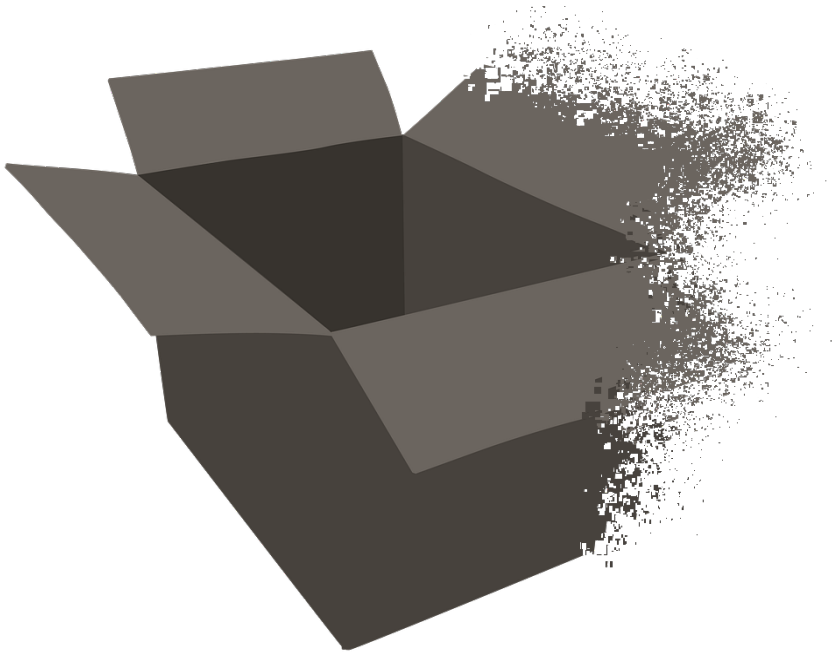


Faculty 1: Chair of IT-Security | Master Thesis Colloquium

Representation Learning for Content-Sensitive Anomaly Detection in Industrial Networks

B. Sc. Fabian Kopp

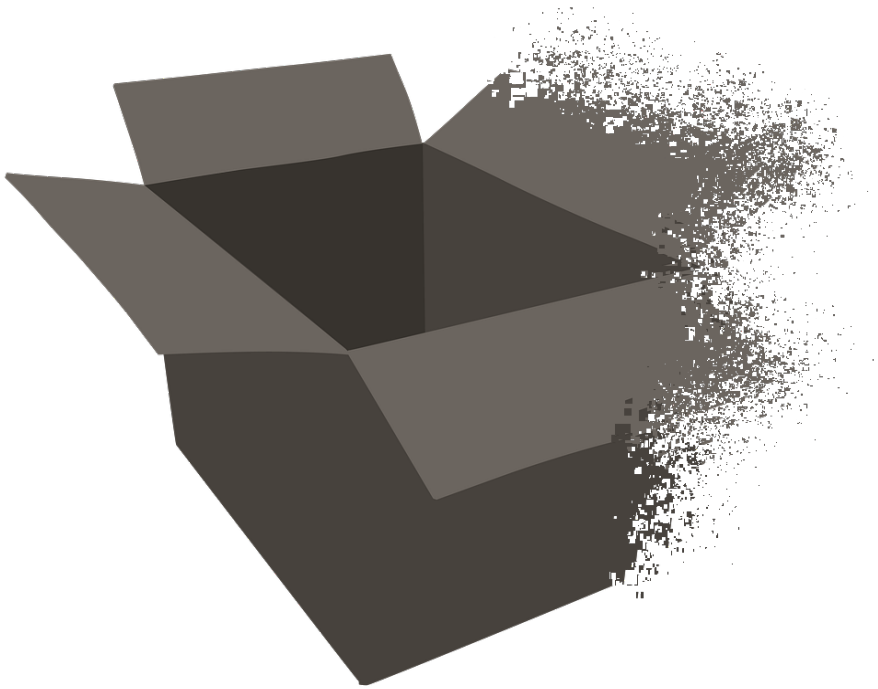
Supervisor:

M. Sc. Franka Schuster

Examination Board:

Prof. Dr.-Ing. Andriy Panchenko

Prof. Dr. rer. nat. Oliver Hohlfeld



0 | Preamble: **Problem & Motivation**

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

0 | Preamble: **Problem & Motivation**

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

→ ***Can one use this data to detect attacks against a network in an automatic fashion ?***

0 | Preamble: **Problem & Motivation**

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

→ ***Can one use this data to detect attacks against a network in an automatic fashion ?***
Yes, with ***intrusion detection systems*** *that rely on* ***anomaly detection*** *!*

0 | Preamble: **Problem & Motivation**

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

→ ***Can one use this data to detect attacks against a network in an automatic fashion ?***
Yes, with ***intrusion detection systems*** *that rely on* ***anomaly detection*** *!*

Focus on **feature extraction** and model **interpretability**

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

→ ***Can one use this data to detect attacks against a network in an automatic fashion ?***
Yes, with ***intrusion detection systems*** *that rely on* ***anomaly detection*** *!*

Focus on **feature extraction** and model **interpretability**

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

→ ***Can one use this data to detect attacks against a network in an automatic fashion ?***
Yes, with ***intrusion detection systems*** *that rely on* ***anomaly detection*** *!*

Focus on **feature extraction** and model **interpretability**

– Automatic transformation of input data into a set of derived values (features)
– Features intended to be informative & non-redundant
– Extracted features are basis for any downstream task

Representation Learning ***for Content-Sensitive Anomaly Detection in Industrial Networks***

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

→ ***Can one use this data to detect attacks against a network in an automatic fashion ?***
Yes, with ***intrusion detection systems*** *that rely on* ***anomaly detection*** *!*

Focus on **feature extraction** and model **interpretability**

– *Automatic transformation of input data into a set of derived values (features)*
– *Features intended to be informative & non-redundant*
– *Extracted features are basis for any downstream task*

– *Crucial for the detection of unknown attacks*
– *Detection based on raw data*

Representation Learning for Content-Sensitive Anomaly Detection in Industrial Networks

- Threat defense is an *open* problem
- Computer networks generate lots of data
- Artificial neural networks are *powerful* computational systems

→ ***Can one use this data to detect attacks against a network in an automatic fashion ?***
Yes, with ***intrusion detection systems*** *that rely on* ***anomaly detection*** *!*

Focus on **feature extraction** and model **interpretability**

– *Automatic transformation of input data into a set of derived values (features)*
– *Features intended to be informative & non-redundant*
– *Extracted features are basis for any downstream task*

– *Crucial for the detection of unknown attacks*
– *Detection based on raw data*

– *Valuable asset to protect*
– *Fixed network topology*

Representation Learning for Content-Sensitive Anomaly Detection in Industrial Networks

I. *Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?*

I. Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?

II. Can anomaly detection be improved when trained on extracted features ?

I. Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?

II. Can anomaly detection be improved when trained on extracted features ?

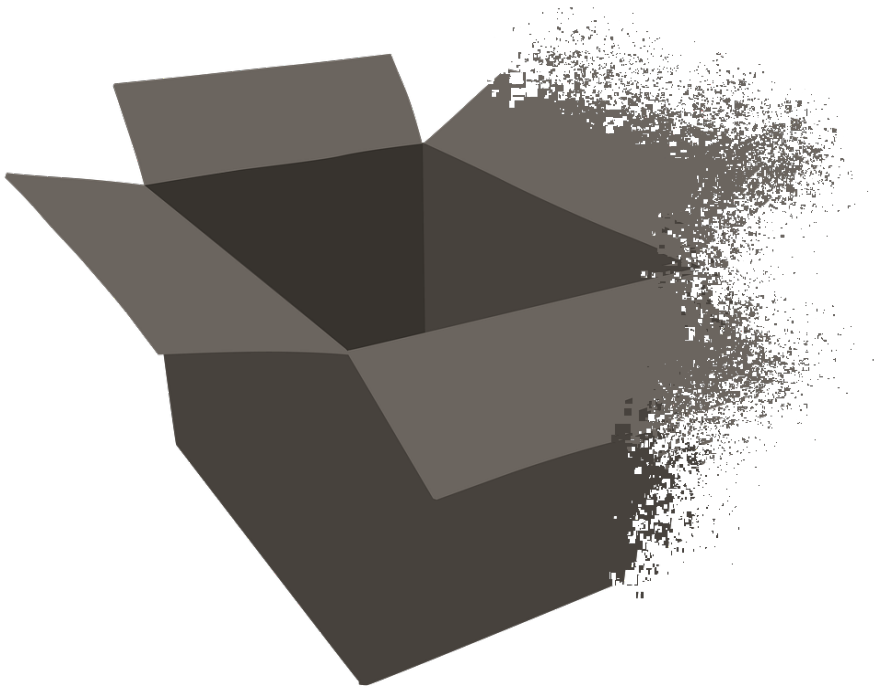
III. What type of input data representations yields better results for different anomalies ?

I. Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?

II. Can anomaly detection be improved when trained on extracted features ?

III. What type of input data representations yields better results for different anomalies ?

IV. How can the semantic gap be closed with the help of explainable models ?



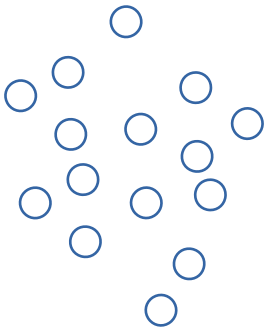
Anomaly detection (*AD*)

- Unsupervised one class classification
- *Learn* normal behavior - alert deviations
- Used in: fraud detection, intrusion detection

Brandenburg University of Technology Cottbus - Senftenberg

Anomaly detection (*AD*)

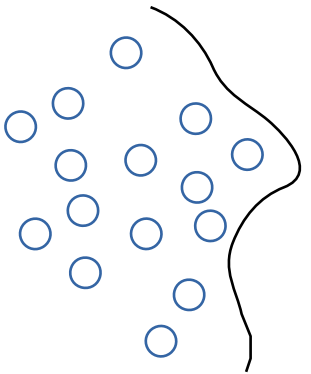
- Unsupervised one class classification
- *Learn* normal behavior - alert deviations
- Used in: fraud detection, intrusion detection



Normal

Anomaly detection (*AD*)

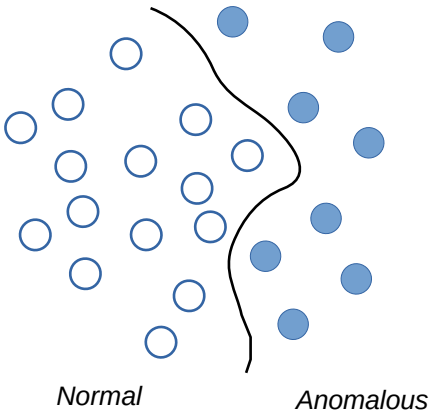
- Unsupervised one class classification
- *Learn* normal behavior - alert deviations
- Used in: fraud detection, intrusion detection



Normal

Anomaly detection (*AD*)

- Unsupervised one class classification
- *Learn* normal behavior - alert deviations
- Used in: fraud detection, intrusion detection

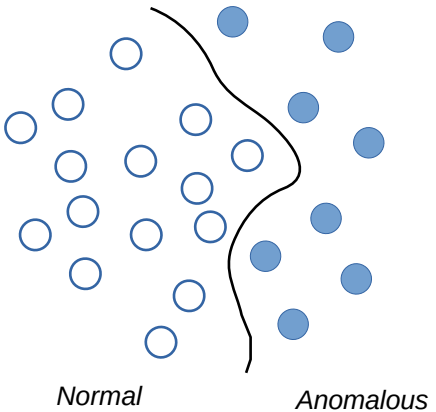


Anomaly detection (*AD*)

- Unsupervised one class classification
- *Learn* normal behavior - alert deviations
- Used in: fraud detection, intrusion detection

Types of anomalies

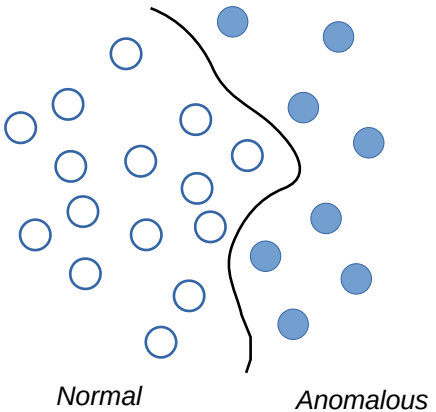
- (Un)known attacks
- Human error
- Insider abuse
- Faulty hardware



Brandenburg University of Technology Cottbus - Senftenberg

Anomaly detection (*AD*)

- Unsupervised one class classification
- *Learn* normal behavior - alert deviations
- Used in: fraud detection, intrusion detection



Types of anomalies

- (Un)known attacks
- Human error
- Insider abuse
- Faulty hardware

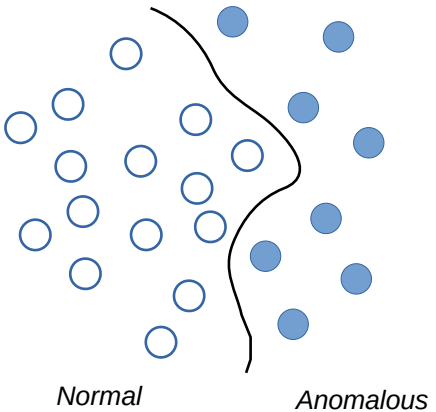
Anomaly classes

- Intra-packet anomalies

Brandenburg University of Technology Cottbus - Senftenberg

Anomaly detection (*AD*)

- Unsupervised one class classification
- *Learn* normal behavior - alert deviations
- Used in: fraud detection, intrusion detection

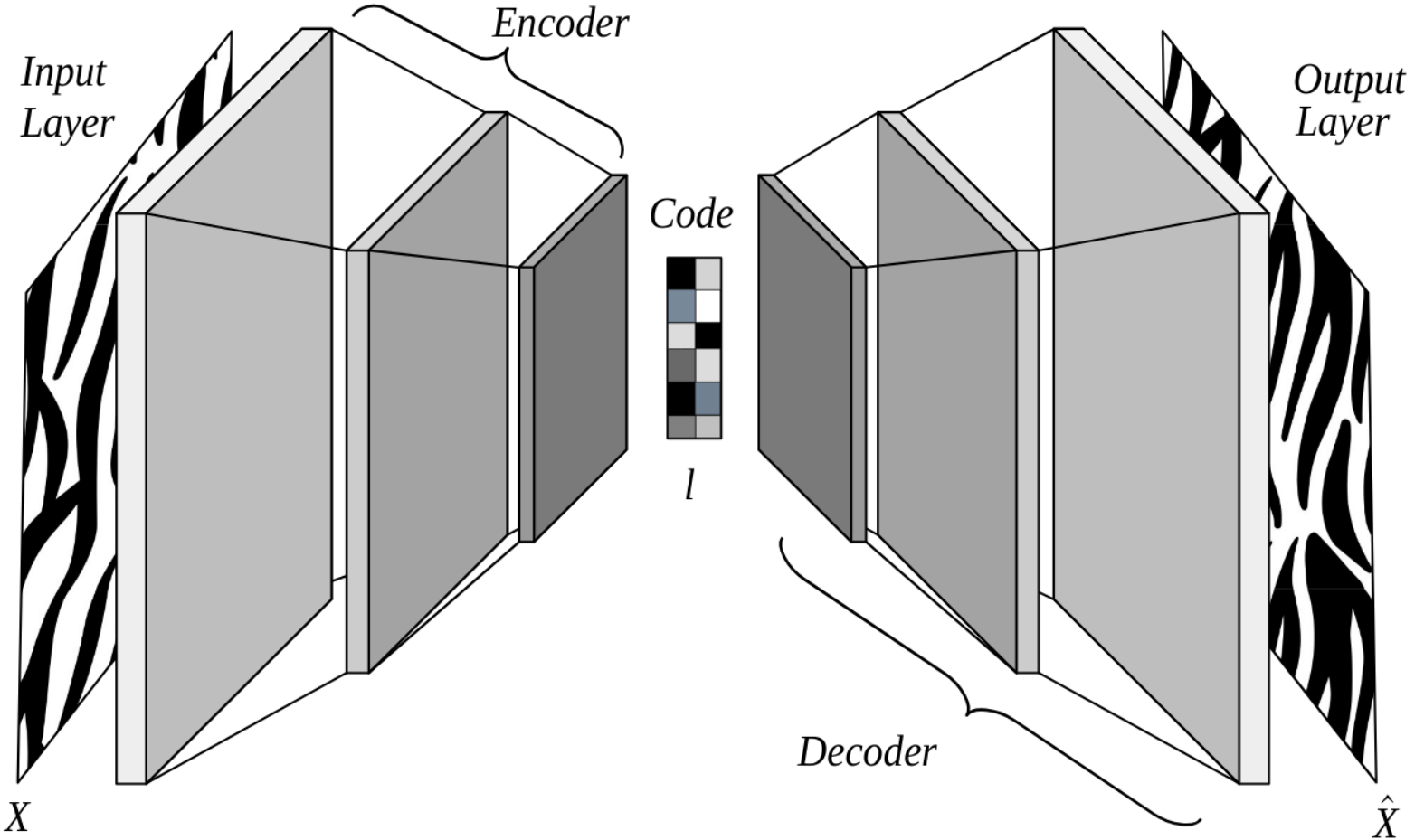


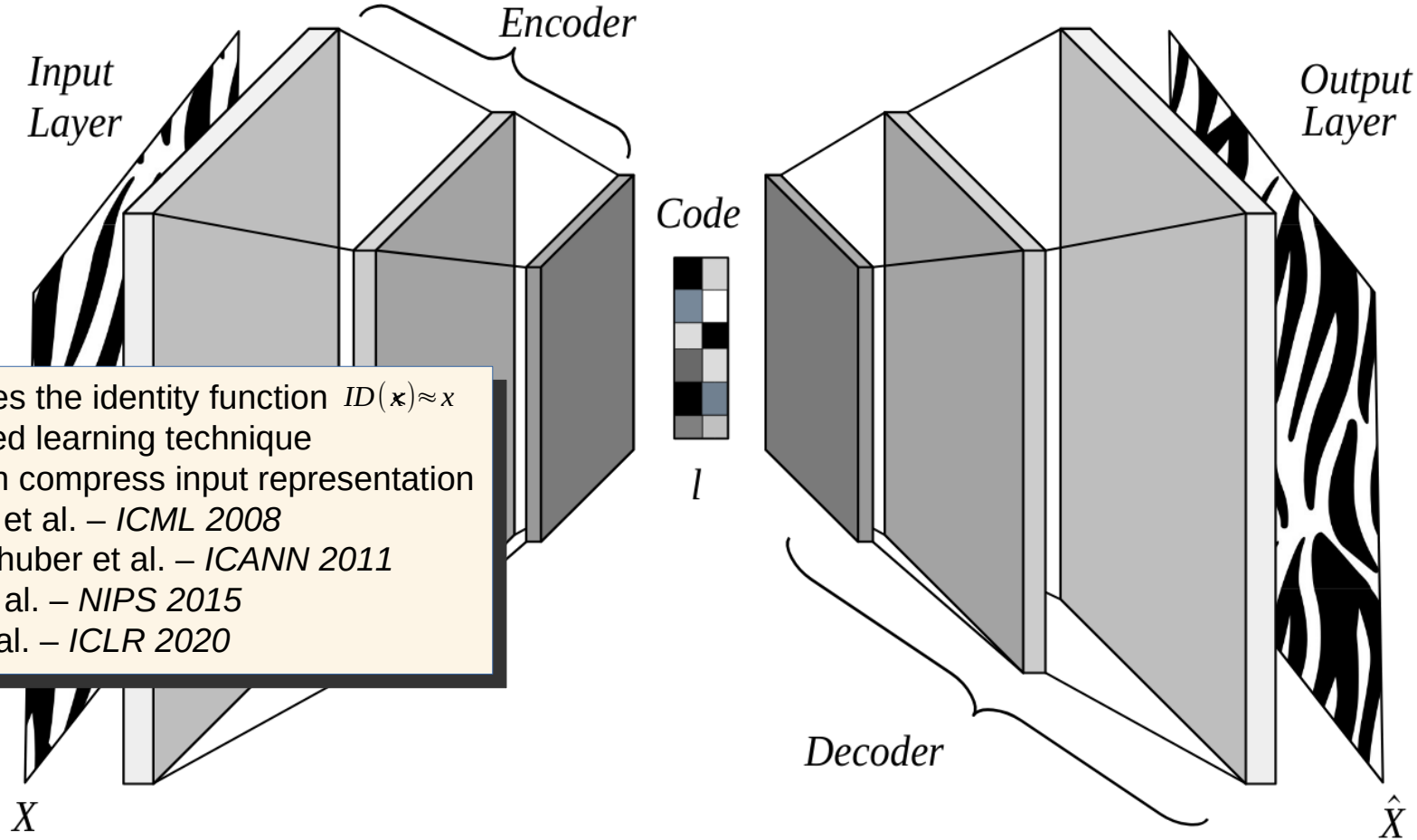
Types of anomalies

- (Un)known attacks
- Human error
- Insider abuse
- Faulty hardware

Anomaly classes

- Intra-packet anomalies
- Inter-packet anomalies

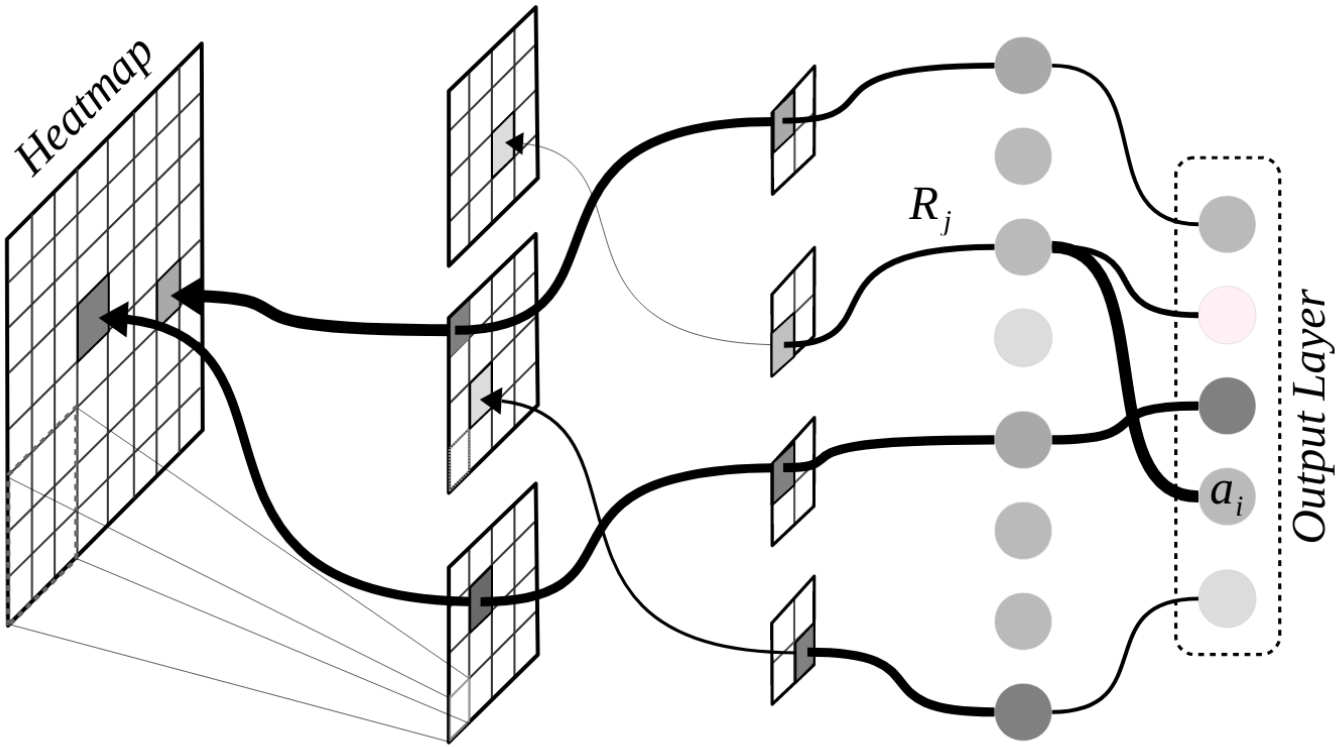

Encoder
Input Layer
Code
Output Layer
Decoder
X
l
$\hat{X}$



- Approximates the identity function $ID(x) \approx x$
- Unsupervised learning technique
- Encoder can compress input representation
  – Y. Bengio et al. – *ICML 2008*
  – J. Schmidhuber et al. – *ICANN 2011*
  – B. Frey et al. – *NIPS 2015*
  – K. Kim et al. – *ICLR 2020*

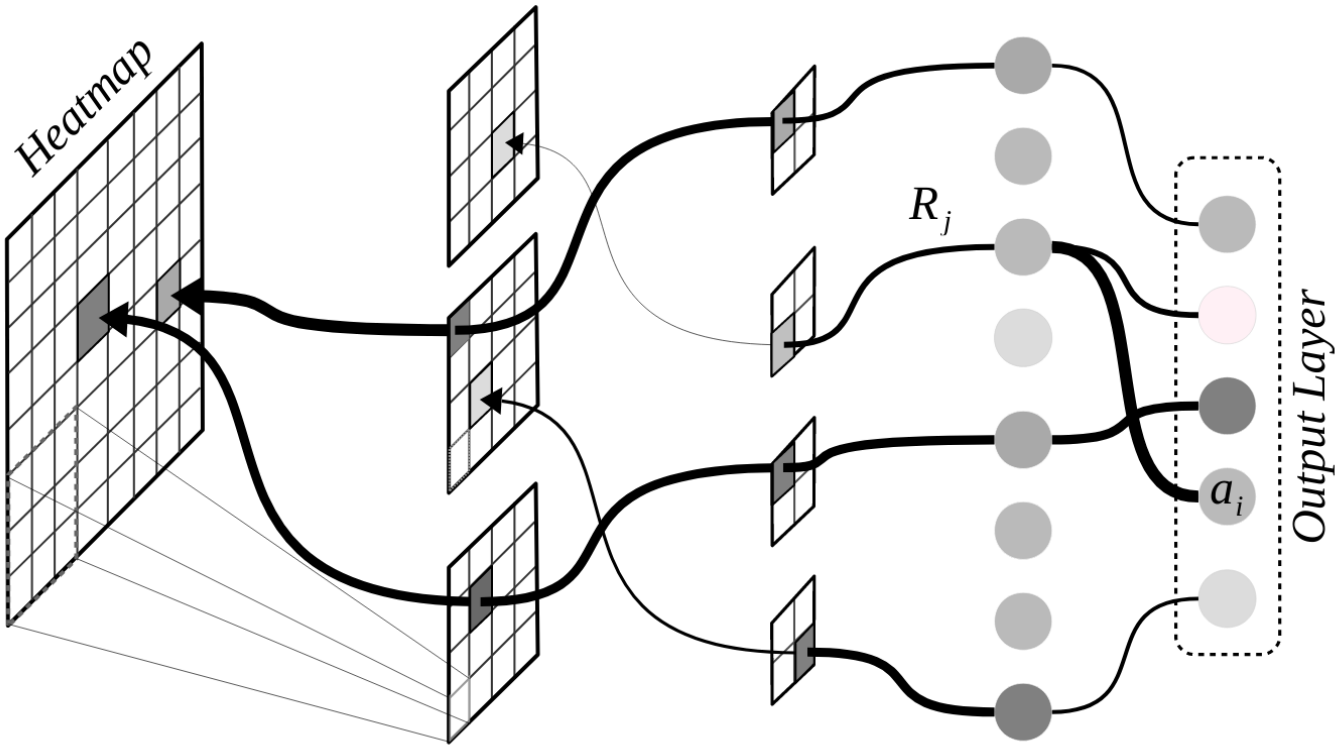
→ **Model Inference**: "*What class is present in a given image ?*"

← **Decision Analysis**: "*Why is a given image classified as a cat ?*"



→ **Model Inference**: "*What class is present in a given image ?*"

← **Decision Analysis**: "*Why is a given image classified as a cat ?*"



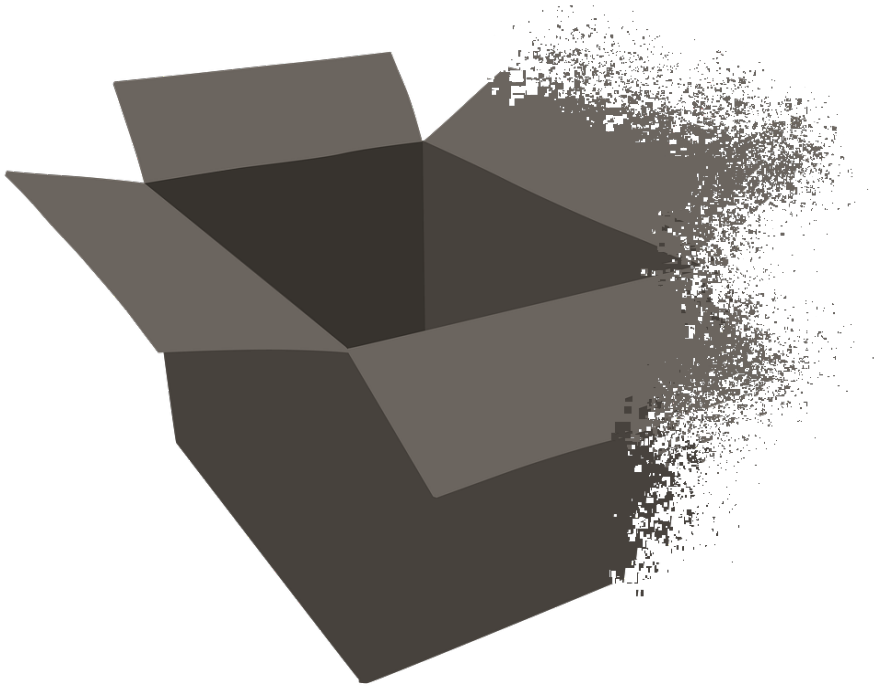
$$a_k = max\left(0, \sum_0^j a_j w_{jk}\right) \quad with \quad a_0 = 1$$

$$R_j = \sum_k^j \frac{a_j w_{jk}}{\sum_0^j a_j w_{jk}} R_k \quad where \quad R_0 = 1$$

Relevance Propagation Rule
A. Binder et al. – ICANN 2016

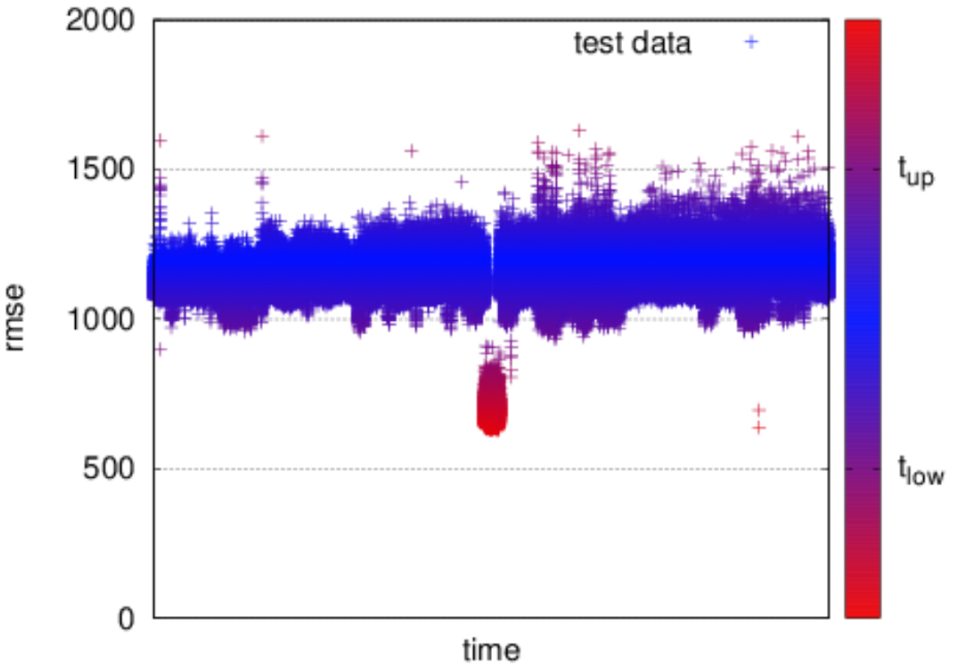
→ **Model Inference**: "*What class is present in a given image ?*"

← **Decision Analysis**: "*Why is a given image classified as a cat ?*"



High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks
[P. Schneider et al. – CPS-SPC 2018]

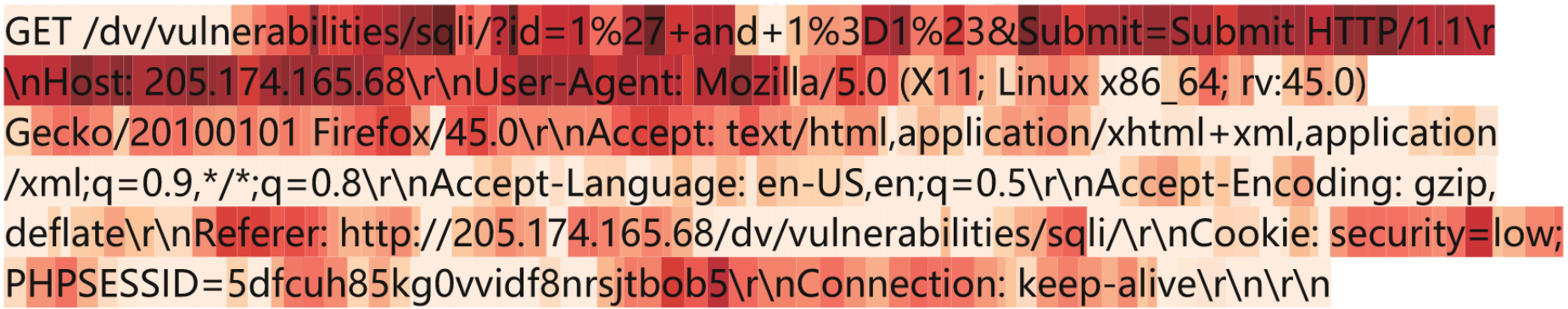
- 3 layer linear denosing autoencoder
- First 1000 bytes used for detection
- Loss-based anomaly detection
- Unsupervised end-to-end approach (+)
- No direct temporal learning (–)
- Drawbacks on detecting short anomalies (–)

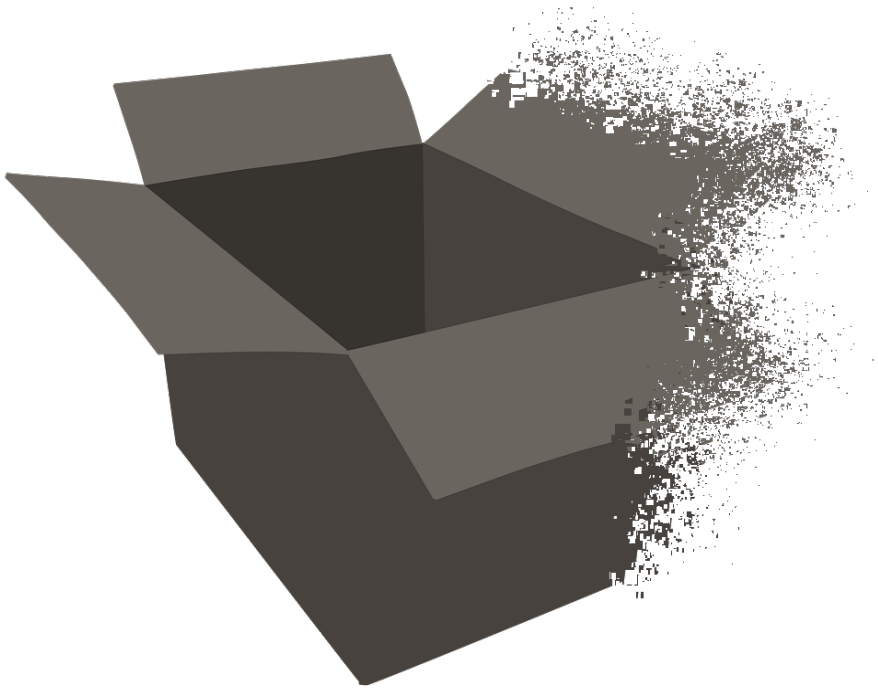


Attentional Payload Anomaly Detector for Web Applications

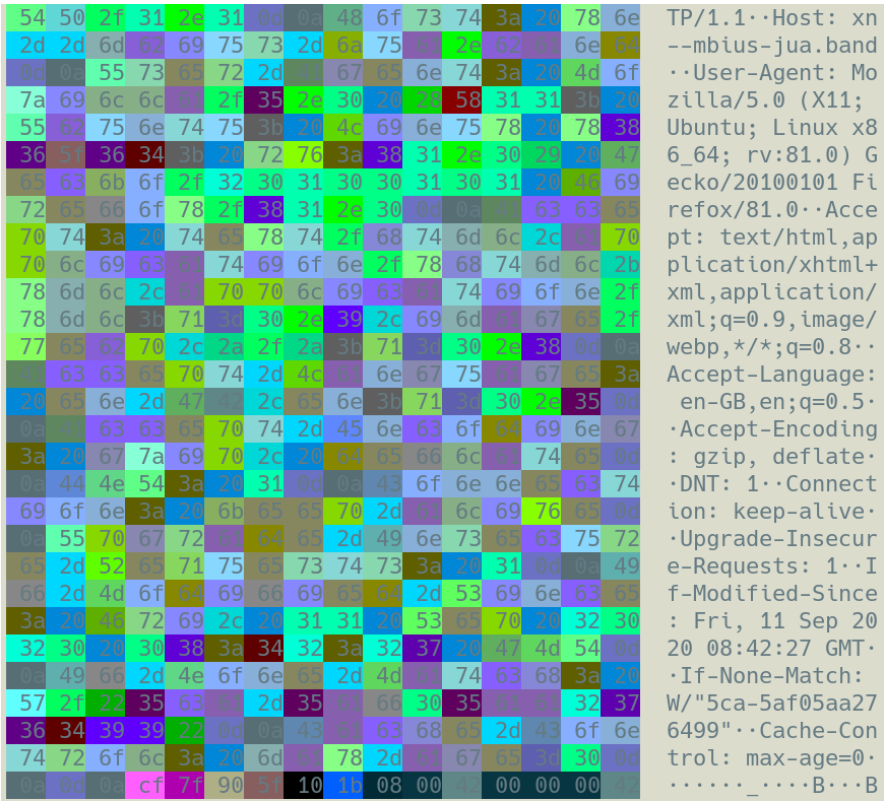
[ZQ. Qin et al. – ICONIP 2018]

- Anomaly detection based on raw bytes sequences
- TCP payloads are basis for detection
- Attention-based RNN model
- Supervised training (–)
- Heuristic to visualize the model's interpretation (+)

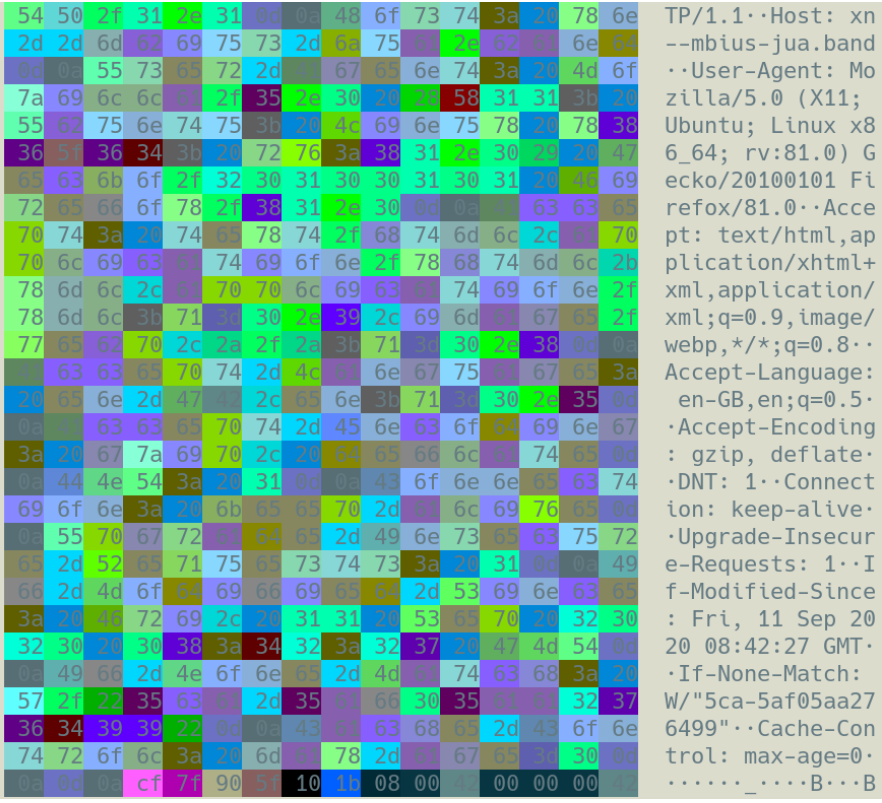




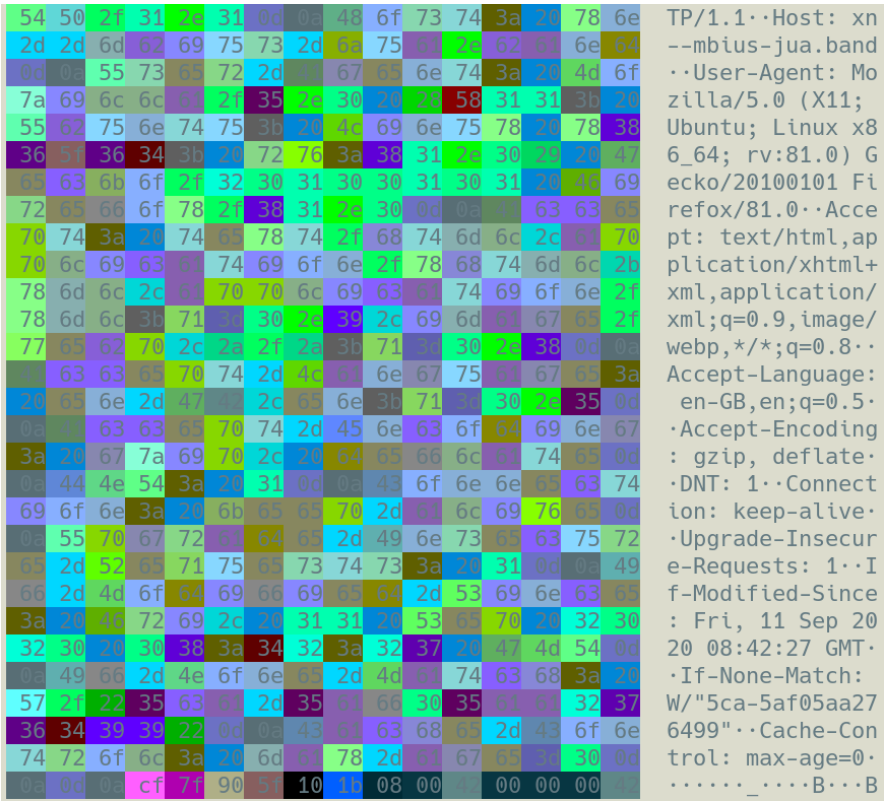
- Extract features from raw frames for a *content sensitive* AD
  - Enables a protocol-agnostic processing
  - Comparison between *byte*- and *flow*-oriented input data



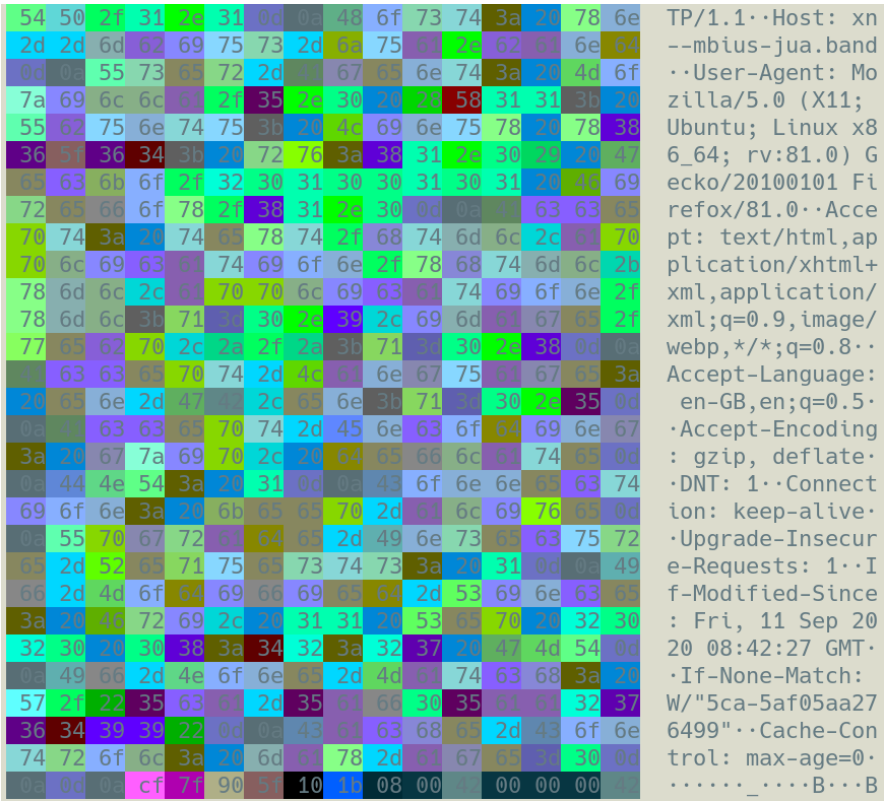
- Extract features from raw frames for a *content sensitive* AD
  - Enables a protocol-agnostic processing
  - Comparison between *byte*- and *flow*-oriented input data
- Mixture of different types of neurons for versatile features
  - *Convolutional* neurons for spatial extraction
    → intra-packet anomalies
  - *Recurrent* neurons for temporal extraction
    → inter-packet anomalies

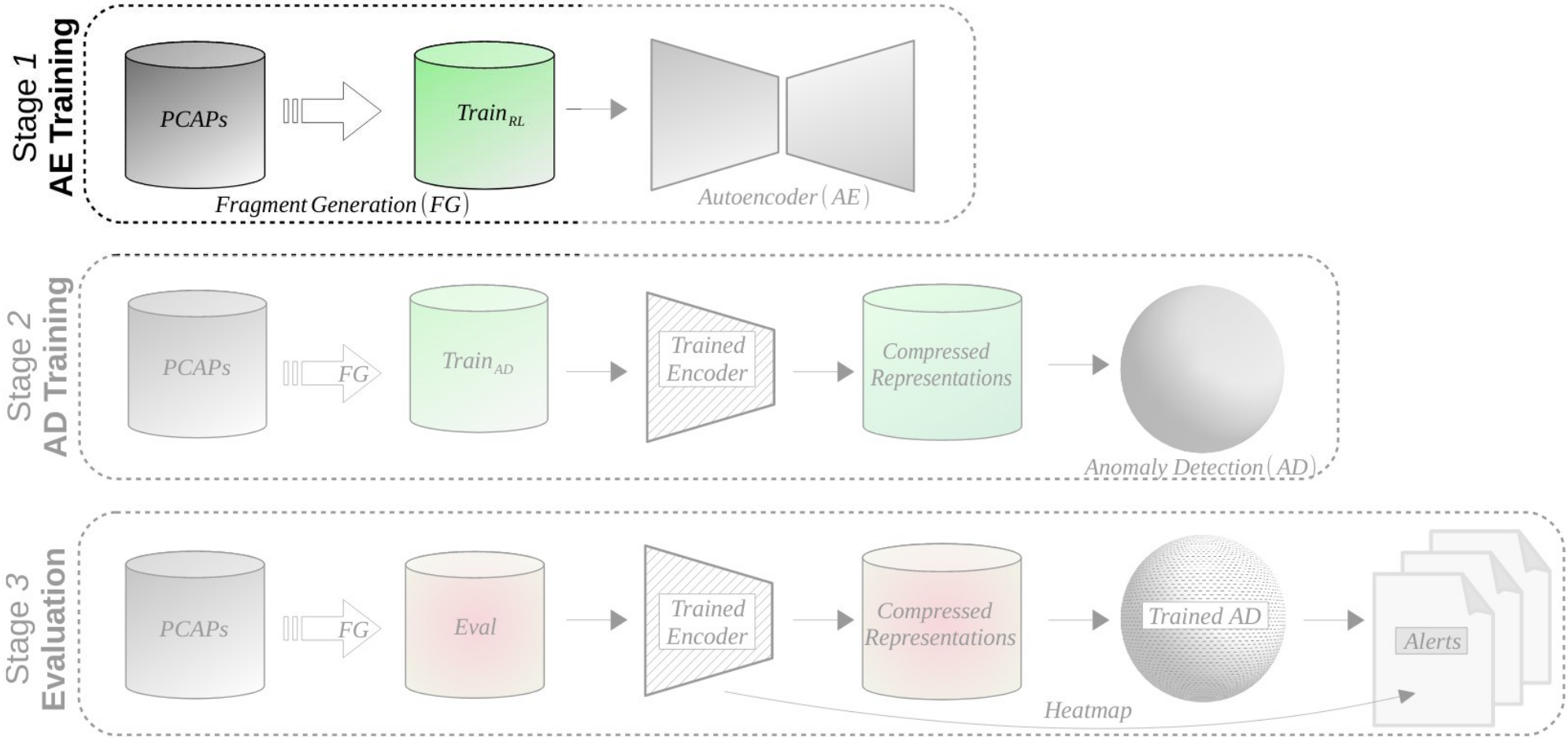


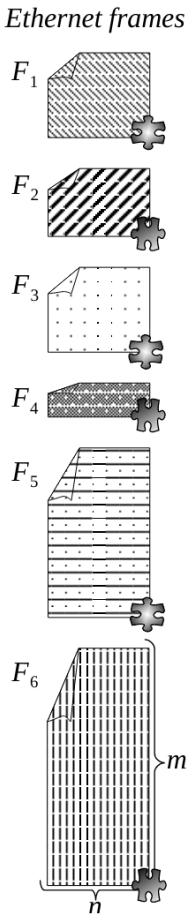
- Extract features from raw frames for a *content sensitive* AD
  - Enables a protocol-agnostic processing
  - Comparison between *byte*- and *flow*-oriented input data
- Mixture of different types of neurons for versatile features
  - *Convolutional* neurons for spatial extraction
    → intra-packet anomalies
  - *Recurrent* neurons for temporal extraction
    → inter-packet anomalies
- *Unsupervised* feature learning based on autoencoders
  - *Leverage the* encoder to extract features

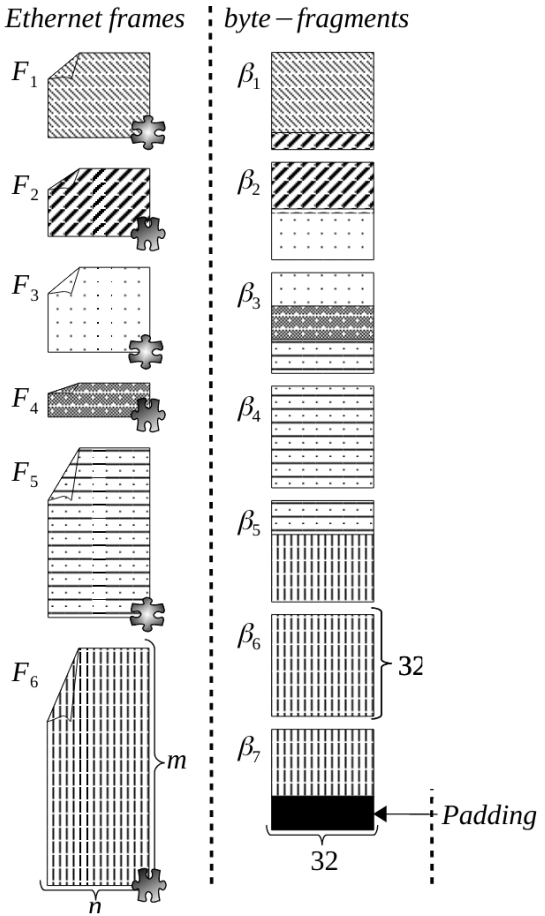


- Extract features from raw frames for a *content sensitive* AD
  - Enables a protocol-agnostic processing
  - Comparison between *byte*- and *flow*-oriented input data
- Mixture of different types of neurons for versatile features
  - *Convolutional* neurons for spatial extraction
    → intra-packet anomalies
  - *Recurrent* neurons for temporal extraction
    → inter-packet anomalies
- *Unsupervised* feature learning based on autoencoders
  - *Leverage the* encoder to extract features
- Use *layer-wise relevance propagation* for raised alerts
  - Helps to interpret the model's decisions

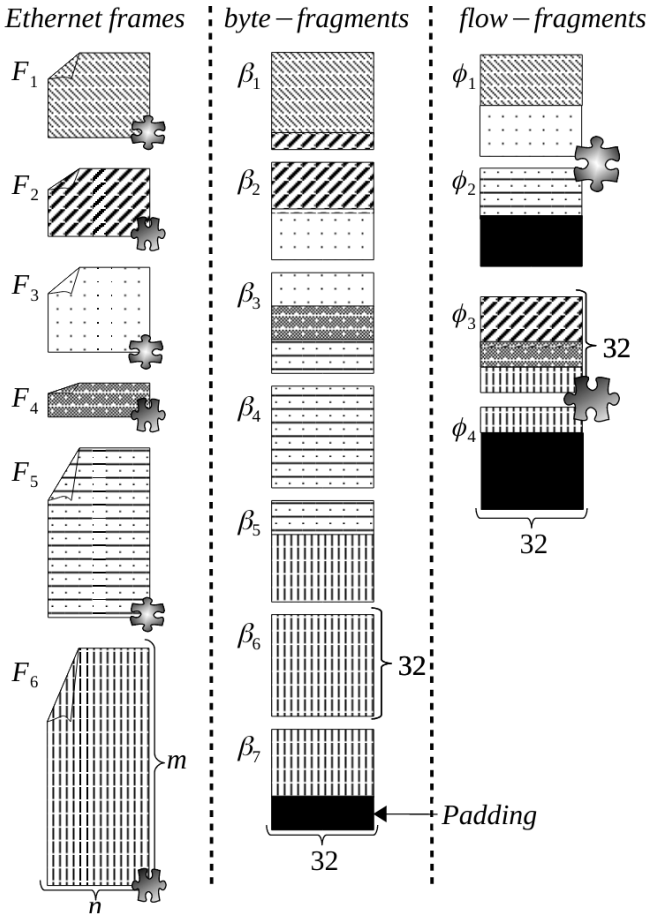



Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Compressed Representations
Trained AD
Alerts
Heatmap


Ethernet frames
F_1
F_2
F_3
F_4
F_5
F_6
m
n

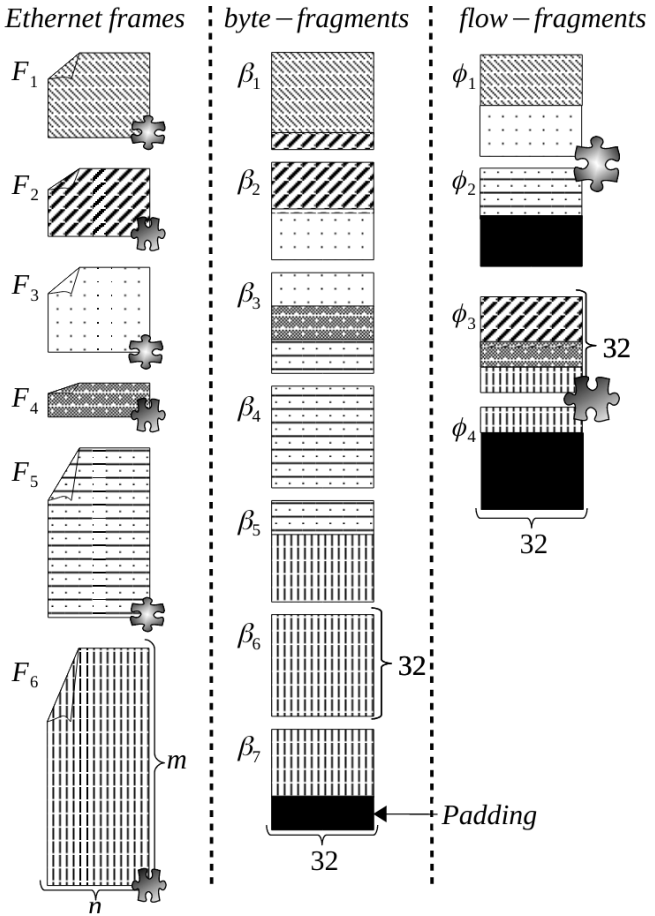


- Byte-Fragments
  - Jumping sliding-window extracted fixed fragments
  - Processes the whole input stream of frames



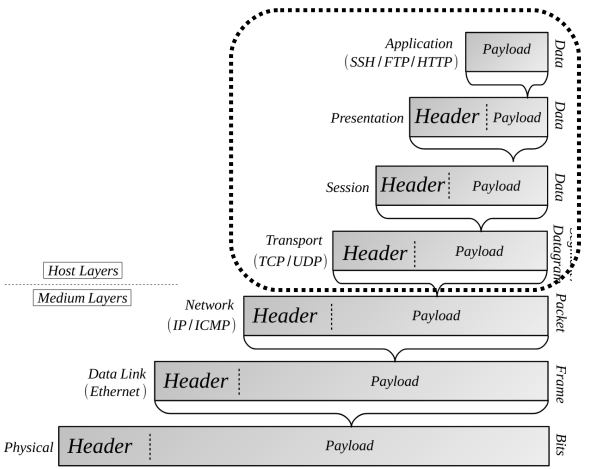
- Byte-Fragments
  - Jumping sliding-window extracted fixed fragments
  - Processes the whole input stream of frames
- Flow-Fragments
  - Sorts input stream by *network flow*

    → (Transport Protocol, src:IP, src:Port, dst:IP, dst:Port)



- Byte-Fragments
  - Jumping sliding-window extracted fixed fragments
  - Processes the whole input stream of frames
- Flow-Fragments
  - Sorts input stream by *network flow*

    → (Transport Protocol, src:IP, src:Port, dst:IP, dst:Port)
  - *Use first 64 bytes of every network layer payload*



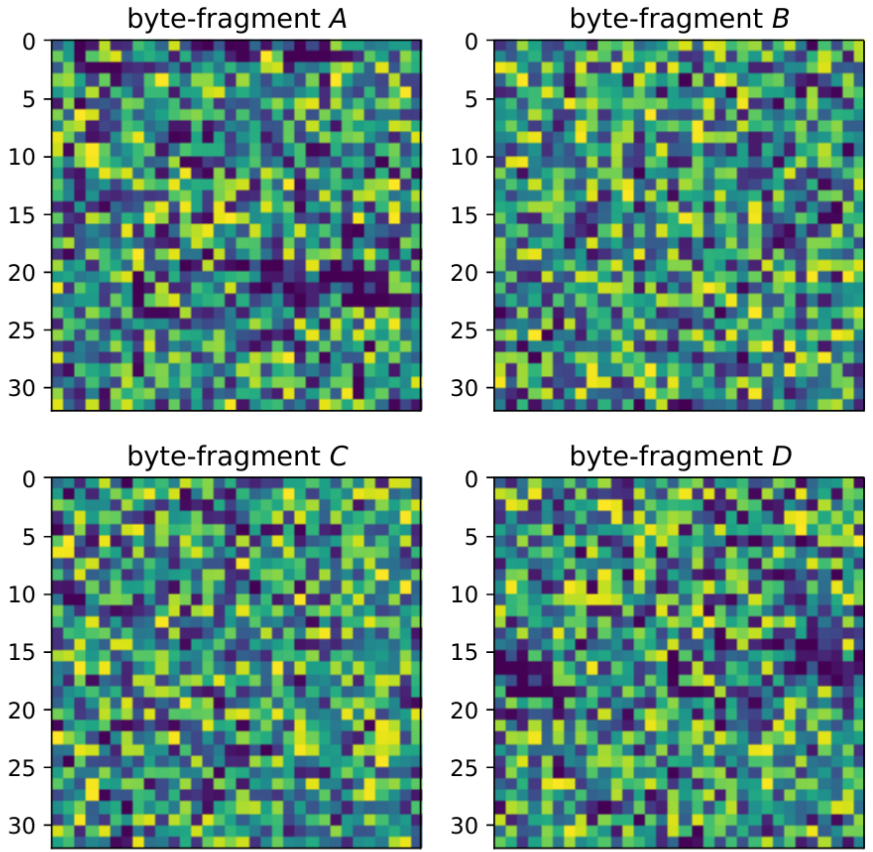


Figure 5.2: Four consecutive examples of byte-fragments shaded using a *viridis* color map

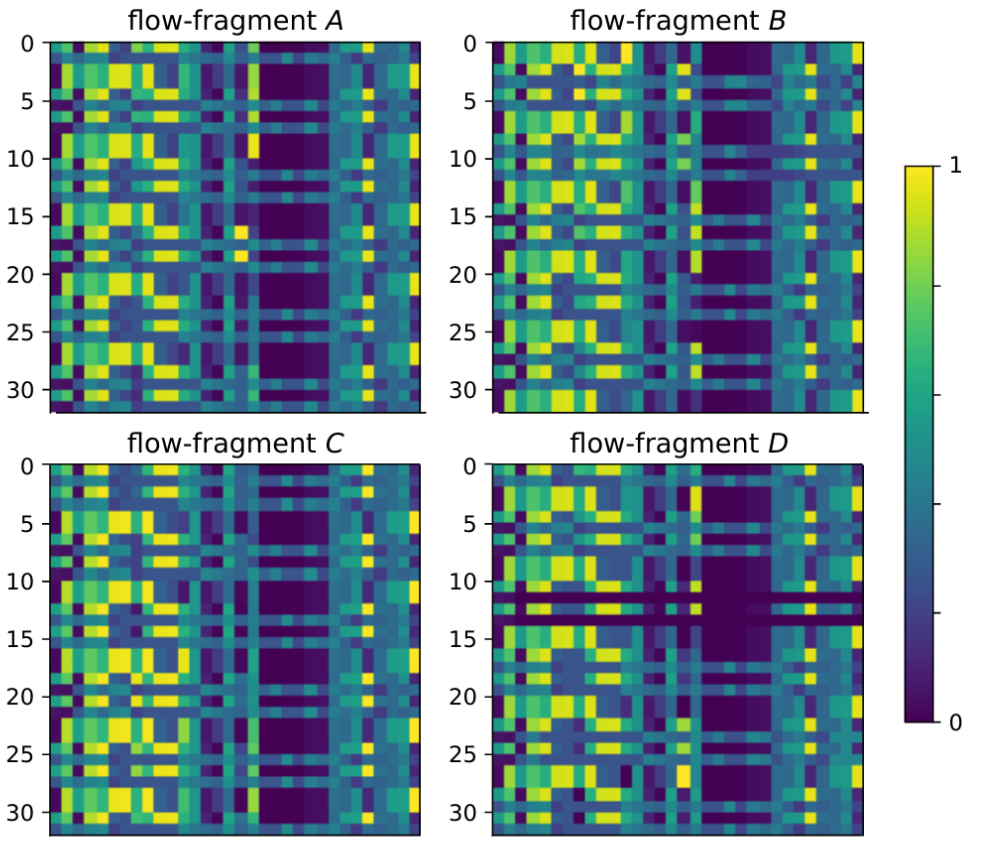
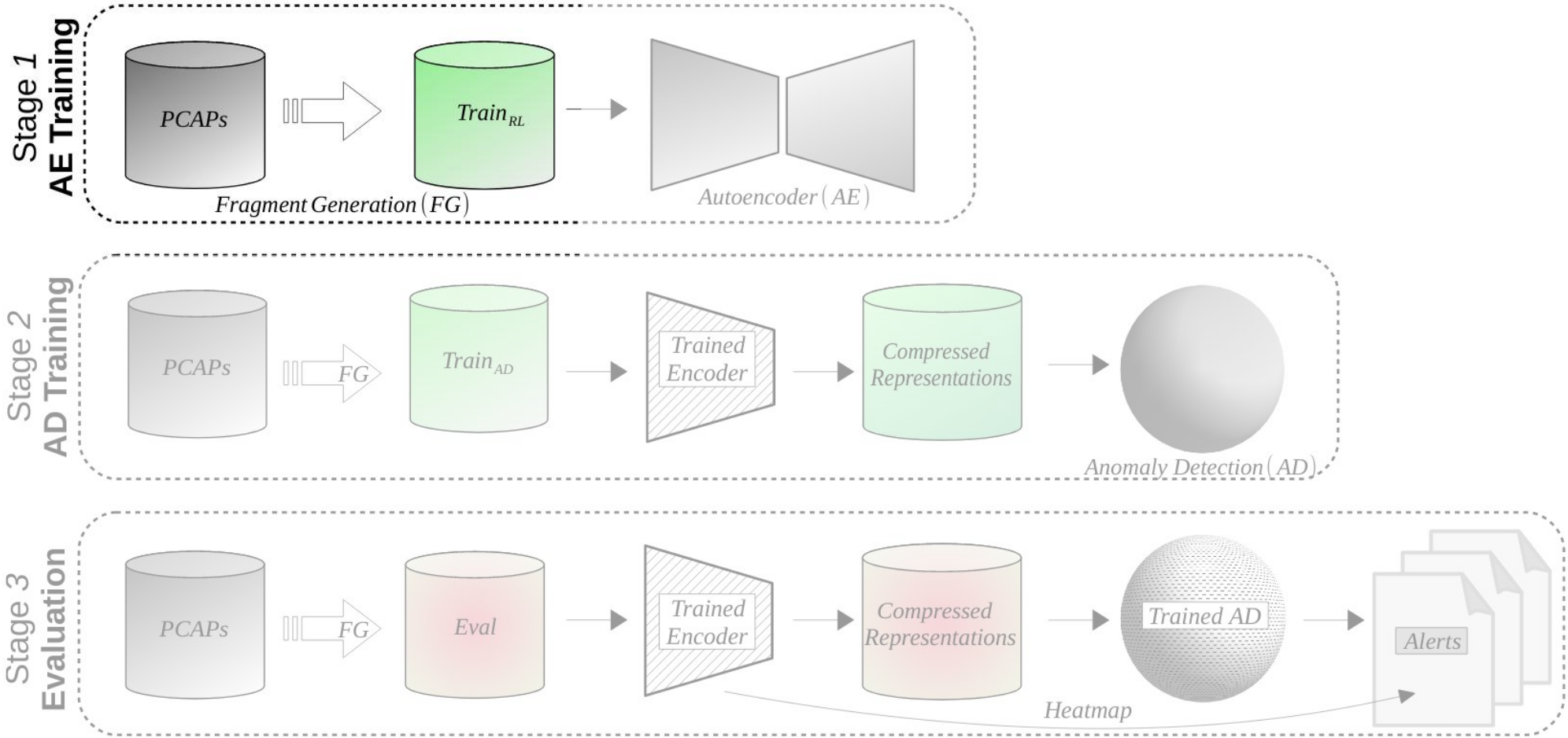
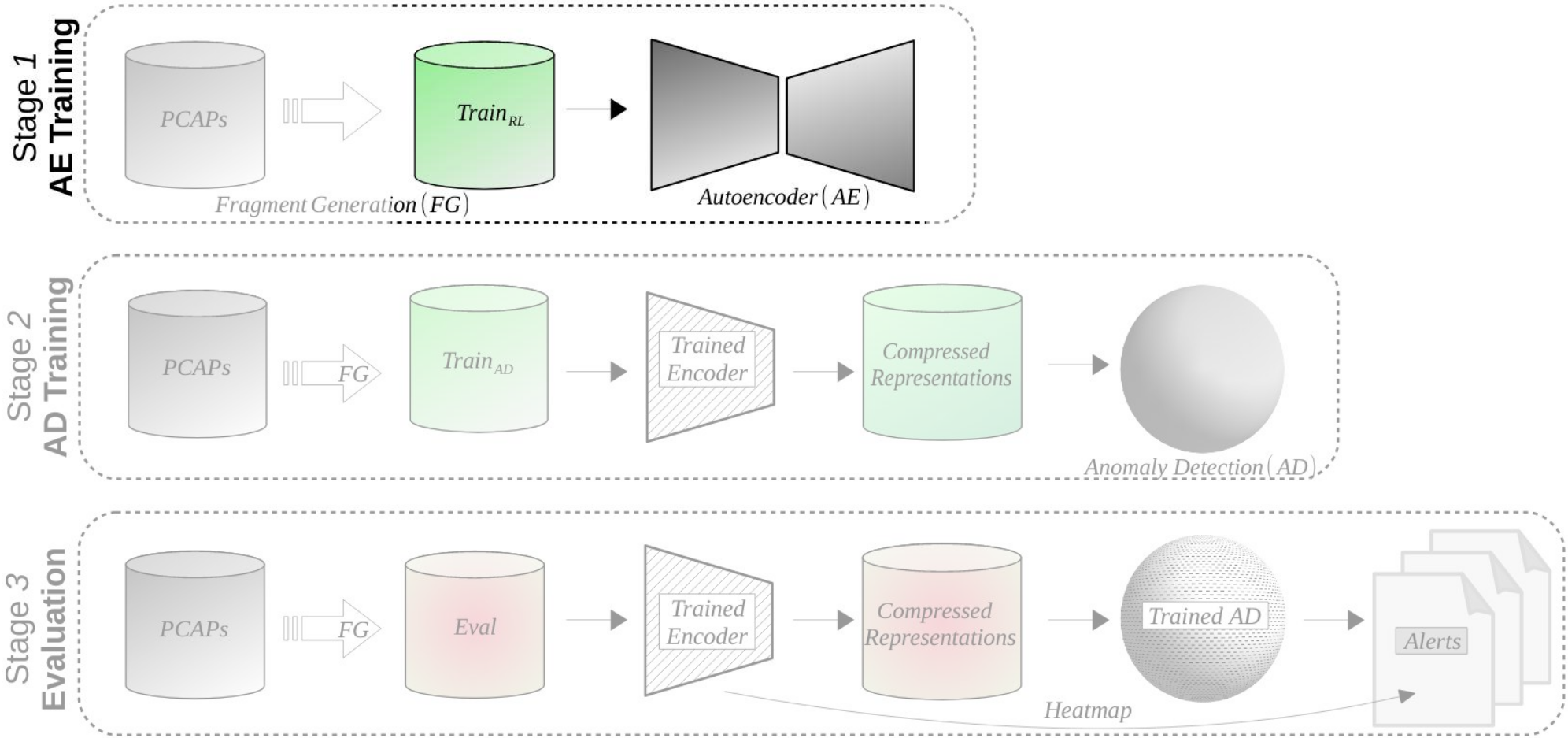
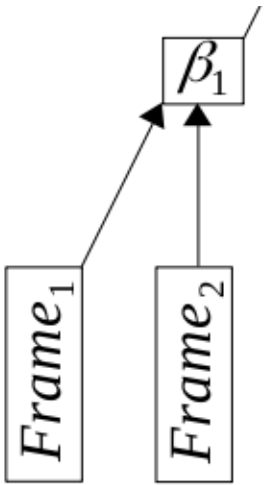
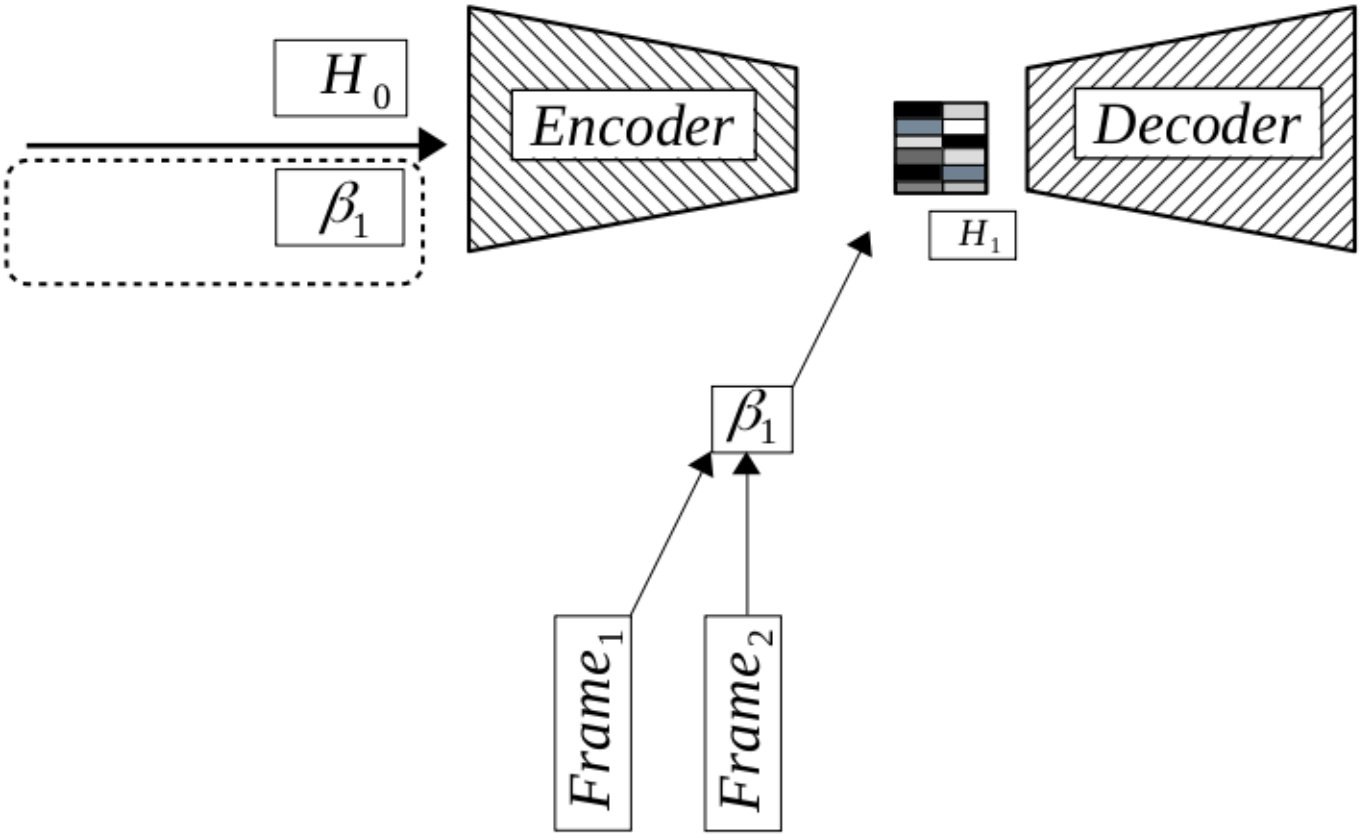


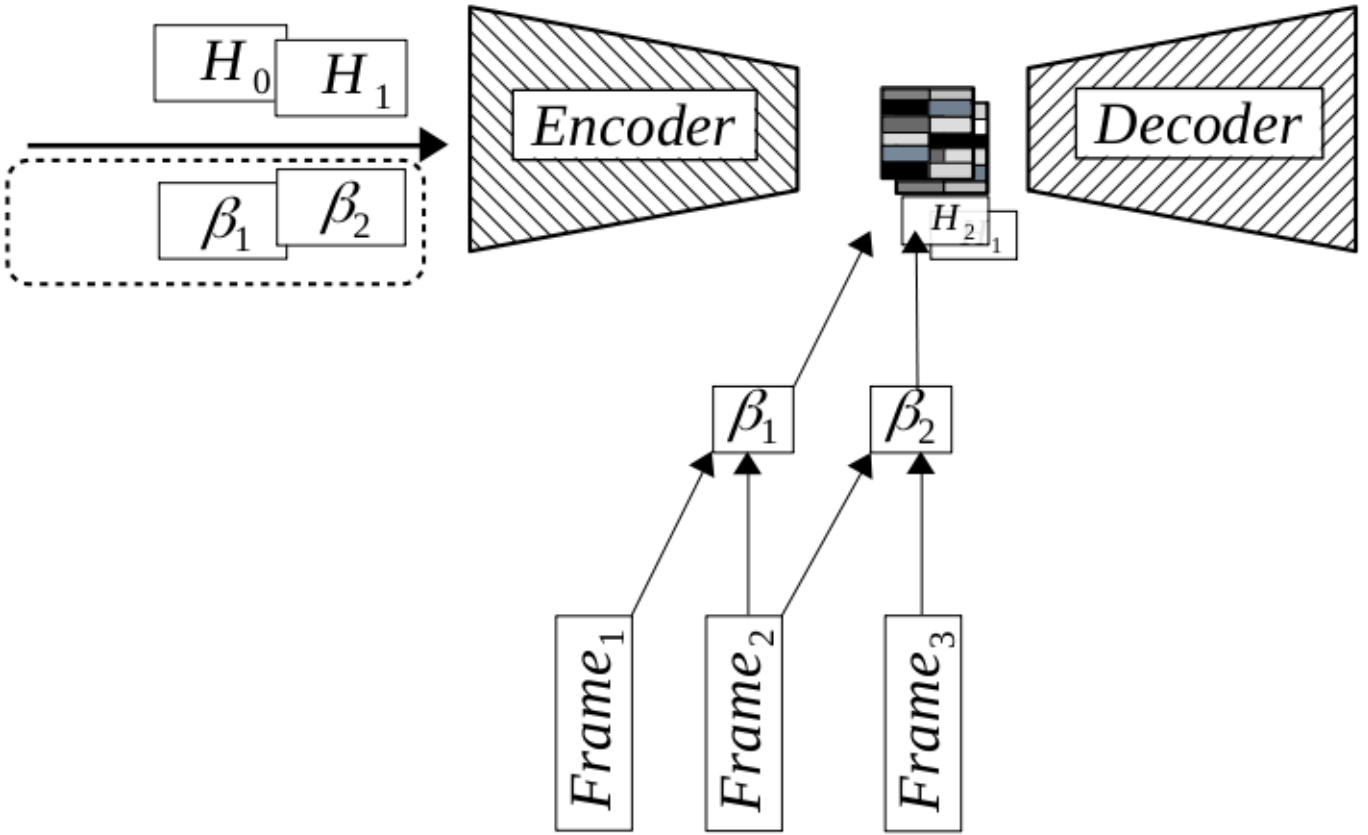
Figure 5.3: Four consecutive examples of flow-fragments shaded using a *viridis* color map

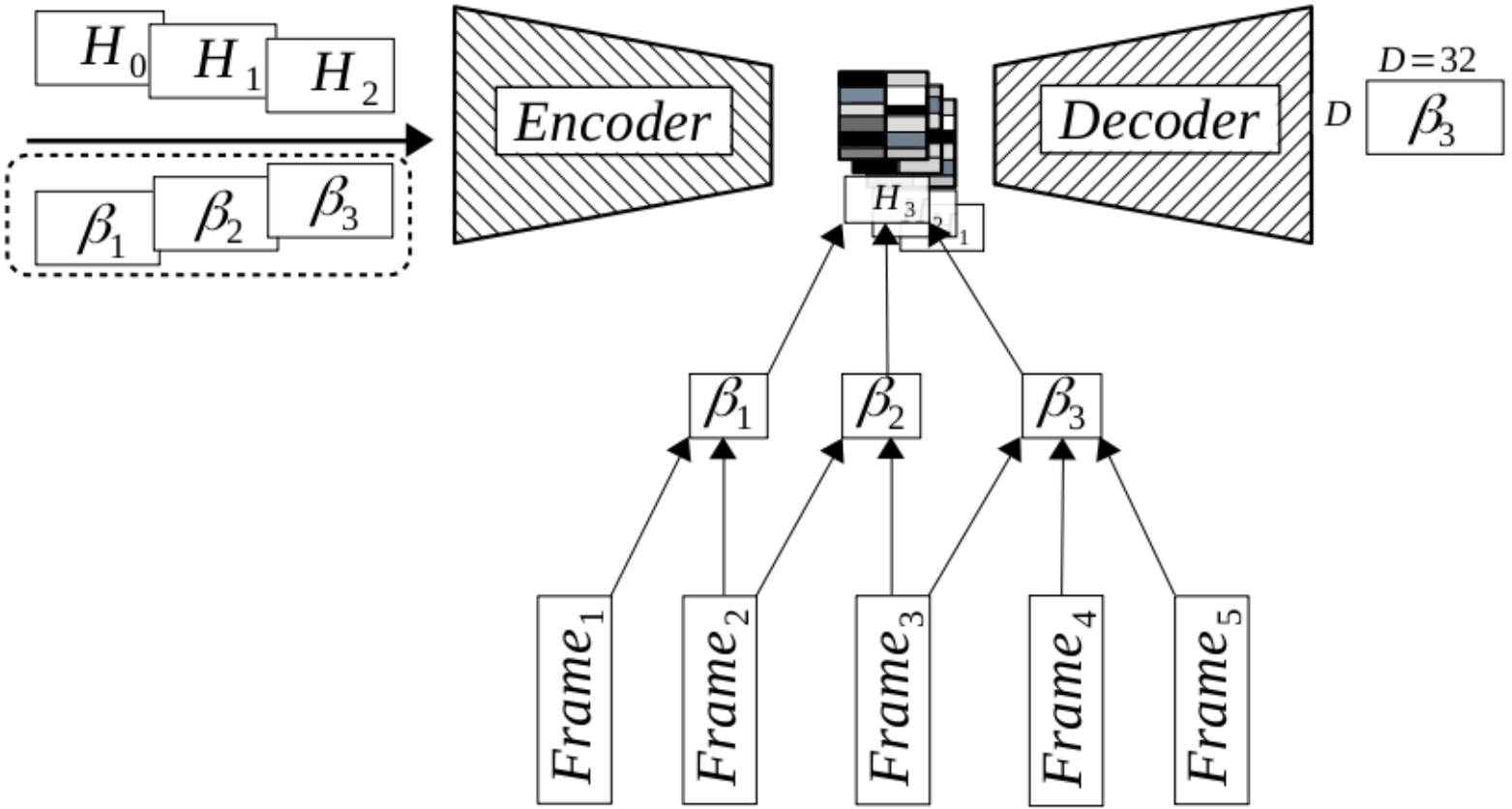

Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Compressed Representations
Trained AD
Alerts
Heatmap

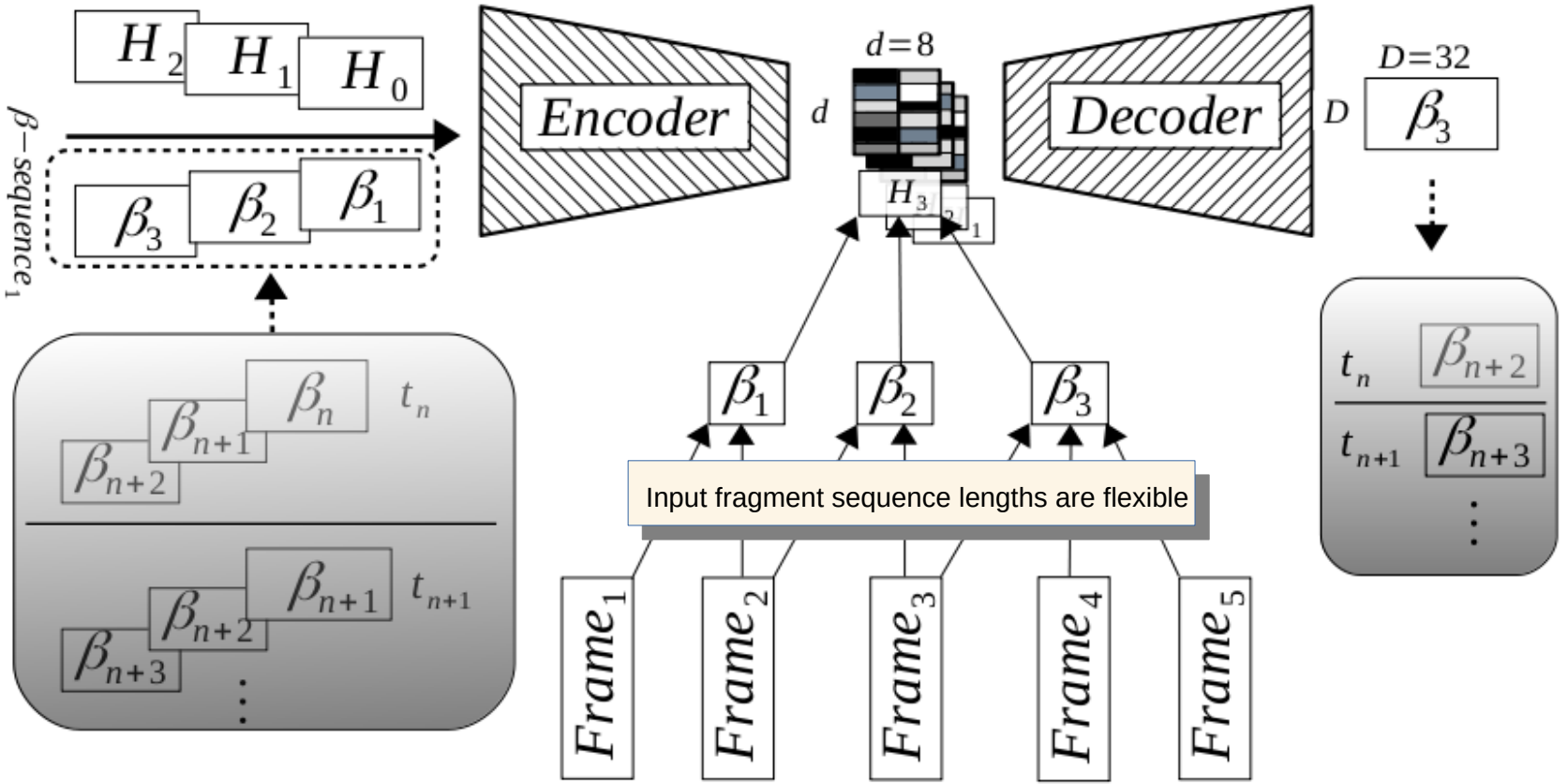

Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Compressed Representations
Trained AD
Alerts
Heatmap


β_1
$Frame_1$
$Frame_2$


H_0
β_1
Encoder
H_1
Decoder
β_1
$Frame_1$
$Frame_2$

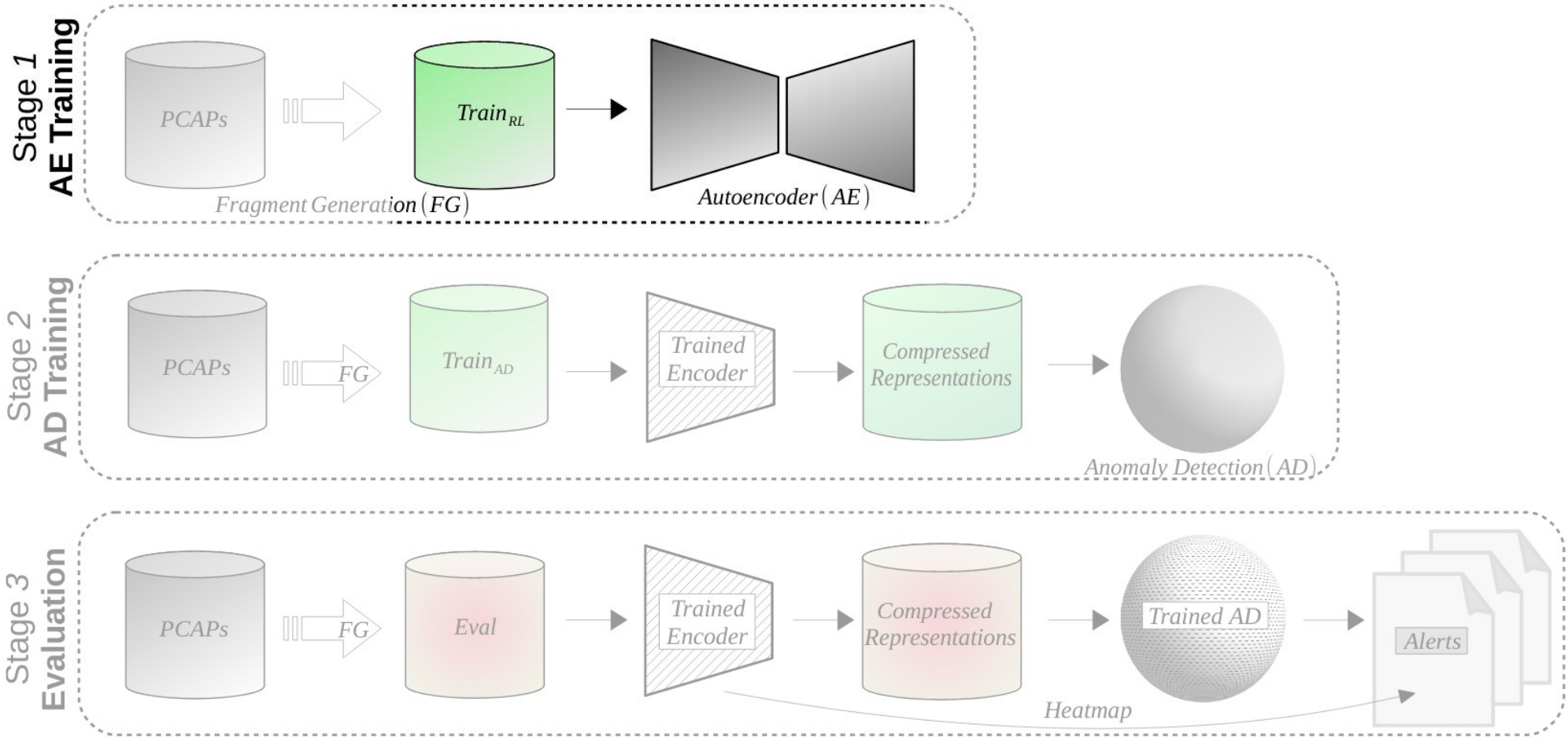

H_0
H_1
β_1
β_2
Encoder
Decoder
$H_2{}_1$
β_1
β_2
$Frame_1$
$Frame_2$
$Frame_3$

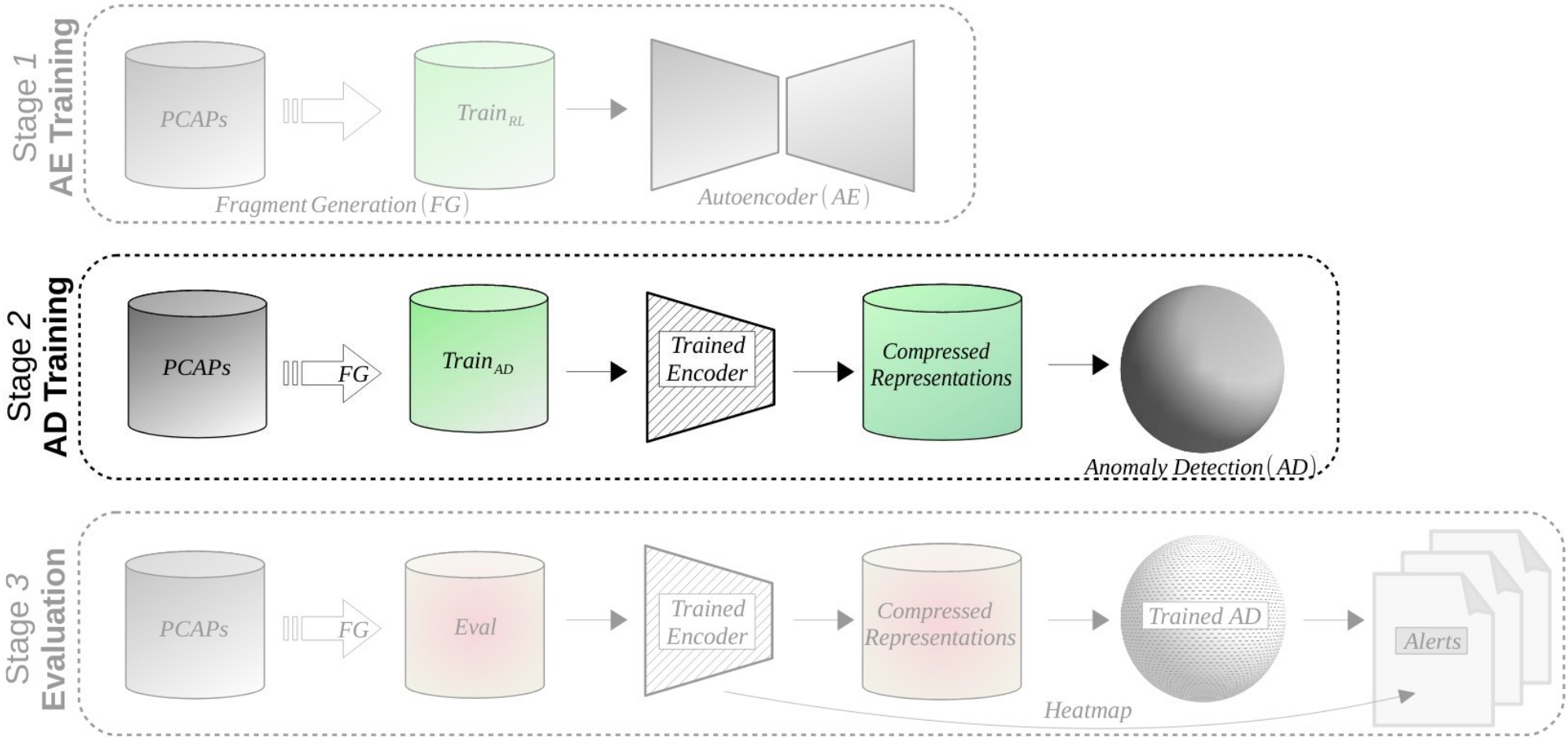

H_0
H_1
H_2
β_1
β_2
β_3
Encoder
H_3
Decoder
$D=32$
D
β_3
β_1
β_2
β_3
$Frame_1$
$Frame_2$
$Frame_3$
$Frame_4$
$Frame_5$

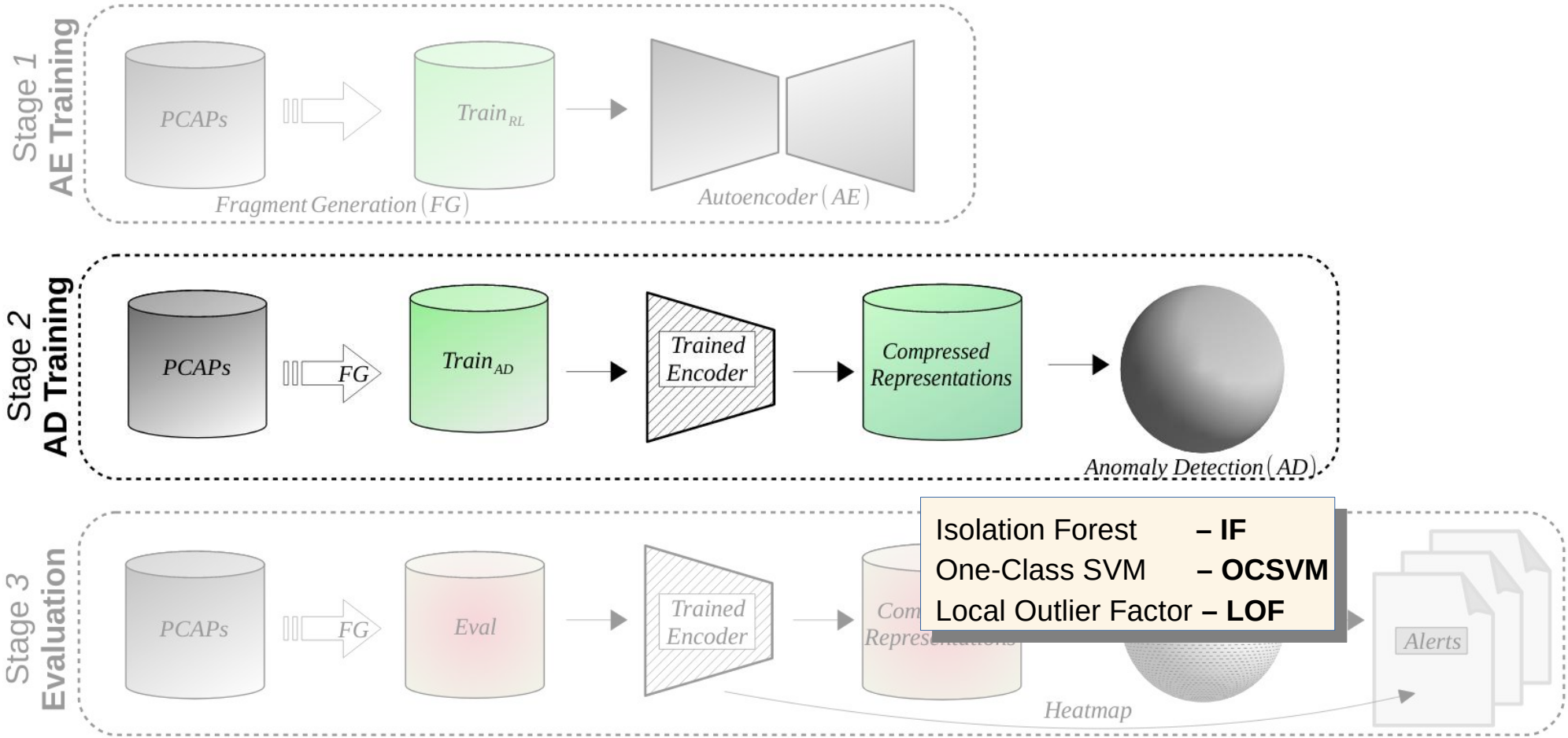

H_2 H_1 H_0
β−sequence$_1$
β_3 β_2 β_1
Encoder
d=8
d
H_3
Decoder
D
D=32
β_3
t_n β_{n+2}
t_{n+1} β_{n+3}
β_n β_{n+1} β_{n+2} t_n
β_{n+1} β_{n+2} β_{n+3} t_{n+1}
β_1 β_2 β_3
Input fragment sequence lengths are flexible
Frame$_1$ Frame$_2$ Frame$_3$ Frame$_4$ Frame$_5$

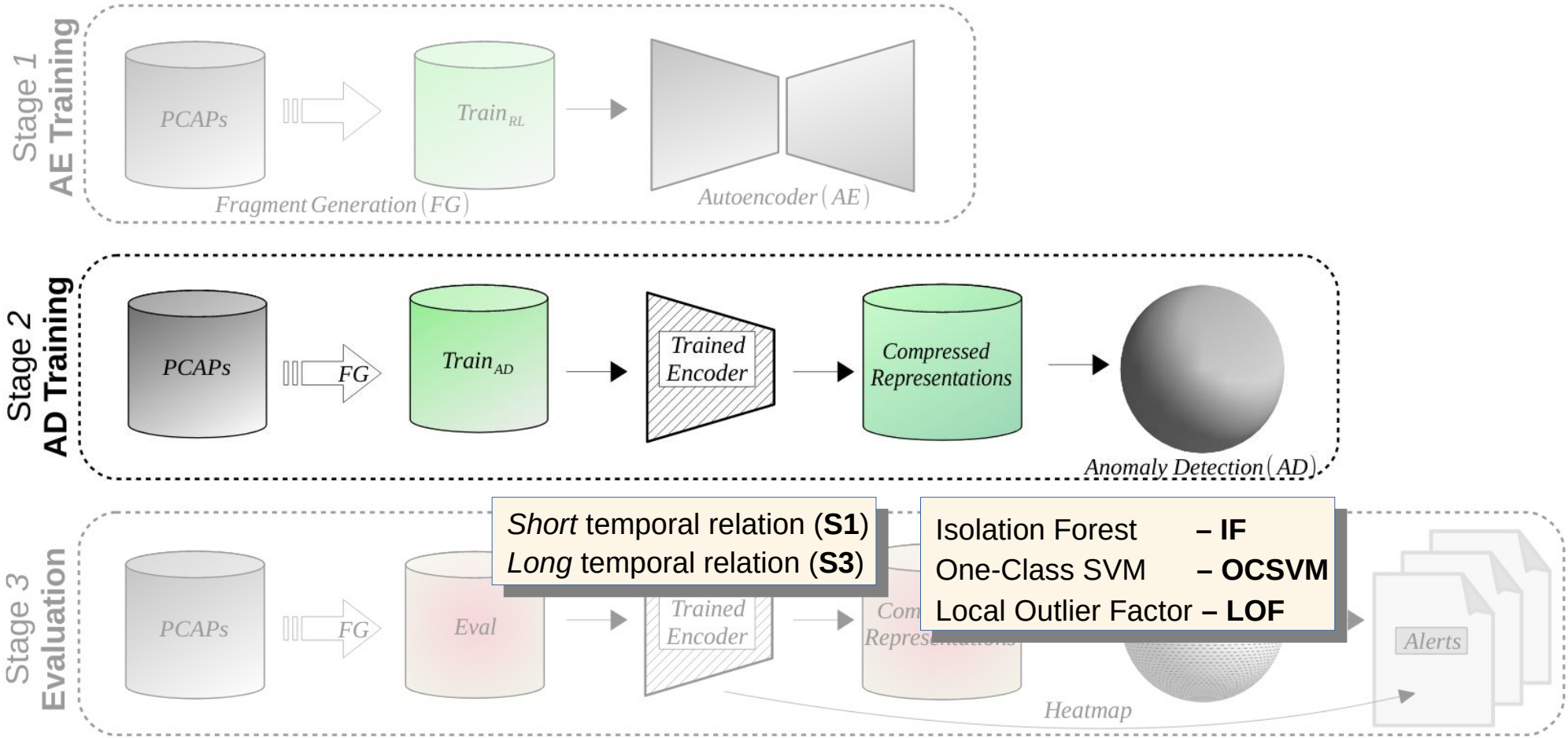
```
AutoEncoder(
  (encoder): Encoder(
    (stage1): Sequential(
      (conv1_leaky_1): Conv2d(in=1, out=2, kernel_size=(3, 3),
                              stride=(1, 1),
                              padding=(1, 1))
      (leaky_conv1_leaky_1): LeakyReLU(negative_slope=0.2, inplace=True)
    )
    (rnn1): CGRU_cell(
      (dropout): Dropout2d(p=0.1, inplace=False)
      (conv_gates): Sequential(
        (0): Conv2d(in=6, out=8, kernel_size=(5, 5),
                    stride=(1, 1),
                    padding=(2, 2))
        (1): GroupNorm(groups=8, channels=8, eps=1e-05, affine=True)
      )
      (conv_can): Sequential(
        (0): Conv2d(in=6, out=4, kernel_size=(5, 5),
                    stride=(1, 1),
                    padding=(2, 2))
        (1): GroupNorm(groups=4, channels=4, eps=1e-05, affine=True)
      )
    )
```

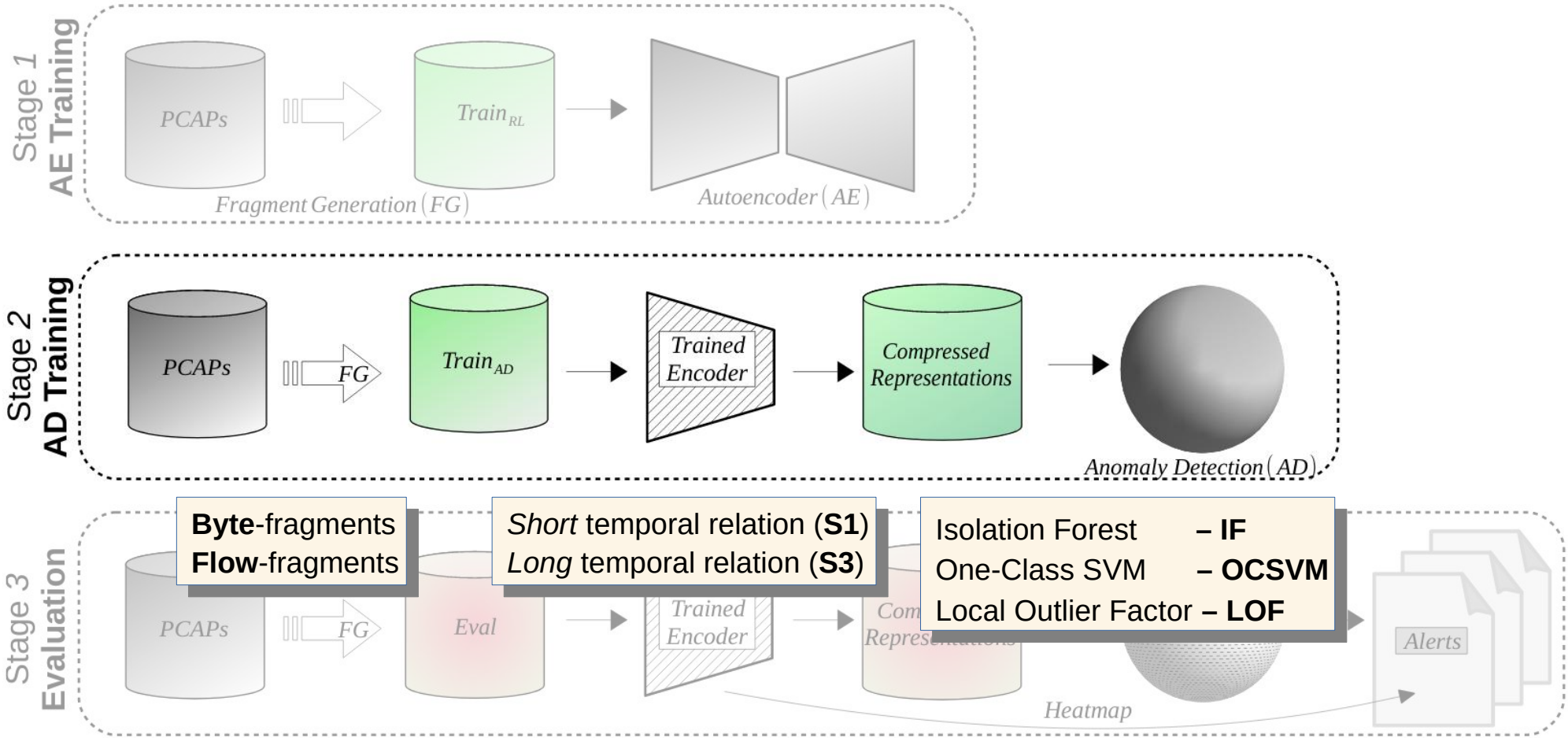
- Encoder compresses input data (32^2 bytes) to **64 features**
- **convGRU** (X. Shi *et al. – NIPS 2015*) neurons for each layer
- Trained to **reconstruct the last input** of an input sequence
- **ReLU** (V. *Nair et al. – ICML 2010*) as an activation function
- **Mean square error (MSE)** as a loss function
- **Cycle LR-scheduler** (L. *Smith et al. – SPIE 2019*)
- Gradient descent with **small batch sizes** (J. Nocedal et al. – ICLR 2017)

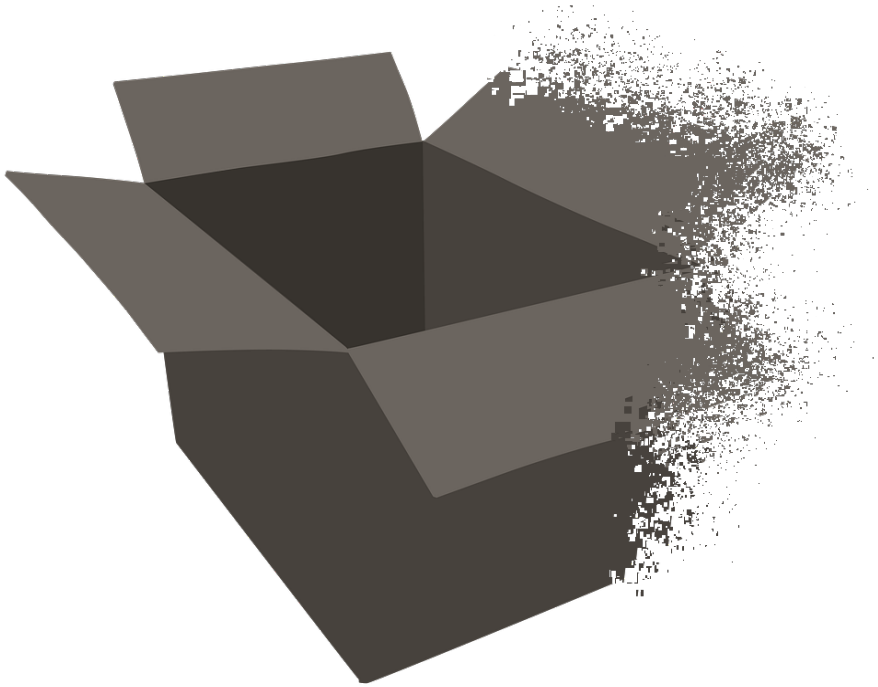

Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Compressed Representations
Trained AD
Alerts
Heatmap


Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Compressed Representations
Trained AD
Alerts
Heatmap

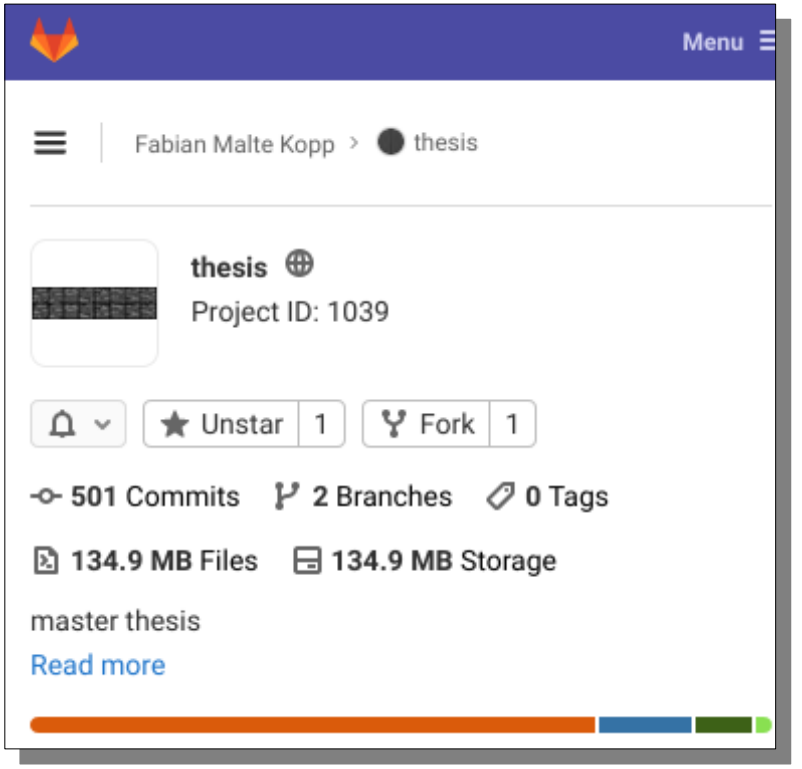

Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Heatmap
Alerts
Isolation Forest – IF
One-Class SVM – OCSVM
Local Outlier Factor – LOF

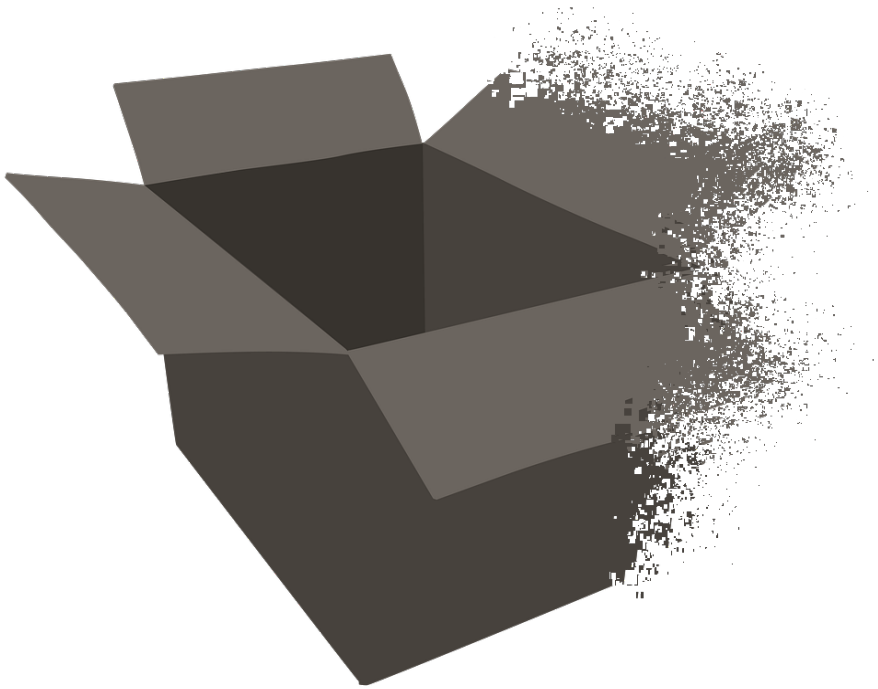

Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Heatmap
Alerts
Short temporal relation (S1)
Long temporal relation (S3)
Isolation Forest – IF
One-Class SVM – OCSVM
Local Outlier Factor – LOF

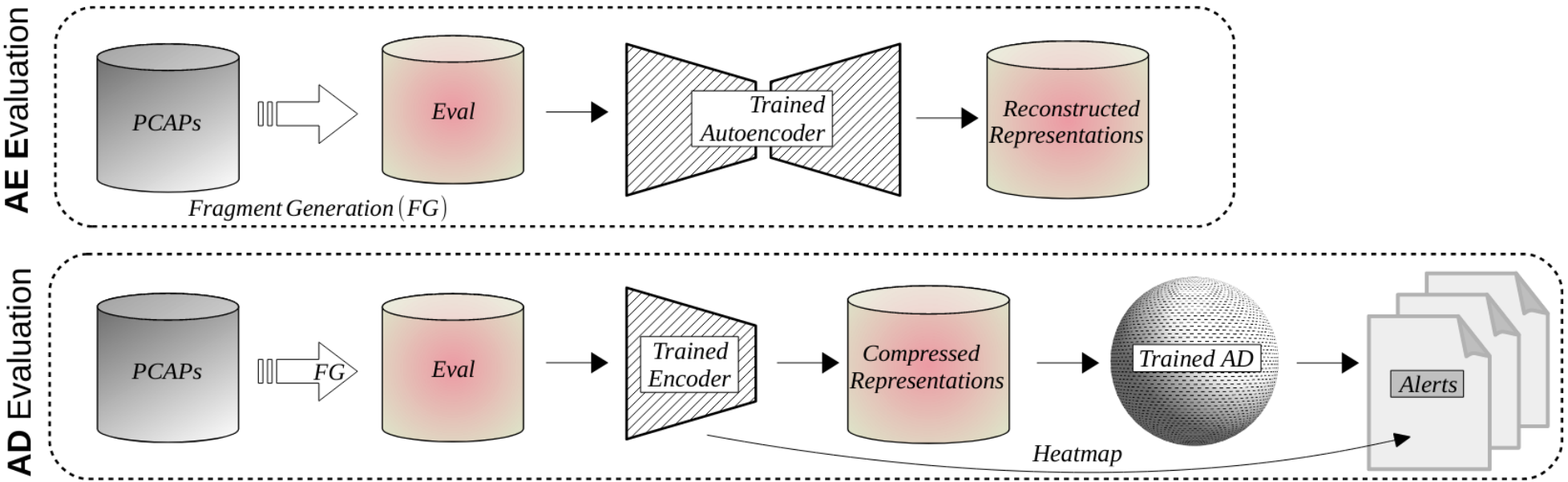

Stage 1
AE Training
PCAPs
Train RL
Fragment Generation (FG)
Autoencoder (AE)
Stage 2
AD Training
PCAPs
FG
Train AD
Trained Encoder
Compressed Representations
Anomaly Detection (AD)
Stage 3
Evaluation
PCAPs
FG
Eval
Trained Encoder
Heatmap
Alerts
Byte-fragments
Flow-fragments
Short temporal relation (S1)
Long temporal relation (S3)
Isolation Forest – IF
One-Class SVM – OCSVM
Local Outlier Factor – LOF



- Multi-threaded PCAP converter based on dpkt
  - Processes ~2,000 Ethernet frames/sec per computing core
  - Data is stored in HDF5 containers (~10% overhead)
- PyTorch-based GPU-enabled model
  - Added integration for convGRU layers
- Scikit-learn-based anomaly detection integration
- Integrated LRP implementation for model interpretabilty
- Experiment tracking and logging via Tensorboard
  - tensorboard.dev/experiment/OTD0qLBeQYK1DajlvkwoyQ/#graphs
- Developed and maintained via gitlab
  - git.informatik.tu-cottbus.de/koppfabi/thesis
  - Supplementary Jupyter notebooks for simple tutorials
  - ~3200 LOC / 500 commits






AE Evaluation
PCAPs
Eval
Trained Autoencoder
Reconstructed Representations
Fragment Generation (FG)
AD Evaluation
PCAPs
FG
Eval
Trained Encoder
Compressed Representations
Trained AD
Alerts
Heatmap

- Industrial control system data
  - Power Plant traffic (BTU research data set, NYD)
  - Secure Water Treatment (*Tippenhauer – CySWater 2016*)

- Industrial control system data
  - Power Plant traffic (BTU research data set, NYD)
  - Secure Water Treatment (*Tippenhauer – CySWater 2016*)
- Ground truth information
  - Estimated ground truth
  - Synthetic data generation (*Hanusch – BTU 2018)*

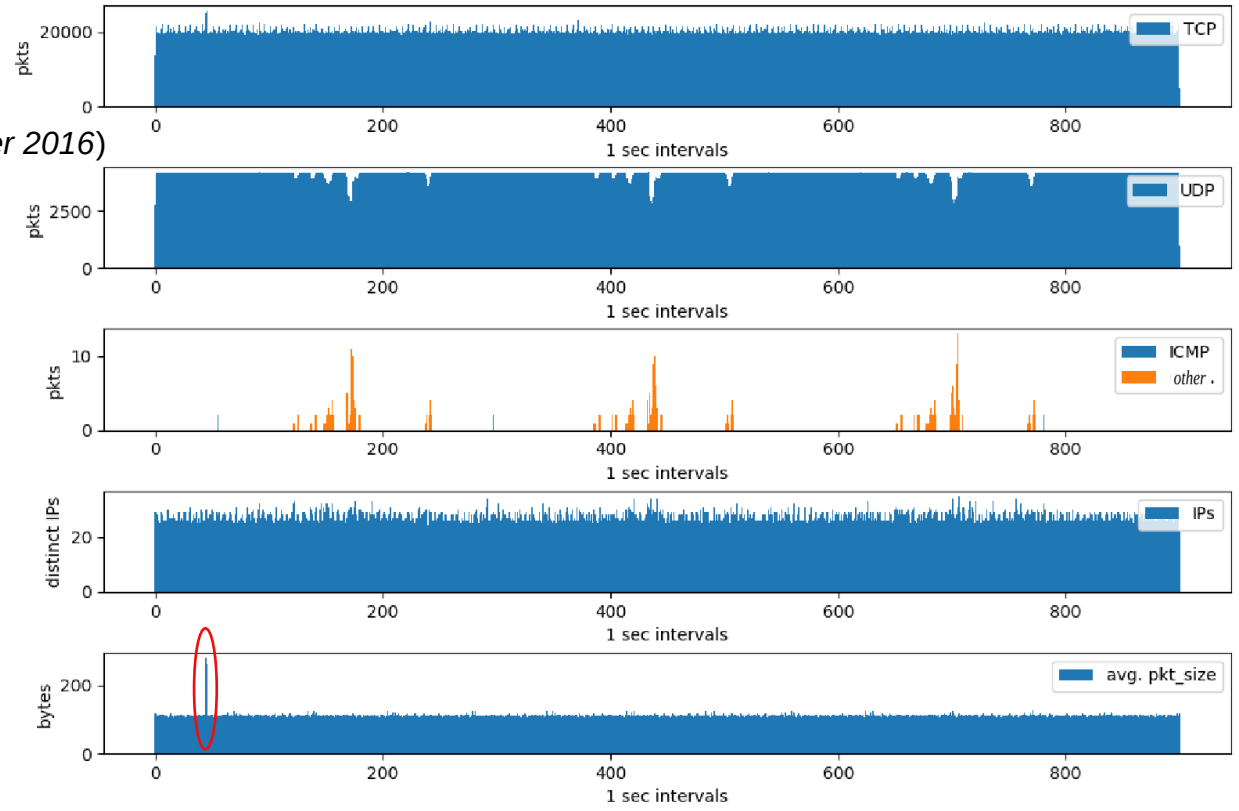


Figure 4.1: High-level traffic analysis of the SWaT data set

- Industrial control system data
  - Power Plant traffic (BTU research data set, NYD)
  - Secure Water Treatment (*Tippenhauer – CySWater 2016*)
- Ground truth information
  - Estimated ground truth
  - Synthetic data generation (*Hanusch – BTU 2018)*
- Data splitting
  - ~ 1.5 GB/3 min of training data (10M frames)
  - 10% for training validation

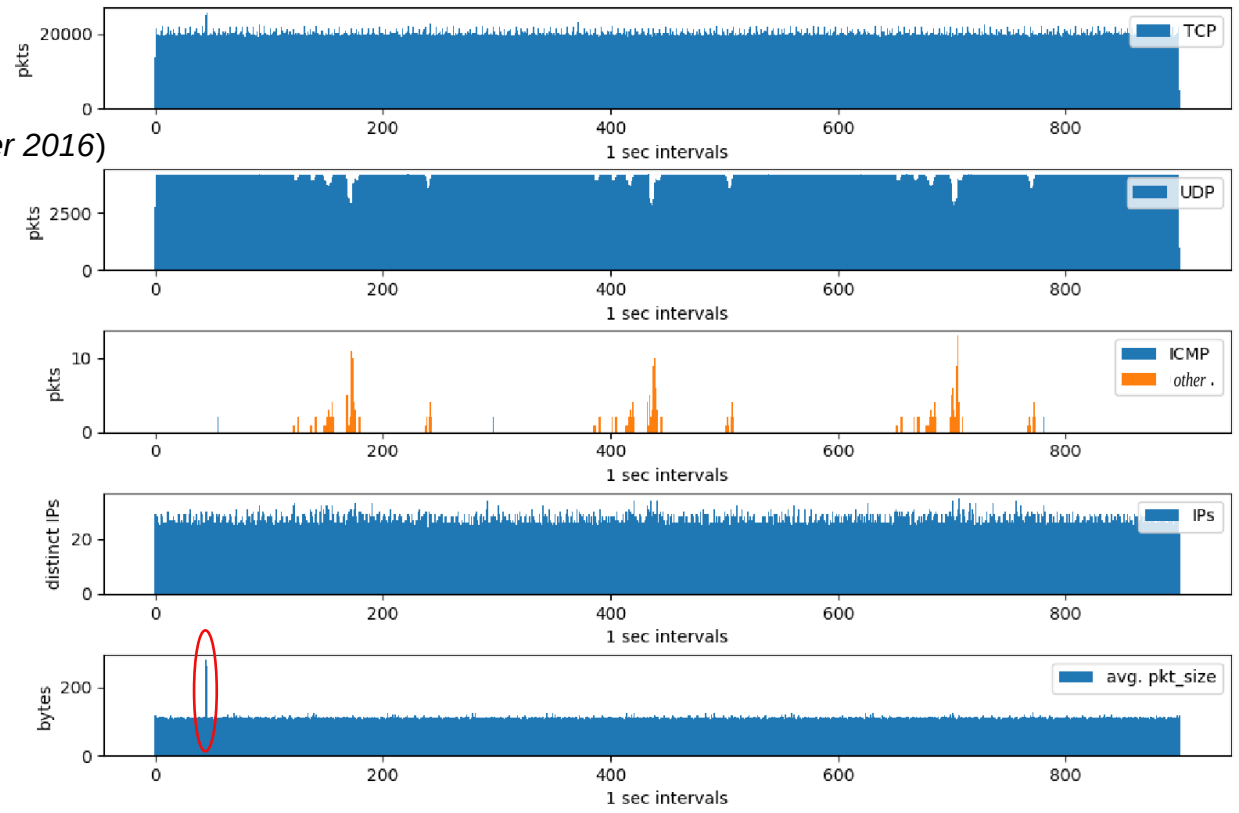


Figure 4.1: High-level traffic analysis of the SWaT data set

- Industrial control system data
  - Power Plant traffic (BTU research data set, NYD)
  - Secure Water Treatment (*Tippenhauer – CySWater 2016*)
- Ground truth information
  - Estimated ground truth
  - Synthetic data generation (*Hanusch – BTU 2018)*
- Data splitting
  - ~ 1.5 GB/3 min of training data (10M frames)
  - 10% for training validation
- Experiment trails
  - 2 input data orientations (*byte- & flow-frag.*)
  - 2 feature extraction models (*S1* & *S3*)
  - 4 evaluation sets per data set (H, D, E, R)
  - 3+1 anomaly detection methods (IF, OCSVM, LOF, BL)

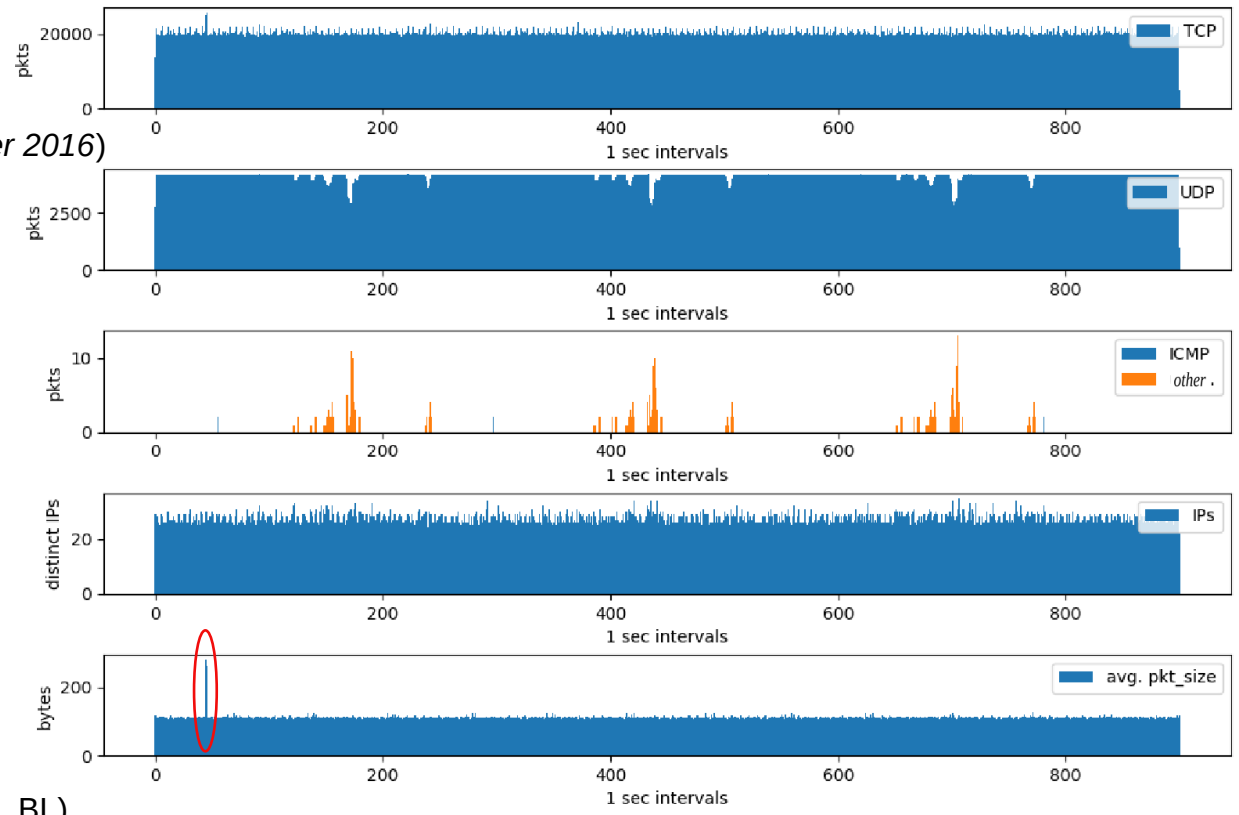
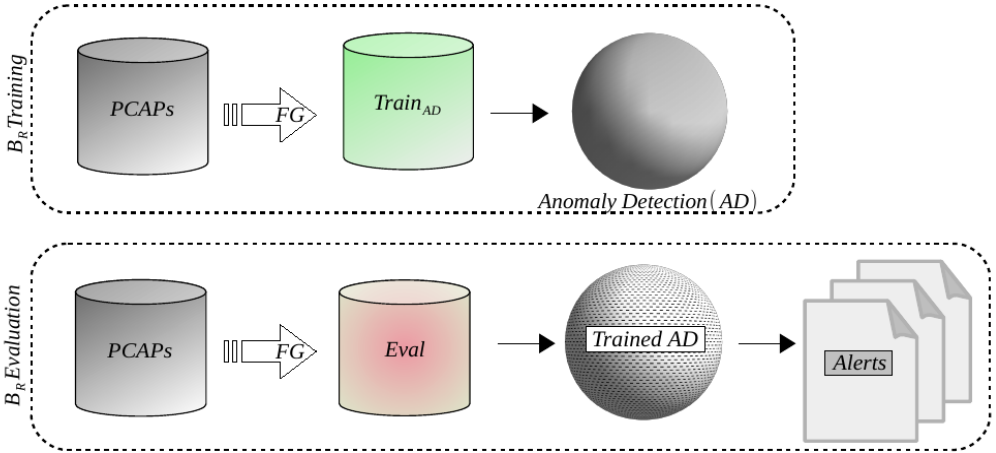


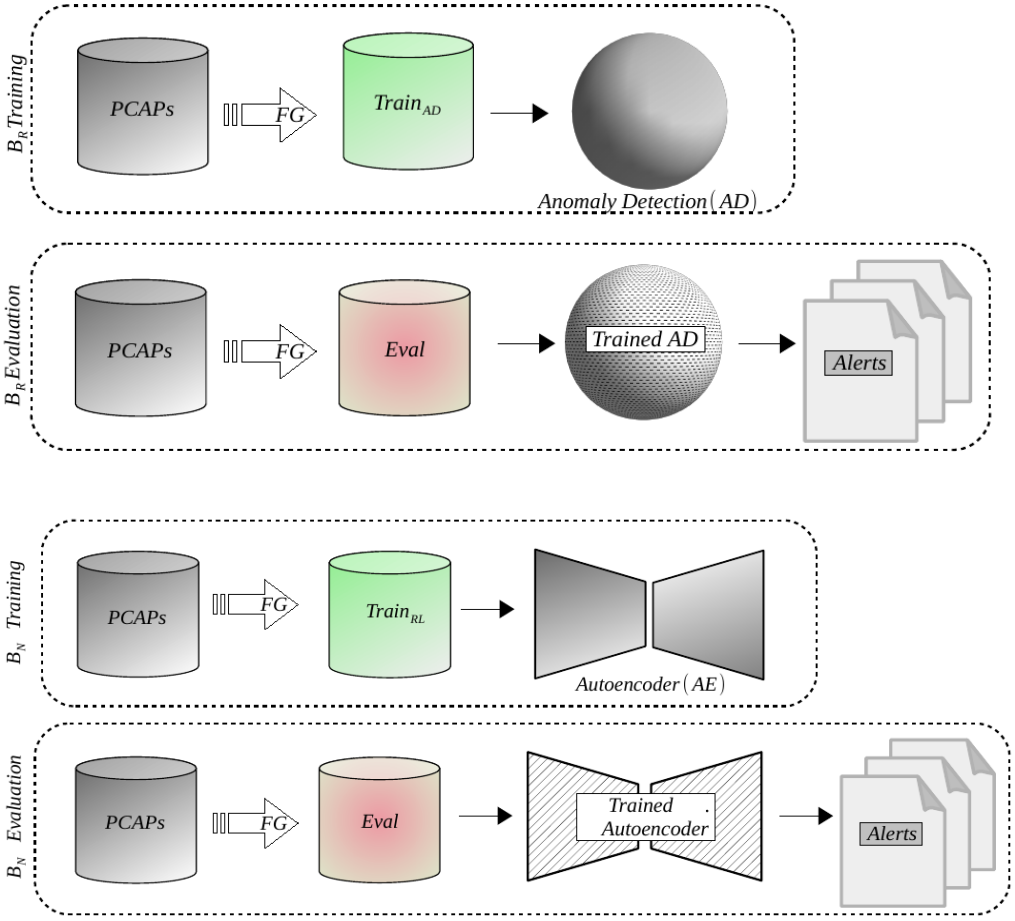
Figure 4.1: High-level traffic analysis of the SWaT data set

- Raw Baseline (B_r)
  - No additional feature extraction
  - Anomaly detection based on raw byte- or flow-frag.



- Raw Baseline (B_r)
  - No additional feature extraction
  - Anomaly detection based on raw byte- or flow-frag.

- Naive Baseline (B_n)
  - Feature extraction as an end-to-end model
  - Anomaly detection based on the model's loss values
  - Upper bound for normal behavior based on the average loss (*AML*) plus a multiple (ν) of its standard deviation (σ)
    - $thr := AML + \nu\sigma$
    - $MSE(x, \hat{x}) > thr\ ?\ \ x \in A_{pred} :\ \ x \in N_{pred}$
    - For $\nu = 2.5 \Rightarrow 98{,}9\%$ of normal examples are classified correctly (normal distribution)



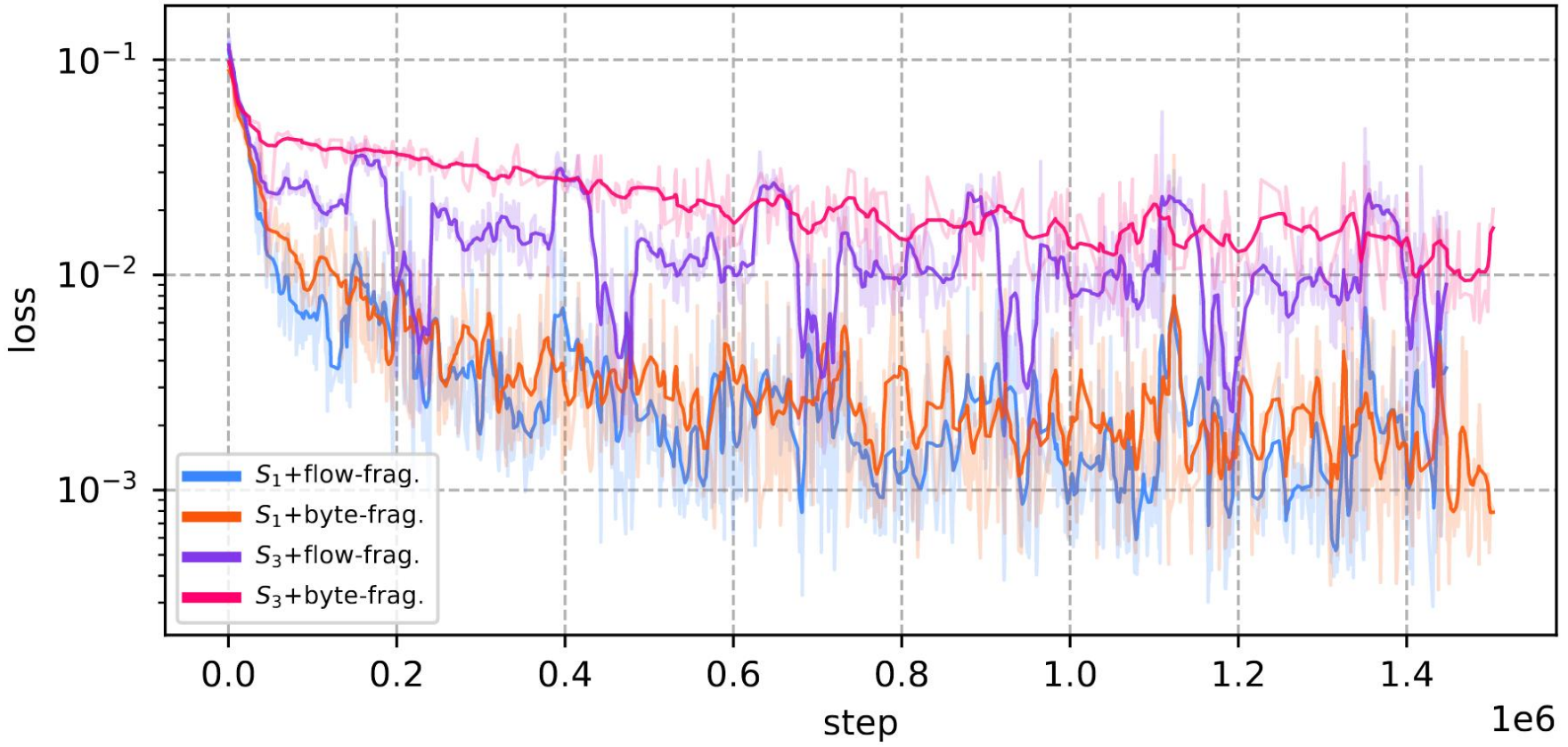


Figure 6.2: Training loss-curves for the 4 different trained models for the SWaT A6 data set

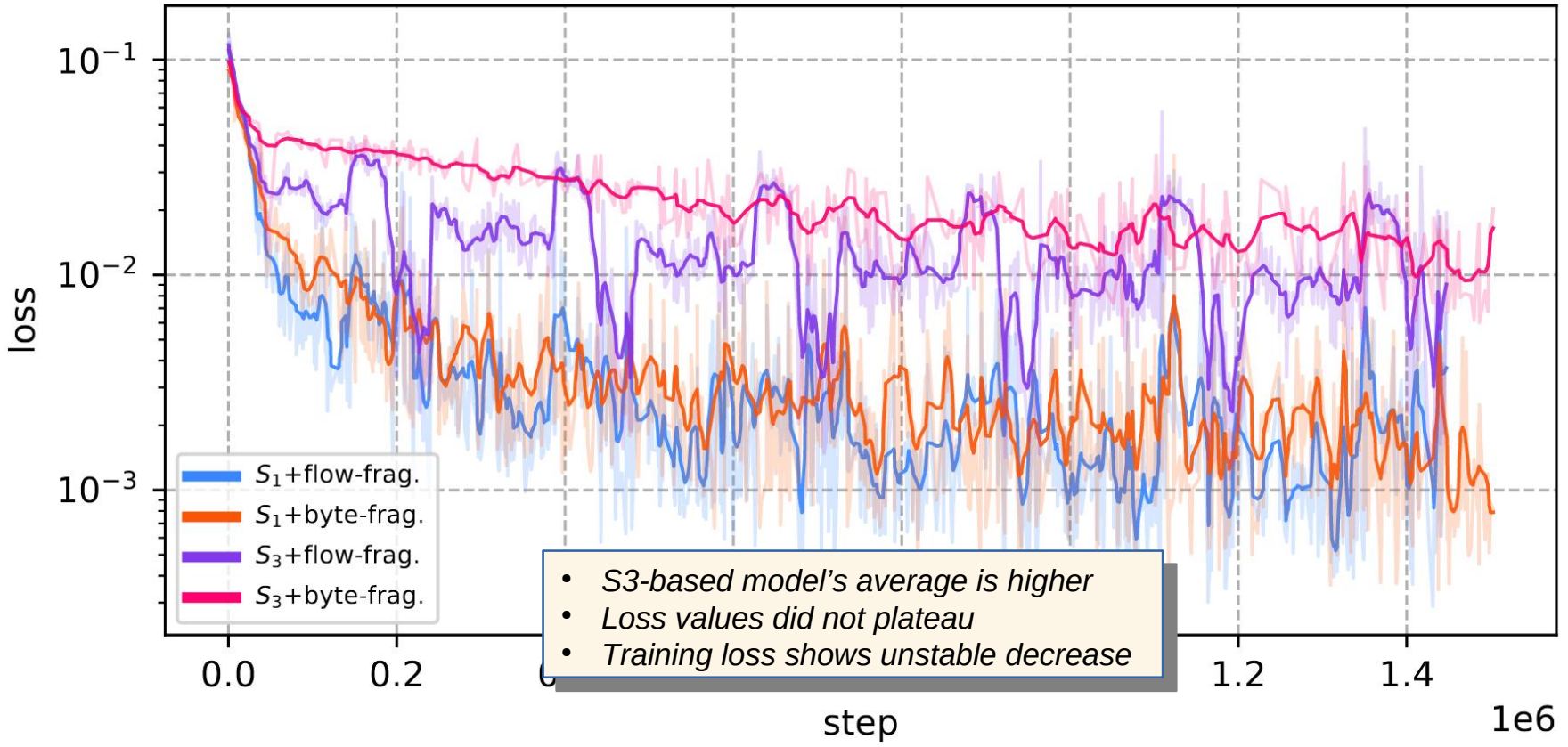


Figure 6.2: Training loss-curves for the 4 different trained models for the SWaT A6 data set

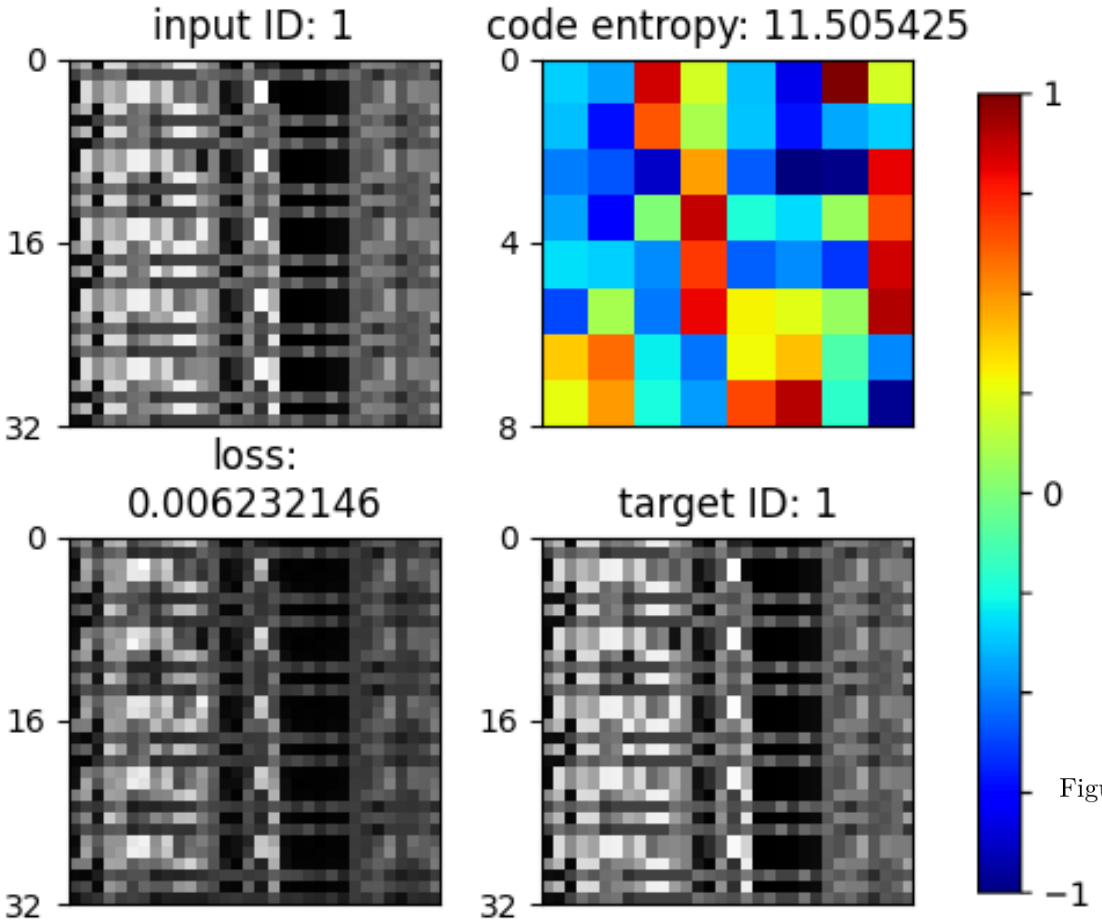


Figure 6.6: Input fragment (top left), extracted features (top right), reconstructed fragment (lower left), target fragment (lower right)

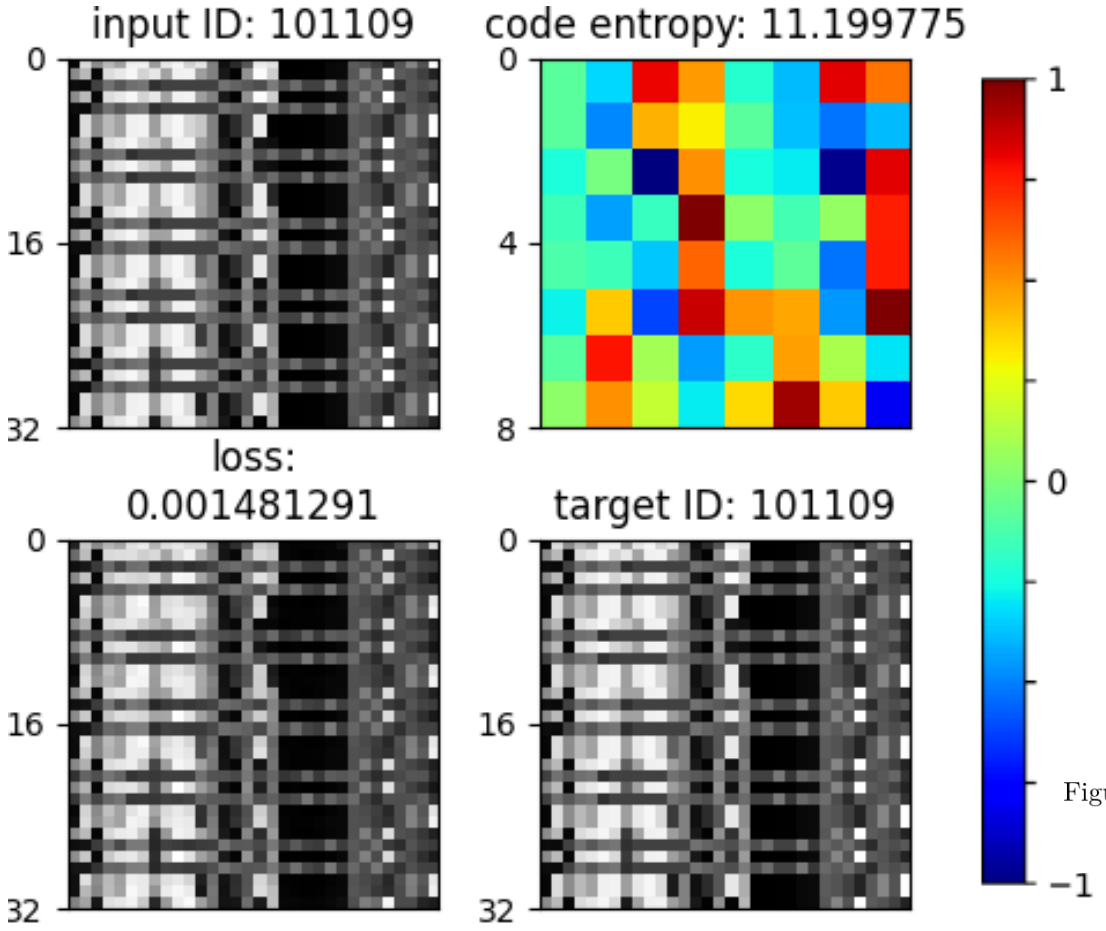
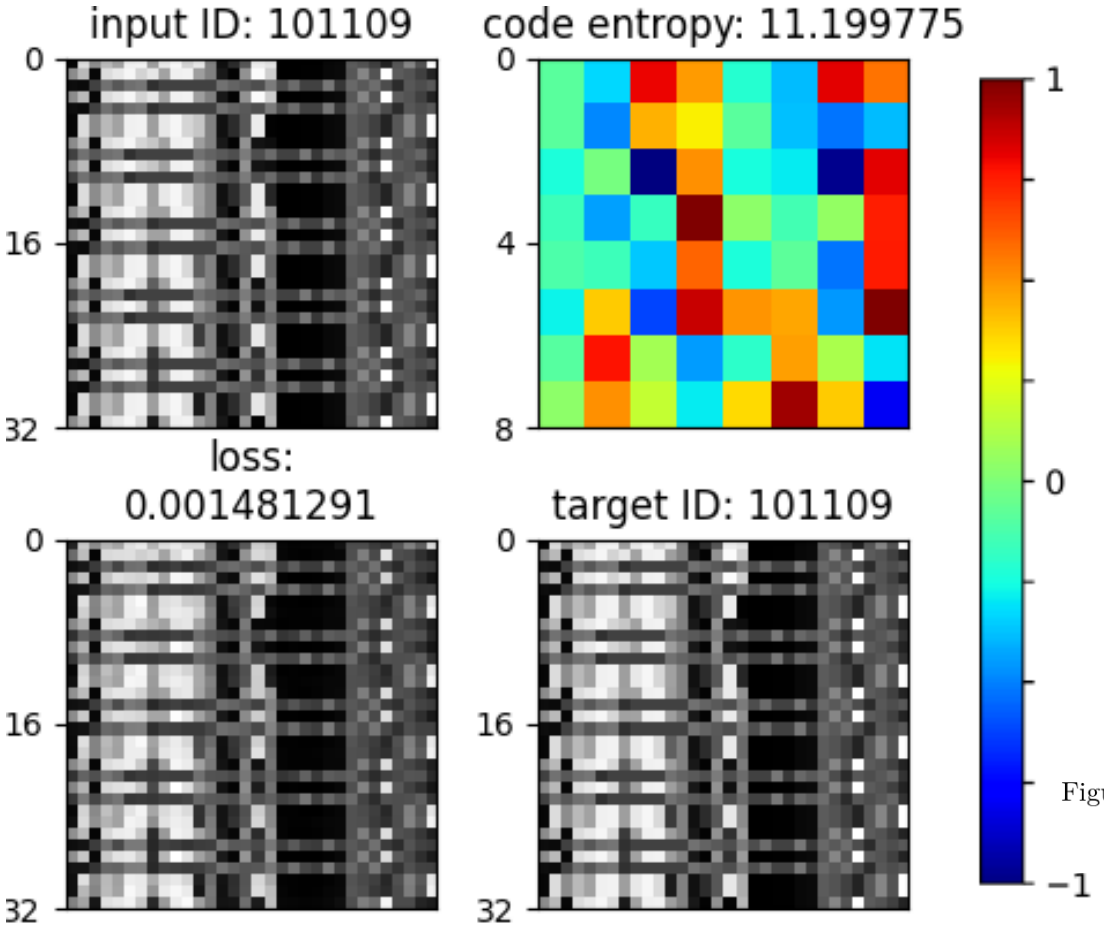
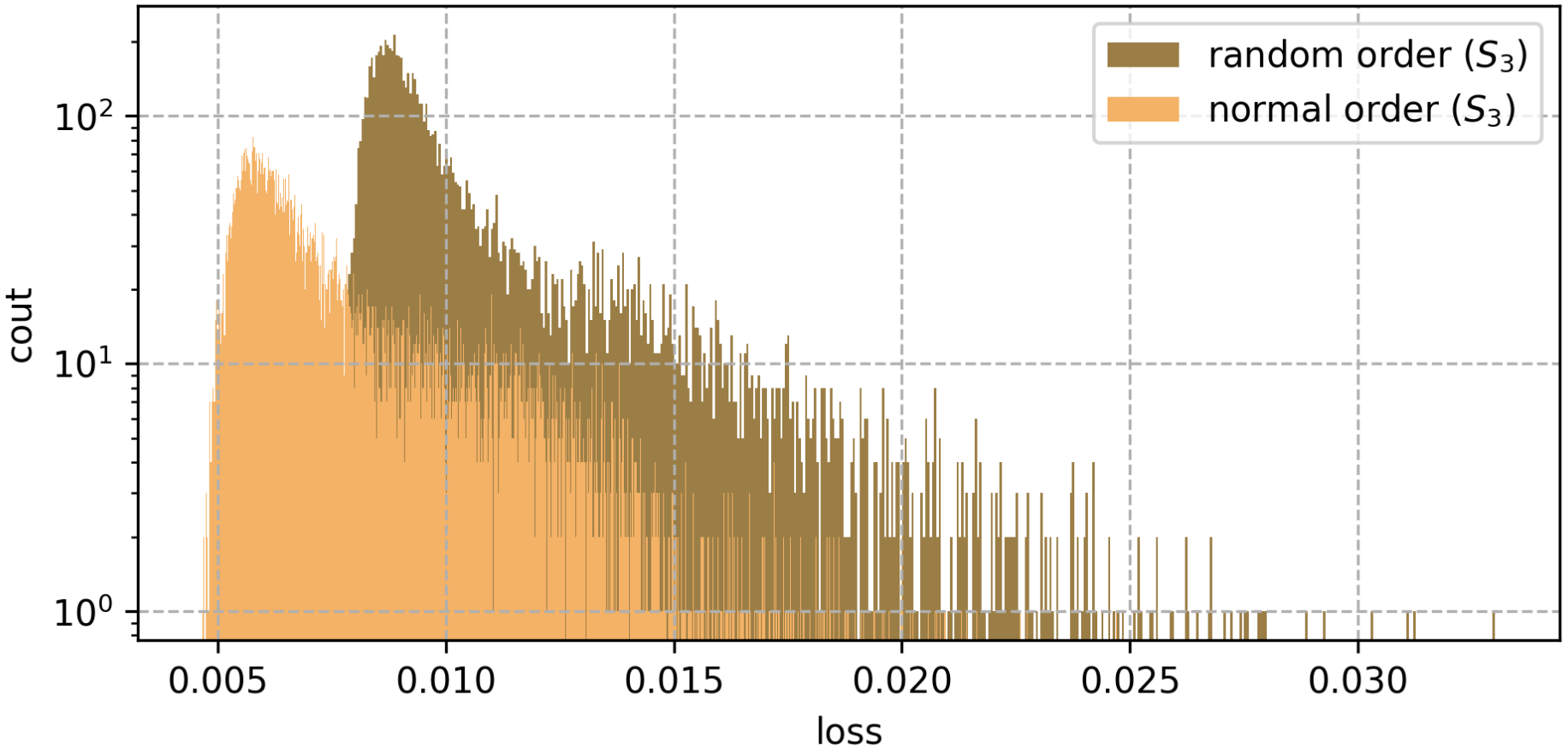


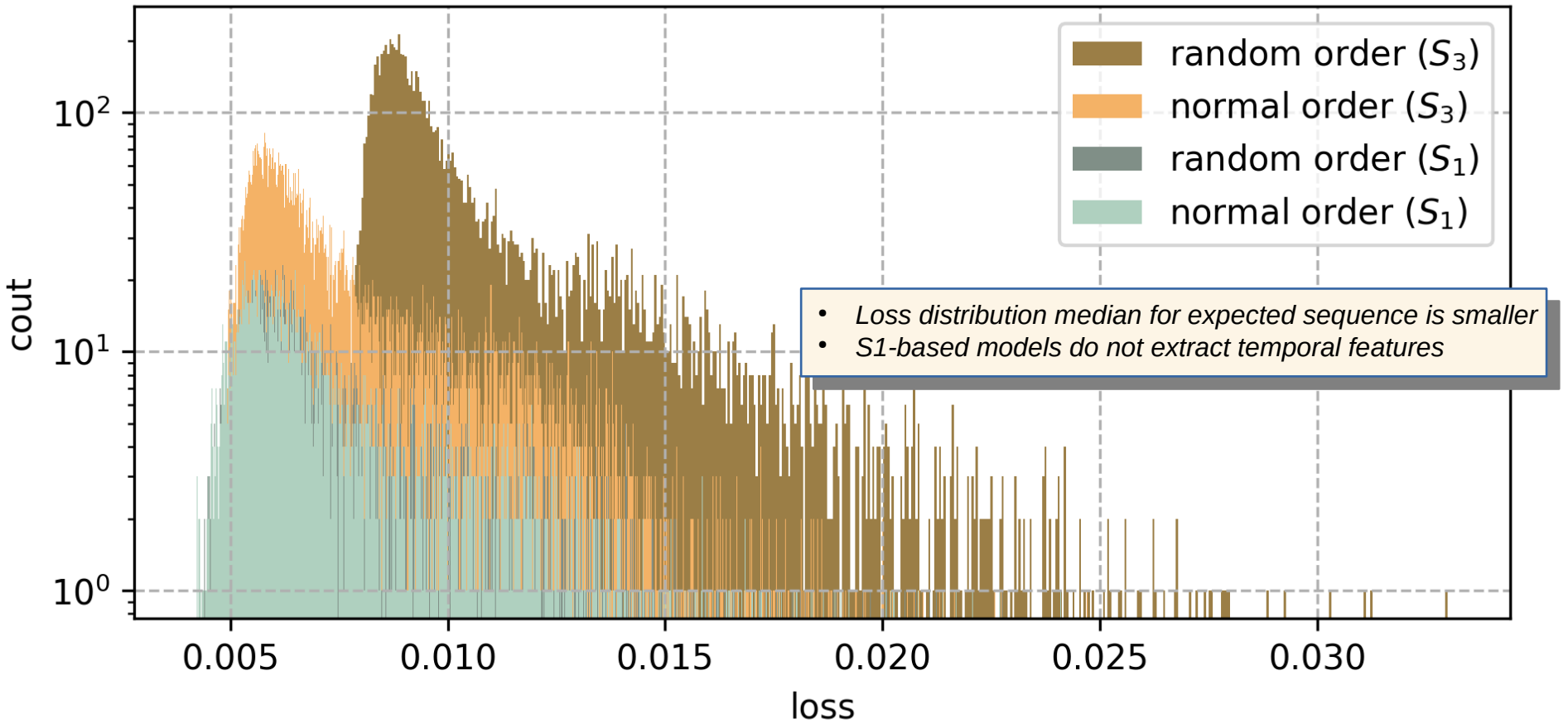
Figure 6.6: Input fragment (top left), extracted features (top right), reconstructed fragment (lower left), target fragment (lower right)



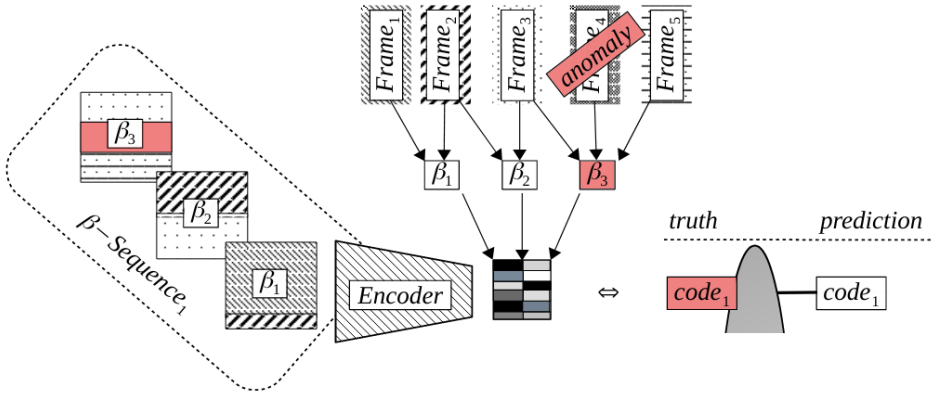
- *Spatial features are reconstructed accurately*
- *Extracted features are structured & not intuitive*

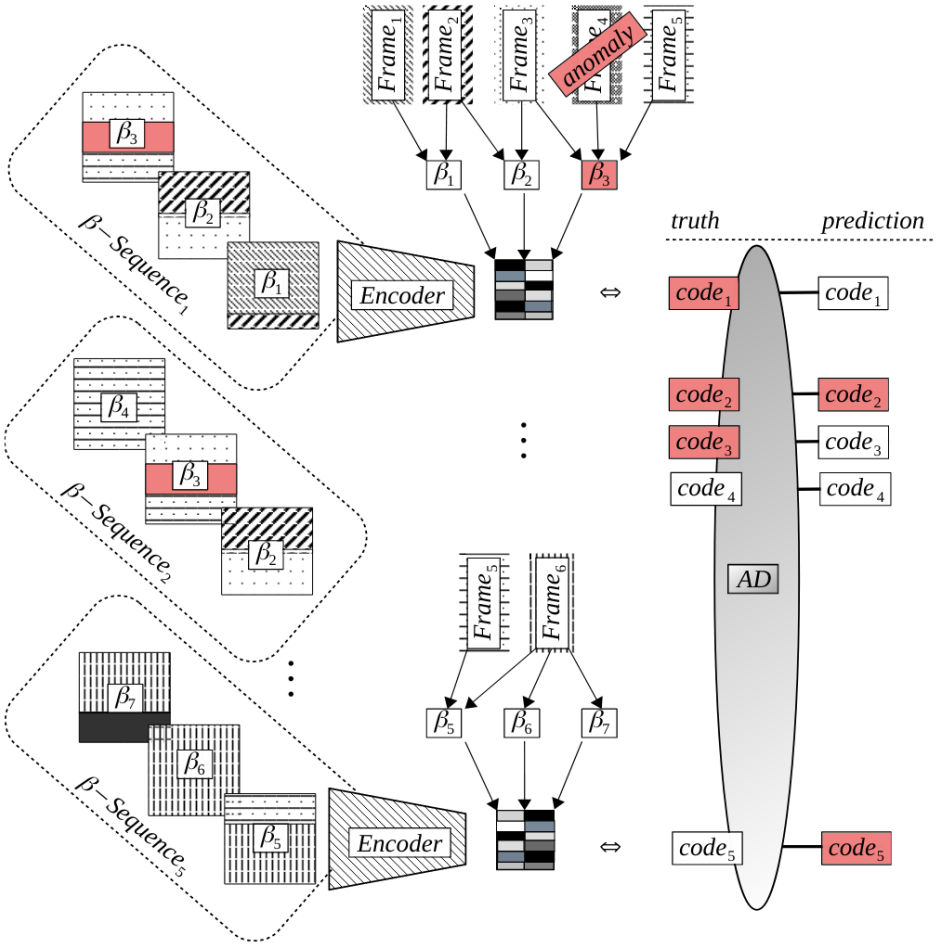
Figure 6.6: Input fragment (top left), extracted features (top right), reconstructed fragment (lower left), target fragment (lower right)

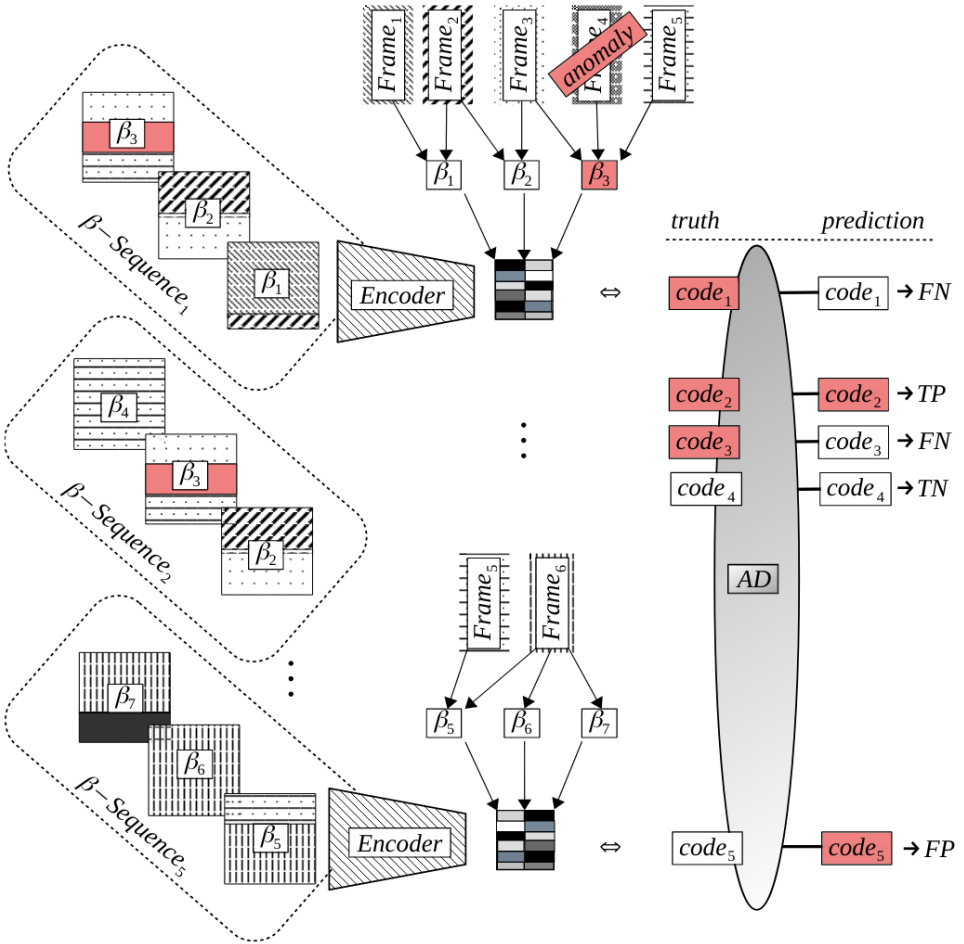

random order (S_3)
normal order (S_3)
cout
10^2
10^1
10^0
0.005
0.010
0.015
0.020
0.025
0.030
loss



- *Loss distribution median for expected sequence is smaller*
- *S1-based models do not extract temporal features*


Frame1
Frame2
Frame3
anomaly
Frame5
β3
β2
β1
β−Sequence1
Encoder
β1
β2
β3
truth
prediction
code1
code1


Frame1
Frame2
Frame3
anomaly
Frame5
β3
β2
β1
β−Sequence1
Encoder
β1
β2
β3
truth
prediction
code1
code1
code2
code2
code3
code3
code4
code4
AD
β4
β3
β2
β−Sequence2
Frame5
Frame6
β5
β6
β7
β7
β6
β5
β−Sequence5
Encoder
code5
code5

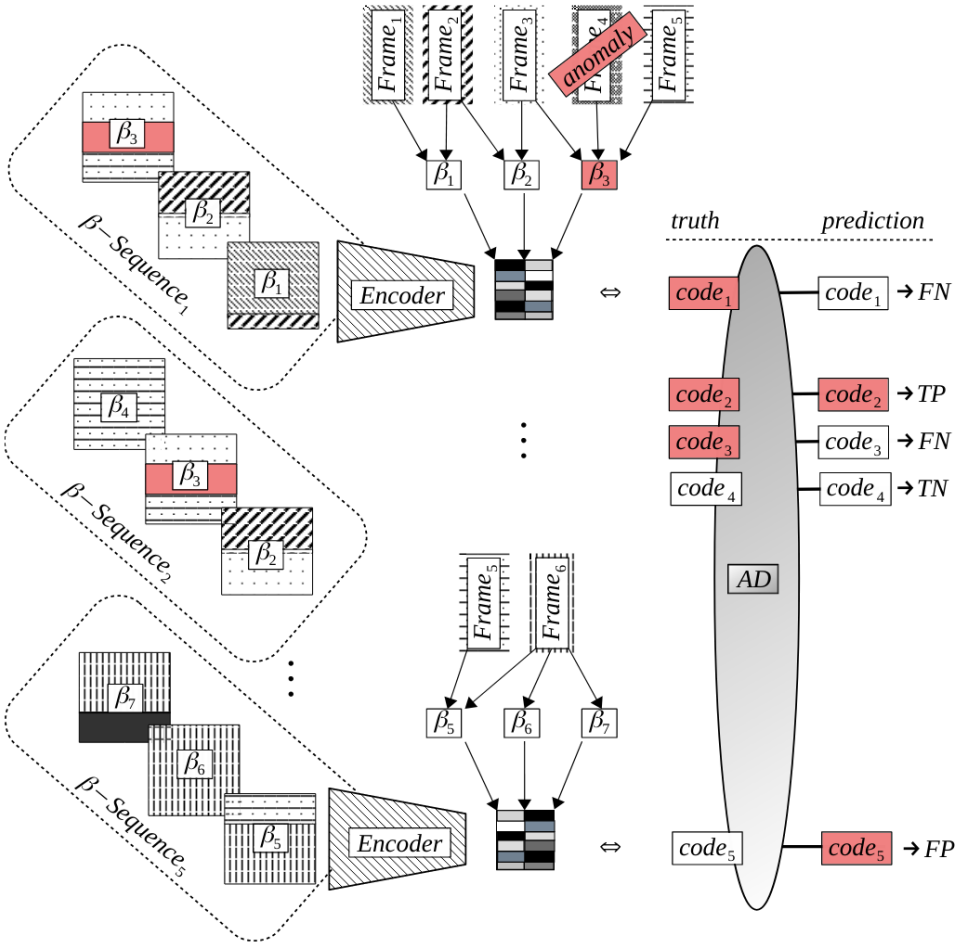


False Negatives – **FN**

True Positives – **TP**

True Negatives – **TN**

False Positives – **FP**



False Negatives – **FN**

True Positives – **TP**

True Negatives – **TN**

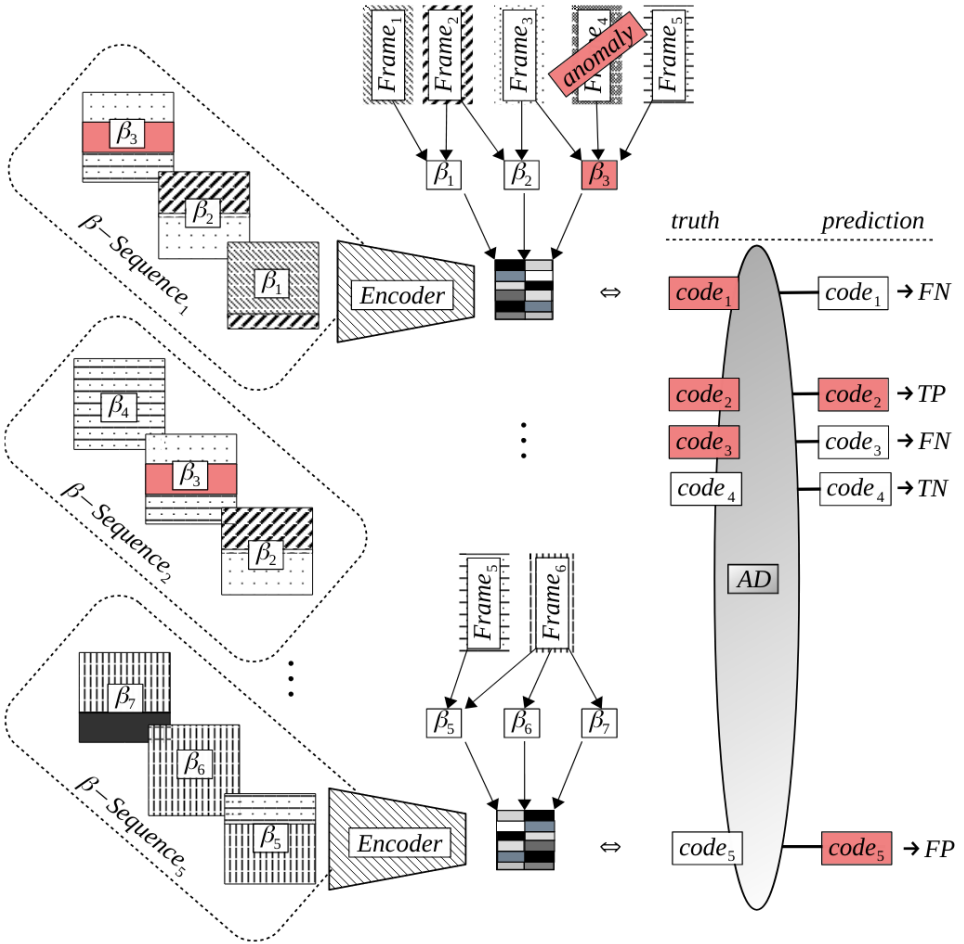
False Positives – **FP**

Recall – **RC**

$$RC = \frac{|TP|}{|TP|+|FN|}$$

Precision – **PR**

$$PR = \frac{|TP|}{|TP|+|FP|}$$



False Negatives – **FN**

True Positives – **TP**

True Negatives – **TN**

False Positives – **FP**

Recall – **RC**

$$RC = \frac{|TP|}{|TP|+|FN|}$$

Precision – **PR**

$$PR = \frac{|TP|}{|TP|+|FP|}$$

F1-Score

$$F_1 = 2\times\left(\frac{|PR|*|RC|}{|PR|+|RC|}\right)$$

False Pos. Rate – **FPR**

$$FPR = \frac{|FP|}{|FP|+|TN|}$$

Brandenburg University of Technology Cottbus - Senftenberg

Fragment	S_n	Split	Time	#Samples	#Anomalies
Byte				SWaT	
	S_1				
		$Train$	01:17:12	1,107,198	0
		$Validation$	00:00:44	10,990	0
		$Eval_H$	01:06:35	821,442	11,208
		$Eval_E$	00:42:25	684,501	288
		$Eval_D$	00:40:44	697,304	25,883
		$Eval_R$	00:49:32	684,426	101
	S_3				
		$Train$	03:06:50	1,107,196	0
		$Validation$	00:01:47	10,988	0
		$Eval_H$	02:12:25	821,440	11,210
		$Eval_E$	01:35:33	684,499	578
		$Eval_D$	01:32:32	697,302	35,642
		$Eval_R$	00:49:32	684,424	232

Fragment	S_n	Split	Time	#Samples	#Anomalies
Flow				SWaT	
	S_1				
		$Train$	00:27:18	482,280	0
		$Validation$	00:00:16	4624	0
		$Eval_H$	00:22:09	346,786	1,123
		$Eval_E$	00:20:37	291,373	29
		$Eval_D$	00:18:03	295,128	3,776
		$Eval_R$	00:18:45	291,443	14
	S_3				
		$Train$	00:28:08	482,278	0
		$Validation$	00:00:16	4,622	0
		$Eval_H$	00:21:33	346,784	1,349
		$Eval_E$	00:45:10	291,371	55
		$Eval_D$	00:38:51	295,126	3,808
		$Eval_R$	00:17:32	291,441	42

- *Fragment generation condescend the total number of samples*
- *Fewer Flow-Fragments were extracted*
- *Feature extraction took longer on for models trained on longer sequences*

AD
IF
OCSVM
LOF
B_n

AD	Fragments
IF	
	Byte
	Flow
OCSVM	
	Byte
	Flow
LOF	
	Byte
	Flow
B_n	
	Byte
	Flow

AD	Fragments	Model
IF		
	Byte	
		S_1
		S_3
		B_r
	Flow	
		S_1
		S_3
		B_r
OCSVM		
	Byte	
		S_1
		S_3
		B_r
	Flow	
		S_1
		S_3
		B_r
LOF		
	Byte	
		S_1
		S_3
		B_r
	Flow	
		S_1
		S_3
		B_r
B_n		
	Byte	
		S_1
		S_3
	Flow	
		S_1
		S_3

AD	Fragments	Model	T2F↓	FPR↓
IF				
	Byte			
		S_1	00:07:31	0
		S_3	00:05:58	0
		B_r	00:45:14	0.0001
	Flow			
		S_1	00:05:15	0
		S_3	00:05:11	0
		B_r	00:36:12	0.0089
OCSVM				
	Byte			
		S_1	00:05:52	0
		S_3	00:04:32	0.7869
		B_r	19:51:57	0.5220
	Flow			
		S_1	00:01:02	0
		S_3	00:01:07	0
		B_r	10:13:46	0.5220
LOF				
	Byte			
		S_1	00:01:19	0
		S_3	00:05:50	0
		B_r	40:13:40	0
	Flow			
		S_1	00:01:12	0
		S_3	01:24:37	0
		B_r	06:00:33	0
B_n				
	Byte			
		S_1	05:31:37	0.0035
		S_3	11:07:54	0.0234
	Flow			
		S_1	02:19:16	0.0702
		S_3	04:41:41	0.0014

Table B.3: SWaT train validation results

AD	Fragments	Model	T2F↓	FPR↓
IF				
	Byte			
		S_1	00:07:31	0
		S_3	00:05:58	0
		B_r	00:45:14	0.0001
	Flow			
		S_1	00:05:15	0
		S_3	00:05:11	0
		B_r	00:36:12	0.0089
OCSVM				
	Byte			
		S_1	00:05:52	0
		S_3	00:04:32	0.7869
		B_r	19:51:57	0.5220
	Flow			
		S_1	00:01:02	0
		S_3	00:01:07	0
		B_r	10:13:46	0.5220
LOF				
	Byte			
		S_1	00:01:19	0
		S_3	00:05:50	0
		B_r	40:13:40	0
	Flow			
		S_1	00:01:12	0
		S_3	01:24:37	0
		B_r	06:00:33	0
B_n				
	Byte			
		S_1	05:31:37	0.0035
		S_3	11:07:54	0.0234
	Flow			
		S_1	02:19:16	0.0702
		S_3	04:41:41	0.0014

Table B.3: SWaT train validation results

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0	1	0	0	00:13:29
		S_3	0	0	0	0	00:11:38
		B_r	0	0.9783	0.0120	0.0238	01:15:11
	Flow						
		S_1	0	0	0	0	00:09:32
		S_3	0	1	0	0	00:03:33
		B_r	0	0	0	0	00:31:27
OCSVM							
	Byte						
		S_1	0	1	0	0	00:03:36
		S_3	0.7915	0.0132	0.7656	0.0260	00:03:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
	Flow						
		S_1	0	1	0	0	00:02:45
		S_3	0	1	0	0	00:01:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
LOF							
	Byte						
		S_1	0	1	0	0	01:32:54
		S_3	0	0	0	0	06:03:28
		B_r	0	0.5294	0.0008	0.0016	69:04:37
	Flow						
		S_1	0	0	0	0	01:28:06
		S_3	0	0	0	0	00:21:47
		B_r	0	0.0349	0.0001	0.0001	10:14:01
B_n							
	Byte						
		S_1	0.01	0.4959	0.7079	0.5832	05:31:39
		S_3	0.0302	0.3059	0.9620	0.4642	11:07:56
	Flow						
		S_1	0.0732	0.0185	0.4239	0.0354	02:19:17
		S_3	0.0040	0.2463	0.3306	0.2823	04:41:42

Table B.4: SWaT anomaly detection results of $Eval_H$

AD	Fragments	Model	T2F↓	FPR↓
IF				
	Byte			
		S_1	00:07:31	0
		S_3	00:05:58	0
		B_r	00:45:14	0.0001
	Flow			
		S_1	00:05:15	0
		S_3	00:05:11	0
		B_r	00:36:12	0.0089
OCSVM				
	Byte			
		S_1	00:05:52	0
		S_3	00:04:32	0.7869
		B_r	19:51:57	0.5220
	Flow			
		S_1	00:01:02	0
		S_3	00:01:07	0
		B_r	10:13:46	0.5220
LOF				
	Byte			
		S_1	00:01:19	0
		S_3	00:05:50	0
		B_r	40:13:40	0
	Flow			
		S_1	00:01:12	0
		S_3	01:24:37	0
		B_r	06:00:33	0
B_n				
	Byte			
		S_1	05:31:37	0.0035
		S_3	11:07:54	0.0234
	Flow			
		S_1	02:19:16	0.0702
		S_3	04:41:41	0.0014

Table B.3: SWaT train validation results

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0	1	0	0	00:13:29
		S_3	0	0	0	0	00:11:38
		B_r	0	0.9783	0.0120	0.0238	01:15:11
	Flow						
		S_1	0	0	0	0	00:09:32
		S_3	0	1	0	0	00:03:33
		B_r	0	0	0	0	00:31:27
OCSVM							
	Byte						
		S_1	0	1	0	0	00:03:36
		S_3	0.7915	0.0132	0.7656	0.0260	00:03:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
	Flow						
		S_1	0	1	0	0	00:02:45
		S_3	0	1	0	0	00:01:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
LOF							
	Byte						
		S_1	0	1	0	0	01:32:54
		S_3	0	0	0	0	06:03:28
		B_r	0	0.5294	0.0008	0.0016	69:04:37
	Flow						
		S_1	0	0	0	0	01:28:06
		S_3	0	0	0	0	00:21:47
		B_r	0	0.0349	0.0001	0.0001	10:14:01
B_n							
	Byte						
		S_1	0.01	0.4959	0.7079	0.5832	05:31:39
		S_3	0.0302	0.3059	0.9620	0.4642	11:07:56
	Flow						
		S_1	0.0732	0.0185	0.4239	0.0354	02:19:17
		S_3	0.0040	0.2463	0.3306	0.2823	04:41:42

Table B.4: SWaT anomaly detection results of $Eval_H$

AD	Fragments	Model	T2F↓	FPR↓
IF				
	Byte			
		S_1	00:07:31	0
		S_3	00:05:58	0
		B_r	00:45:14	0.0001
	Flow			
		S_1	00:05:15	0
		S_3	00:05:11	0
		B_r	00:36:12	0.0089
OCSVM				
	Byte			
		S_1	00:05:52	0
		S_3	00:04:32	0.7869
		B_r	19:51:57	0.5220
	Flow			
		S_1	00:01:02	0
		S_3	00:01:07	0
		B_r	10:13:46	0.5220
LOF				
	Byte			
		S_1	00:01:19	0
		S_3	00:05:50	0
		B_r	40:13:40	0
	Flow			
		S_1	00:01:12	0
		S_3	01:24:37	0
		B_r	06:00:33	0
B_n				
	Byte			
		S_1	05:31:37	0.0035
		S_3	11:07:54	0.0234
	Flow			
		S_1	02:19:16	0.0702
		S_3	04:41:41	0.0014

Table B.3: SWaT train validation results

Over-fitted to training data

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0	1	0	0	00:13:29
		S_3	0	0	0	0	00:11:38
		B_r	0	0.9783	0.0120	0.0238	01:15:11
	Flow						
		S_1	0	0	0	0	00:09:32
		S_3	0	1	0	0	00:03:33
		B_r	0	0	0	0	00:31:27
OCSVM							
	Byte						
		S_1	0	1	0	0	00:03:36
		S_3	0.7915	0.0132	0.7656	0.0260	00:03:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
	Flow						
		S_1	0	1	0	0	00:02:45
		S_3	0	1	0	0	00:01:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
LOF							
	Byte						
		S_1	0	1	0	0	01:32:54
		S_3	0	0	0	0	06:03:28
		B_r	0	0.5294	0.0008	0.0016	69:04:37
	Flow						
		S_1	0	0	0	0	01:28:06
		S_3	0	0	0	0	00:21:47
		B_r	0	0.0349	0.0001	0.0001	10:14:01
B_n							
	Byte						
		S_1	0.01	0.4959	0.7079	0.5832	05:31:39
		S_3	0.0302	0.3059	0.9620	0.4642	11:07:56
	Flow						
		S_1	0.0732	0.0185	0.4239	0.0354	02:19:17
		S_3	0.0040	0.2463	0.3306	0.2823	04:41:42

Table B.4: SWaT anomaly detection results of $Eval_H$

AD	Fragments	Model	T2F↓	FPR↓
IF				
	Byte			
		S_1	00:07:31	0
		S_3	00:05:58	0
		B_r	00:45:14	0.0001
	Flow			
		S_1	00:05:15	0
		S_3	00:05:11	0
		B_r	00:36:12	0.0089
OCSVM				
	Byte			
		S_1	00:05:52	0
		S_3	00:04:32	0.7869
		B_r	19:51:57	0.5220
	Flow			
		S_1	00:01:02	0
		S_3	00:01:07	0
		B_r	10:13:46	0.5220
LOF				
	Byte			
		S_1	00:01:19	0
		S_3	00:05:50	0
		B_r	40:13:40	0
	Flow			
		S_1	00:01:12	0
		S_3	01:24:37	0
		B_r	06:00:35	0
B_n				
	Byte			
		S_1	05:31:37	0.0035
		S_3	11:07:54	0.0234
	Flow			
		S_1	02:19:16	0.0702
		S_3	04:41:41	0.0014

Table B.3: SWaT train validation results

Over-fitted to training data

Long training time

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0	1	0	0	00:13:29
		S_3	0	0	0	0	00:11:38
		B_r	0	0.9783	0.0120	0.0238	01:15:11
	Flow						
		S_1	0	0	0	0	00:09:32
		S_3	0	1	0	0	00:03:33
		B_r	0	0	0	0	00:31:27
OCSVM							
	Byte						
		S_1	0	1	0	0	00:03:36
		S_3	0.7915	0.0132	0.7656	0.0260	00:03:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
	Flow						
		S_1	0	1	0	0	00:02:45
		S_3	0	1	0	0	00:01:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
LOF							
	Byte						
		S_1	0	1	0	0	01:32:54
		S_3	0	0	0	0	06:03:28
		B_r	0	0.5294	0.0008	0.0016	69:04:37
	Flow						
		S_1	0	0	0	0	01:28:06
		S_3	0	0	0	0	00:21:47
		B_r	0	0.0349	0.0001	0.0001	10:14:01
B_n							
	Byte						
		S_1	0.01	0.4959	0.7079	0.5832	05:31:39
		S_3	0.0302	0.3059	0.9620	0.4642	11:07:56
	Flow						
		S_1	0.0732	0.0185	0.4239	0.0354	02:19:17
		S_3	0.0040	0.2463	0.3306	0.2823	04:41:42

Table B.4: SWaT anomaly detection results of $Eval_H$

Brandenburg University of Technology Cottbus - Senftenberg

AD	Fragments	Model	T2F↓	FPR↓
IF				
	Byte			
		S_1	00:07:31	0
		S_3	00:05:58	0
		B_r	00:45:14	0.0001
	Flow			
		S_1	00:05:15	0
		S_3	00:05:11	0
		B_r	00:36:12	0.0089
OCSVM				
	Byte			
		S_1	00:05:52	0
		S_3	00:04:32	0.7869
		B_r	19:51:57	0.5220
	Flow			
		S_1	00:01:02	0
		S_3	00:01:07	0
		B_r	10:13:46	0.5220
LOF				
	Byte			
		S_1	00:01:19	0
		S_3	00:05:50	0
		B_r	40:13:40	0
	Flow			
		S_1	00:01:12	0
		S_3	01:24:37	0
		B_r	06:00:35	0
B_n				
	Byte			
		S_1	05:31:37	0.0035
		S_3	11:07:54	0.0234
	Flow			
		S_1	02:19:16	0.0702
		S_3	04:41:41	0.0014

Table B.3: SWaT train validation results

Over-fitted to training data

Long training time

Better then guessing

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0	1	0	0	00:13:29
		S_3	0	0	0	0	00:11:38
		B_r	0	0.9783	0.0120	0.0238	01:15:11
	Flow						
		S_1	0	0	0	0	00:09:32
		S_3	0	1	0	0	00:03:33
		B_r	0	0	0	0	00:31:27
OCSVM							
	Byte						
		S_1	0	1	0	0	00:03:36
		S_3	0.7915	0.0132	0.7656	0.0260	00:03:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
	Flow						
		S_1	0	1	0	0	00:02:45
		S_3	0	1	0	0	00:01:14
		B_r	0.4316	0.0022	0.2876	0.0043	62:53:53
LOF							
	Byte						
		S_1	0	1	0	0	01:32:54
		S_3	0	0	0	0	06:03:28
		B_r	0	0.5294	0.0008	0.0016	69:04:37
	Flow						
		S_1	0	0	0	0	01:28:06
		S_3	0	0	0	0	00:21:47
		B_r	0	0.0349	0.0001	0.0001	10:14:01
B_n							
	Byte						
		S_1	0.01	0.4959	0.7079	0.5832	05:31:39
		S_3	0.0302	0.3059	0.9620	0.4642	11:07:56
	Flow						
		S_1	0.0732	0.0185	0.4239	0.0354	02:19:17
		S_3	0.0040	0.2463	0.3306	0.2823	04:41:42

Table B.4: SWaT anomaly detection results of $Eval_H$

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0.9999	0.0371	0.9999	0.0716	00:01:08
		S_3	0.9987	0.0511	0.9992	0.0973	00:01:09
		B_r	1	0.0371	1	0.0716	00:18:27
	Flow						
		S_1	0.0705	0.0261	0.1457	0.0442	00:00:28
		S_3	0	1	0	0	00:00:17
		B_r	0.9346	0.0137	1	0.0270	00:08:07
OCSVM							
	Byte						
		S_1	1	0.0371	1	0.0716	00:00:36
		S_3	0.0005	0.0384	0.0004	0.0008	00:00:38
		B_r	1	0.0369	0.9950	0.0712	07:14:04
	Flow						
		S_1	1	0.0128	1	0.0253	00:00:16
		S_3	1	0.0129	1	0.0255	00:00:18
		B_r	1	0.0127	0.9902	0.0250	03:52:22
LOF							
	Byte						
		S_1	0.9999	0.0371	1	0.0716	06:33:01
		S_3	1	0.0511	1	0.0973	06:14:50
		B_r	1	0.0371	1	0.0716	11:13:28
	Flow						
		S_1	0.9387	0.0136	1	0.0269	01:05:51
		S_3	1	0.0129	1	0.0255	00:48:12
		B_r	1	0.0127	0.9952	0.0251	05:32:56
B_n							
	Byte						
		S_1	0.0091	0.0152	0.0036	0.0059	05:32:02
		S_3	0.0281	0.0543	0.03	0.0386	11:17:33
	Flow						
		S_1	0.0085	0.2227	0.1870	0.2033	02:25:40
		S_3	0.0030	0.0618	0.0150	0.0241	04:52:41

Table B.5: SWaT anomaly detection results of $Eval_D$

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0.9999	0.0004	1	0.0008	00:00:45
		S_3	0.9987	0.0008	1	0.0017	00:00:46
		B_r	1	0.0004	1	0.0008	00:18:57
	Flow						
		S_1	0.0706	0.0001	0.1034	0.0003	00:00:18
		S_3	0	1	0	0	00:00:19
		B_r	0.9346	0.0001	1	0.0002	00:07:56
OCSVM							
	Byte						
		S_1	1	0	1	0.0001	00:00:12
		S_3	1	0.0001	1	0.0003	00:00:12
		B_r	0.0169	0.0004	0.0169	0.0008	06:54:35
	Flow						
		S_1	1	0.0001	1	0.0002	00:00:18
		S_3	1	0.0002	1	0.0004	00:00:12
		B_r	0.0010	0.0001	0.0010	0.0002	02:48:17
LOF							
	Byte						
		S_1	0.9999	0.0004	1	0.0008	06:31:27
		S_3	1	0.0008	1	0.0017	06:08:44
		B_r	0.1050	0.0003	0.1050	0.0007	09:46:53
	Flow						
		S_1	0.9387	0.0001	1	0.0002	01:05:39
		S_3	1	0.0002	1	0.0004	00:42:48
		B_r	0.0968	0.0001	0.0968	0.0002	04:17:47
B_n							
	Byte						
		S_1	0.0090	0	0	0	05:28:49
		S_3	0.0278	0.0015	0.0484	0.0029	11:05:08
	Flow						
		S_1	0.0085	0.0020	0.1724	0.0040	02:19:47
		S_3	0.0030	0.0012	0.0182	0.0022	04:50:35

Table B.6: SWaT anomaly detection results of $Eval_E$

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0.9999	0.0371	0.9999	0.0716	00:01:08
		S_3	0.9987	0.0511	0.9992	0.0973	00:01:09
		B_r	1	0.0371	1	0.0716	00:18:27
	Flow						
		S_1	0.0705	0.0261	0.1457	0.0442	00:00:28
		S_3	0	1	0	0	00:00:17
		B_r	0.9346	0.0137	1	0.0270	00:08:07
OCSVM							
	Byte						
		S_1	1	0.0371	1	0.0716	00:00:36
		S_3	0.0005	0.0384	0.0004	0.0008	00:00:38
		B_r	1	0.0369	0.9950	0.0712	07:14:04
	Flow						
		S_1	1	0.0128	1	0.0253	00:00:16
		S_3	1	0.0129	1	0.0255	00:00:18
		B_r	1	0.0127	0.9902	0.0250	03:52:22
LOF							
	Byte						
		S_1	0.9999	0.0371	1	0.0716	06:33:01
		S_3	1	0.0511	1	0.0973	06:14:50
		B_r	1	0.0371	1	0.0716	11:13:28
	Flow						
		S_1	0.9387	0.0136	1	0.0269	01:05:51
		S_3	1	0.0129	1	0.0255	00:48:12
		B_r	1	0.0127	0.9952	0.0251	05:32:56
B_n							
	Byte						
		S_1	0.0091	0.0152	0.0036	0.0059	05:32:02
		S_3	0.0281	0.0543	0.03	0.0386	11:17:33
	Flow						
		S_1	0.0085	0.2227	0.1870	0.2033	02:25:40
		S_3	0.0030	0.0618	0.0150	0.0241	04:52:41

Table B.5: SWaT anomaly detection results of $Eval_D$

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0.9999	0.0004	1	0.0008	00:00:45
		S_3	0.9987	0.0008	1	0.0017	00:00:46
		B_r	1	0.0004	1	0.0008	00:18:57
	Flow						
		S_1	0.0706	0.0001	0.1034	0.0003	00:00:18
		S_3	0	1	0	0	00:00:19
		B_r	0.9346	0.0001	1	0.0002	00:07:56
OCSVM							
	Byte						
		S_1	1	0	1	0.0001	00:00:12
		S_3	1	0.0001	1	0.0003	00:00:12
		B_r	0.0169	0.0004	0.0169	0.0008	06:54:35
	Flow						
		S_1	1	0.0001	1	0.0002	00:00:18
		S_3	1	0.0002	1	0.0004	00:00:12
		B_r	0.0010	0.0001	0.0010	0.0002	02:48:17
LOF							
	Byte						
		S_1	0.9999	0.0004	1	0.0008	06:31:27
		S_3	1	0.0008	1	0.0017	06:08:44
		B_r	0.1050	0.0003	0.1050	0.0007	09:46:53
	Flow						
		S_1	0.9387	0.0001	1	0.0002	01:05:39
		S_3	1	0.0002	1	0.0004	00:42:48
		B_r	0.0968	0.0001	0.0968	0.0002	04:17:47
B_n							
	Byte						
		S_1	0.0090	0	0	0	05:28:49
		S_3	0.0278	0.0015	0.0484	0.0029	11:05:08
	Flow						
		S_1	0.0085	0.0020	0.1724	0.0040	02:19:47
		S_3	0.0030	0.0012	0.0182	0.0022	04:50:35

Table B.6: SWaT anomaly detection results of $Eval_E$

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0.9999	0.0371	0.9999	0.0716	00:01:08
		S_3	0.9987	0.0511	0.9992	0.0973	00:01:09
		B_r	1	0.0371	1	0.0716	00:18:27
	Flow						
		S_1	0.0705	0.0261	0.1457	0.0442	00:00:28
		S_3	0	1	0	0	00:00:17
		B_r	0.9346	0.0137	1	0.0270	00:08:07
OCSVM							
	Byte						
		S_1	1	0.0371	1	0.0716	00:00:36
		S_3	0.0005	0.0384	0.0004	0.0008	00:00:38
		B_r	1	0.0369	0.9950	0.0712	07:14:04
	Flow						
		S_1	1	0.0128	1	0.0[illegible]	
		S_3	1	0.0129	1	0.0[illegible]	
		B_r	1	0.0127	0.9902	0.0250	03:52:22
LOF							
	Byte						
		S_1	0.9999	0.0371	1	0.0716	06:33:01
		S_3	1	0.0511	1	0.0973	06:14:50
		B_r	1	0.0371	1	0.0716	11:13:28
	Flow						
		S_1	0.9387	0.0136	1	0.0269	01:05:51
		S_3	1	0.0129	1	0.0255	00:48:12
		B_r	1	0.0127	0.9952	0.0251	05:32:56
B_n							
	Byte						
		S_1	0.0091	0.0152	0.0036	0.0059	05:32:02
		S_3	0.0281	0.0543	0.03	0.0386	11:17:33
	Flow						
		S_1	0.0085	0.2227	0.1870	0.2033	02:25:40
		S_3	0.0030	0.0618	0.0150	0.0241	04:52:41

Table B.5: SWaT anomaly detection results of $Eval_D$

Synthetic / inter-packet anomalies could not be detected

AD	Fragments	Model	FPR↓	PR↑	RC↑	F_1↑	T2T↓
IF							
	Byte						
		S_1	0.9999	0.0004	1	0.0008	00:00:45
		S_3	0.9987	0.0008	1	0.0017	00:00:46
		B_r	1	0.0004	1	0.0008	00:18:57
	Flow						
		S_1	0.0706	0.0001	0.1034	0.0003	00:00:18
		S_3	0	1	0	0	00:00:19
		B_r	0.9346	0.0001	1	0.0002	00:07:56
OCSVM							
	Byte						
		S_1	1	0	1	0.0001	00:00:12
		S_3	1	0.0001	1	0.0003	00:00:12
		B_r	0.0169	0.0004	0.0169	0.0008	06:54:35
	Flow						
		S_1	[illegible]	[illegible]	[illegible]	0.0001	00:00:18
		S_3	[illegible]	[illegible]	[illegible]	0.0002	00:00:12
		B_r	0.0010	0.0001	0.0010	0.0002	02:48:17
LOF							
	Byte						
		S_1	0.9999	0.0004	1	0.0008	06:31:27
		S_3	1	0.0008	1	0.0017	06:08:44
		B_r	0.1050	0.0003	0.1050	0.0007	09:46:53
	Flow						
		S_1	0.9387	0.0001	1	0.0002	01:05:39
		S_3	1	0.0002	1	0.0004	00:42:48
		B_r	0.0968	0.0001	0.0968	0.0002	04:17:47
B_n							
	Byte						
		S_1	0.0090	0	0	0	05:28:49
		S_3	0.0278	0.0015	0.0484	0.0029	11:05:08
	Flow						
		S_1	0.0085	0.0020	0.1724	0.0040	02:19:47
		S_3	0.0030	0.0012	0.0182	0.0022	04:50:35

Table B.6: SWaT anomaly detection results of $Eval_E$

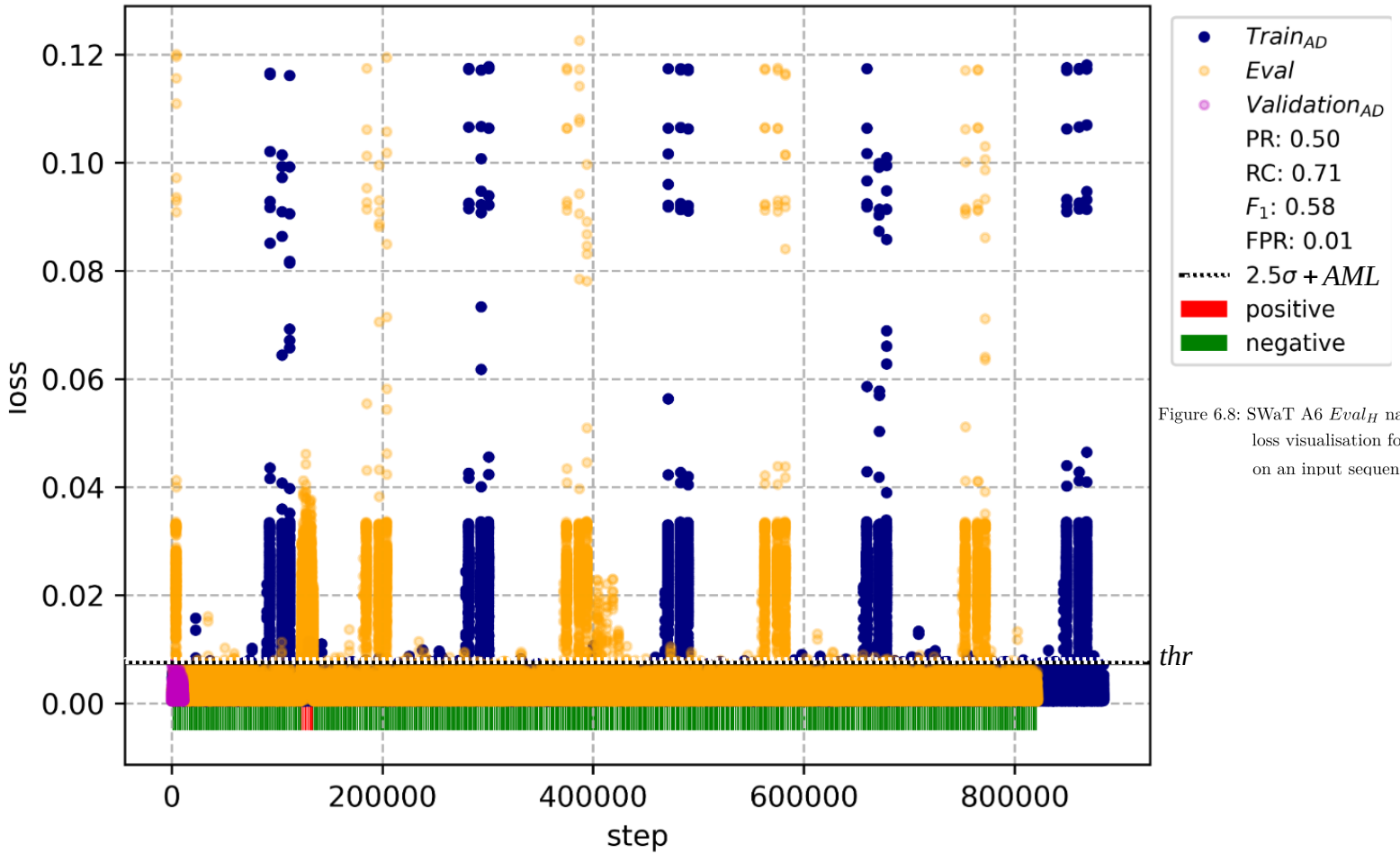


Figure 6.8: SWaT A6 $Eval_H$ naive baseline evaluation loss visualisation for byte-fragments trained on an input sequence length of n=1

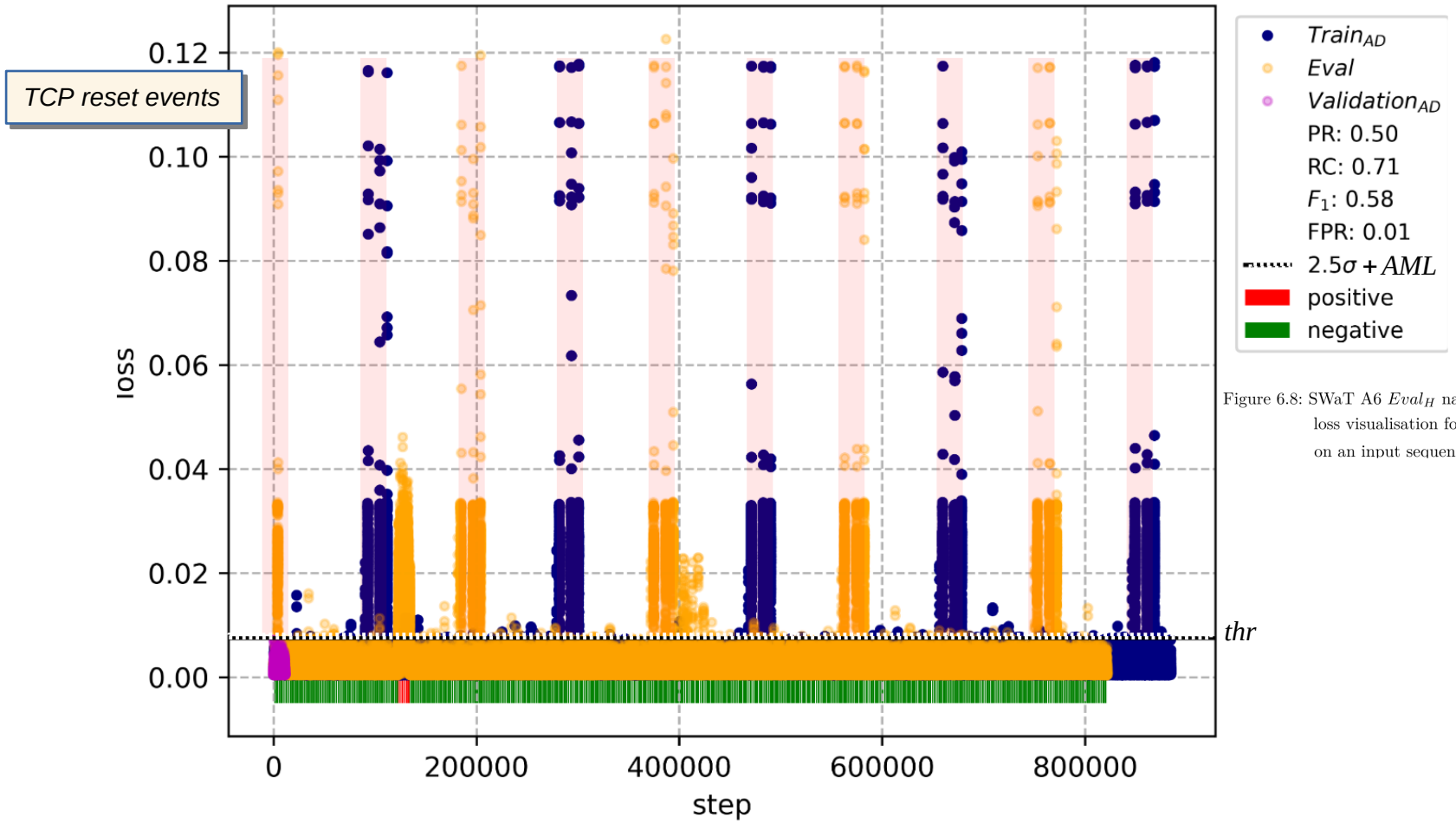


Figure 6.8: SWaT A6 $Eval_H$ naive baseline evaluation loss visualisation for byte-fragments trained on an input sequence length of n=1

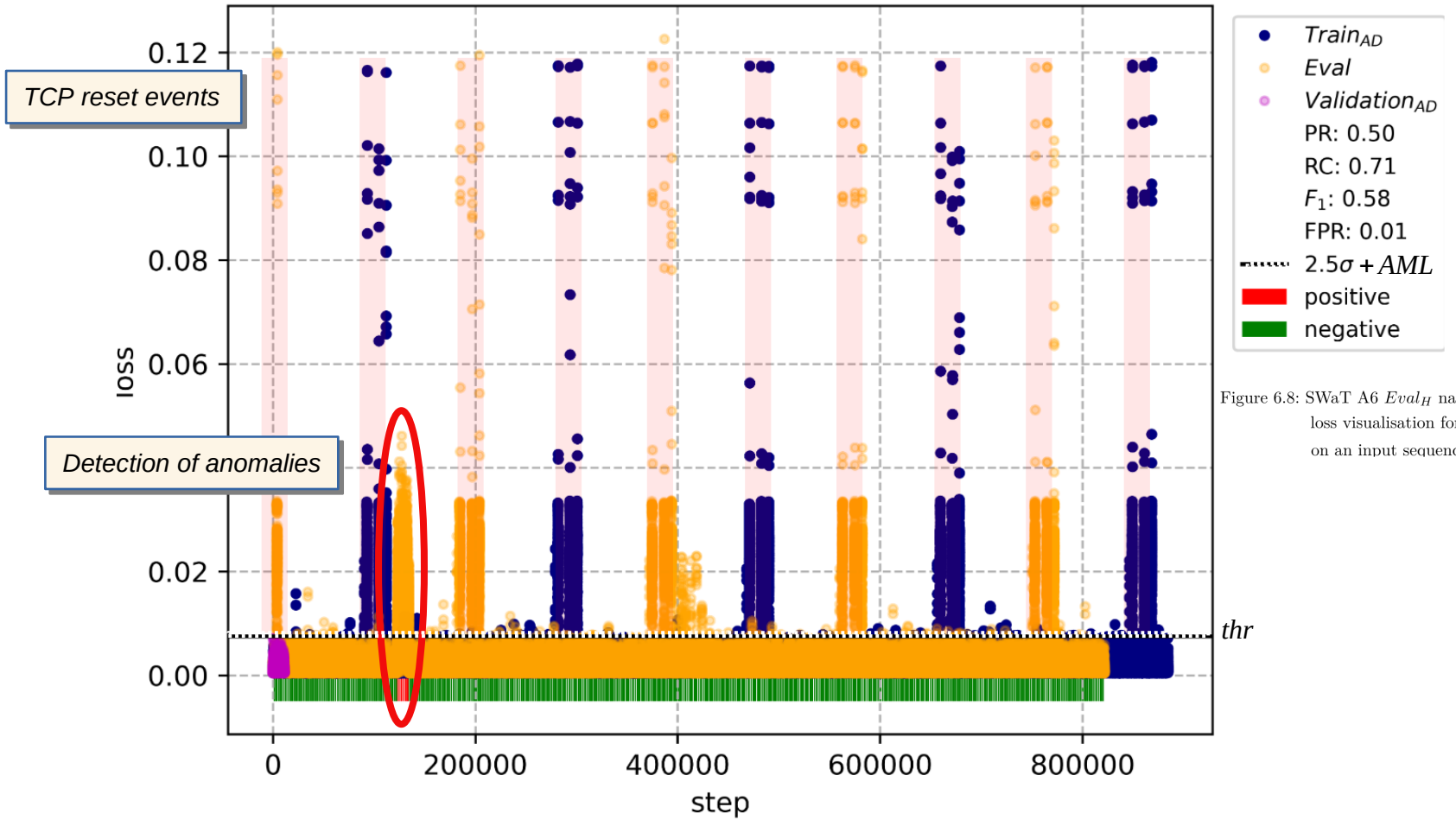


Figure 6.8: SWaT A6 $Eval_H$ naive baseline evaluation loss visualisation for byte-fragments trained on an input sequence length of n=1

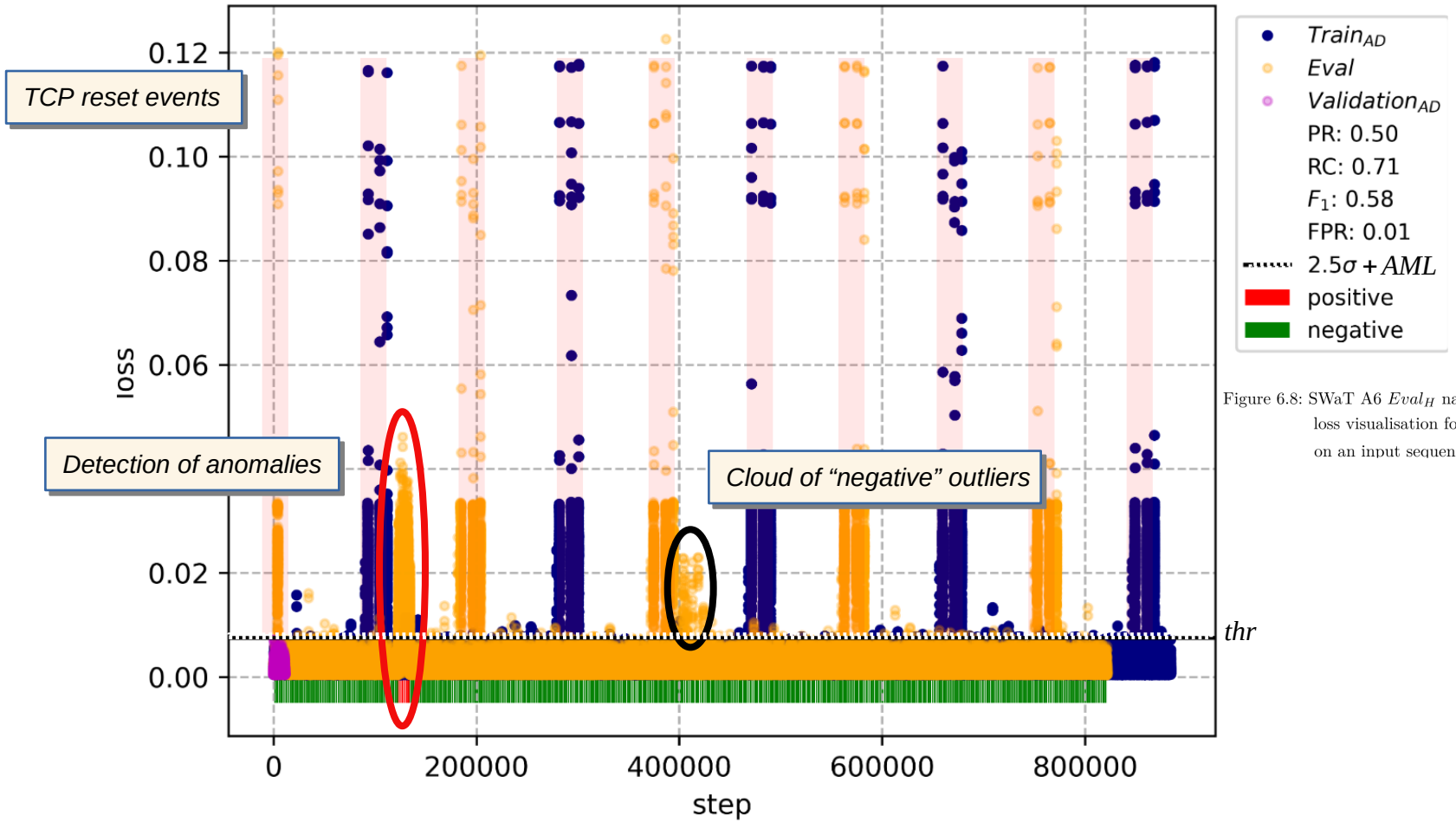


Figure 6.8: SWaT A6 $Eval_H$ naive baseline evaluation loss visualisation for byte-fragments trained on an input sequence length of n=1

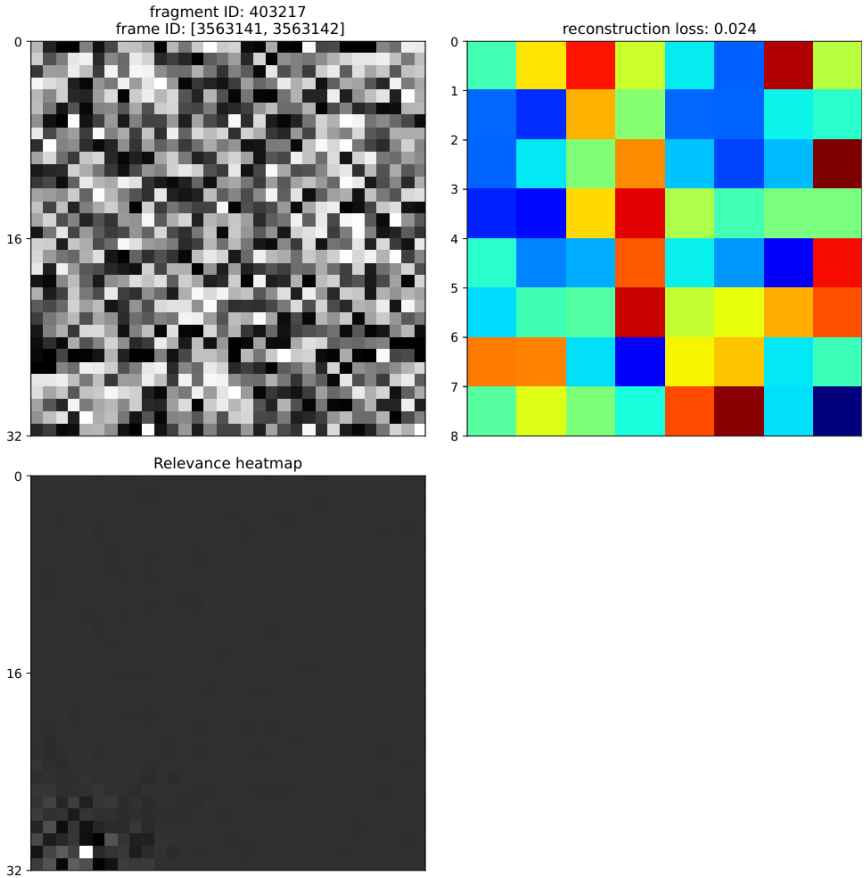


Figure B.9: Byte-fragments relevance heatmaps of the unusual UDP packet

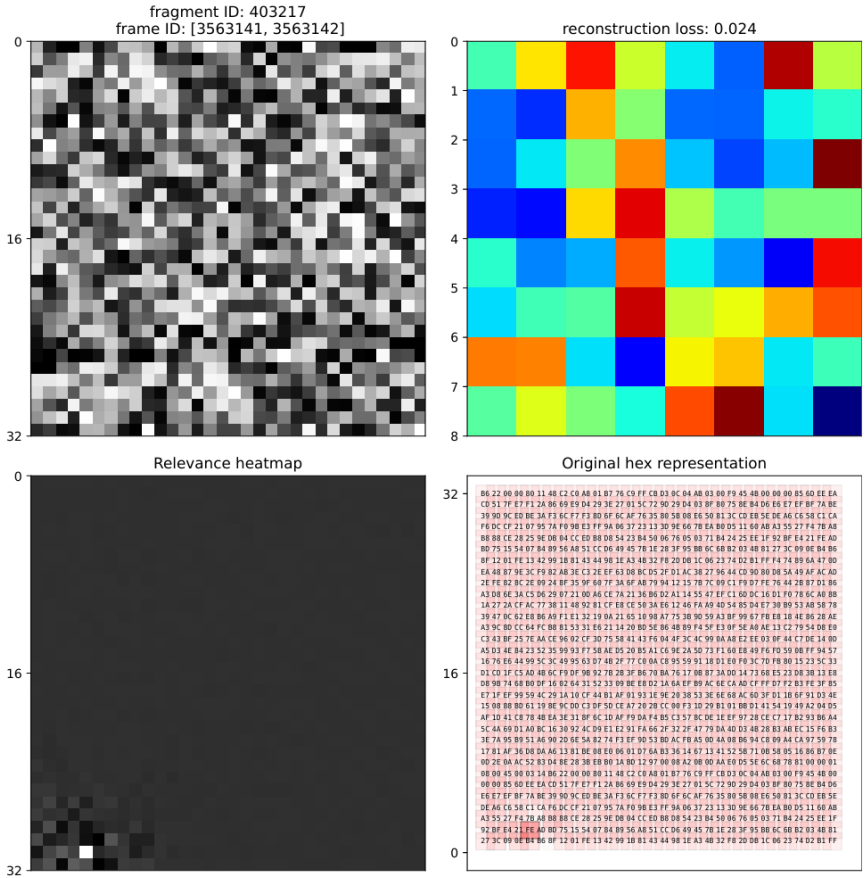
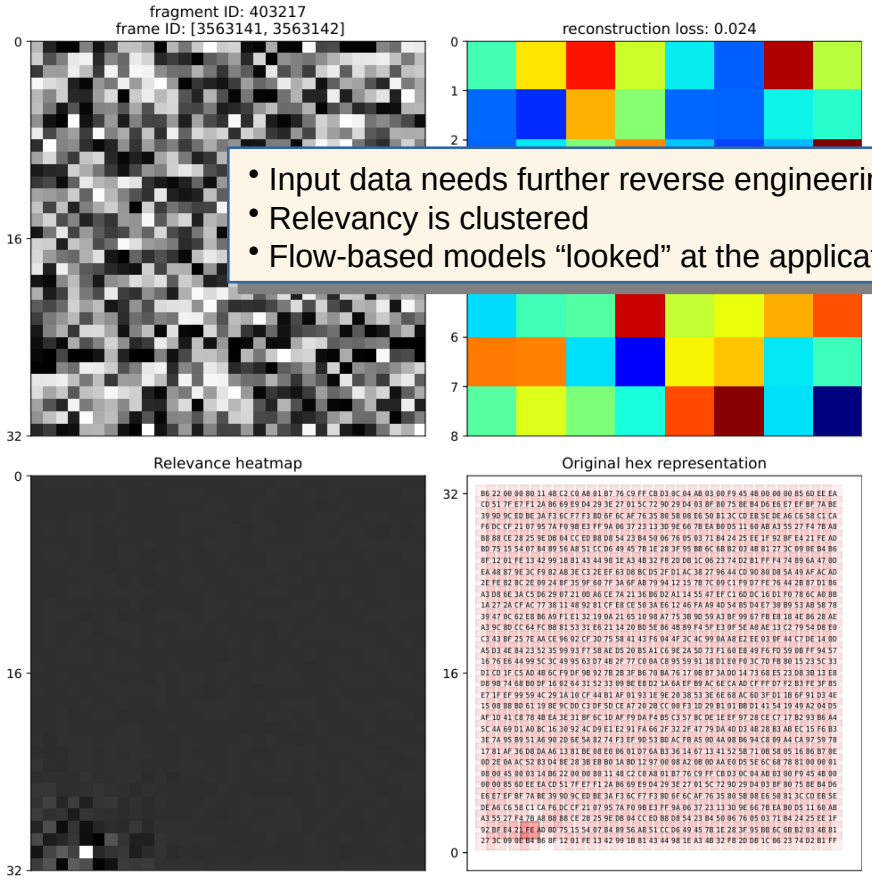


Figure B.9: Byte-fragments relevance heatmaps of the unusual UDP packet



- Input data needs further reverse engineering
- Relevancy is clustered
- Flow-based models “looked” at the application layer

Figure B.9: Byte-fragments relevance heatmaps of the unusual UDP packet

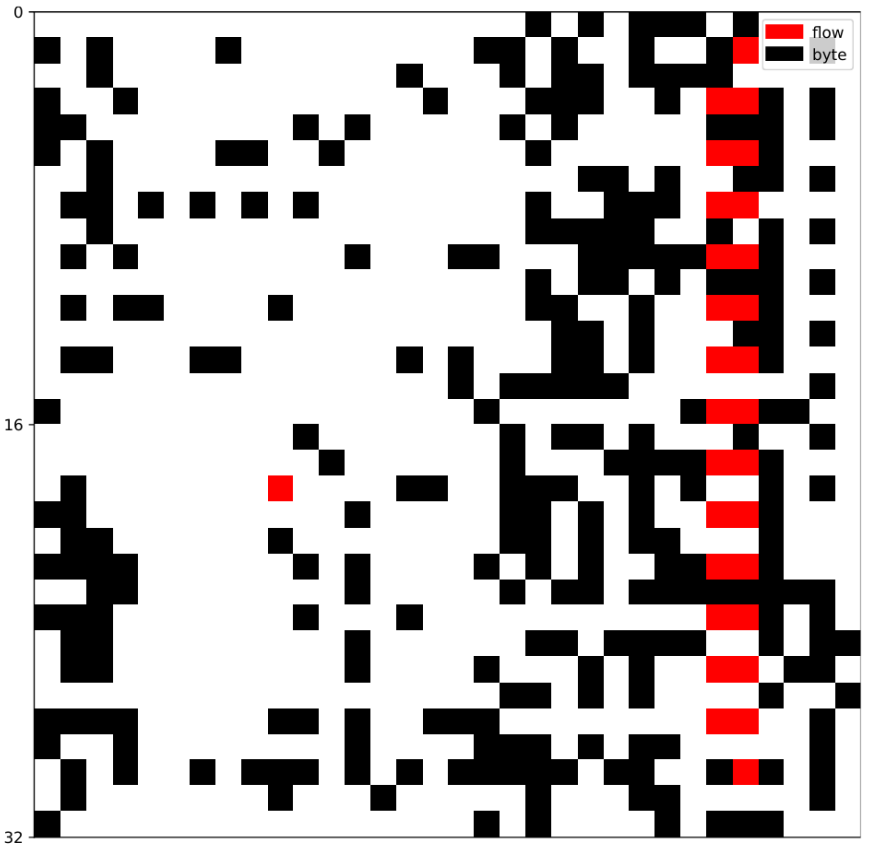


Figure 6.11: Aggregated relevance heatmaps for 1% of randomly selected evaluation fragments

- Hybrid model produces poor results
  - Under-fitted feature extraction model
    - Model loss fluctuates
    - Premature training stop (@6 epochs)
    - Small/fixed validation set
  - Over-fitted anomaly detection model
    - Biased hyper-parameter search

- Hybrid model produces poor results
  - Under-fitted feature extraction model
    - Model loss fluctuates
    - Premature training stop (@6 epochs)
    - Small/fixed validation set
  - Over-fitted anomaly detection model
    - Biased hyper-parameter search
- Detection models took longer to train on raw features
  - Raw baseline model not advantageous

- Hybrid model produces poor results
  - Under-fitted feature extraction model
    - Model loss fluctuates
    - Premature training stop (@6 epochs)
    - Small/fixed validation set
  - Over-fitted anomaly detection model
    - Biased hyper-parameter search
- Detection models took longer to train on raw features
  - Raw baseline model not advantageous
- Naive baseline model identified anomalies
  - Extracted features differed from normal samples

- Hybrid model produces poor results
  - Under-fitted feature extraction model
    - Model loss fluctuates
    - Premature training stop (@6 epochs)
    - Small/fixed validation set
  - Over-fitted anomaly detection model
    - Biased hyper-parameter search
- Detection models took longer to train on raw features
  - Raw baseline model not advantageous
- Naive baseline model identified anomalies
  - Extracted features differed from normal samples

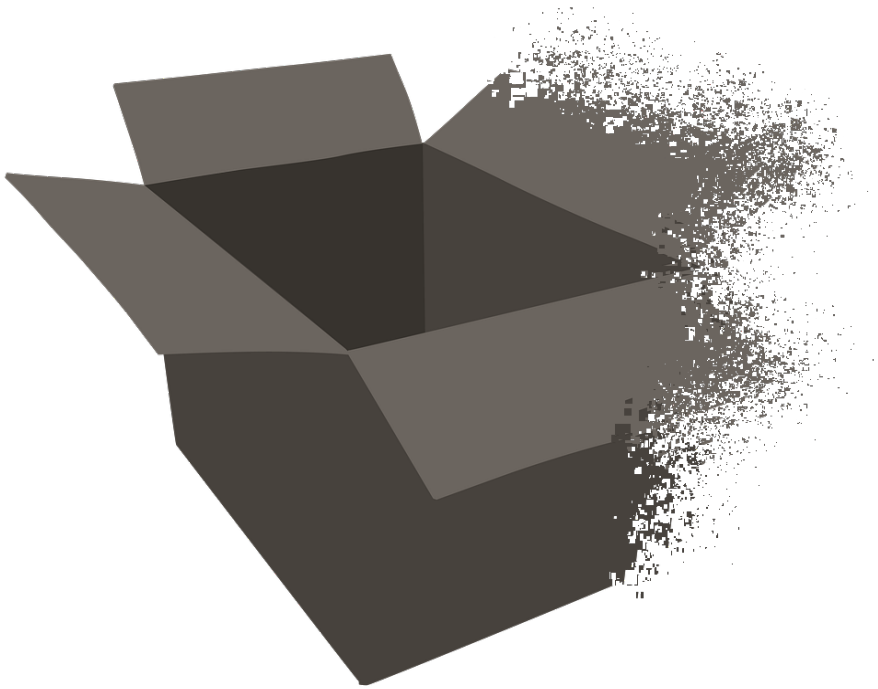
- Byte-fragments outperformed flow-fragments
  - Byte-Fragments contain more information
  - Fewer flow-fragments were generated

- Hybrid model produces poor results
  - Under-fitted feature extraction model
    - Model loss fluctuates
    - Premature training stop (@6 epochs)
    - Small/fixed validation set
  - Over-fitted anomaly detection model
    - Biased hyper-parameter search
- Detection models took longer to train on raw features
  - Raw baseline model not advantageous
- Naive baseline model identified anomalies
  - Extracted features differed from normal samples

- Byte-fragments outperformed flow-fragments
  - Byte-Fragments contain more information
  - Fewer flow-fragments were generated
- Inter-packet anomalies were not identified
  - Feature extraction learned temporal aspects
  - Detecting of temporal deviations were not demonstrated

- Hybrid model produces poor results
  - Under-fitted feature extraction model
    - Model loss fluctuates
    - Premature training stop (@6 epochs)
    - Small/fixed validation set
  - Over-fitted anomaly detection model
    - Biased hyper-parameter search
- Detection models took longer to train on raw features
  - Raw baseline model not advantageous
- Naive baseline model identified anomalies
  - Extracted features differed from normal samples

- Byte-fragments outperformed flow-fragments
  - Byte-Fragments contain more information
  - Fewer flow-fragments were generated
- Inter-packet anomalies were not identified
  - Feature extraction learned temporal aspects
  - Detecting of temporal deviations were not demonstrated
- Heatmaps reveal insights about extracted features
  - LRP does explain *why* certain areas were important
  - Manual reverse engineering is needed for analysis



- Introduction of an ANN-based feature extraction framework
  - *Unsupervised*, *protocol-agnostic* and *content sensitive* feature extraction
  - Extraction of spatial- & temporal-features from a stream of frames
  - Offer capabilities to detect anomalies with out the need for additional models (end-to-end detection)
  - Helps to bridges the *semantic gap* by offering LRP-generated heatmaps for alerts

- Introduction of an ANN-based feature extraction framework
  - *Unsupervised*, *protocol-agnostic* and *content sensitive* feature extraction
  - Extraction of spatial- & temporal-features from a stream of frames
  - Offer capabilities to detect anomalies with out the need for additional models (end-to-end detection)
  - Helps to bridges the *semantic gap* by offering LRP-generated heatmaps for alerts
- Comparison between different detection configurations over 2 data sets with 2 baselines
  - Input data representations (byte- & flow-fragments)
  - Small & longer input sequences (S1 & S3)
  - Isolation Forest & OC-SVM & Local outlier factor

- Introduction of an ANN-based feature extraction framework
  - *Unsupervised*, *protocol-agnostic* and *content sensitive* feature extraction
  - Extraction of spatial- & temporal-features from a stream of frames
  - Offer capabilities to detect anomalies with out the need for additional models (end-to-end detection)
  - Helps to bridges the *semantic gap* by offering LRP-generated heatmaps for alerts
- Comparison between different detection configurations over 2 data sets with 2 baselines
  - Input data representations (byte- & flow-fragments)
  - Small & longer input sequences (S1 & S3)
  - Isolation Forest & OC-SVM & Local outlier factor
- Bad evaluation results
  - Feature extraction model training was stopped prematurely
  - Validation data was not representative
  - Limited hyper-parameter grid-search (AE & AD)
  - Unbalanced evaluation (more byte-fragments then flow-fragments)
  - Estimated ground truth labels

I. Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?

– Detection of intra-packet anomalies demonstrated the extraction of spatial features

– Temporal extraction was demonstrated through constructed experiments

I. Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?

– Detection of intra-packet anomalies demonstrated the extraction of spatial features

– Temporal extraction was demonstrated through constructed experiments

II. Can anomaly detection be improved when trained on extracted features ?

– Conducted evaluation does not give evidence to support this claim

I. Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?

– Detection of intra-packet anomalies demonstrated the extraction of spatial features

– Temporal extraction was demonstrated through constructed experiments

II. Can anomaly detection be improved when trained on extracted features ?

– Conducted evaluation does not give evidence to support this claim

III. What type of input data representations yields better results for different anomalies ?

– Byte-fragments were advantageous (higher F1-Score) for intra-packet anomalies

– Inter-packet anomalies were not detected overall

I. Are computing systems able to perform for a spatial-temporal extraction of feature from raw data ?

– Detection of intra-packet anomalies demonstrated the extraction of spatial features

– Temporal extraction was demonstrated through constructed experiments

II. Can anomaly detection be improved when trained on extracted features ?

– Conducted evaluation does not give evidence to support this claim

III. What type of input data representations yields better results for different anomalies ?

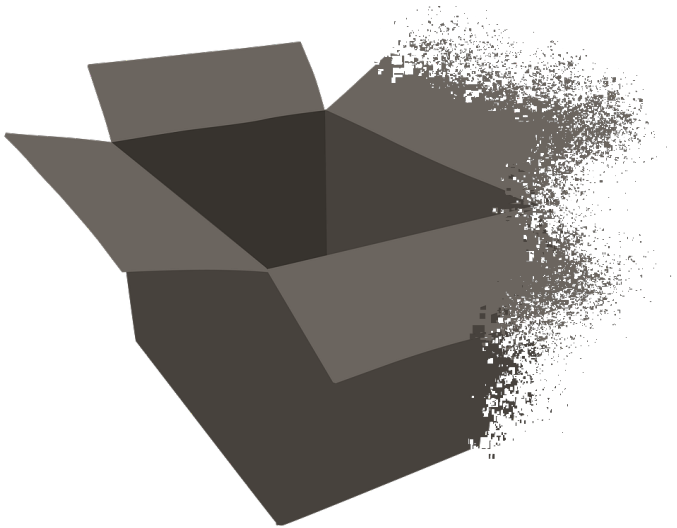
– Byte-fragments were advantageous (higher F1-Score) for intra-packet anomalies

– Inter-packet anomalies were not detected overall

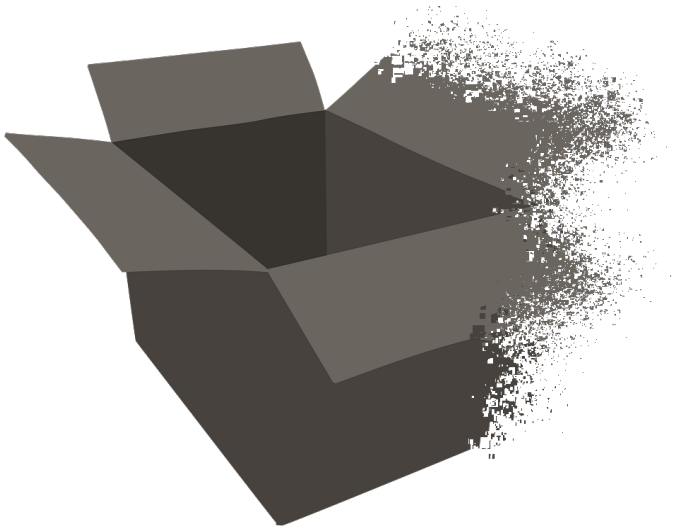
IV. How can the semantic gap be closed with the help of explainable models ?

– Auxiliary techniques (LRP) offer a natural investigation for ANN

- Simplify problem space
  - *Reassembly* of packets before features are extracted
  - *Restrict* model to train on only certain *network protocols / hosts addresses*



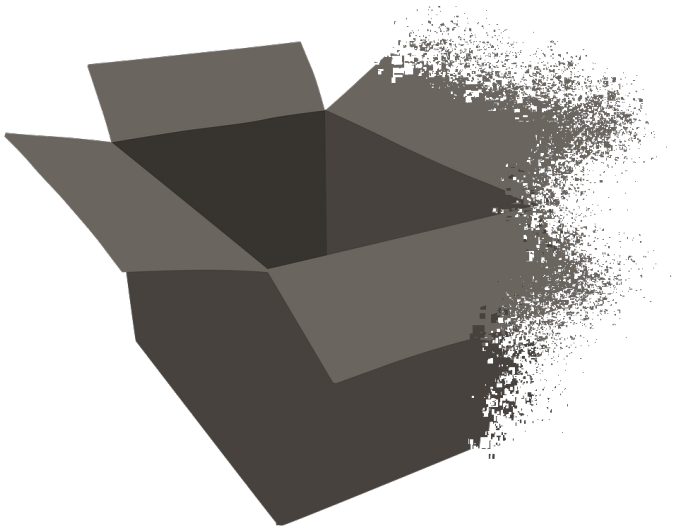
- Simplify problem space
  - *Reassembly* of packets before features are extracted
  - *Restrict* model to train on only certain *network protocols / hosts addresses*
- Extend feature extraction model
  - Introduce more layers per sub-network / longer training
  - Train to reconstruct future fragments
  - Process *1d representations* of network data



- Simplify problem space
  - *Reassembly* of packets before features are extracted
  - *Restrict* model to train on only certain *network protocols / hosts addresses*
- Extend feature extraction model
  - Introduce more layers per sub-network / longer training
  - Train to reconstruct future fragments
  - Process *1d representations* of network data
- Extend model interpretability
  - Use *negative & positive* relevance propagation
  - Introduce protocol decoders and UI's for semantic interpretation

- Simplify problem space
  - *Reassembly* of packets before features are extracted
  - *Restrict* model to train on only certain *network protocols / hosts addresses*
- Extend feature extraction model
  - Introduce more layers per sub-network / longer training
  - Train to reconstruct future fragments
  - Process *1d representations* of network data
- Extend model interpretability
  - Use *negative & positive* relevance propagation
  - Introduce protocol decoders and UI's for semantic interpretation
- Utilize synthetic anomalies
  - Aid the feature extraction training
  - Aid the anomaly detection training

Thanks for the attention!
Questions?



git.informatik.tu-cottbus.de/koppfabi/thesis

A | Appendix: **References Papers**

Y. Bengio et al. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction – ***ICML 2008***

J. Schmidhuber et al. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction – ***ICANN 2011***

B. Frey et al. Winner-Take-All Autoencoders – ***NIPS 2015***

K. Kim et al. RaPP: Novelty Detection with Reconstruction along Projection Pathway – ***ICLR 2020***

A. Binder et al. Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers – ***ICANN 2016***

P. Schneider et al. High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks – ***CPS-SPC 2018***

P. Casas et al. DeepMAL – Deep Learning Models for Malware Traffic Detection and Classification – ***iDSC 2020***

ZQ. Qin et al. Attentional Payload Anomaly Detector for Web Applications – ***ICONIP 2018***

X. Shi et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting – ***NIPS 2015***

J. Nocedal et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima – ***ICLR 2017***

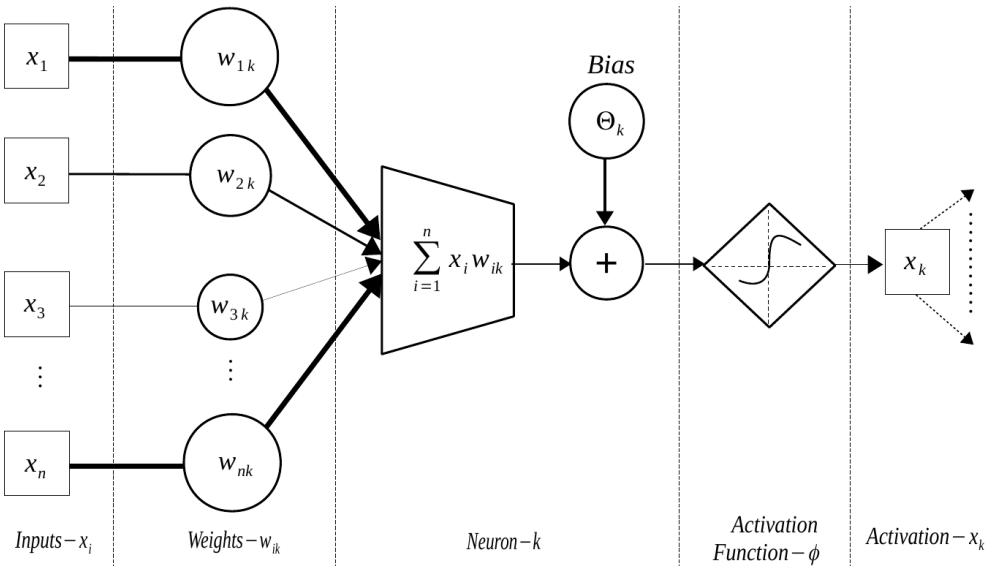
V. Nair et al. Rectified Linear Units Improve Restricted Boltzmann Machines – ***ICML 2010***

L. Smith et al. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates – ***SPIE 2019***

N. Tippenhauer et al. SWaT: A water treatment testbed for research and training on ICS security – ***CySWater 2016***

R. Hanusch Synthetische Generierung von netzbasierten Angriffsdaten – ***BTU 2018***

- *Hierarchical network of layers of neurons*
  - *Extracts information through the hierarchy*
- Loss function is used to judge how wrong the model's decisions are
- ANN trained by minimizing an optimization problem
- Advantages:
  (+) No feature extraction need
  (+) Scales with more data & more compute
- Disadvantage:
  (–) Many hyper-parameters & model architectures
  (–) Complex models are hard to train

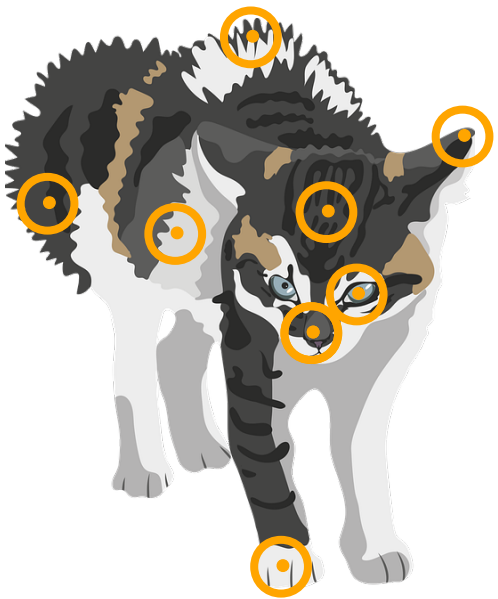


- Transformation of input data into a set of derived values (features)
- Features intended to be informative, non-redundant and describe the data with sufficient accuracy

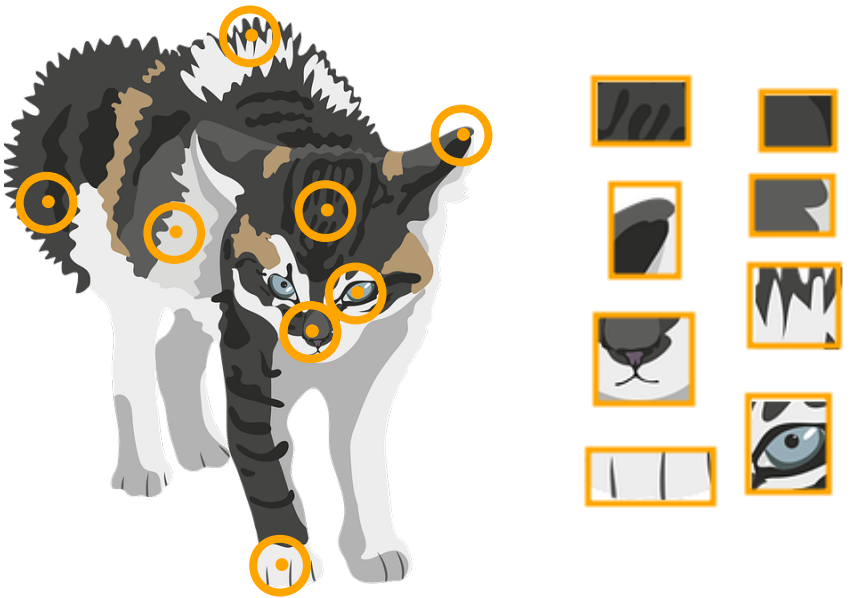
- Transformation of input data into a set of derived values (features)
- Features intended to be informative, non-redundant and describe the data with sufficient accuracy
- Special form of dimensional reduction

- Transformation of input data into a set of derived values (features)
- Features intended to be informative, non-redundant and describe the data with sufficient accuracy
- Special form of dimensional reduction

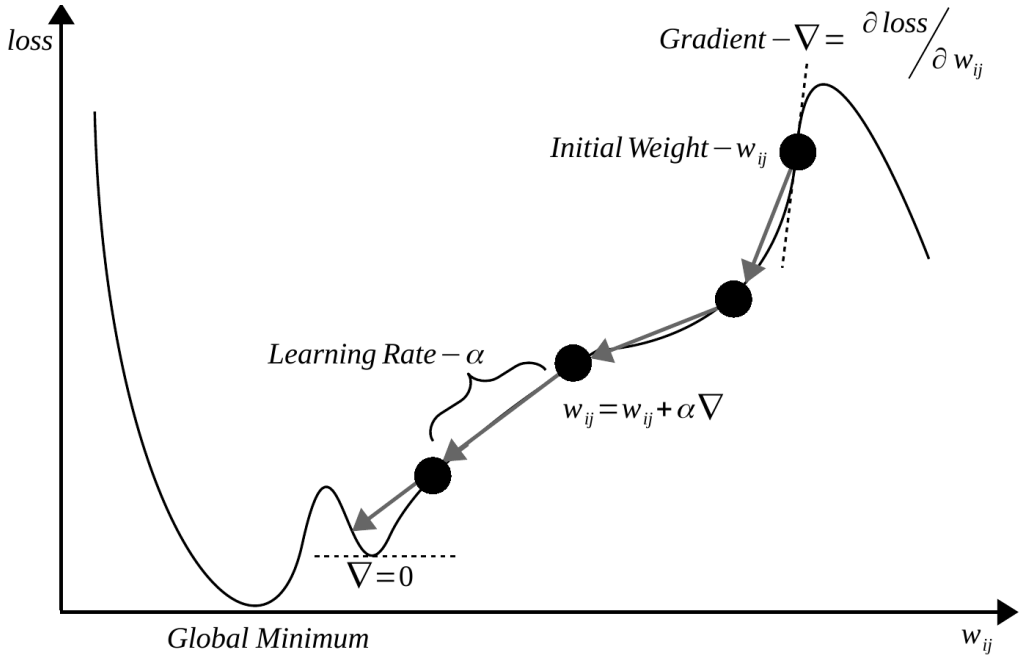
- Transformation of input data into a set of derived values (features)
- Features intended to be informative, non-redundant and describe the data with sufficient accuracy
- Special form of dimensional reduction



- Transformation of input data into a set of derived values (features)
- Features intended to be informative, non-redundant and describe the data with sufficient accuracy
- Special form of dimensional reduction
- Representation learning (feature learning) - automatically discovery of suitable representations



A | Appendix: **Gradient Desecent**

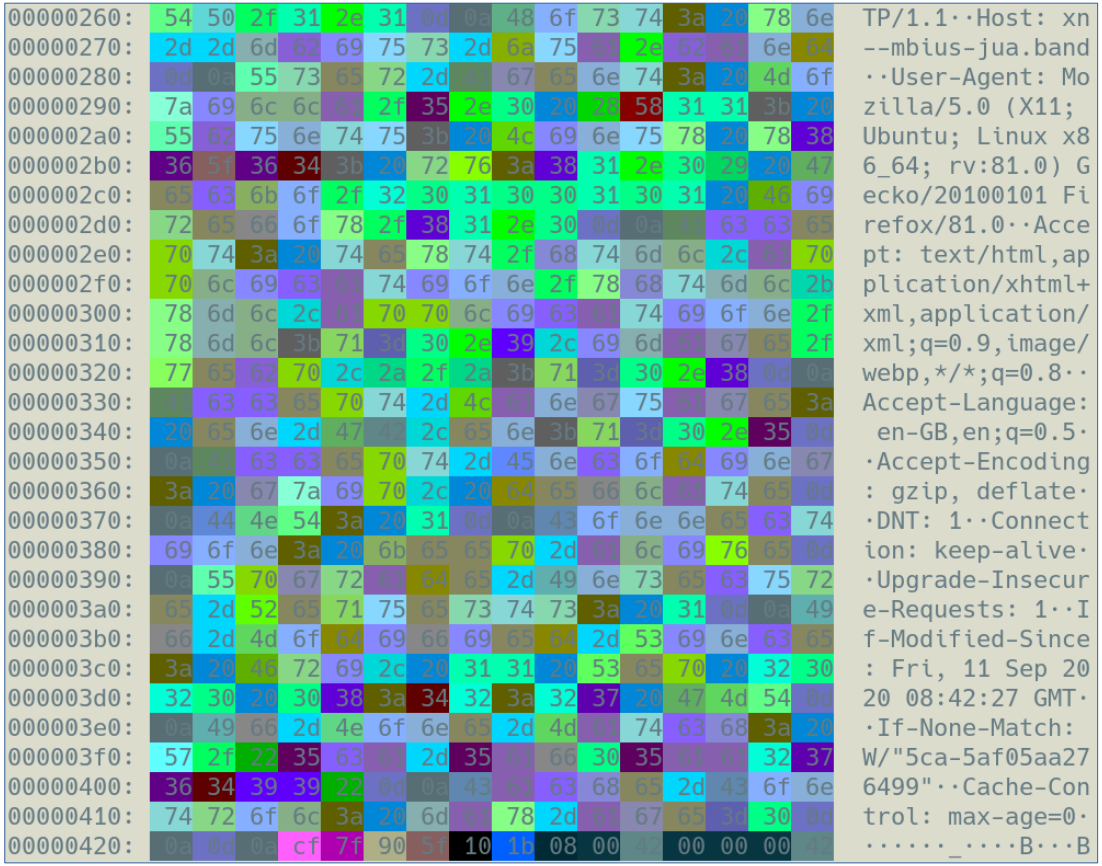


- *Loss function* is used to measure how *good* the network understood the problem
- Error is propagated back after each *batch* of samples → *backpropagation*
- *Partial derivatives for each weight (*gradient) *shows direction where weight will minimize*
- *Learn rate* dictates amount of change in each step

Brandenburg University of Technology Cottbus - Senftenberg

Network Data

- Series of *packets*
- Weak order of packets
- Packets consist of *layers*
- Different types of protocols
- Layers have *header* and *payload* fields



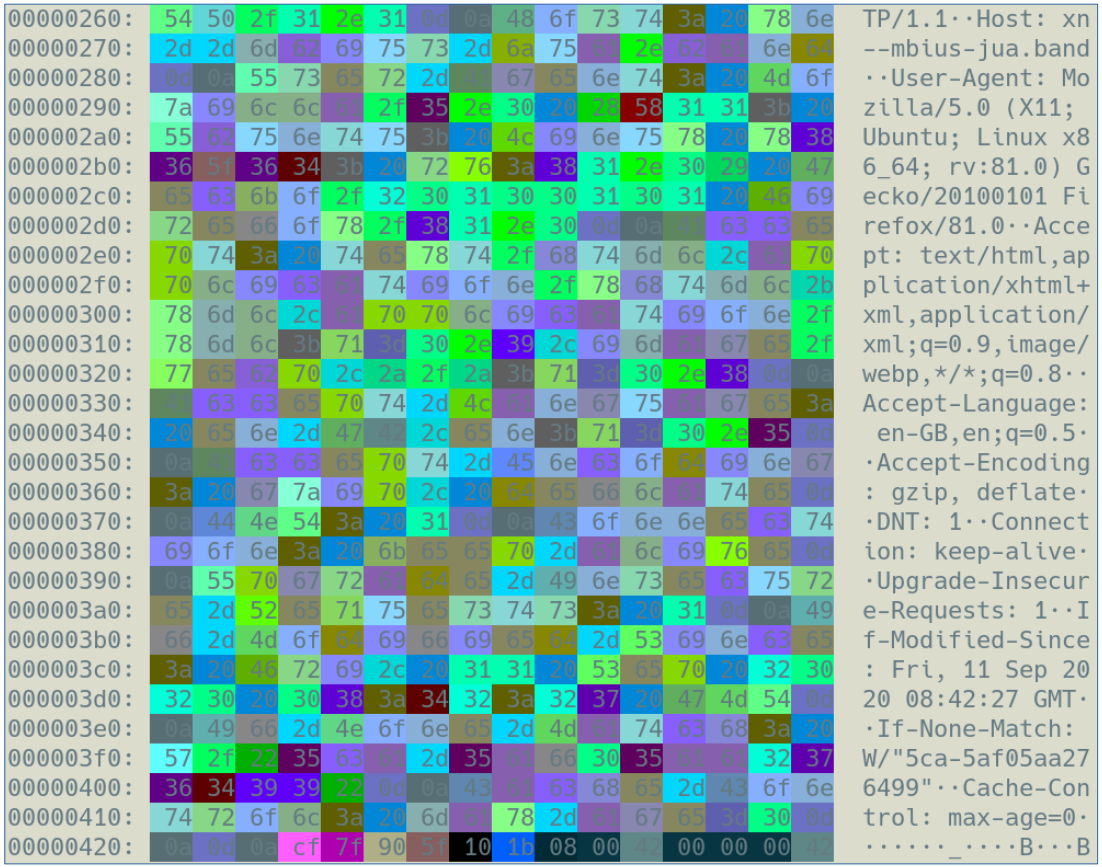
Brandenburg University of Technology Cottbus - Senftenberg

Network Data

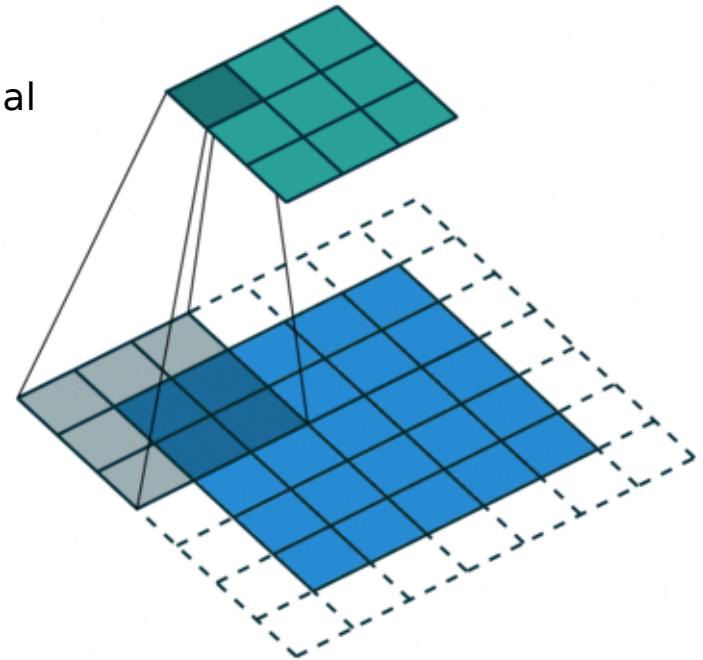
- Series of *packets*
- Weak order of packets
- Packets consist of *layers*
- Different types of protocols
- Layers have *header* and *payload* fields

Industrial Control Systems (*ICS*)

- Monitoring & Control functions
- Fixed network topology
- Proprietary Infrastructure
- Cycle information exchange

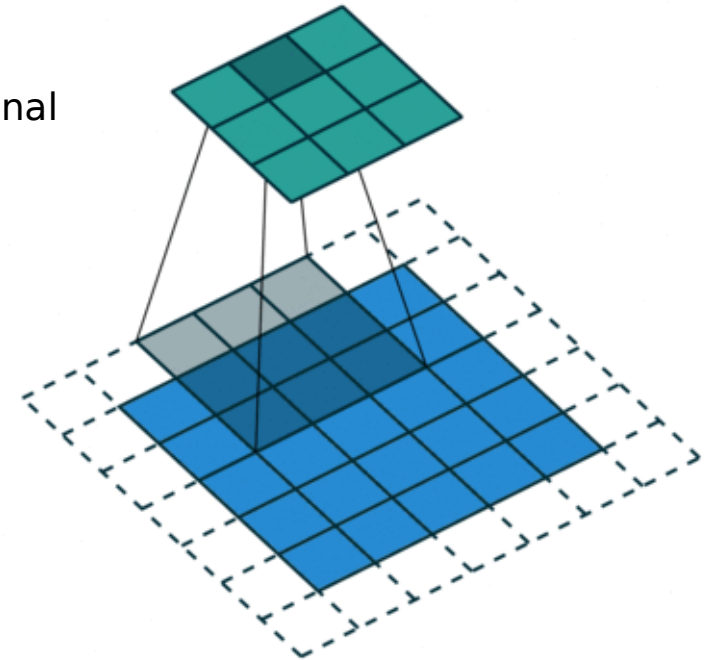


- Neurons that *convolve (dot product)* a filter with the input signal
- Filters/Kernels are adjustable through learning
- Hierarchical layers of neurons learn spatial patterns
- Often used for images based signal processing



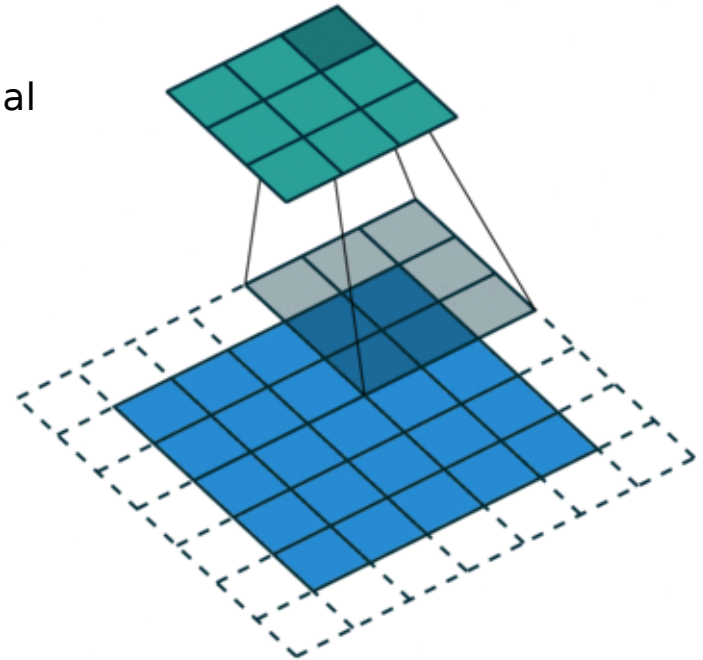
github.com/vdumoulin/conv_arithmetic

- Neurons that *convolve (dot product)* a filter with the input signal
- Filters/Kernels are adjustable through learning
- Hierarchical layers of neurons learn spatial patterns
- Often used for images based signal processing



github.com/vdumoulin/conv_arithmetic

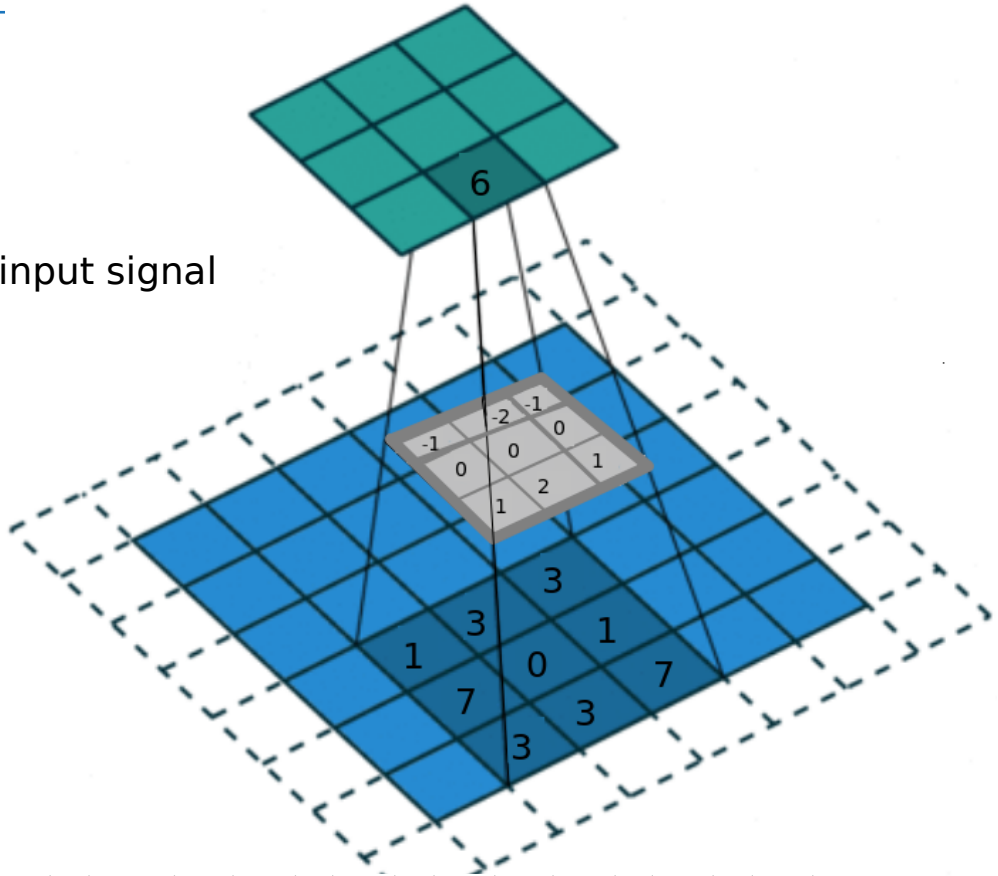
- Neurons that *convolve (dot product)* a filter with the input signal
- Filters/Kernels are adjustable through learning
- Hierarchical layers of neurons learn spatial patterns
- Often used for images based signal processing



github.com/vdumoulin/conv_arithmetic

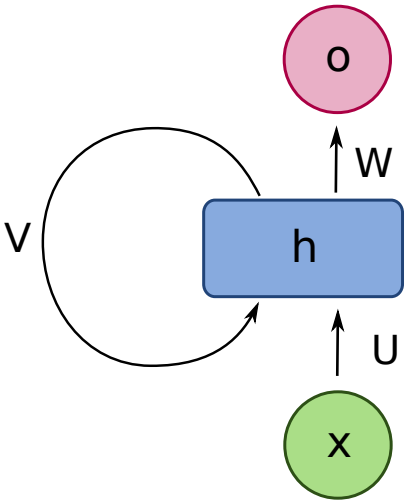
A | Appendix: **Convulutional Neurons**

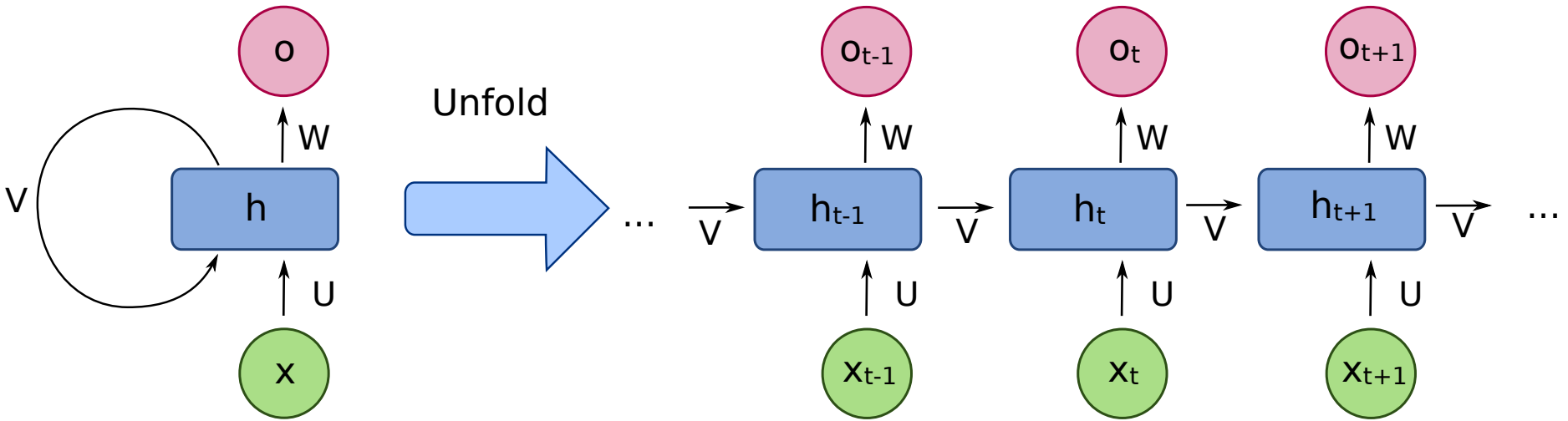
- Neurons that *convolve (dot product)* a filter with the input signal
- Filters/Kernels are adjustable through learning
- Hierarchical layers of neurons learn spatial patterns
- Often used for images based signal processing



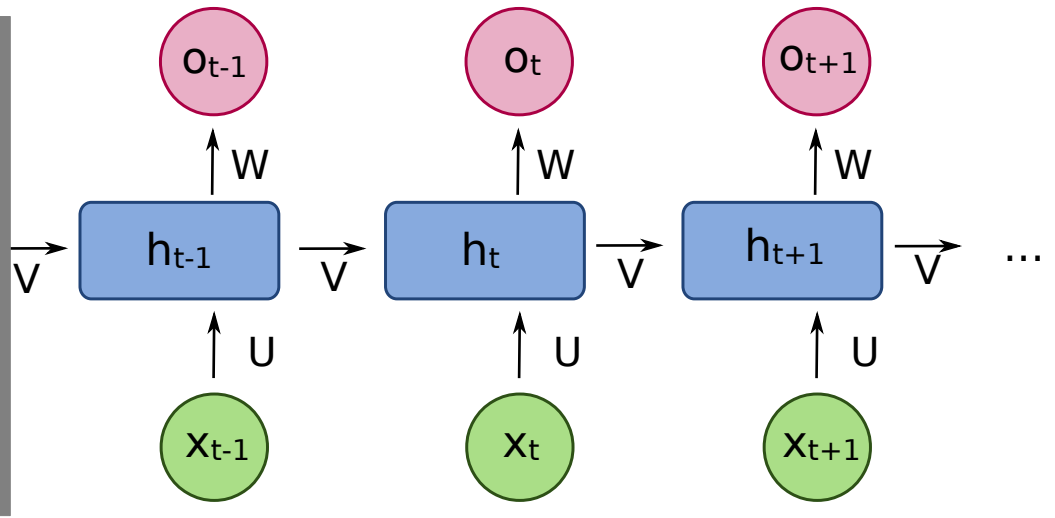
$$[(1*-1)+(3*-2)+(3*-1)]+[(7*0)+(0*0)+(1*0)]+[(3*1)+(3*2)+(7*1)]=6$$

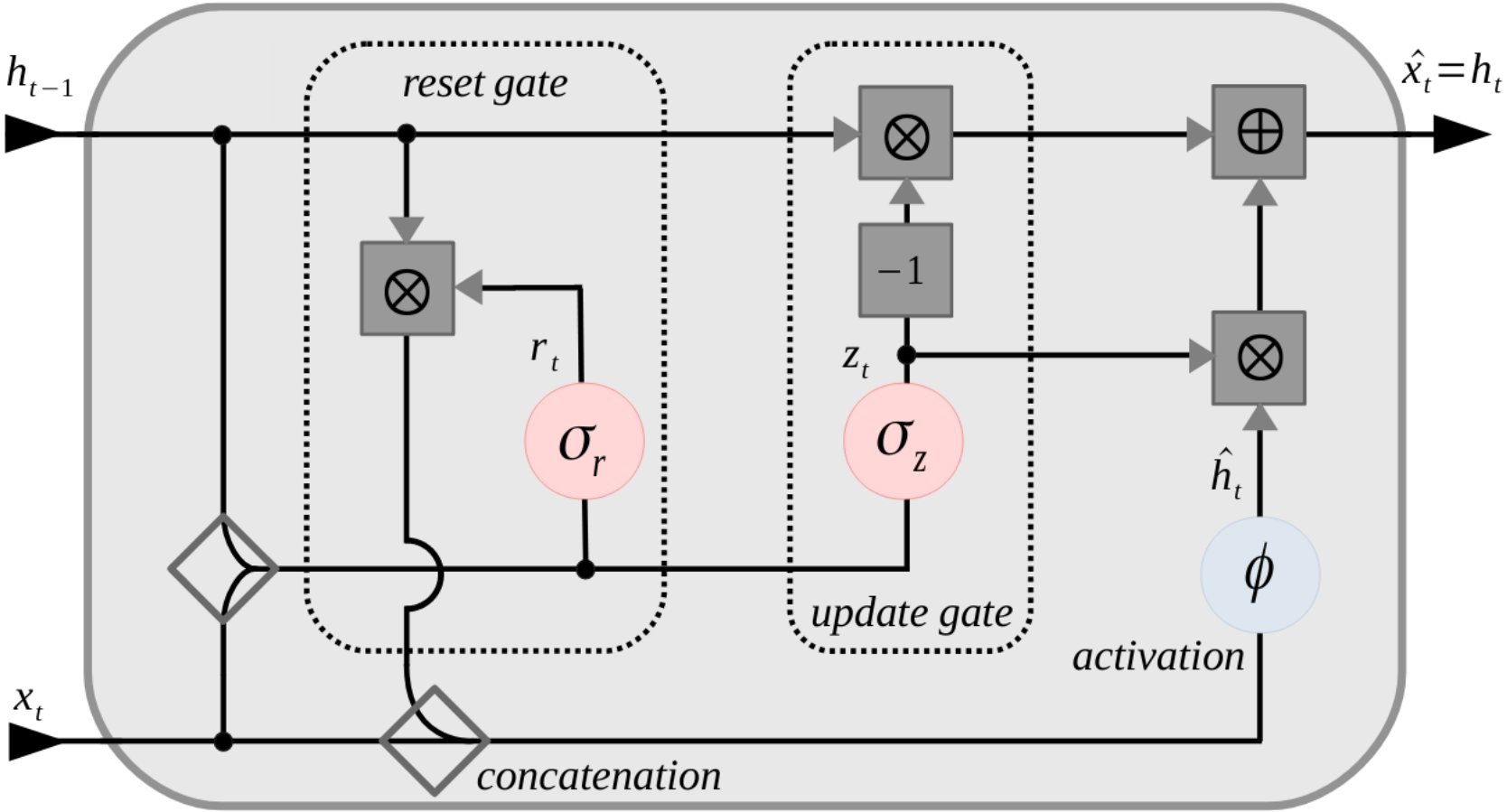
github.com/vdumoulin/conv_arithmetic

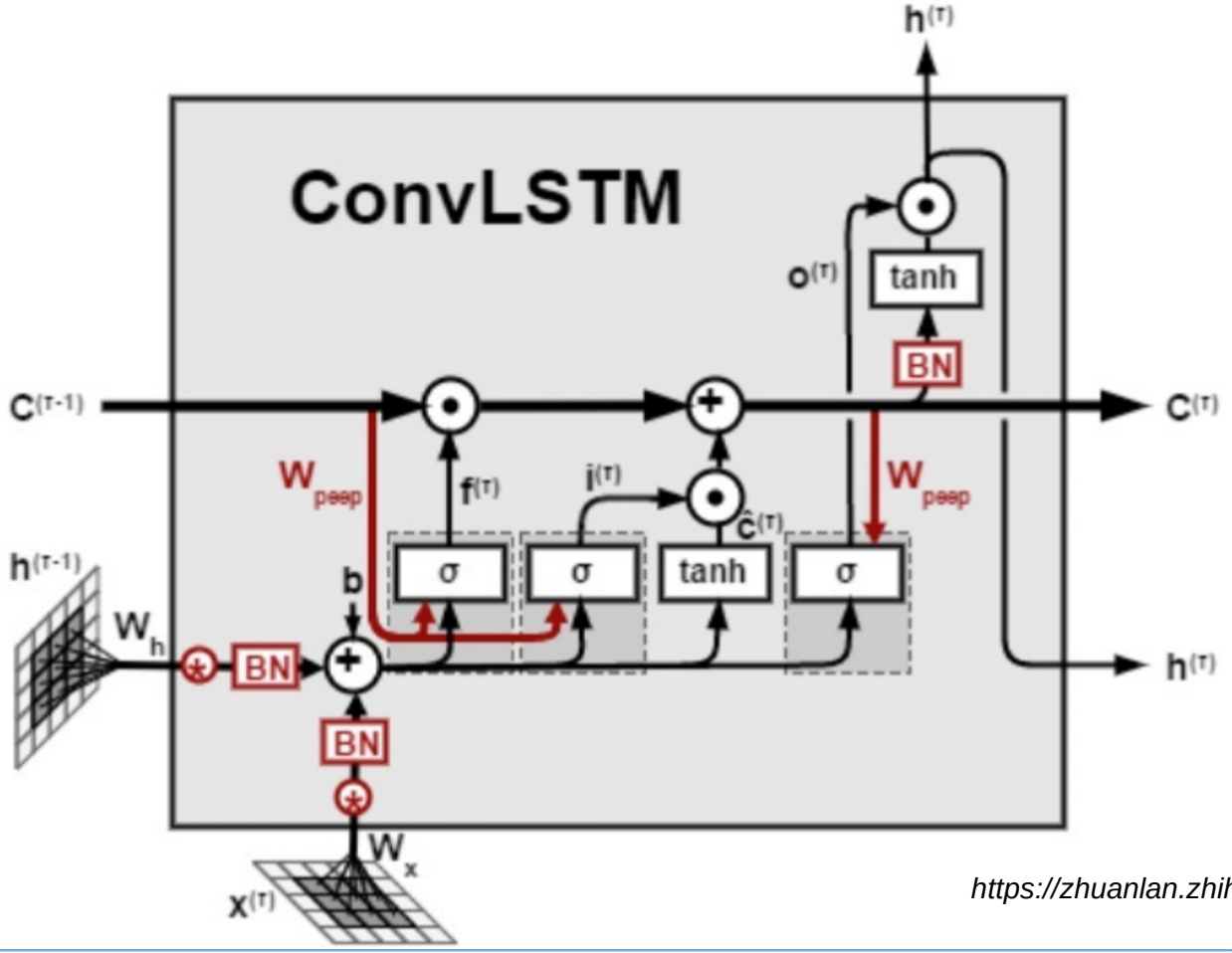




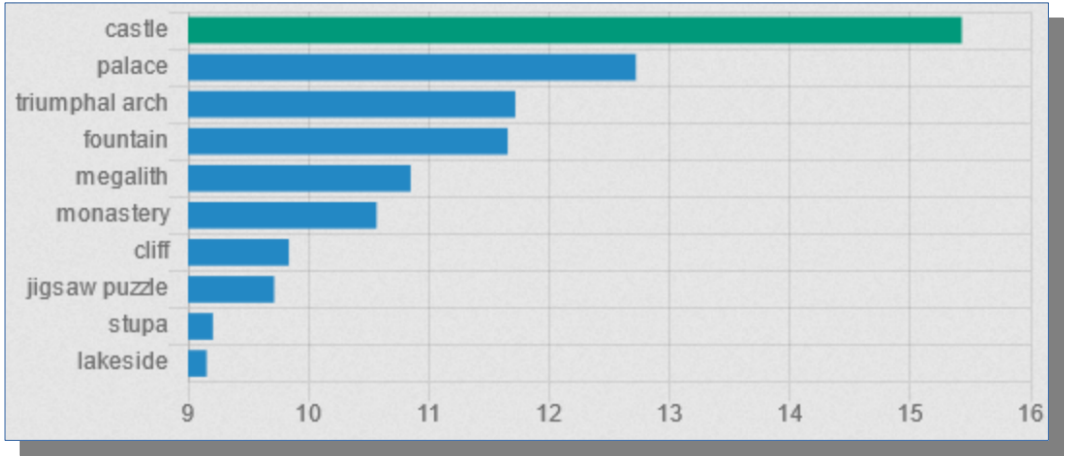
- Incorporate historical data
- Output signal is used for the next step
- Designed for sequential problems
- Used in generative models
- Complex units can host a number of gates

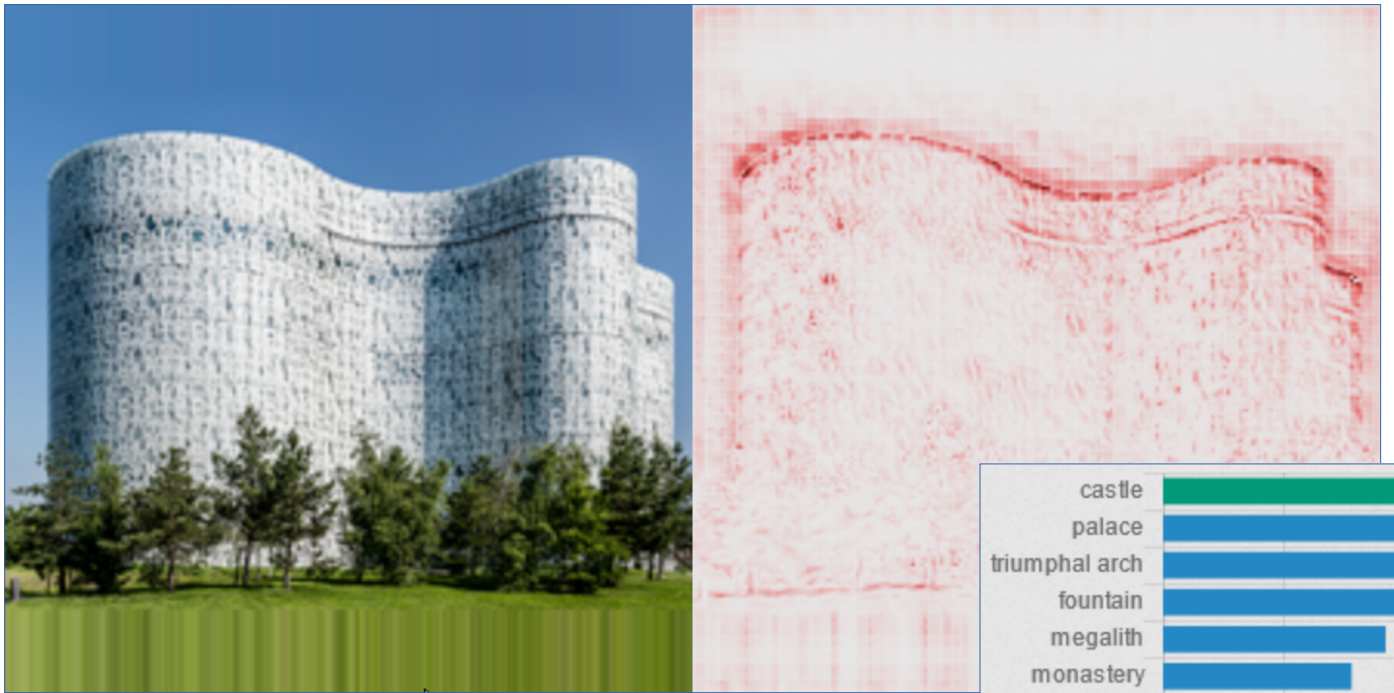



h_{t-1}
reset gate
$\hat{x}_t = h_t$
-1
r_t
z_t
σ_r
σ_z
$\hat{h}_t$
ϕ
update gate
activation
x_t
concatenation

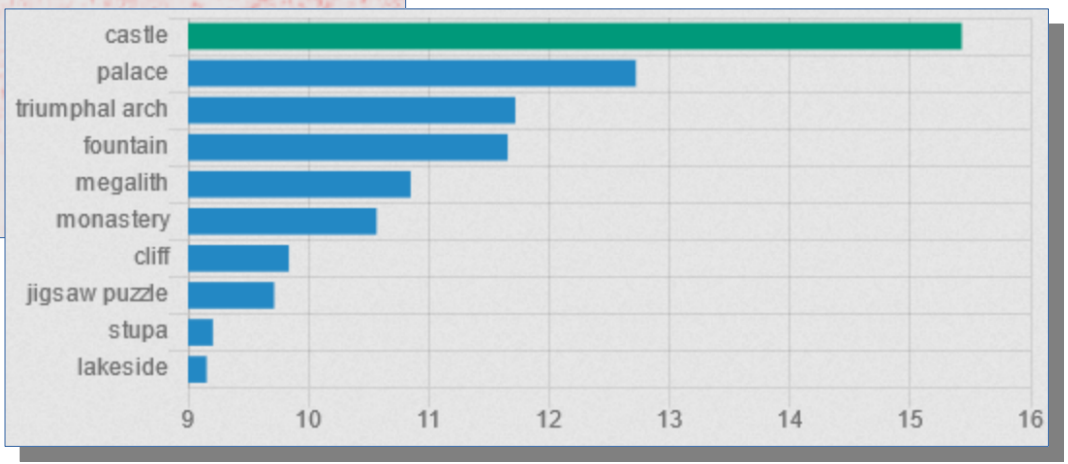


https://zhuanlan.zhihu.com/p/148122328



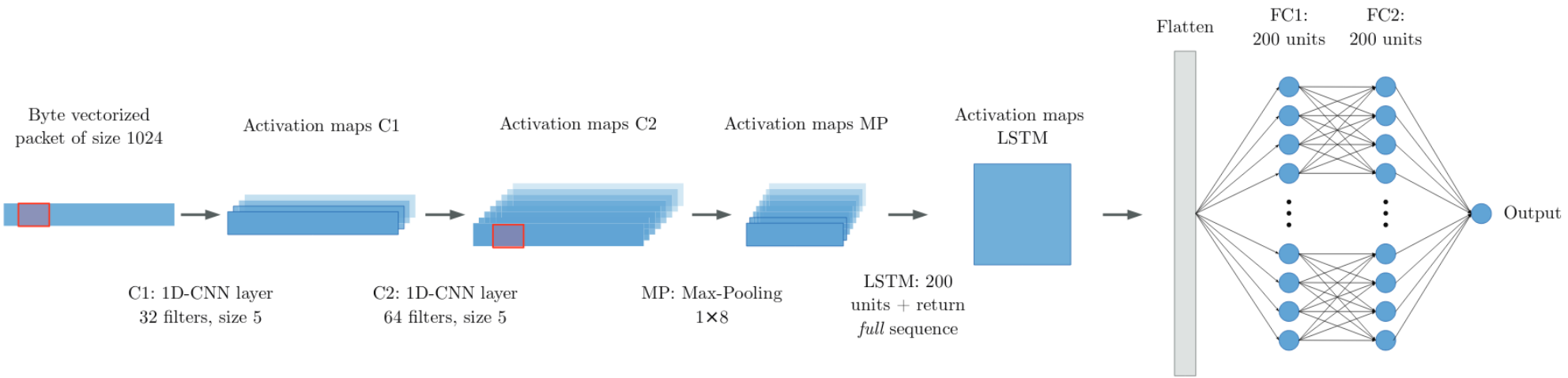


lrpserver.hhi.fraunhofer.de



DeepMAL - Deep Learning Models for Malware Traffic Detection and Classification
(P. Casas et al. – iDSC 2020)

- Feature extraction on first *1000* bytes of every packet / flow
- Spatial–Temporal representation learning on raw traffic (+)
- Supervised softmax-based classification (–)
- Comparison to shallow methods & handcrafted features extraction
- Evaluated on ICS unrelated data



An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection

- *IEEE Access, 2020*

An LSTM-Based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level

- *Applied Sciences, 2019*

ZOE: Content-based Anomaly Detection for Industrial Control Systems

- *IEEE/IFIP DSN, 2018*

A Deep Learning Approach for Network Intrusion Detection System

- *ACM BIONETICS, 2016*

Deep packet: a novel approach for encrypted traffic classification using deep learning

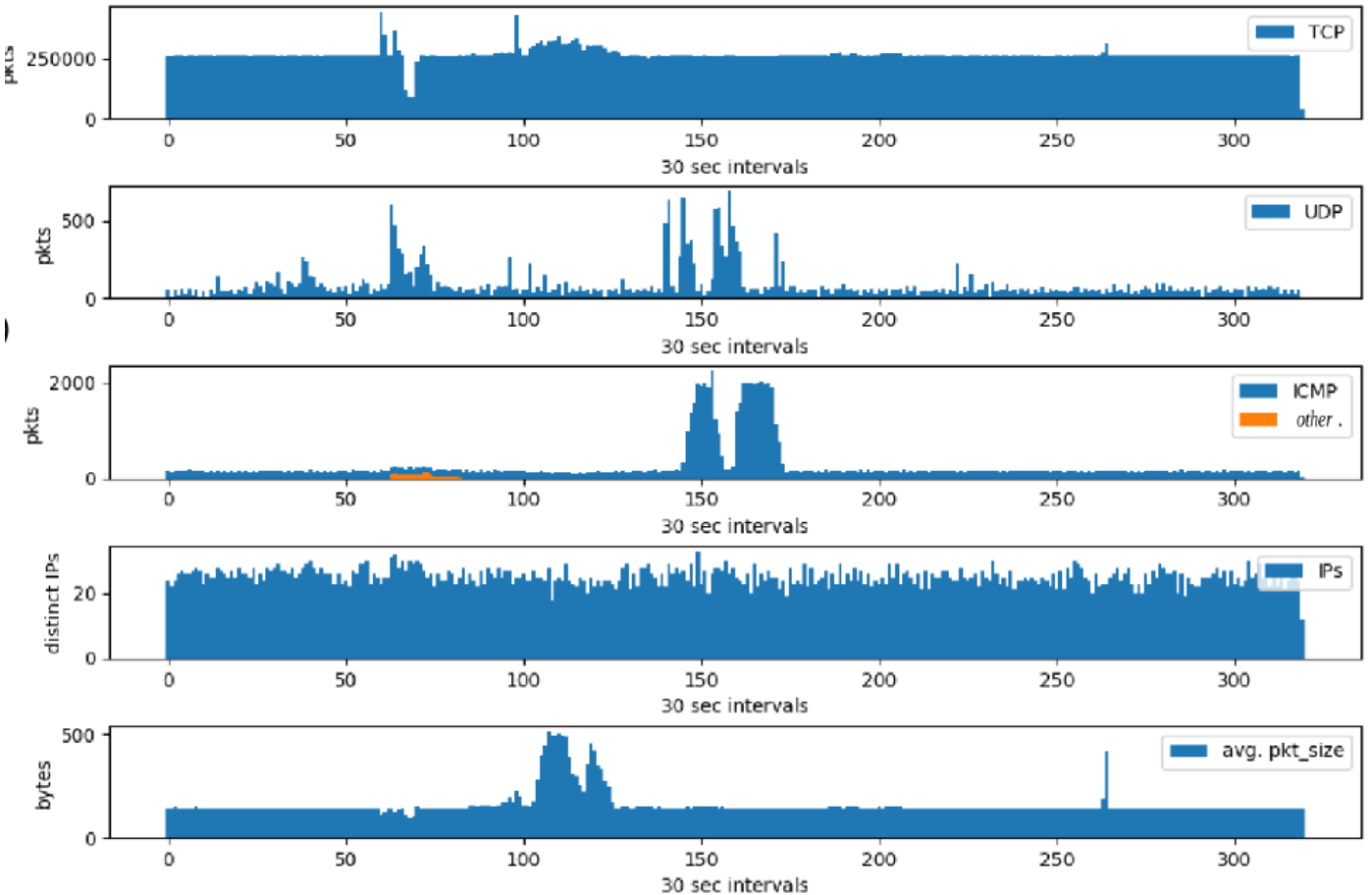
- *Soft Computing, 2020*

HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection

- *IEEE ACCESS, 2017*

Malware Traffic Classification Using Convolutional Neural Network for Representation Learning

- *IEEE ICOIN, 2017*


TCP
250000
0
0
50
100
150
200
250
300
30 sec intervals
UDP
pkts
500
0
30 sec intervals
ICMP
other .
pkts
2000
0
30 sec intervals
IPs
distinct IPs
20
0
30 sec intervals
avg. pkt_size
bytes
500
0
30 sec intervals

	mean	median	min	max	std	IQR
PCA	$-1.26 \cdot 10^{-7}$	0.4823	-4.0832	7.1323	0.3601	0.4690
k-PCA	$-1.85 \cdot 10^{-8}$	-0.0003	-4.0831	7.1324	0.3670	0.4781
ICA	$-1.51 \cdot 10^{-9}$	$-2.60 \cdot 10^{-5}$	-0.0539	0.0661	0.00617	0.008
Isomap	n/a	n/a	n/a	n/a	n/a	n/a
LLE	$1.66 \cdot 10^{-7}$	$-2.74 \cdot 10^{-5}$	-0.3514	0.3098	0.0061	0.0025
SE	n/a	n/a	n/a	n/a	n/a	n/a
AE	0.0162	0.0016	-0.833	0.7694	0.341	0.4591
original	0.4844	0.4823	0	1	0.2924	0.5098

	non-linearity	OOS processing	ability to retrain	space complexity	time complexity
PCA		✗		$\mathcal{O}(N * D)$	$\mathcal{O}(N * D + D^3)$
k-PCA	✗	✗		$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$
ICA	✗	✗		$\mathcal{O}(N^2)$	$\mathcal{O}(D^2 * N)$
Isomap	✗	✗		$\mathcal{O}(d * N^2)$	$\mathcal{O}(N^2)$
LLE	✗			$\mathcal{O}(N^2)$	$\mathcal{O}(d * N^2)$
SE	✗			$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$
AE	✗	✗	✗	$\mathcal{O}(2 * \theta)$	$\mathcal{O}(epoch * D * N)$

Table 5.1: Comparison of different dimension reduction methods

	$Train$	$Validation$	$Eval_H$	$Eval_D$	$Eval_E$	$Eval_R$
Duration [sec]	1139.7	162	2907	702	702	702
Size [GB]	1.7	0.24	0.72	1.2	1.2	1.2
# Frames	10,774,388	1,539,556	1,518,761	3,521,649	6,157,648	5,232,729
# Anomalies (%)	0	0	424,364 (27.9)	135,048 (3.8)	2,460 (0.04)	1,930 (0.037)
# MACs	39	33	110	37	57	54
# IP	42	34	34	38	40	40
# Protocols	25	19	21	20	22	22
# Flows [tcp/udp]	993/417	144/51	6,648/39	255/95	448/161	377/136
ATU (std)	146.3 (166)	145.6 (158)	477.8 (1228)	144.6 (12)	146.3(0.5)	146.3(0.5)
$k$packets/sec	0.4	9.5	0.5	5.0	8.7	7.5

Table 4.1: V traffic statistics per sub set

	$Train_{RL}$	$Validation_{RL}$	$Train_{AD}$	$Validation_{AD}$
Duration [sec]	898.6	4.1	408.5	4.1
Size [GB]	1.4	0.14	1.4	0.14
# Frames	10,000,000	100,000	10,000,000	100,000
# Anomalies [%]	0	0	0	0
# MACs	54	31	57	33
# IP	51	32	60	35
# Protocols	36	21	39	21
# Flows [tcp/udp]	87,961 / 172	971/19	87,368/236	1,572/21
ATU (std)	113.23 (105)	108.23 (104)	113.38 (105)	112.55 (105)
*k*packets/sec	11.1	71.4	24.0	24.0
	$Eval_H$	$Eval_D$	$Eval_E$	$Eval_R$
Duration [sec]	299.7	487.6	487.2	486.9
Size [GB]	1.1	1.3	1.3	1.3
# Frames	7,350,000	6,284,294	6,194,072	6,194,576
# Anomalies [%]	18,900 (0.25)	170,608 (2.7)	516 (0.008)	148 (0.002)
# MACs	56	56	67	67
# IP	52	48	48	48
# Protocols	37	34	36	33
# Flows [tcp/udp]	70,316/148	28,649/119	28,649/119	28,649/122
ATU (std)	114.44 (117)	113.65 (322)	113.12 (101)	113.41 (103)
*k*packets/sec	24.0	12.8	12.7	12.7

Table 4.2: SWaT traffic statistics per sub set

RNN-Cell	*Loss Function*	*Optimization*	*LR-Scheduler*	*Loss after Epoch 5*
GRU	MSE	SGD	cycle	$2.82 \cdot 10^{-4}$
GRU	MSE	SGD	plateau	$1.72 \cdot 10^{-4}$
GRU	MSE	SGD	step	$1.72 \cdot 10^{-4}$
GRU	MSE	adamW	cycle	$2.34 \cdot 10^{-5}$
GRU	MSE	adamW	step	$6.11 \cdot 10^{-5}$
GRU	MSE	adamW	plateau	$1.26 \cdot 10^{-4}$
GRU	BCE	SGD	cycle	$1.14 \cdot 10^{-3}$
GRU	BCE	SGD	step	$2.93 \cdot 10^{-4}$
GRU	BCE	SGD	plateau	$6.34 \cdot 10^{-4}$
GRU	BCE	adamW	cycle	$3.15 \cdot 10^{-5}$
GRU	BCE	adamW	step	$6.92 \cdot 10^{-5}$
GRU	BCE	adamW	plateau	$1.71 \cdot 10^{-4}$
LSTM	MSE	SGD	cycle	$9.49 \cdot 10^{-5}$
LSTM	MSE	SGD	step	$5.81 \cdot 10^{-5}$
LSTM	MSE	SGD	plateau	$5.81 \cdot 10^{-5}$
LSTM	MSE	adamW	cycle	$2.21 \cdot 10^{-5}$
LSTM	MSE	adamW	step	$2.84 \cdot 10^{-4}$
LSTM	MSE	adamW	plateau	$7.16 \cdot 10^{-4}$
LSTM	BCE	SGD	cycle	$9.75 \cdot 10^{-5}$
LSTM	BCE	SGD	step	$6.52 \cdot 10^{-5}$
LSTM	BCE	SGD	plateau	$6.33 \cdot 10^{-5}$
LSTM	BCE	adamW	cycle	$2.39 \cdot 10^{-5}$
LSTM	BCE	adamW	step	$1.15 \cdot 10^{-3}$
LSTM	BCE	adamW	plateau	$9.35 \cdot 10^{-4}$

Table 6.1: Hyperparameter grid search results

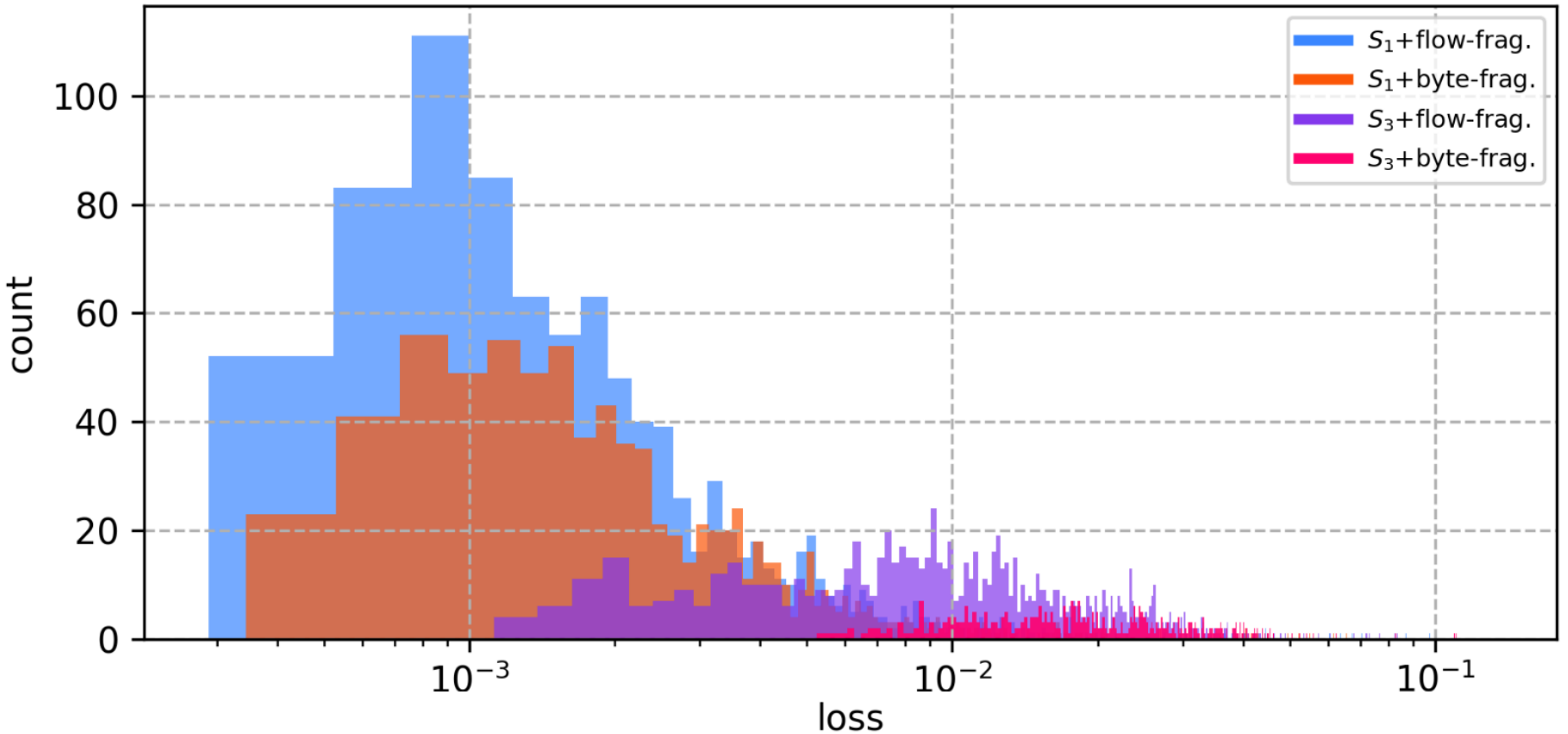


Figure 6.3: Training loss distribution for the SWaT A6 data set

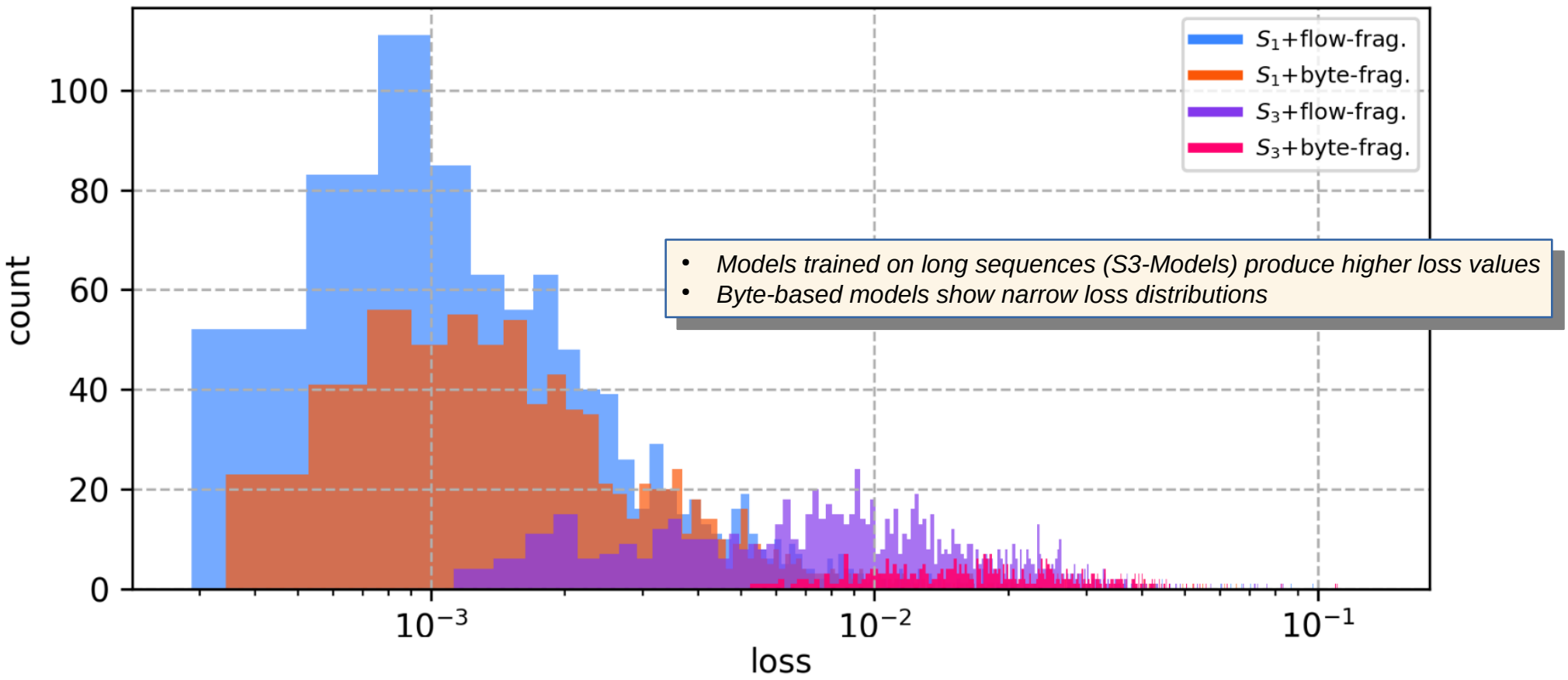


Figure 6.3: Training loss distribution for the SWaT A6 data set

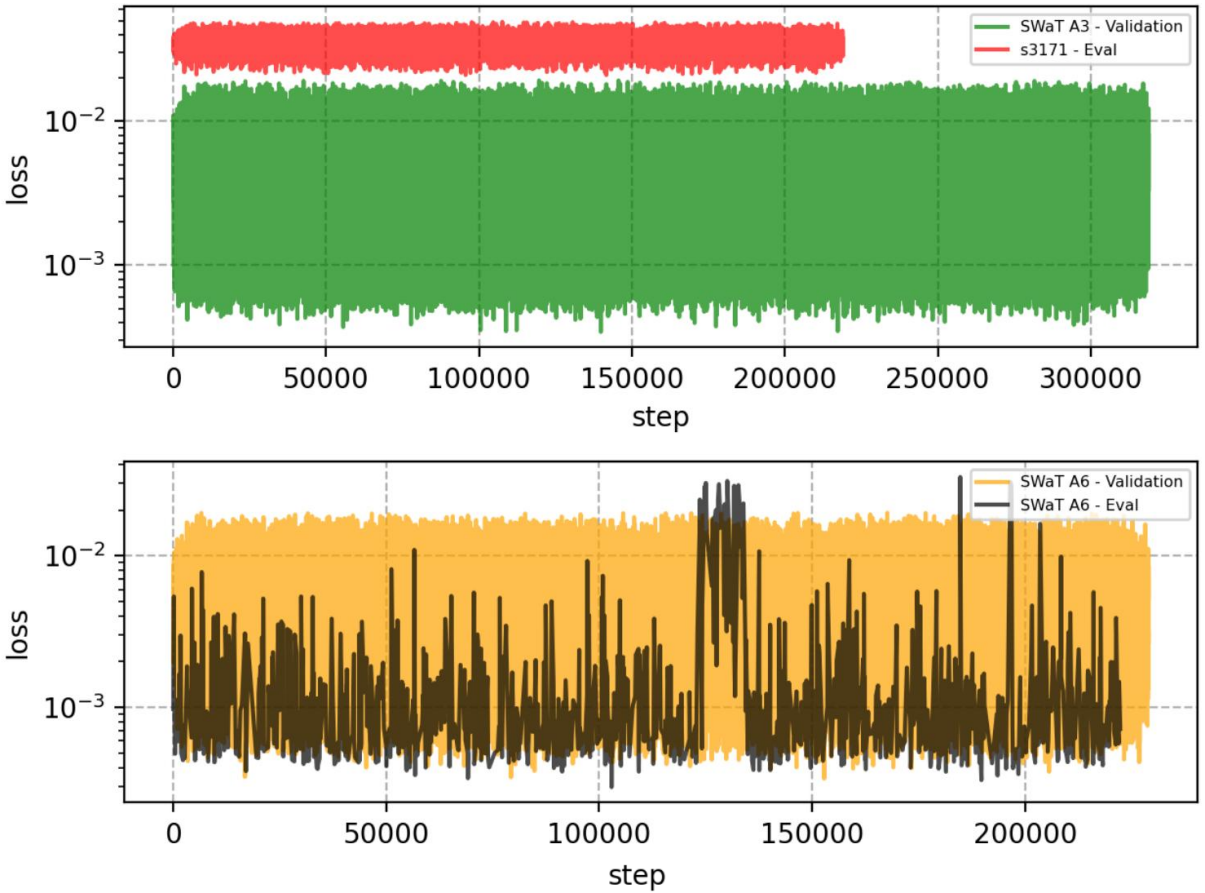
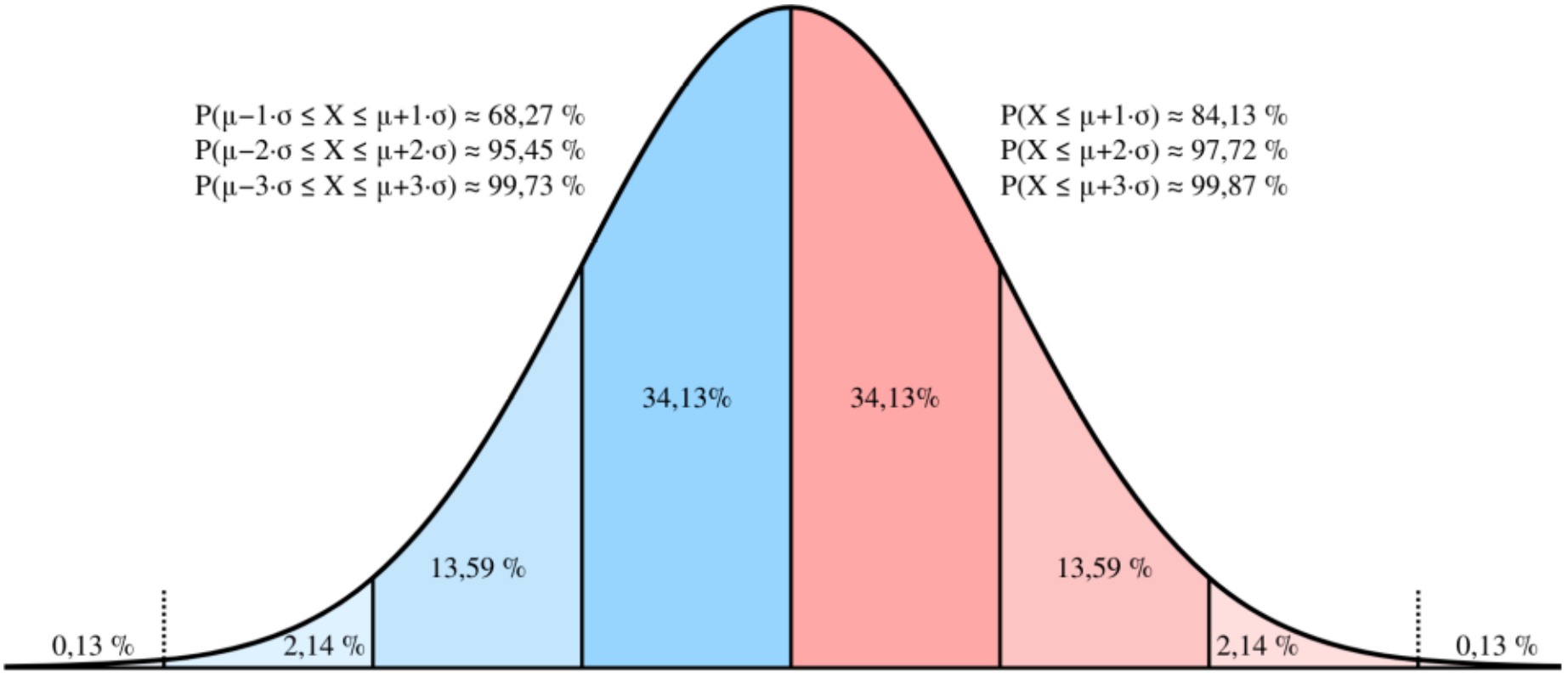
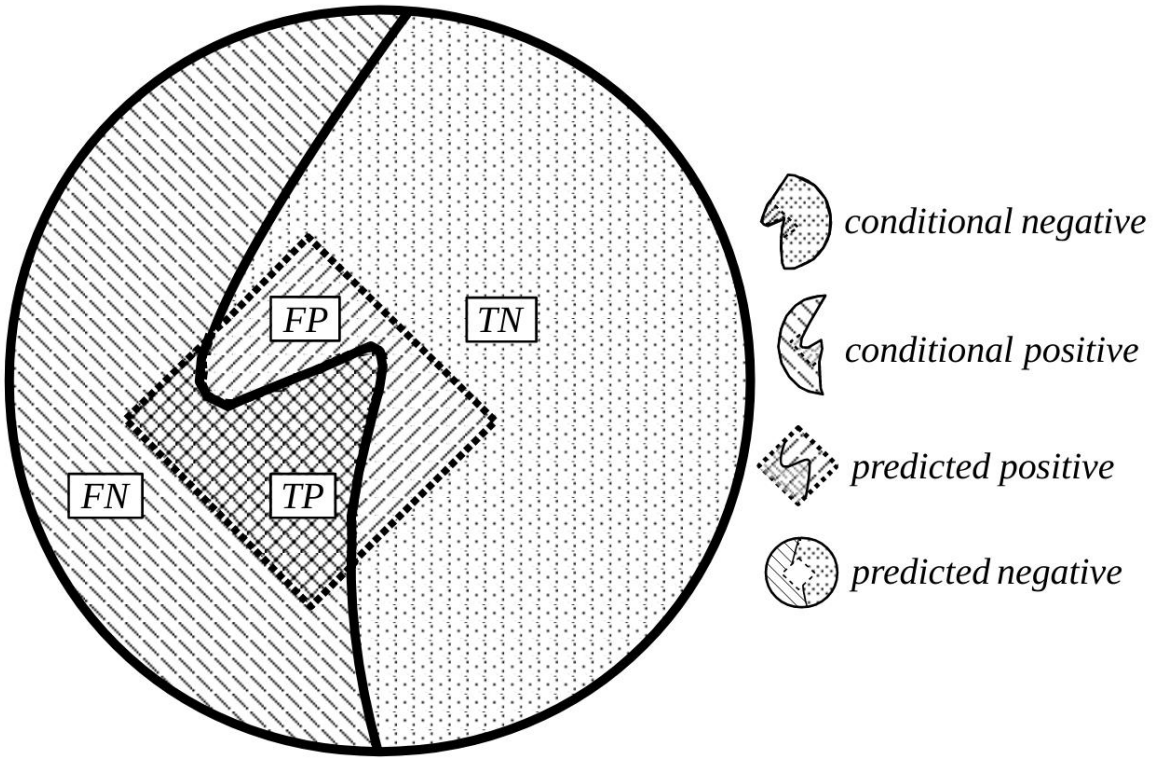


Figure 6.1: Residual loss for different iTrust data sets


P(μ−1·σ ≤ X ≤ μ+1·σ) ≈ 68,27 %
P(μ−2·σ ≤ X ≤ μ+2·σ) ≈ 95,45 %
P(μ−3·σ ≤ X ≤ μ+3·σ) ≈ 99,73 %
P(X ≤ μ+1·σ) ≈ 84,13 %
P(X ≤ μ+2·σ) ≈ 97,72 %
P(X ≤ μ+3·σ) ≈ 99,87 %
34,13%
34,13%
13,59 %
13,59 %
0,13 %
2,14 %
2,14 %
0,13 %


FP
TN
FN
TP
conditional negative
conditional positive
predicted positive
predicted negative

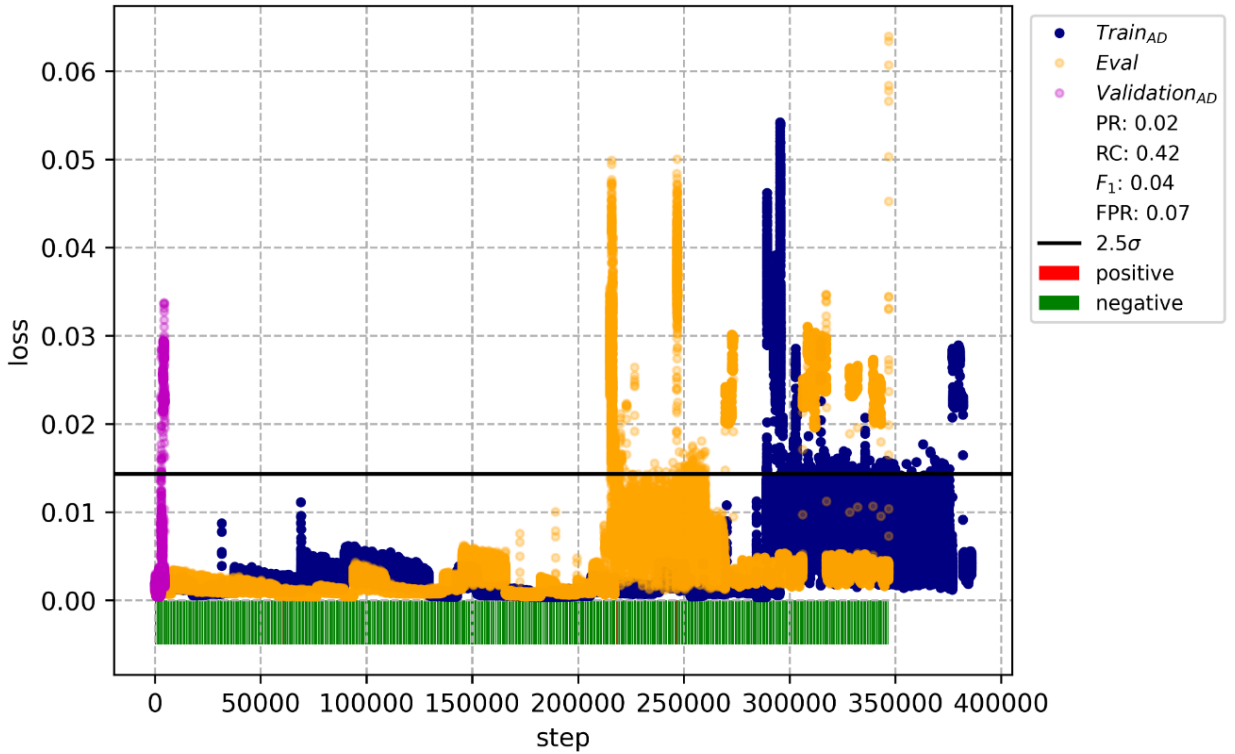


Figure B.3: SWaT A6 $Eval_H$ naive baseline loss visualisation for flow-fragments trained on an input sequence length of $n = 1$

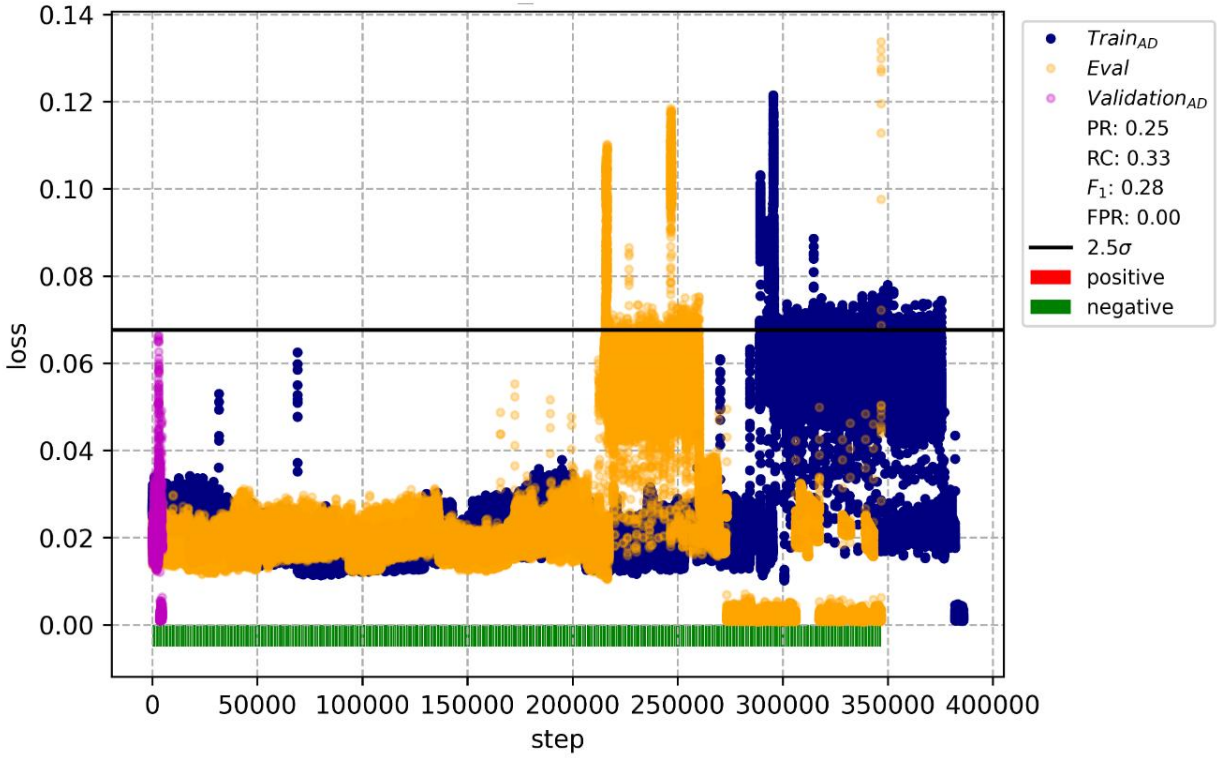


Figure B.4: SWaT A6 $Eval_H$ naive baseline loss visualisation for flow-fragments trained on an input sequence length of $n = 3$

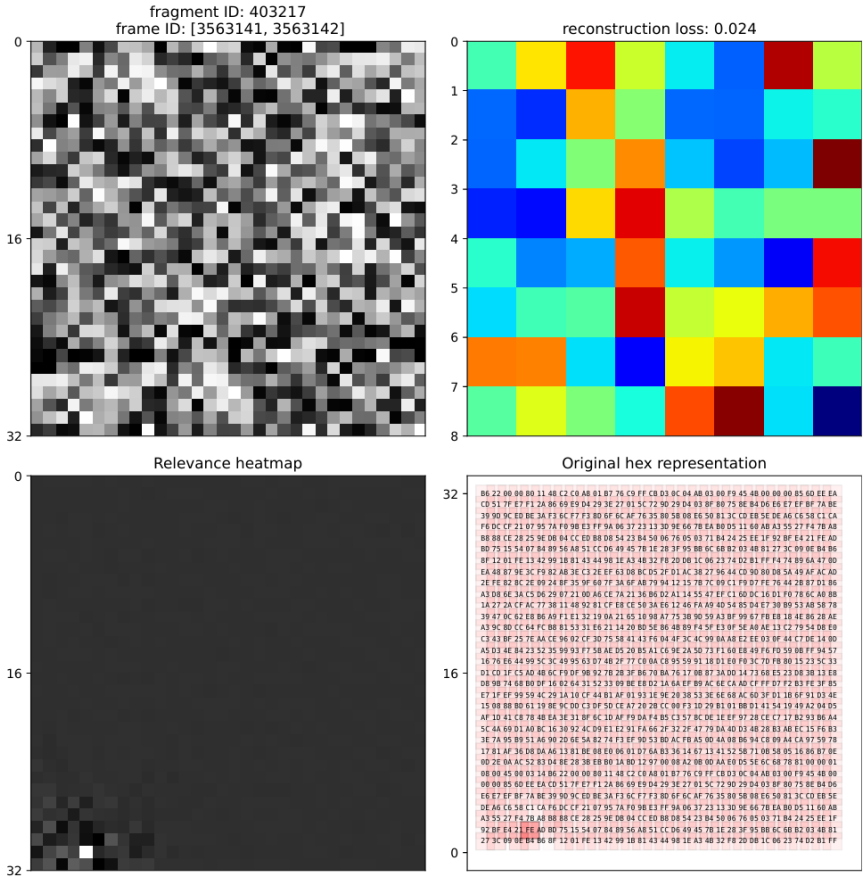


Figure B.9: Byte-fragments relevance heatmaps of the unusual UDP packet

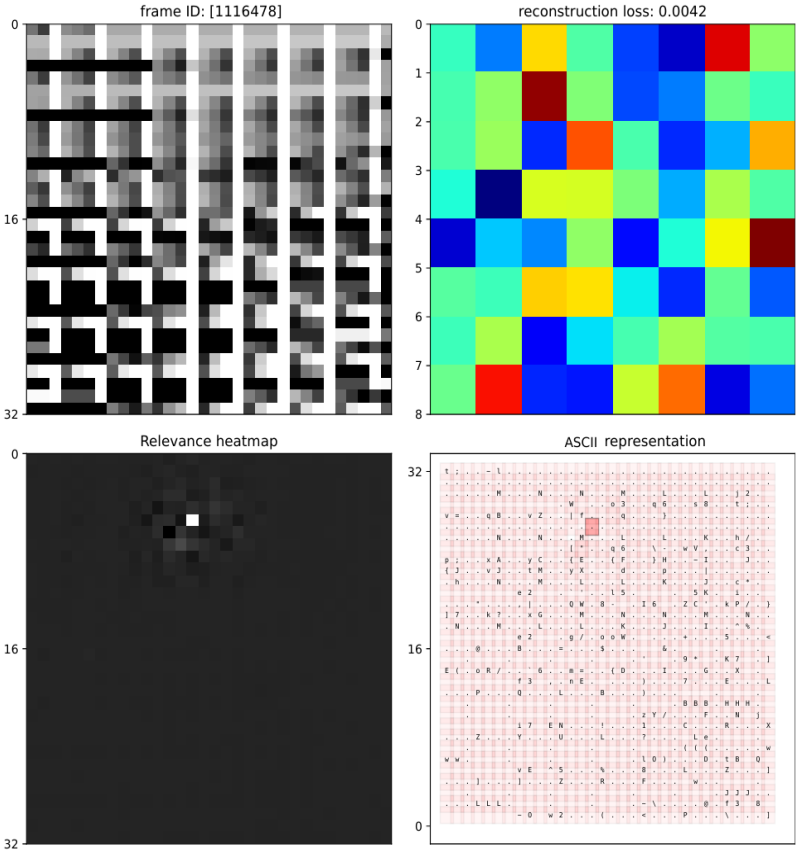


Figure 6.10: Byte-fragments relevance heatmaps of the SWaT file download event

	Time	Source	Destination	Protocol	Leng	Info
844…	3.420098	192.168.1.183	118.201.255.203	UDP	802	54028 → 1195 Len=760
84498	3.420099	192.168.1.183	118.201.255.203	UDP	806	54028 → 1195 Len=760
84499	3.420186	192.168.1.183	239.255.255.250	UDP	698	54029 → 3702 Len=656

```
Frame 84497: 802 bytes on wire (6416 bits), 802 bytes captured (6416 bits) on interface 0
Ethernet II, Src: Giga-Byt_6c:68:78 (e0:d5:5e:6c:68:78), Dst: AdiEngin_0b:0d:aa (00:08:a2:0b:0d:aa)
Internet Protocol Version 4, Src: 192.168.1.183, Dst: 118.201.255.203
User Datagram Protocol, Src Port: 54028, Dst Port: 1195
Data (760 bytes)
```

```
000  00 08 a2 0b 0d aa e0 d5  5e 6c 68 78 08 00 45 00   ........ ^lhx..E.
010  03 14 b6 22 00 00 80 11  48 c2 c0 a8 01 b7 76 c9   ..."....  H.....v.
020  ff cb d3 0c 04 ab 03 00  f9 45 4b 00 00 00 85 6d   ........  .EK....m
030  ee ea cd 51 7f e7 f1 2a  86 69 e9 d4 29 3e 27 01   ...Q...*  .i..)>'.
040  5c 72 9d 29 d4 03 8f 80  75 8e b4 d6 e6 e7 ef bf   \r.)....  u.......
050  7a be 39 9d 9c ed be 3a  f3 6c f7 f3 8d 6f 6c af   z.9....:  .l...ol.
060  76 35 80 5b 08 e6 50 81  3c cd eb 5e de a6 c6 58   v5.[..P.  <..^...X
070  c1 ca f6 dc cf 21 07 95  7a f0 9b e3 ff 9a 06 37   .....!..  z......7
080  23 13 3d 9e 66 7b ea b0  d5 11 60 ab a3 55 27 f4   #.=.f{..  ..`..U'.
090  7b a8 b8 88 ce 28 25 9e  db 04 cc ed b8 d8 54 23   {....(%.  ......T#
0a0  b4 50 06 76 05 03 71 b4  24 25 ee 1f 92 bf e4 21   .P.v..q.  $%.....!
```

Figure 6.9: *Wireshark* excerpt from a UDP packet with an unusual payload