Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection

Yisroel Mirsky, Tomer Doitshman, Y uval Elovici and Asaf Shabtai

NDSS 2018

Background

A common security system used to secure networks is a network intrusion detection system(NIDS). One popular approach is to use an artificial neural network (ANN) to perform the network traffic inspection.

1) Have an expert collect a dataset containing both normal traffic and network attacks.
2) Train the ANN to classify the difference between normal and attack traffic, using a strong CPU or GPU.
3) Transfer a copy of the trained model to the network/organization's NIDS.
4) Have the NIDS execute the trained model on the observed network traffic

A distributed deployment strategy+ NIDSs directly into inexpensive routers

Offline Processing **Supervised Learning** **High Complexity**

Kitsune Overview

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

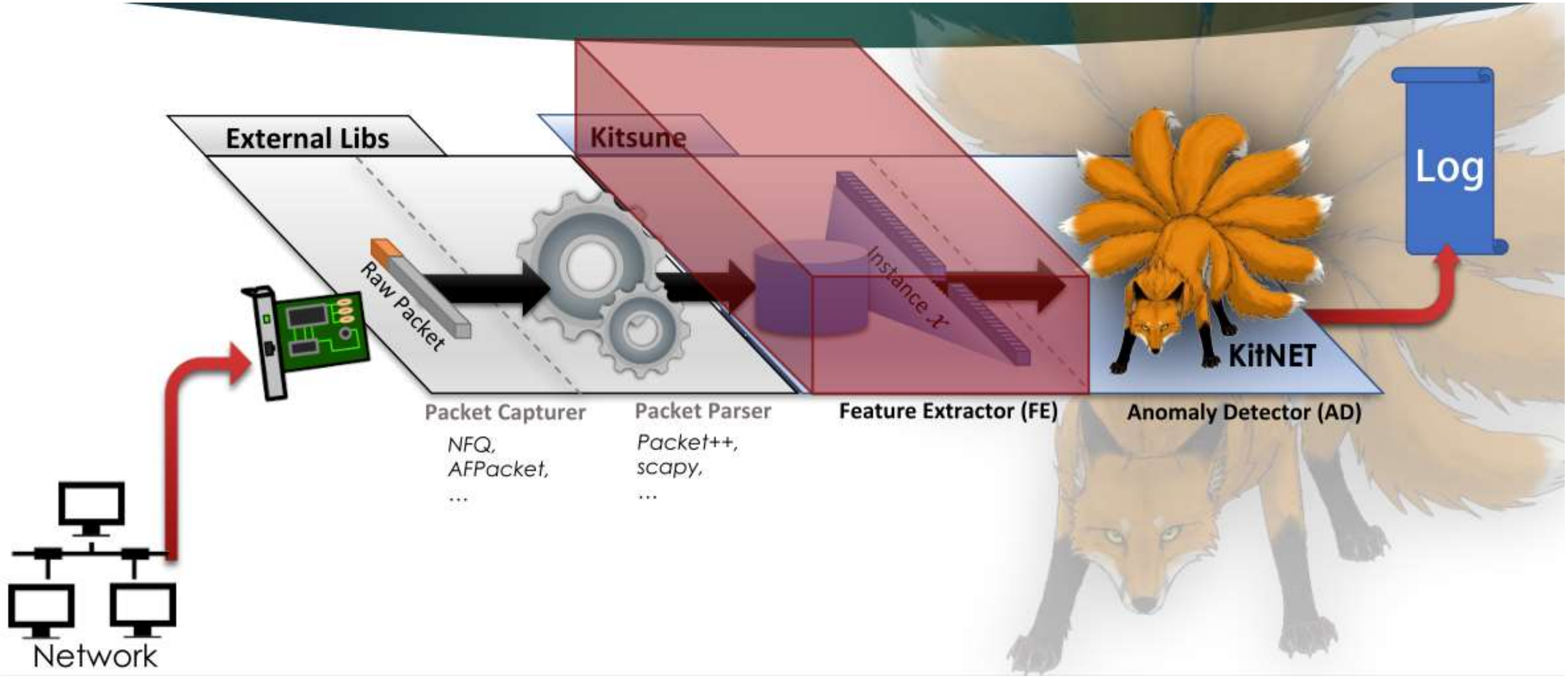
Enables NN on network traffic

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

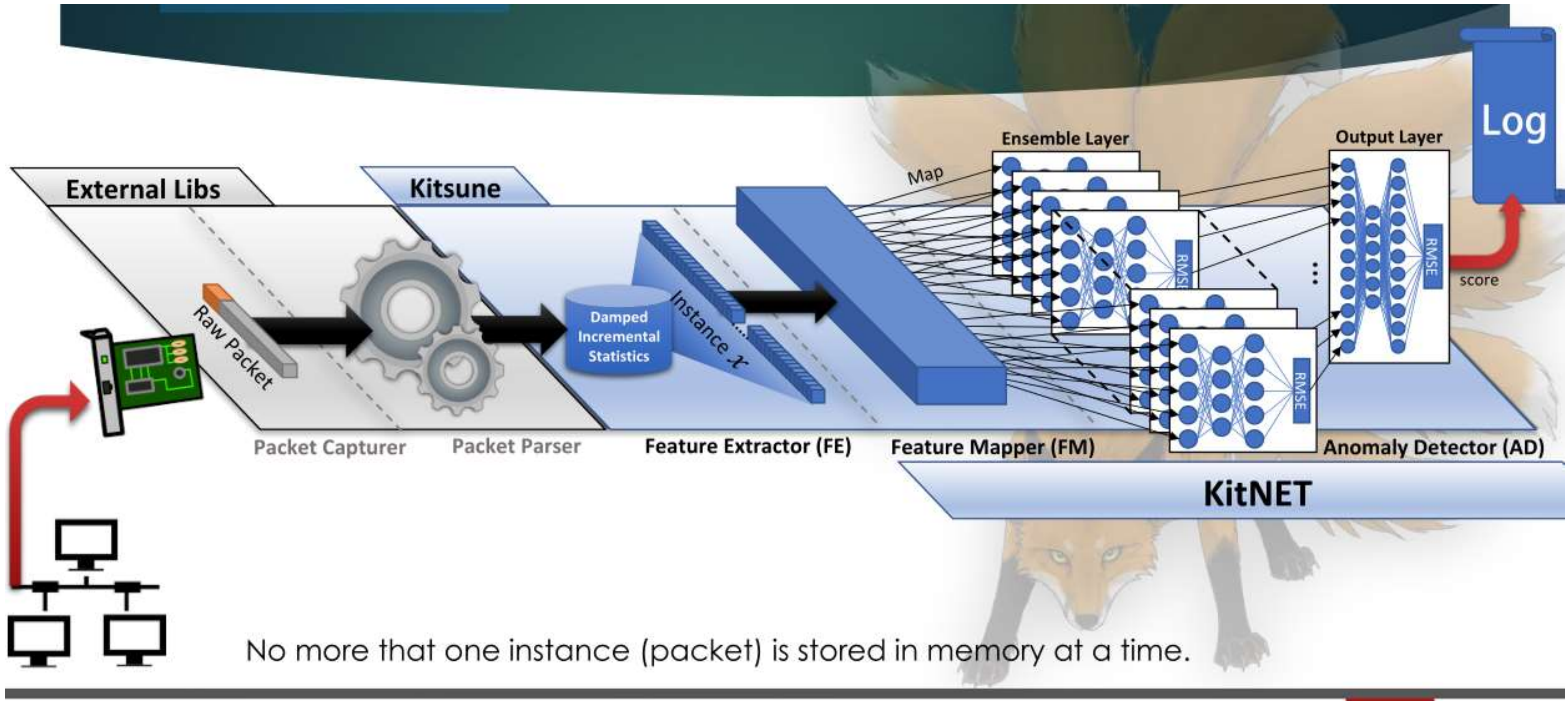
Enables realistic deployments

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

Kitsune Framework



Kitsune NIDS



Kitsune Feature Extractor (FE)

FE uses **damped incremental statistics** to efficiently

Algorithm 3: The algorithm for inserting a new value into a damped incremental statistic.

procedure: update($IS_{i,\lambda}$,x_{cur},t_{cur},r_j)

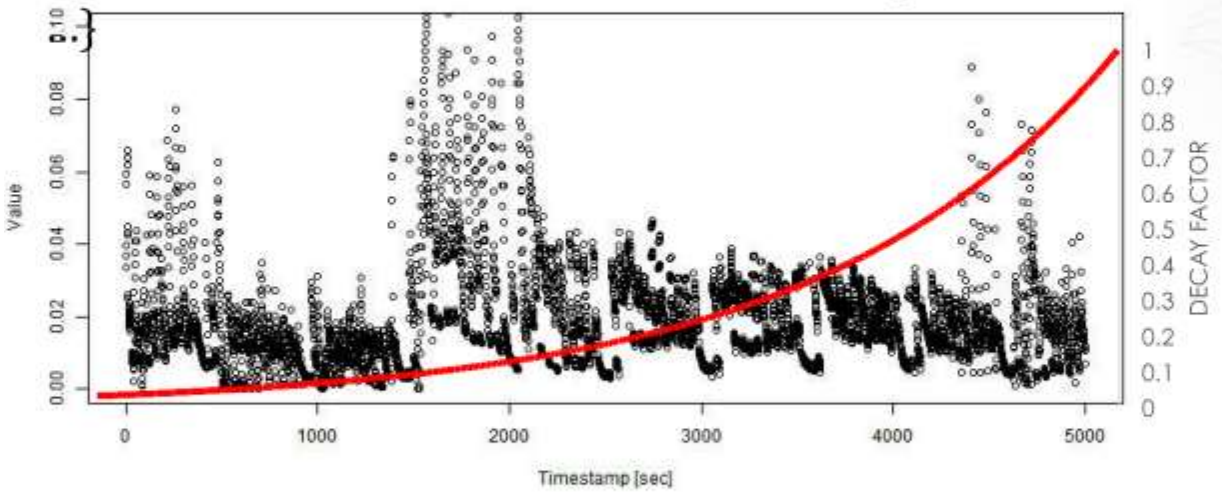
$\gamma \leftarrow d_\lambda(t_{cur} - t_{last})$ ▷ Compute decay factor

$IS_{i,\lambda} \leftarrow (\gamma w, \gamma LS, \gamma SS, \gamma SR, T_{cur})$ ▷ Process decay

$IS_{i,\lambda} \leftarrow (w+1, LS+x_{cur}, SS+x_i^2, SR_{ij}+r_i r_j, T_{cur})$ ▷ Insert value

return $IS_{i,\lambda}$

An unbounded stream of values $S=\{x\downarrow 1, x\downarrow 2,$



Objective: Compute the stats *(μ, δ,...)* over the recent history of *S*, given limited memory and non-uniform sample rates (timestamps)

tuple ***IS := (N, LS, SS)***

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}	$\sqrt{(\sigma_{S_i}^2)^2+(\sigma_{S_j}^2)^2}$
	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)

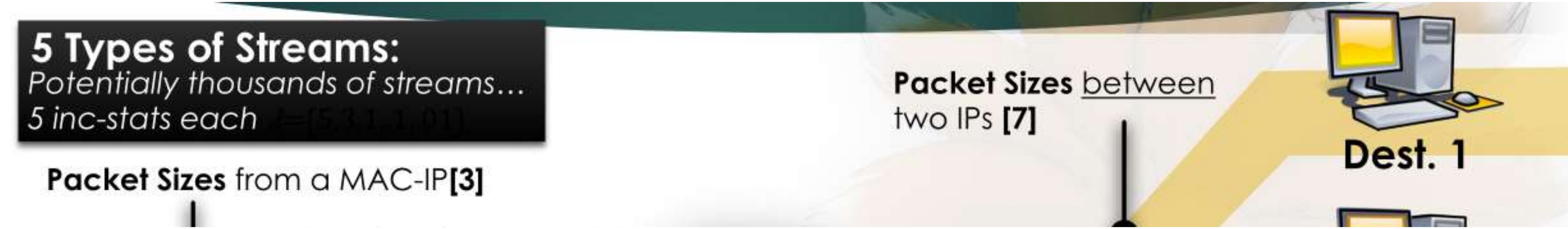
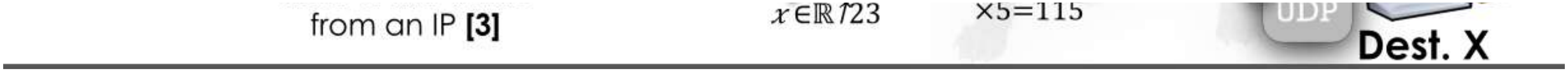
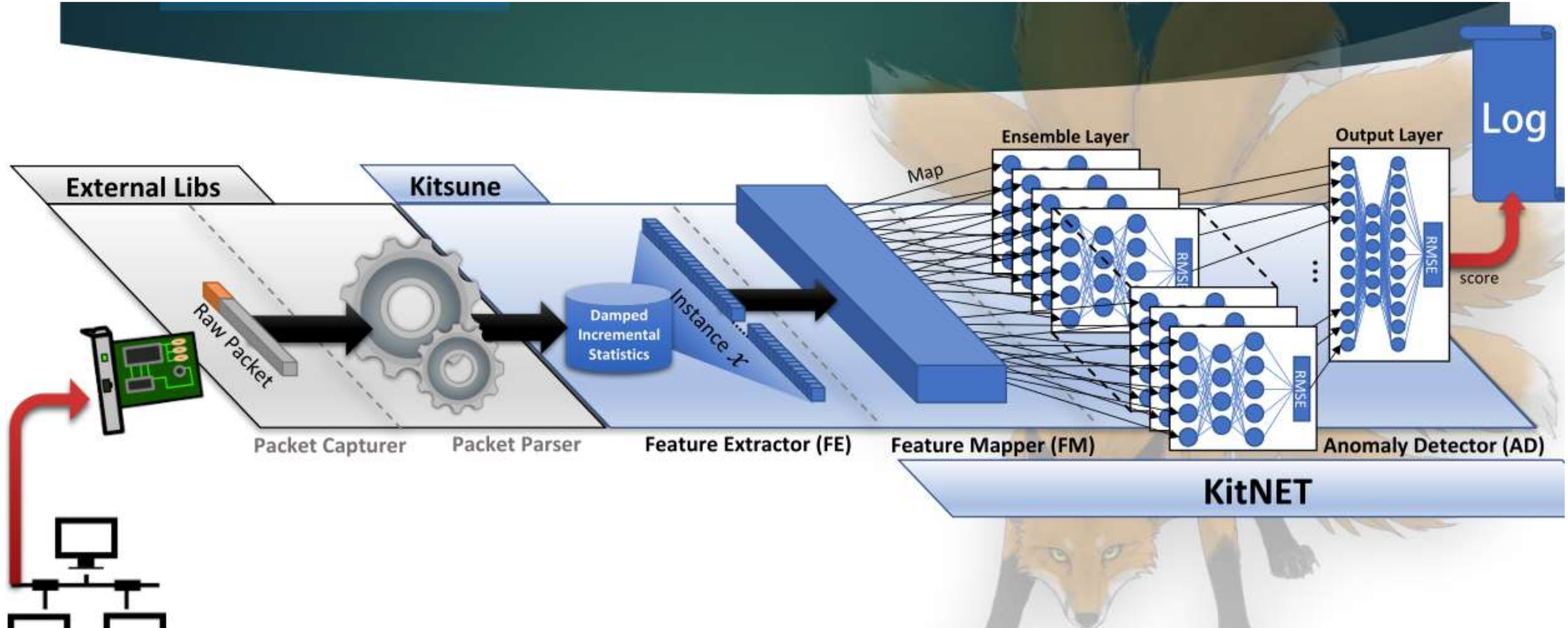


TABLE II: The statistics (features) extracted from each time window λ when a packet arrives.

The packet's…	Statistics	Aggregated by	# Features	Description of the Statistics
…size	μ_i, σ_i	SrcMAC-IP, SrcIP, Channel, Socket	8	*Bandwidth of the outbound traffic*
…size	$\lVert S_i, S_j \rVert, R_{S_i,S_j}, Cov_{S_i,S_j}, P_{S_i,S_j}$	Channel, Socket	8	*Bandwidth of the outbound and inbound traffic together*
…count	w_i	SrcMAC-IP, SrcIP, Channel, Socket	4	*Packet rate of the outbound traffic*
…jitter	w_i, μ_i, σ_i	Channel	3	*Inter-packet delays of the outbound traffic*



Kitsune NIDS

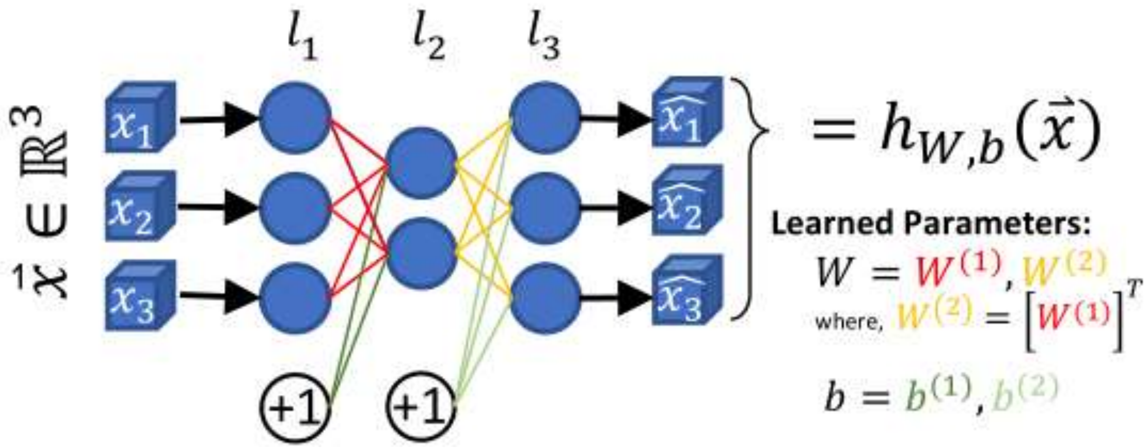


No more that one instance (packet) is stored in memory at a time.

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



$$h_\theta(\vec{x}) \approx \vec{x}$$

Reconstruction Error

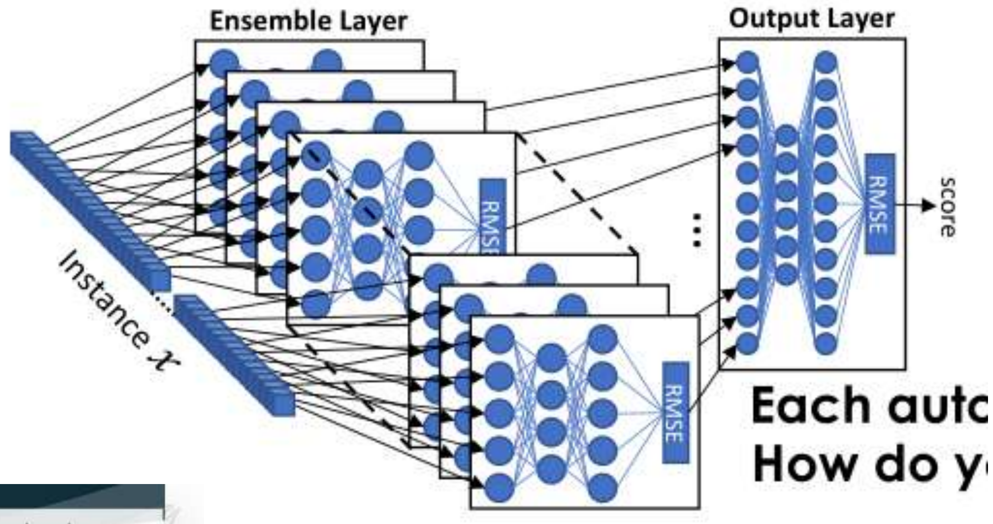
$$\mathrm{RMSE}\left(\vec{x}, \vec{y}\right) = \sqrt{\frac{\sum_{i=1}^{n}\left(x_i - y_i\right)^2}{n}}$$

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

1) **Training Phase:** Train an autoencoder on clean (normal) data. For each instance x_i in the training set X:
   a) Execute: $s = \mathrm{RMSE}\left(\vec{x}, h_\theta(\vec{x})\right)$
   b) Update: if$(s \geq \phi)$ then $\phi \leftarrow s$
   c) Train: Update θ by learning from x_i
2) **Execution Phase:**
   When an unseen instance $\vec{x}$ arrives:
   a) Execute: $s = \mathrm{RMSE}\left(\vec{x}, h_\theta(\vec{x})\right)$
   b) Verdict: if$(s \geq \phi\beta)$ then *Alert*

The **KitNET** Anomaly Detector

Our Solution:



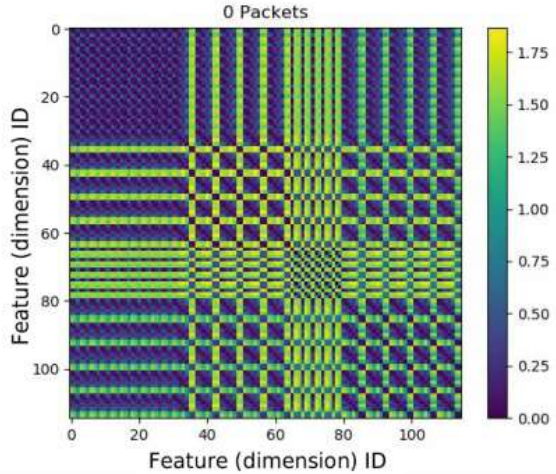
Each auto
How do y

Why not one massive deep autoencoder?

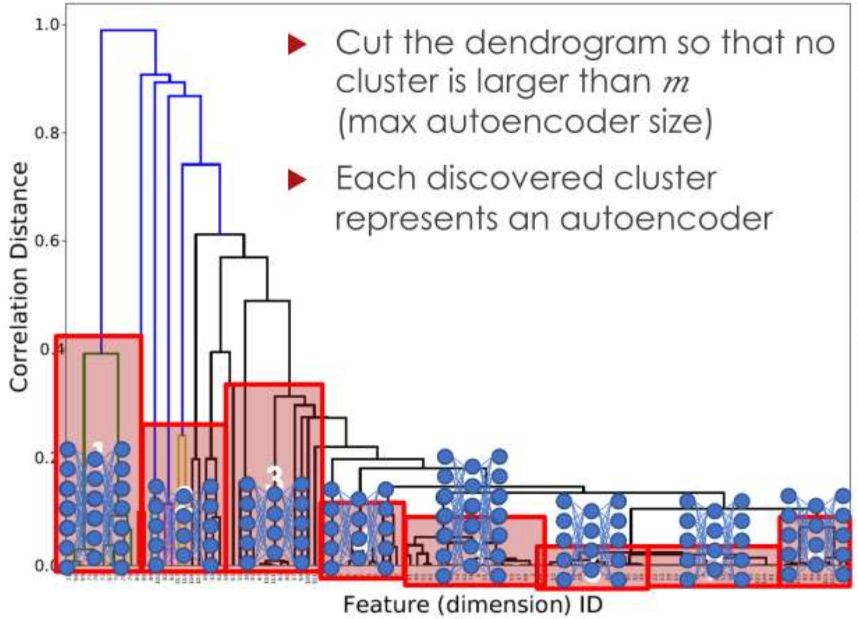
- ➢ Curse of dimensionality!
- ➢ Train/Execute Complexity

$$d_{cor}(u,v) = 1 - \frac{(u-\bar{u})\cdot(v-\bar{v})}{\|(u-\bar{u})\|_2\|(v-\bar{v})\|_2}$$

- For the first N observations (x), **incrementally** update a correlation distance matrix
  $D=[D_{ij}]=1-(x_i-x_i)\cdot(x_j-x_j)/\|(x_i-x_i)\|_2\|(x_j-x_j)\|_2$



- 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

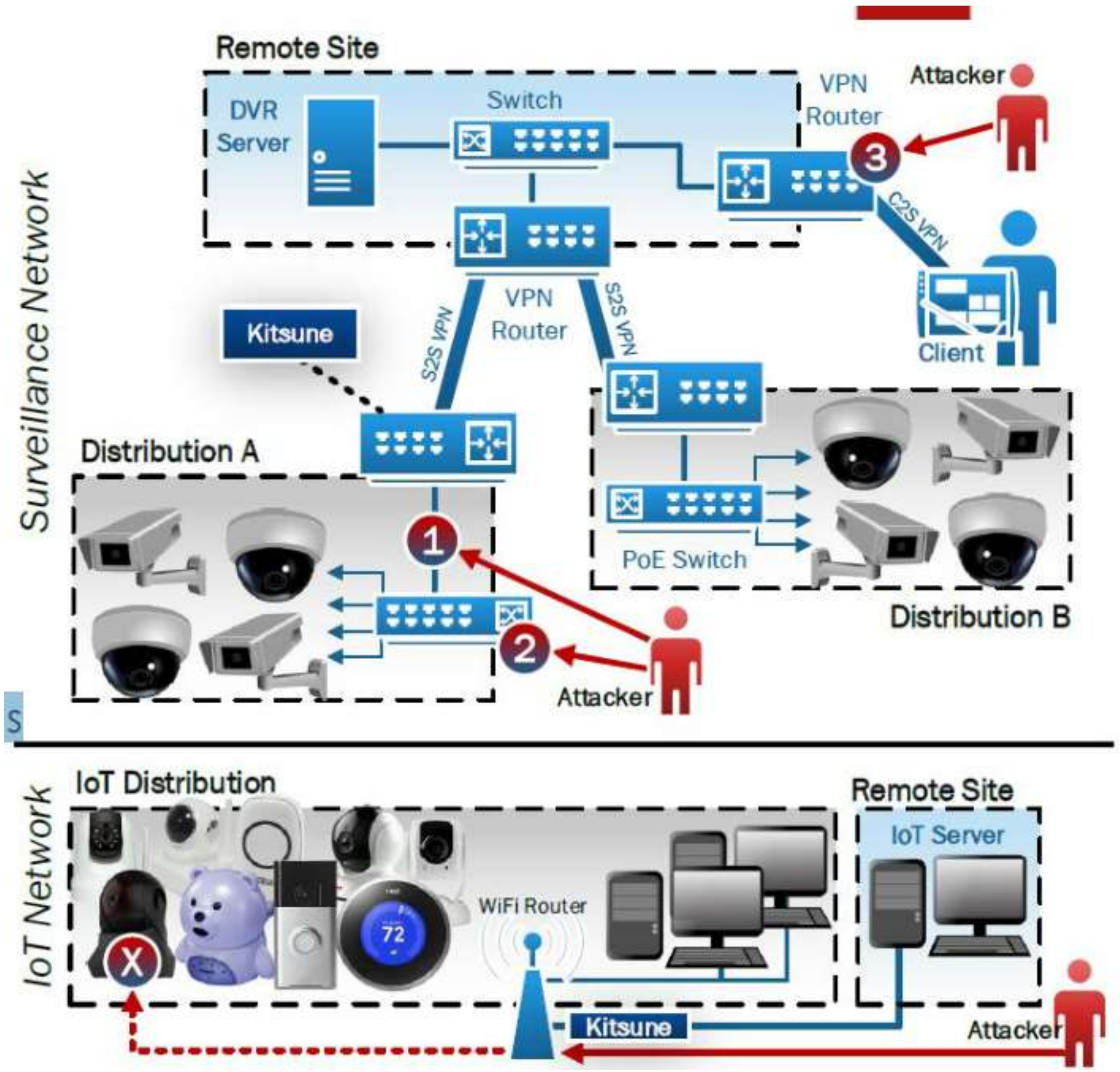
Experimental Results

Networks:

- Surveillance
- IoT

Algorithms:

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

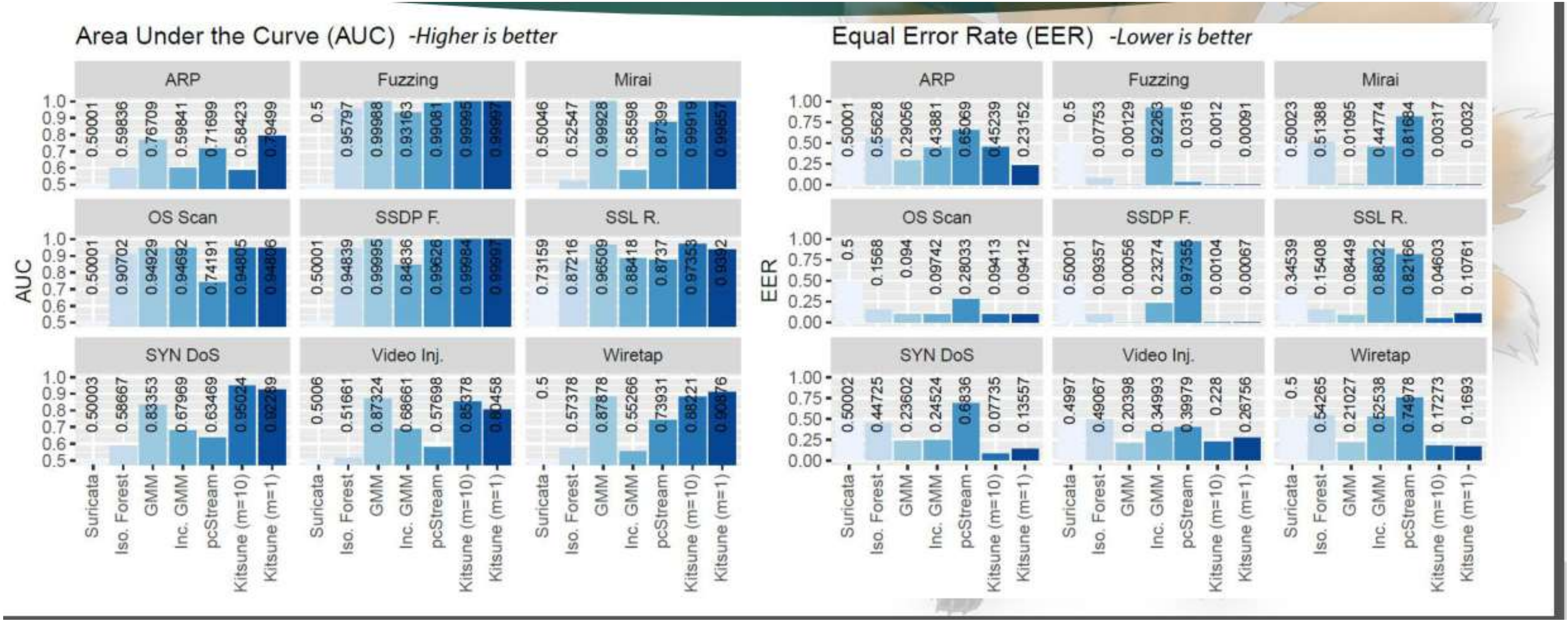


Experimental Results

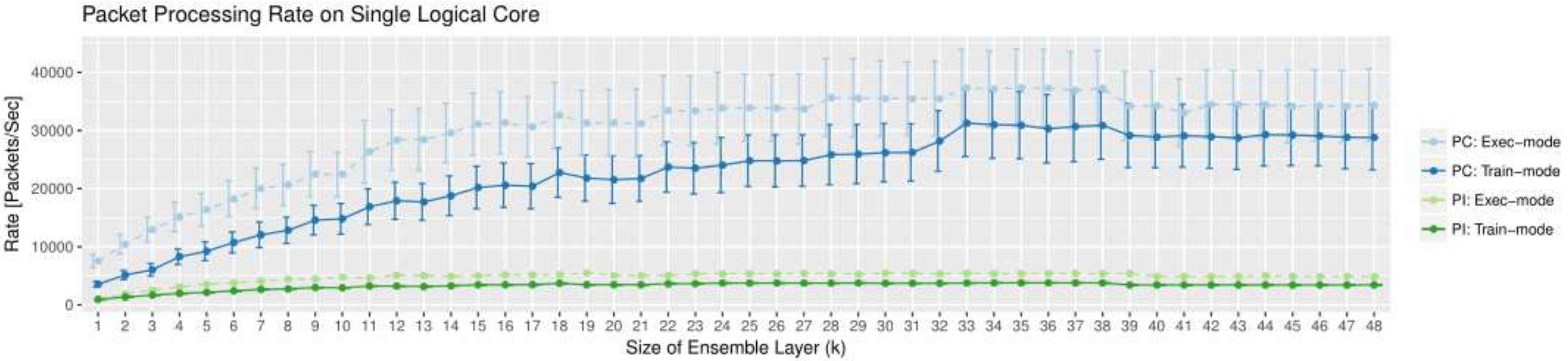
TABLE III: The datasets used to evaluate **Kitsune**.

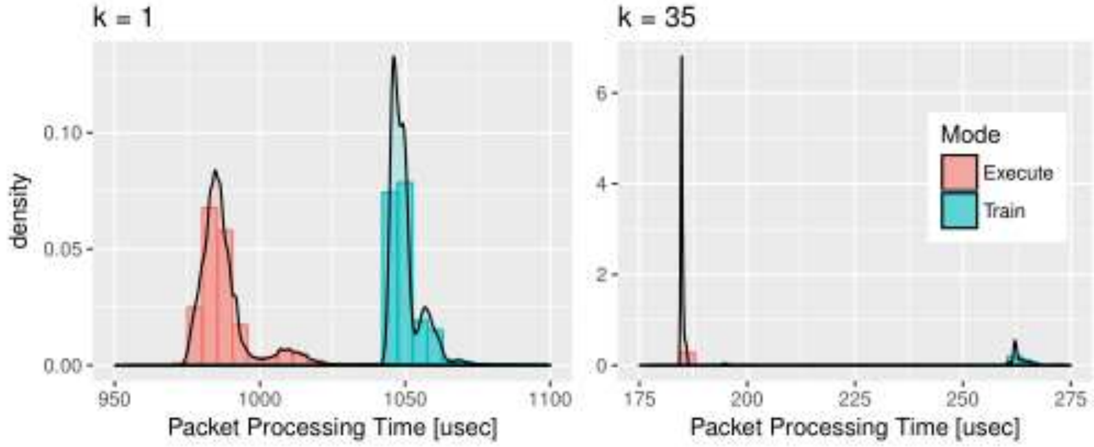
Attack Type	Attack Name	Tool	Description: *The attacker...*	Violation	Vector	# Packets	Train [min.]	Execute [min.]
Recon.	**OS Scan**	Nmap	*...scans the network for hosts, and their operating systems, to reveal possible vulnerabilities.*	**C**	1	1,697,851	33.3	18.9
	Fuzzing	SFuzz	*...searches for vulnerabilities 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



Experimental Results





Conclusion

- 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)

adversarial machine learning