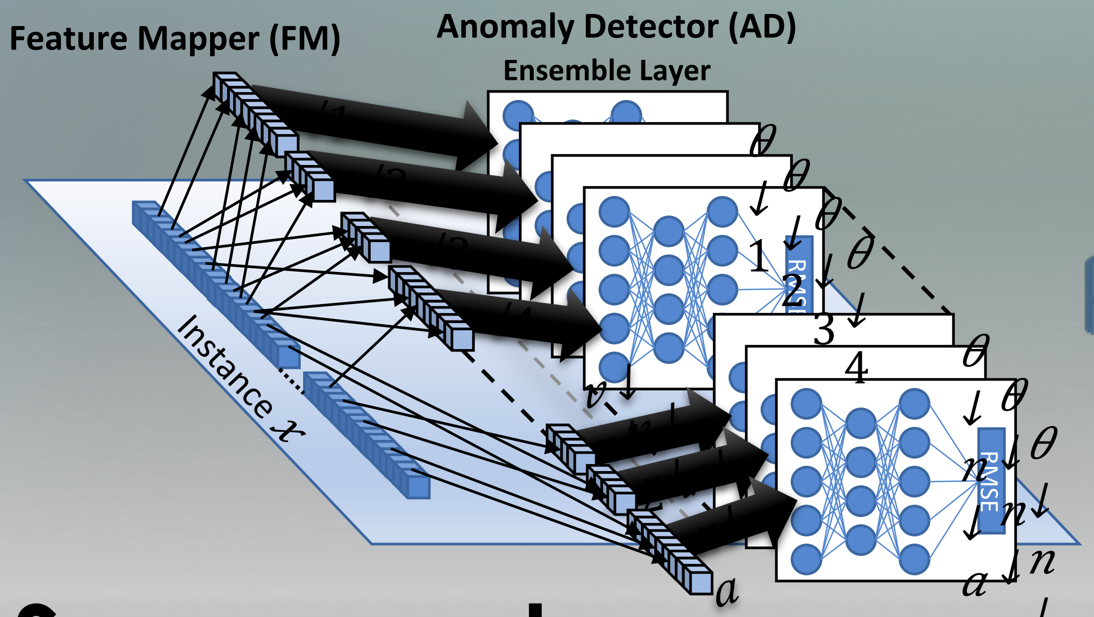
- ~20,000 packets/sec on a PI
- ~140,000 packets/sec on a desktop PC

Summary

- In the past, NNs on NIDS were used for the task of **classification**
- We propose using NNs for the task of **anomaly detection**
  - Eliminates the need for labeling data (endless traffic & unknown threats)
  - Enables plug-and-play
- **Kitsune Achieves this by,**
  - Efficient feature extraction
  - Efficient anomaly detection (**KitNET**)

KitNET

The core-anomaly detection algorithm of Kitsune



Source code:

https://github.com/ymirsky/KitNET-py

Cyber@Ben-Gurion
University of the Negev

Contact: *yisroel@post.bgu.ac.il*

Thank you!