

Устойчивость GNN и ограничения в их применении

Зелинский Никита

План лекции

О1. История вопроса, картинки

План лекции

O1. История вопроса, картинки

O2. Как в графах

План лекции

О1. История вопроса, картинки

О2. Как в графах

О3. Защита в графах

В картинках

01

С чего все началось

Представим что есть обученный многоклассовый классификатор $p(x) \in \mathbb{R}^c$

Как выразить направление движения к конкретному классу $class_i$?

$$\nabla \log p(class_i | x)$$

Самая простая атака – small norm – найдем наименьшее ε , такое что:

$$\operatorname{argmax} p(x + \varepsilon) \neq \operatorname{argmax} p(x)$$

С чего все началось

$$J(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) \approx J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x}).$$

Maximize

$$J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_\infty \leq \epsilon$$

$$\Rightarrow \tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})).$$

FGSM (Fast Gradient Signed Method)

Обновлять не веса и исходное изображение!

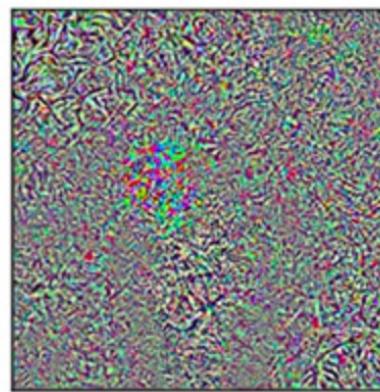
$$x_{adv} = x + \epsilon \operatorname{sign}(\nabla_x J(x, y_{true}))$$



X

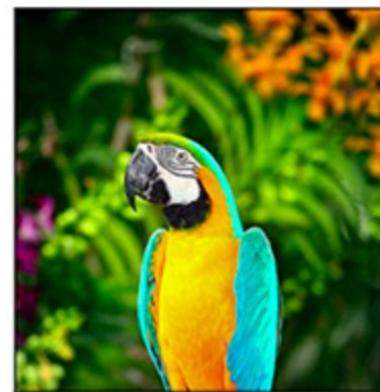
97.3% macaw

+



$\operatorname{sign}(\nabla_x J(\theta, X, Y))$

=



$X + \epsilon \cdot \operatorname{sign}(\nabla_x J(\theta, X, Y))$

88.9% bookcase

J – функция потерь классификации

y_{true} -- истинная метка класса

"The sign function is used to ensure that the perturbation is added in the direction that maximizes the loss function"

FGSM (Fast Gradient Signed Method)

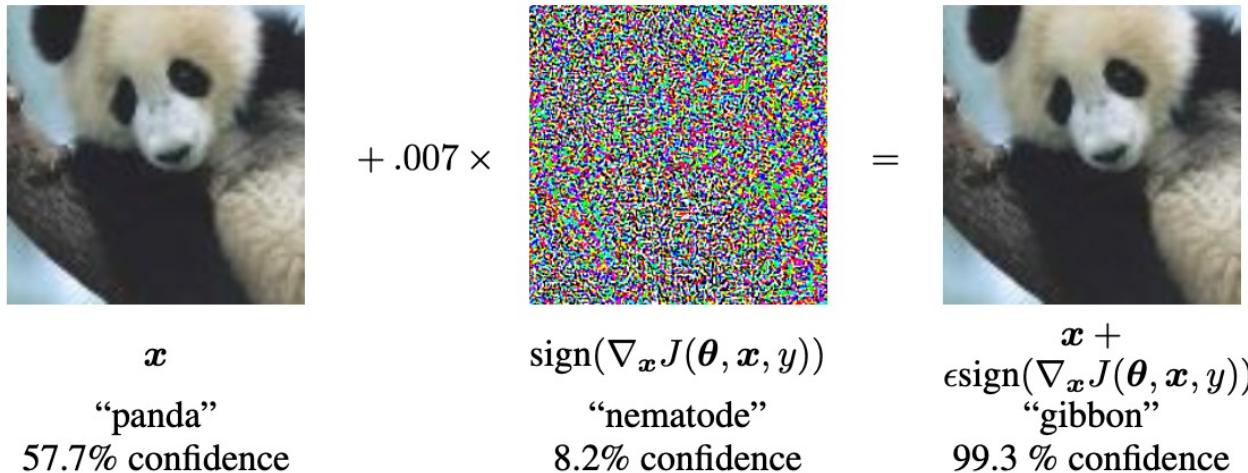


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet’s classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet’s conversion to real numbers.

Adversarial examples are not noise!

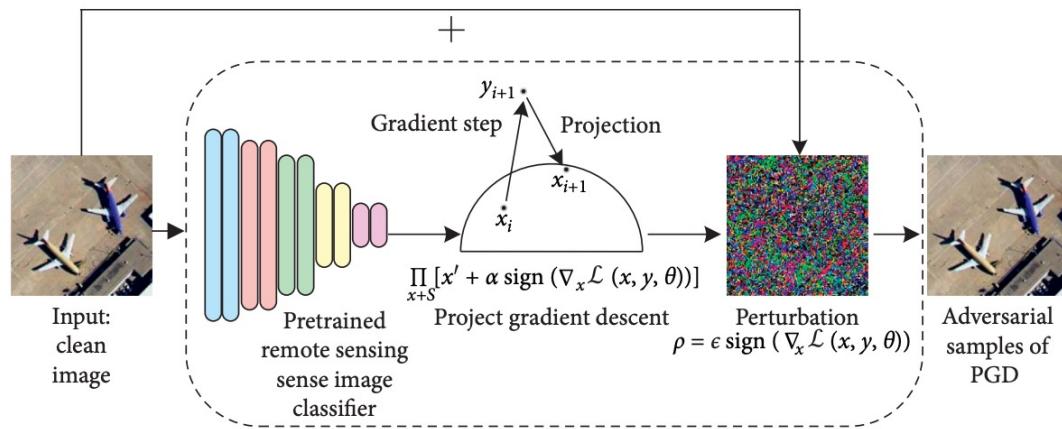
Если добавить шума – они все равно останутся adversarial examples

Projected Gradient Descent (PGD)

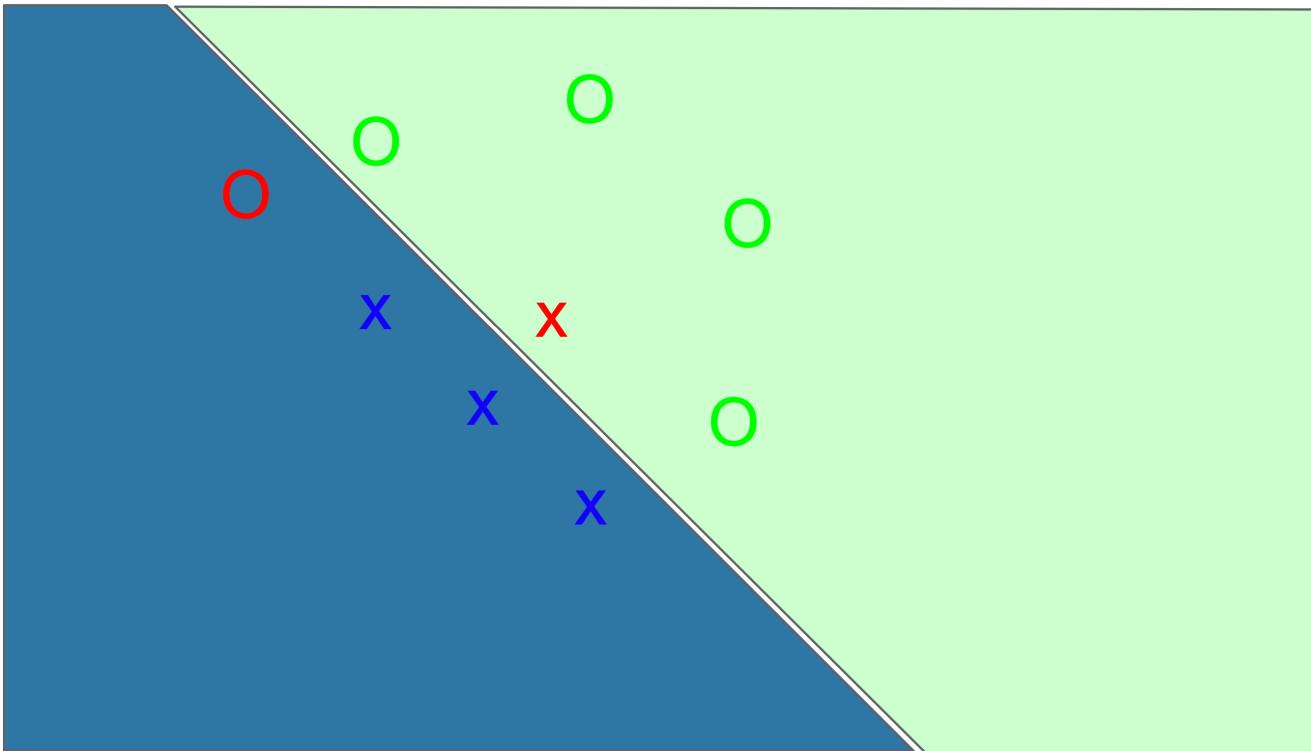
$$x_0 = x$$

$$x_{t+1} = Clip(x_t + \epsilon sign (\nabla_x J(x, y_{true})))$$

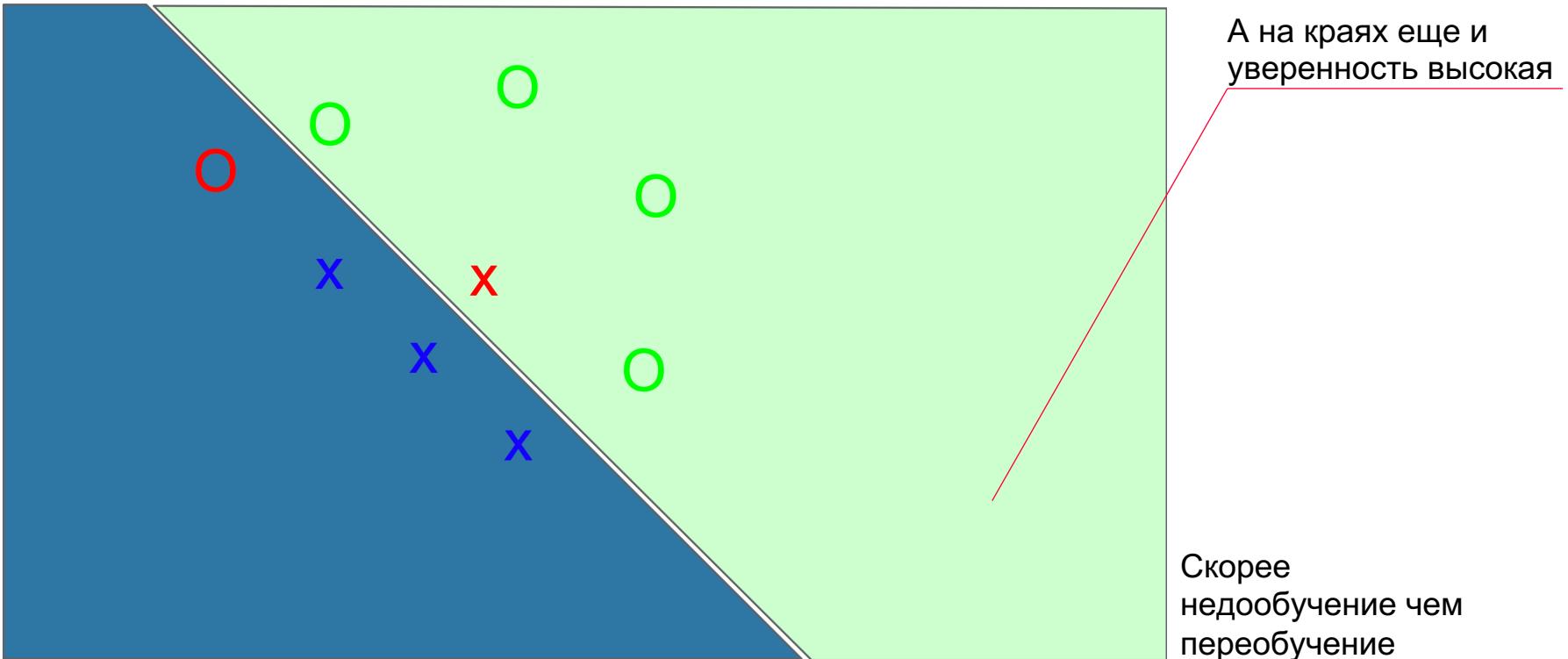
		Target Classification (L_2)									
		0	1	2	3	4	5	6	7	8	9
Source Classification	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3	3	3	3
	4	4	4	4	4	4	4	4	4	4	4
	5	5	5	5	5	5	5	5	5	5	5
	6	6	6	6	6	6	6	6	6	6	6
	7	7	7	7	7	7	7	7	7	7	7
	8	8	8	8	8	8	8	8	8	8	8
	9	9	9	9	9	9	9	9	9	9	9



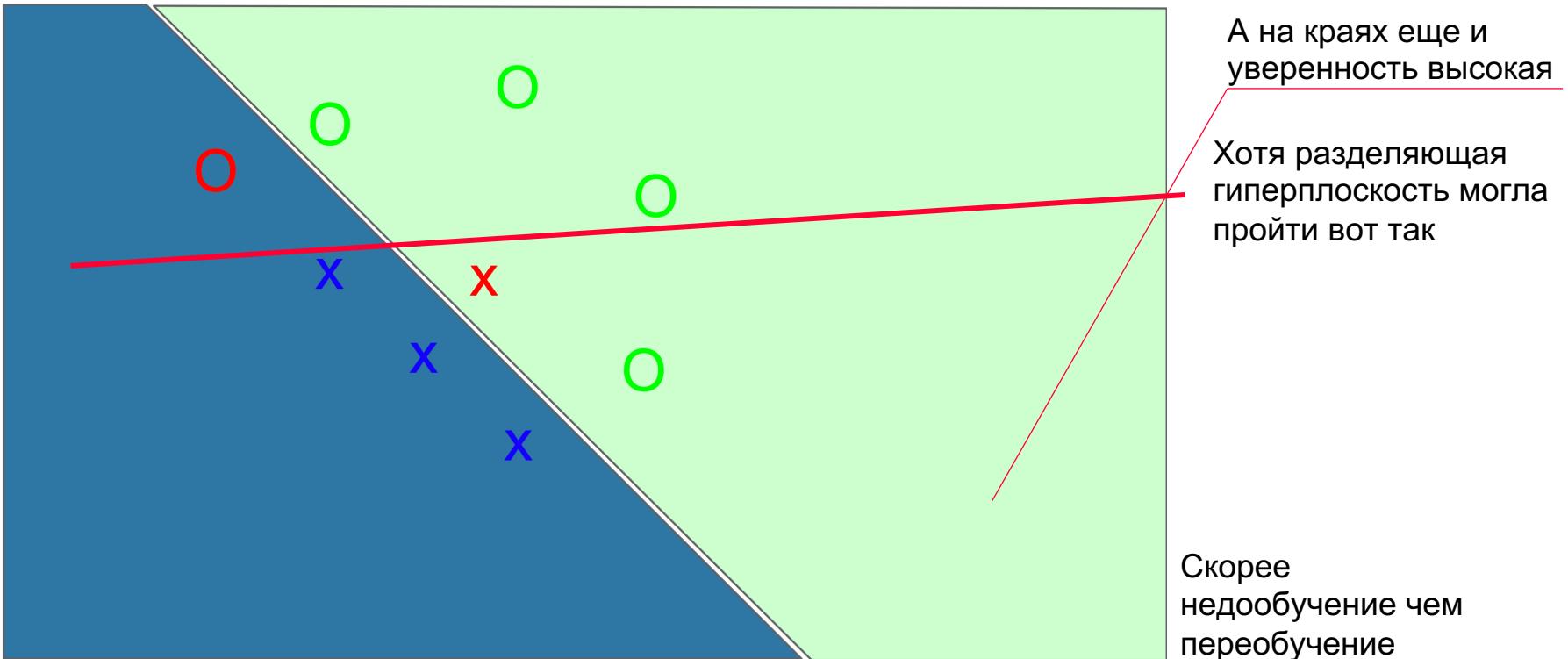
Мб наши модели слишком линейны?



Мб наши модели слишком линейны?



Мб наши модели слишком линейны?



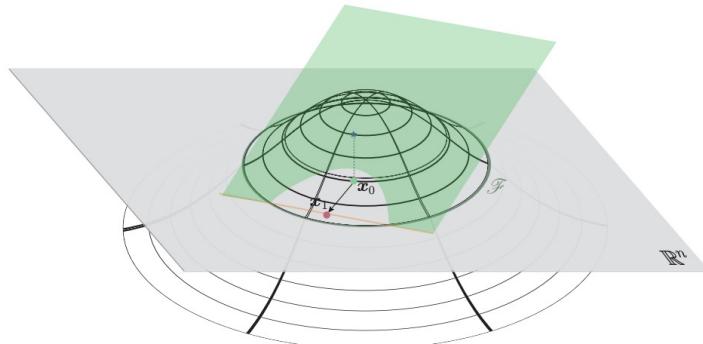
DeepFool

Линеаризация разделяющей поверхности

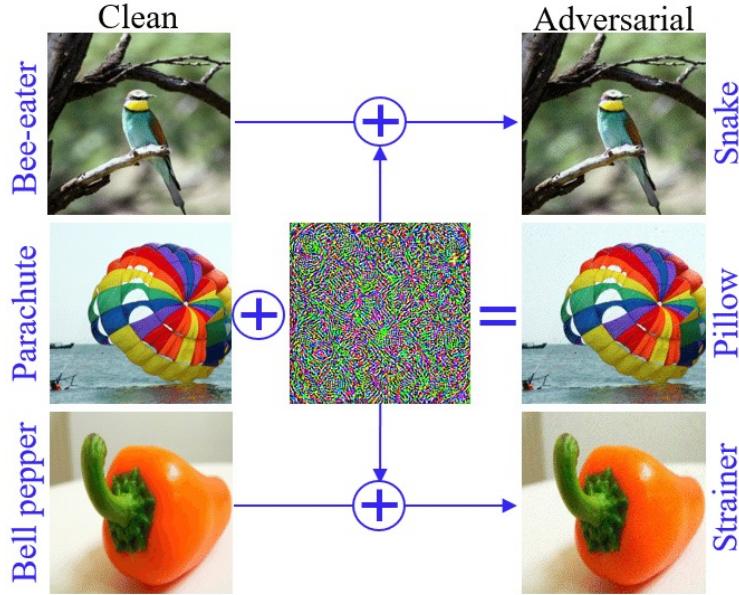
$$f(x) \approx f(x_0) + f'(x)(x - x_0)$$

Algorithm 1 DeepFool for binary classifiers

- 1: **input:** Image \mathbf{x} , classifier f .
- 2: **output:** Perturbation $\hat{\mathbf{r}}$.
- 3: Initialize $\mathbf{x}_0 \leftarrow \mathbf{x}$, $i \leftarrow 0$.
- 4: **while** $\text{sign}(f(\mathbf{x}_i)) = \text{sign}(f(\mathbf{x}_0))$ **do**
- 5: $\mathbf{r}_i \leftarrow -\frac{f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|_2^2} \nabla f(\mathbf{x}_i)$,
- 6: $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \mathbf{r}_i$,
- 7: $i \leftarrow i + 1$.
- 8: **end while**
- 9: **return** $\hat{\mathbf{r}} = \sum_i \mathbf{r}_i$.



Universal Adversarial Perturbation (UAP)



Algorithm 1 Computation of universal perturbations.

```
1: input: Data points  $X$ , classifier  $\hat{k}$ , desired  $\ell_p$  norm of  
the perturbation  $\xi$ , desired accuracy on perturbed sam-  
ples  $\delta$ .  
2: output: Universal perturbation vector  $v$ .  
3: Initialize  $v \leftarrow 0$ .  
4: while Err( $X_v$ )  $\leq 1 - \delta$  do  
5:   for each datapoint  $x_i \in X$  do  
6:     if  $\hat{k}(x_i + v) = \hat{k}(x_i)$  then  
7:       Compute the minimal perturbation that  
       sends  $x_i + v$  to the decision boundary:  
         
$$\Delta v_i \leftarrow \arg \min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$
  
8:       Update the perturbation:  
         
$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i).$$
  
9:     end if  
10:   end for  
11: end while
```

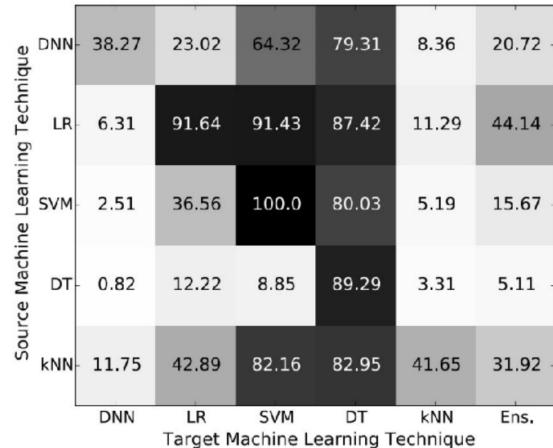
А еще

А еще:

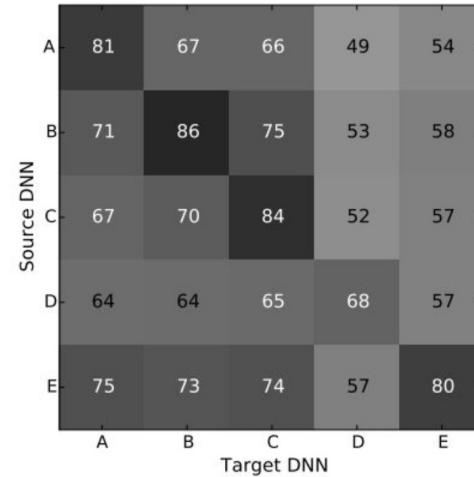
- Carlini-Wagner attack (adam оптимизатор)
- L-BFGS
- JSMA
- JSMA-Z
- JSMA-F
- Target class method $x_{adv} = x - \varepsilon \operatorname{sign}(\nabla_x J(x, y_{target}))$
- etc

Внезапно

Cross-technique transferability

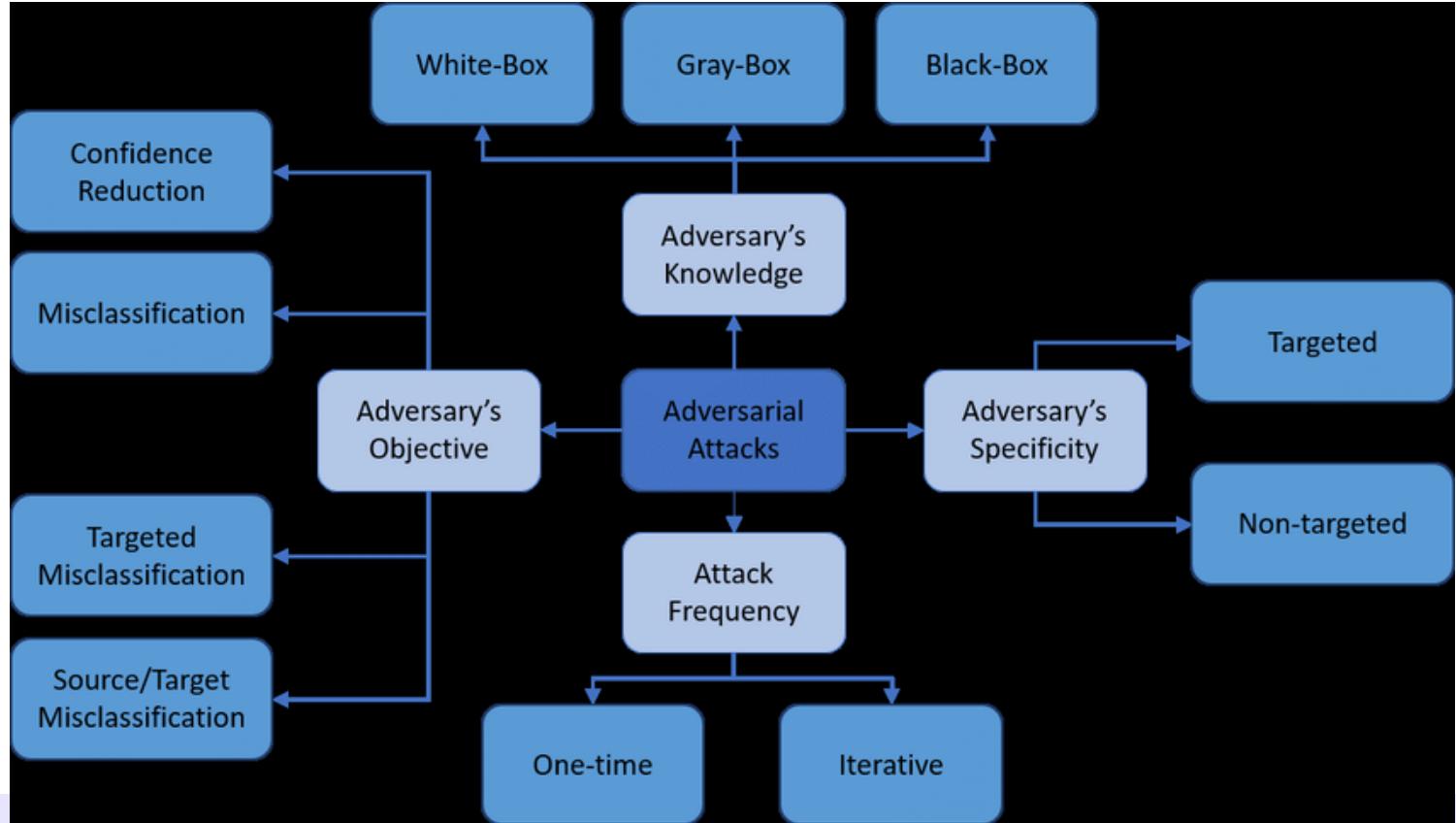


Cross-Training Data Transferability



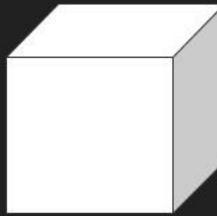
(Papernot 2016)

Какие еще бывают атаки

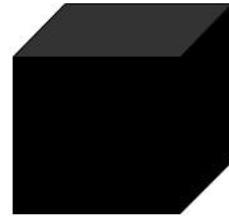


White / Black

White box vs Black box



- Архитектура сети известна
- Гиперпараметры известны
- Можно получить предсказания и градиент



- Архитектура сети неизвестна
- Гиперпараметры неизвестны
- Можно получить предсказания (с ограничениями)

/ Gray

Non-adaptive black box attack: атакующий алгоритм имеет доступ к распределению трейна исходной модели. Тогда строится своя модель и задача решается как white box.

Adaptive black box attack: атакующий алгоритм имеет возможность получать предикты исходной модели отправляя запросы

Strict black box attack: атакующий алгоритм имеет возможность получить часть трейна исходной модели.

Poison / Evasion

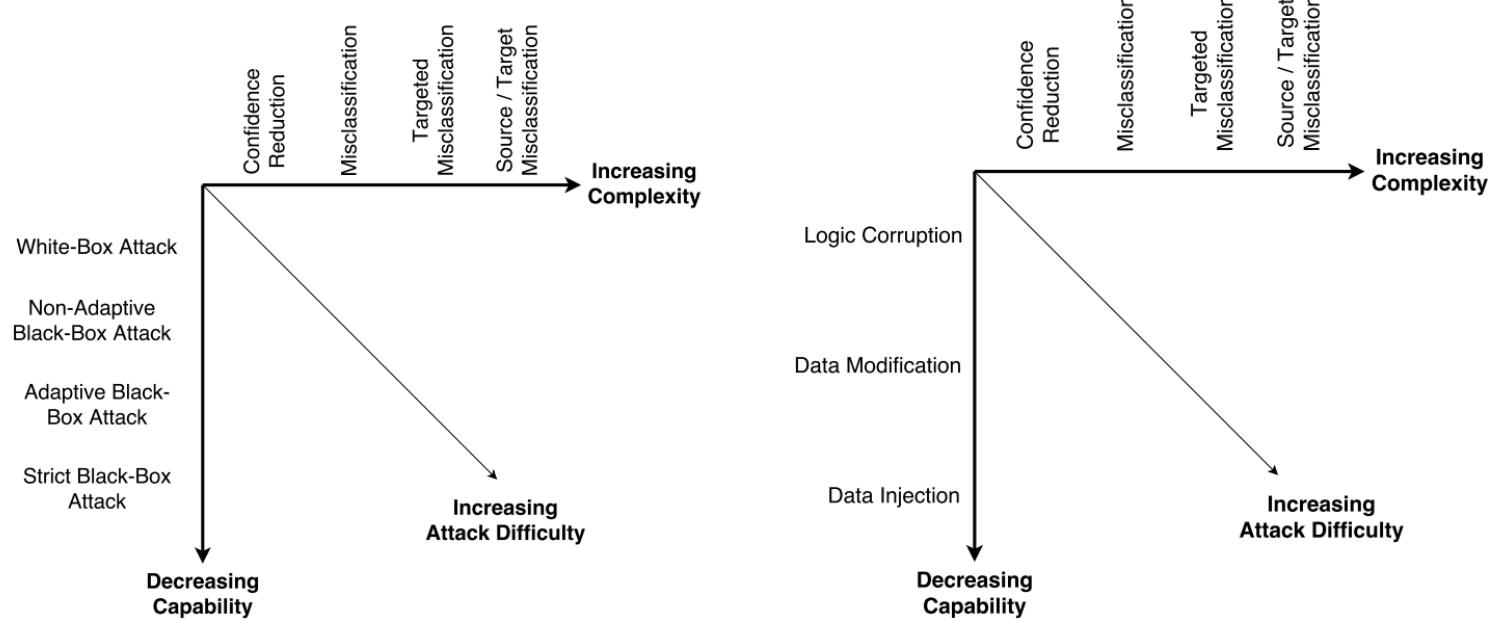
Evasion – атака уже обученной модели

Poison – атака во время обучения (например, изменение данных)

Цели

- Confidence reduction
- Misclassification
- Targeted misclassification (конкретный целевой класс)
- Source/Target misclassification (пара исходный класс – целевой класс)

Какие еще бывают атаки



a) Attack Difficulty with respect to adversarial capabilities and goals for Evasion Attacks

b) Attack Difficulty with respect to adversarial capabilities and goals for Poisoning Attacks

Бенчмарки



ROBUSTBENCH

A standardized benchmark for adversarial robustness

<https://robustbench.github.io/>



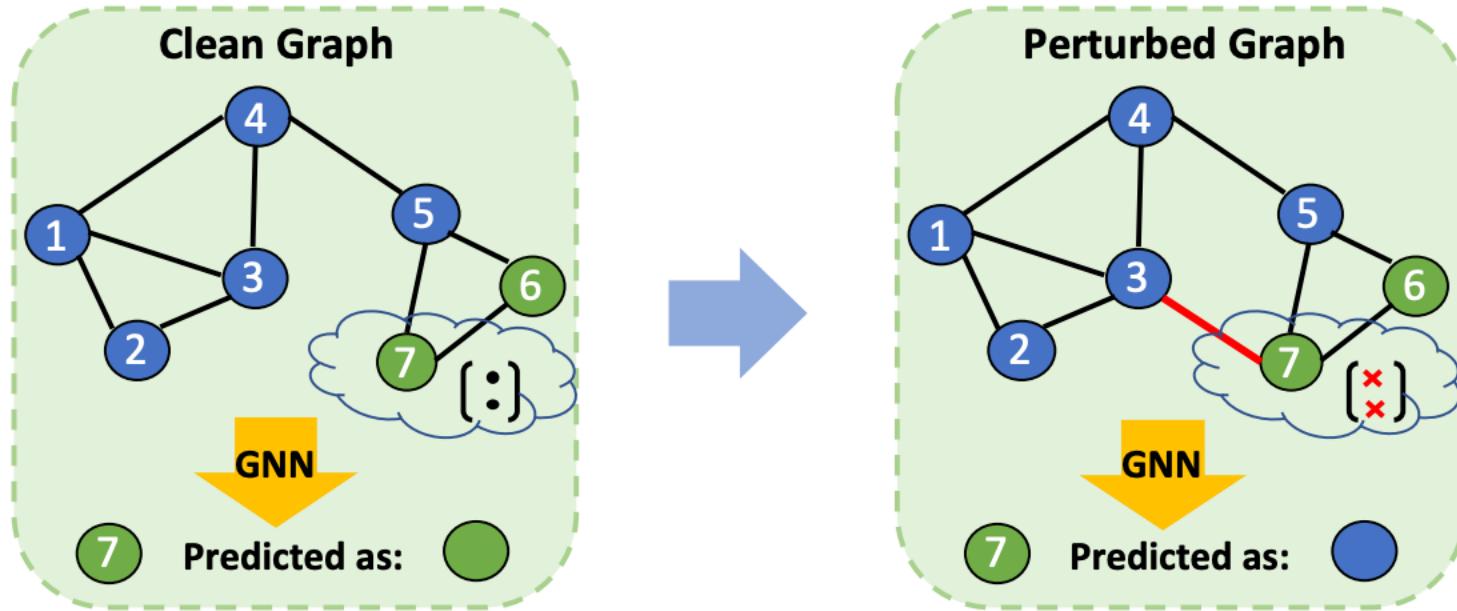
**Deep
Robust**

<https://github.com/DSE-MSU/DeepRobust?tab=readme-ov-file#graph-attack-and-defense>

В графах

02

На пальцах



Чуть менее на пальцах

$$\min_{\theta} \mathcal{L}_{\text{train}}(f_{\theta}(G)) = \sum_{v_i \in V_L} \ell(f_{\theta}(\mathbf{X}, \mathbf{A})_i, y_i)$$

$$\max \mathcal{L}_{\text{atk}}(f_{\theta}(\hat{G})) = \sum_{u \in V_t} \ell_{\text{atk}}(f_{\theta^*}(\hat{G})_u, y_u)$$

такое что $\theta^* = \arg \min_{\theta} \mathcal{L}_{\text{train}}(f_{\theta}(G'))$

и $\|\hat{\mathbf{A}} - \mathbf{A}\|_0 + \|\hat{\mathbf{X}} - \mathbf{X}\|_0 \leq \Delta$. Бюджет

Сеттинг

Кроме node classification есть атаки на node embeddings и link prediction

Часто в статьях рассматривается конкретная target вершина, на которой надо добиться ошибки модели

Метрики

Attack success rate

$$ASR = \frac{\# \text{успешных атак}}{\# \text{число всех атак}}$$

Classification margin

$$CM(t) = \log \frac{p_{t, \text{class}_t}}{\max_{\text{class} \neq \text{class}_t} p_{t, \text{class}}}$$

Misclassification Rate

$$MSR = \frac{\# \text{неверное классифицированных элементов}}{\# \text{всех элементов}}$$

Average Modified links

$$AML = \frac{\# \text{ измененных ребер}}{\# \text{всего ребер}}$$

Метрики

А еще

- DPR (Damage Prevention Ratio)
- Certified Accuracy
- Concealment measures
- Similarity score
- Average Worst-case Margin
- Robustness Merit (RM)
- Average Defense Rate (ADR)
- Average Confidence Different (ACD)
- ...

White / Gray /Black



Все параметры модели,
трейн, фичи, матрица
связности



Все промежуточные
варианты (например:
доступ к трейну, но не к
параметрам модели)



Все фичи, матрица
связности, предикты
атакуемой модели через
запросы

Эвристики

The ROAM heuristic given a budget b :

- **Step 1:** Remove the link between the source node, v^\dagger , and its neighbour of choice, v_0 ;
- **Step 2:** Connect v_0 to $b - 1$ nodes of choice, who are neighbours of v^\dagger but not of v_0 (if there are fewer than $b - 1$ such neighbours, connect v_0 to all of them).

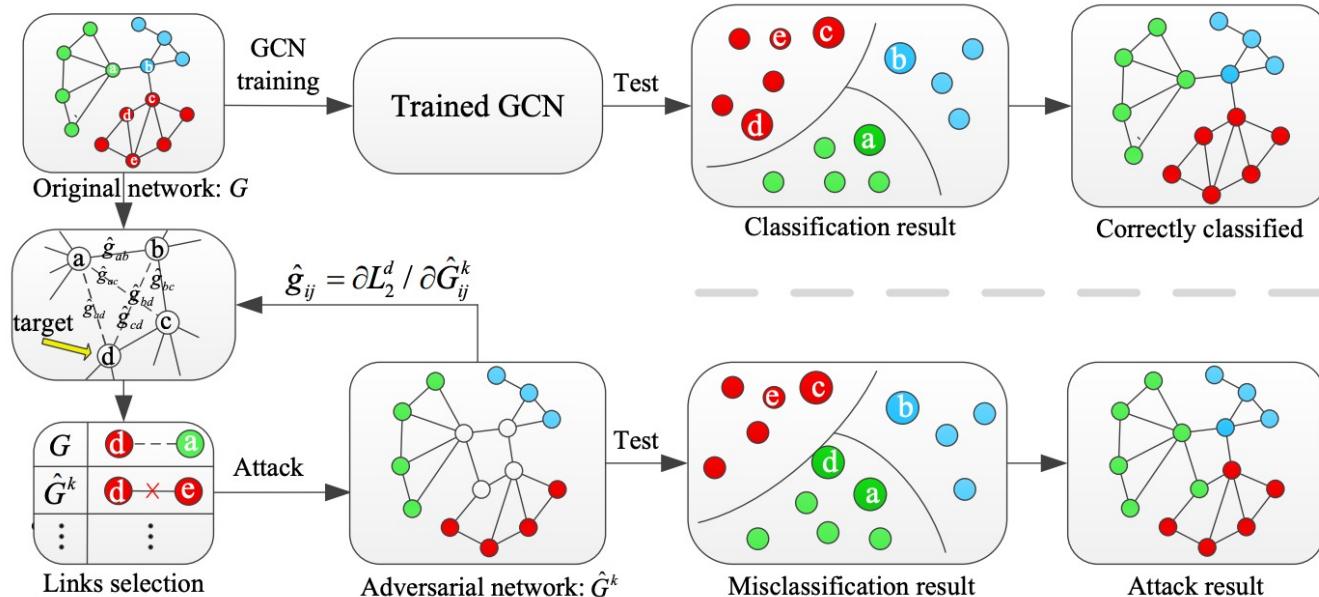
Remove one add many

The DICE heuristic given a budget b :

- **Step 1:** Disconnect $d \leq b$ links from within C^\dagger ;
- **Step 2:** Connect $b - d$ nodes from within C^\dagger to $b - d$ nodes from outside of C^\dagger .

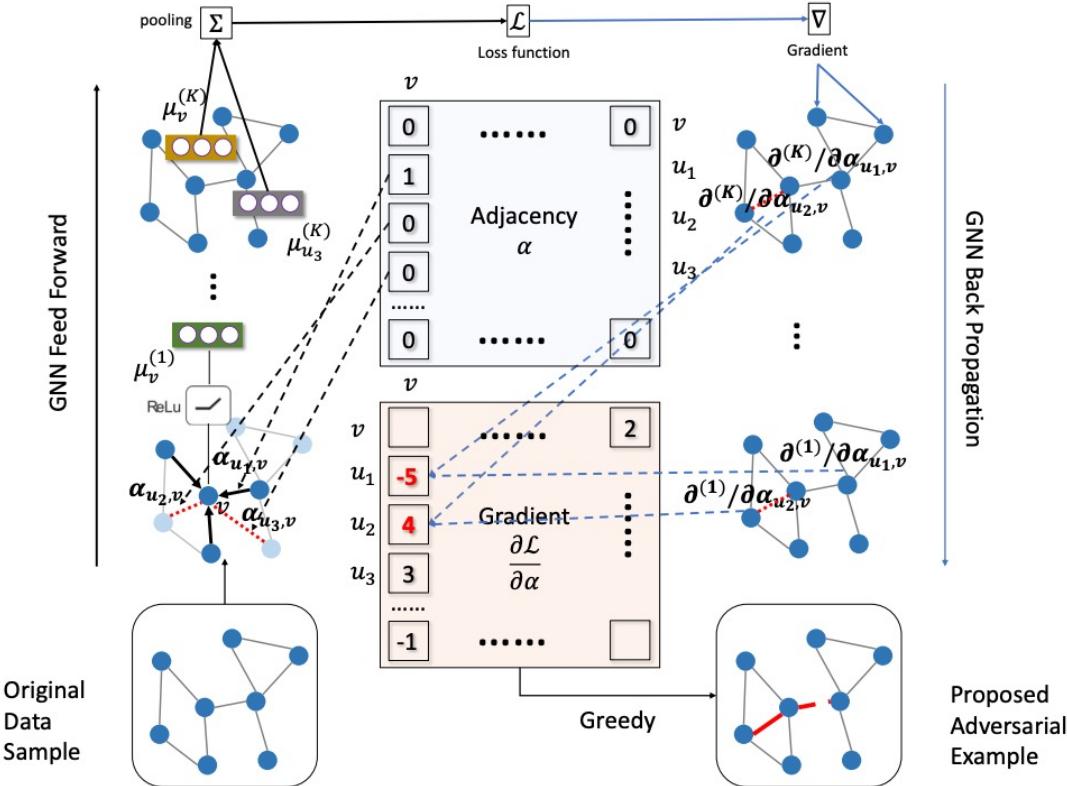
Disconnect Internally, Connect Externally

White-box gradient attack



"Then, for each target node, we design a target loss function, based on which we calculate the partial derivative, and further the gradient information, for each pair of nodes in the network. After that, we select the pair of nodes of the maximum absolute gradient to update the adversarial network."

White-box gradient attack



Adversarial Attack on Graph Structured Data (2018)

<https://arxiv.org/pdf/1806.02371.pdf>

PGD -> Projected Randomized Block Coordinate Descent (PR-BCD)

$$x_0 = x$$

$$x_{t+1} = Clip(x_t + \varepsilon sign(\nabla_x J(x, y_{true})))$$

vs

Algorithm 1 Projected Randomized Block Coordinate Descent (PR-BCD)

- 1: **Input:** Gr. (\mathbf{A}, \mathbf{X}), lab. \mathbf{y} , GNN $f_\theta(\cdot)$, loss \mathcal{L}
 - 2: **Parameter:** budget Δ , block size b , epochs E & $E_{\text{res.}}$, heuristic $h(\dots)$, learning rate α_t
 - 3: Draw w/o replacement $\mathbf{i}_0 \in \{0, 1, \dots, n^2 - 1\}^b$
 - 4: Initialize zeros for $\mathbf{p}_0 \in \mathbb{R}^b$
 - 5: **for** $t \in \{1, 2, \dots, E\}$ **do**
 - 6: $\hat{\mathbf{y}} \leftarrow f_\theta(\mathbf{A} \oplus \mathbf{p}_{t-1}, \mathbf{X})$
 - 7: $\mathbf{p}_t \leftarrow \mathbf{p}_{t-1} + \alpha_t \nabla_{\mathbf{p}_{t-1}[\mathbf{i}_{t-1}]} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
 - 8: Projection $\mathbf{p}_t \leftarrow \Pi_{\mathbb{E}[\text{Bernoulli}(\mathbf{p}_t)] \leq \Delta}(\mathbf{p}_t)$
 - 9: $\mathbf{i}_t \leftarrow \mathbf{i}_{t-1}$
 - 10: **if** $t \leq E_{\text{res.}}$ **then**
 - 11: $\text{mask}_{\text{res.}} \leftarrow h(\mathbf{p}_t)$
 - 12: $\mathbf{p}_t[\text{mask}_{\text{res.}}] \leftarrow \mathbf{0}$
 - 13: Resample $\mathbf{i}_t[\text{mask}_{\text{res.}}]$
 - 14: $\mathbf{P} \sim \text{Bernoulli}(\mathbf{p}_E)$ s.t. $\sum \mathbf{P} \leq \Delta$
 - 15: Return $\mathbf{A} \oplus \mathbf{P}$
-

NETTACK

Graph structure preserving perturbations
 (речь о сохранении распределения степеней вершин)

Feature statistics preserving perturbations
 (сохраняем совместное распределение фич и их совстречаемость)

"Among these elements we pick the one which obtains the highest difference in the log-probabilities"

Adversarial Attacks on Neural Networks for Graph Data (2021)

<https://arxiv.org/pdf/1805.07984.pdf>

Algorithm 1: NETTACK: Adversarial attacks on graphs

Input: Graph $G^{(0)} \leftarrow (A^{(0)}, X^{(0)})$, target node v_0 ,
 attacker nodes \mathcal{A} , modification budget Δ

Output: Modified Graph $G' = (A', X')$

Train surrogate model on $G^{(0)}$ to obtain W // Eq. (13);

$t \leftarrow 0$;

while $|A^{(t)} - A^{(0)}| + |X^{(t)} - X^{(0)}| < \Delta$ **do**

$C_{struct} \leftarrow \text{candidate_edge_perturbations}(A^{(t)}, \mathcal{A})$;

$e^* = (u^*, v^*) \leftarrow \arg \max_{e \in C_{struct}} s_{struct}(e; G^{(t)}, v_0)$;

$C_{feat} \leftarrow \text{candidate_feature_perturbations}(X^{(t)}, \mathcal{A})$;

$f^* = (u^*, i^*) \leftarrow \arg \max_{f \in C_{feat}} s_{feat}(f; G^{(t)}, v_0)$;

if $s_{struct}(e^*; G^{(t)}, v_0) > s_{feat}(f^*; G^{(t)}, v_0)$ **then**

$G^{(t+1)} \leftarrow G^{(t)} \pm e^*$;

else $G^{(t+1)} \leftarrow G^{(t)} \pm f^*$;

$t \leftarrow t + 1$;

return : $G^{(t)}$

// Train final graph model on the corrupted graph $G^{(t)}$;

Защиты

ОЗ

Как повышают устойчивость

Adversarial training

Тренируем модель на исходных данных и на сгенерированных атакующими сетями, используем информацию о распределении на teste

Empowering Graph Representation Learning with Test-Time Graph Transformation (ICLR 2023)

<https://openreview.net/pdf?id=LnxI5pr018>

IDEA: Invariant Causal Defense for Graph Adversarial Robustness (2023)

<https://arxiv.org/pdf/2305.15792.pdf>

Сглаживания и рандомизации

Добавляем при обучении модели случайный шум в параметры самой модели, либо сэмплируем атаки и выбираем наиболее частый класс предикта

Certified Adversarial Robustness via Randomized Smoothing (2019)

<https://arxiv.org/pdf/1902.02918.pdf>

Certified Robustness of Graph Classification against Topology Attack with Randomized Smoothing (2020)

<https://arxiv.org/pdf/2009.05872.pdf>

Adversarial training

Labeled as bird



Still has same label (bird)



Decrease
probability
of bird class

Virtual adversarial training

Unlabeled; model
guesses it's probably
a bird, maybe a plane



→ Adversarial
perturbation
intended to
change the guess

New guess should
match old guess
(probably bird, maybe plane)



Как повышают устойчивость

Обфускация

Маскирование или обфускация градиента, модификация выходов модели (например, не нормализовать скоры и пр.)

Adversary for Social Good: Leveraging Attribute-Obfuscating Attack to Protect User Privacy on Social Networks (2023)

https://link.springer.com/chapter/10.1007/978-3-031-25538-0_37

Ансамблирование

Используем несколько разных моделей для предикта, результат голосованием

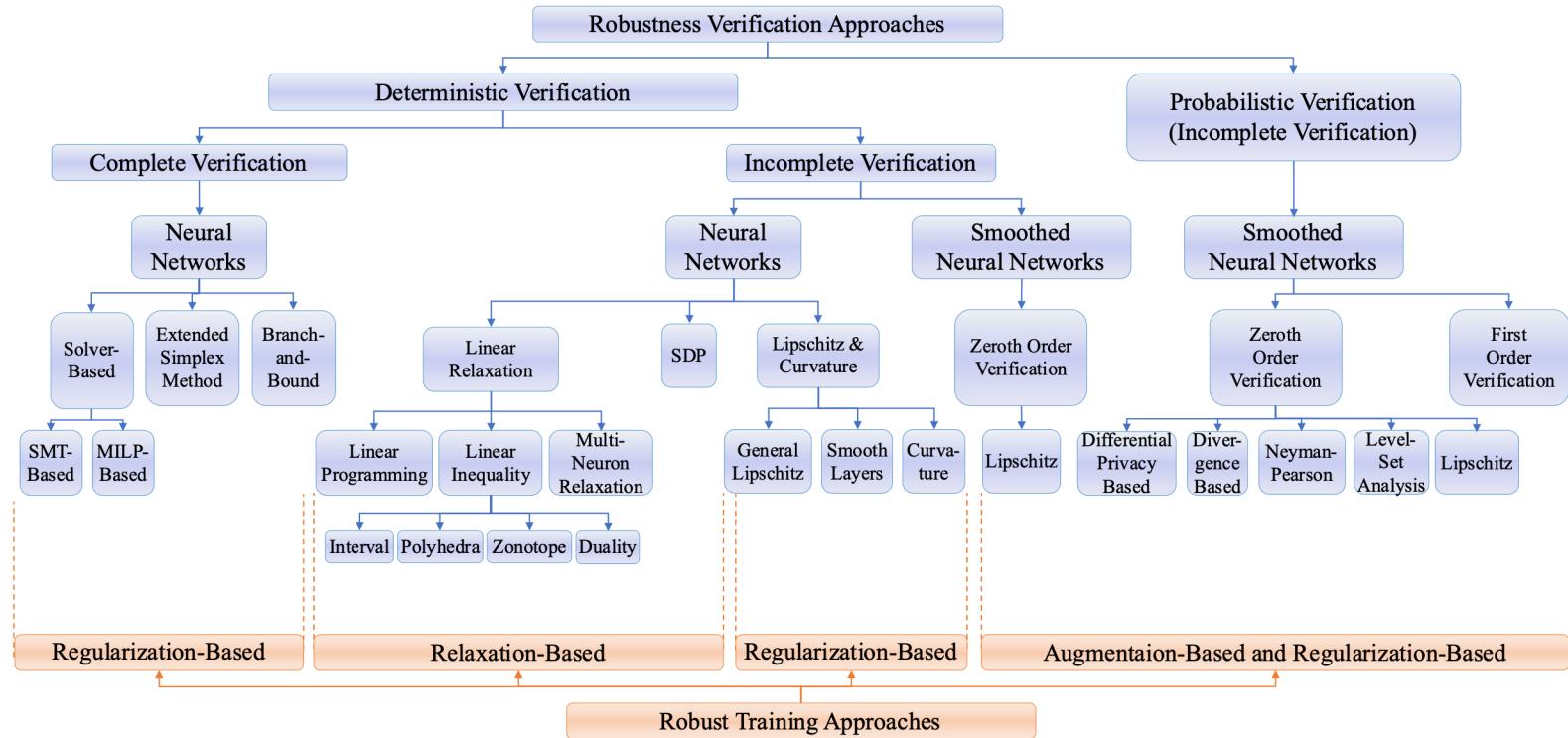
Certifiable Robustness

Получаем оценки на долю успешно атакованных нод для разных типов атак, определяем границы где модель робастна

Certifiable Robustness and Robust Training for Graph Convolutional Networks (KDD, 2019)

<https://arxiv.org/pdf/1906.12269.pdf>

Certifiable Robustness – тоже кроличья нора



Как повышают устойчивость

Adversarial Perturbation Detection и Graph Purification

Пытаемся либо идентифицировать либо очистить граф от данных, которые могут быть poisoned

Characterizing malicious edges targeting on graph neural networks (2018)

<https://openreview.net/pdf?id=HJxdAoCcYX>

GraphSAC: Detecting anomalies in large-scale graphs (2019)

<https://arxiv.org/pdf/1910.09589.pdf>

All you need is low (rank) defending against adversarial attacks on graphs (2020)

<https://dl.acm.org/doi/pdf/10.1145/3336191.3371789>

Adversarial examples for graph data: deep insights into attack and defense (2019)

<https://arxiv.org/pdf/1903.01610.pdf>

Attention Mechanism

С помощью attention определяем какие ребра и ноды poisoned, далее они получают меньший вес

GNNGUARD: Defending Graph Neural Networks against Adversarial Attacks

<https://arxiv.org/pdf/2006.08149.pdf>

Еще один взгляд на защиты

Taxonomy		Selected Defenses	Other Defenses
Improving graph	Unsupervised	Jaccard-GCN [48] SVD-GCN [12]	[10, 26, 50, 59, 60]
	Supervised	ProGNN [30]	[51, 43, 56]
Improving training	Robust training	n/a (see § C)	[6, 9, 14, 22, 27, 28, 41, 52, 53, 54]
	Further training principles	GRAND [15]	[5, 11, 29, 39, 42, 55, 61, 64, 65]
Improving architecture	Adaptively weighting edges	Rule-based	GNNGuard [58]
		Probabilistic	RGCN [63]
		Robust agg.	Soft-Median-GDC [17]
Miscellaneous		n/a (see above)	[40, 46, 49]

Are Defenses for Graph Neural Networks Robust? (NIPS, 2022)

<https://openreview.net/pdf?id=yCJVkELVT9d>

Кейсы

- E-com
- Fake news detectors
- Probability of default
- Устойчивость графа DNS-серверов

Attacking Fake News Detectors via Manipulating News Social Engagement (2023)

<https://arxiv.org/pdf/2302.07363.pdf>

Материалы

<https://github.com/safe-graph/graph-adversarial-learning-literature>

<https://github.com/ChandlerBang/awesome-graph-attack-papers>