

# Поиск аномалий на графах

Зелинский Никита

# План лекции

01. Задача Outlier detection

02. Графовые автоэнкодеры

03. SOTA 2023

04. Зачем уметь решать эту задачу

# Задача Outlier detection

01

# 1. Не так очевидно



→ Это задача Supervised / Semi-supervised / Unsupervised?

# 1. Не так очевидно



- Это задача Supervised / Semi-supervised / Unsupervised?
- Есть ли предположения о распределении выбросов?

# 1. Не так очевидно



- Это задача Supervised / Semi-supervised / Unsupervised?
- Есть ли предположения о распределении выбросов?
- Есть ли предположения о распределении нормальных наблюдений?

# 1. Не так очевидно



- Это задача Supervised / Semi-supervised / Unsupervised?
- Есть ли предположения о распределении выбросов?
- Есть ли предположения о распределении нормальных наблюдений?
- Есть какие-то априорные представления по доле выбросов?

# 1. Не так очевидно



- Это задача Supervised / Semi-supervised / Unsupervised?
- Есть ли предположения о распределении выбросов?
- Есть ли предположения о распределении нормальных наблюдений?
- Есть какие-то априорные представления по доле выбросов?
- Разве эта задача принципиально отличается от кластеризации?

## 2. Классические предположения



- Нормальные наблюдения лежат “кучно” – близко друг к другу, а выбросы – “некучно” (2002)<sup>1</sup>

## 2. Классические предположения



- Нормальные наблюдения лежат “кучно” – близко друг к другу, а выбросы – “некучно” (2002)<sup>1</sup>
- Выбросы распределены равномерно

неявное предположение многих алгоритмов – теряем априорные знания о распределении выбросов (2002)<sup>1</sup>

## 2. Классические предположения



- Нормальные наблюдения лежат “кучно” – близко друг к другу, а выбросы – “некучно” (2002)<sup>1</sup>
- Выбросы распределены равномерно

неявное предположение многих алгоритмов – теряем априорные знания о распределении выбросов (2002)<sup>1</sup>

- Обычно понимаем задачу semi-supervised или unsupervised (именно для табличных данных! тк насэмплировать репрезентативный train для шума очень сложно)

## 2. Классические предположения

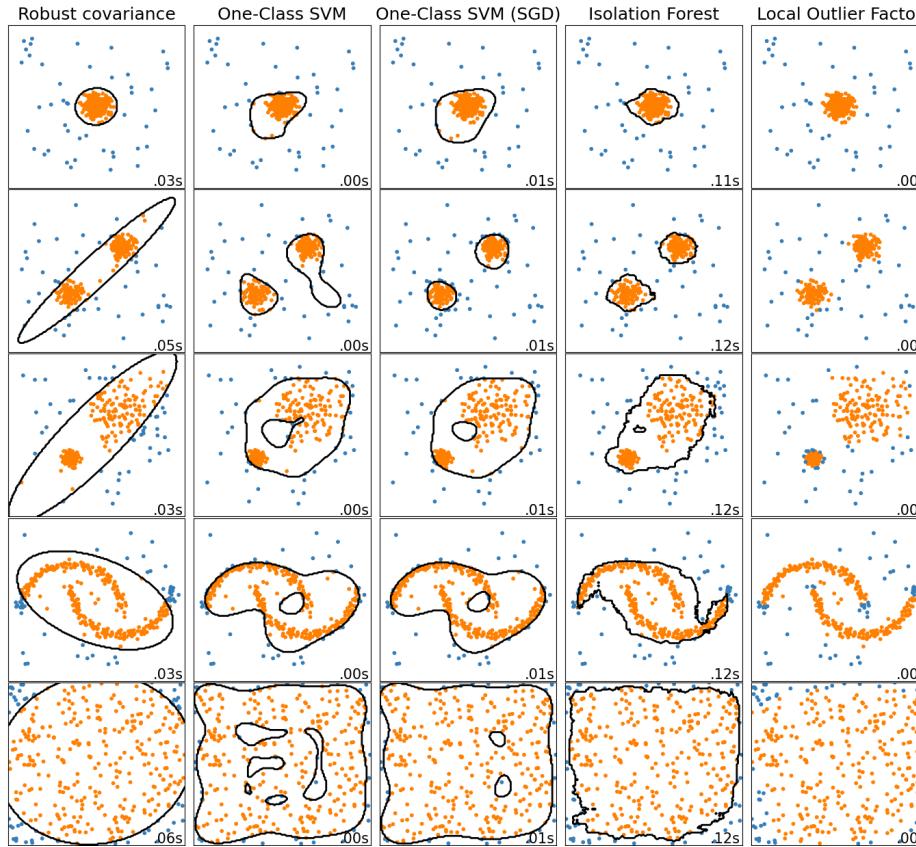


- Нормальные наблюдения лежат “кучно” – близко друг к другу, а выбросы – “некучно” (2002)<sup>1</sup>
- Выбросы распределены равномерно

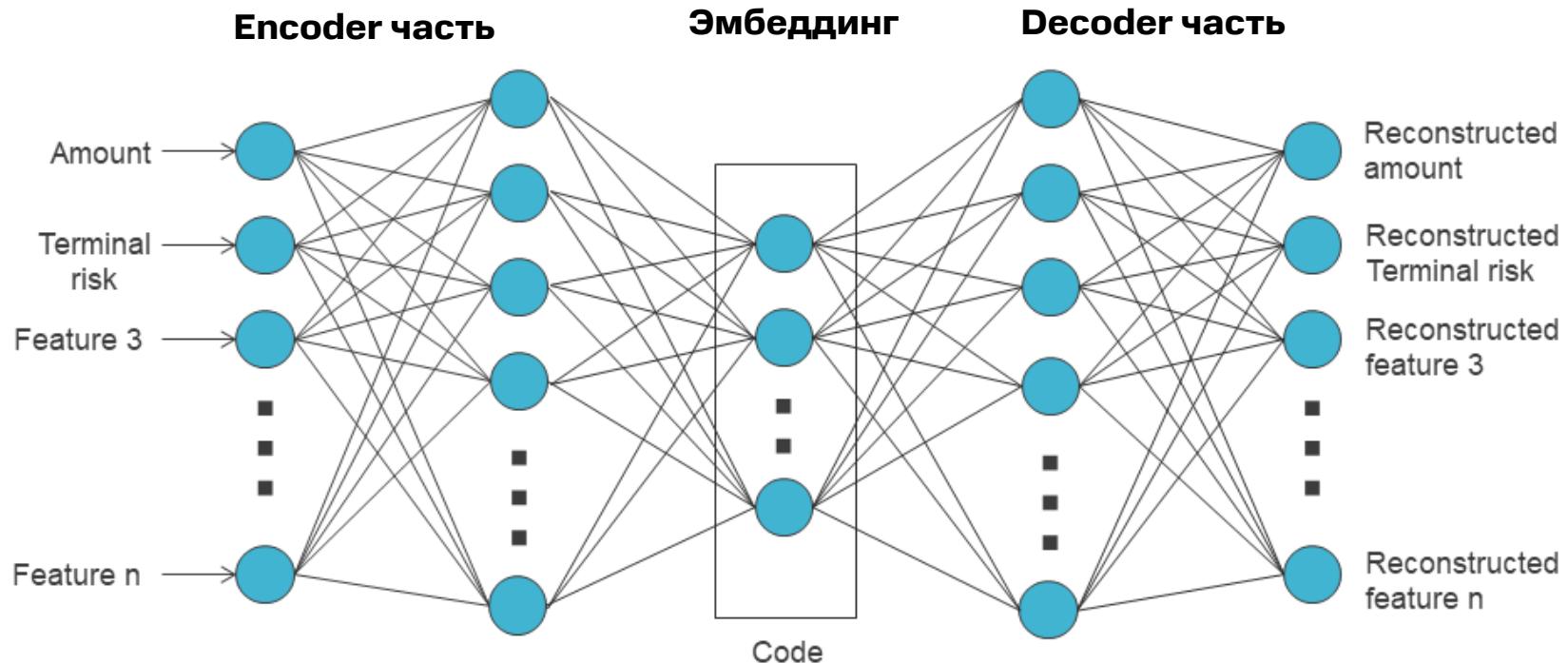
неявное предположение многих алгоритмов – теряем априорные знания о распределении выбросов (2002)<sup>1</sup>

- Обычно понимаем задачу semi-supervised или unsupervised (именно для табличных данных! тк насэмплировать репрезентативный train для шума очень сложно)
- Если не указано иное, решаем стационарную задачу (аномальность не меняется со временем) – хотя на практике это почти никогда не так

### 3. Табличные методы



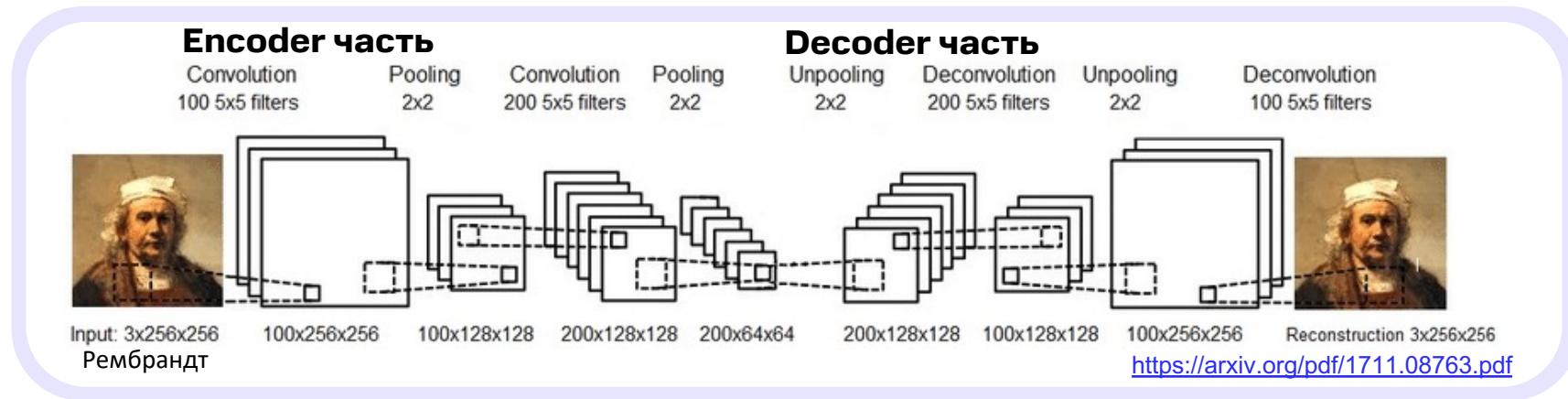
# 4. Автокодировщики



[https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter\\_7\\_DeepLearning/Autoencoders.html](https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_7_DeepLearning/Autoencoders.html)

# 6. Сверточные автокодировщики (САЕ)

Пример сверточного автокодировщика (2016)



A Deep Learning Approach to Video Anomaly Detection using Convolutional Autoencoders (7 ноября 2023)

<https://arxiv.org/abs/2311.04351>

Variational Convolutional Autoencoders for Anomaly Detection in Scanning Transmission Electron Microscopy (2022)

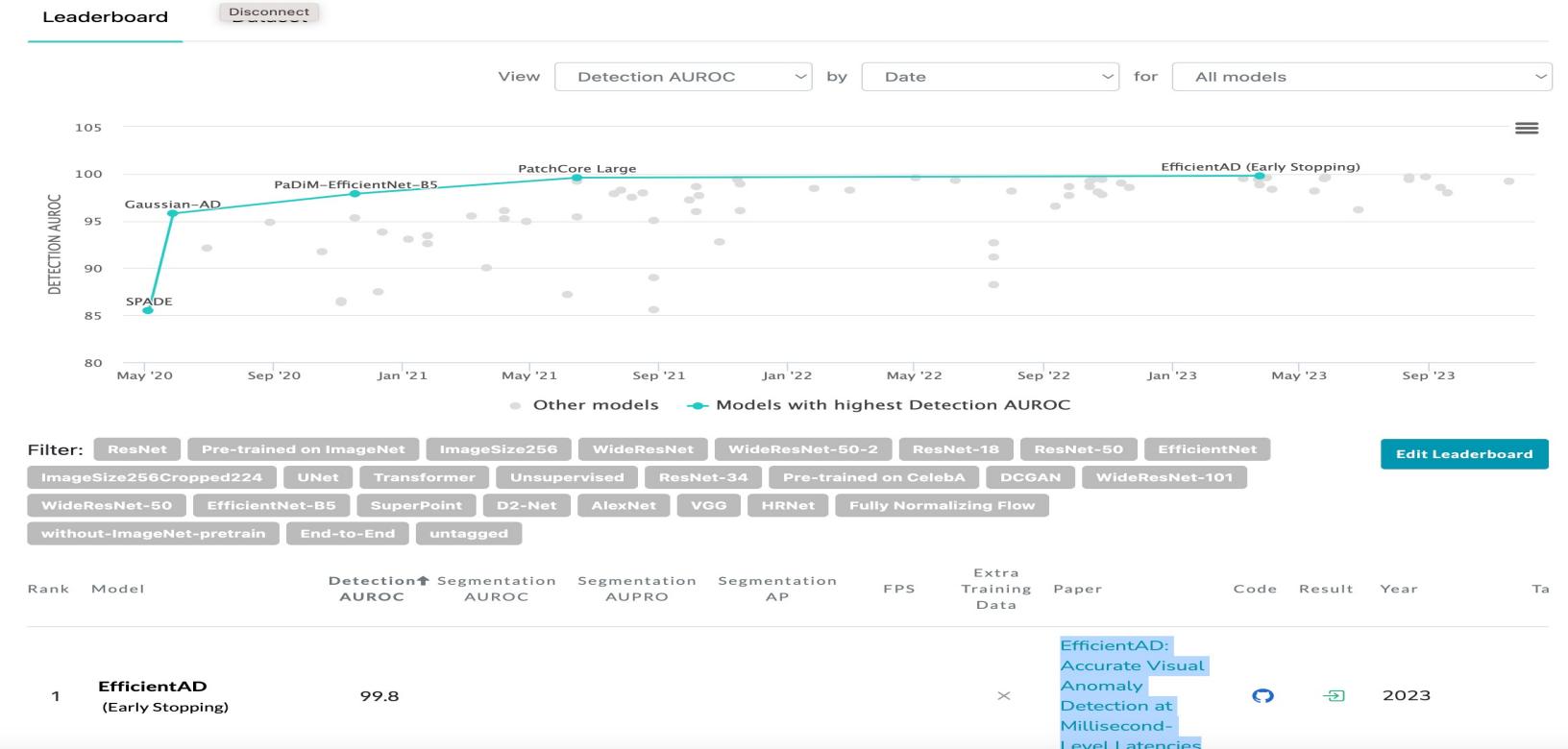
<https://onlinelibrary.wiley.com/doi/full/10.1002/smll.202205977>

Convolutional Autoencoder-Based Anomaly Detection for Photovoltaic Power Forecasting of Virtual Power Plants (2023)

<https://www.mdpi.com/1996-1073/16/14/5293>

# 7. SOTA в задаче Image AD

## Anomaly Detection on MVTec AD



# 8. SOTA-архитектура в Image AD

## Efficient AD

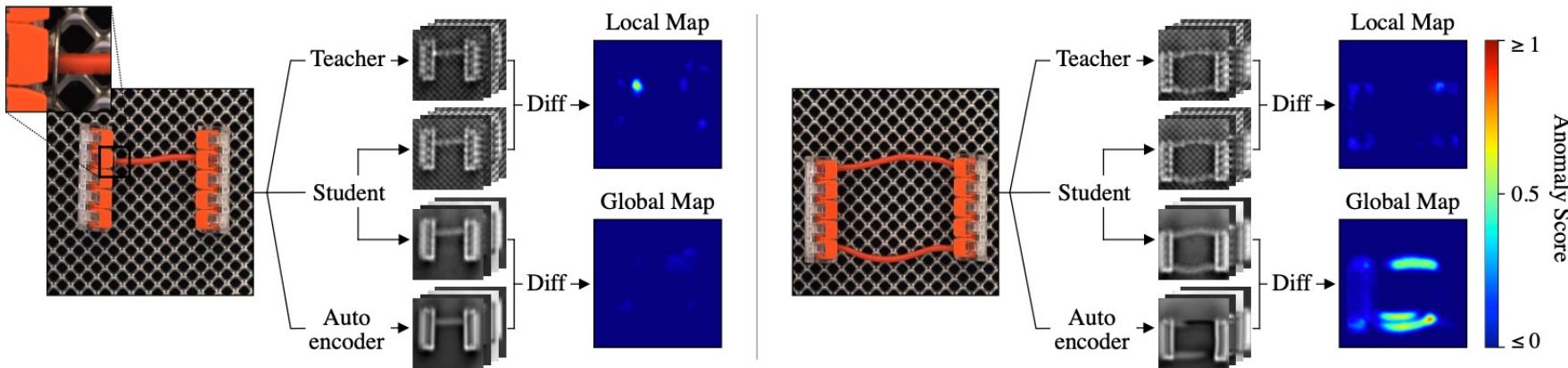
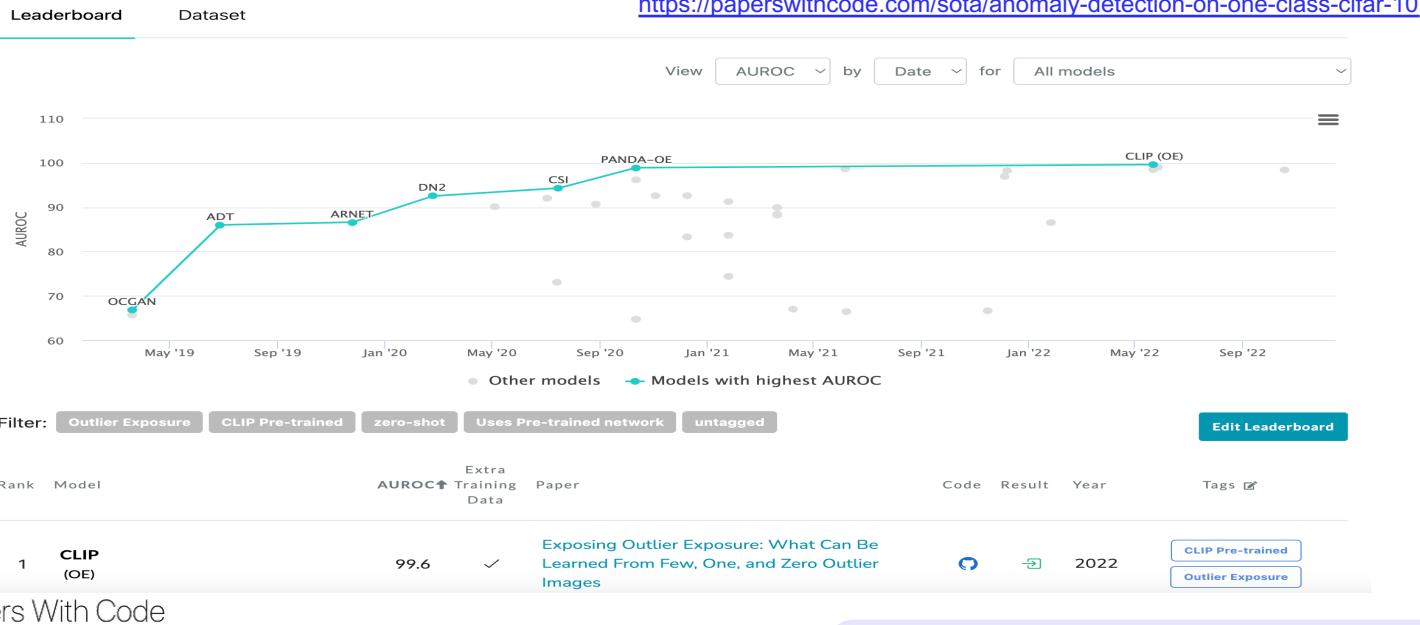


Figure 5. <...> The anomaly on the left is a foreign object in the form of a small metal washer at the end of the cable. It is visible in the local anomaly map because the outputs of the student and the teacher differ.

The logical anomaly on the right is the presence of a second cable. The autoencoder fails to reconstruct the two cables on the right in the feature space of the teacher. The student also predicts the output of the autoencoder in addition to that of the teacher. Because its receptive field is restricted to small patches of the image, it is not influenced by the presence of the additional red cable. This causes the outputs of the autoencoder and the student to differ. "Diff" refers to computing the element-wise squared difference between two collections of output feature maps and computing its average across feature maps. <...>

# 9. SOTA в задаче Image AD

## Anomaly Detection on One-class CIFAR-10



<https://github.com/openai/CLIP>

"Despite the overall strong performance of standard classifiers, we find that semi-supervised one-class methods are more robust to the choice of OE when only few OE examples are available."

<https://arxiv.org/pdf/2205.11474v2.pdf>

# 10. Отличия в картинках серьезные

Rethinking Assumptions in Deep Anomaly Detection (2020)

*“classifiers trained to discern between normal samples and just a few (64) random natural images are able to outperform the current state of the art in deep AD.”*

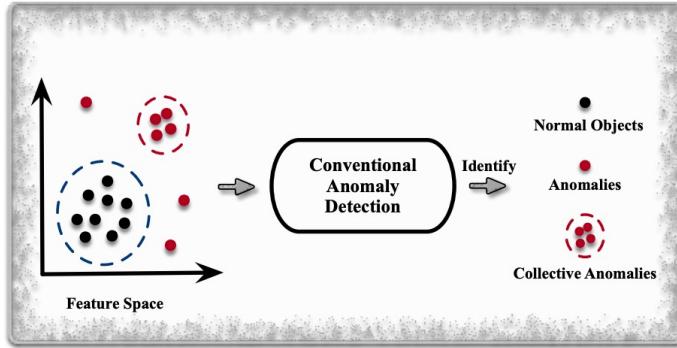
<https://arxiv.org/abs/2006.00339>

\* on ImageNet

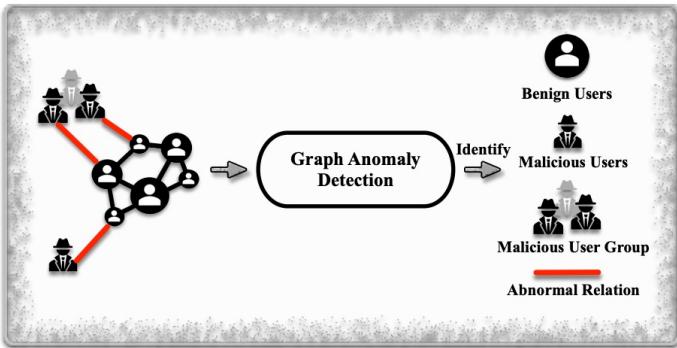
# Задача Outlier detection на графах

02

# 1. Что за задача?



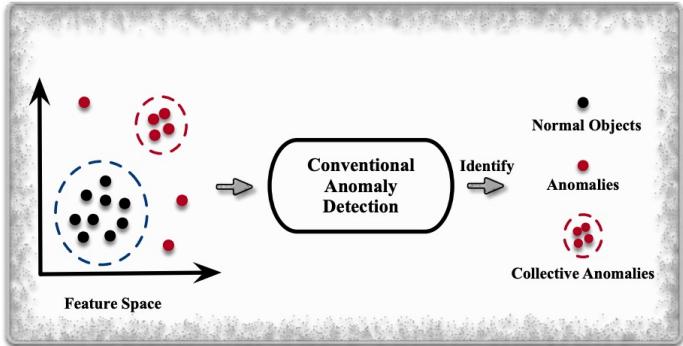
(a) Conventional Anomaly Detection



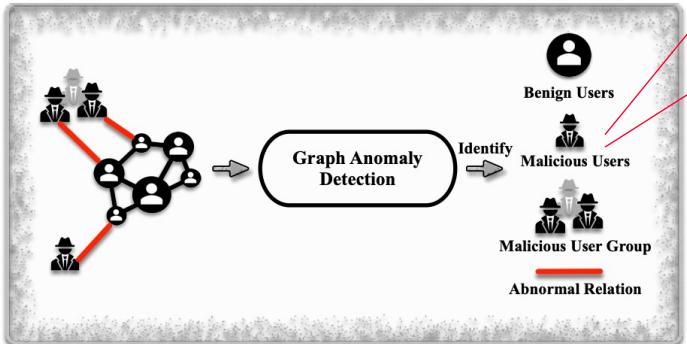
(b) Graph Anomaly Detection

A Comprehensive Survey on Graph Anomaly Detection  
with Deep Learning (2021)  
<https://arxiv.org/pdf/2106.07178.pdf>

# 1. Что за задача?



(a) Conventional Anomaly Detection



(b) Graph Anomaly Detection

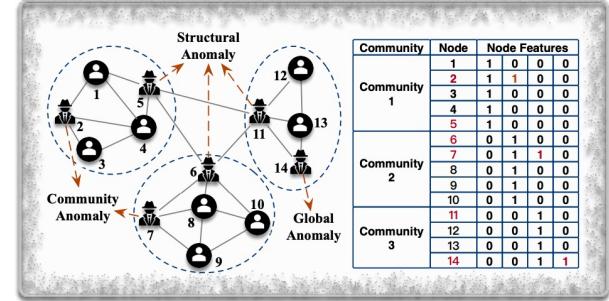


Fig. 3: Three Types of Anomalous Nodes: Structural Anomalies, Community Anomalies and Global Anomalies.

- **Global anomalies** only consider the node attributes. They are nodes that have attributes significantly different from all other nodes in the graph.
- **Structural anomalies** only consider the graph structural information. They are abnormal nodes that have different connection patterns (*e.g.*, connecting different communities, forming dense links with others).
- **Community anomalies** consider both node attributes and graph structural information. They are defined as nodes that have different attribute values compared to other nodes in the same community.

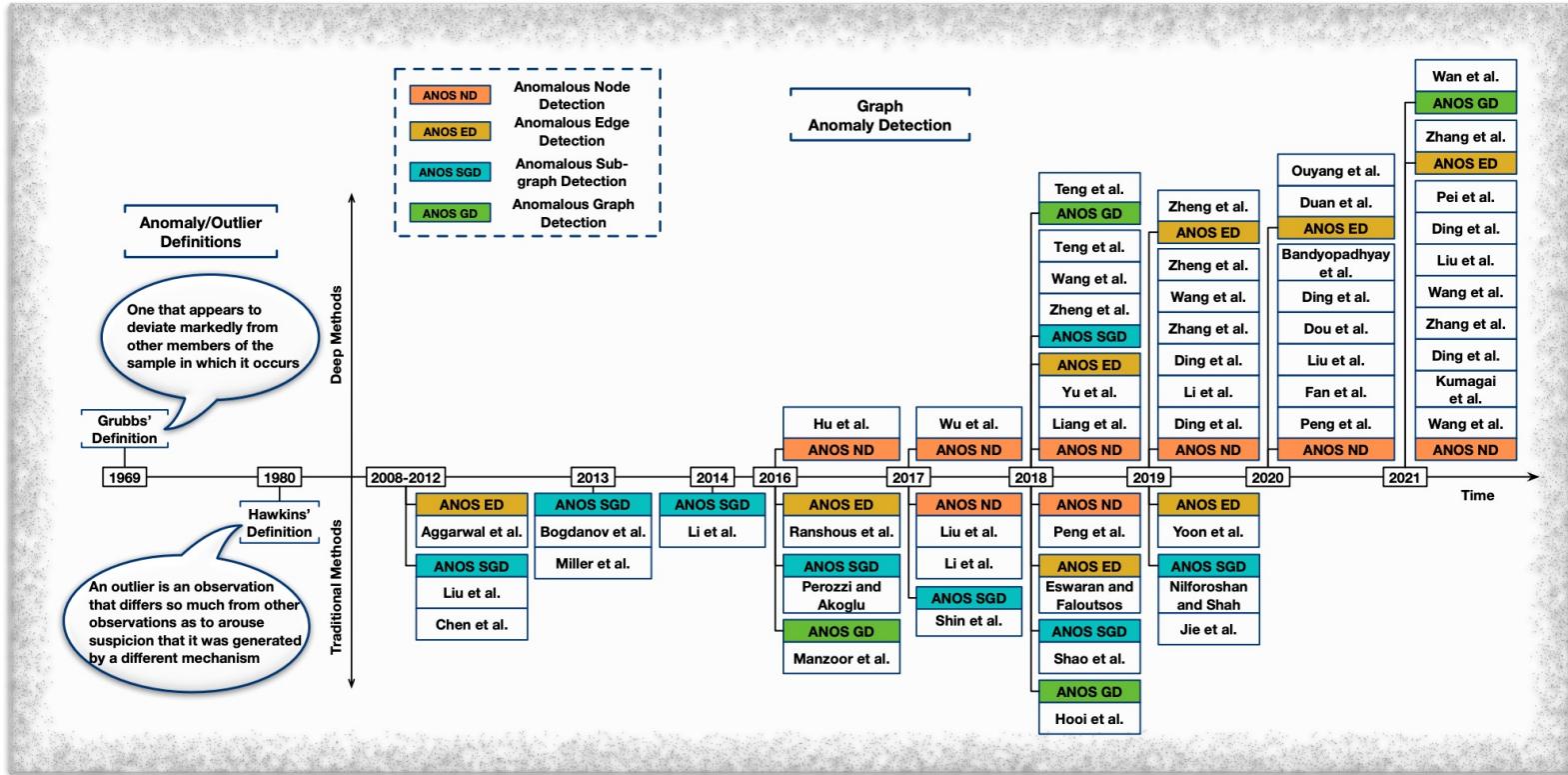


Fig. 2: A Timeline of Graph Anomaly Detection and Reviewed Techniques.

<https://arxiv.org/pdf/2106.07178.pdf>

# 1. И тут автоэнкодеры!

JOURNAL OF LATEX CLASS FILES, VOL. , NO. , AUGUST 2021

6

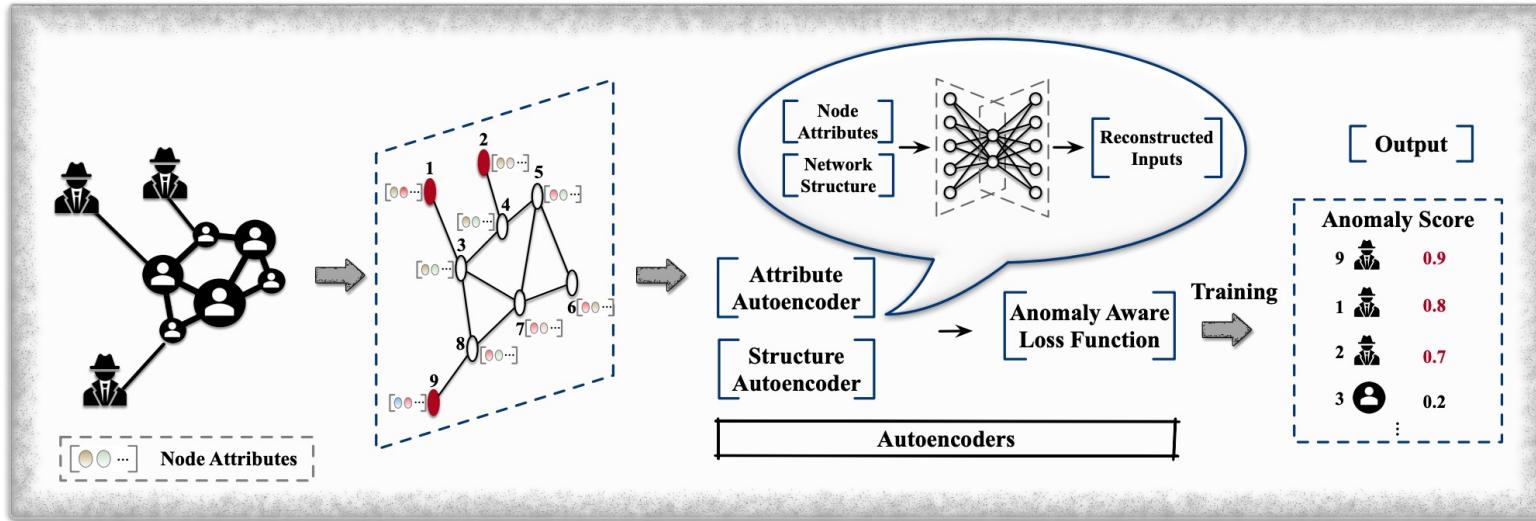
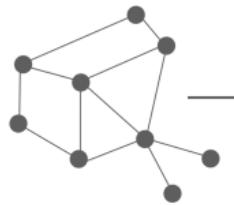


Fig. 4: ANOS ND on attributed graphs – Deep NN based approaches. As an example, the autoencoder is used to capture the graph structure and node attributes. With a specially-designed anomaly aware loss function, anomaly scores will be assigned to every node, and the top-k nodes are anomalies (e.g., nodes 9, 1, and 2 at top-3).

### 3. Графовые автокодировщики – GAE (2016)

**Encoder часть**

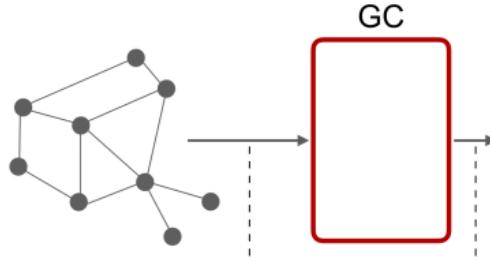


<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

### 3. Графовые автокодировщики – GAE (2016)

#### Encoder часть



$$\tilde{X} = GCN(A, X) = \text{ReLU}(\tilde{A}XW_0)$$

где  $\tilde{A} = D^{-1/2}AD^{-1/2}$

$D$  – диагональная матрица

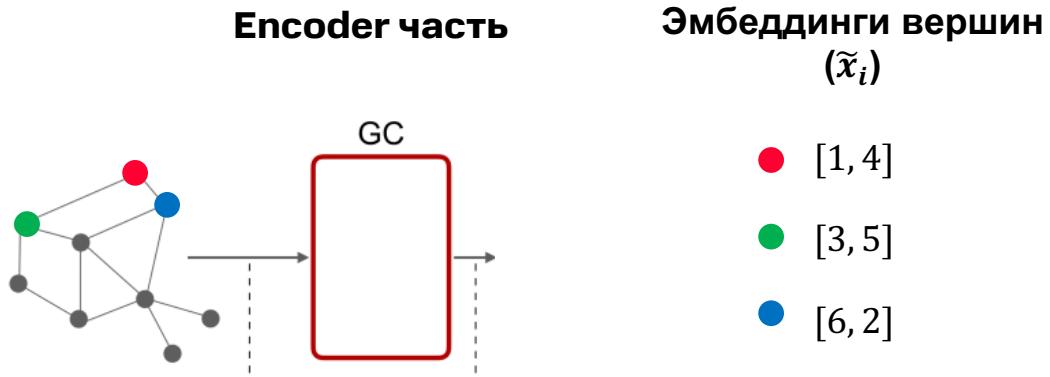
$A$  – исходная матрица смежности

$X$  – матрица фичей вершин

<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

### 3. Графовые автокодировщики – GAE (2016)



$$\tilde{X} = GCN(A, X) = \text{ReLU}(\tilde{A}XW_0)$$

где  $\tilde{A} = D^{-1/2}AD^{-1/2}$

$D$  – диагональная матрица

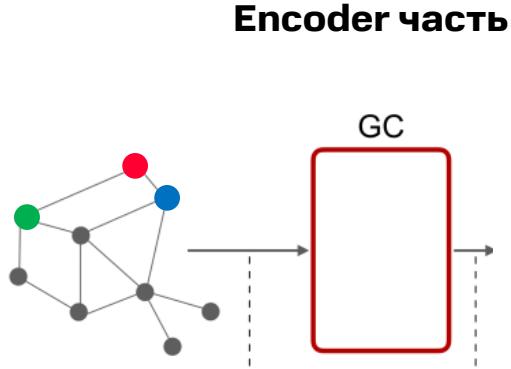
$A$  – исходная матрица смежности

$X$  – матрица фичей вершин

<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

### 3. Графовые автокодировщики – GAE (2016)



**Эмбеддинги вершин  
( $\tilde{x}_i$ )**

- [1, 4]
- [3, 5]
- [6, 2]

**Decoder часть**

$$a_{ij} = \sigma(\tilde{x}_i * \tilde{x}_j^T)$$

$$a_{\text{red}\text{green}} = \sigma([1, 4] * [3, 5]^T)$$

$$a_{\text{green}\text{blue}} = \sigma([3, 5] * [6, 2]^T)$$

$$a_{\text{red}\text{blue}} = \sigma([1, 4] * [6, 2]^T)$$

*"inner product decoder"*

$$\tilde{X} = GCN(A, X) = \text{ReLU}(\tilde{A}XW_0)$$

$$\text{где } \tilde{A} = D^{-1/2}AD^{-1/2}$$

$D$  – диагональная матрица

$A$  – исходная матрица смежности

$X$  – матрица фичей вершин

$$\min \|A - \sigma(\tilde{X} * \tilde{X}^T)\|$$

<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

# 4. Классический VAE (2013)

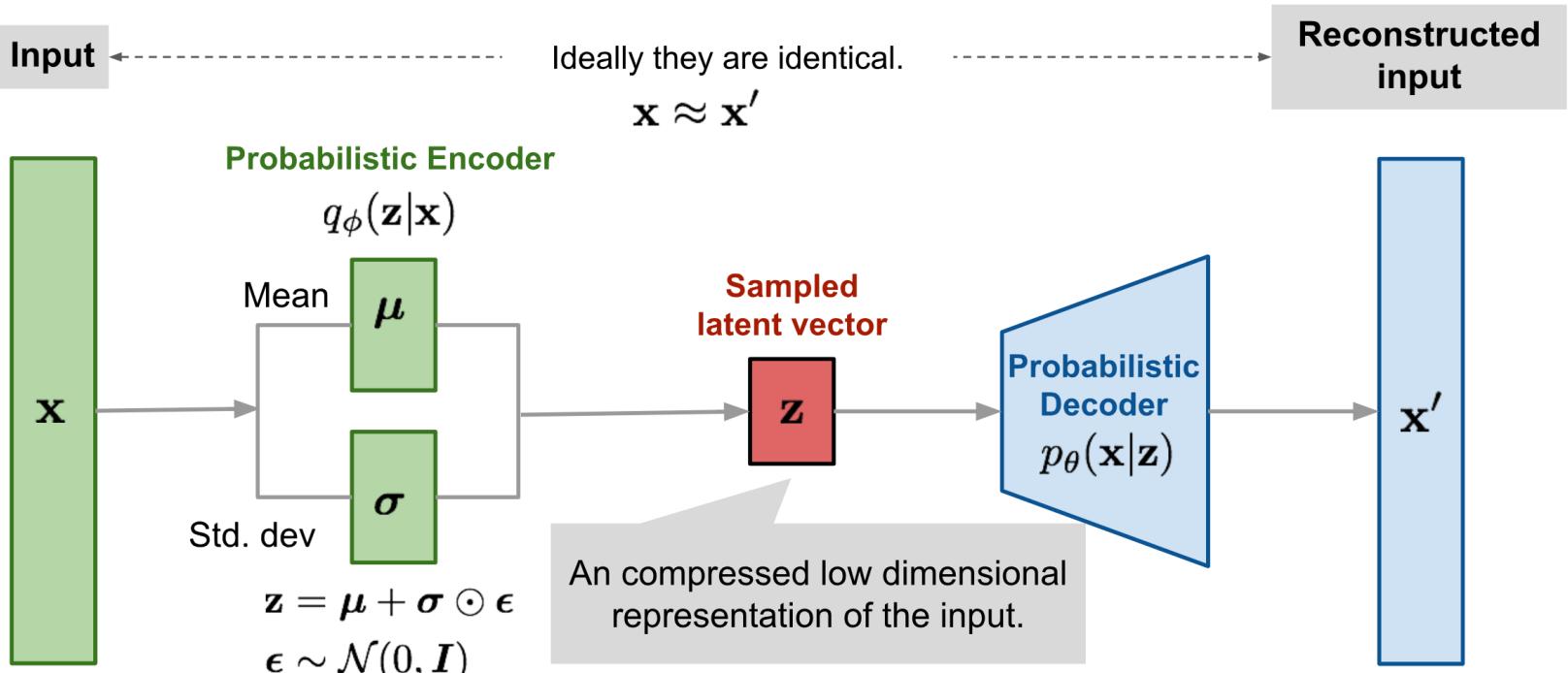


Illustration of variational autoencoder model with the multivariate Gaussian assumption.

<https://lilianweng.github.io/posts/2018-08-12-vae/>

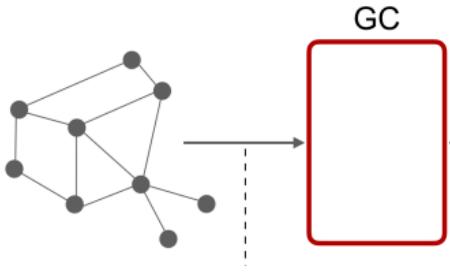
$$KL(N(\mu, \sigma^2) \mid\mid N(0, 1)) = \frac{1}{2}(-\log \sigma^2 + \mu^2 + \sigma^2 - 1)$$

Auto-Encoding Variational Bayes

<https://arxiv.org/abs/1312.6114>

# 4. Графовые автокодировщики – Vgae (2016)

Encoder часть



$$\tilde{X} = GCN(A, X) = \text{ReLU}(\tilde{A}XW_0)$$

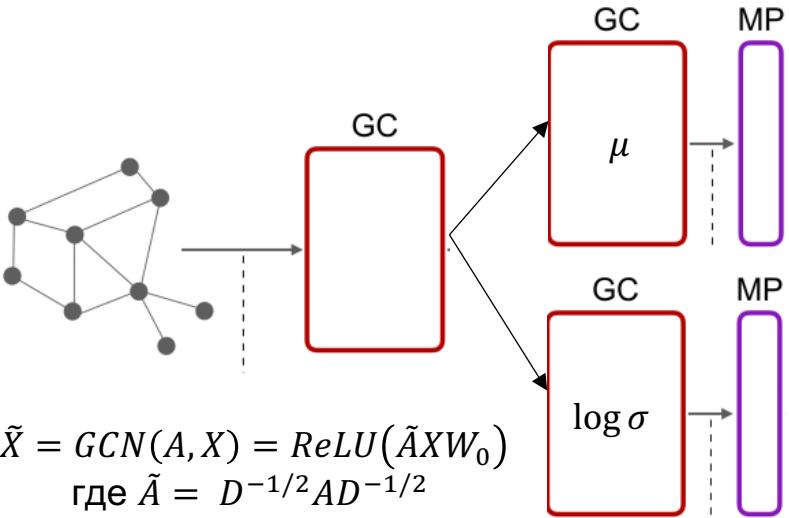
где  $\tilde{A} = D^{-1/2}AD^{-1/2}$

<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

# 4. Графовые автокодировщики – Vgae (2016)

Encoder часть



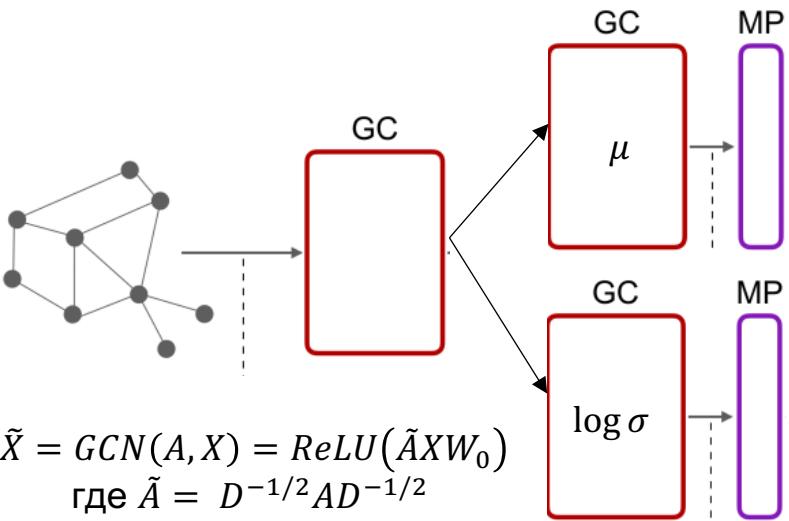
$\mu$  и  $\log \sigma$  -- это вектора!

<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

# 4. Графовые автокодировщики – Vgae (2016)

Encoder часть



Эмбеддинг

Reparametrization  
trick

$$Z = \mu + \sigma * \varepsilon$$
$$\varepsilon \sim N(0, 1)$$

```
def reparametrize(self, mu: Tensor, logstd: Tensor) -> Tensor:  
    if self.training:  
        return mu + torch.randn_like(logstd) * torch.exp(logstd)  
    else:  
        return mu
```

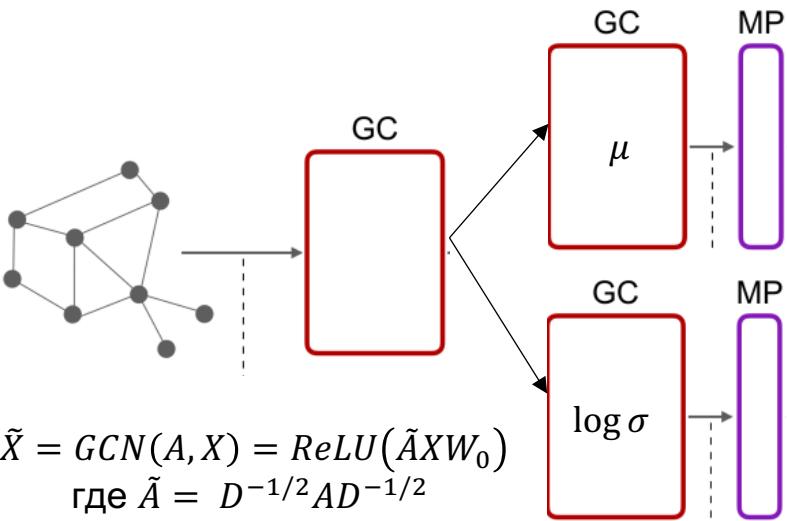
$\mu$  и  $\log \sigma$  -- это вектора!

<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

# 4. Графовые автокодировщики – Vgae (2016)

## Encoder часть



$\mu$  и  $\log \sigma$  -- это вектора!

## Эмбеддинг

Reparametrization trick  
 $Z = \mu + \sigma * \varepsilon$   
 $\varepsilon \sim N(0, 1)$

## Decoder часть

*"inner product decoder"*

$$\min \|A - \sigma(\tilde{X} * \tilde{X}^T)\|$$

+ регуляризация

$$KL(N(\mu, \sigma^2) | N(0, 1))$$
$$= \frac{1}{2} (\mu^2 + \sigma^2 - 1 - 2 \log \sigma)$$

```
def kl_loss(self, mu: Optional[Tensor] = None,
           logstd: Optional[Tensor] = None) -> Tensor:
    r"""Computes the KL loss, either for the passed arguments :obj:`mu` and :obj:`logstd`, or based on latent variables from last encoding.

    Args:
        mu (torch.Tensor, optional): The latent space for :math:`\mu`. If set to :obj:`None`, uses the last computation of :math:`\mu`.
        (default: :obj:`None`)
        logstd (torch.Tensor, optional): The latent space for :math:`\log\sigma`. If set to :obj:`None`, uses the last computation of :math:`\log\sigma^2`. (default: :obj:`None`)
    """
    mu = self.__mu__ if mu is None else mu
    logstd = self.__logstd__ if logstd is None else logstd.clamp(
        min=MAX_LOGSTD)
    return -0.5 * torch.mean(
        torch.sum(1 + 2 * logstd - mu**2 - logstd.exp()**2, dim=1))
```

<https://arxiv.org/pdf/1611.07308.pdf>

<https://www.youtube.com/watch?v=qA6U4nIK62E>

[https://pytorch-geometric.readthedocs.io/en/latest/\\_modules/torch\\_geometric/nn/models/autoencoder.html](https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/autoencoder.html)

## 5. torch\_geometric.nn.models.autoencoder



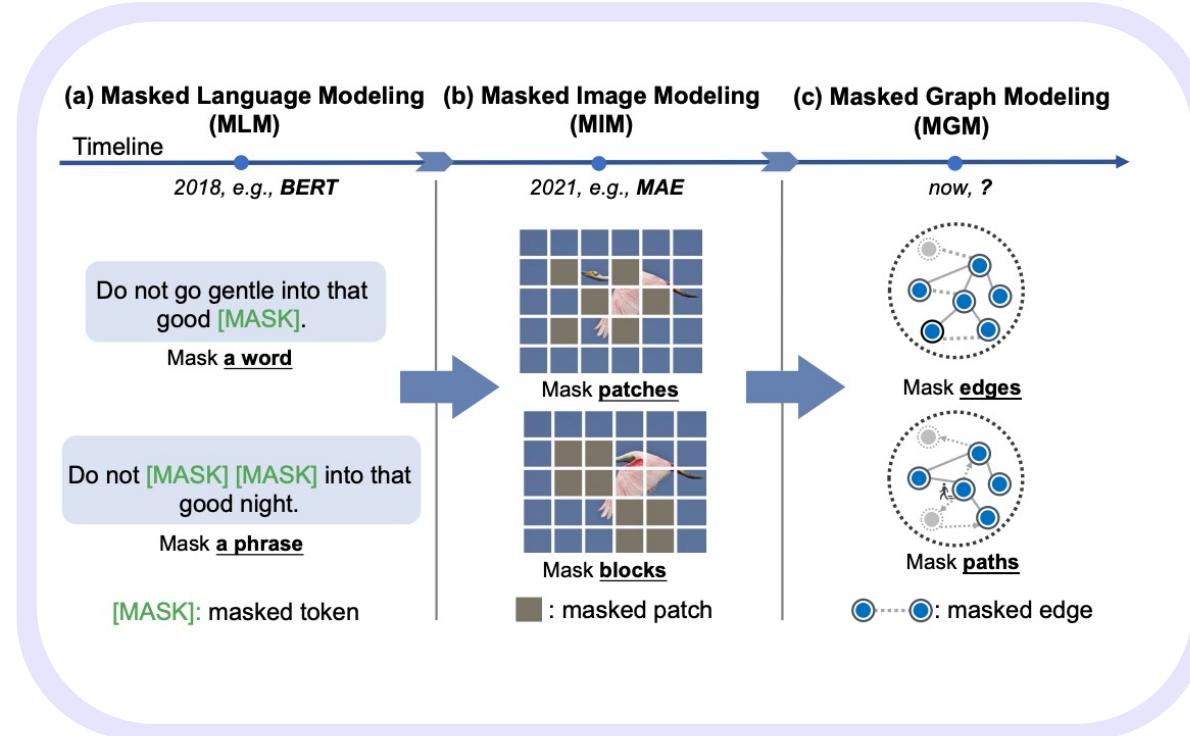
PyTorch  
geometric



- GAE
- VGAE
- ARGA
- ARGVA

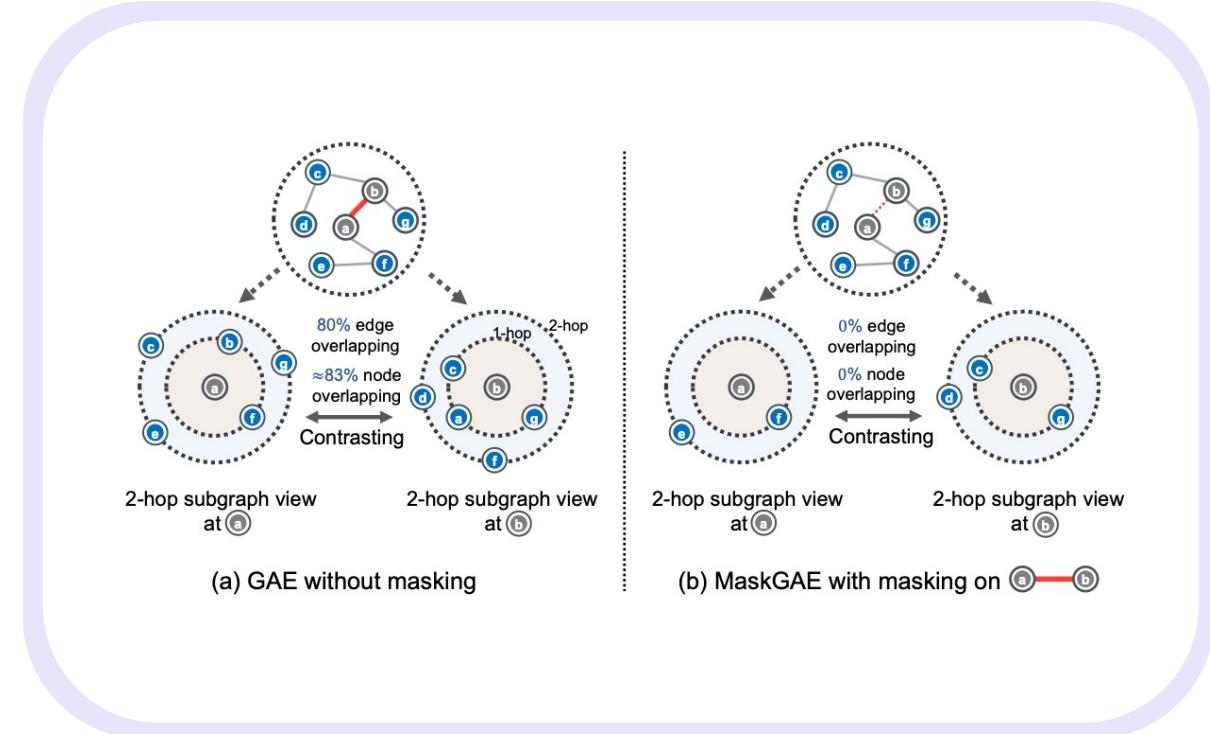
[https://pytorch-geometric.readthedocs.io/en/latest/\\_modules/torch\\_geometric/nn/models/autoencoder.html](https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/autoencoder.html)

# 5. Графовые автокодировщики – MASKGAE (2023)

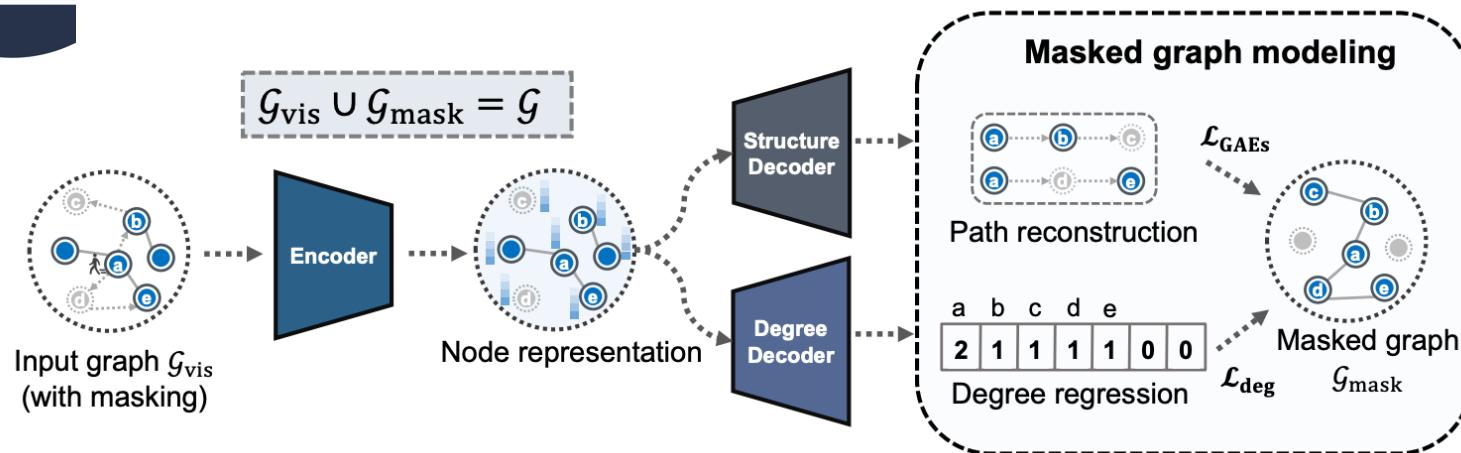


What's Behind the Mask: Understanding Masked Graph Modeling for Graph Autoencoders  
<https://arxiv.org/pdf/2205.10053.pdf>

# 5. Графовые автокодировщики – MASKGAE (2023)



# 5. Графовые автокодировщики – MASKGAE (2023)



# SOTA Outlier detection на графах

03

# 1. Статья и бенчмарки



NEURAL INFORMATION  
PROCESSING SYSTEMS

BOND: Benchmarking Unsupervised Outlier Node Detection  
on Static Attributed Graphs (2022)

<https://arxiv.org/abs/2206.10071>



Python library for graph outlier detection (anomaly detection)

<https://github.com/pygod-team/pygod>

<https://github.com/FelixDJC/Awesome-Graph-Anomaly-Detection>  
<https://github.com/Kaslanarian/SAGOD>



## Implemented Algorithms

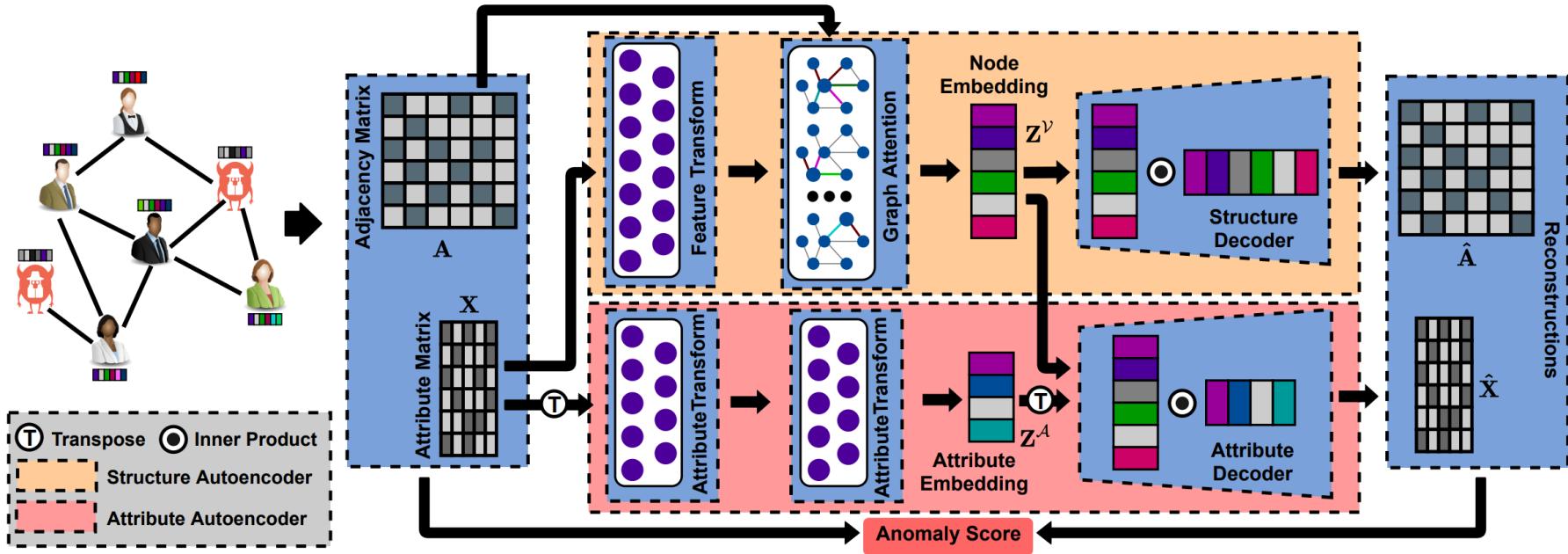
Abbr	Year	Backbone	Sampling	Ref
SCAN	2007	Clustering	No	<a href="#">[3]</a>
GAE	2016	GNN+AE	Yes	<a href="#">[4]</a>
Radar	2017	MF	No	<a href="#">[5]</a>
ANOMALOUS	2018	MF	No	<a href="#">[6]</a>
ONE	2019	MF	No	<a href="#">[7]</a>
DOMINANT	2019	GNN+AE	Yes	<a href="#">[8]</a>
DONE	2020	MLP+AE	Yes	<a href="#">[9]</a>
AdONE	2020	MLP+AE	Yes	<a href="#">[9]</a>
AnomalyDAE	2020	GNN+AE	Yes	<a href="#">[10]</a>
GAAN	2020	GAN	Yes	<a href="#">[11]</a>
OCGNN	2021	GNN	Yes	<a href="#">[12]</a>
CoLA	2021	GNN+AE+SSL	Yes	<a href="#">[13]</a>
GUIDE	2021	GNN+AE	Yes	<a href="#">[14]</a>
CONAD	2022	GNN+AE+SSL	Yes	<a href="#">[15]</a>

## 2. Ключевые поинты



- Нет универсального метода – зависит и от задачи (структурные аномалии или контекстные) и датасета
- Ближайшего окружения (1-hop) как правило достаточно для детекции структурных выбросов
- DL-методы менее стабильны в терминах дисперсии ROC-AUC
- На маленьких графах DL-методы проигрывают классическим

### 3. Anomaly DAE (2020)

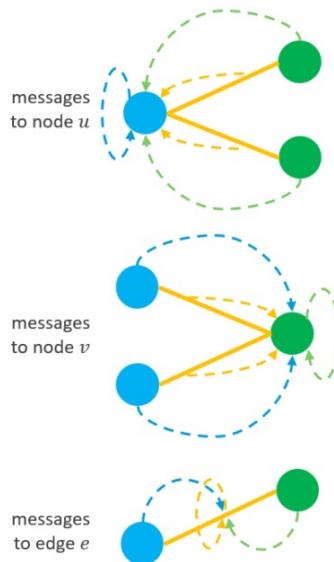


# 4. Состояние на 2022 год

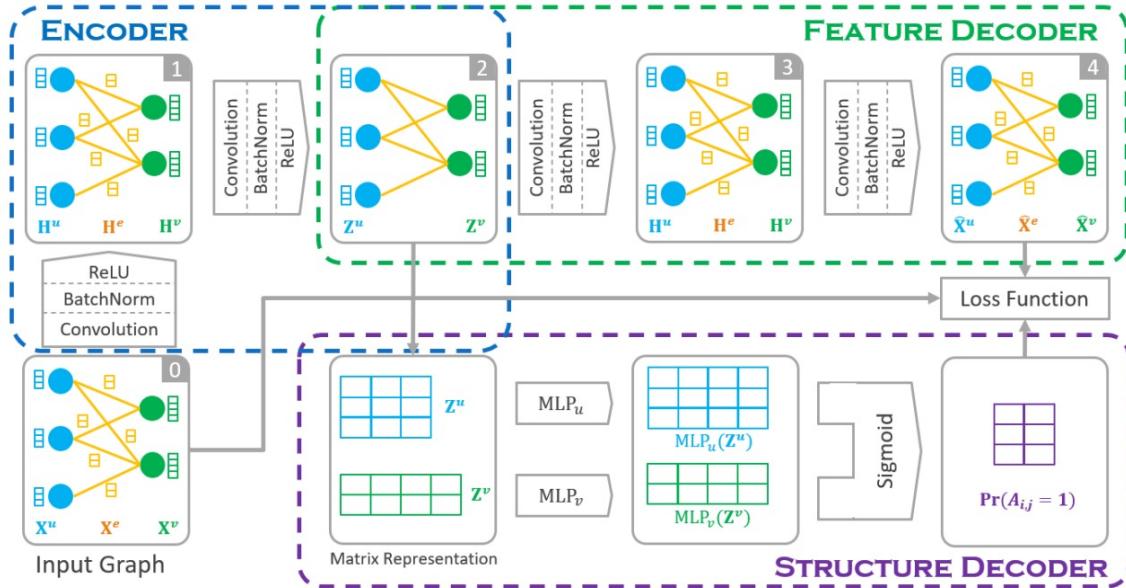
TAXONOMY AND SUMMARY OF GRAPH ANOMALY DETECTION METHODS USING GNN MODELS.

Graph type	Anomaly type	Network architecture	Method	Summary (key issue addressed → solution)
Static graph	Node anomaly	GCN-based GAE	DOMINANT [7] (2019)	Complex interactions, sparsity, non-linearity → GCN-based encoder
			Dual-SVDAE [21] (2021)	Overfitting for normal & abnormal → hypersphere embedding space
			GUIDE [22] (2021)	Complex interactions → higher-order structure decoder
			SpecAE [8] (2019)	Over-smoothing issue → tailored embedding space
			ComGA [23] (2022)	Over-smoothing issue → community-specific representation
			ALARM [24] (2020)	Heterogeneous attributes → multiple GCN-based encoders
			AnomMAN [25] (2022)	Heterogeneous attributes → multiple GCN-based encoders
			SL-GAD [26] (2021)	Contextual information → subgraph sampling & contrastive learning
		GCN alone	Semi-GCN [27] (2021)	Label information → semi-supervised learning by GCN
			HCM [28] (2021)	Label & contextual information → hop-count prediction model
	Edge anomaly	GAT-based GAE	ResGCN [29] (2021)	Over-smoothing issue → GCN with residual-based attention
			CoLA [30] (2021)	Targeting issue of GAE → contrastive self-supervised learning
			ANEMONE [31] (2021)	Contextual information → multi-scale contrastive learning
			PAMFUL [32] (2021)	Contextual information → pattern mining algorithm with GCN
		Other GNN-based model	AnomalyDAE [33] (2020)	Complex interactions → GAT-based encoder
			GATAE [34] (2020)	Over-smoothing issue → GAT-based encoder
Dynamic graph	Subgraph anomaly	GCN-based GAE	AEGIS [35] (2020)	Handling unseen nodes → generative adversarial learning with GAE
			OCGNN [36] (2021)	Targeting issue of GAE → GNN with hypersphere embedding space
		GCN alone	AAGNN [37] (2021)	Targeting issue of GAE → GNN with hypersphere embedding space
			Meta-GDN [38] (2021)	Hard work to label anomalies → meta-learning with auxiliary graphs
	Graph-level anomaly	GCN-based GAE	AANE [39] (2020)	Noise or adversarial links → GAE with a loss for anomalous links
			eFraudCom [40] (2022)	Fraud detection → heterogeneous graph and representative data sampling
	Edge anomaly	GCN and GRU	SubGNN [41] (2021)	Fraud detection → GIN and extracting and relabeling subgraphs
			HO-GAT [42] (2021)	Abnormal subgraphs → hybrid-order attention with motif instances
			OCGIN [43] (2021)	Graph-level anomaly detection → graph classification with GIN
	Node anomaly	GCN alone	OCGTL [44] (2022)	Hypersphere collapse → set of GNNs for embedding
			GLocalKD [45] (2022)	Graph-level anomalies → joint learning global & local normality
			AddGraph [10] (2019)	Long-term patterns → temporal GCN with attention-based GRU
	Dynamic graph	GCN & DRNN-based GAE	DynAD [46] (2020)	Long-term patterns → temporal GCN with attention-based GRU
			Hierarchical-GCN [47] (2020)	Dynamic data evaluation → temporal & hierarchical GCN
			StrGNN [48] (2021)	Structural change → mining unusual temporal subgraph structures
			H-VGRAE [49] (2020)	Anomalous nodes → modeling stochasticity and multi-scale ST dependency
	Node anomaly	GCN and GRU	DEGCN [50] (2022)	To capture node- and global-level patterns → DGCRN and GGRU
			TDG with GCN [51] (2022)	Malicious connections on traffic → extracting TDGs

# 5. GraphBEAN (2023)



(a) Message passing flow



(b) GraphBEAN architecture: encoder, feature decoder, and structure decoder (4 layers)

Fig. 1. GraphBEAN architecture and message passing scheme

Interaction-Focused Anomaly Detection on Bipartite  
Node-and-Edge-Attributed Graphs

# 5. GraphBEAN (2023)

THE MEAN AND (IN PARENTHESIS) STANDARD DEVIATION OF THE AUC-PR METRICS OVER MULTIPLE EXPERIMENT RUNS IN SMALL DATASETS. BOLD NUMBERS INDICATE THE BEST OR NOT-SIGNIFICANTLY-WORSE-THAN-THE-BEST RESULT (WILCOXON SIGNED-RANK TEST,  $\alpha = 0.05$ ).

Model	Dataset	FINEFOODS-SMALL			MOVIES-SMALL			WIKIPEDIA			REDDIT		
		$\mathcal{U}$	$\mathcal{V}$	$\mathcal{E}$									
FRAUDAR		0.256 (0.07)	0.392 (0.07)	0.279 (0.13)	0.229 (0.12)	0.188 (0.11)	0.260 (0.16)	0.102 (0.03)	0.085 (0.09)	0.043 (0.04)	0.059 (0.02)	0.101 (0.01)	0.011 (0.007)
IsoForest		0.090 (0.02)	0.166 (0.04)	0.794 (0.12)	0.127 (0.05)	0.181 (0.08)	0.827 (0.10)	0.226 (0.06)	0.499 (0.12)	0.278 (0.10)	0.361 (0.08)	<b>0.608</b> (0.07)	0.172 (0.039)
DOMINANT		0.735 (0.10)	0.721 (0.10)	0.686 (0.12)	0.631 (0.09)	0.708 (0.08)	0.389 (0.16)	0.164 (0.03)	0.179 (0.04)	0.049 (0.02)	0.121 (0.02)	0.186 (0.01)	0.016 (0.003)
AnomalyDAE		0.770 (0.09)	0.773 (0.09)	0.683 (0.12)	0.679 (0.12)	0.753 (0.10)	0.556 (0.10)	0.174 (0.03)	0.193 (0.04)	0.051 (0.02)	0.128 (0.02)	0.192 (0.02)	0.015 (0.003)
CONAD		0.740 (0.10)	0.721 (0.10)	0.691 (0.12)	0.684 (0.09)	0.695 (0.08)	0.564 (0.09)	0.165 (0.03)	0.182 (0.04)	0.052 (0.05)	0.116 (0.02)	0.180 (0.18)	0.016 (0.003)
AdOne		0.239 (0.05)	0.162 (0.03)	0.048 (0.01)	0.164 (0.03)	0.129 (0.03)	0.021 (0.01)	0.205 (0.04)	0.128 (0.03)	0.025 (0.01)	0.138 (0.02)	0.133 (0.01)	0.008 (0.001)
GraphBEAN (ours)		<b>0.855</b> (0.08)	<b>0.875</b> (0.07)	<b>0.876</b> (0.09)	<b>0.911</b> (0.04)	<b>0.911</b> (0.04)	<b>0.888</b> (0.08)	<b>0.441</b> (0.09)	<b>0.571</b> (0.03)	<b>0.415</b> (0.11)	<b>0.427</b> (0.06)	<b>0.631</b> (0.04)	<b>0.296</b> (0.038)

(2 типа вершин –  $u$  и  $v$ )

# Зачем уметь решать эту задачу

## 04

# 1. Где применяется



- Детекция фрода и обнала
- Детекция спамеров и злоумышленников в социальных сетях
- Да почти везде:

**Node embedding-based graph autoencoder outlier detection for adverse pregnancy outcomes  
(14 ноября 2023)**

<https://www.nature.com/articles/s41598-023-46726-4>