AI4SEC

Enhancing ~~Cybersecurity~~ Network Security through AI/ML

Dr. Pedro Casas

Senior Scientist

Data Science & Artificial Intelligence

AIT Austrian Institute of Technology @Vienna

Forum Numerica Seminar

28.09.2021

We are *Losing the Battle* against Cybercrime

- Cybercrime *as a country* → 3rd largest economy behind US and China in 2021 (***Cybercrime could cost 10.5 trillion USD by 2025***)

- Operational complexity and diversity – ***cyberattack surface growth outpacing humans' ability to secure it***

- ***Need*** for (more) ***automated approaches*** with ***less human intervention*** to improve cyber defenses

AI/ML to the Resecue

Extensive research harnessing the capabilities of AI/ML to improve security solutions

but, the success and *speed of adoption of AI/ML* in the cybersecurity practice is rather slow…

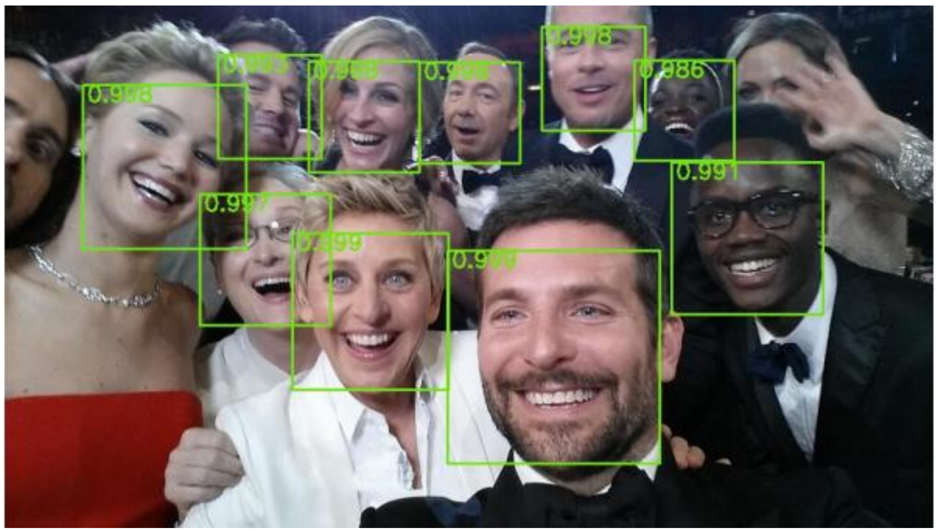
…especially as compared to the success of AI/ML in *natural* domains (NLP, image processing, etc)

What is *Limiting* AI/ML Success in Networking?

What is Blocking AI Success in Networking?

- ***Data Complexity:*** the complexity (and heterogeneity) of the data related to Internet-like networks is one of the most significant bottlenecks to AI4NETS

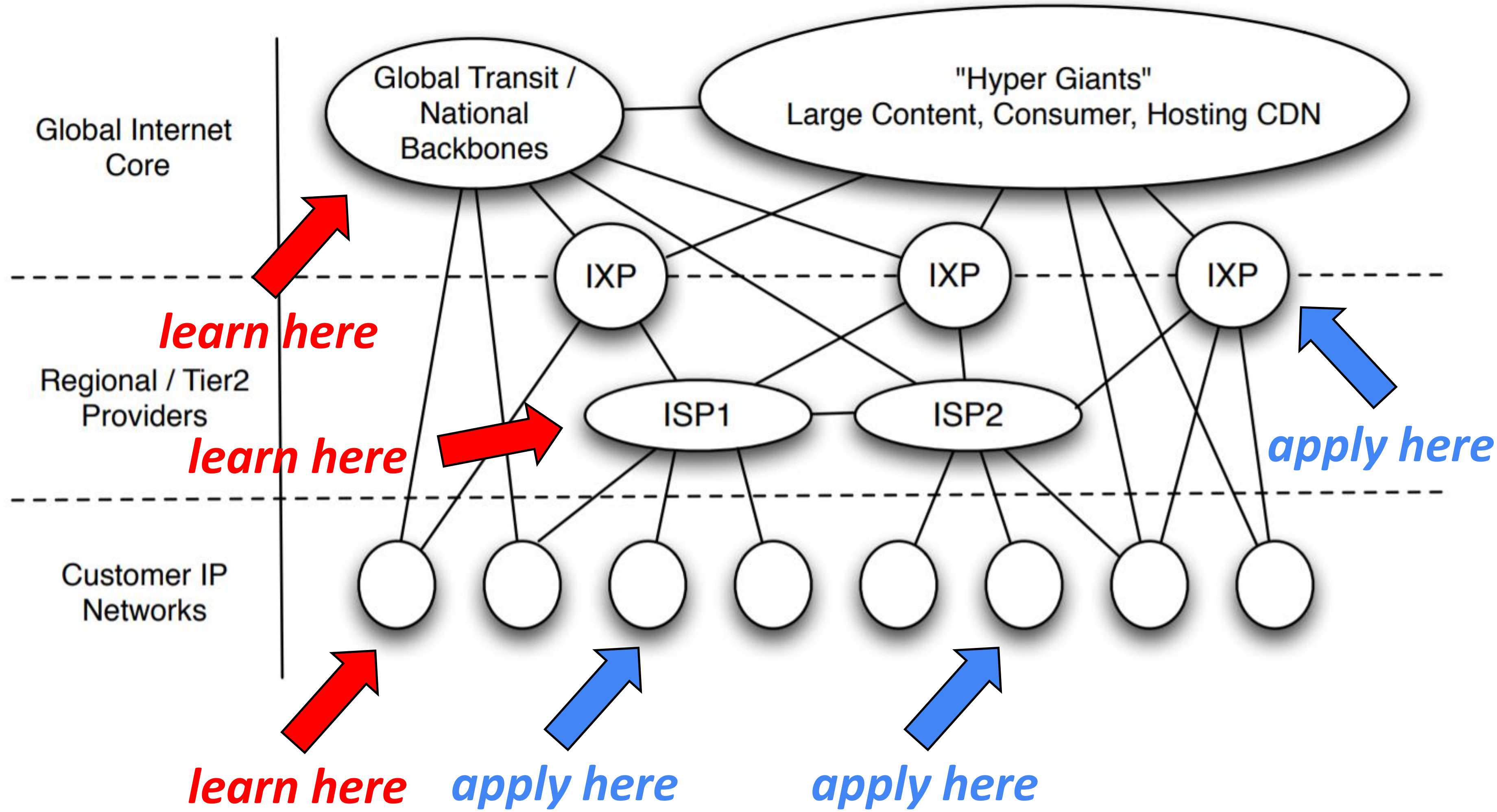
- The Internet, and in general large-scale networks, are a **complex tangle** of networks, technologies, applications, services, devices and end-users



- **AI has so far shown very successful results** generally in **data from** more **predictable** and **easy to understand sources** (natural sources)

What is Blocking AI Success in Networking?

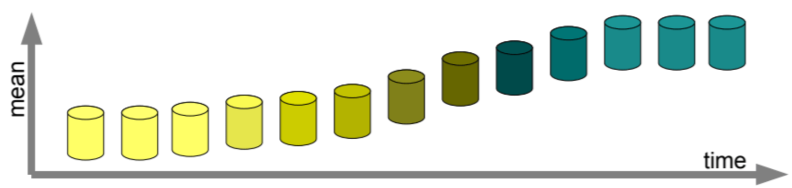
- ***Diversity of Network Data:*** besides complexity, network data often exhibits much more diversity than one would intuitively expect

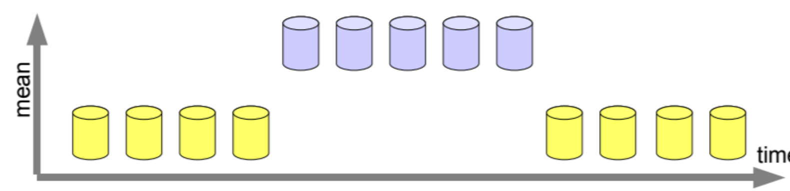


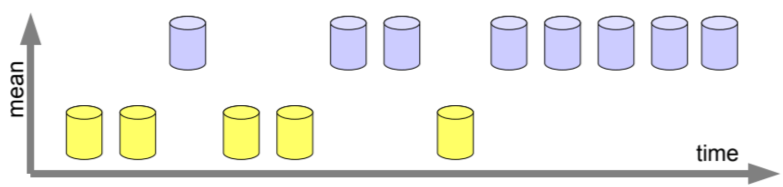
What is Blocking AI Success in Networking?

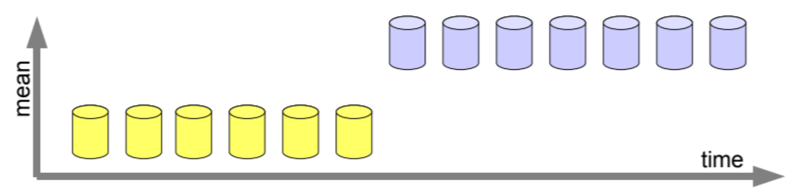
- ***Data Dynamics:*** networking data is non-stationary, generally comes in the form of data streams, and is full of constant concept drifts

time









What is Blocking AI Success in Networking?

- ***Lack of Ground Truth:*** in the wild networking data is usually non-labeled
- ***Lack of Standardized and Representative Datasets***: datasets are generally biased, difficult to find appropriate public datasets to assess AI4NETS

- There is **no IMAGENET** or the like in Networking
- **Network data labeling**, and even **data interpretation**, is **too complex** for humans, even for **domain experts** (e.g., malware vs benign traffic instead of cat vs dog)
  - Easier for naturally generated data: images, text, audio

What is Blocking AI Success in Networking?

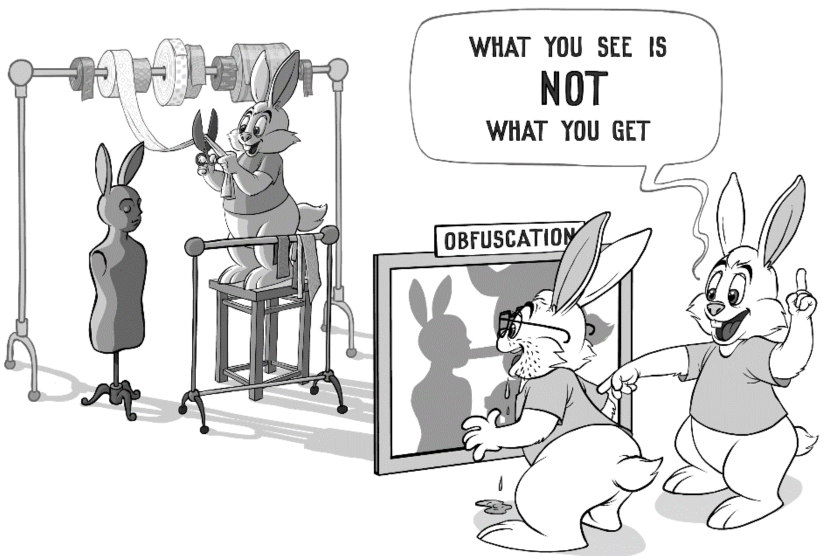
- ***Lack of Interpretability:*** this is a general problem of ML models (e.g., DL provides *beautiful* black-boxes)…but the issue is even more complex in AI4NETS

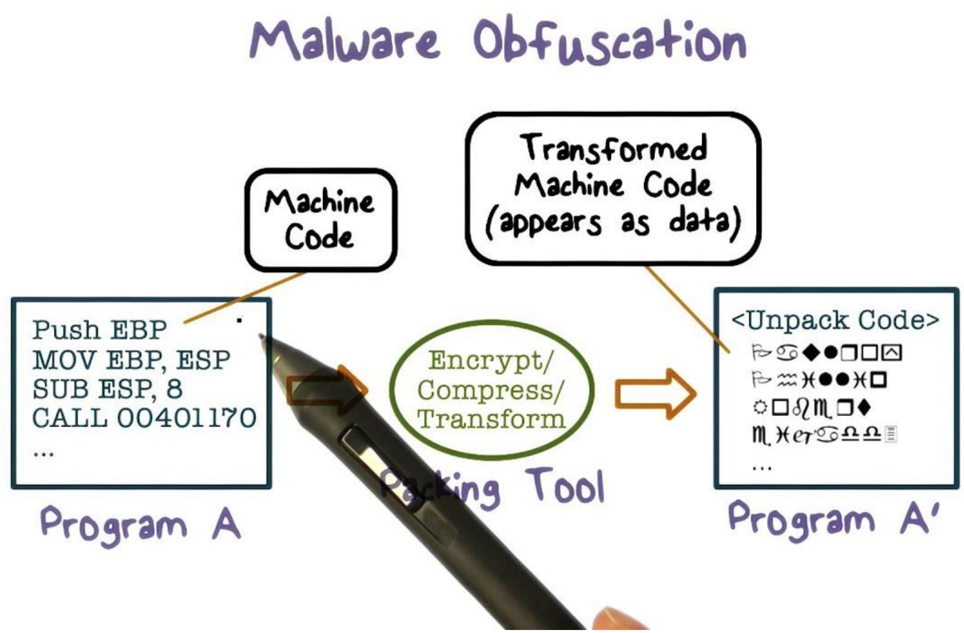
- To **improve trust**, the end-user (**humans**) has to **trust model predictions**, for example, by **understanding which inputs lead to a specific output**, but generally difficult to interpret networking features

- The **lack of interpretability and trust stops AI deployments**:
    - **Network security – AI4SEC**
    - Dynamic Traffic Engineering – AI4NETTE
    - Dynamic network instantiation (NFV) and (re)-configuration (SDN) – AI4SELFNET

What is Blocking AI Success in Networking?

- **AI for cybersecurity** is a ***double-edged sword***: a security solution or a ***weapon used by attackers*** *(needs much more research)*
- ***Learning occurs in an Adversarial Setting:*** services obfuscate and modify their functioning to bypass monitoring and avoid traffic engineering policies





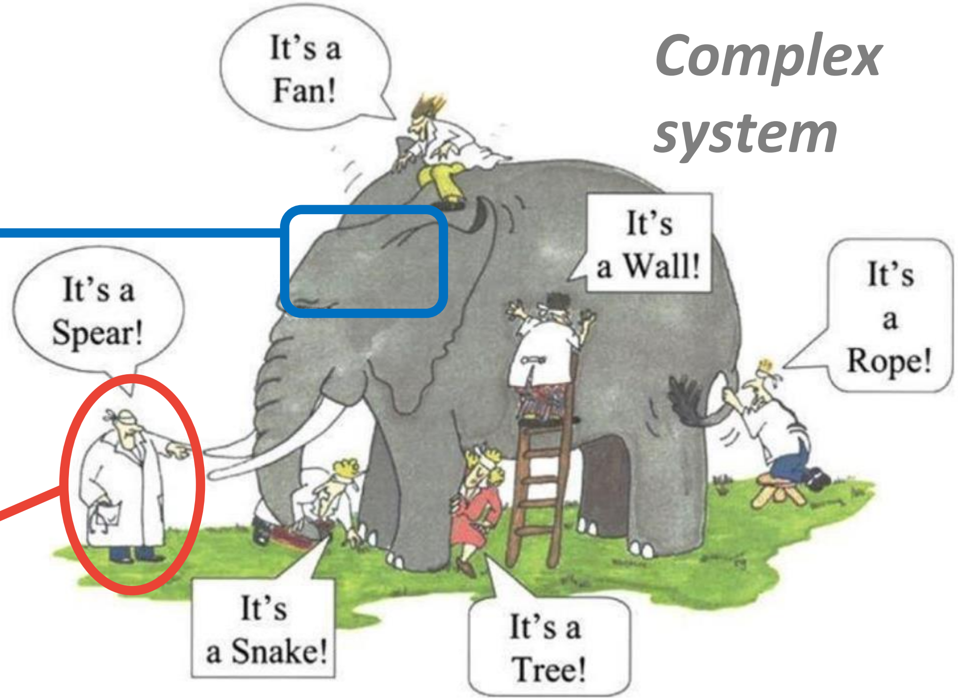
- It becomes even **more trickier to learn**, when the ***adversary*** constantly tries to fool the learner
- **Not only malign actors**, but standard *services*: Skype, QUIC, etc.

What is Blocking AI Success in Networking?

- ***Robust Learning:*** lack of formal guarantees (***formal methods***), especially in safety-critical contexts (cybersecurity)
- ***Data and Model bias:***

data is biased: partial or misrepresentation of real system

models are biased: assumptions or hypotheses of behavior, mathematical properties, lack of transparency

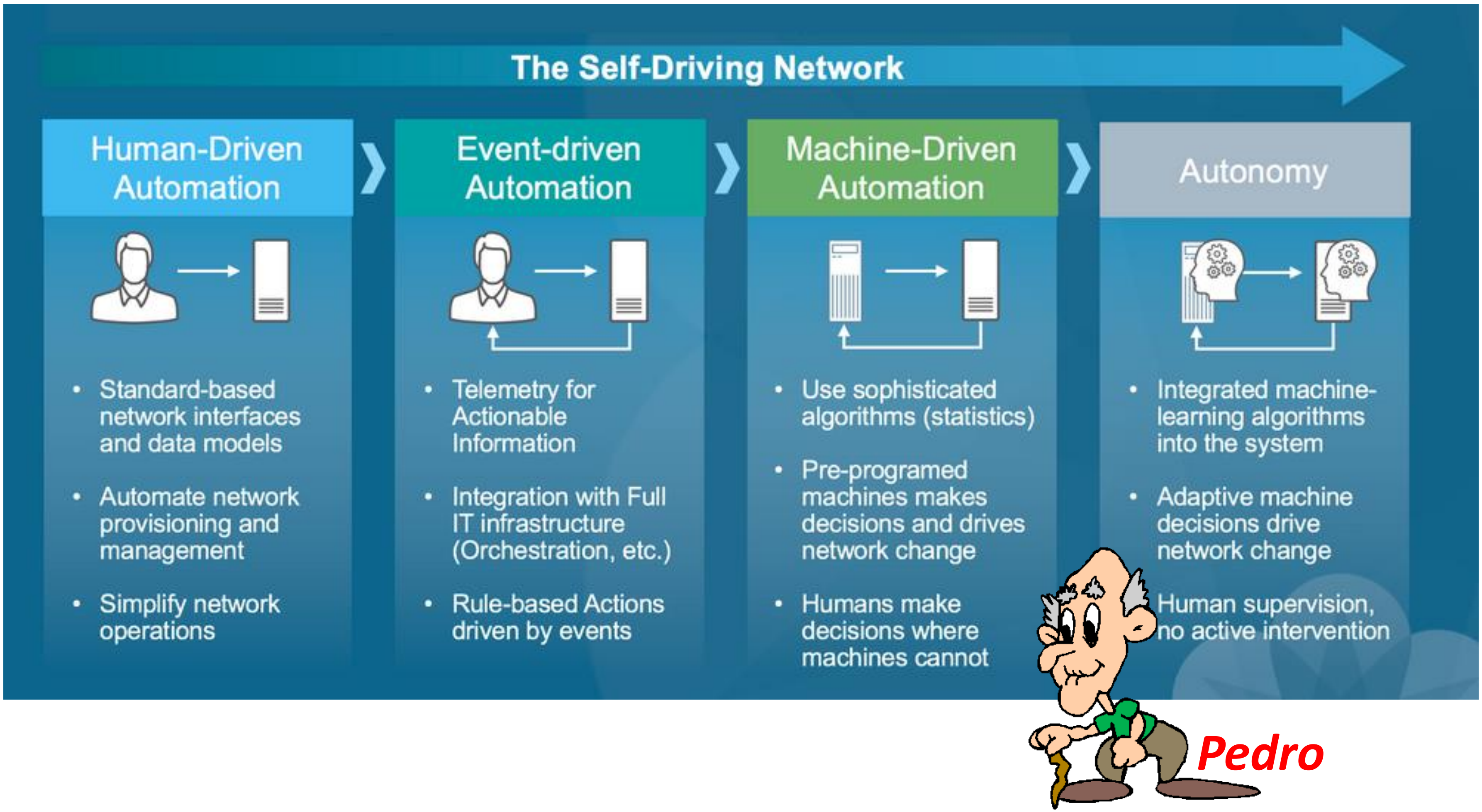


Complex system

- The AI/ML model ***user is biased***, or unaware of the limitations of AI/ML: model ***evaluation/testing***, model ***certification***, ***correlation vs causality***

What is Blocking AI Success in Networking?

- ***Lack of Learning Generalization:*** as a consequence of previous issues, **it becomes extremely difficult in the networking practice to learn models which can generalize to operational environments**



Pedro

Organization of the Talk

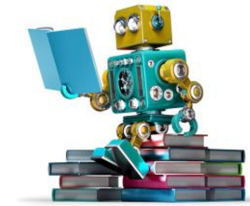
Dealing with Some of these Challenges

- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- Generative Models for Anomaly Detection – ***Avoid Traffic Modeling***
- Explainable Artificial Intelligence (XAI) **–** ***Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

Organization of the Talk
Dealing with Some of these Challenges

- **Deep Learning for Malware Detection** – ***Avoid Feature Engineering***
- Generative Models for Anomaly Detection – ***Avoid Traffic Modeling***
- Explainable Artificial Intelligence (XAI) **–** ***Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

Artificial Intelligence – As Smart as a Donut!



- **Machine Learning** is still **very unintelligent** – the **big revolution** is on **big data processing** and **data availability**/accessibility

- **Current ML benefits** are fundamentally due to machines ability to **blindly**:
  - compute lots of math operations per second
  - handle large amounts of data
  - deal with data in high-dimensional spaces

- **A lot of data required** to “learn“ simple logical inter-relations

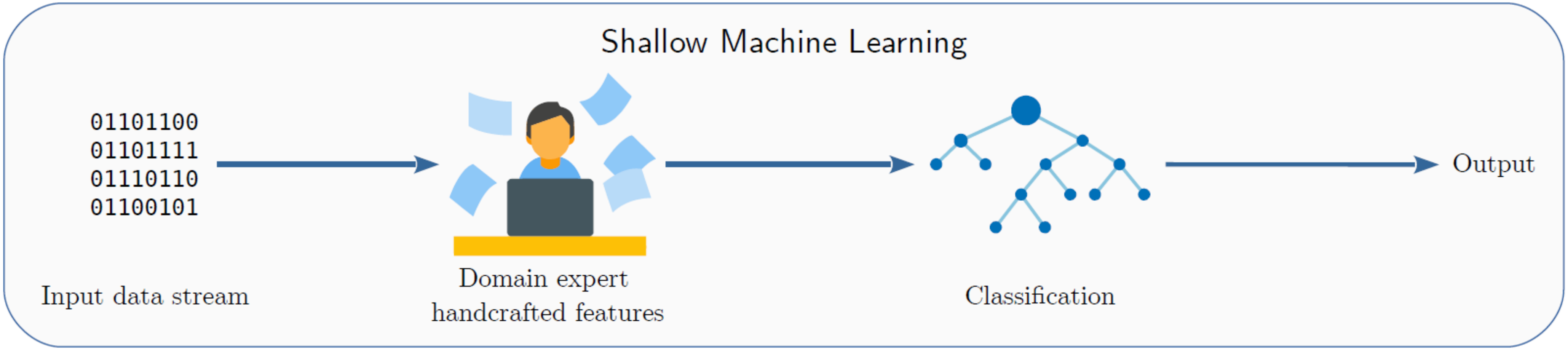
- **Shallow Learning**: less data but ***human expert knowledge required***, to properly guide the ***feature engineering*** process

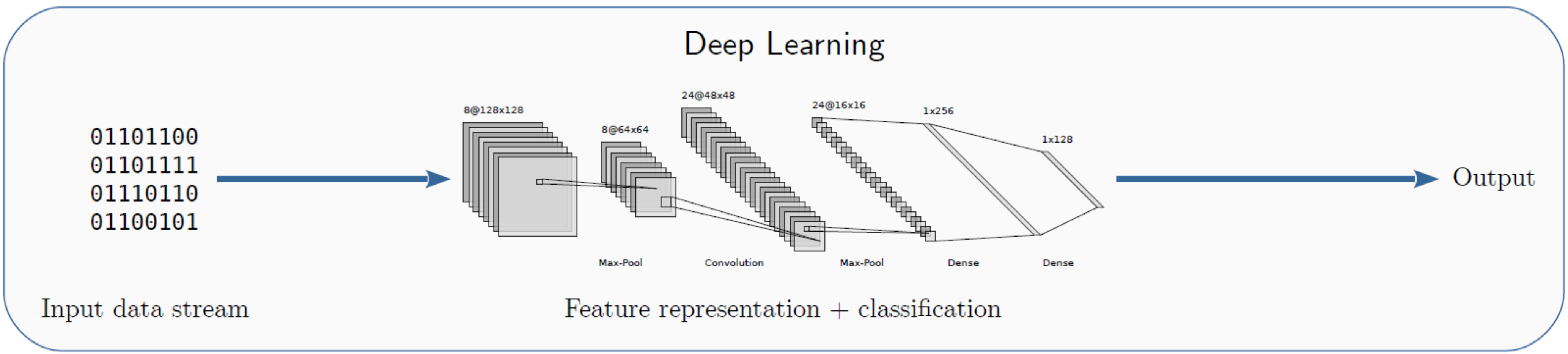
- **Deep Learning**: ***automated feature engineering*** (representation learning) but ***needs much more data***

RawPower

we explore **deep learning** for ***blind*** malware detection in network traffic

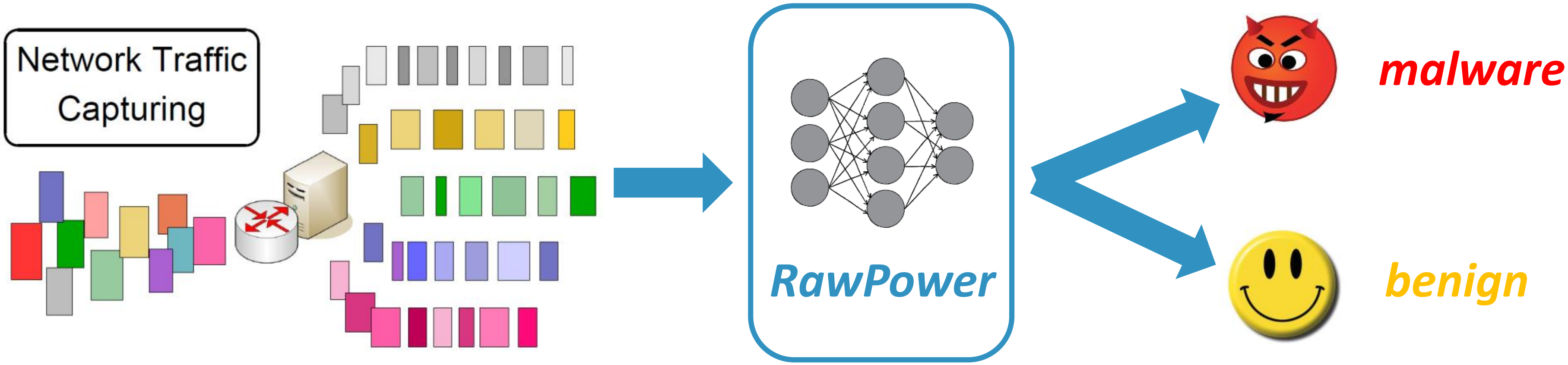
Shallow Learning vs Deep Learning





Basic Concepts of RawPower

- The ***input to the Deep Learning*** model is **RAW** – ***only byte-streams***
- **No need to define** tailored, domain-knowledge-based **input features**



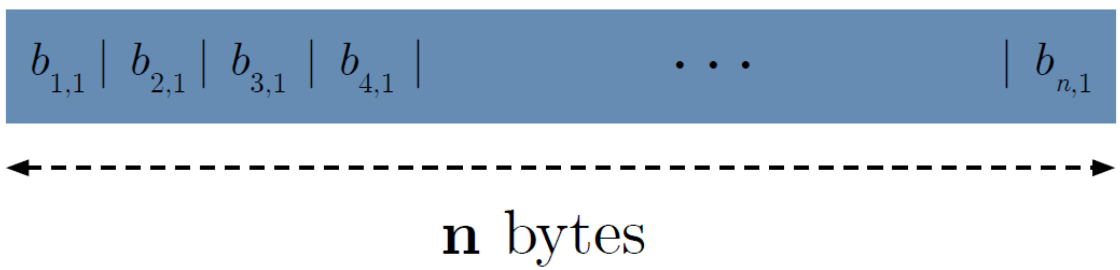
- Different architectures to **analyze both packet-based** and **flow-based byte aggregations**
- Models for **binary malware detection** – fully **supervised-based training**

Raw Input Representations

- **Input representation of the data**, as well as **network architecture**, are both **key elements** to consider when **building a DL model**

- We take **two types of raw input representations:** ***packets*** and ***flows***. Decimal normalized representation of every byte of every packet is a different input

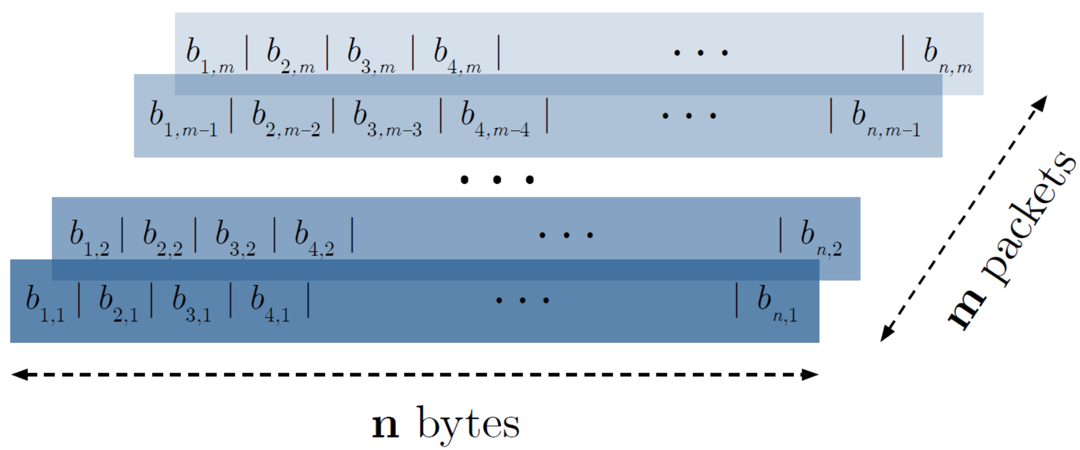
- **Flow representation:** matrix-like input, first **m** packets x first **n** bytes



(a) Packet representation for the input data. The shape of the input data is (N, n): N is the # of instances –packets– and n the number of steps –bytes–.

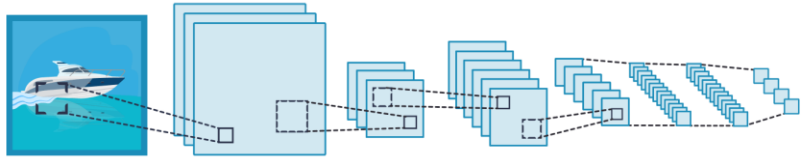
Raw Input Representations

- **Input representation of the data**, as well as **network architecture**, are both **key elements** to consider when **building a DL model**
- We take **two types of raw input representations:** ***packets*** and ***flows***. Decimal normalized representation of every byte of every packet is a different input
- **Flow representation:** matrix-like input, first **m** packets x first **n** bytes



(b) Flow representation for the input data. A tensor of size (N, m, n) where N represents the number of instances –flows–, m the number of channels –packets– and n the number of steps –bytes–.

Deep Learning
Architectural Principles

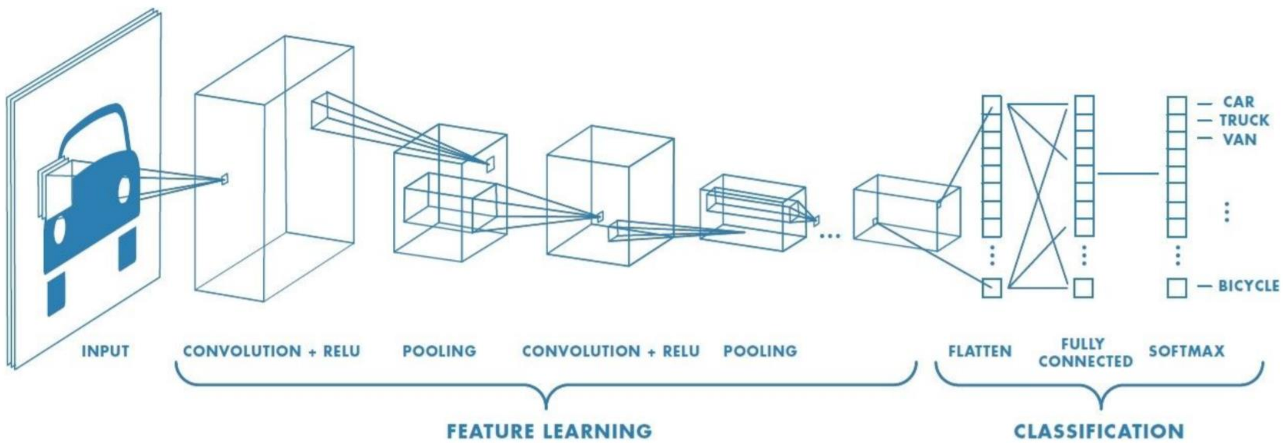


The **core layers** used for both models are basically two: **convolutional and recurrent**

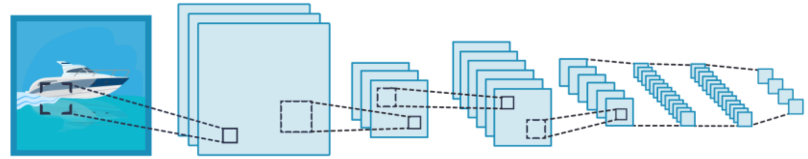
- Convolutional, to build the feature representation of the ***spatial*** data inside the packets and flows
- Recurrent layers are used together with the convolutional ones to allow the model keeping track of ***temporal information***
- **Fully-connected layers** to deal with the different feature combinations

Tricks for training convergence:

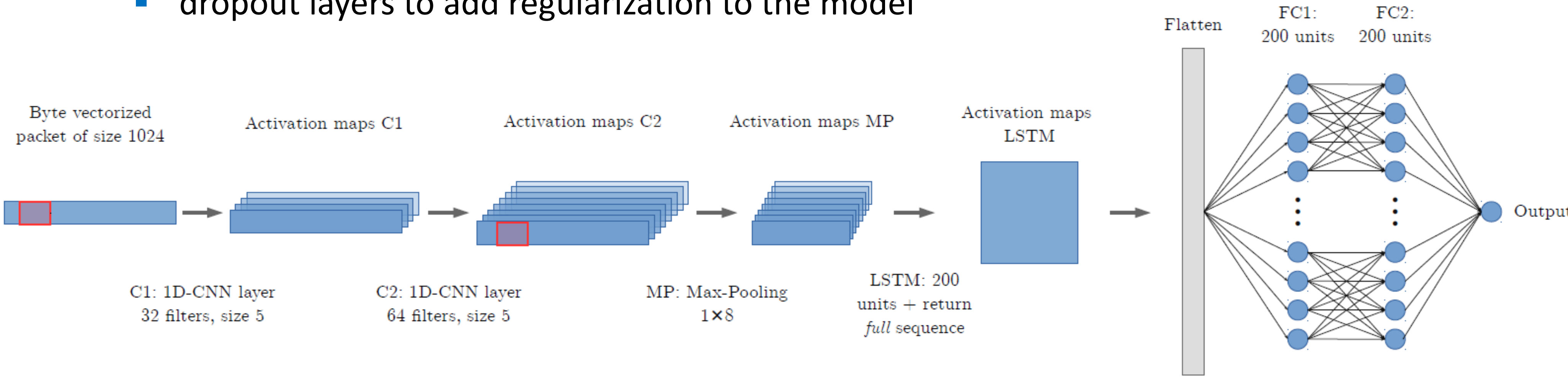
- **Batch Normalization**: layer inputs are normalized for each mini-batch → higher learning rates can be used, model less sensitive to initialization, adds regularization
- **Dropout**: randomly drop units (along with their connections) from the neural network during training → **model averaging**



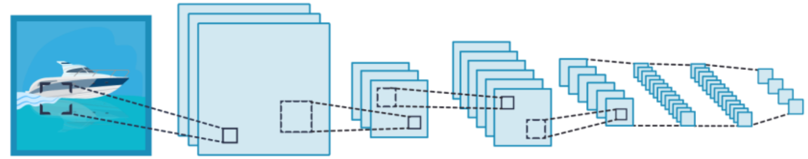
DL Architectures – Packets



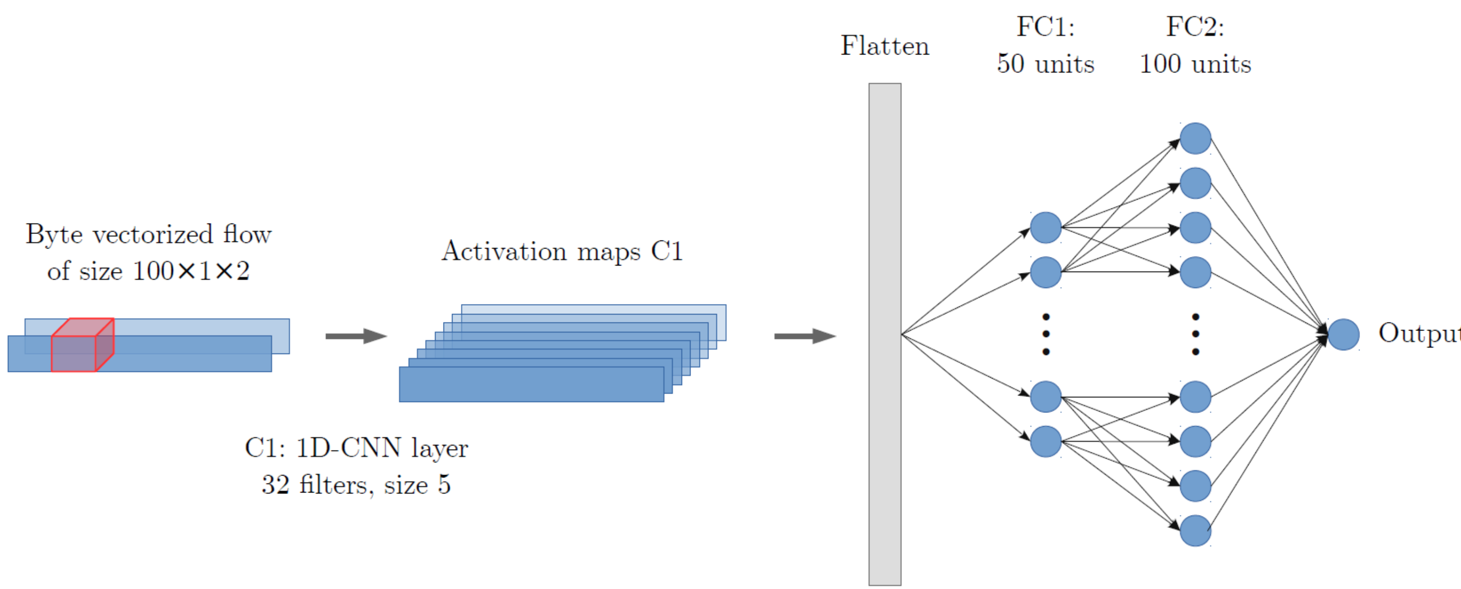
- **Raw Packets Architecture:**
  - **n** is set to first **1024 bytes**
  - two 1D-CNN layers of 32 and 64 filters (size 5) respectively
  - MP - max pooling layer (size 8)
  - LSTM layer with 200 neurons
  - two fully-connected layers of 200 neurons each
  - **binary cross-entropy as loss function**
  - spatial and normal batch normalization layers after each 1D-CNN and FC layers to ease training
  - dropout layers to add regularization to the model



DL Architectures – Flows

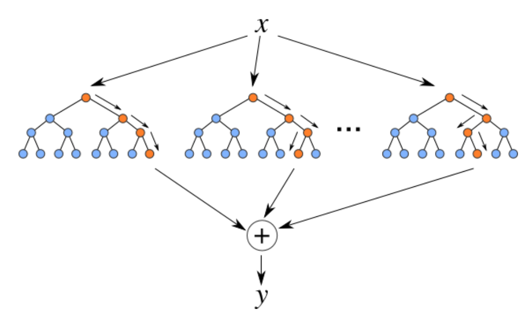


- **Raw Flows Architecture:** we go for a simpler model, with ***less features***
  - **n** is set to first **100 bytes**, and **m** to first **2 packets**
  - one 1D-CNN layers of 32 filter (size 5)
  - two fully-connected layers of 50 and 100 neurons each
  - **binary cross-entropy as loss function**
  - spatial and normal batch normalization layers
  - dropout layers to add regularization to the model



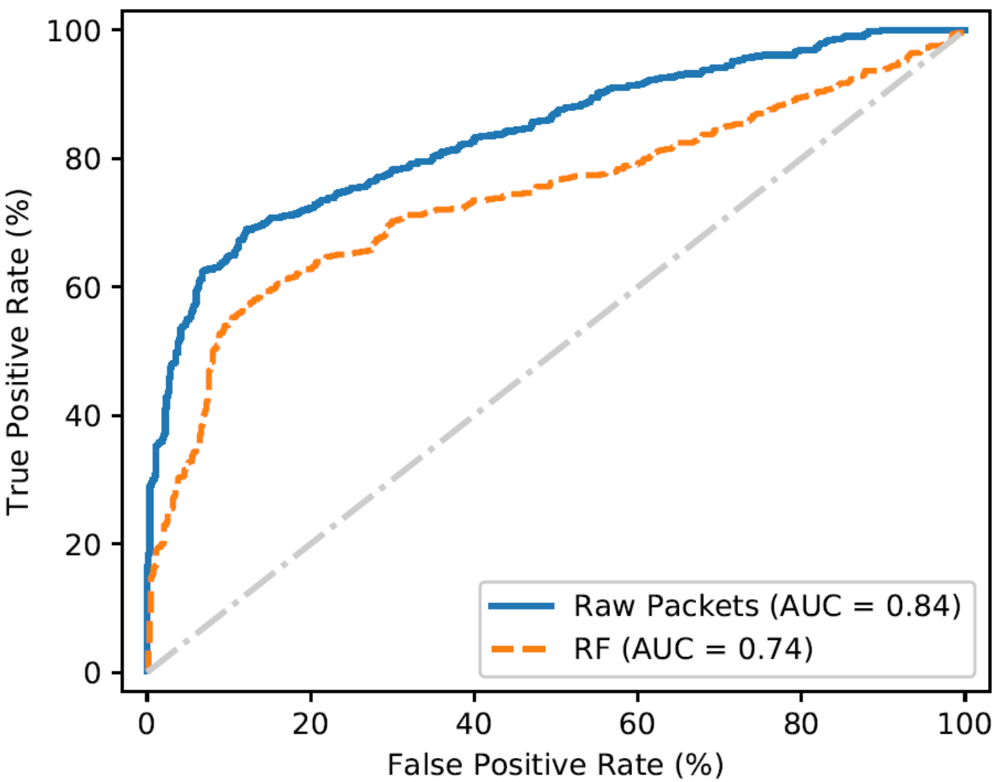
Evaluations

- All evaluations run on top of **Big-DAMA cluster** (distributed CPU)
- **Keras framework** running **on top of TensorFlow**
- **Dataset**: ***malware*** and ***normal*** traffic captures (pcap) performed by the Stratosphere IPS Project of the CTU University of Prague
- **250.000 raw packet** instances, **70.000 raw flow** instances
- **80%** of the samples for **training**, **10%** for **validation** and **10%** for **testing**
- **Compare** performance to **highly expressive Random Forest**:
  - same raw inputs
  - 100 trees
  - max depth and instances per leaf set for high expression
  - **selected based on great outperformance in state of the art**



RawPower – Packet Representation

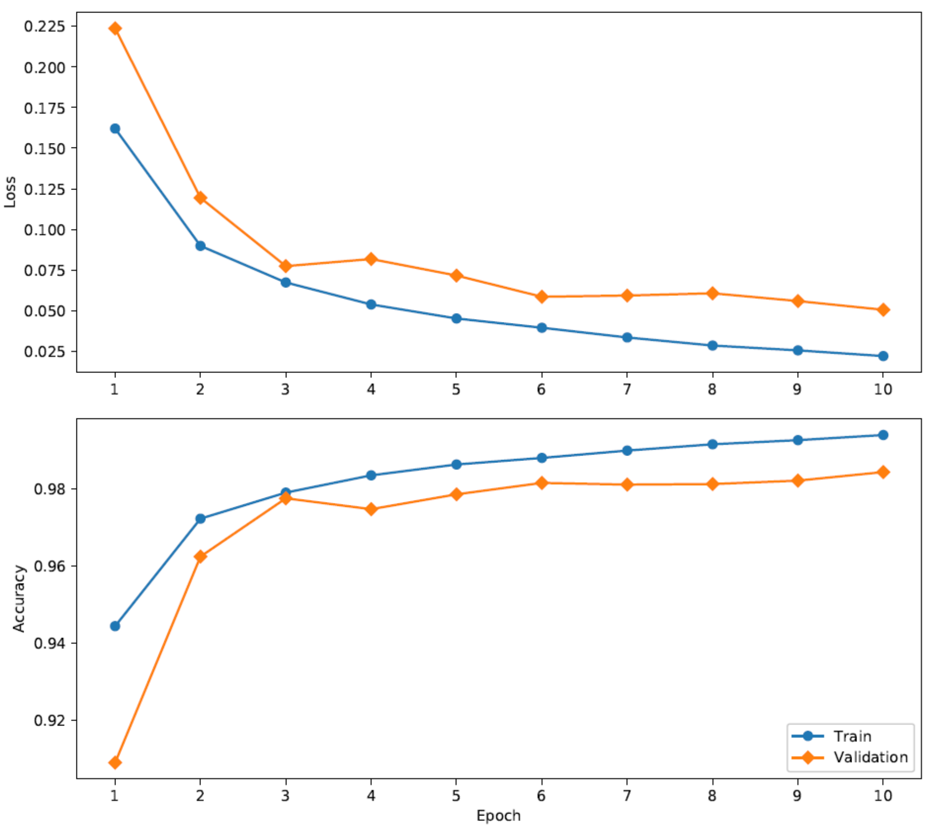
- Malware consists of **10 different malware types**, collected at controlled environment

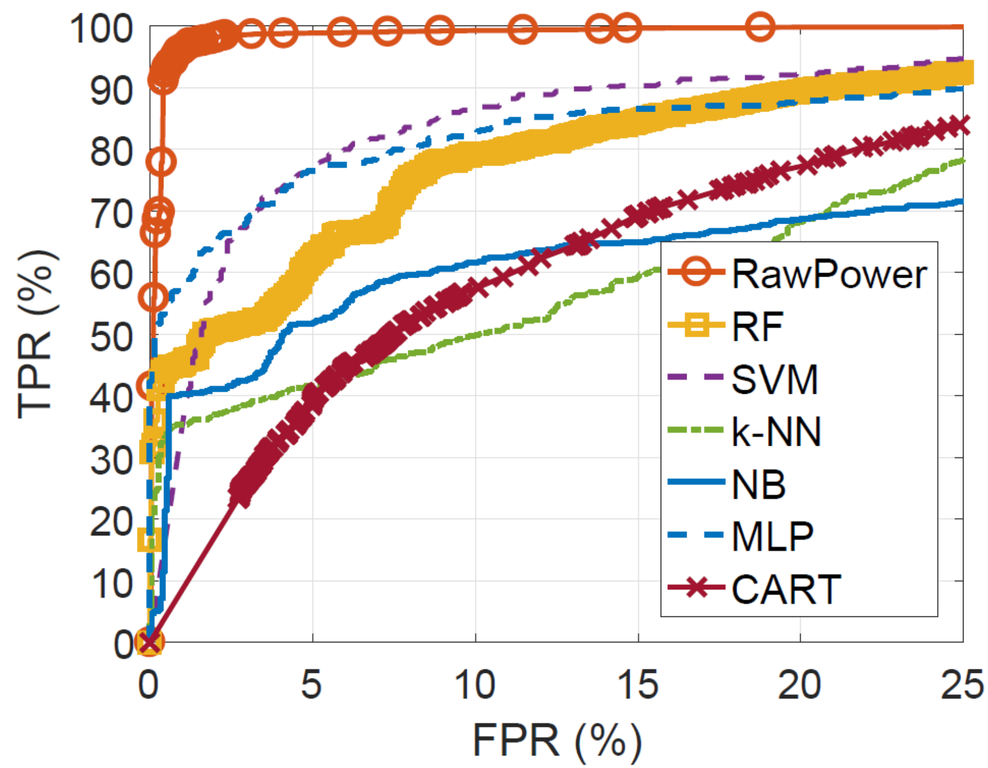


- **ROC curves** for both RawPower and RF
- Both models using the same **raw packet inputs**
- **Performance is not good at the packet-level**
- **Little gain w.r.t. a simple RF model**

RawPower – Flow Representation vs Shallow ML

- **Training and validation** evolution over **10 epochs**
- **Much better performance** at the **flow level (malware fingerprints)**
- RawPower can **detect almost 98% of the malware** flows with a FPR < 0.5%
- **Shallow models not able to capture the underlying relations**

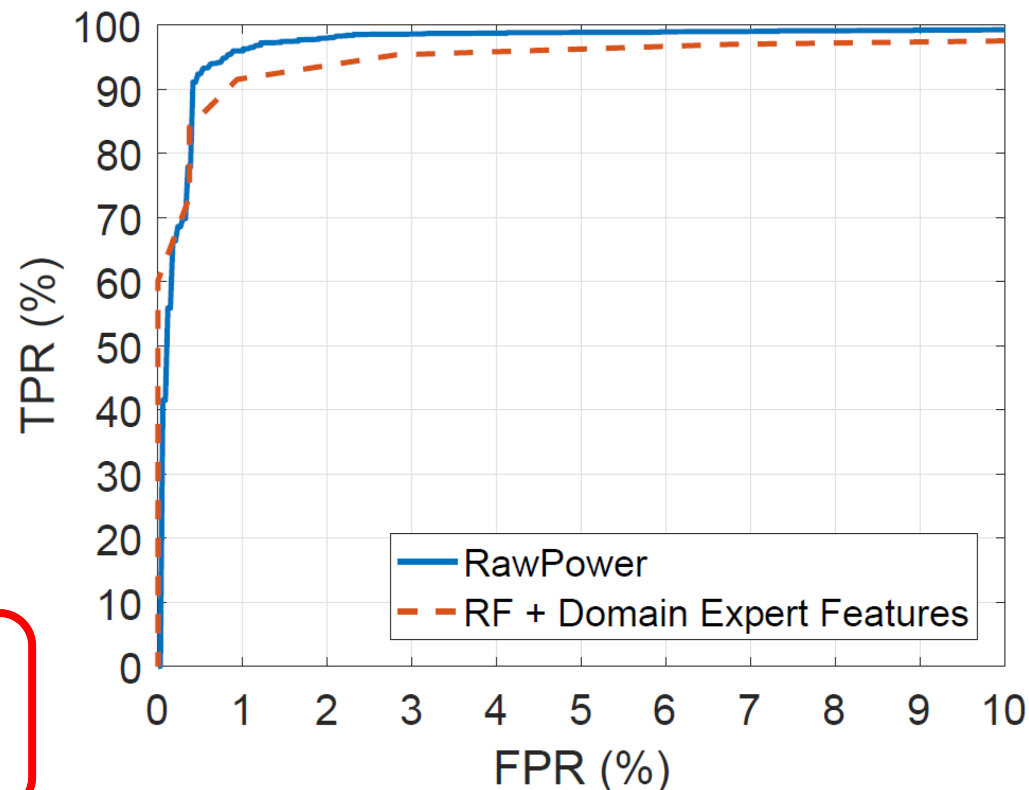




RawPower vs. shallow models.

RawPower – Flow Representation vs Expert Features

- Comparison **against** traditional **RF-based model**, which uses highly engineered **input features**, extracted from **domain knowledge**
- Both models provide **comparable results**
- The **key advantage of RawPower** is to rely directly on the usage of **bytestream raw data as input**
- **Input representation learning**: no the need for feature engineering



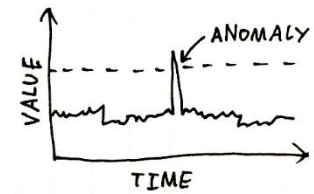
RawPower vs. knowledge-based inputs.

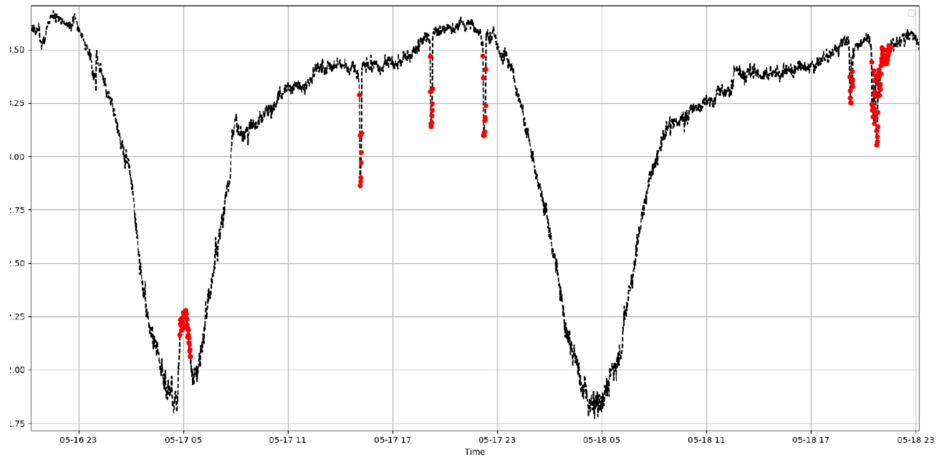
Organization of the Talk

Dealing with Some of these Challenges

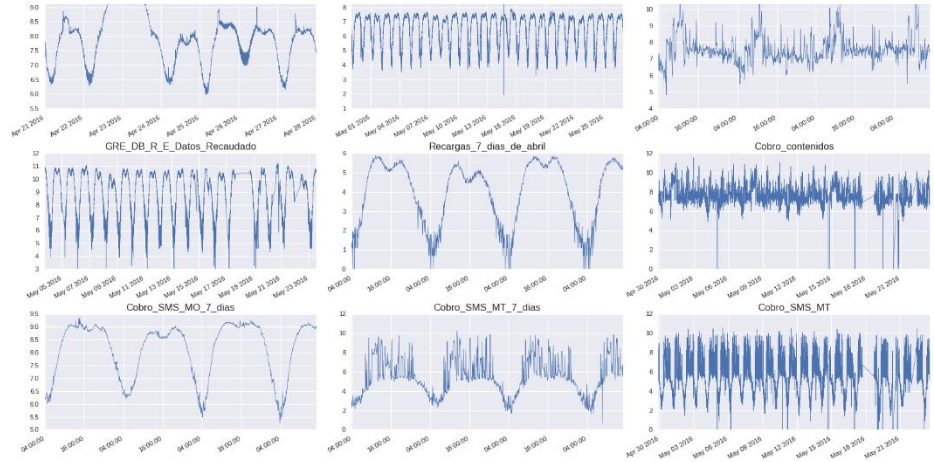
- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- **Generative Models for Anomaly Detection** – ***Avoid Traffic Modeling***
- Explainable Artificial Intelligence (XAI) **–** ***Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

Anomaly Detection in Multivariate Time-Series





Anomalies in an univariate time series



Different univariate time series of the same system

- **Anomaly Detection (AD)** is, by definition, an ***unsupervised process*** (detect what is different from the majority – the ***baseline***)
- **Baseline construction** (i.e., system modeling) is ***complex*** and ***error prone***, especially when dealing with ***multi-dimensional*** system characterization
- Solution: delegate the baseline construction to ***generative models***

Generative Models

- Given training data, ***generate new samples*** from ***same distribution***

Training data ~ $p_{data}(x)$

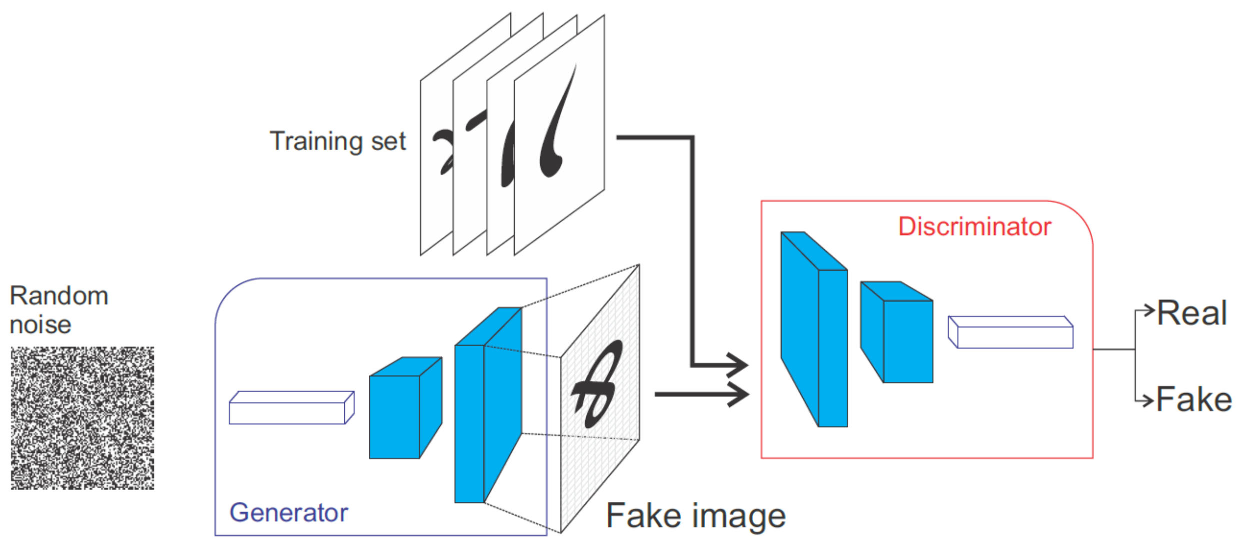
Generated samples ~ $p_{model}(x)$

- The **problem of generative models** is about **learning $p_{model}(x)$ similar to $p_{data}(x)$**

- Generative model learning is about **density estimation:**
  - **Explicit density estimation**: explicitly define and solve for $p_{model}(x)$
  - ***Implicit density estimation***: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

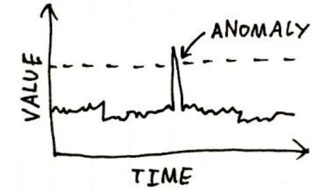
Generative Adversarial Networks (GANs)

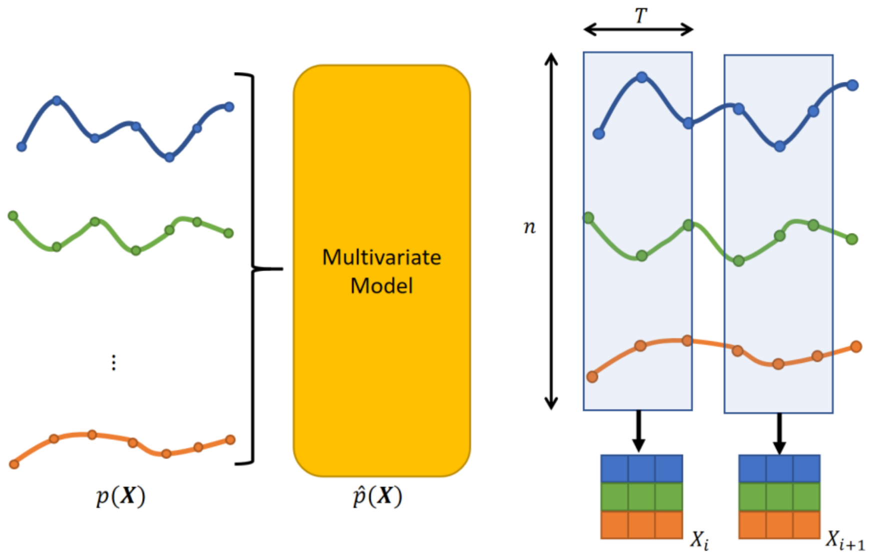
- Implicit density estimation through **game-theoretic approach**
- Learn to generate samples from training distribution through **2-players (minimax) game**
- **Problem:** want to sample from potentially complex, high-dimensional training distribution. No direct way to do this!
- **Solution:** sample from a simple distribution, e.g. **random noise**. Learn transformation to training distribution, **using a neural network**

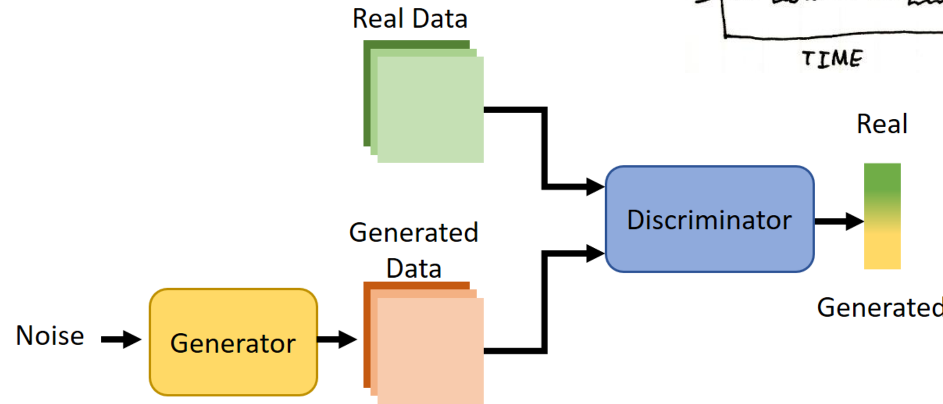


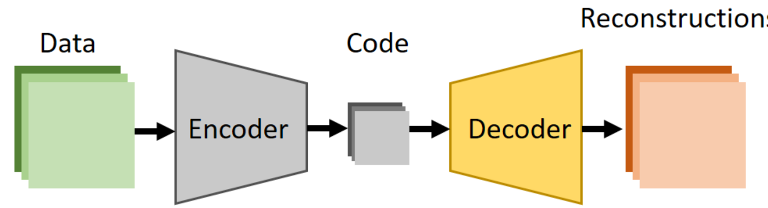
- **Generator** network: tries to fool the discriminator by generating real-looking instances from random noise
- **Discriminator** network: tries to distinguish between real and fake instances

Generative Models for MV-TS Anomaly Detection







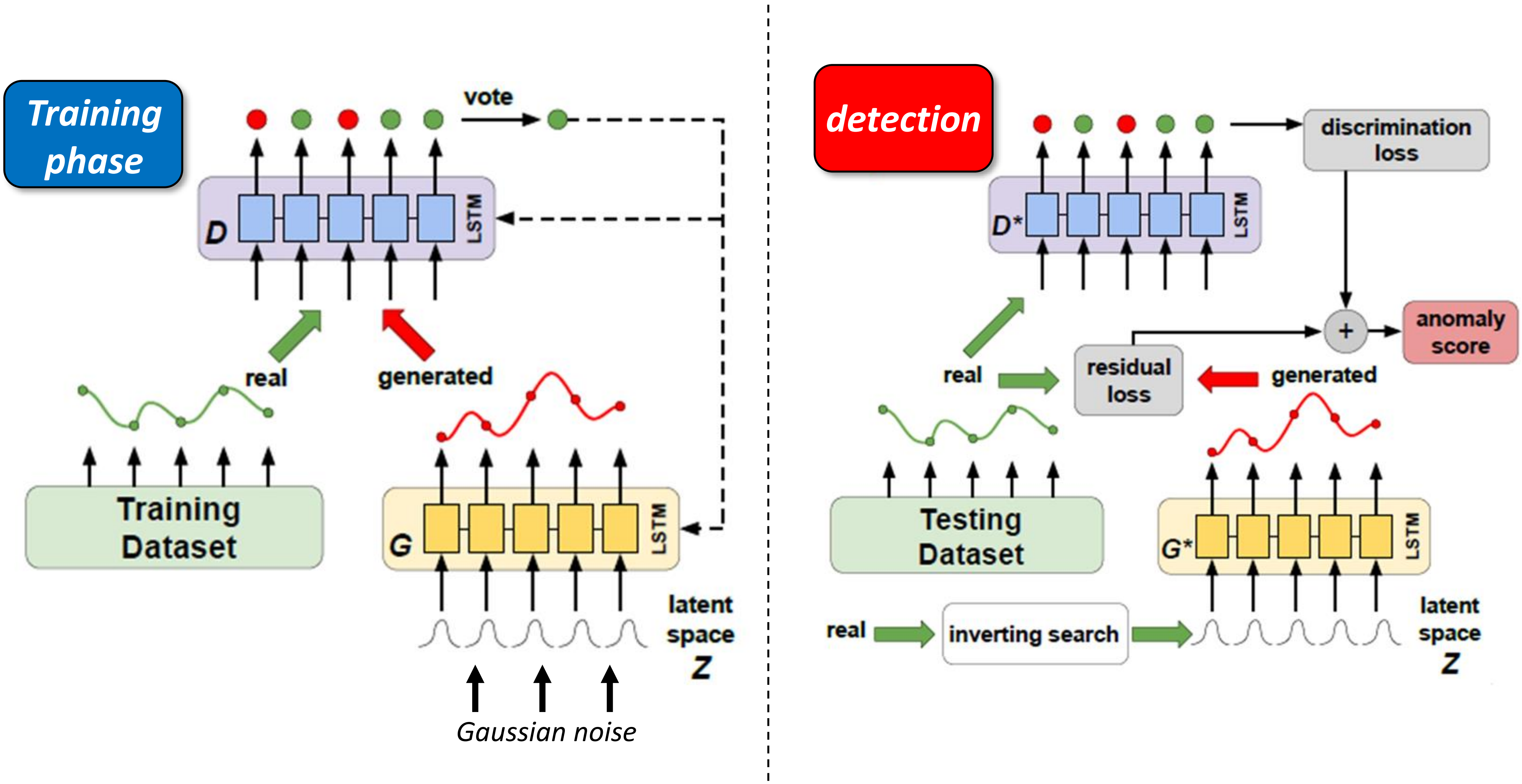


Two different generative models for ***AD in multi-variate time series***

- ***Net-GAN***: Recurrent Neural Networks (LSTM) trained through GANs
- ***Net-VAE***: Variational Auto-Encoders (VAE) using feed-forward NNs
  - VAEs improve Auto-Encoders by ***regularizing the latent-space*** → enabling ***generative process***
- Input samples: matrix with *n* (number of variables) x *T* (length of sequence)

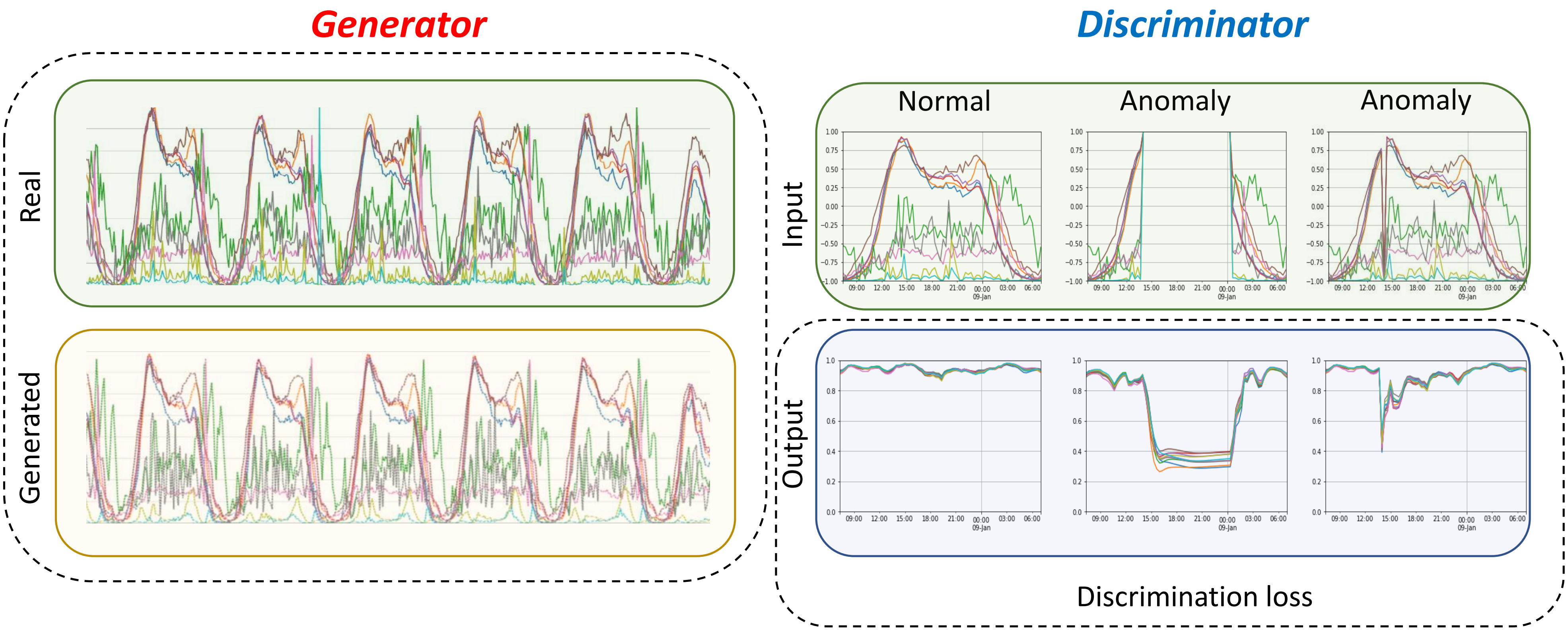
Network Anomaly Detection with Net-GAN

- ***Net-GAN detection*** can be done both through the ***generator (G)*** and the ***discriminator (D)***



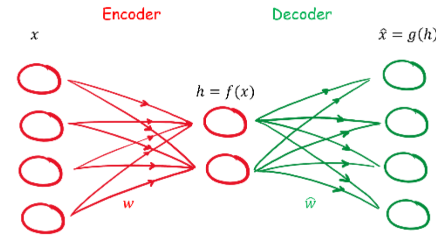
Examples on Real (Mobile) ISP Network Data

Net-GAN application phase

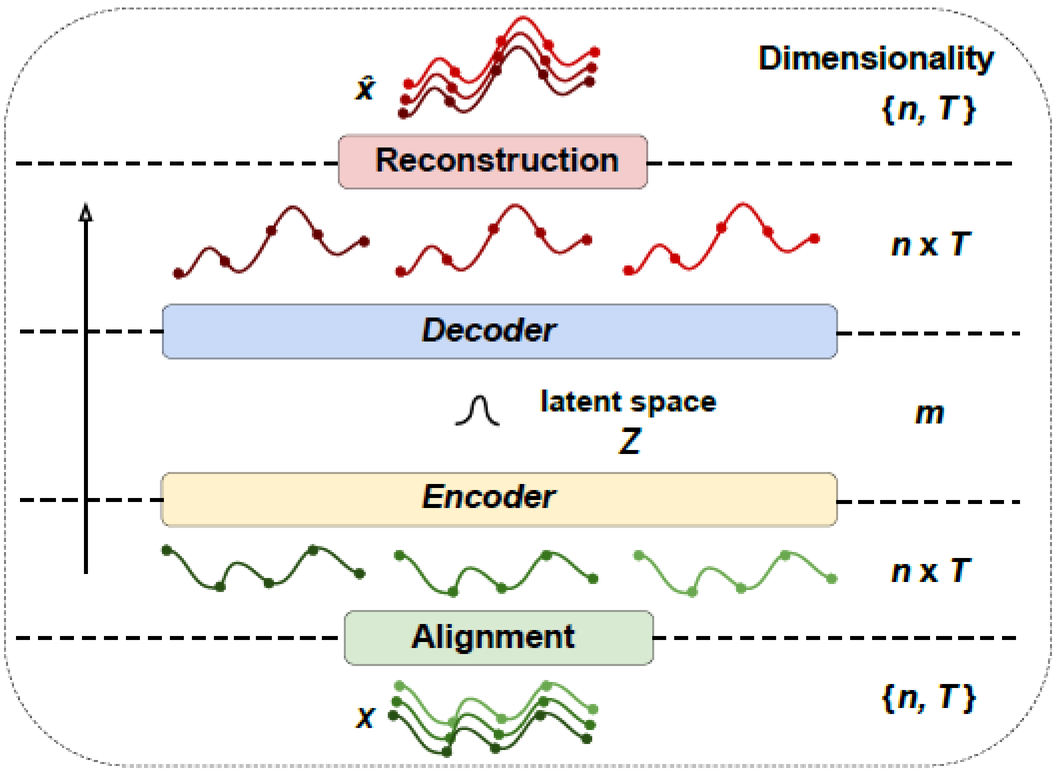


- ***Net-GAN AD generator (G)*** → residual loss
- ***Net-GAN AD discriminator (D)*** → discrimination loss

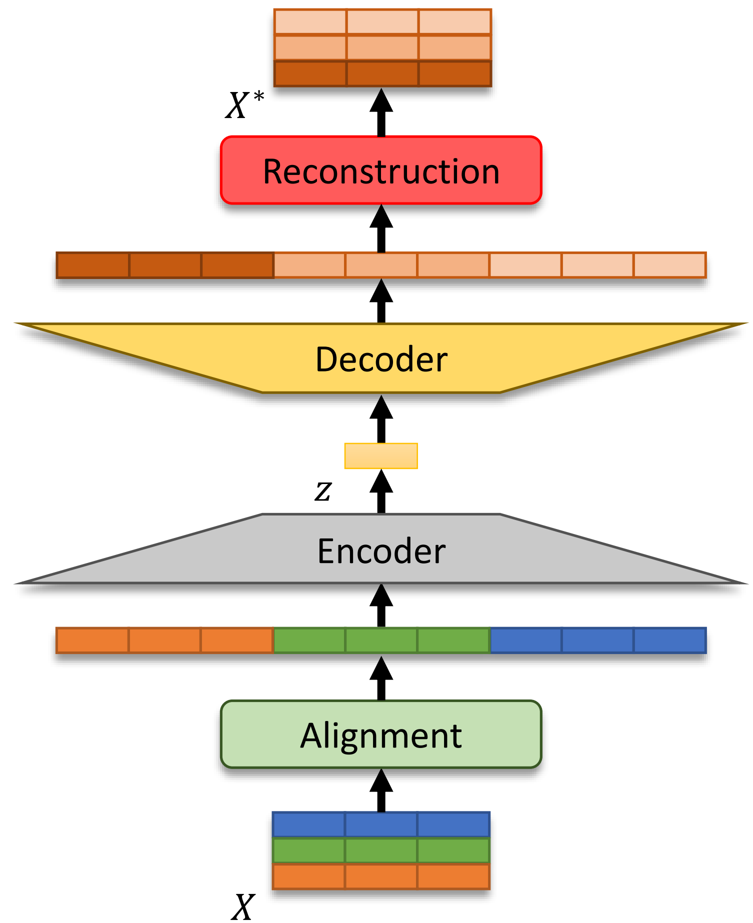
Network Anomaly Detection with Net-VAE



- Net-VAE architecture:
  - standard encoder and decoder functions
  - encoder/decoder using ***3-layer FF networks***
  - detection on ***residual loss***

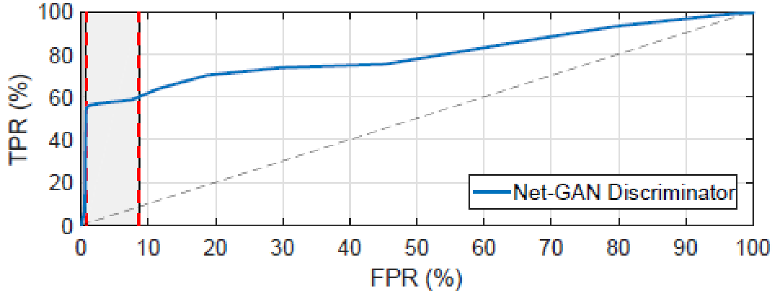


Net-VAE architecture.

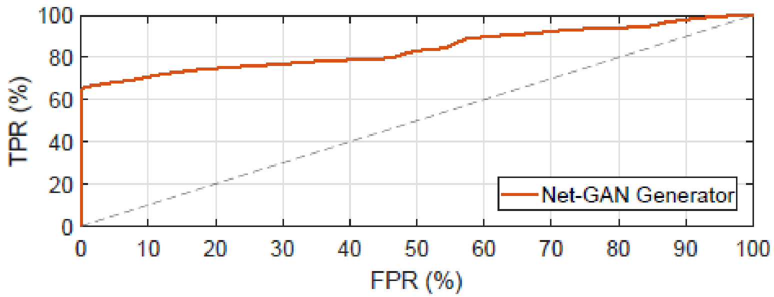


Anomaly Detection with Net-GAN and Net-VAE

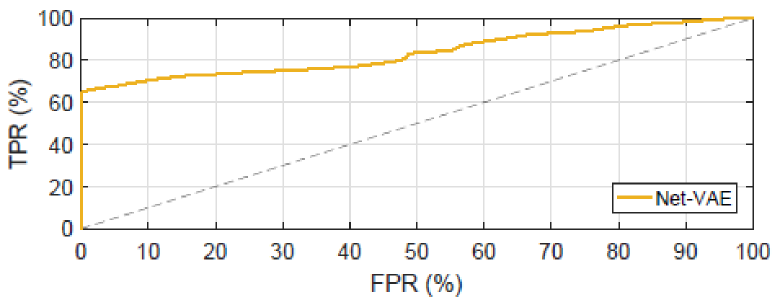
SWaT (CPS measurement)



(a) Detection performance with Net-GAN-D.

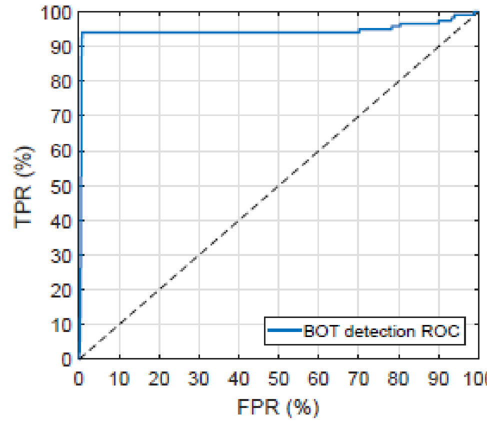


(b) Detection performance with Net-GAN-G.

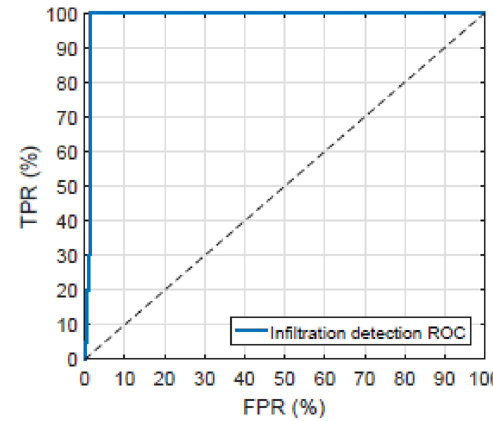


(c) Detection performance with Net-VAE.

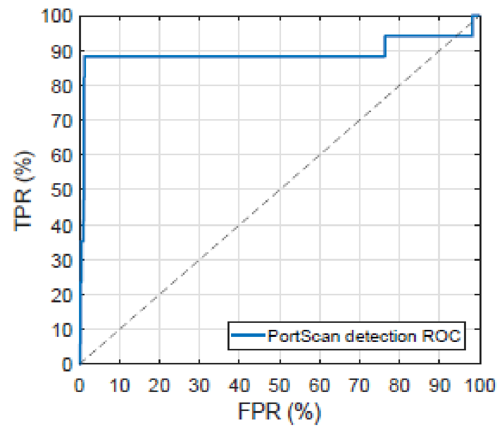
CICIDS2017 (SYN-NET measurements)



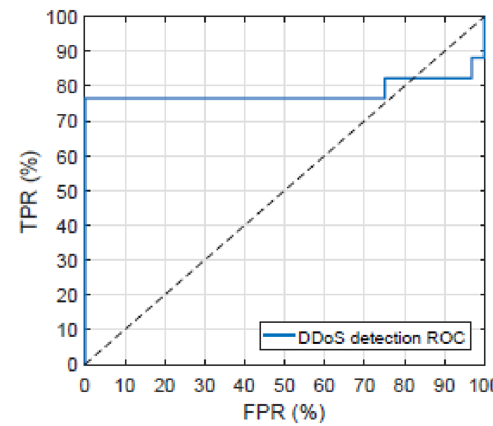
(a) Botnet



(b) Infiltration



(c) Port Scan



(d) DDoS

Organization of the Talk

Dealing with Some of these Challenges

- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- Generative Models for Anomaly Detection – ***Avoid Traffic Modeling***
- **Explainable Artificial Intelligence (XAI) –** ***Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

EXplainable AI (XAI) – Why Should I Trust You?

- **ML models →** mostly are **black boxes** (exceptions: linear models, decision trees, etc.) – e.g.: some popular ML models have 10s of millions of parameters!

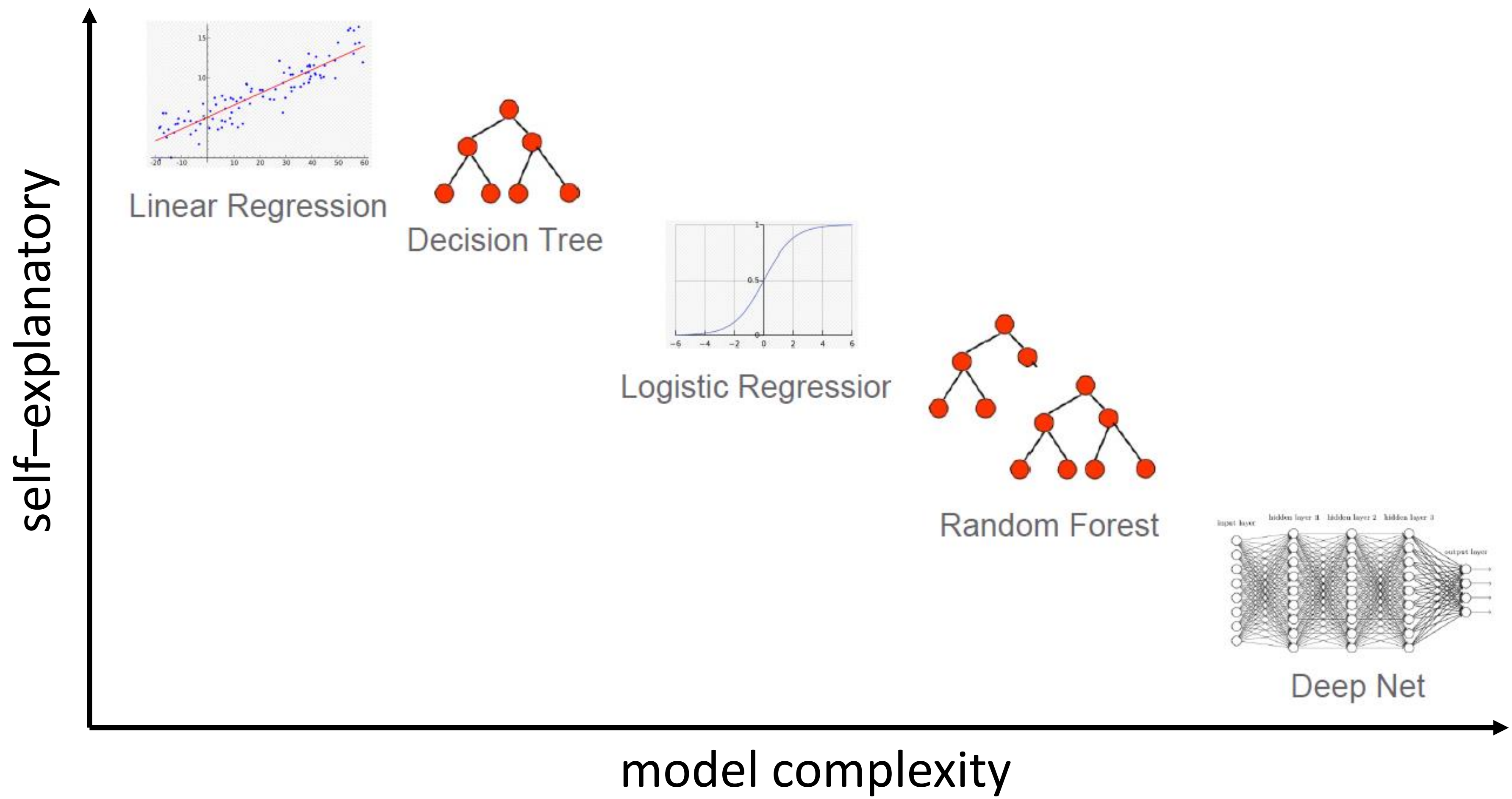
- Models are evaluated off-line before deployment on available test datasets – **data @runtime might change** (concept drift)

- Humans want to **understand model's behavior to gain trust** (applicability in the practice)
  - trusting an individual model's prediction
  - trusting a model (inspect a set of representative individual predictions)

- **Explainable AI:** approaches capable to explain models and individual predictions, by tracking back to the inputs leading to a certain output

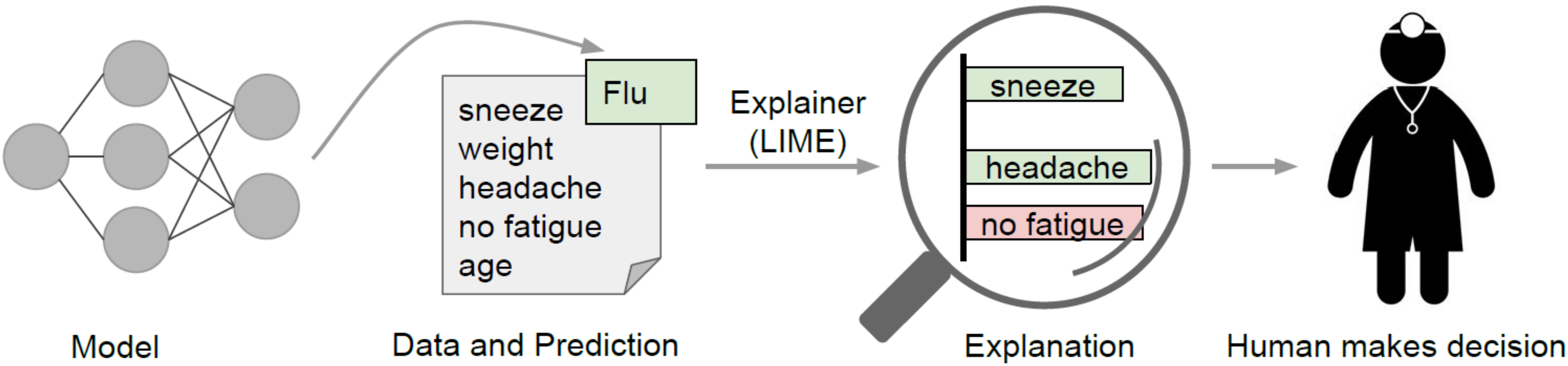
Why XAI?

- **Ideally**, **ML models** should be ***self-explanatory:*** **improve end-user understanding** and **trust**, by offering simple **explanations** of the ***"whys" of certain decision***
- **Only few models are self-explanatory:**



A Simple XAI Example

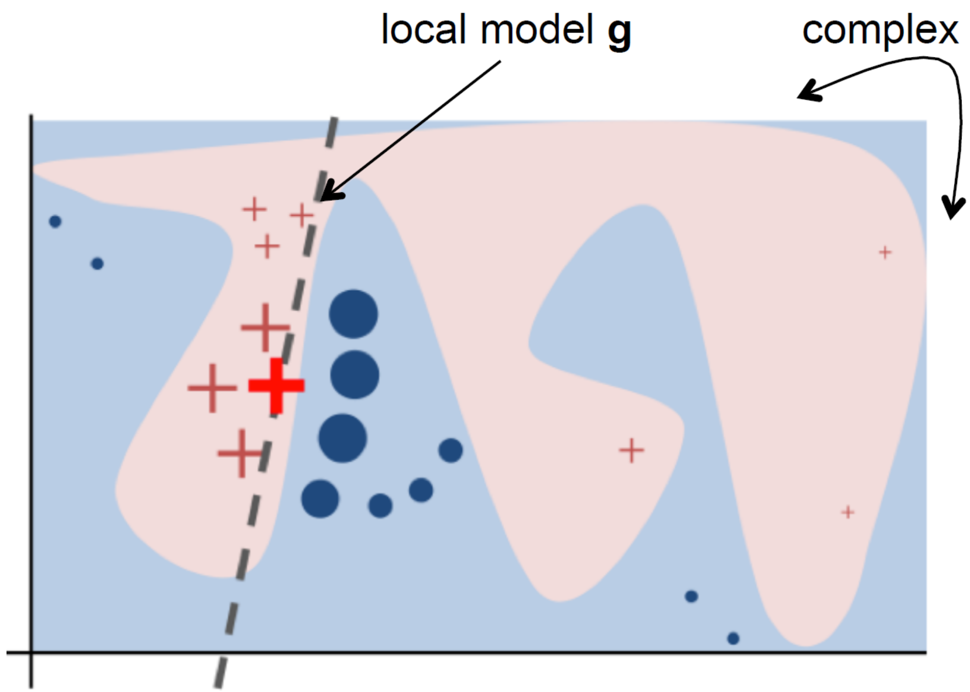
- **Application Example:** AI-supported disease diagnosis



- Explainer: **LIME** - **Local** Interpretable **Model-agnostic** Explanations
- ***LIME approach***: builds an **interpretable model** that is **locally** faithful to the classifier under analysis
- Other approaches: **SHAP**, LRP (NNs), PDP, etc.

LIME in a Nutshell – Sampling for Local Exploration

- Let **f** be an unknown complex decision function (**blue**/**pink** background)

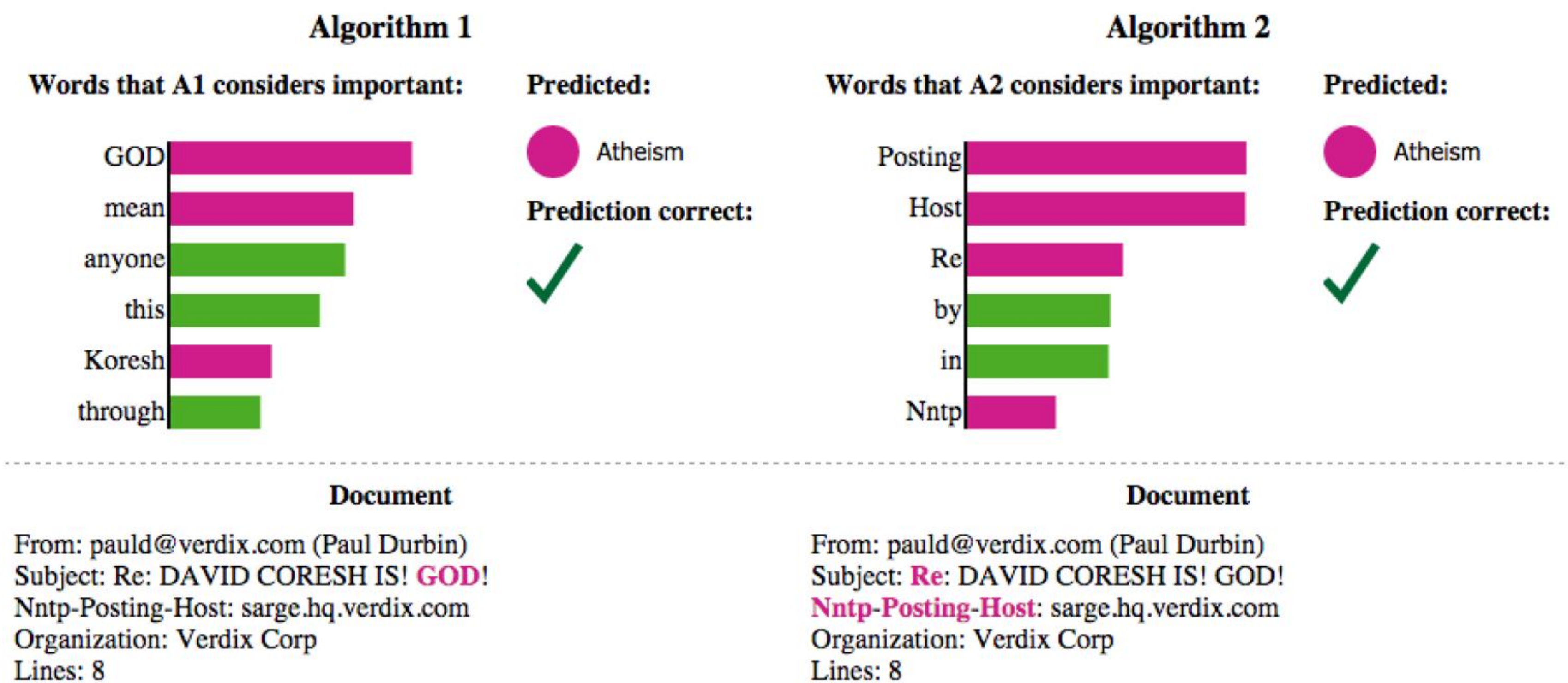


- **g** is *interpretable*, *locally faithful to* **f** (captured by ***D*$_{x}$**), and *model agnostic* (uses **f(z)** as labels)
- *robust to sampling noise*, thanks to ***D*$_{x}$**

- The **bold red-cross (x)** is the instance we want to explain
- LIME samples instances **z** around **x**, weighted by some **similarity measure *D*$_{x}$** → ***D*$_{x}$**(z) is higher for instances closer to x
- Using model **f**, gets the corresponding predictions **f(z)**
- Finally, it uses **z** and **f(z)** to **build an interpretable model g** (e.g, linear) around **x**

LIME Examples (I) – Model Comparison/Selection

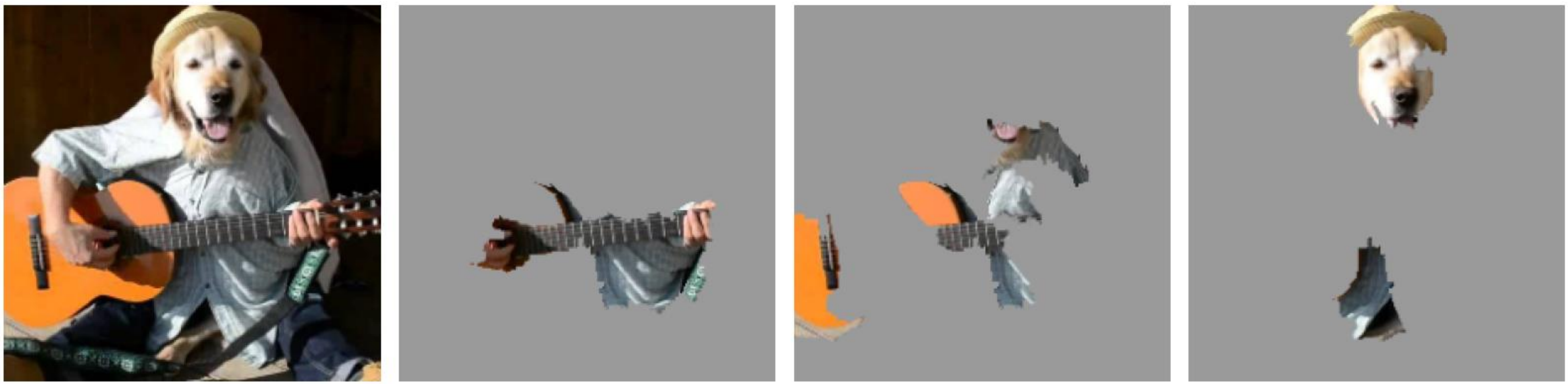
- Task: word-based **email classification**, **Christianity** or **Atheism**
- 2 models (**Algorithm 1** vs **Algorithm 2**), **which one is better?**



- **Algorithm 2** is better than **Algorithm 1** in terms of **accuracy in validation…**
- …but **Algorithm 2** makes predictions for arbitrary reasons…**Algorithm 1** is better
- **Performance metrics should be carefully considered**

LIME Examples (II) – Model Performance Evaluation

- Task: **image classification**, using Google's pre-trained Inception CNN architecture



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

- Figs. (b,c,d) report super-pixel explanations provided by LIME

- Top 3 classes: ***Electric Guitar*** (p = 0.32), ***Acoustic Guitar*** (p = 0.24), and ***Labrador*** (p = 0.21)

- The image is wrongly classified, but **explanations provide trust** in the model, as they are reasonable

LIME Examples (III) – Discover Biased Data

- Task: train a classifier to distinguish between Wolves and Huskies
- Biased data (e.g., **undesirable strong correlations**) → wrong classifier
- **Hard to identify** by looking at the raw data and predictions

(a) Husky classified as wolf (b) Explanation

- Bias@training: all pictures of Wolves had **snow in background**
- The **classifier performs well** according to **cross-validation** in this **biased dataset...**

- ...but **explanations** of individual predictions show that **the model learnt a biased pattern**: if snow → wolf, else → Husky

Organization of the Talk
Dealing with Some of these Challenges

- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- Generative Models for Anomaly Detection – ***Avoid Traffic Modeling***
- Explainable Artificial Intelligence (XAI) **–** ***Interpret Model Decisions***
- **Super Learning for Network Security** – ***Avoid Model Decision***
- Adaptive/Stream Learning for NetSec – ***Deal with Concept Drifts***

Ensemble Learning for Network Security

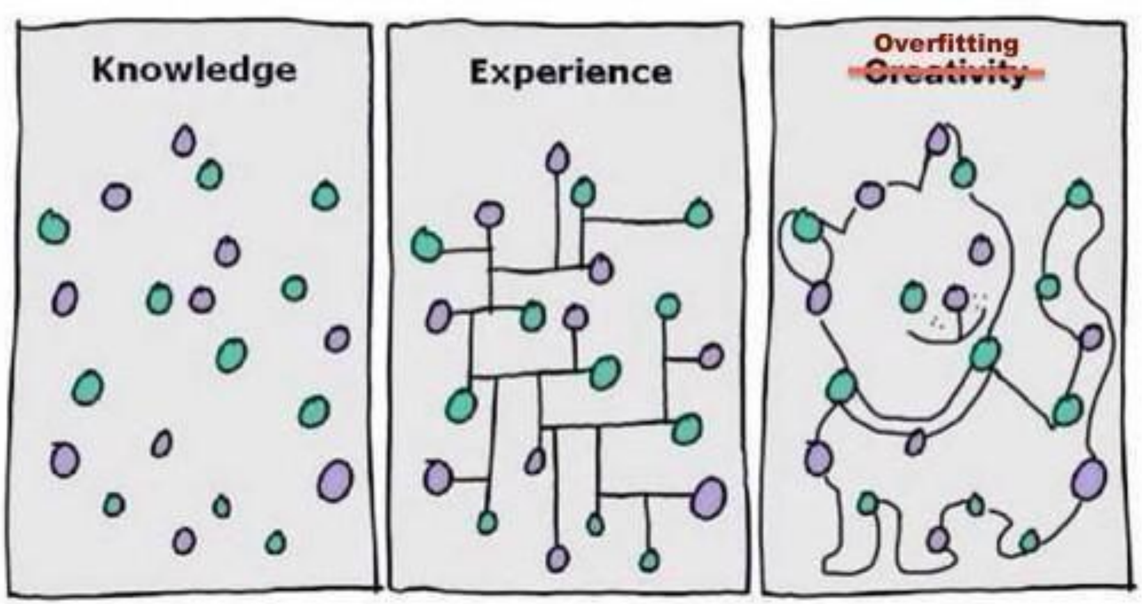
- Which is the **best model** or category of models for a **specific learning task**?
- Deep Learning? Not obvious in the context of Network traffic Monitoring and Analysis (NMA)
- Our claim: "***multiple-eyes principle***" → **ensemble learning** models
- We explore the application of **ensemble learning models** to multiple NMA problems…
- …following a particularly promising model known as the **Super Learner**

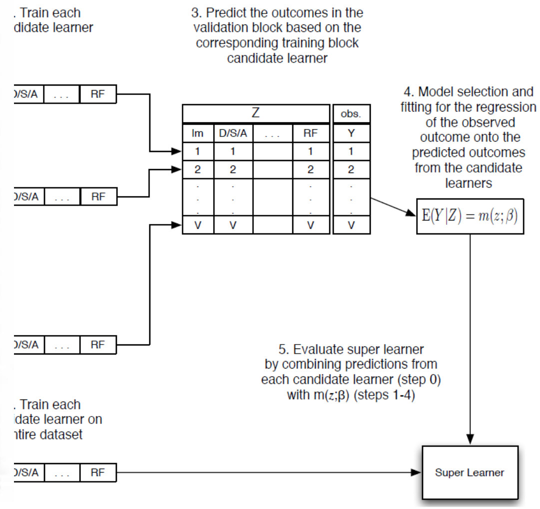
Ensemble Learning for Network Security

ensemble learning: combine multiple (base) learning models to obtain better performance.

- If a set of base learners do not capture the true prediction function (the oracle), **ensembles** can give a **good approximation** to that **oracle function**.
- **Ensembles perform better** than the individual **base algorithms**.
- **Multiple approaches** to ensemble learning, including **bagging** (decrease variance), **boosting** (decrease bias), and **stacking** (improve predictive performance)

Super Learner



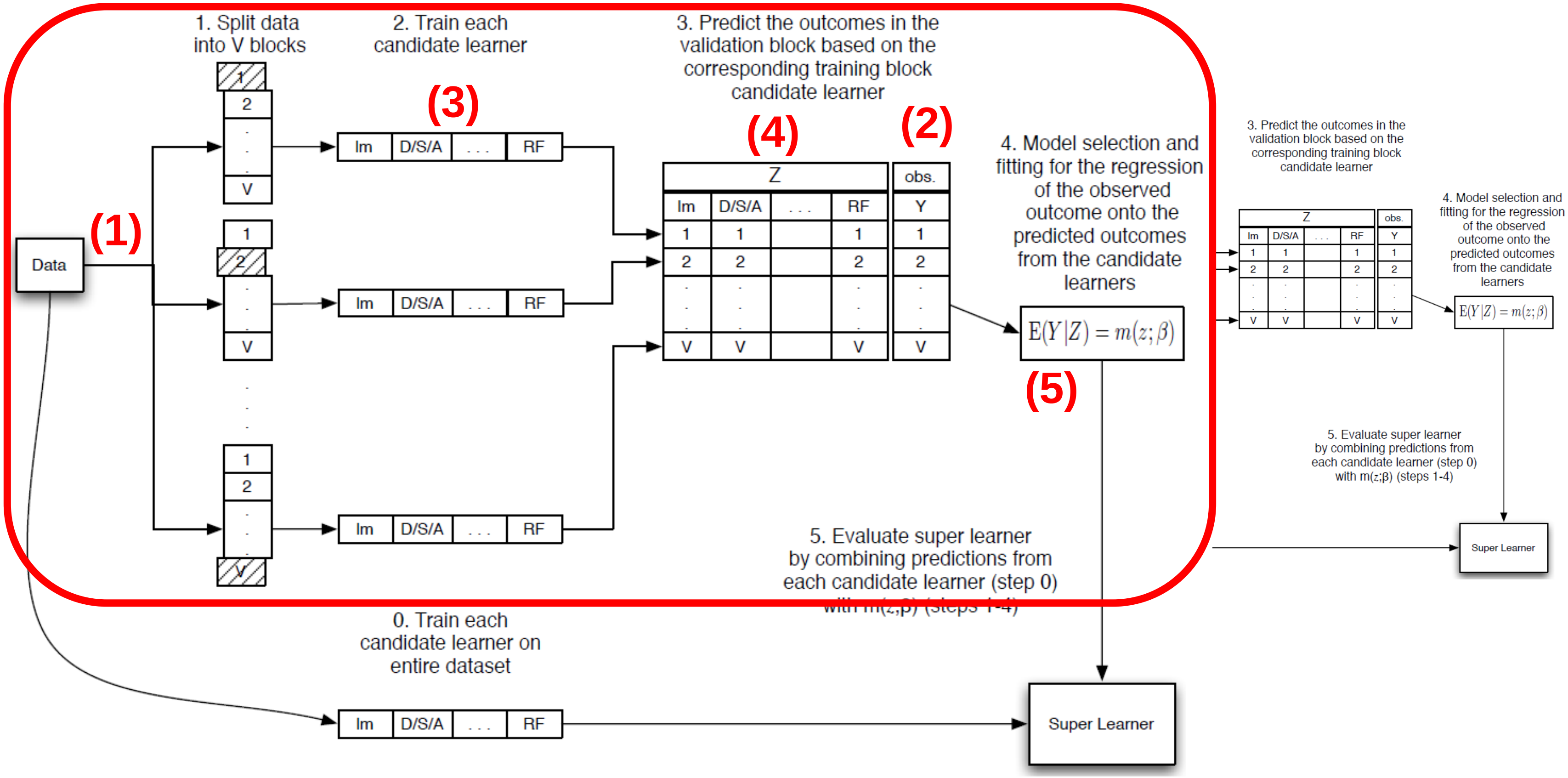


- **General ensemble learning** approaches might be **prone to over-fitting**.

- **Super Learner** [Van der Laan'07]: **stacking** ensemble learning **meta-model** that **minimizes over-fitting likelihood** using a **variant of cross-validation**.

- Finds the **optimal combination of a collection of prediction algorithms** → performs asymptotically as well – or better, than any of the ***base learners***.

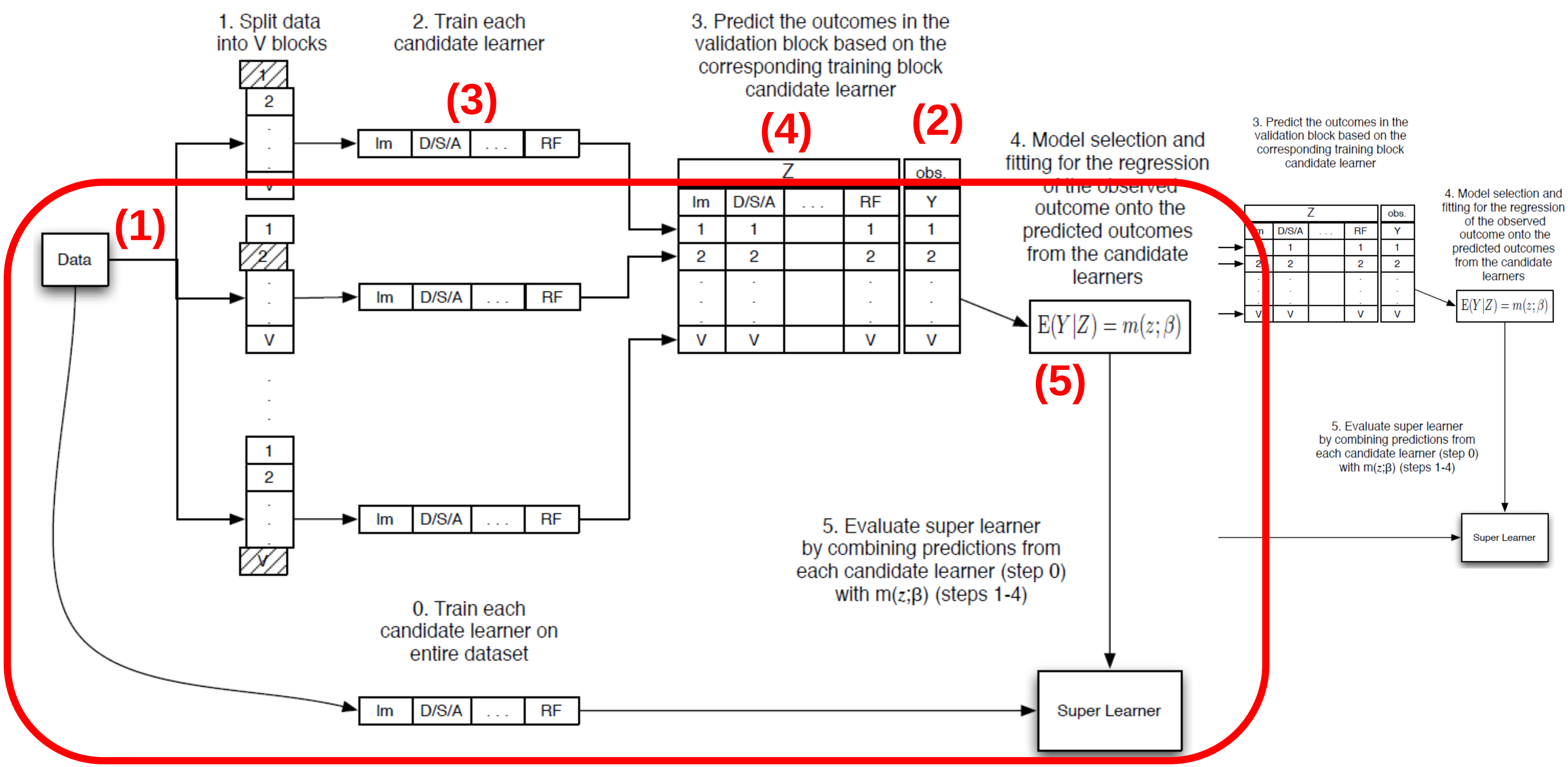
Super Learner – How Does it Work?



- **2-steps approach (training and validation of Super Learner):**

  1st → *given a (1) **dataset X(k,l)** with **labels Y(k)** (2), and a set of (3) **n base learners** (e.g., DTs, ANNs, SVM, etc.), build a (4) **new dataset {Z(k,n), Y(k)} (by cross validation)** to (5) train the **Super Learner model m(z,β)***

Super Learner – How Does it Work?



- **2-steps approach (training and validation of Super Learner):**

  2nd → *train each of the **n base learners** using training/validation split of {X,Y}, and **compute predictions** (on top of validation set) **using meta-model m(z,β) (trained in step 1)***

GML Learning for NMA

- The **Super Learner meta-model** could be **whatever algorithm**
- The **original work** [Van der Laan'07] uses a **simple minimum square linear regression model** as the example **Super Learner**.
- ***Problem***: how to **define weights** to **perform** properly in **every dataset**?
- ***GML Learning***: computes **weights** with an **exponential probability of success, reducing** the **influence** of **poor base learning models**.

$$m_{GML}(X) = \sum_{j=1}^{J} w_j h_j(X)$$

base learners

$$w_j = \frac{e^{\lambda \alpha_j}}{\sum_{i=1}^{J} e^{\lambda \alpha_i}}$$

base learner accuracy

control variable: reduces weight for low accuracy predictors

Models Benchmarking

We compare several models for NMA:

- We take **5 standard base learning models**: linear **SVM, CART, *k*-NN, ANN (MLP)** and **Naïve Bayes**

  We **build 4 different Super Learners:**

  1. **Logistic regression** (binary output 0/1)
  2. **Weighted Majority Voting (MV):**
     - **MVuniform**: same weight to each base learner
     - **MVaccuracy**: weights are computed using base learner accuracy
  3. **Decision Tree meta-learner** (CART)

- **Boosting** (ensemble learning)**: AdaBoost tree**
- **Bagging** (ensemble learning): **Bagging tree** and **Random Forest**
- ***GML Learning***

Multiple NMA Problems

Five network measurement problems for model benchmarking:

1. **NS –** ***detection of network attacks*** in **WIDE/MAWI traffic** (transpacific links)

2. **AD –** ***detection of smartphone-apps anomalies*** in cellular networks (data captured at core cellular network)

3. **QoE-P –** ***QoE prediction*** in cellular networks (data captured at smartphones)

4. **QoE-M –** ***QoE-modeling for video streaming*** (smartphones public datasets)

5. **PPC –** ***Internet-paths*** dynamics tracking – ***prediction of path changes*** (M-Lab traceroute measurements)

(some) Evaluation Datasets

- We focus on two NMA problems:
  - Detection of **Network Attacks** in **WIDE/MAWI** network traffic
  - Detection of **App-related Anomalies** in an **Operational Cellular Network**

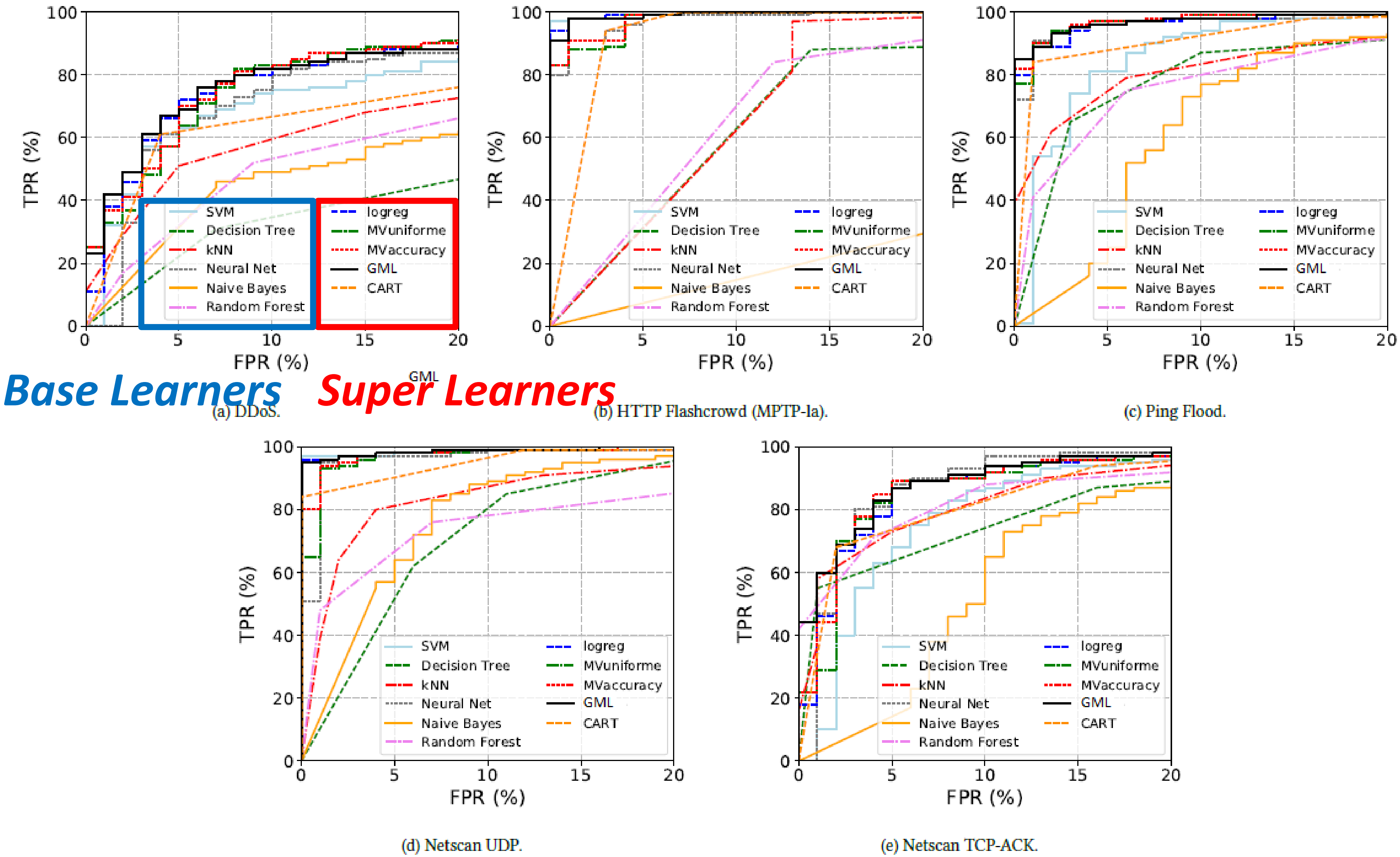
- **WIDE Network traffic using MAWI labels**
  - traffic traces captured daily on backbone link between Japan and the US.
  - **MAWI labels**: uses a combination of four traditional anomaly detectors to label the collected traffic by majority voting.
  - 5 attack classes: **DDoS, flashcrowd, netscans (TCP/UDP), flooding.**
  - The dataset spans a **full week of traffic traces collected in late 2015**; traces are split in **consecutive time slots of 1 second**.
  - **245 features** describe the traffic in each of these slots.
  - These include throughput, packet sizes, IP addresses and ports, transport protocols, flags (empirical distributions, sampled at multiple percentiles), and more

(some) Evaluation Datasets

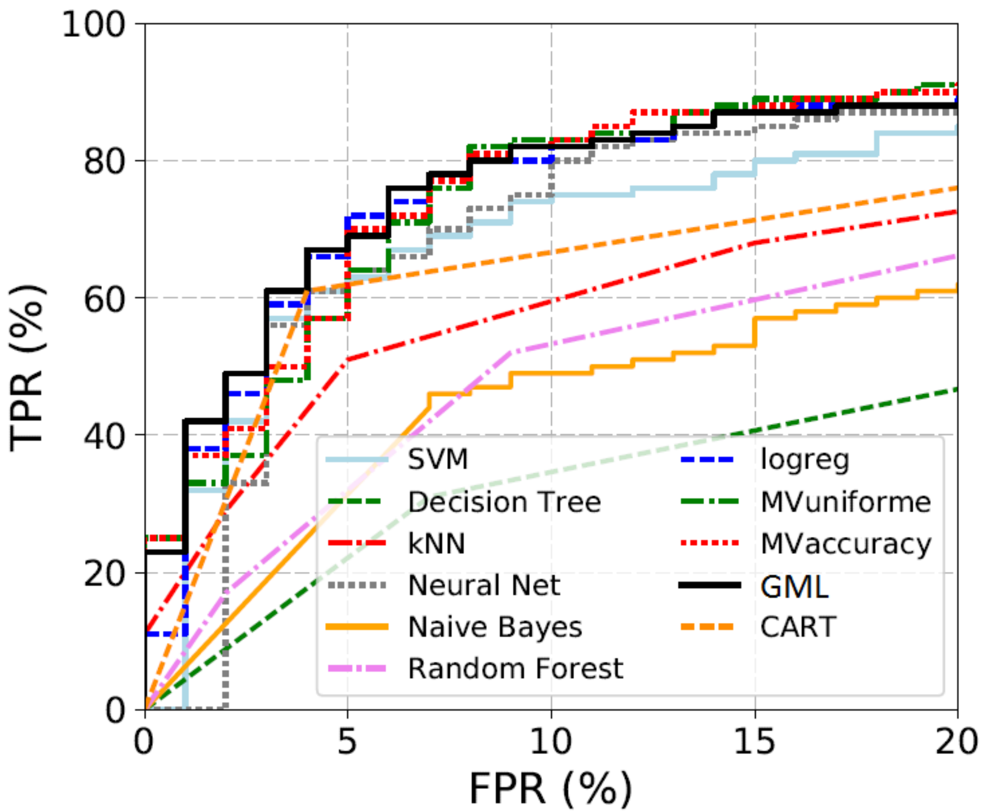
- We focus on two NMA problems:
  - Detection of **Network Attacks** in **WIDE/MAWI** network traffic
  - Detection of **App-related Anomalies** in an **Operational Cellular Network**

- **Synthetically generated dataset for AD in cellular networks**
  - **derived from real cellular ISP measurements** (traffic measurements collected during 6-months in 2014)
  - **Anomaly Templates**, derived from real app-related anomalies observed in the cellular traffic → in this paper, anomaly types E1, E2 and E3
  - **Evaluation labelled dataset**: 1 month of normal operation traffic, and 16 different anomaly instances of E1, E2 and E3 types, with different intensity (number of involved devices varies from 0.5% to 20%)
  - **36 features** describing 10’ time slots
  - These include FQDNs, DNS error flags, APN, operative system and manufacturer (empirical distributions, sampled at multiple percentiles)

Benchmark for Network Security



Benchmark for Network Security



(a) DDoS.

- **Super Learners (SLs) outperform** both **base learners**, as well as the **RF model**
- The CART SL performs the worst → **regression-based models are more accurate for SL**
- **GML slightly outperforms other SLs**

Benchmark for Network Security

	DDoS	HTTP	S-TCP	S-UDP	Flooding
CART	0.745	0.856	0.909	0.923	0.928
Naïve Bayes	0.730	0.655	0.897	0.933	0.917
MLP	0.907	0.993	0.979	0.983	0.989
SVM	0.883	0.992	0.941	0.995	0.968
kNN	0.720	0.936	0.936	0.924	0.944
Random Forest	0.827	0.905	0.941	0.913	0.930
Bagging Tree	0.823	0.908	0.911	0.915	0.921
AdaBoost Tree	0.892	0.991	0.923	0.920	0.927
logreg	0.926	0.956	0.952	0.980	0.987
MVaccuracy	0.924	0.992	0.971	0.993	**0.993**
MVuniform	0.923	0.991	0.970	0.992	0.991
CART	0.867	0.992	0.933	0.985	0.954
GML	**0.935**	**0.998**	**0.983**	**0.997**	**0.993**

- We take the **Area Under the ROC Curve** (AUC) as **benchmarking metric**
- **SLs performance increase is higher when base learners perform worse**
- Even if slightly, the **GML model systematically outperforms other models**

Benchmark for Anomaly Detection

	E1	E2	E3
CART	0.993	0.873	0.978
Naïve Bayes	0.956	0.861	0.959
MLP	0.997	0.944	0.996
SVM	0.996	0.944	0.995
kNN	0.995	0.859	0.963
Random Forest	**0.999**	0.876	0.993
Bagging Tree	0.996	0.885	0.983
AdaBoost Tree	0.998	0.945	0.995
logreg	**0.999**	0.952	0.996
MVaccuracy	**0.999**	0.948	0.996
MVuniform	**0.999**	0.945	0.996
CART	0.997	0.924	0.994
GML	**0.999**	**0.963**	**0.997**

- Similar observations are drawn from the AD benchmark
- **Anomalies E1 and E3 are easier to detect,** and **base learners** provide already **very accurate** results
- **E2 anomalies are stealthier** (long duration, small volume), and ***GML provides a clear performance increase***

Full Benchmark in multiple NMA Problems

	AD	NS	QoE-P	QoE-M	PPC	ALL
CART	0.948 (**3.9%**)	0.872 (**11.1%**)	0.956 (**3.7%**)	0.952 (**4.4%**)	0.966 (**1.9%**)	0.935 (**5.4%**)
Naïve Bayes	0.925 (**6.2%**)	0.826 (**15.8%**)	0.752 (**24.2%**)	0.754 (**24.3%**)	0.924 (**6.3%**)	0.819 (**17.1%**)
MLP	0.979 (**0.7%**)	0.970 (**1.1%**)	0.887 (**10.7%**)	0.882 (**11.5%**)	0.964 (**2.1%**)	0.929 (**6.0%**)
SVM	0.978 (**0.8%**)	0.955 (**2.6%**)	0.786 (**20.8%**)	0.790 (**20.7%**)	0.886 (**10.1%**)	0.869 (**12.1%**)
kNN	0.939 (**4.8%**)	0.892 (**9.1%**)	0.788 (**20.6%**)	0.793 (**20.4%**)	0.920 (**6.7%**)	0.854 (**13.6%**)
Random Forest	0.956 (**3.1%**)	0.903 (**7.9%**)	0.983 (**1%**)	0.978 (**1.8%**)	0.969 (**1.6%**)	0.957 (**3.2%**)
Bagging Tree	0.954 (**3.2%**)	0.895 (**8.7%**)	0.976 (**1.7%**)	0.975 (**2.1%**)	0.973 (**1.3%**)	0.953 (**3.6%**)
AdaBoost Tree	0.979 (**0.7%**)	0.930 (**5.2%**)	0.982 (**1.1%**)	0.984 (**1.2%**)	0.875 (**11.2%**)	0.954 (**3.5%**)
logreg	0.982 (**0.4%**)	0.960 (**2.1%**)	0.981 (**1.1%**)	0.978 (**1.9%**)	0.941 (**4.5%**)	0.970 (**1.9%**)
MVaccuracy	0.981 (**0.5%**)	0.974 (**0.7%**)	0.984 (**0.9%**)	0.991 (**0.6%**)	0.972 (**1.3%**)	0.981 (**0.8%**)
MVuniform	0.980 (**0.6%**)	0.973 (**0.8%**)	0.980 (**1.3%**)	0.984 (**1.2%**)	0.980 (**0.5%**)	0.979 (**1.0%**)
CART	0.971 (**1.5%**)	0.946 (**3.6%**)	0.956 (**3.6%**)	0.960 (**3.6%**)	0.968 (**1.8%**)	0.959 (**3.0%**)
GML	**0.986**	**0.981**	**0.993**	**0.996**	**0.985**	**0.989**

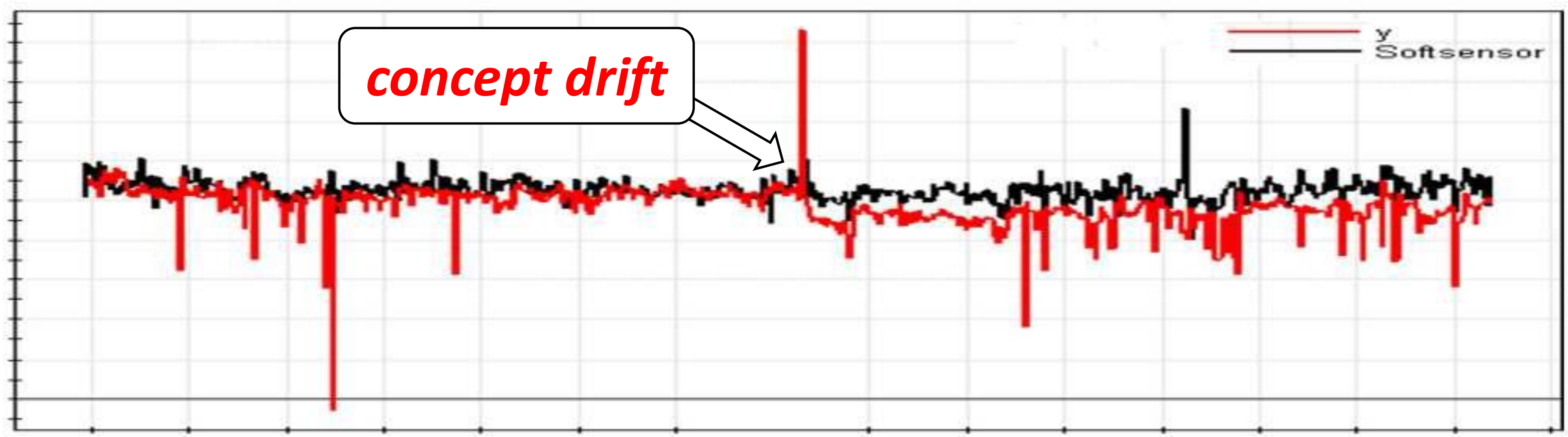
- *GML* does not only **outperforms the most accurate first level learners**…
- …but **also outperforms other ensemble-learning models** based on bagging, boosting and stacking
- The **GML model performs the best for all scenarios**, suggesting a potentially good approach to go for by default in similar NMA problems

Organization of the Talk
Dealing with Some of these Challenges

- Deep Learning for Malware Detection – ***Avoid Feature Engineering***
- Generative Models for Anomaly Detection – ***Avoid Traffic Modeling***
- Explainable Artificial Intelligence (XAI) **–** ***Interpret Model Decisions***
- Super Learning for Network Security – ***Avoid Model Decision***
- **Adaptive/Stream Learning for NetSec** – ***Deal with Concept Drifts***

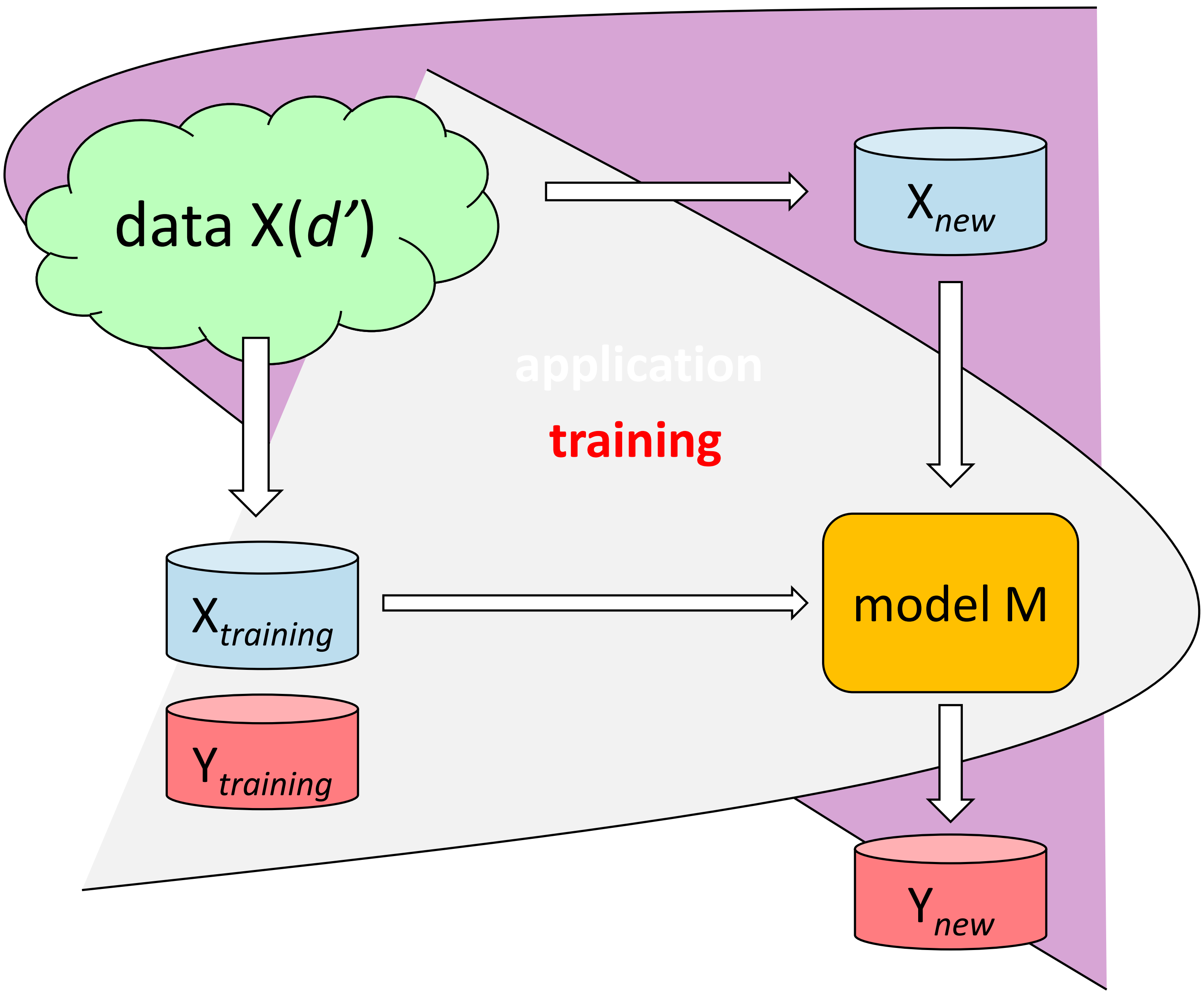
Adaptive or Stream-based Learning (*credits to Albert Bifet*)

- Let us go a bit deeper into the problem of **concept drift in supervised learning**
- And overview the main principles **how to deal with concept drift**



- ***Concept Drift (non-stationarity)***: the **statistical properties** defining the relationships between **input data and output target** **change over time**.
- This causes problems because the **predictions become less accurate as time passes**.

Concept Drift: a Trap for (off-line) Supervised Learning



training: learn a mapping function
$\mathbf{Y}_{training} = \mathbf{M}\ (\mathbf{X}_{training})$

application: use learnt function/model on newly, unseen data
$\mathbf{Y}_{new} = \mathbf{M}\ (\mathbf{X}_{new})$

…but what happens if/when $\mathbf{X}_{new}$ is **derived from a different distribution** $d' \neq d$?

(off-line) Supervised Learning under Concept Drifts

- Detection of **network attacks** in MAWI – WIDE network
- 10-fold cross-validation, **high detection performance with low FPR**…

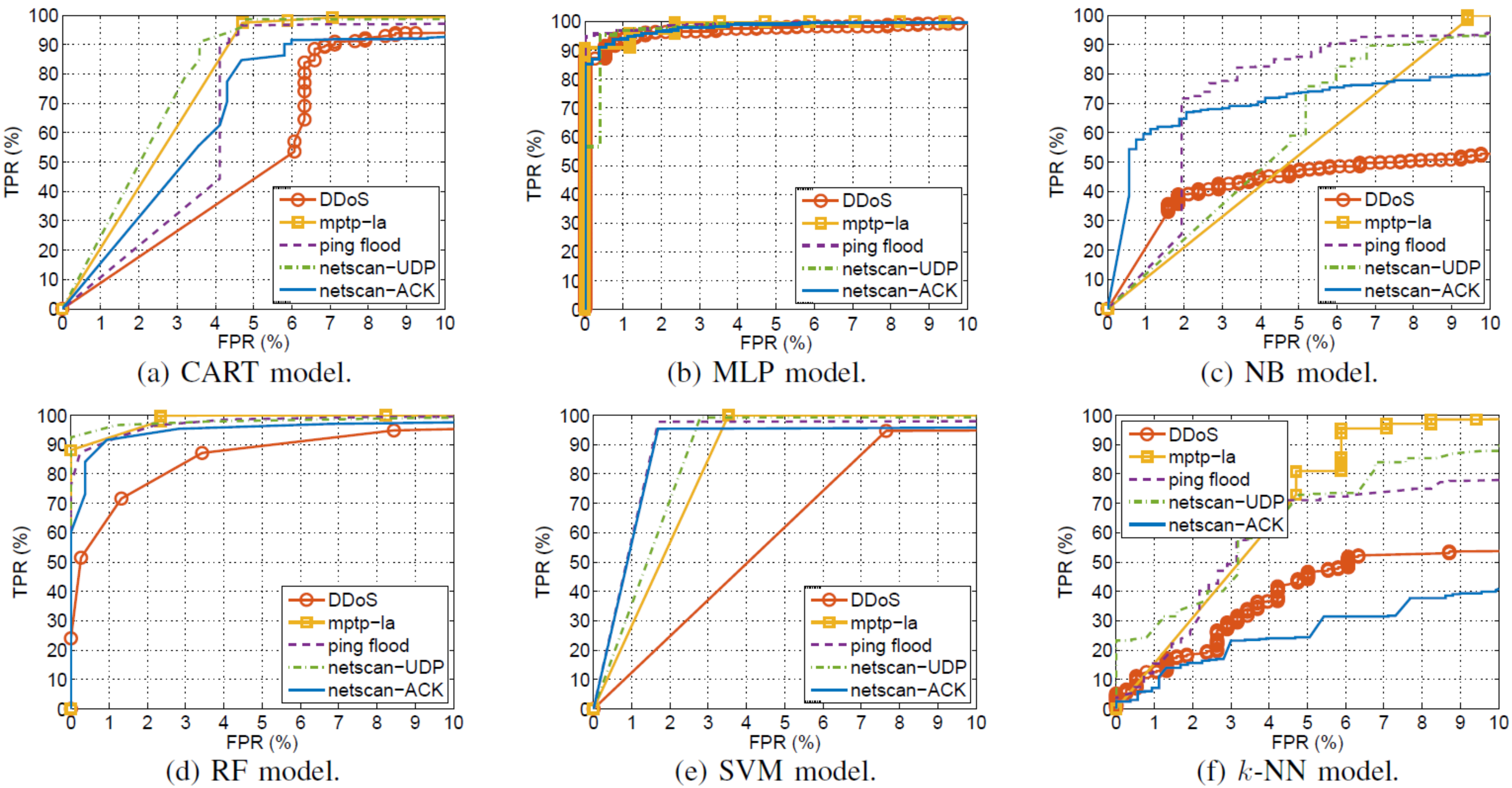
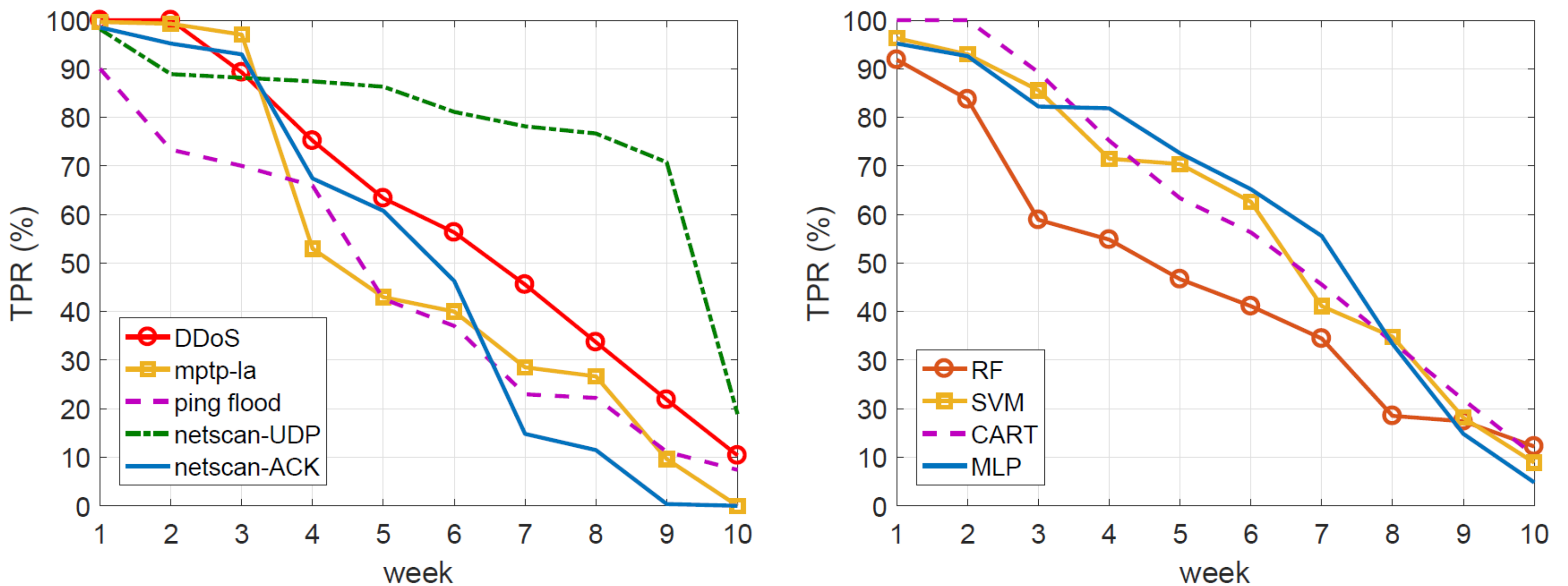


Figure 1: Detection performance (ROC curves) achieved by the different models for detection of network attacks.

(off-line) Supervised Learning under Concept Drifts

- …accuracy remains high for the first 3 weeks (**training on first 3 days**)…
- …but **models accuracy start to rapidly degrade over time**

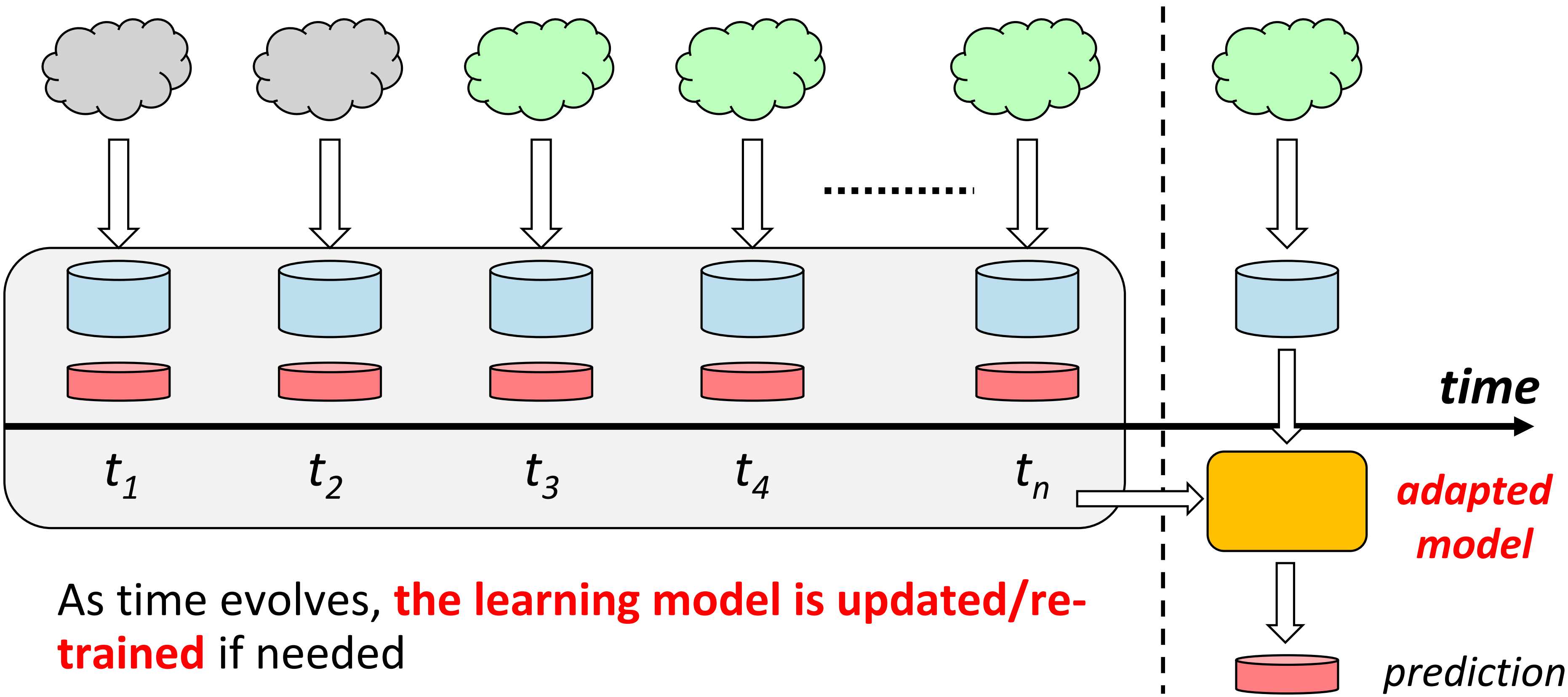


(a) CART model. (b) Detection of DDoS attacks.

Figure 2: Performance drift for the off-line trained models along time. Training is done on the first 3 days of data.

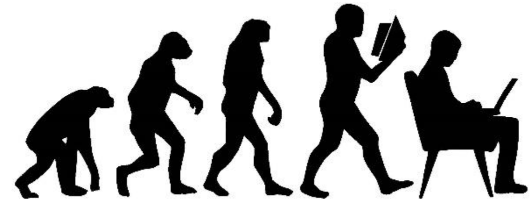
Learning in an Online Setting – Stream/Adaptive Learning

- In an **online setting**, **data** arrives continuously, as a **stream of samples**
- **Adaptive learning** consists of **learning from continuous data** in efficient way, using a **limited amount of memory**
- Adaptive learning approaches work in a **limited amount of time**

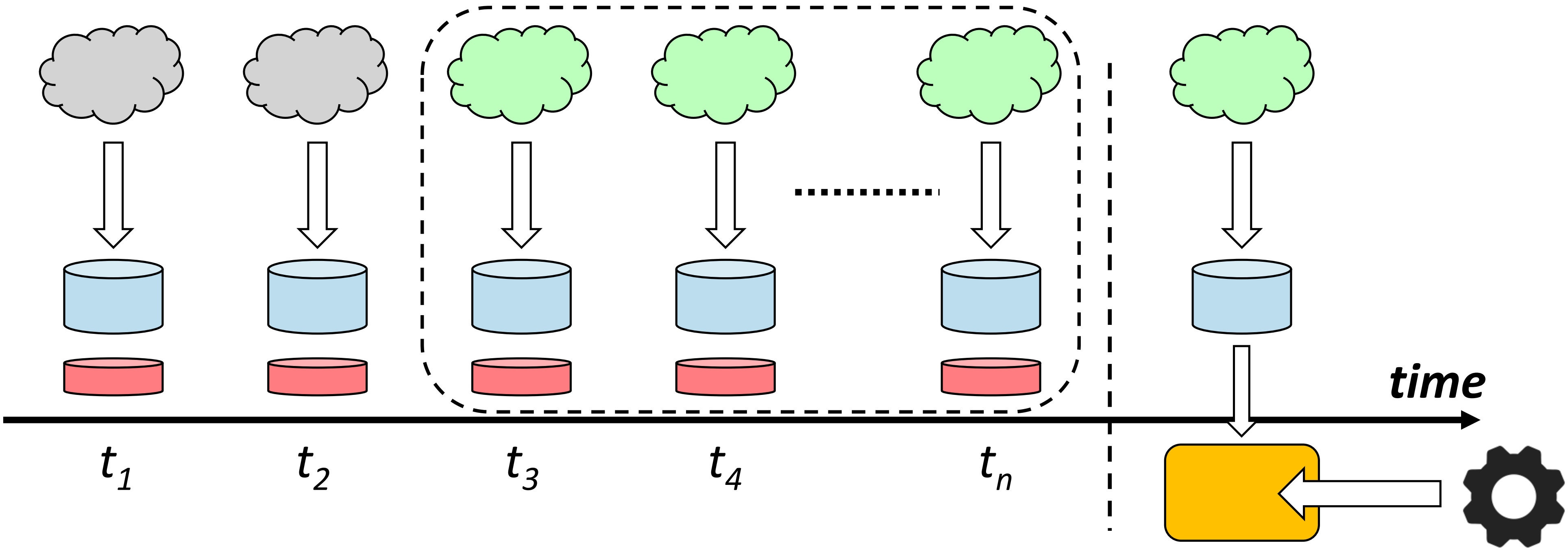


As time evolves, **the learning model is updated/re-trained** if needed

Adaptation Strategies



- Two main approaches for adaptation:
  - **re-train the model** by carefully selecting the ***best*** data
  - **adjust** the previously learnt **model incrementally**



Desired Properties of a System to Handle Concept Drift

- Adapt **fast** to concept drift

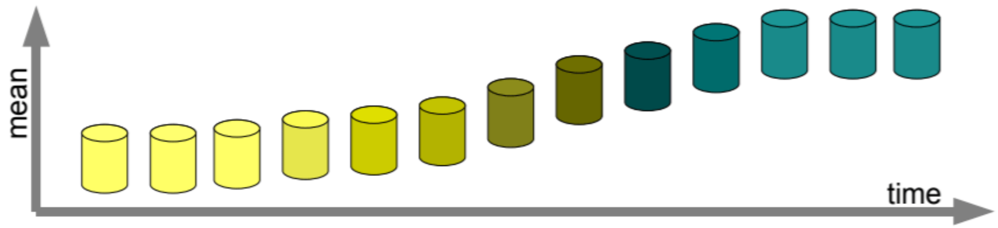
- **Robust** to noise, but **adaptive to changes**

- Capable to **deal with reoccurring contexts** (avoid catastrophic forgetting)

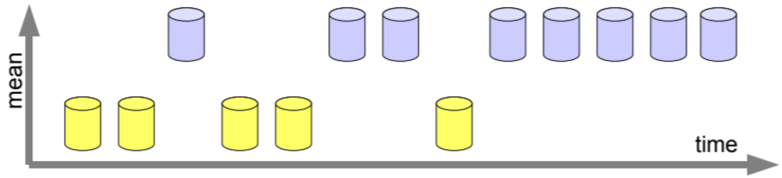
- **Use limited resources in terms of time and memory**

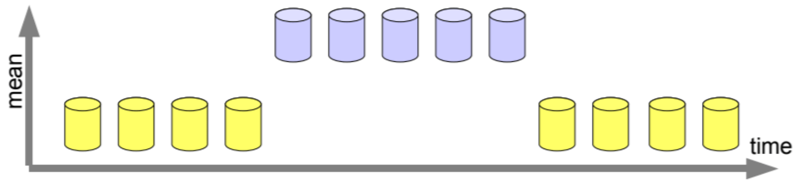
What types of Concept Drift can we get?

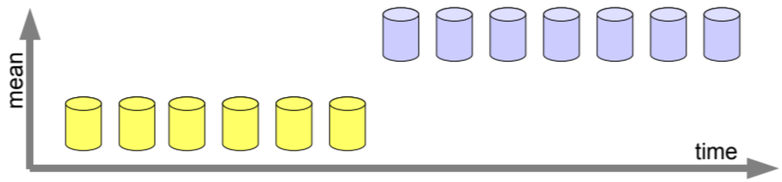
- The **change** to the data **could take any form**
- It is **conceptually easier** to consider the case where there is some **temporal consistency** to the change
- **Incremental drift:** one could assume that data collected within a specific time period show the same relationship and that this **changes smoothly over time**



- But of course, other types of changes may include:
  - A **gradual drift** over time
  - A **recurring** or cyclical **drift**
  - A sudden or **abrupt drift**







Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

strategy

memory

change detection and follow up
triggering

adapt at every step
evolving

reactive forgetting
single model

ensemble
maintain memory

Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

strategy

memory

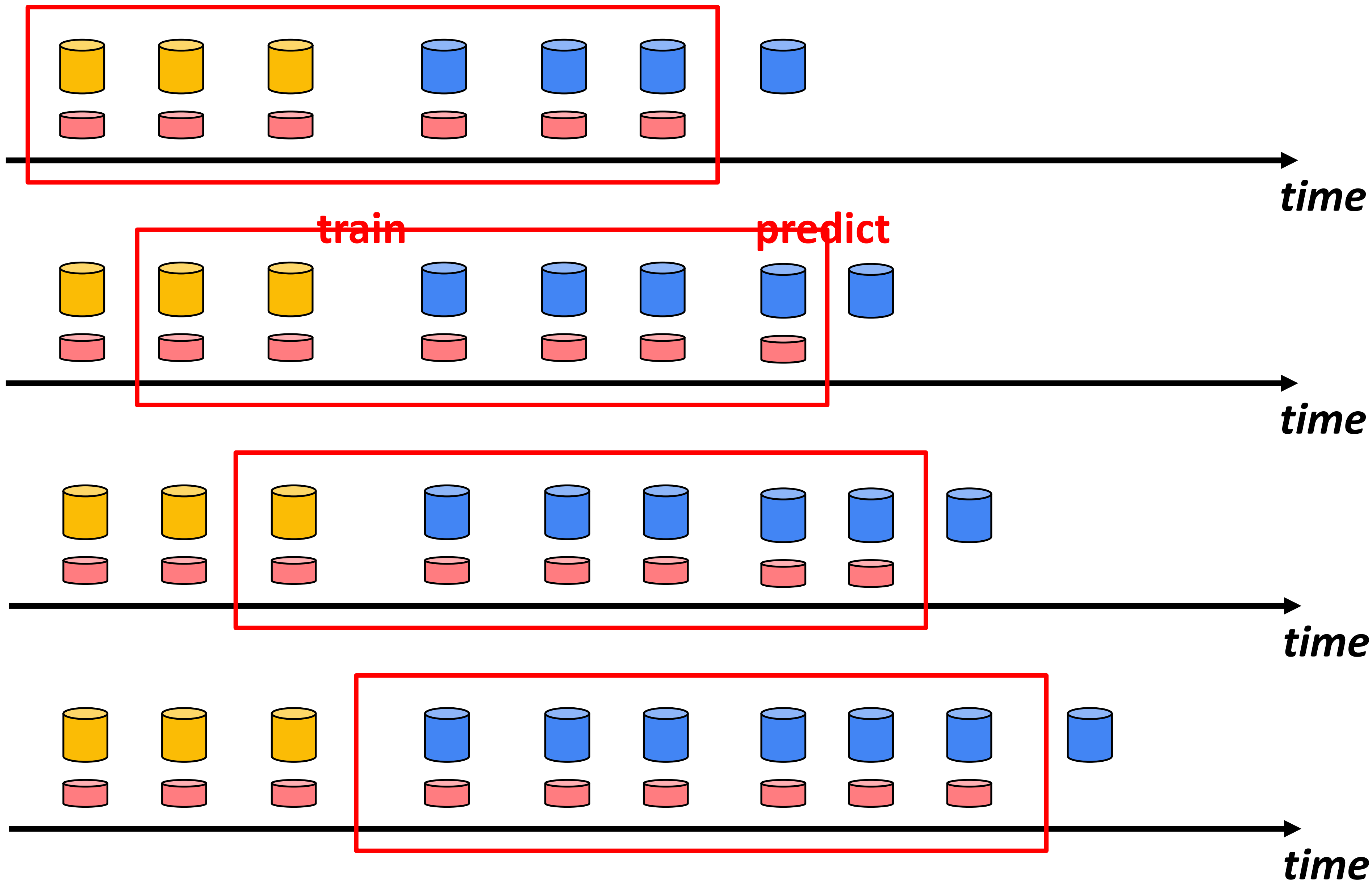
	triggering	evolving
single model		
ensemble		

Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

strategy memory	triggering	evolving
single model		**forgetting** • forget old data • re-train at fixed rate • fixed windows • instance weighting
ensemble		

Fixed-size Training Window

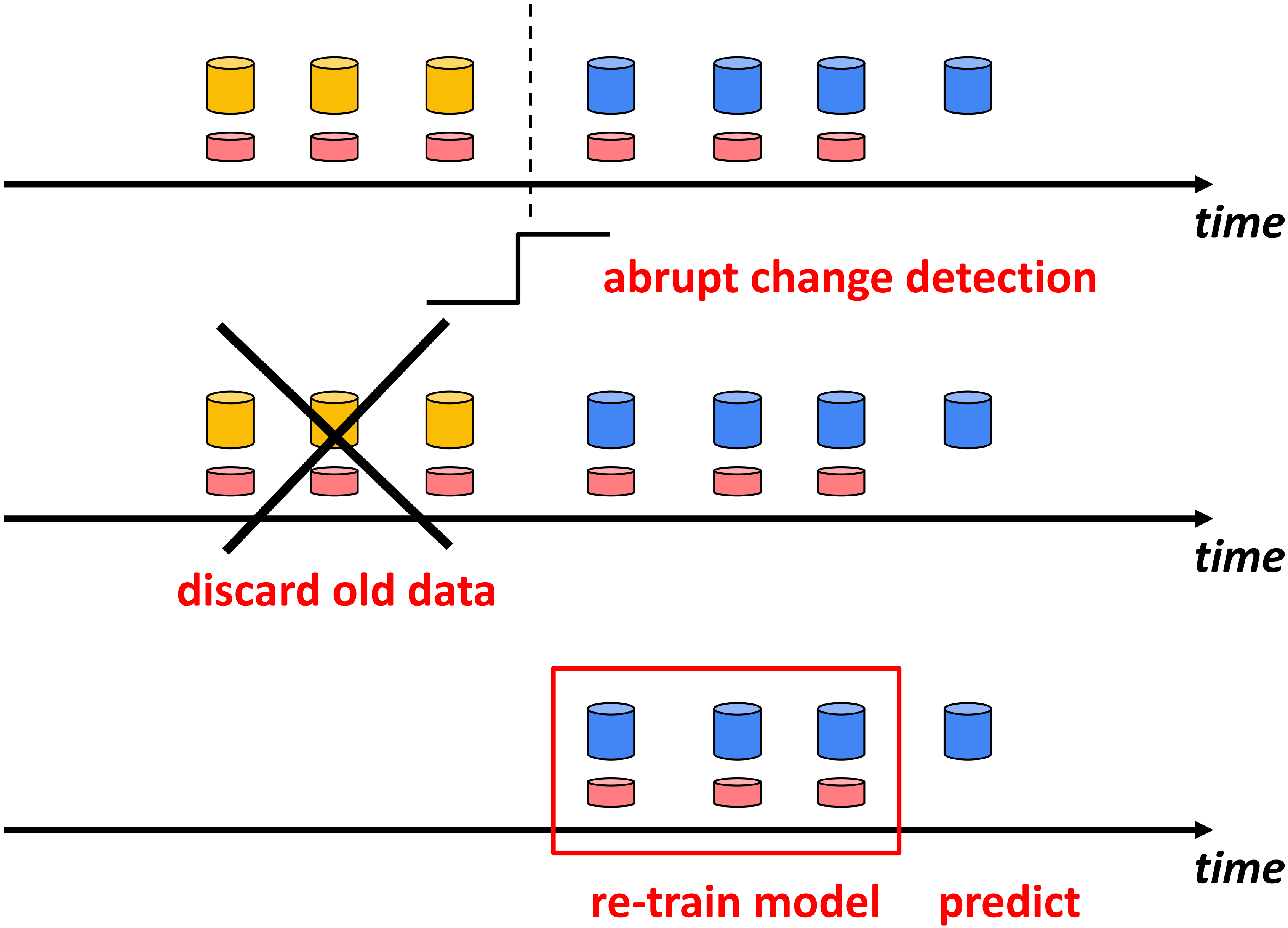


Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

memory \ strategy	triggering	evolving
single model	**detectors** • **detect a change and discard the past** • **variable windows**	
ensemble		

Variable Training Window, Change Detection and Cut

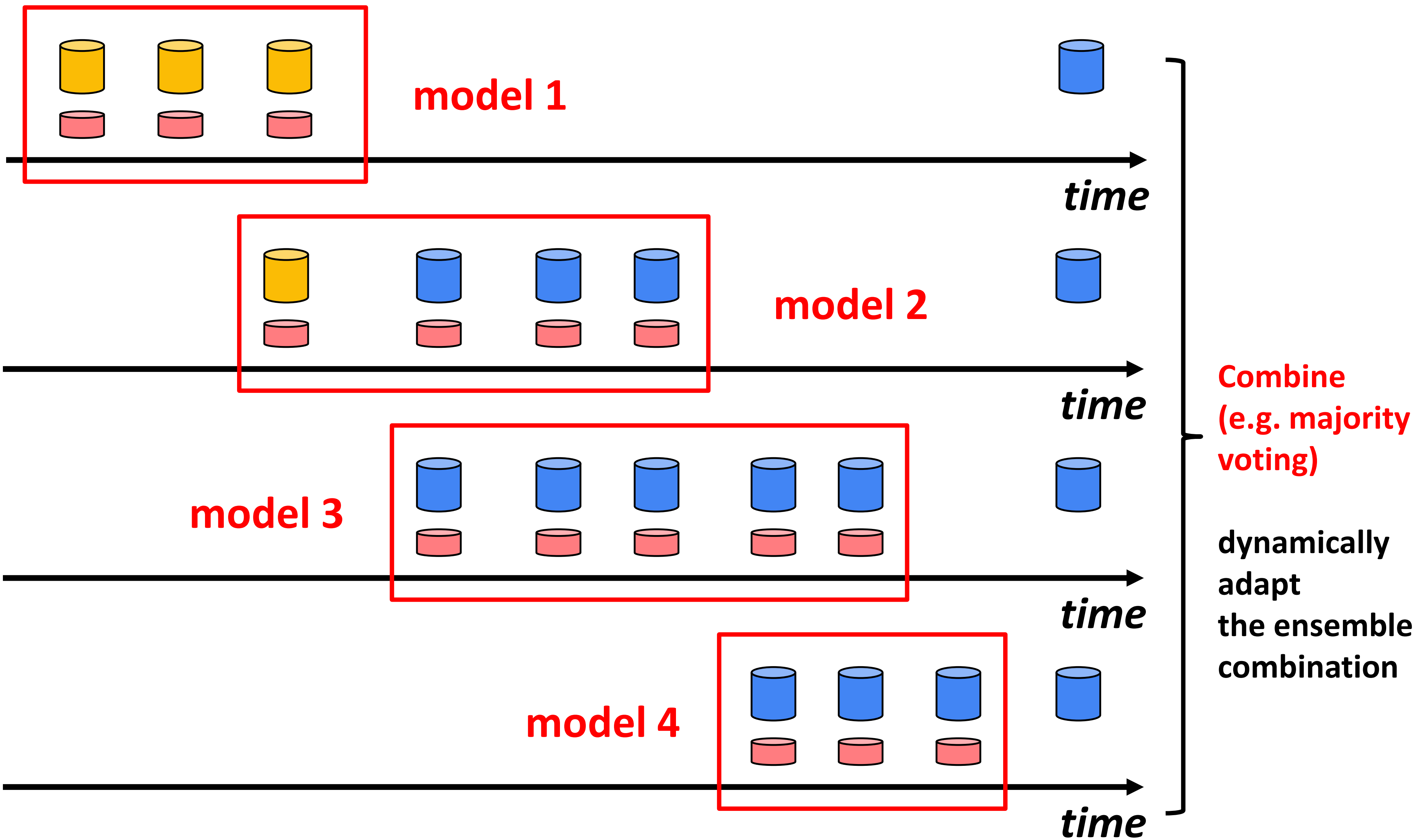


Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

strategy / memory	triggering	evolving
single model		
ensemble		**dynamic ensemble** • **build many models** • **dynamically combine** • **dynamic combination rules**

Dynamic Ensemble Learning

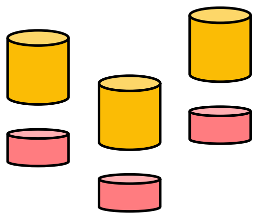


Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

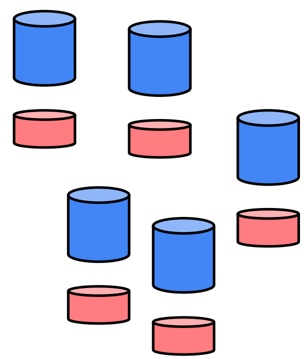
memory \ strategy	triggering	evolving
single model		
ensemble	**contextual** • build many models • switch among them based on input • meta–learning	

Contextual (Meta) Approaches

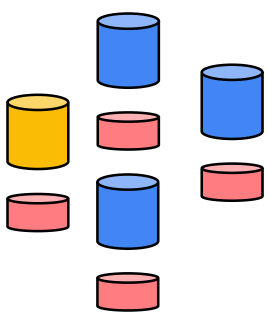


set 1 → model 1

partition training data to build multiple models

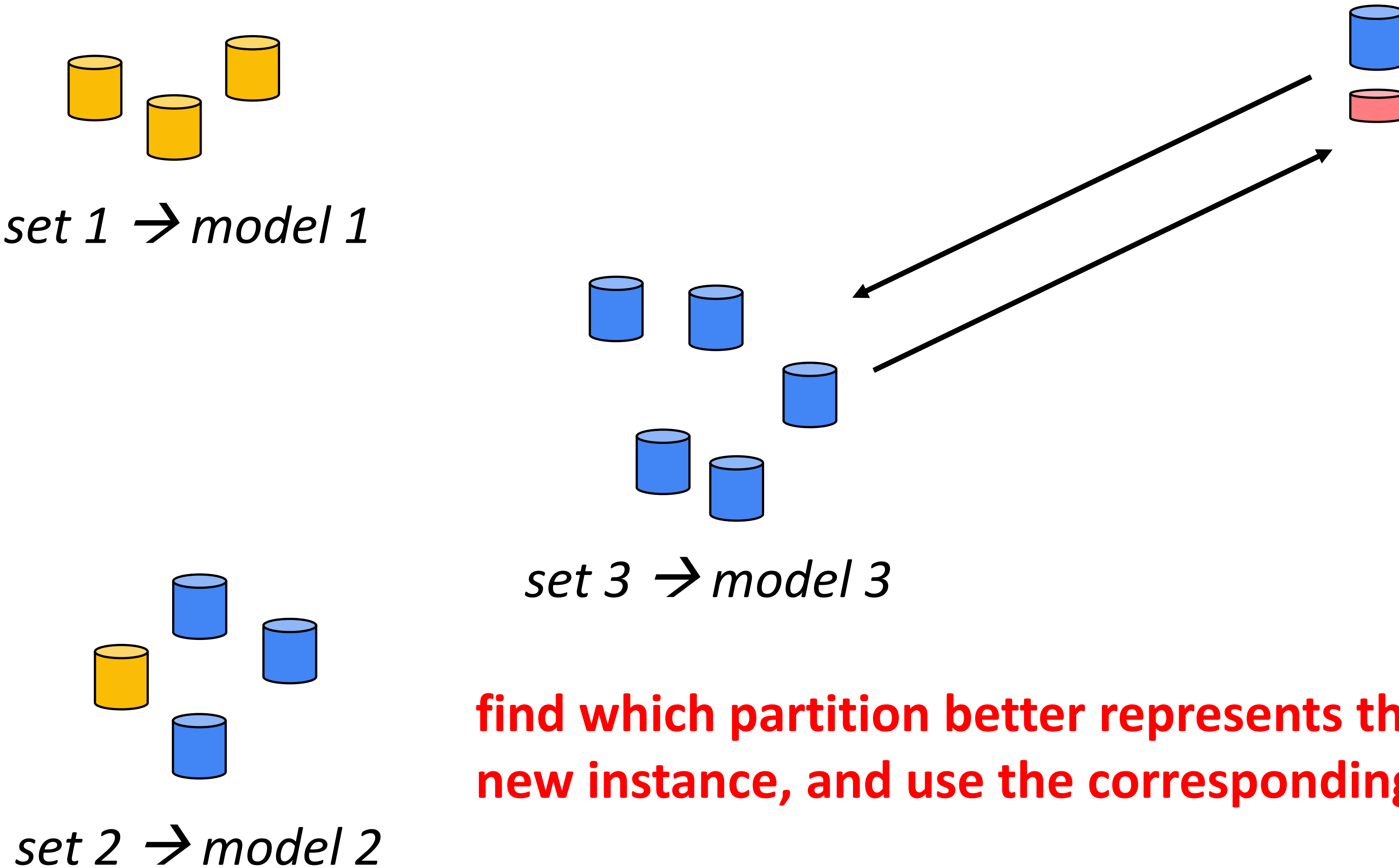


set 3 → model 3



set 2 → model 2

Contextual (Meta) Approaches



find which partition better represents the new instance, and use the corresponding model

Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

memory \ strategy	triggering	evolving
single model	**detectors** • detect a change and discard the past • variable windows	**forgetting** • forget old data • re-train at fixed rate • fixed windows • instance weighting
ensemble	**contextual** • build many models • switch among them based on input • meta–learning	**dynamic ensemble** • build many models • dynamically combine • dynamic combination rules

Adaptation Strategies to Concept Drift

- A taxonomy of approaches (***A. Bifet, J. Gama***)

memory \ strategy	triggering	evolving
single model	**detectors** **abrupt drift**	**forgetting** **abrupt drift**
ensemble	**contextual** **recurring drift**	**dynamic ensemble** **gradual drift**

Adaptive/Stream Learning Models for NetSec

- Implement an adaptive approach using ***single models*** and a ***change-detection*** algorithm to detect concept drifts

- Take ADWIN (**Adaptive WINdowing**) to detect changes

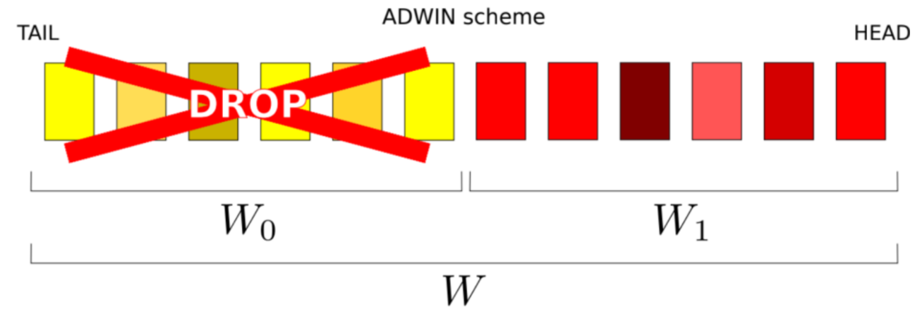
- ADWIN automatically **grows the learning window** when **no change** is apparent, and ***shrinks it when concept drifts*** are detected

- **Properties:** automatically adjusts its **window size** to the **optimum** balance point **between reaction time** and **small variance**

Adaptive WINdowing algorithm

The idea of ADWIN is straightforward:

- it keeps a sliding window W with the most recently observed data
- whenever two ***large enough*** sub-windows of W exhibit ***distinct enough*** averages, **the older portion of the window is dropped**.

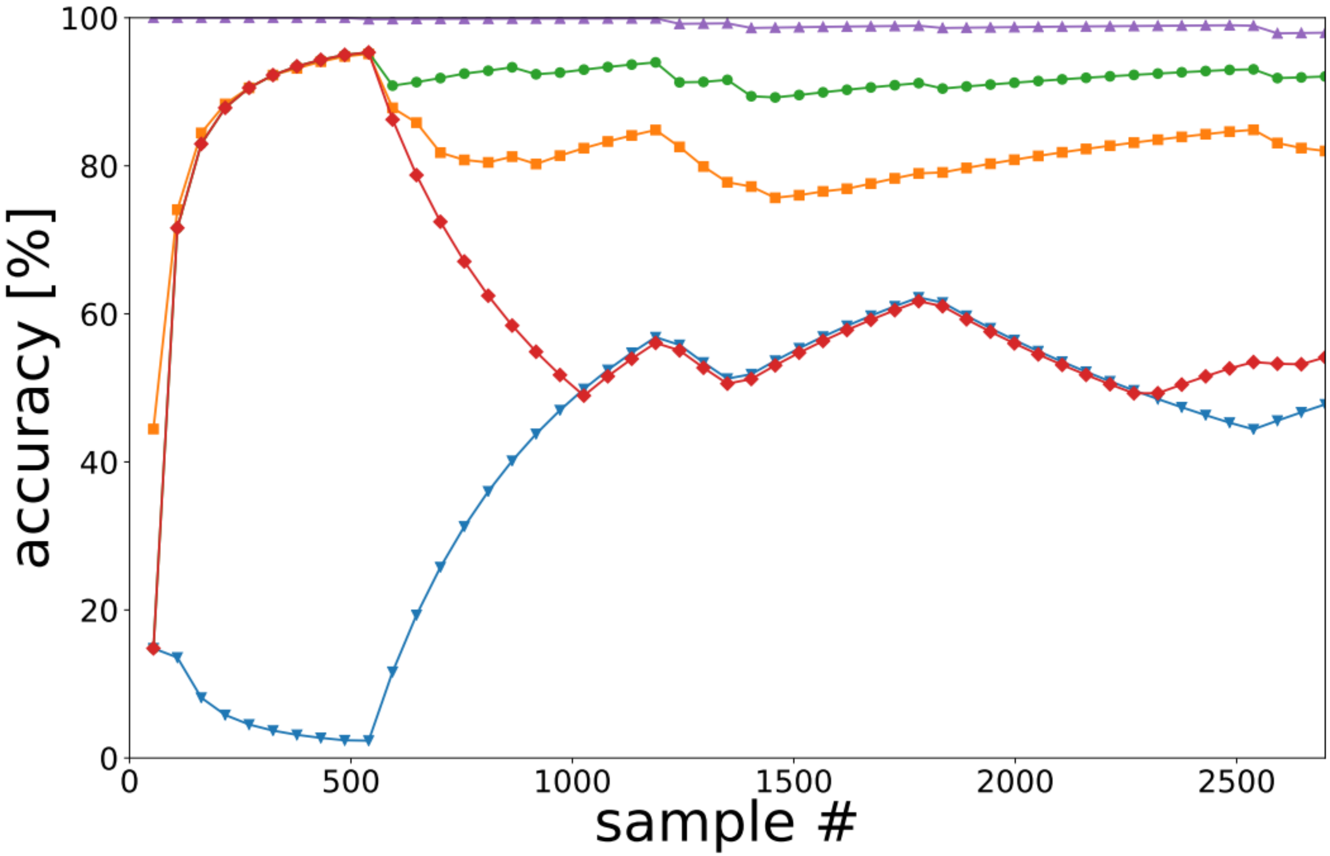


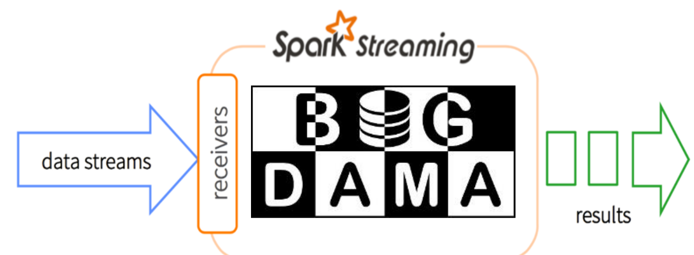
initialize window W
for each $t > 0$ **do**
  $W \leftarrow W \cup \{x_t\}$ (add x_t to the head of W)
  repeatdrop instances from the tail of W
  until $\|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}\| \geq \epsilon$ for every split of $W = W_0 \cdot W_1$
  return $\hat{\mu}_W$
end for

where $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$ are the averages of the instances in W_0 and W_1 respectively.

Adaptive/Stream Learning Models for NetSec

- Adaptive learning algorithms **trained on labelled data, using ADWIN**





Stream-based Learning Models Performance

- Multiple stream machine learning models, using ADWIN
- Detection accuracy, ***normalized to batch-based algorithms*** performance

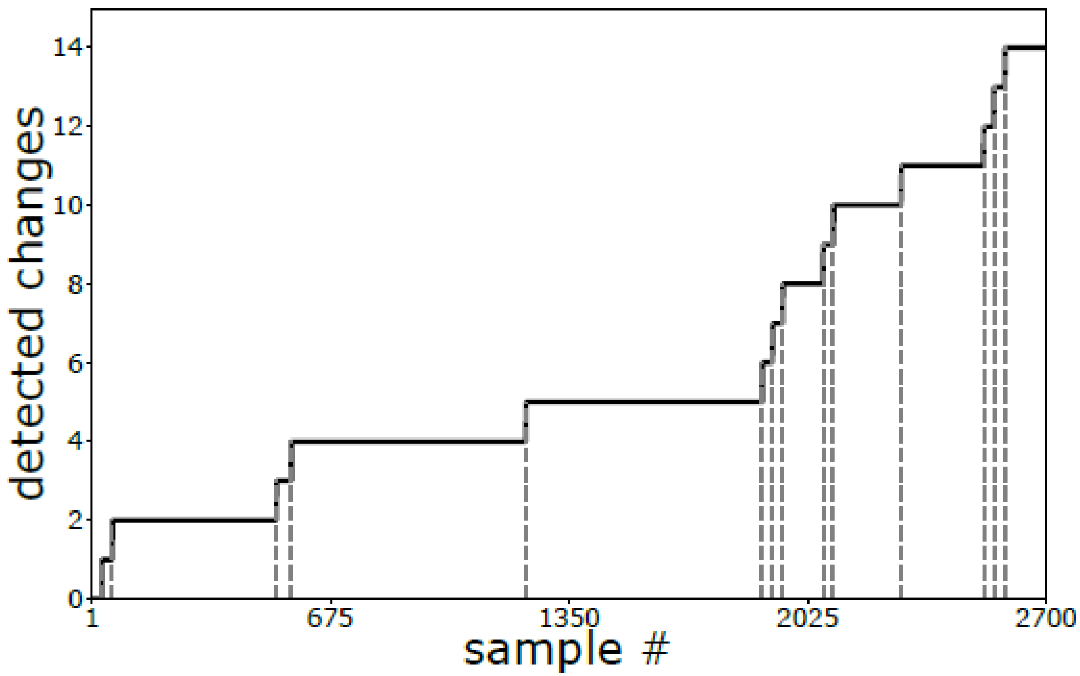
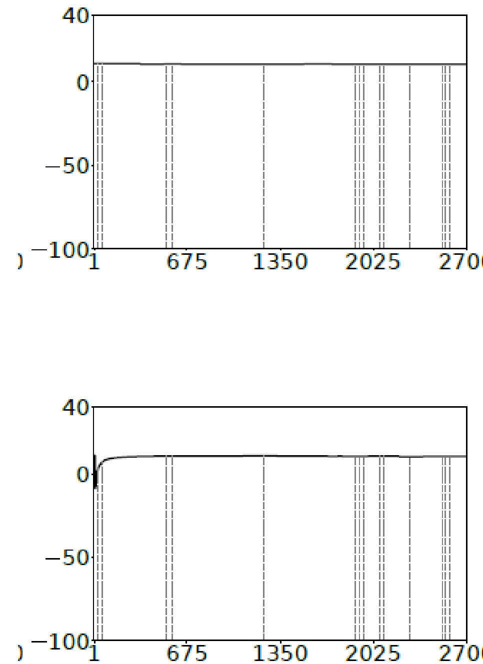


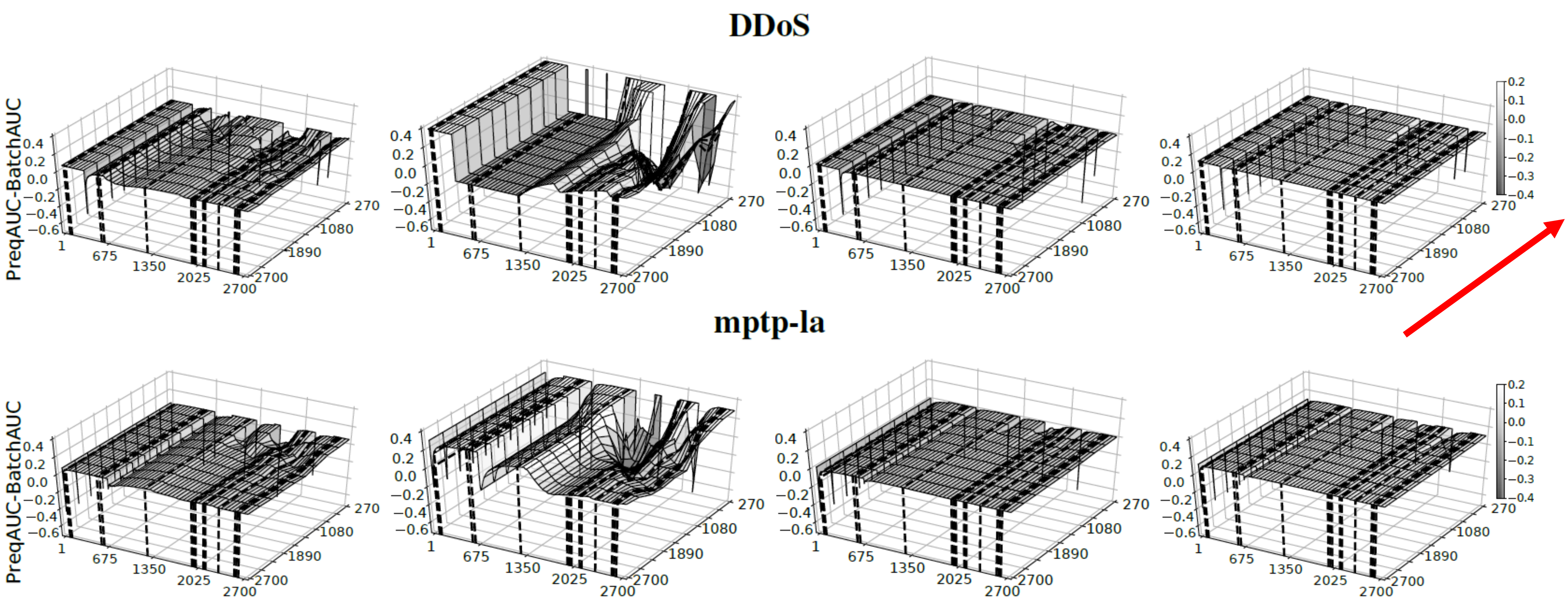
Figure 1: *Page-Hinkley Concept Drift Detection.* Changes in the dataset distribution detected by the Page-Hinkley test. Detected changes are marked with dashed lines.



(d) *SGD*

Stream-based Learning Models Performance

- Multiple stream machine learning models, using ***fixed windowing***
- AUC (ROC curve), ***normalized to batch-based algorithms*** performance
- **Different window sizes tested**



(a) *k-NN* (b) *HAT* (c) *ARF* (d) *SGD*

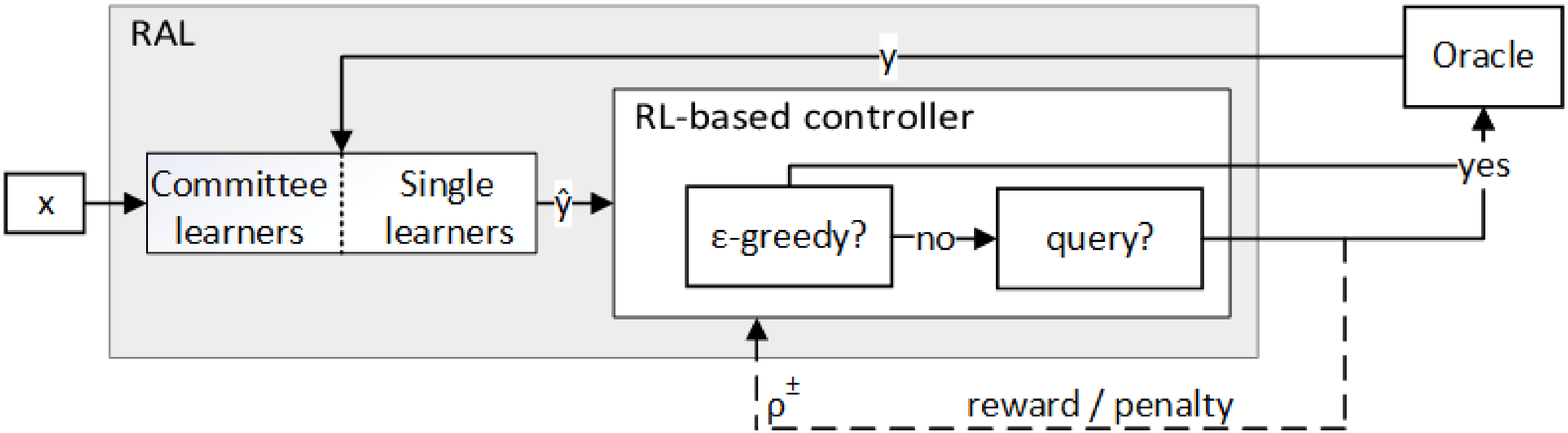
Improving Stream-based Active Learning by Reinforcement (RAL)

- How do we deal with the **limited amount of labeled data?**
- **Active Learning (AL):** aims at labelling only the most informative samples
- **AL** can be applied to the **streaming scenario**, to complement previous approaches and **reduce the amount of labeled data**

- RAL – improves stream-based AL by Reinforcement Learning (RL)
  - Standard AL bases its decisions based on ***model uncertainty***
  - **RAL permits to additionally learn in a feedback loop**, based on the **effectiveness of the requested labels**
  - **Reward** in case asking oracle was informative (models would have predicted wrong label)
  - **Penalty** otherwise

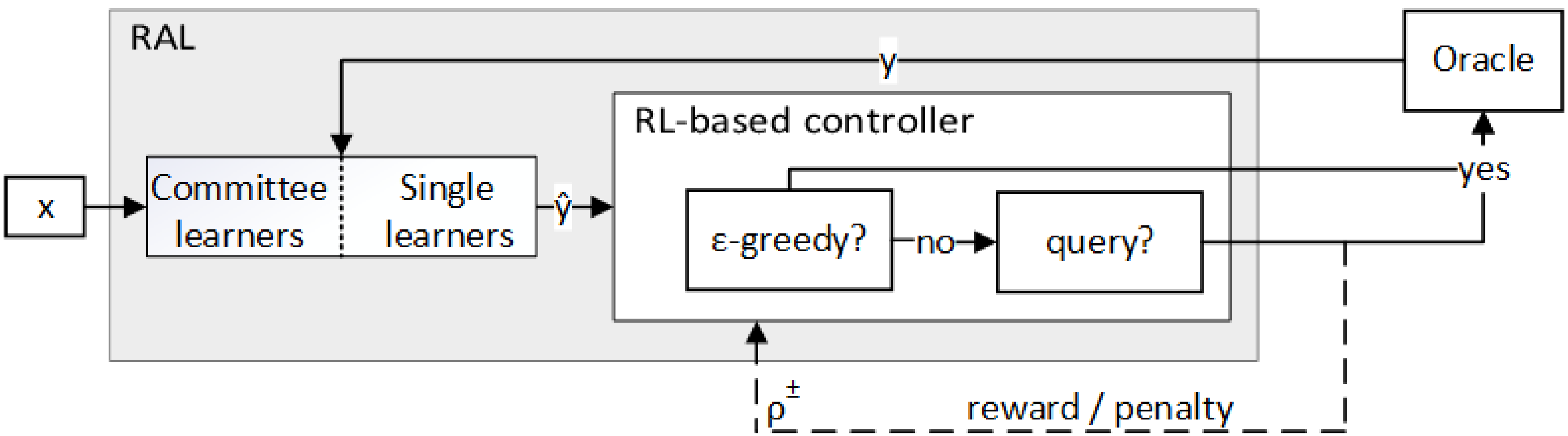
RAL Principles and Components

- RAL is based on an ensemble of models
- RAL makes use of **contextual-bandit algorithms** (EXP4) to tune the decision powers of the different models depending on their behavior
- RAL uses a **ε-greedy approach** to handle concept drift and **improve the exploration/exploitation trade-off**



RAL Principles and Components

- The **querying decision** (ask or not for a label) is taken **based on model prediction uncertainty** and a **threshold**
- Each algorithm in the ensemble (committee) gives its advice, based on its prediction uncertainty
- RAL takes into account the decisions of the members + their decision power
- Obtained **feedback influences the querying threshold**:
  - In case of **penalty**, the threshold decreases.....otherwise, it **slightly increases**



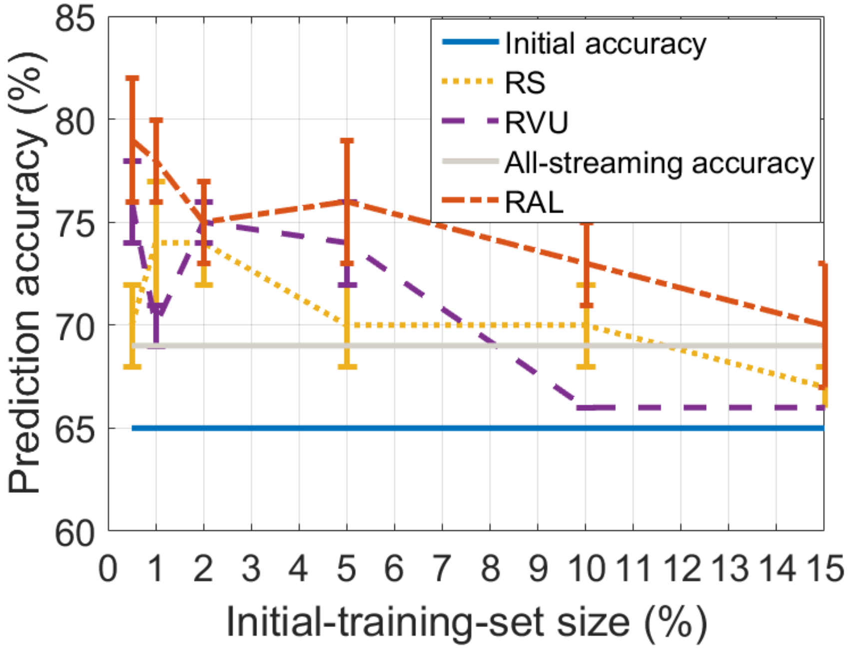
RAL Evaluation vs. State of the Art

- **RAL vs** RVU (**Randomized Variable Uncertainty**) and simple **random sampling** (RS)

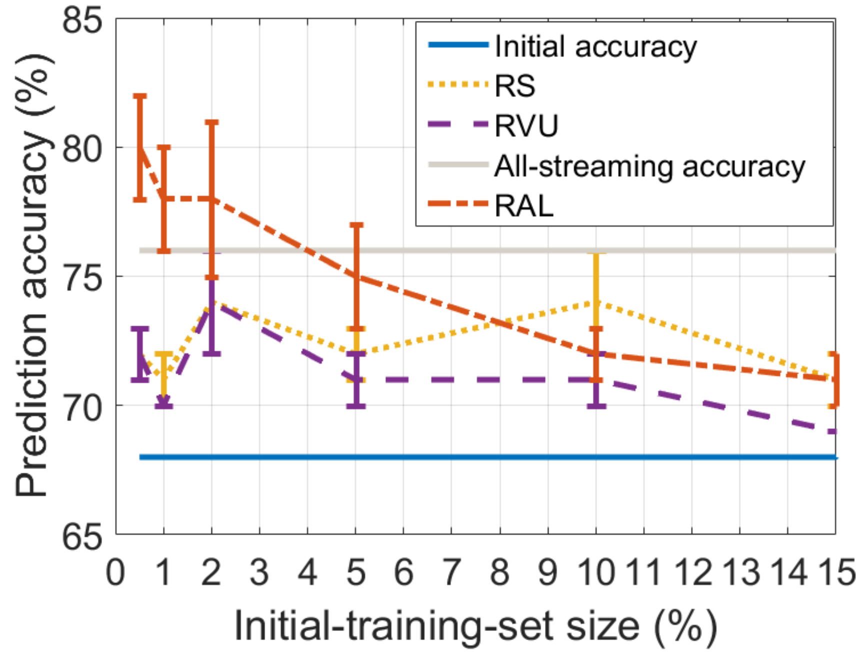
- Evaluation on data extracted from **MAWILab – in the wild network security**

- We divide each dataset into three consecutive parts:
  - **Initial training set** (variable size)
  - **Validation set** (last 30%), to evaluate the classifiers
  - **Streaming set** (remaining part of the dataset), for picking samples to learn from

RAL Evaluation vs. State of the Art – Prediction Accuracy

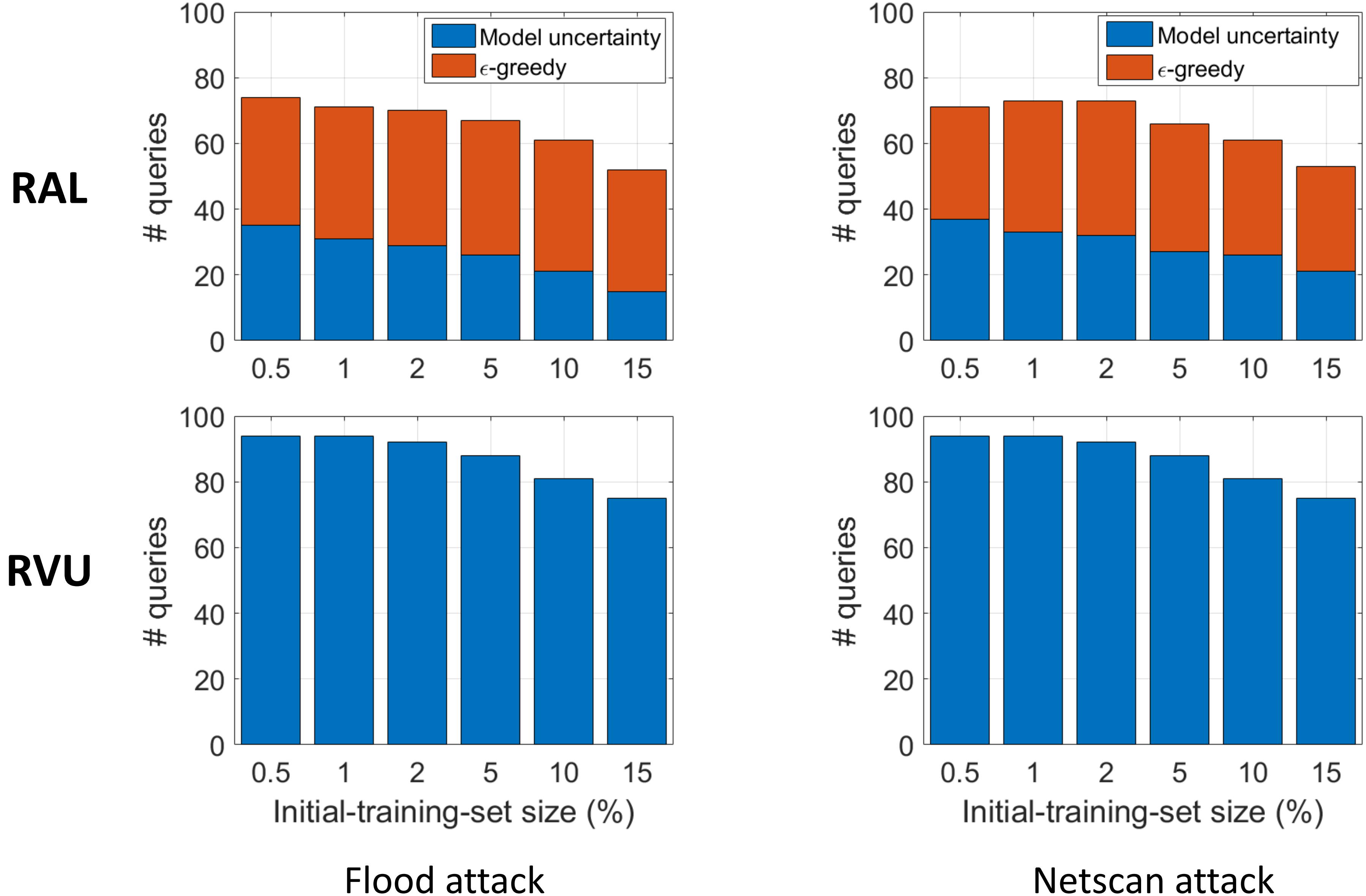


Flood attack



Netscan attack

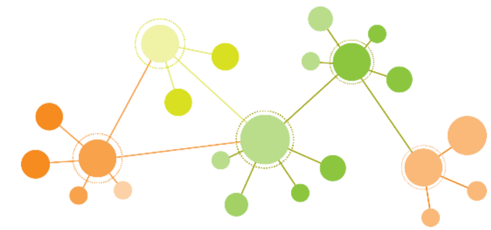
RAL Evaluation vs. State of the Art – Querying Cost



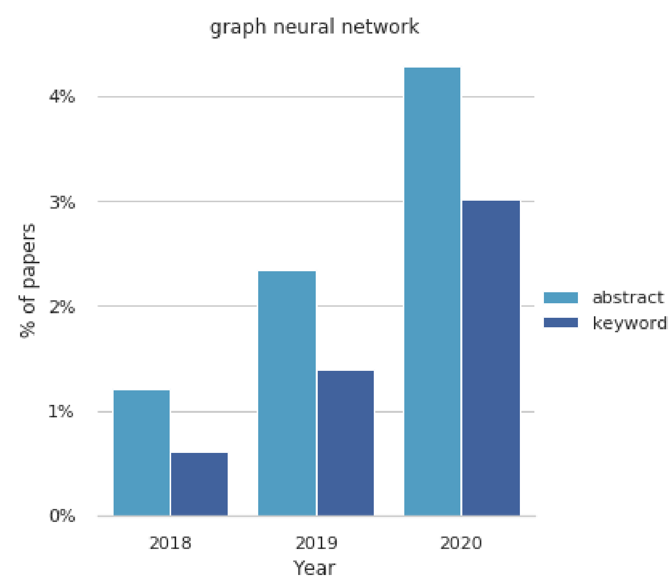
So What's Next?

- We're still far from ***making AI immediately applicable to Cybersecurity***
  - Limitations of learning process, data, models
  - Lack of generalization
  - Continual learning challenges – catastrophic forgetting and transfer
  - Lack of real knowledge generation – building simple mappings is *easy*
  - Portability of models to real deployments – *plug & play?*
- ***Effective Machine Learning*** – a mix of interesting challenges:
  - Transfer learning
  - Explainable AI (XAI)
  - Robust and adversarial learning (worst-case operation boundaries)
  - Formal methods in (and for) AI/ML
- And back right to the start: **the successful application of AI to network measurement problems is still on an early stage**

Graph Neural Networks (GNNs)



- In a nutshell: **deep learning** architecture for **graph-structured data**
- **Lots of domains where graph-structured data makes much more sense**: social networks, knowledge graphs, recommender systems, ***communication networks***
- Typical application of GNN: node classification → every node in the graph is associated with a label, and we want to predict the label of the nodes without ground-truth
- Have so far proved very powerful in modeling the dependencies between nodes in graph-like structures



- **About 4% of ICLR 2020 submitted papers using GNNs** (2585 submissions)
- **Graph Neural Networking Challenges 2020/2021**
  RouteNET: a GNN architecture to estimate per-source-destination performance metrics in communication networks

Thanks

Dr. Pedro Casas
Data Science & Artificial Intelligence
AIT Austrian Institute of Technology @Vienna

pedro.casas@ait.ac.at
http://pcasas.info