Continual Learning for Intrusion Detection Systems

Sai Prasath S (17CS02002)

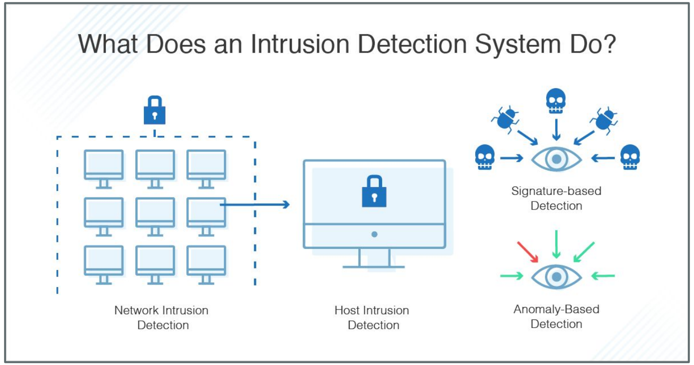
Dr Padmalochan Bera
IIT Bhubaneswar

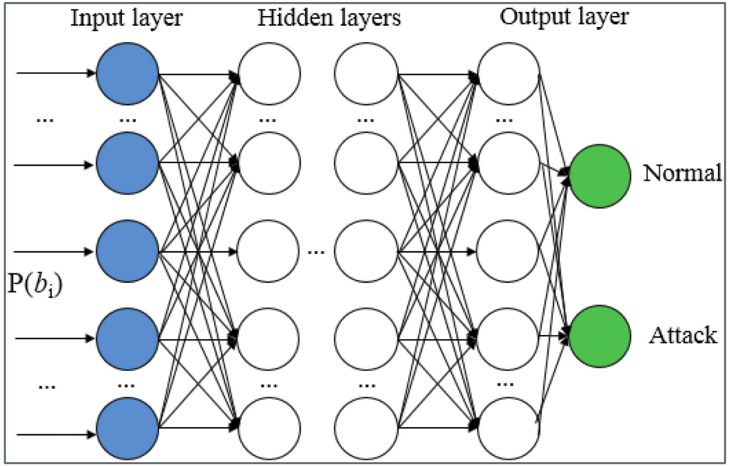
Contents:

1. What are Intrusion Detection Systems(IDS)?
2. Motivation
3. Problems with neural networks and proposed solutions
4. Analysis of Dataset Drift
5. Continual Learning Strategies

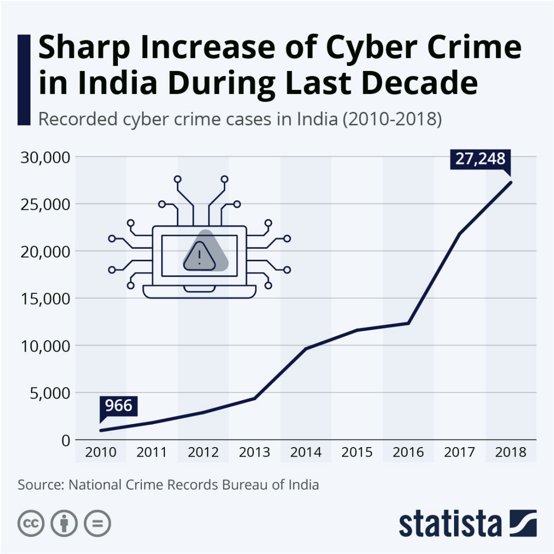
Intrusion Detection System:

1. Monitors the network, host, router etc. (any level)
2. Collect Data
3. Analyses the collected data
4. Detects any anomaly or suspicious activity





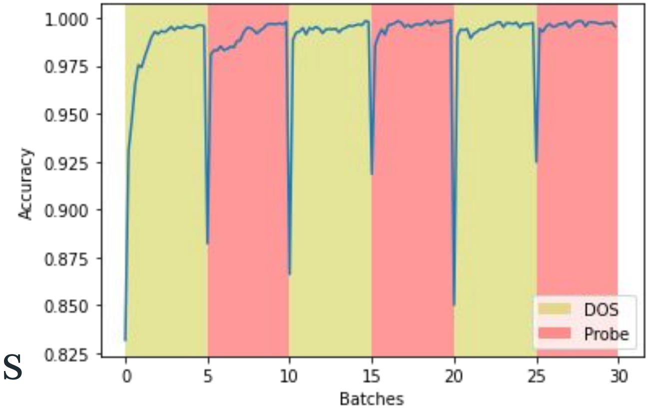
Motivation:



- New attacks everyday
  - US Pipelines Ransomware
- University Of Maryland:
  - Attack once in every 39 seconds
  - 64% of all medium scale companies
- Easy to attack, new devices, networks etc.
  - Cheap: $39 buy malware
  - Transfer money via cryptocurrency

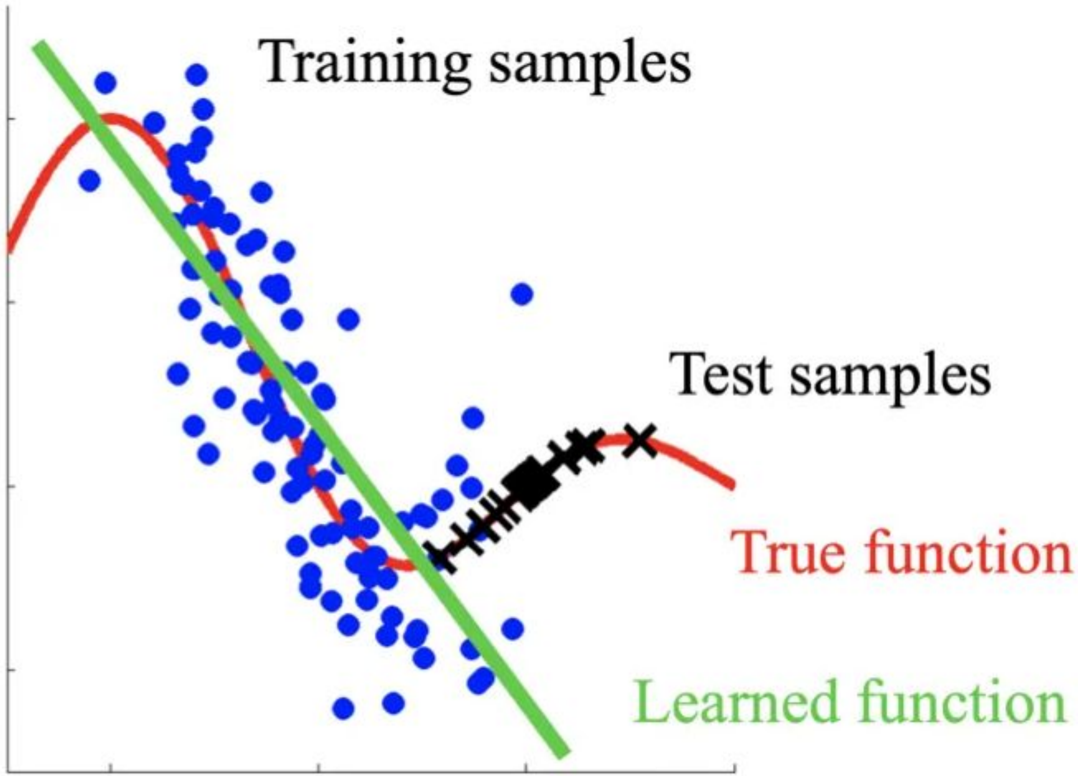
Continual Learning

- Learning something new → Difficult for NN
  - Catastrophic forgetting
  - Sequential Learning: Not effective
  - Solve these issues: **Continual Learning**
- Learn Sequentially in an incremental way
- When learning new knowledge, retain old ones



Why don't you just collect all data and train the model once again?

Computationally expensive, huge data storage required, not using the knowledge available
Ex: Amazon, Google
(100 million people * 10 mins/person * 60 seconds/minute * 10 packets/second) = 600 billion packets/day



Test Samples:

- New attack types
- Old attack shifted

Data distribution keeps shifting: Never constant

Detect the Shift in NSL-KDD dataset:

Dataset	Normal	DOS	Probe	U2R	R2L
Training	67,343	45,927	11,656	52	995
Test	9,711	7,458	2,241	67	2,887

Attack Type	Train Dataset	Test Dataset
Shared	99.29%	83.36%
Exclusive to Train	0.71%	0.0%
Exclusive to Test	0.0%	16.64%

Trainset: 23, Testset: 38
Pooled into 5 major classes
No of day 0 attacks in the test set: 17 (Not in train set)
Shared attacks: 21
Exclusive to trainset: 2

1. Classifier Bias:

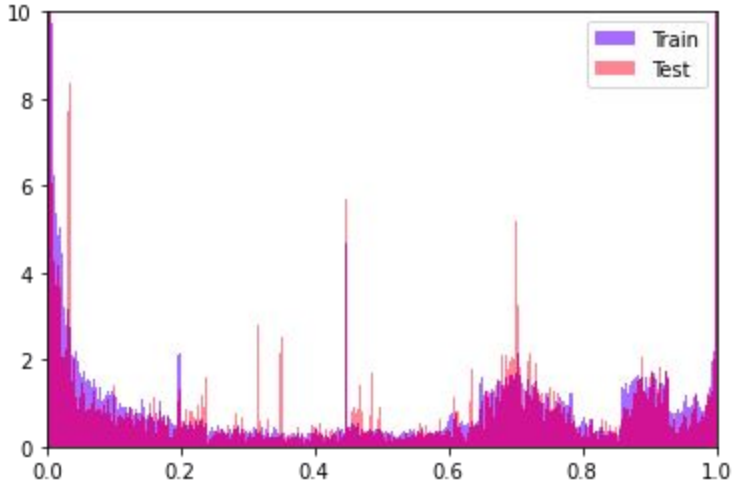
Model	Train Set	Test Set
Random Forest	99.67	77.22
DNN (20 epochs)	96.42	74.29

Is it overfitting? No
Random forests ----> Only 30 trees
DNN ----> Only 20 epochs

What if classification in test set is inherently difficult?

- Mix the train and test set, also shuffle
- 10-fold cross validation
- Accuracy: 99.49%, FPR: 0.39%
- High accuracy: no such difficulty

2. Histogram Overlap:

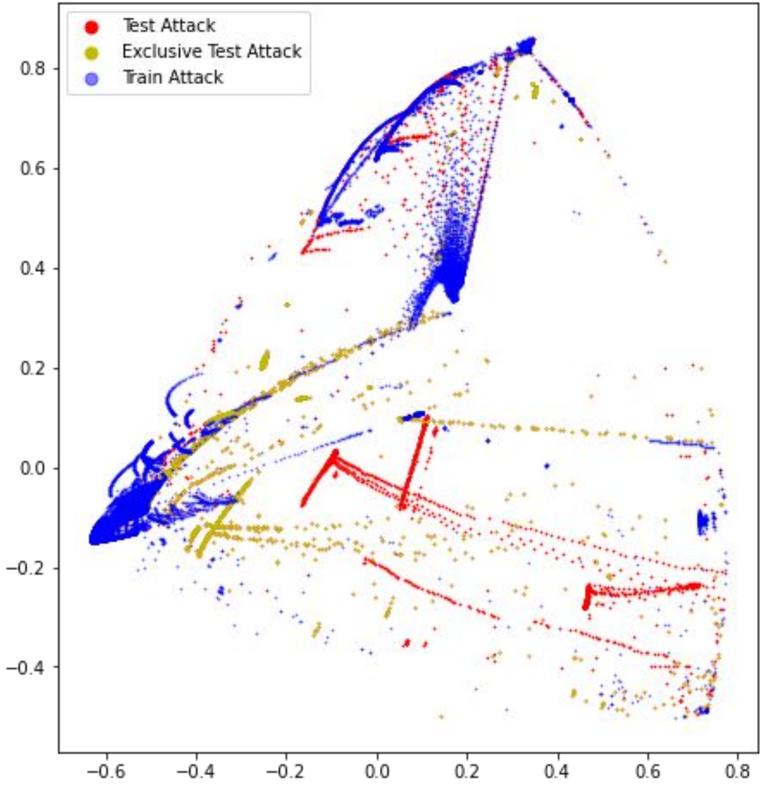


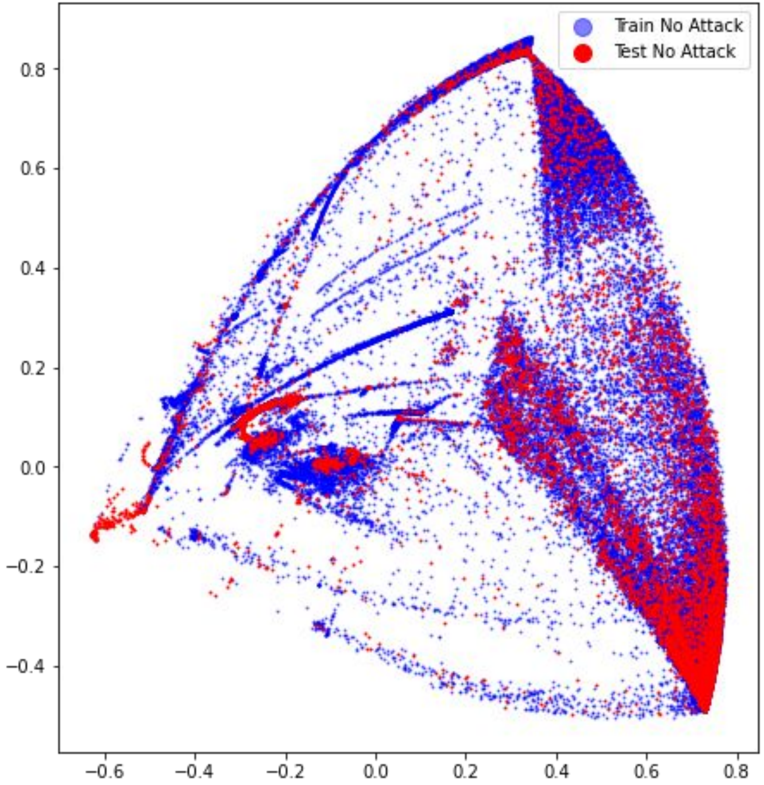
Features	Overlap (%)
dst_bytes	0.947
duration	0.953
service_pop_3	0.963
hot	0.978
dst_host_srv_count	0.979
dst_host_count	0.979
num_failed_logins	0.981
is_guest_login	0.981
service_ftp	0.984
srv_count	0.985

The non-parametric KS test also yields similar results by comparing the p-values and the KS-statistics.

All individual features across the train and test set → Similar distribution

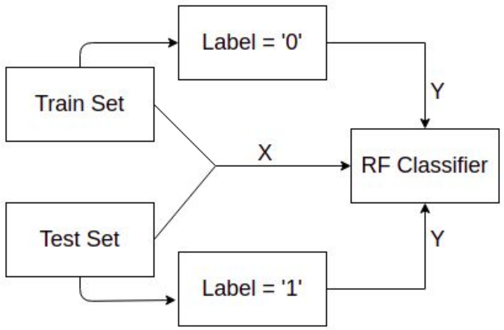
3. 2D visualization:





4. Discriminative Distance:

The RF classifier was able to successfully classify the train and the test set with a 10-fold cross validation accuracy of 91.28%.



Why?

Higher accuracy → Definite boundaries

Lower accuracy → Similar Distributions

Why so many analysis?

- Multiple papers of IDS using NSL KDD dataset
- Performance on the test set 80-85%
- No previous dataset drift analysis has been performed
- Proof: Real-world problem

How to solve this issue?

1. Reduce bias in dataset:

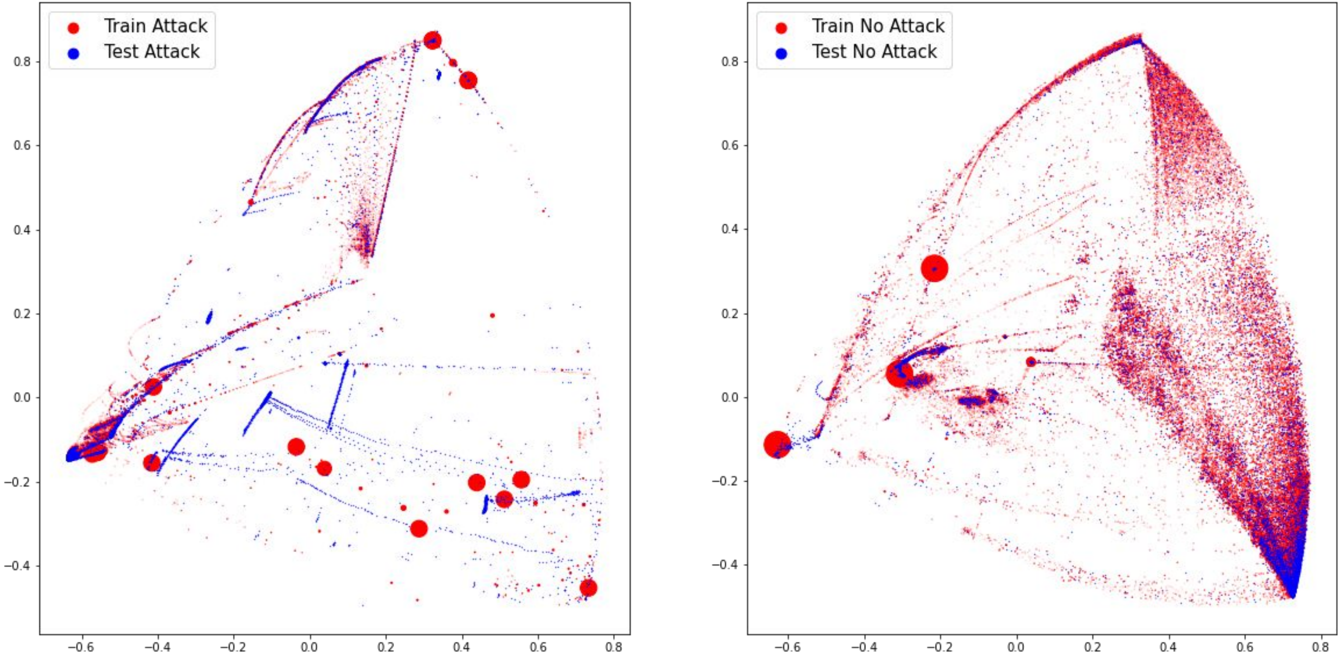
Some features might have drifted more than other, drop them.

Features	Overlap (%)
dst_bytes	0.947
duration	0.953
service_pop_3	0.963
hot	0.978
dst_host_srv_count	0.979
dst_host_count	0.979
num_failed_logins	0.981
is_guest_login	0.981
service_ftp	0.984
srv_count	0.985

Features	Train-Test	Train Binary	Test Binary
dst_host_srv_count	6.10	6.77	5.07
dst_host_count	5.77	18.22	10.10
count	5.35	13.58	5.64
srv_count	5.06	2.89	3.90
dst_host_diff_srv_rate	4.93	0.61	0.67
src_bytes	4.42	8.42	7.73
dst_host_same_srv_rate	4.39	0.58	1.22
dst_bytes	4.23	22.26	23.37
same_srv_rate	4.10	0.63	1.22
protocol_type_tcp	3.91	0.81	2.28
Total Contribution		74.78	61.20

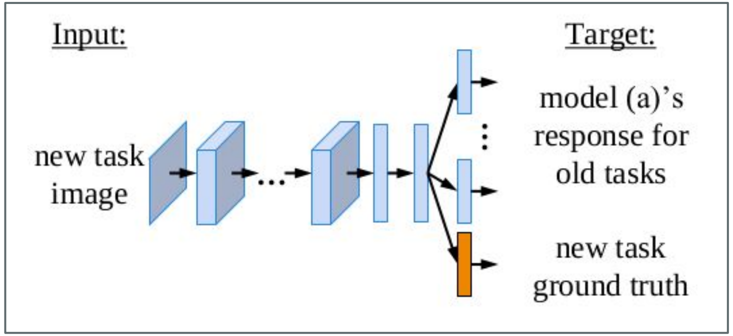
- No single features drifts a lot
- Even those features have high importance in the classification problem
- Ex: dst_bytes: Overlap 94.7% and importance 22-24%

2. Reweighting the train set data points:

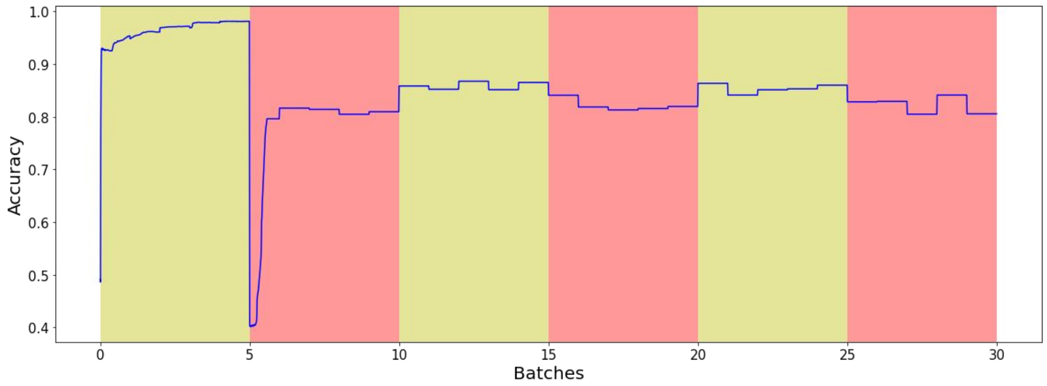


Train set Accuracy: 98.36% → 96.73% (-1.63)
Test set Accuracy: 76.40% → 78.93% (+2.53)

3. Learning without forgetting:

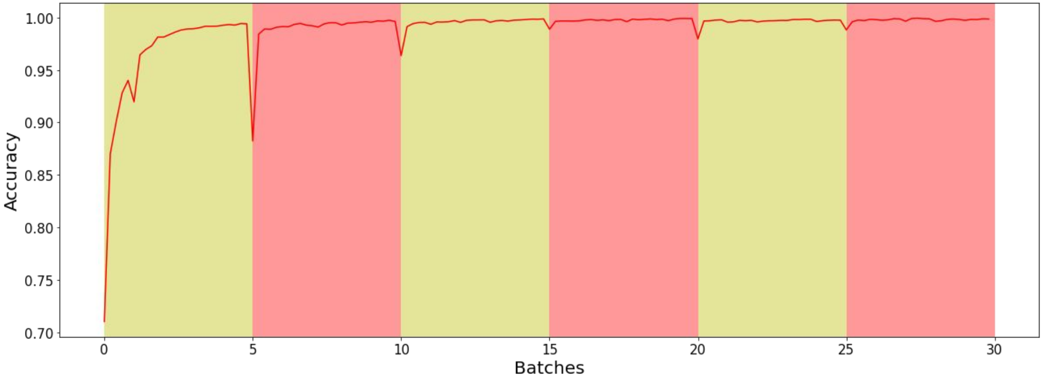


- Add more output nodes → for new attacks
- Train the new parameters by constraining the old parameters
- Regularization based strategy
- Difficult to learn completely different classes



***No combined learning**

4. Experience Replay

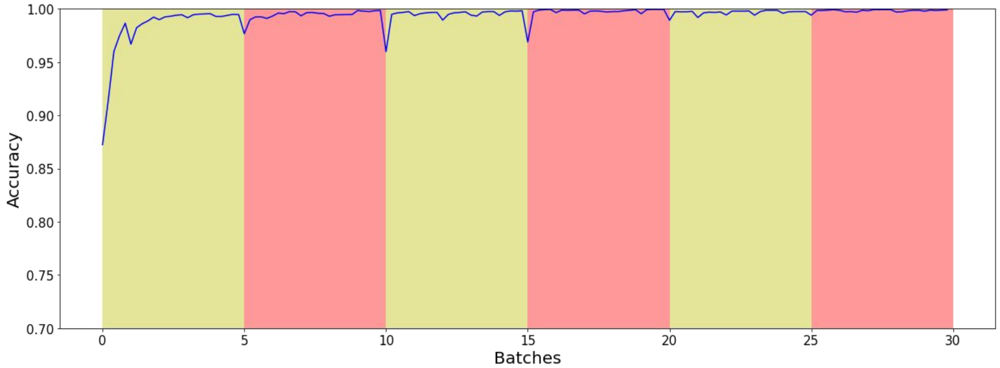


Algorithm 1: Reservoir Sampling

Input: Memory buffer M, number of samples encountered N, datapoints (x, y)
if $M < N$ **then**
  M[N] $\longleftarrow (x, y)$
else
  j = sample random integer (min = 0, max = N);
  if $j <| M |$ **then**
    M[N] $\longleftarrow (x, y)$
  end
end

- Buffer size = 5000 (Ensures Combined Learning)
- Select a random points from the buffer and replace it with points from the new batch (if the buffer is full)
- The buffer represents the collective data distribution of all the batches encountered thus far

5. Dark Experience Replay



Algorithm 2: Dark Experience Replay

Input: dataset D, parameter θ, trade-off value α and learning rate λ

M $\longleftarrow$ {}

for (x,y) *in* D **do**

$(x', z') \longleftarrow sample(M)$

$z \longleftarrow h_\theta(x)$

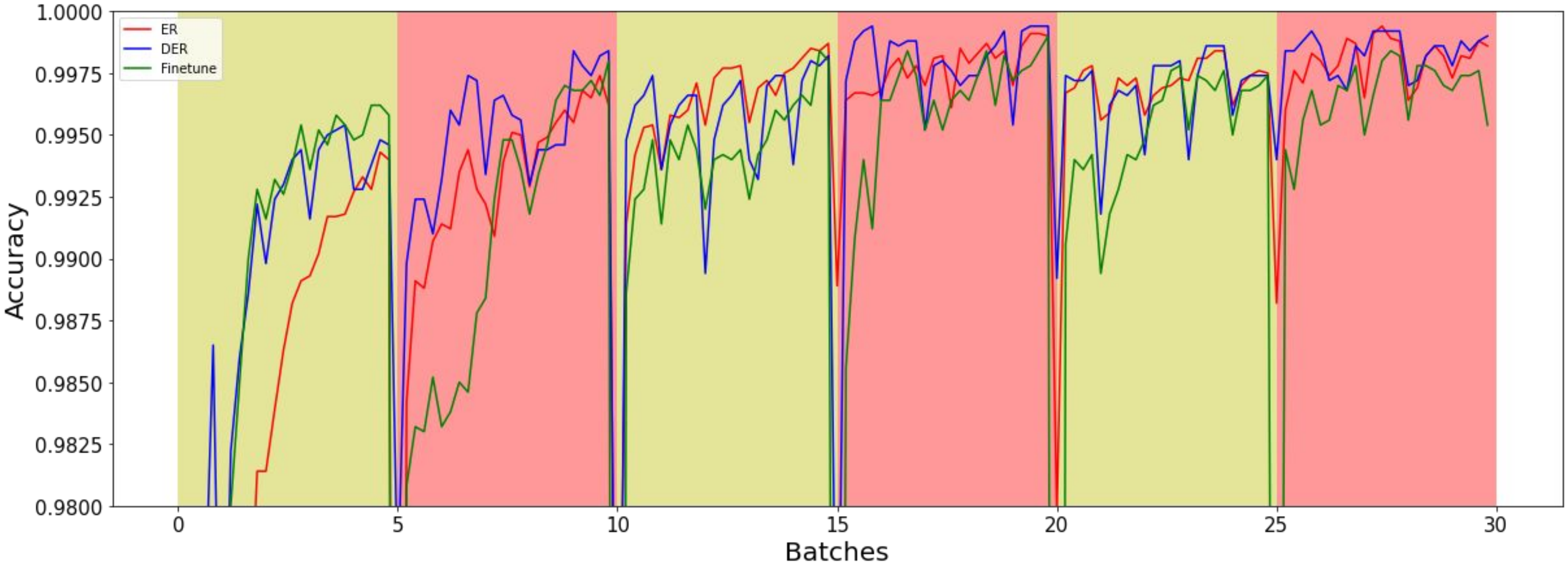
$reg \longleftarrow \alpha \mid\mid z' - h_\theta(x') \mid\mid_2^2$

$\theta \longleftarrow \theta + \lambda . \nabla_\theta [l(y, f_\theta(x)) + reg]$

$M \longleftarrow reservoir(M, (x, z))$

end

- Buffer + Regularization (Combining both the above methods)
- Buffer: Retain previous data distribution
- Regularization: Don’t change the parameters too much
- Optimize for current + old data



Compare the drop in accuracies at batch 20 and batch 25

DER > ER > Finetune

Algorithm	Overall Accuracy
Finetuning	98.85%
Experience Replay	98.93%
Dark Experience Replay	99.36%
Learning Without Forgetting	84.92%

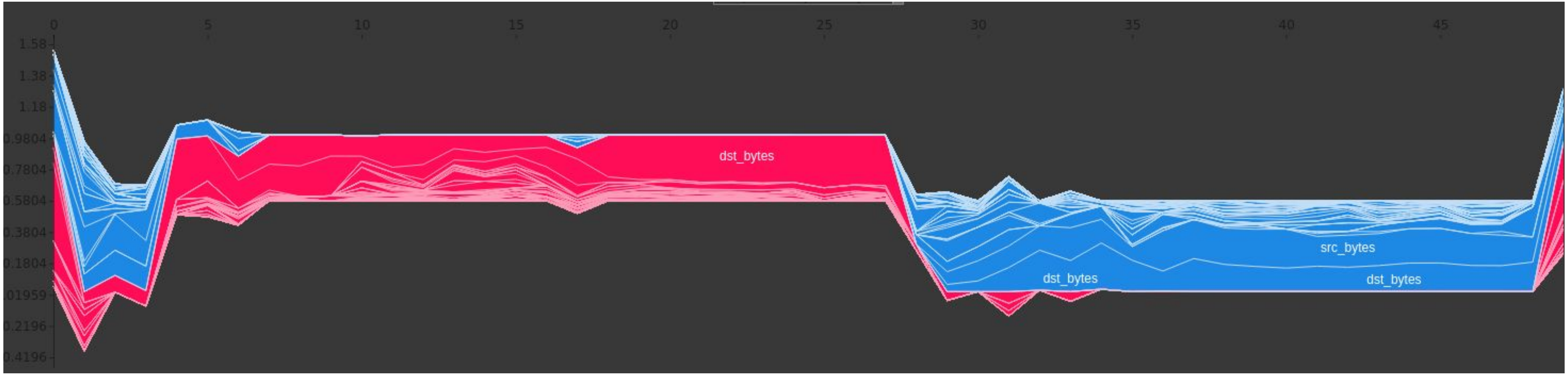
Conclusion:

- Continual Learning based IDS is the way forward considering the evolving environment in cyber security
- We extensively analyse the NSL-KDD dataset and find the dataset drift
- We present multiple ML based techniques to quantify the shift
- Multiple CL based models were tested and we conclude that rehearsal based replay methods work best for this task.

Future Works:

1. GAN based generative replay models
2. Privacy preserving ML for security
3. Extending to cloud based recent datasets


Thank You
for
Listening...
any
questions?


0
5
10
15
20
25
30
35
40
45
1.58
1.38
1.18
0.9804
0.7804
0.5804
0.3804
0.1804
0.01959
0.2196
0.4196
dst_bytes
src_bytes
dst_bytes
dst_bytes