3D-IDS: Doubly Disentangled Dynamic Intrusion Detection
NSE ML+Security Reading Group

Kim Hammar

kimham@kth.se

Division of Network and Systems Engineering
KTH Royal Institute of Technology

September 22, 2023

Context of the Paper

3D-IDS: Doubly Disentangled Dynamic Intrusion Detection

Chenyang Qiu
Beijing University of Posts and Telecommunications
cyqiu@bupt.edu.cn

Yingsheng Geng
Beijing University of Posts and Telecommunications
gyswasdfre2255@bupt.edu.cn

Junrui Lu
Beijing University of Posts and Telecommunications
672@bupt.edu.cn

Kaida Chen
Beijing University of Posts and Telecommunications
chenkaida@bupt.edu.cn

Shitong Zhu
Beijing University of Posts and Telecommunications
1337612789@bupt.edu.cn

Ya Su
HUAWEI Technologies Co., Ltd.
suya9@huawei.com

Guoshun Nan*
Beijing University of Posts and Telecommunications
nanguo2021@bupt.edu.cn

Can Zhang
Beijing University of Posts and Telecommunications
zhangcan_bupt@bupt.edu.cn

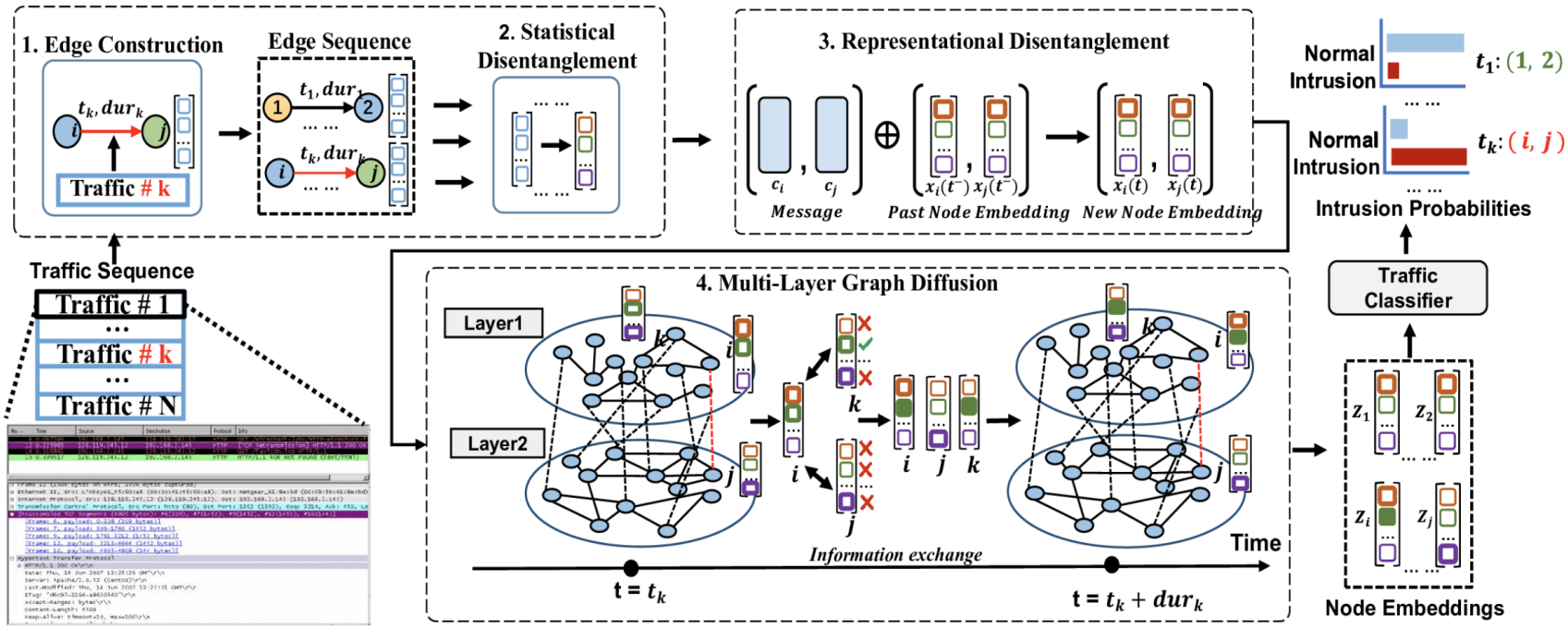
Junsong Fu
Beijing University of Posts and Telecommunications
fujs@bupt.edu.cn

Qimei Cui
Beijing University of Posts and Telecommunications
cuiqimei@bupt.edu.cn

Xiaofeng Tao
Beijing University of Posts and Telecommunications
taoxf@bupt.edu.cn

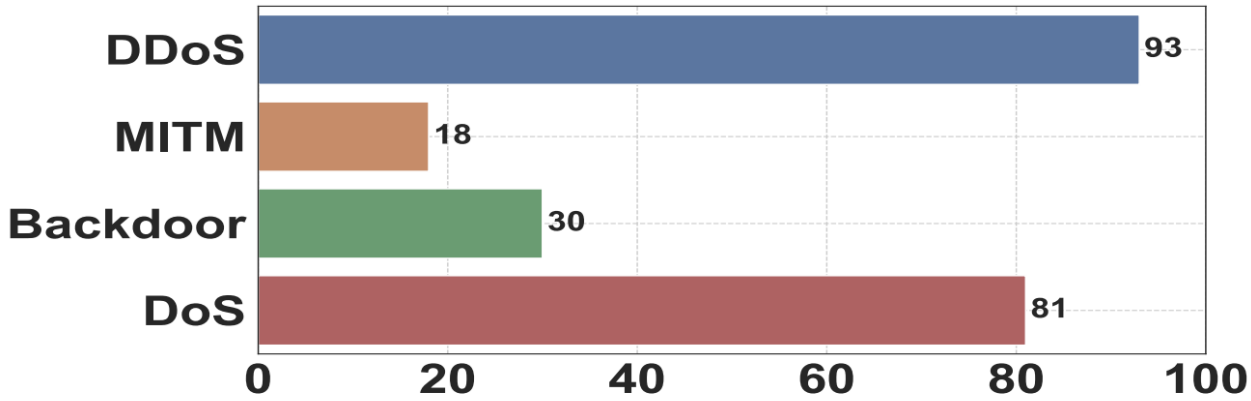
Published in KDD'23.

Context of the Paper



- Presents a new **Intrusion Detection System (IDS)** based on deep learning.
- The proposed IDS achieves state-of-the-art results on several benchmarks.

Motivation

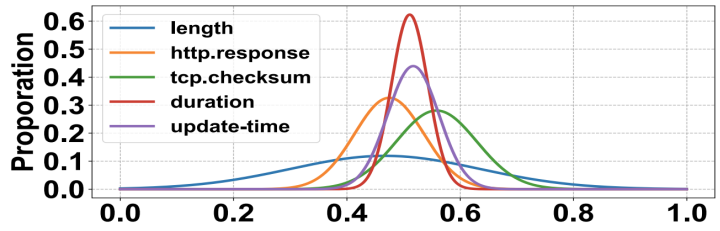


(d) F1-Score

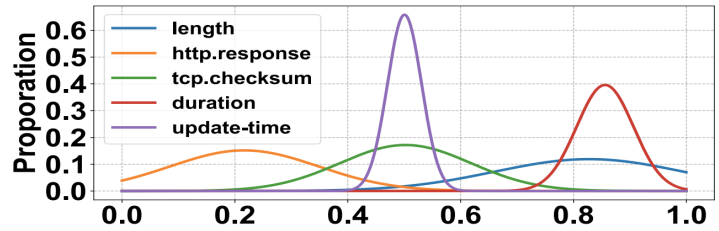
- Previous state-of-the-art deep learning IDSs **perform inconsistently** for detecting different types of attacks.
- DDoS attacks are detected reliably but MITM, injection and backdoor attacks are not detected very well.

- $\Longrightarrow$ To improve state-of-the-art, the focus should be on detecting the MITM/backdoor/injection attacks.

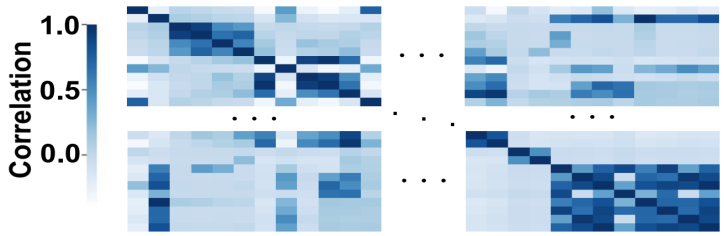
Why Do Existing IDSs Perform Inconsistently on Different Attacks?



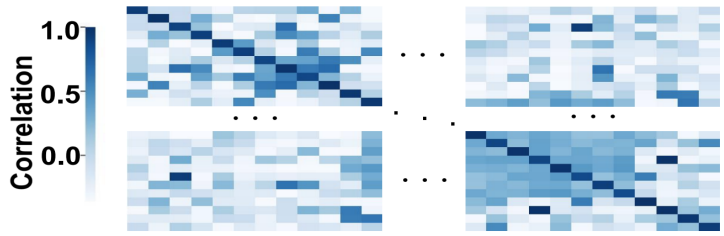
(b) Normalized feature distribution of MITM attacks



(c) Normalized feature distributions of DDoS attacks



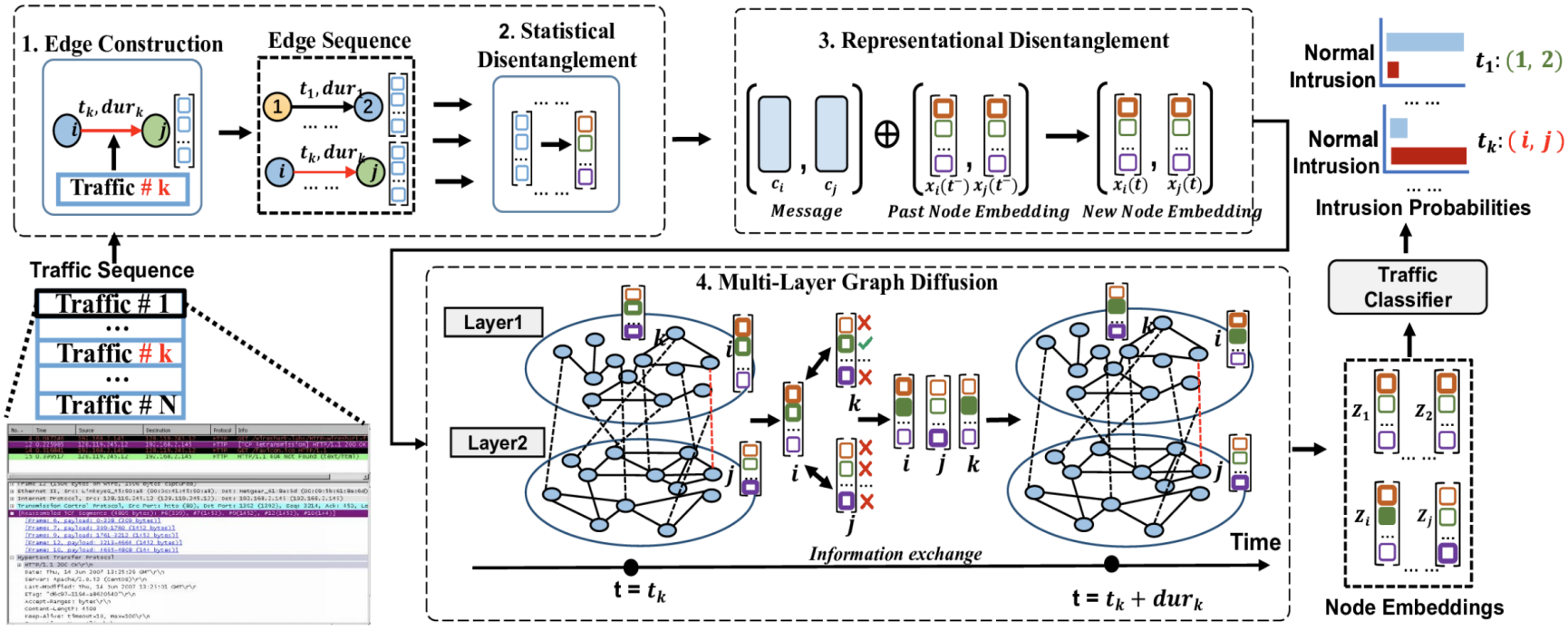
(e) Representation correlation map of MITM attacks



(f) Representation correlation map of DDoS attacks

- The authors argue that the **poor detection performance of certain attacks is due to entanglement of features**.
- The statistical distributions of different features (e.g., network traffic statistics) look identical to the model.
- For DDoS attacks the feature distributions are separated $\Longrightarrow$ Better Detection Performance.

Overview of 3D-IDS

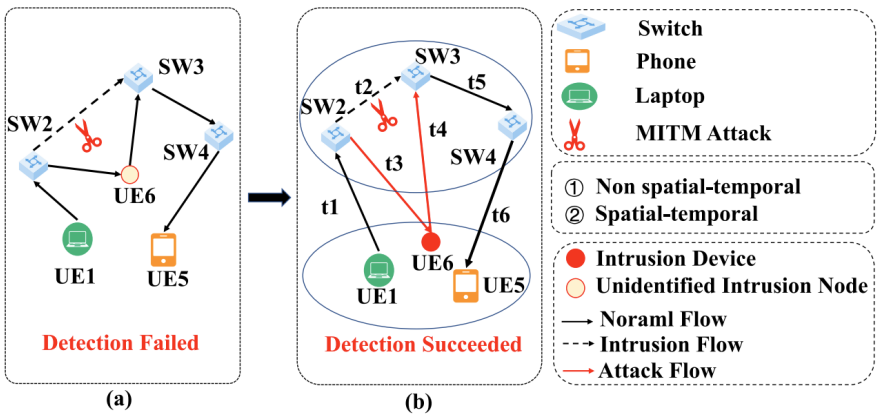


Graph Construction (1/2)

- Each device $i \in \mathcal{N}$ in the network is associated with a **level** l_i.
  - Terminal devices, e.g., PCs or an IoT device are in level $l_i = 0$
  - Routers and switches are in level $l_i = 1$
- For each netflow record (`source ip`, `destination ip`, timestamp t, flow duration Δt, `flow statistics`) the following edge is created in the graph:

$$\mathbf{E}_{ij}(t) = (v_i, l_i, v_j, l_j, t, \Delta t, \mathbf{F}_{ij}(t))$$

where v_i, v_j are the nodes, l_i, l_j are the node levels, and $\mathbf{F}_{ij}(t)$ are the flow statistics

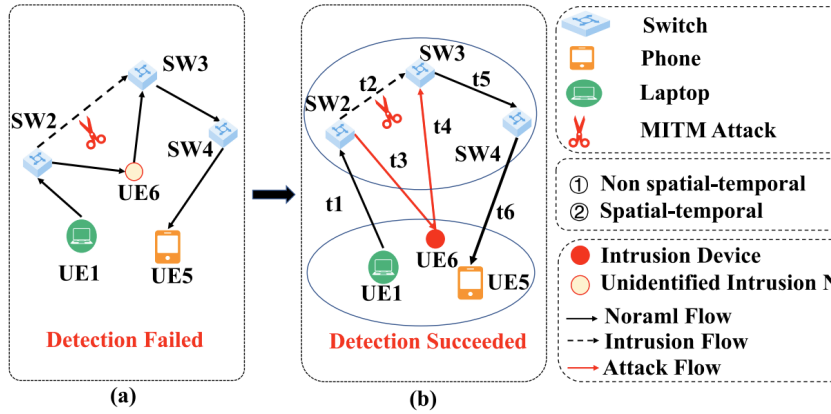


Graph Construction (2/2)

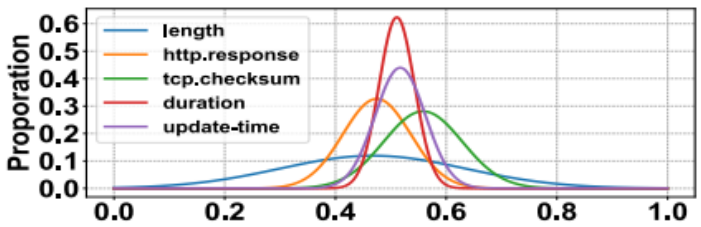
- The **sequence of NetFlow records generates a sequence of edges** $\{\mathcal{E}^t\}_{t=1}^{T}$ and thus a sequence of graphs $\{\mathcal{G}^t\}_{t=1}^{T}$.
- In other words, the graph is dynamic.
- The graph at each time-step t is modeled as a multi-layered graph:

$$\mathbb{A} = \begin{bmatrix} A_{(1,1)} & \cdots & A_{(1,k)} & \cdots & A_{(1,m)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{(l,1)} & \cdots & A_{(k,k)} & \cdots & A_{(l,m)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{(m,1)} & \cdots & A_{(m,k)} & \cdots & A_{(m,m)} \end{bmatrix}$$

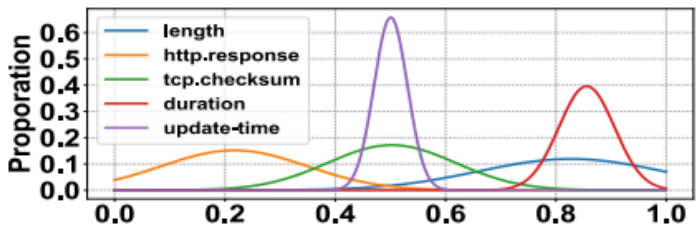
where $A_{i,i}$ is the intra-layer adjacency matrix of layer i and $A_{i,j}$ where $i \neq j$ is the cross-layer adjacency matrix



Statistical Disentanglement



(b) Normalized feature distribution of MITM attacks

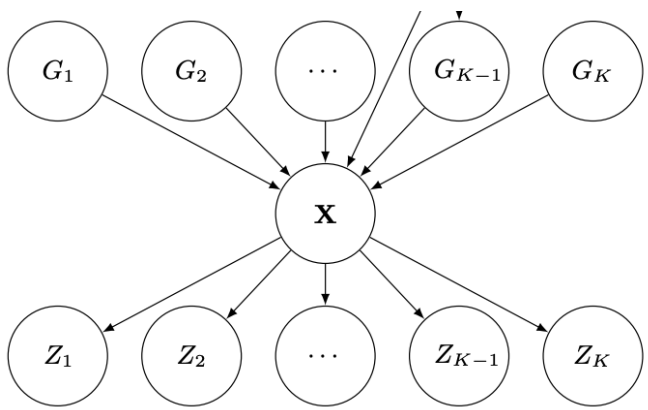


(c) Normalized feature distributions of DDoS attacks

- Disentangled representation learning is used to disentangle features, which should improve performance on downstream tasks such as classification.

Learning Disentangled Representations

- The goal in **disentangled representation learning** is to learn a compact representation $r(\mathbf{x})$ of some high-dimensional feature vector $\mathbf{x}$, where $r(\mathbf{x})$ captures all of the *factors of variation* in $\mathbf{x}$.
- **Hypothesis:**
  - $\mathbf{x}$ is a realization of some high-dimensional random variable $\mathbf{X} \in \mathbb{R}^N$ which is generated by $K << N$ independent causal mechanisms $\mathbf{G} = (G_1, \ldots, G_K)$, which are latent (hidden).
  - Our goal: we want to learn a representation $\mathbf{z} = r(\mathbf{x})$ that captures only the factors of variation in $\mathbf{x}$.
  - Downstream tasks, such as classification and prediction should be much easier given $r(\mathbf{x})$ rather than $\mathbf{x}$.



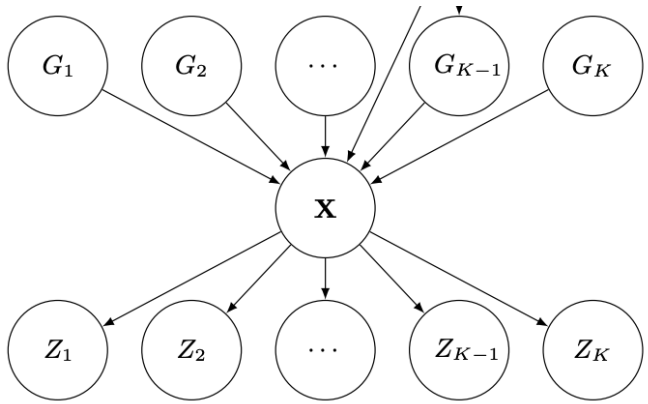
Statistical Disentanglement

- Factors of variation: dog color, breed, age, background scenery..

Learning Disentangled Representations

- Assume probabilistic model $P(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})$
- Typically learn the generative model $P(\mathbf{x} \mid \mathbf{z})$ and the posterior $P(\mathbf{z} \mid \mathbf{x})$ using **variational auto-encoders**.
- Example of a causal disentangled representation:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid G_i) \tag{1}$$



Statistical Heuristic Disentanglement in 3D-IDS

- Let $\mathcal{F} \in \mathbb{R}^{K \times N}$ denote the matrix of normalized features for a given edge.
- Define a weight matrix $\mathbf{w} \in \mathbb{R}^{K \times N}$.
- The disentangelement problem is then formulated as the following constrained optimization problem

$$\text{maximize } \mathbf{w}_N \mathcal{F}_N - \mathbf{w}_1 \mathcal{F}_1 - \sum_{i=2}^{N-1} |2\mathbf{w}_i \mathcal{F}_i - \mathbf{w}_{i+1} \mathcal{F}_{i+1} - \mathbf{w}_{i-1} \mathcal{F}_{i-1}|$$

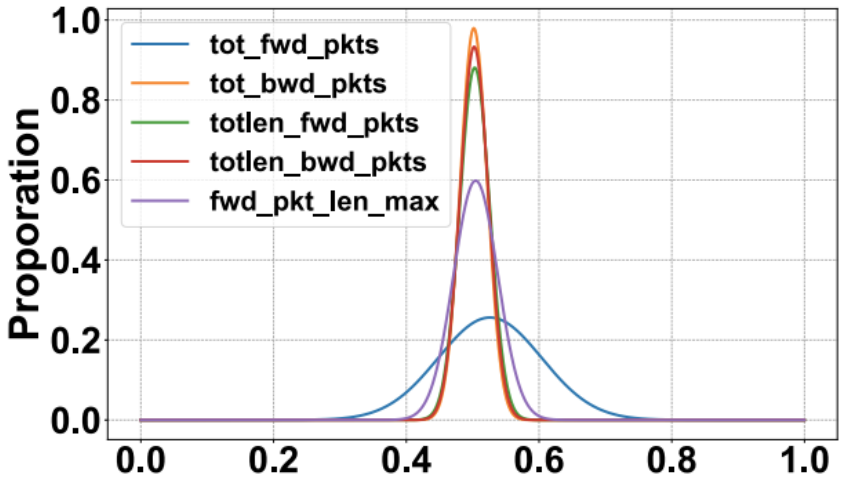
$$\text{subject to } W_{min} \leq \mathbf{w}_i \leq W_{max} \qquad i = 1, 2, \ldots, N$$

$$\sum_{i=1}^{N} \mathbf{w}_i \mathcal{F}_i \leq B$$

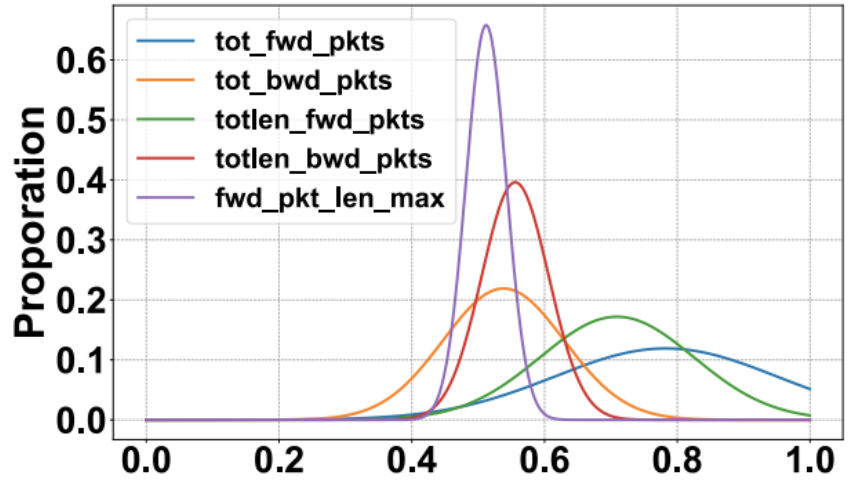
$$\mathbf{w}_i \mathcal{F}_i \leq \mathbf{w}_{i+1} \mathcal{F}_{i+1} \qquad i = 1, 2, \ldots, N-1 \qquad \text{(A)}$$

where W_{min}, W_{max}, B are constants and (A) is an ordering constraint and $\mathbf{w}_i \mathcal{F}_i$ is the disentangled representation of feature i. and $\mathbf{h}_{i,j}$ is the vector of disentangled features for edge (i, j).

Statistical Heuristic Disentanglement in 3D-IDS



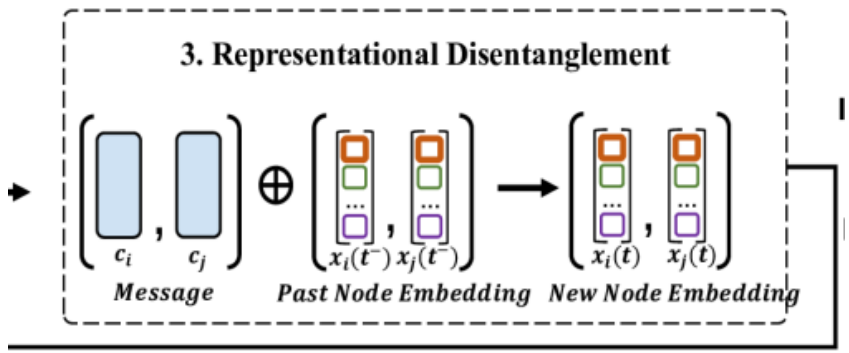
(a) Origin



(b) First Distangled

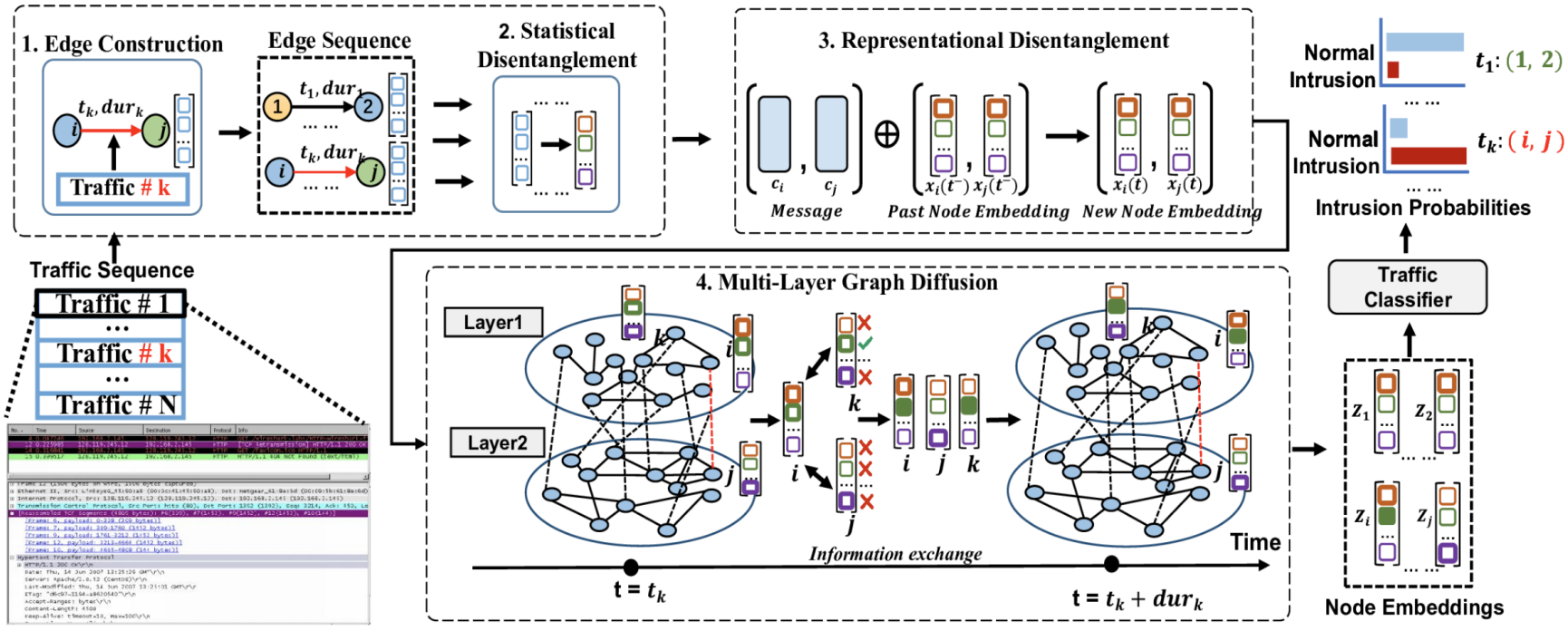
- The effect of solving the constrained optimization problem is that the **feature distributions are shifted** in a certain order.
- The intuition is that this shift should minimize the mutual information (overlap) between each two features.
- I.e, a rather heuristic form of feature disentanglement.

Learning Node Embeddings



- The embeddings are trained in a supervised manner using recurrent neural networks.

Learning Node Embeddings



Learning Node Embeddings

- **The embedding of node i at time t is denoted by $\mathbf{m}_i(t)$.** It is defined by an GRU encoder neural network called "Mem" which takes as input:

$$\mathbf{m}_i(t) = Mem(c_i(t), \mathbf{m}_i(t^-)) \qquad \mathbf{m}_i(0) = 0 \ \forall i$$

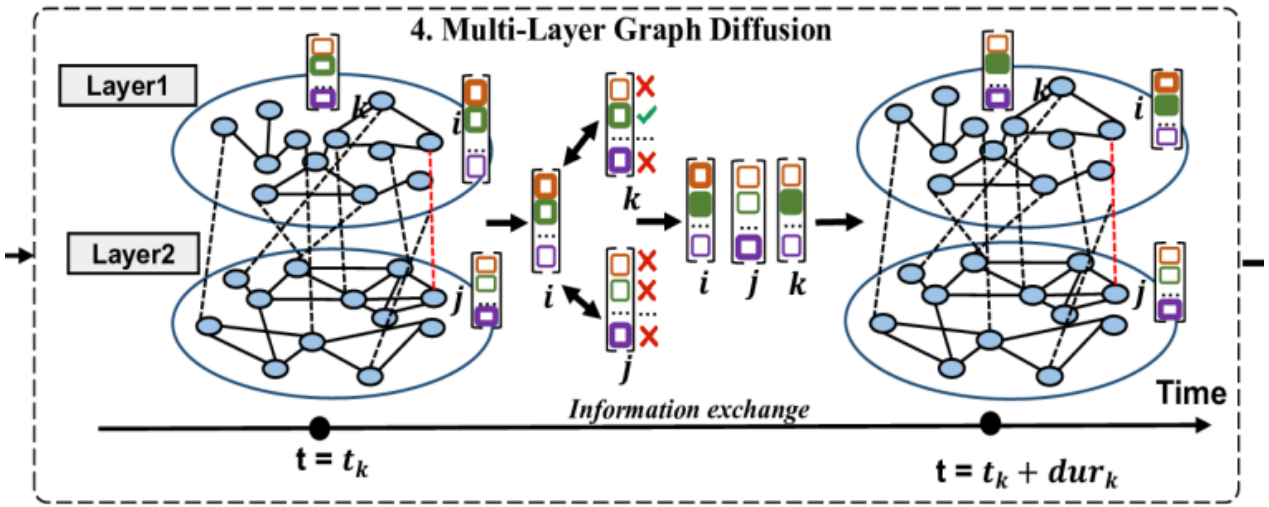
where $c_i(t)$ encodes the edge features related to node i at time t:

$$\mathbf{c}_i(t) = Msg(\mathbf{m}_i(t^-), \mathbf{m}_j(t^-), t, \Delta t, l_I, l_j, \mathbf{h}_{i,j})$$

where $\mathbf{h}_{i,j}$ is the disentangled edge representation, Δt is the edge duration, l_i, l_j are the node levels, and $m_i(t^-)$ and $m_j(t^-)$ are the node-embeddings from the previous time-step.

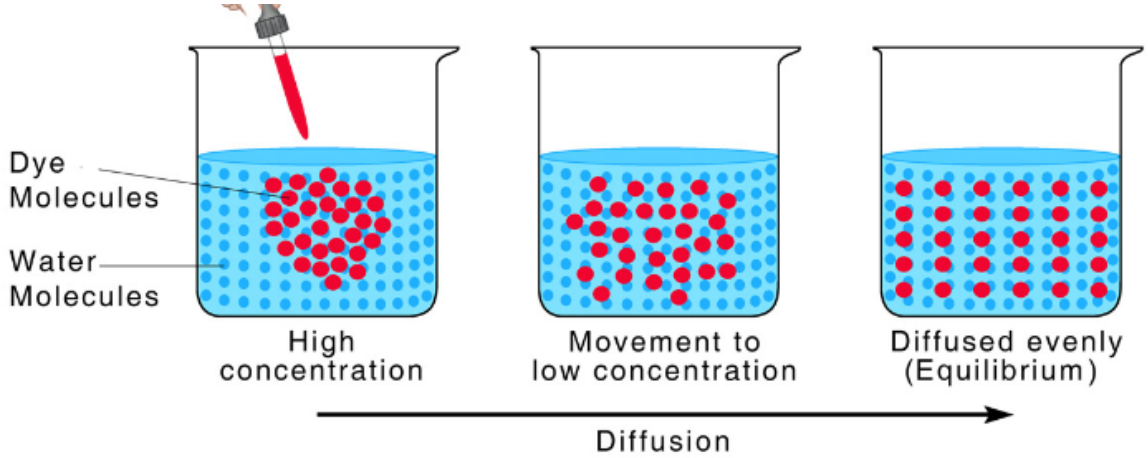
- Here "Msg" is a recurrent neural network.

Graph Diffusion



- Graph diffusion is used to capture how flow features evolve from the time the flow is started at time t to the time it ends $t + \Delta t$.

Diffusion Processes



- Diffusion describes the movement of some quantity from regions of high concentration to lower concentration over time.
- e.g., Heat on an iron rod diffuses from warmer parts of the rod to colder parts.
- This process can be described by the heat equation (a Partial Differential Equation):

$$\frac{\partial}{\partial t}x(u,t) = \frac{\partial^2}{\partial u^2}x(u,t) \tag{2}$$

where $x(u,t)$ is the temperature at position u at time t.

Perona-Malik Diffusion

- **Perona-Malik** (also known as anisotropic) diffusion is a technique to reduce noise in images.
- It is defined by a Partial Differential Equation (PDE):

$$\frac{\partial \mathrm{Img}(x, y, t)}{\partial t} = \mathrm{div}(c(x, y, t)\nabla \mathrm{Img}) \tag{3}$$

where x, y are the coordinates of the image, t is time, div is the divergence operator, ∇ is the gradient, and c controls the diffusion rate.

Perona-Malik Diffusion

- When $t = 0$ the function that satisfies the PDE is equal to the original image.
- As t increases, the image becomes blurrier while still maintaining important characteristics/edges in the image, which has the effect of removing noise.
- That is, Perona-Malik diffusion can be used to **remove noise from images without blurring edges**.

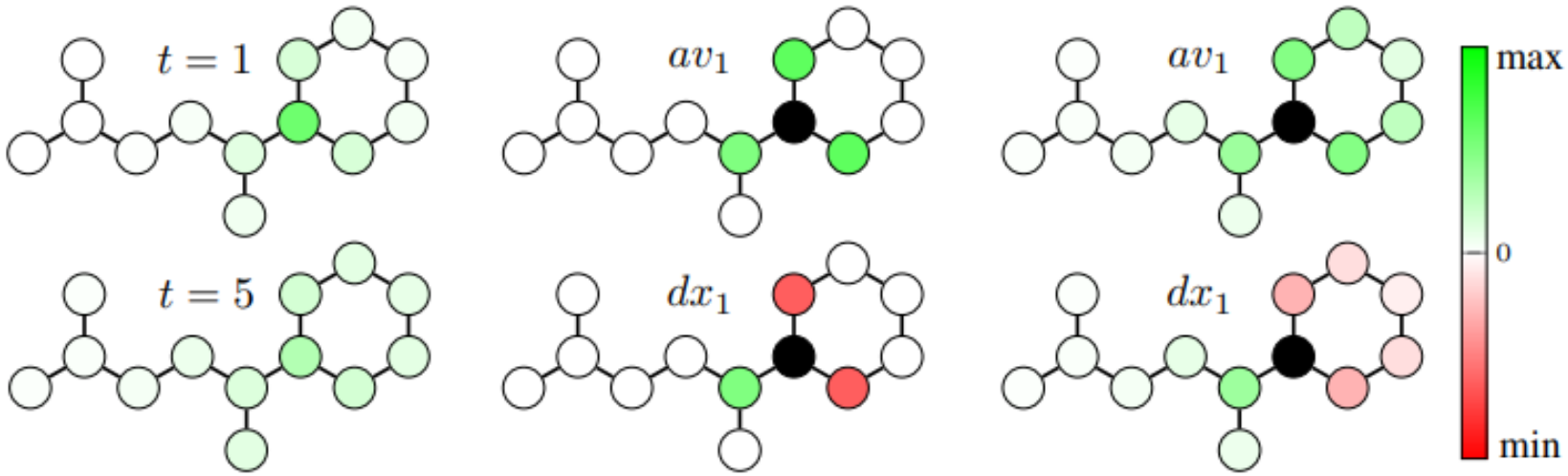
Perona-Malik Diffusion

- The diffusion rate function c works as an *"edge stopper"*.

$$\frac{\partial \mathrm{Img}(x, y, t)}{\partial t} = \mathrm{div}(c(x, y, t)\nabla \mathrm{Img}) \tag{4}$$

- It satisfies $c(x) \to 0$ as $x \to \infty$
- Which means that the diffusion (the blurring) is stopped at sharp edges of the image
- When x is not an edge, $c(x) > 0$, which means that the diffusion causes blurring.

Perona-Malik Diffusion Applied to Graphs



(a) Diffusion for different times t (b) Anisotropic filters: av_1 and dx_1 (c) Anisotropic diffusion kernels

- Since a graph can be defined by its adjacency matrix and feature matrix, similar diffusion equations that are applied in image processing can be used on graphs.
- For example, if there is a high traffic load on node i in a computer network at time t, we can use graph diffusion and a PDE to describe how the network load will spread through the neighbors of i until time $t + \Delta t$

Graph Diffusion in 3D-IDS

- Given the node embeddings $\mathbf{m}_1(t), \mathbf{m}_2(t), \ldots$ and the edges $\mathcal{E}^t$, the new graph $\mathcal{G}^t$ is fused with the previous graphs $\mathcal{G}^{t-1}, \mathcal{G}^{t-2}, \ldots$ through a **graph diffusion method**, which fuses the topological information of the evolving graph.

- They utilize the Perona-Malik diffusion form image processing, which is defined by the following partial differential equation:

$$\frac{\partial x(u,t)}{\partial t} = \mathrm{div}[g(|\nabla x(u,t)|)\nabla_x(u,t)]$$

  with initial condition $x(u,0) = c$.

- Here $x(u,t)$ represents the node embedding of a given node at time t after the update u.

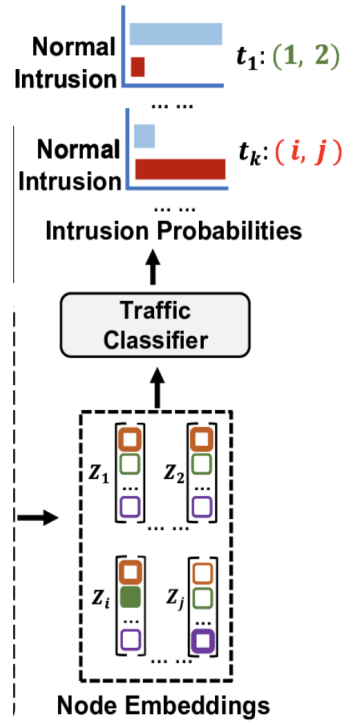
Graph Diffusion 3D-IDS

- Applying the gradient and divergence operators to the graph, one can obtain from spectral graph theory that the PDE can be expressed as:

$$\partial \mathbf{X}_t = -\mathbf{M}^T \sigma(\mathbf{M}\mathbf{X}\mathbf{K}^T)\mathbf{S}(\mathbf{M}\mathbf{X}\mathbf{K}^T)\mathbf{K} \tag{5}$$

  where $\mathbf{X}$ is the matrix with the node embeddings, $\mathbf{K}$ is a transformation matrix, $\mathbf{S}$ is a matrix with coefficients computed by a neural network, and $\sigma(x) = exp(-|x|)$.
- A solution to the above PDE is **approximated using the Runge-Kutta numerical methods**
- The obtained solution of the PDE then represents the node embeddings at time $t + \Delta t$, i.e., $\mathbf{X}_{t+\Delta t}$.

Intrusion Detection and Attack Classification



- Finally, after computing the updated node embeddings $\mathbf{X}_{t+\Delta t}$ they are fed into two feed-forward neural networks
- The first neural network is a binary classifier that predicts whether an intrusion occurs
- The second neural network predicts the attack type
- The loss function is defined as

$$\mathcal{L}_{int} = -\sum_{i=1}^{m} \log(1 - p_{nor,i}) + \log(p_{att,i}) + \sum_{i=1}^{m}\sum_{j=1}^{K} y_{i,k} \log(p_{i,k})$$

where m is the batch size, K is the number of attack types and $p_{nor,i}$ is the predicted probability of no intrusion.

Regularization and Representational Disentanglement

- Two regularization terms are added to the loss function during the supervised training:
  1. A term to incentivize disentangled node embeddings:

$$\mathcal{L}_{Dis} = \frac{1}{2}||X(t)X(t^{-})^{T} - I||_{F}^{2} \tag{6}$$

     I.e the norm of the matrix product of the node representations, which should ensure that the node representations are close to orthogonal (i.e that their dot products are zero).
  2. A term to incentivize smooth updates:

$$\mathcal{L}_{smooth} = \sum_{t=0}^{T} ||X_{t+\Delta t} - X_t||_2 \tag{7}$$

- The final loss is thus:

$$\mathcal{L} = \mathcal{L}_{int} + \alpha\mathcal{L}_{Smooth} + \beta\mathcal{L}_{Dis} \tag{8}$$

  where α and β are constants.

Comparison with State-of-the-art in terms of Binary Intrusion Detection

Table 1: Comparisons of binary classification on five datasets. The results with ‡ are directly copied from [8].

Methods	CIC-TON-IoT		CIC-BoT-IoT		EdgeIIoT		NF-UNSW-NB15-v2		NF-CSE-CIC-IDS2018-v2	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
TGN [61]	89.90±1.66	82.09±1.36	96.84±0.44	94.41±0.81	94.99±0.61	89.50±2.04	93.55±0.23	88.01±1.97	95.11±0.46	91.30±0.76
EULER [36]	89.73±1.13	80.48±2.46	96.00±0.29	91.47±1.36	92.89±0.32	90.64±1.80	92.76±0.86	86.97±1.11	95.87±0.51	90.76±0.64
AnomRank [78]	76.51±0.98	77.41±1.64	84.84±0.49	82.50±0.59	78.37±0.43	81.36±0.41	90.54±2.40	79.63±0.12	90.08±0.82	83.76±0.35
DynAnom [27]	79.23±1.81	75.22±0.92	83.25±0.62	79.04±0.84	81.56±0.94	83.94±0.36	89.11±1.48	85.25±0.64	91.21±0.95	88.79±0.54
Anomal-E [8]	-	-	-	-	-	-	91.89‡	-	94.51‡	-
GAT [74]	86.30±1.16	74.66±1.37	94.56±0.75	93.09±2.83	93.30±0.14	88.30±1.56	92.20±1.60	89.91±0.62	96.08±0.24	90.56±0.34
E-GraphSAGE [46]	89.46±1.25	79.56±1.63	93.74±0.76	90.53±1.90	92.10±1.46	89.10±0.64	94.10±0.33	90.39±0.26	95.71±0.35	90.22±0.48
DMGI [55]	88.83±0.48	79.13±2.11	96.07±1.89	92.65±1.57	93.83±1.67	86.03±2.45	93.11±0.98	88.51±1.00	93.87±0.84	87.56±0.55
SSDCM [50]	89.23±0.87	80.84±2.32	97.11±0.63	94.82±0.96	94.72±1.59	86.69±0.76	93.30±0.25	89.22±1.94	94.96±0.52	88.61±0.38
MLP [60]	80.74±0.43	61.80±1.48	93.01±0.60	87.90±0.54	88.78±0.44	86.00±1.49	93.12±0.64	89.92±0.55	94.59±0.94	90.42±0.89
MStream [5]	73.90±1.13	70.22±1.61	78.48±0.19	74.04±1.66	82.47±1.67	77.89±0.58	89.47±1.13	84.38±1.01	88.34±0.45	83.66±1.79
LUCID [18]	83.62±1.69	72.31±1.14	94.36±0.41	89.46±0.72	88.94±1.73	85.23±0.94	92.77±1.39	88.32±0.91	95.84±1.46	90.75±0.79
Ours (3D-IDS)	**91.57**±0.40	**84.06**±1.01	**98.24**±0.32	**96.32**±0.25	**96.83**±0.36	**92.34**±1.10	**95.45**±0.67	**91.55**±1.03	**96.34**±0.21	**93.23**±1.50

E-GraphSAGE TGN AdaBoost† ML† Ours (3D-IDS)

- State-of-the-art results on all metrics on five datasets (!).

Comparison with State-of-the-art in terms of Attack Type Classification

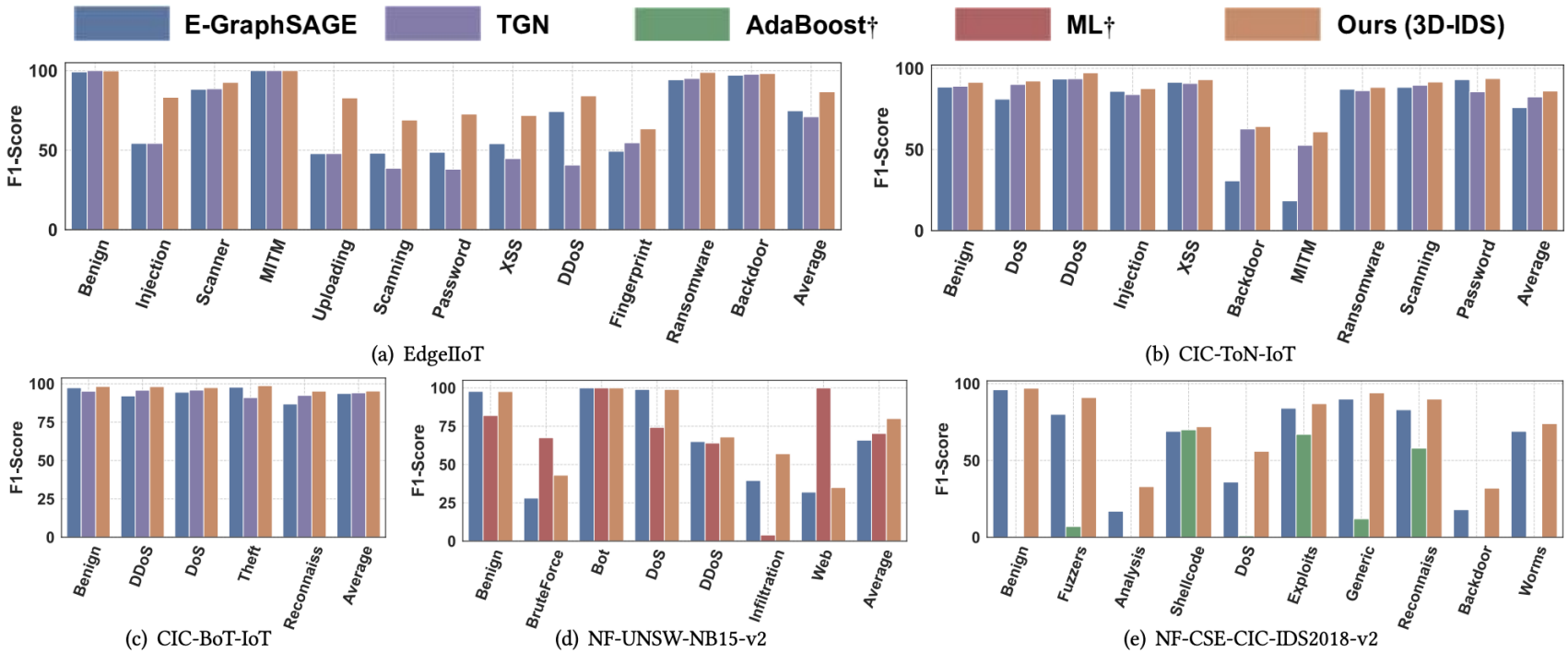


Figure 4: Comparisons of multi-classification. Here † indicates that the results are directly copied from the previous works.

- State-of-the-art results on all metrics except 2(!).

Ablation Study

Variants	P	R	F1	AUC
w/o SD	92.70±0.46	90.71±0.89	91.69±0.33	86.87±0.54
w/o RD	91.06±0.57	87.32±0.67	89.15±0.42	83.57±1.33
w/o MLGRAND	88.76±0.54	84.43±0.26	86.54±0.71	79.32±0.30
3D-IDS(ours)	**97.78**±0.32	**98.06**±0.43	**97.92**±0.26	**96.04**±0.25

- All modules of the deep learning system improves performance.
- **Graph diffusion** improves performance the most

Conclusions

- This paper presents a novel IDS based on deep learning called **3D-IDS**
- 3D-IDS uses two levels of feature disentanglement and graph diffusion in combination with deep neural networks.
- **3D-IDS achieves state-of-the-art results on five benchmarks.**

Discussions

- **Impressive results**, what are the drawbacks?
  - Could be overfitting on these five benchmarks, system is a bit overengineered to beat STOTA.
- Novel use of diffusion processes, can it be used for other tasks in cyber security?
- **Problems**:
  - Kitchen sink of heuristics.
  - Poor description of related work.
  - Usage of representation disentanglement is inconsistent with theory and literature.
  - Definition of the SMT problem for statistical disentanglement is incomplete.
- Questions?