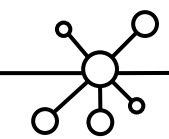
Reliable Graph Machine Learning

Simon Geisler

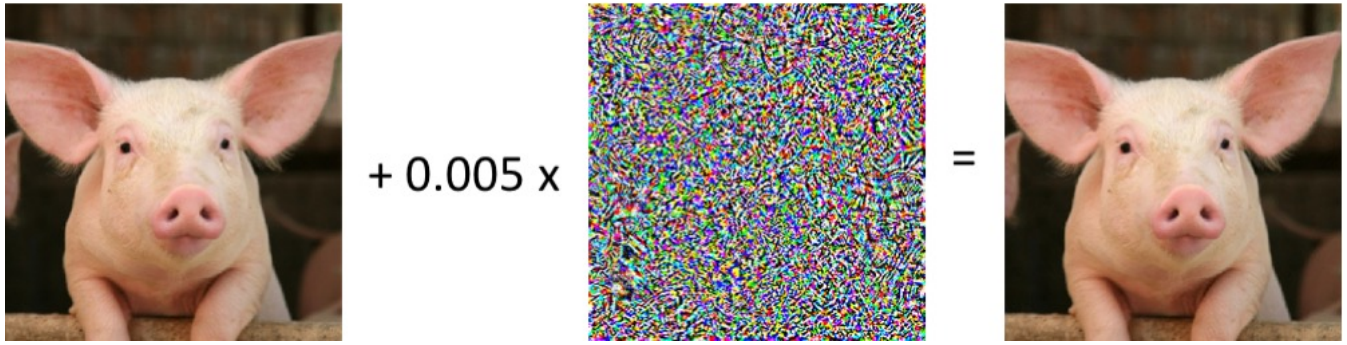


Reliability

This talk focuses on reliability in the sense of
worst case robustness w.r.t. perturbations of the input
– commonly referred to as **adversarial robustness**.

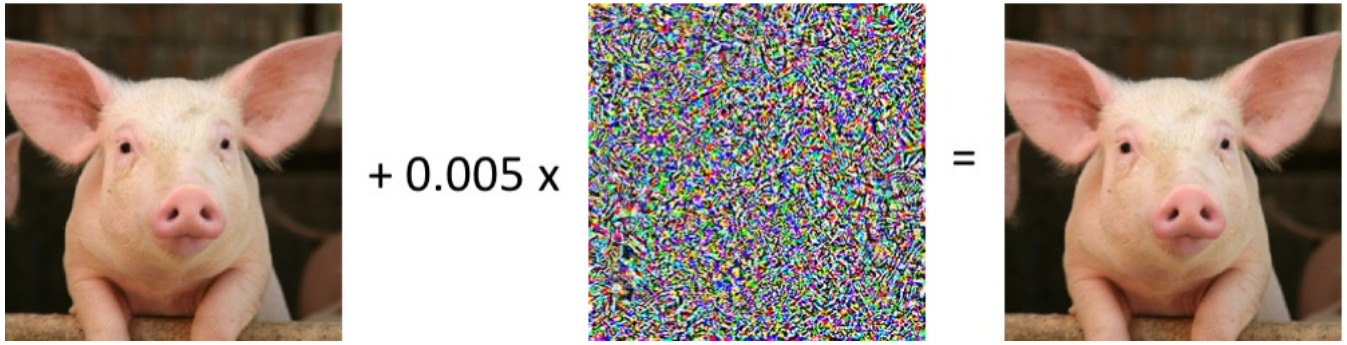
Adversarial Robustness

Predicted class: “pig” ?



Adversarial Robustness

Predicted class: **“pig”** “airliner”



Adversarial Robustness

ML systems classify the adversarially modified STOP sign as a speed limit sign, [Eykholt 2018]

Kevin Eykholt, et al. "Robust physical-world attacks on deep learning visual classification." CVPR 2018.

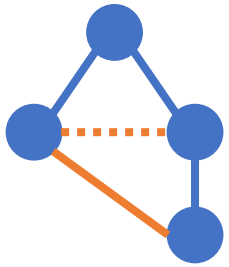
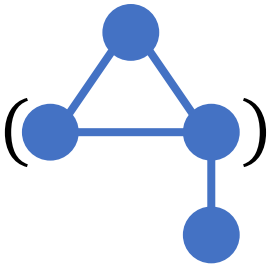
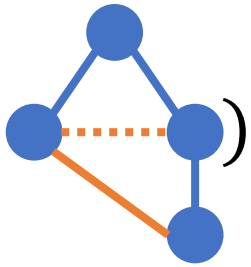
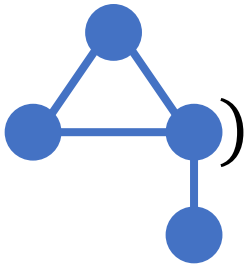
Motivations of Adversarial Robustness

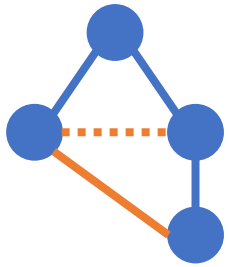
- security: We do not want a real-world adversary to succeed
- safety: Our application entails requirements on worst-case behaviour of f_θ

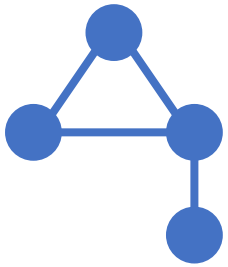
- …

- local sensitivity: We have some expectation about local behavior of f_θ
- generalization: The admissible perturbations $\mathcal{B}$ are linked to the semantics of the input instance

Adversarial Robustness in the Graph Domain

$\exists$  $\in \mathcal{B}($  $)$ s.t. $\operatorname{argmax}\big(f_\theta($  $)\big) \neq \operatorname{argmax}\big(f_\theta($  $)\big)$

Does there exist a **perturbed graph**  in the **neighborhood** $\mathcal{B}$ of

the **clean graph**  for that, e.g., the **model** f_θ **changes** its **prediction**.

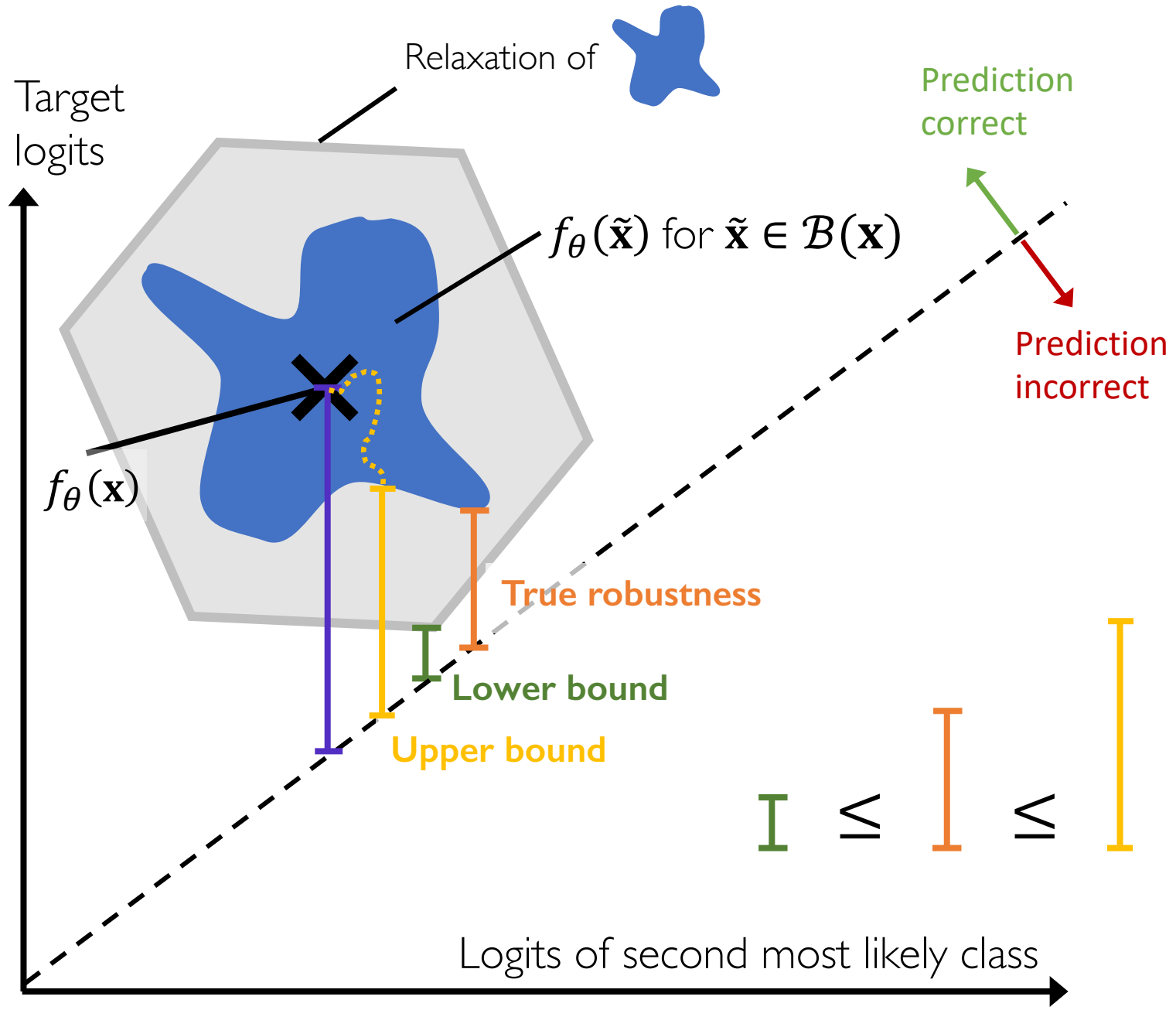
In this talk, $\mathcal{B}$ most frequently describes the insertion/deletion of edges

Certificate vs. Attack

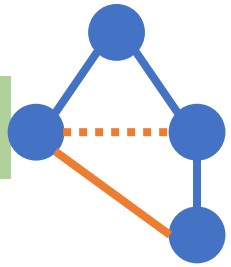
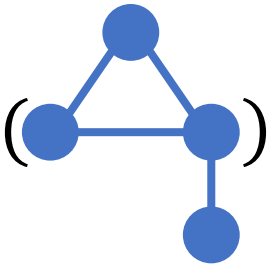
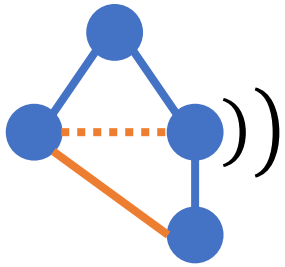
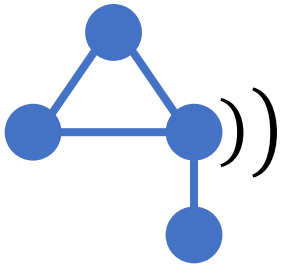
Adversarial attacks are *typically* approximate:
Upper bound

Certificates *typically* require some relaxation:
Lower bound

Determining the **true robustness** is *typically* intractable



Perspectives on Adversarial Robustness

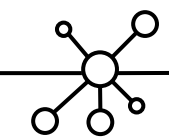
$\exists$  $\in \mathcal{B}($  $)$ s.t. $\text{argmax}(f_\theta($  $)) \neq \text{argmax}(f_\theta($  $))$

Assessing Robustness

- Upper bound: Attacks
- Lower bound: Certification
  (will not be covered today)

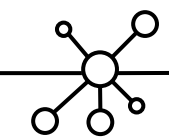
Improving Robustness

- Invariant model architectures
- Enhanced model architecture
- Robust/adversarial training



Outline

1. **Adversarial Robustness**
2. Adversarial Attacks (Assessing Robustness)
   a. Application: Combinatorial Optimization
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. Improving Robustness
   a. Invariant model architectures
   b. Enhanced model architectures
   c. Robust/adversarial training

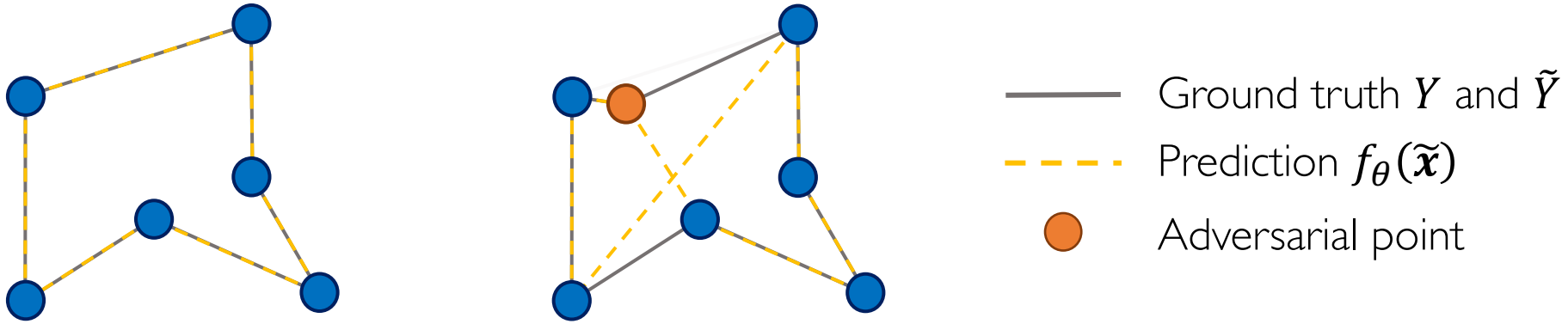


Outline

1. Adversarial Robustness
2. **Adversarial Attacks (Assessing Robustness)**
   a. **Application: Combinatorial Optimization**
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. Improving Robustness
   a. Invariant model architectures
   b. Enhanced model architectures
   c. Robust/adversarial training

Combinatorial Optimization: TSP

From a problem instance, find a perturbed problem instance $\tilde{\boldsymbol{x}}$ that maximizes the surrogate loss $\mathrm{L}\left(f_\theta(\tilde{\boldsymbol{x}}), \tilde{Y}\right)$.



Original problem instance $\boldsymbol{x}$ Perturbed problem instance $\tilde{\boldsymbol{x}}$

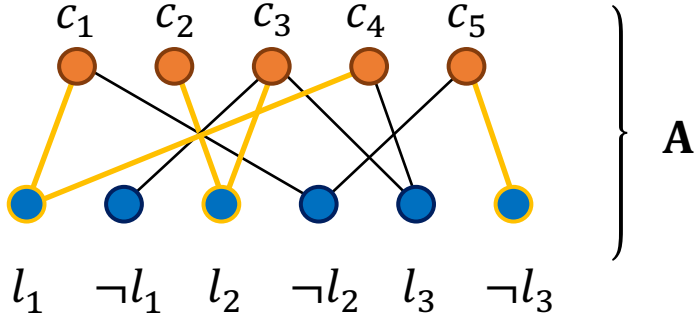
→ The admissible perturbations are linked to the semantics

Geisler et al. "Generalization of Neural Combinatorial Solvers Through the Lens of Adversarial Robustness." ICLR 2022.

SAT Perturbation Model

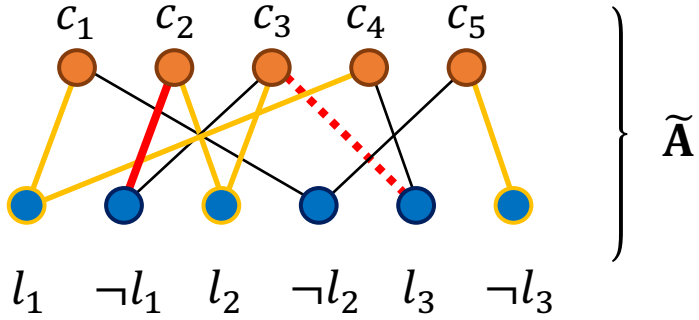
$$\overbrace{(l_1 \vee \neg l_2)}^{c_1} \wedge \overbrace{(l_2)}^{c_2} \wedge \overbrace{(\neg l_1 \vee l_2 \vee l_3)}^{c_3} \wedge \overbrace{(l_1 \vee l_3)}^{c_4} \wedge \overbrace{(\neg l_2 \vee \neg l_3)}^{c_5}$$

$$= \overbrace{(1 \vee 0)}^{c_1} \wedge \overbrace{(1)}^{c_2} \wedge \overbrace{(0 \vee 1 \vee 0)}^{c_3} \wedge \overbrace{(1 \vee 0)}^{c_4} \wedge \overbrace{(0 \vee 1)}^{c_5} = 1$$



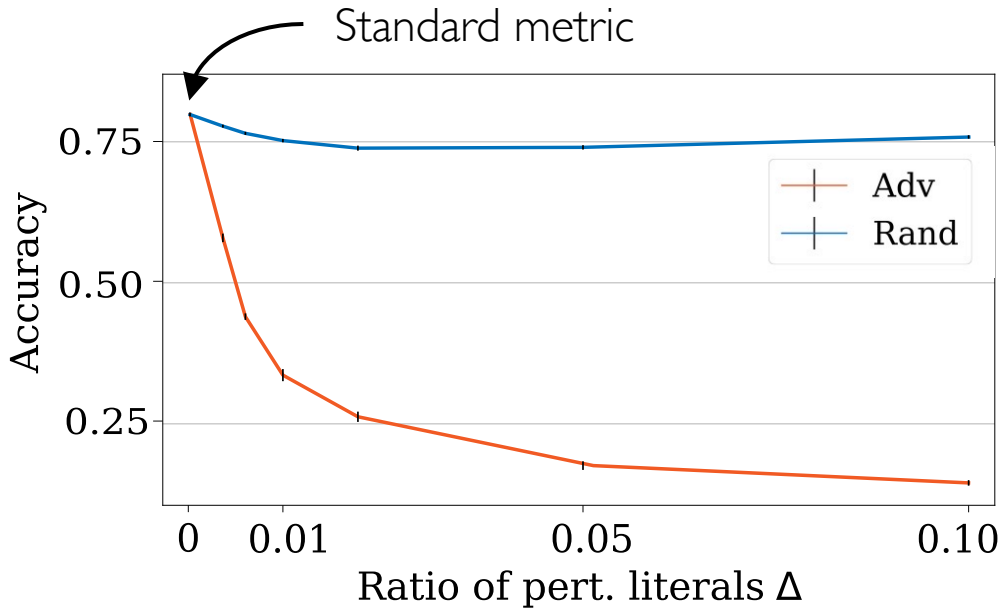
Invariance: we can insert/remove literals as long as a literal of truth assigment $\boldsymbol{Y}$ remains in each clause

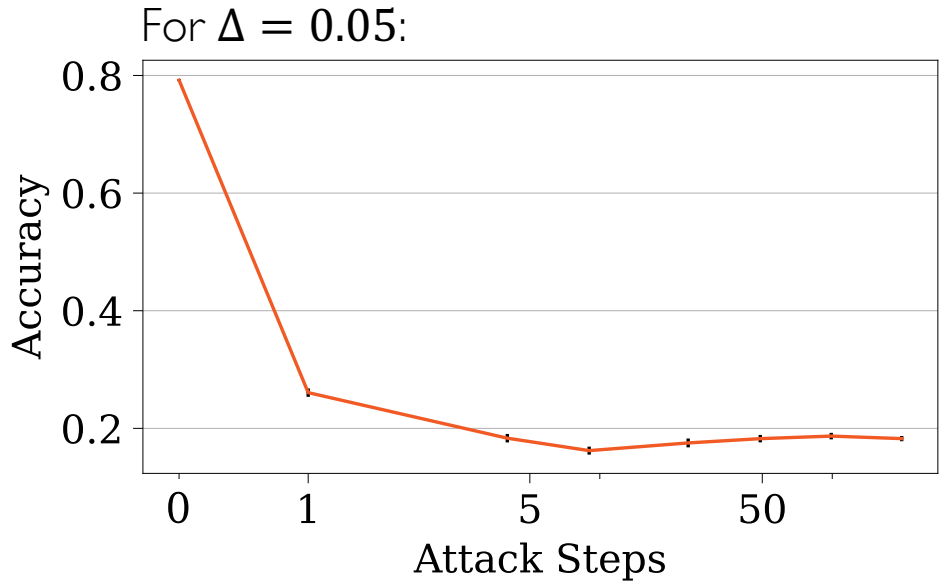
$$\overbrace{(l_1 \vee \neg l_2)}^{c_1} \wedge \overbrace{\underbrace{\neg l_1 \vee l_2}_{\text{insert}}}^{c_2} \wedge \overbrace{(\neg l_1 \vee l_2 \quad \underbrace{\cancel{\vee l_3}}_{\text{remove}})}^{c_3} \wedge \overbrace{(l_1 \vee l_3)}^{c_4} \wedge \overbrace{(\neg l_2 \vee \neg l_3)}^{c_5}$$



Geisler et al. "Generalization of Neural Combinatorial Solvers Through the Lens of Adversarial Robustness." ICLR 2022.

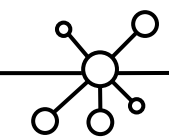
Results (SAT)





Changing approx. 0.5% of the literals suffices to push the accuracy below 50%

Geisler et al. "Generalization of Neural Combinatorial Solvers Through the Lens of Adversarial Robustness." ICLR 2022.

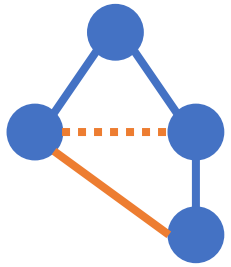
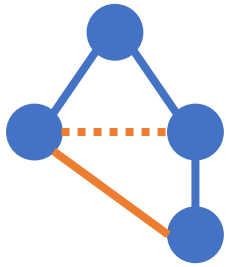
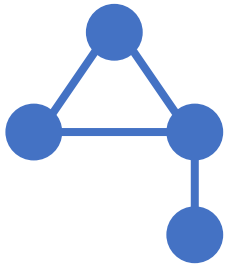


Outline

1. Adversarial Robustness
2. **Adversarial Attacks (Assessing Robustness)**
   a. Application: Combinatorial Optimization
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. Improving Robustness
   a. Enhanced model architectures
   b. Invariant model architectures
   c. Robust/adversarial training

Adversarial Attack for Structure Perturbations

Adversarial attack on a fixed GNN f_θ with loss $\mathcal{L}$ and budget Δ:

$$\max_{\text{  }} \mathcal{L}\big(f_\theta(\ \text{  }\ )\big) \quad \text{s.t.} \quad \| \ \text{  } \ - \ \text{  } \ \|_0 \leq \Delta$$

- Results in a discrete and non-convex optimization problem
- A graph with n nodes has n^2 possible edges

→ ***Projected Randomized Block Coordinate Descent (PR-BCD)*** *(complexity $O(\Delta)$)*

Adversarial Attacks on Graph Neural Networks

$\mathbf{A} \oplus \mathbf{P}, \mathbf{X}$ $\quad$ $\mathbf{A} \oplus \mathbf{P}$ $\quad$ $\mathbf{A} \in \{0,1\}^{n \times n}$ $\quad$ $\oplus$: **XOR**

$$\max \; \mathcal{L}\big(f_\theta(\;)\big) \;\text{ s.t. }\; \| \;\; - \;\; \|_0 \leq \Delta$$

① $\quad$ ②: $\sum_{i,j} \mathbf{P}_{i,j} \leq \Delta$

$$\max_{\mathbf{P}} \mathcal{L}(f_\theta(\mathbf{A} \oplus \mathbf{P}, \mathbf{X})) \;\text{ s.t. }\; \sum_{i,j} \mathbf{P}_{i,j} \leq \Delta \;\wedge\; \mathbf{P} \in \{0,1\}^{n \times n} \;\wedge\; \mathbf{P} = \mathbf{P}^{\mathrm{T}}$$

Details: keep binary if undirected

Adversarial Attacks on Graph Neural Networks

$$\max_{\mathbf{P}} \mathcal{L}(f_\theta(\mathbf{A} \oplus \mathbf{P}, \mathbf{X})) \text{ s.t. } \sum \mathbf{P} \leq \Delta \wedge \mathbf{P} \in \{0,1\}^{n \times n} \wedge \mathbf{P} = \mathbf{P}^{\mathrm{T}}$$

For the applicability of **gradient-based optimization**

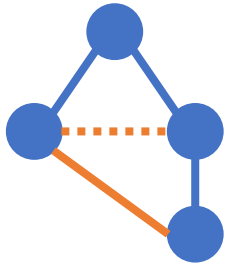
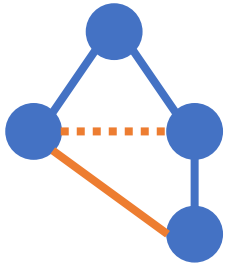
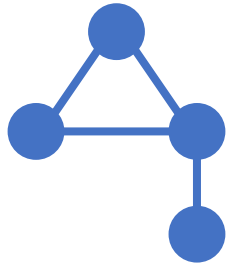
(1) $\mathbf{P} \in \{0,1\}^{n \times n}$ is relaxed to $\mathbf{P} \in [0,1]^{n \times n}$ during the attack

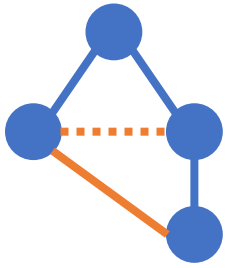
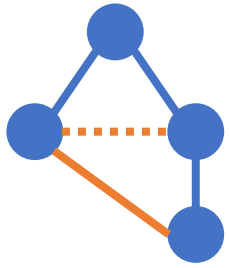
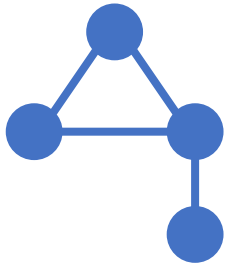
(2) Each entry in $\mathbf{P} \in [0,1]^{n \times n}$ represents the **probability of an edge flip**

(3) After the attack, we sample from $\mathbf{P} \in [0,1]^{n \times n}$ to obtain $\mathbf{P} \in \{0,1\}^{n \times n}$

Assumption: model can handle edge weights

Alternative Argument for the Relaxation

$$\max_{\text{  }} \mathcal{L}\left(f_\theta\left(\text{  }\right)\right) \quad \text{s.t.} \quad \left\| \text{  } - \text{  } \right\|_0 \leq \Delta$$

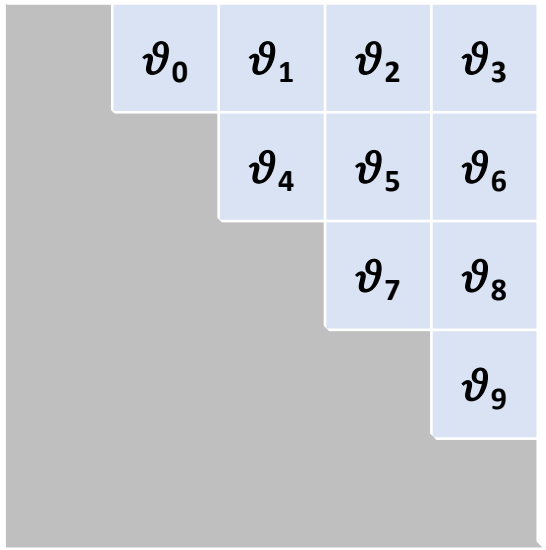
$$\approx \max_{\text{  }} \mathcal{L}\left(f_\theta\left(\text{  }\right)\right) \quad \text{s.t.} \quad \left\| \text{  } - \text{  } \right\|_1 \leq \Delta$$

→ Effectively relaxes the constraint from $\|\cdot\|_0$ to $\|\cdot\|_1$

Adversarial Attacks on Graph Neural Networks

$$\max_{\mathbf{P}} \mathcal{L}(f_\theta(\mathbf{A} \oplus \mathbf{P}, \mathbf{X})) \quad \text{s.t.} \quad \sum \mathbf{P} \leq \Delta \;\wedge\; \mathbf{P} \in \{0,1\}^{n \times n} \;\wedge\; \mathbf{P} = \mathbf{P}^{\mathrm{T}}$$

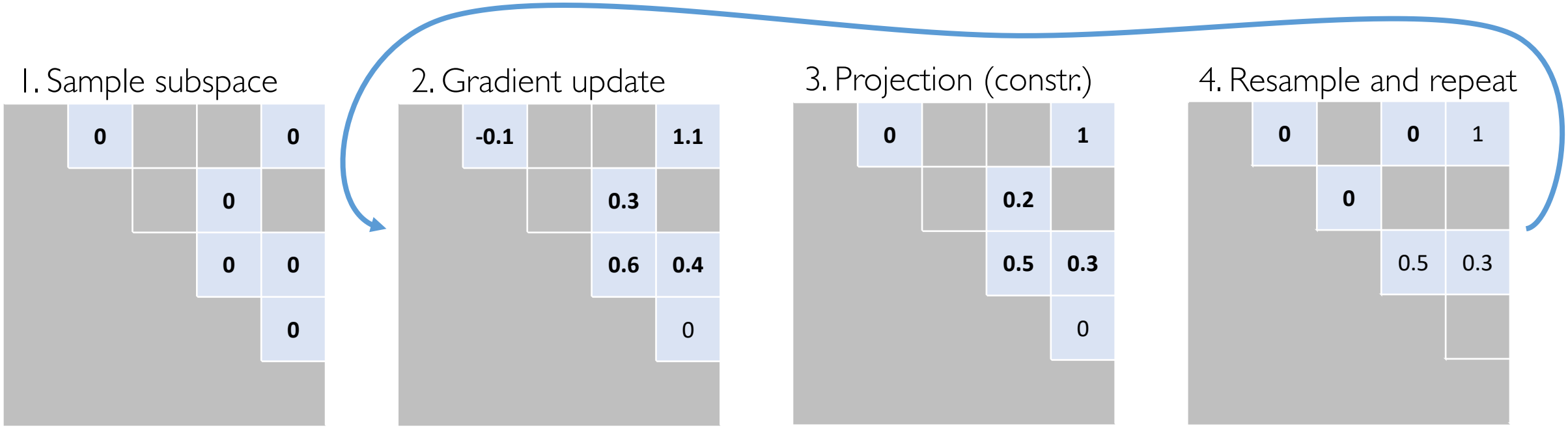
$$\mathbf{P} =$$



⚡ Number of "parameters" scales with $\mathcal{O}(n^2)$ with number of nodes n.

Our Proposed Attack

Our variant of **Projected Randomized Block Coordinate Descent (PR-BCD)** that maintains a *sparse parameter space* throughout the optimization

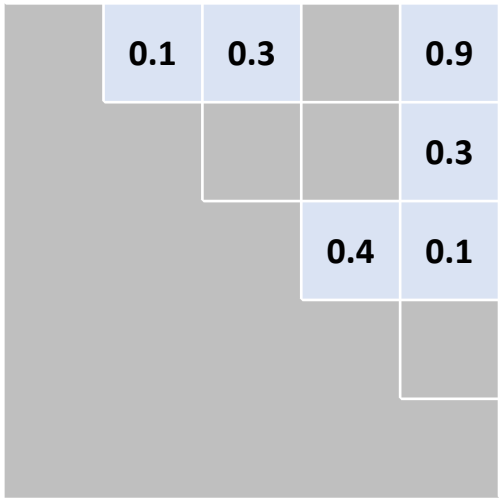


$$\max_{\mathbf{P}} \mathcal{L}(f_\theta(\mathbf{A} \oplus \mathbf{P}, \mathbf{X})) \quad \text{s.t.} \quad \sum \mathbf{P} \leq \Delta \ \wedge \ \mathbf{P} \in \{0,1\}^{n \times n}$$

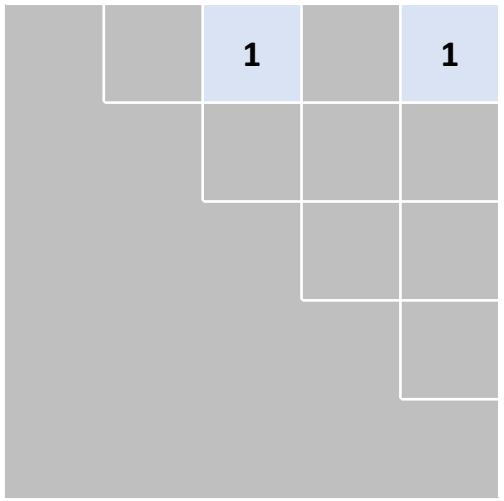
Example: Undirected Graph with $n = 5$ nodes and budget $\Delta = 2$

Our Proposed Attack

Our variant of **Projected Randomized Block Coordinate Descent (PR-BCD)** that maintains a *sparse parameter space* throughout the optimization



Sampling of final **discrete** perturbations



$$\max_{\mathbf{P}} \mathcal{L}(f_\theta(\mathbf{A} \oplus \mathbf{P}, \mathbf{X})) \quad \text{s.t.} \quad \sum \mathbf{P} \leq \Delta \ \wedge \ \mathbf{P} \in \{0,1\}^{n \times n}$$

Example: Undirected Graph with $n = 5$ nodes and budget $\Delta = 2$

Scalable Attacks for Structure Perturbations

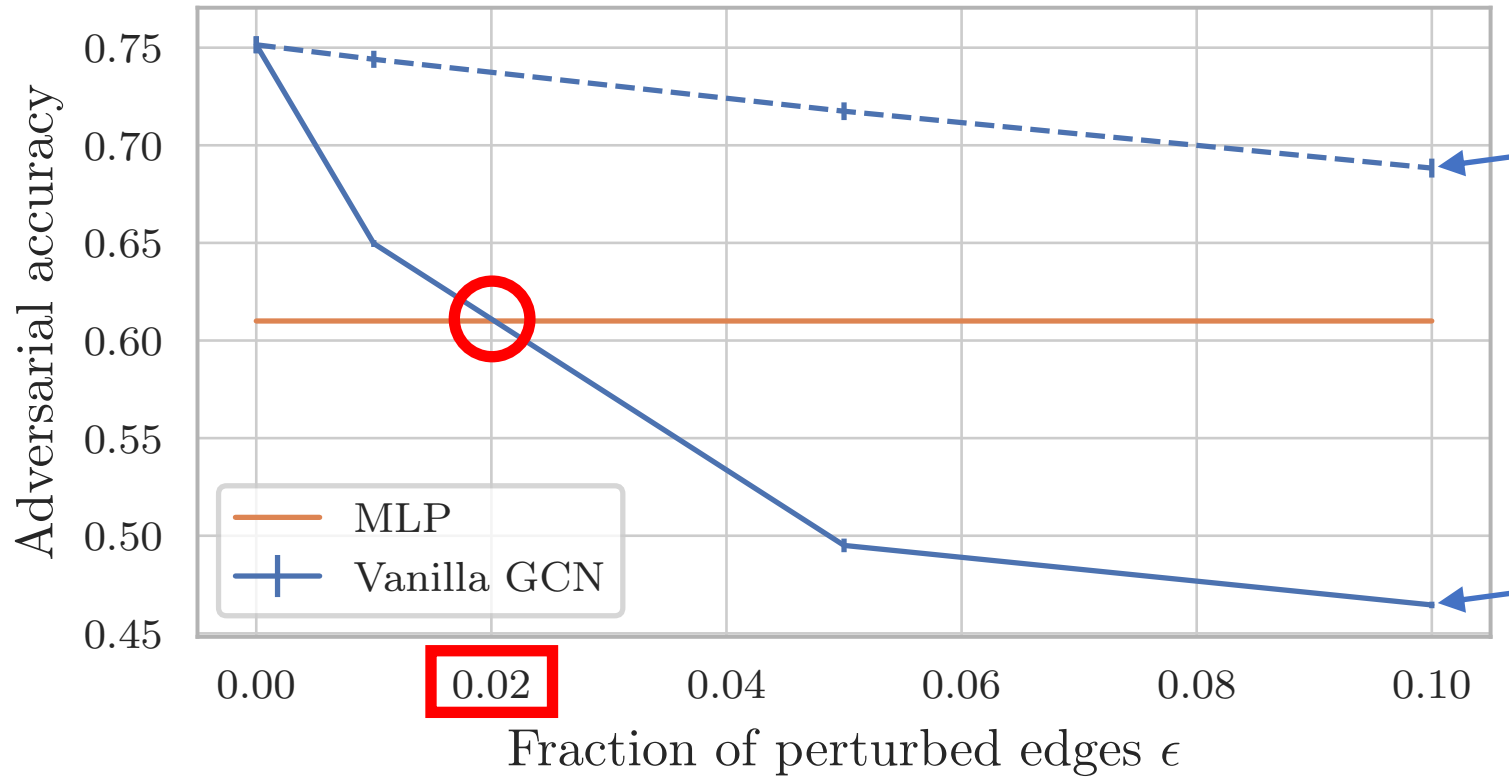
Dataset	# Nodes $\mathbf{n}$	Size (dense)	Size (sparse)	
Citeseer	2.1k	17.8MB	146.7kB	Previous work
Cora ML	2.8k	31.9MB	319.2kB	
PubMed	19.7k	1.6GB	1.8MB	
arXiv *	169.3k	114.7GB	23.3MB	Ours (first-order attack)
Products	2.5M	24.0TB	2.5GB	
Papers	111.1M	49.3PB	32.31GB	

→ *We are the first to study adversarial robustness on such massive graphs.*

** There are a few exceptions that study adversarial robustness for some special cases on graphs with similar size*

Q1: Are GNNs robust if applied to large graphs?

Adversarial attack on GCN on Products (2.5 mio. nodes, 124 mio. edges):



Random DICE attack

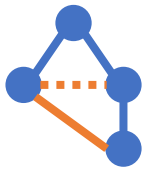
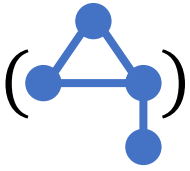
Our **PR-BCD** with novel surrogate loss

PR-BCD is available in PyTorch Geometric

```python
from torch_geometric.contrib.nn import PRBCDAttack

prbcd = PRBCDAttack(gnn, block_size=250_000)
prbcd.attack(data.x, data.edge_index, data.y,
             budget=global_budget, idx_attack=data.test_mask)
```

Local Constraints

What are reasonable / semantic preserving / … perturbations  $\in \mathcal{B}($  $)$?

→ *Open question and most certainly application specific*

It is reasonable to design flexible attacks, e.g., also supporting local constraints:

Global constraint $\sum_{i,j} \mathbf{P}_{i,j} \leq \Delta^{(g)}$ and **local constraints*** $\sum_j \mathbf{P}_{i,j} \leq \Delta_i^{(l)}$

In *Geisler et al. "Adversarial Training for Graph Neural Networks" arXiv 2023* we extend the presented attack to also obey local constraints:

Geisler et al. "Adversarial Training for Graph Neural Networks." In arXiv. 2023.

* with symmetrized $\mathbf{P}$

Poisoning vs. Evasion: Training time vs. test time attacks

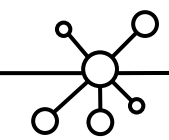
So far, I have only illustrated **test time attacks (evasion)**:

$$\max_{\text{[graph]}} \mathcal{L}\big(f_\theta(\text{[graph]})\big) \text{ s.t. } \| \text{[graph]} - \text{[graph]} \|_0 \leq \Delta$$

Backpropagate through the training

Training time attacks (poisoning):

$$\max_{\text{[graph]}} \mathcal{L}\big(f_{\theta^*}(\text{[graph]})\big) \text{ s.t. } \| \text{[graph]} - \text{[graph]} \|_0 \leq \Delta \wedge \theta^* = \text{Train}(\text{[graph]})$$

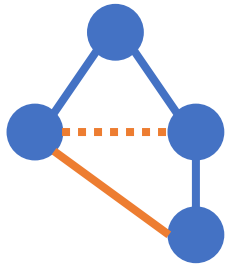
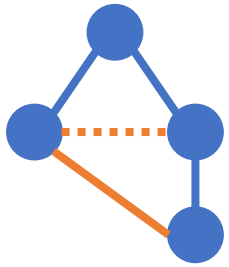
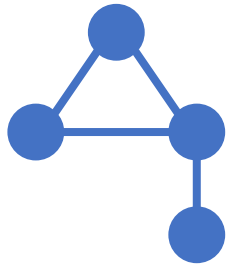


Outline

1. Adversarial Robustness
2. **Adversarial Attacks (Assessing Robustness)**
   a. Application: Combinatorial Optimization
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. Improving Robustness
   a. Enhanced model architectures
   b. Invariant model architectures
   c. Robust/adversarial training

Surrogate Loss

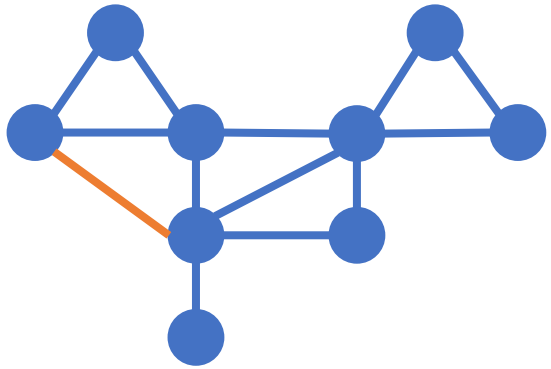
PR-BCD is an efficient first-order optimization method to approximate

$$\max_{\text{  }} \mathcal{L}\big(f_\theta(\text{  })\big) \text{ s.t. } \| \text{  } - \text{  } \|_0 \leq \Delta$$

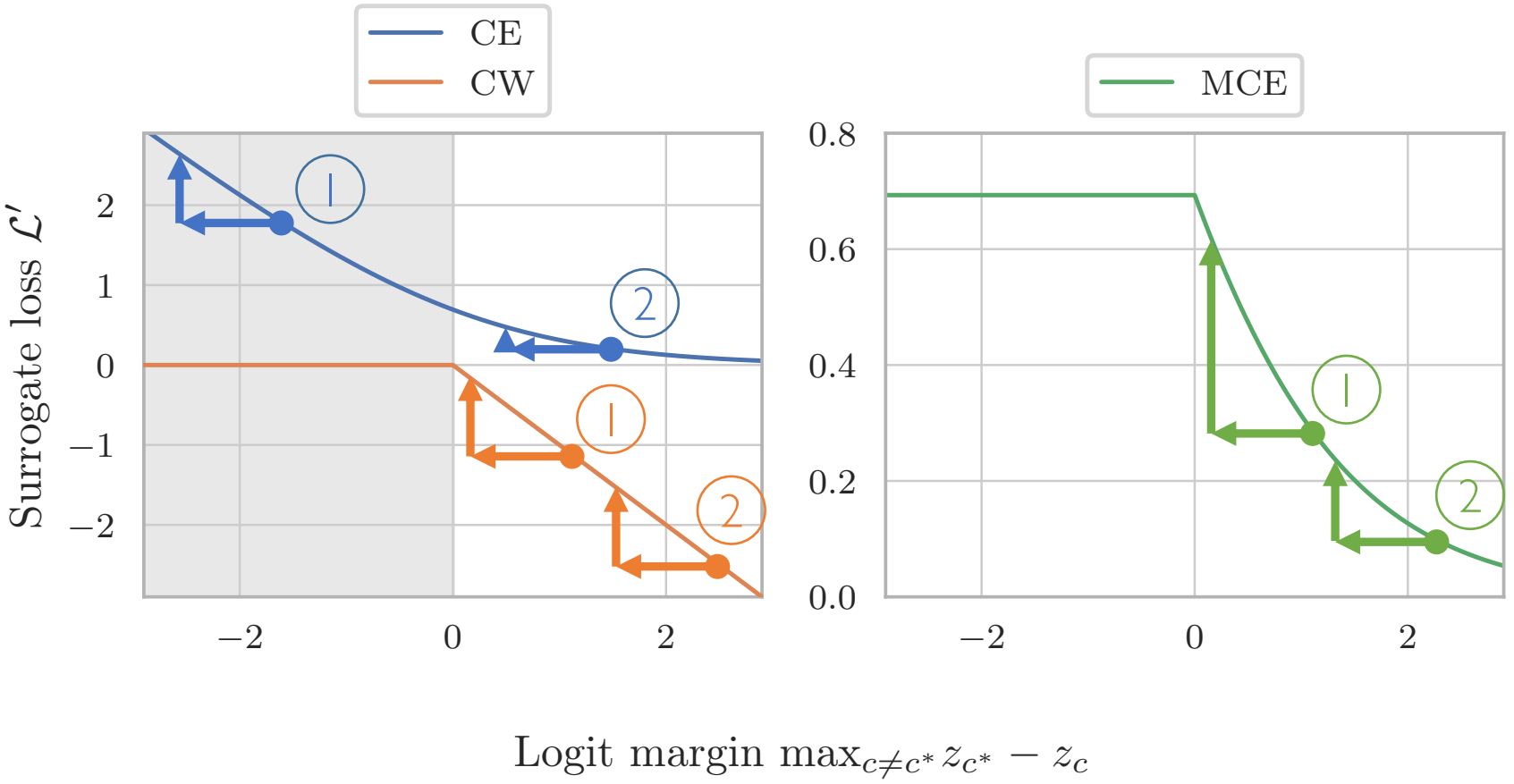
Surrogate losses for interconnected targets

… but do we optimize for a sensible target with common losses?

Surrogate Losses for "Global" Attacks



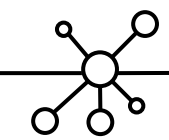
Surrogate Losses for "Global" Attacks (binary)



CE: 1 has a higher "gain" than 2

CW: 1 and 2 have the same "gain"

MCE: 1 has a higher "gain" than 2



Outline

1. Adversarial Robustness
2. Adversarial Attacks (Assessing Robustness)
   a. Application: Combinatorial Optimization
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. **Improving Robustness**
   a. Invariant model architectures
   b. Enhanced model architectures
   c. Robust/adversarial training

How Do Transformers Encode Code?

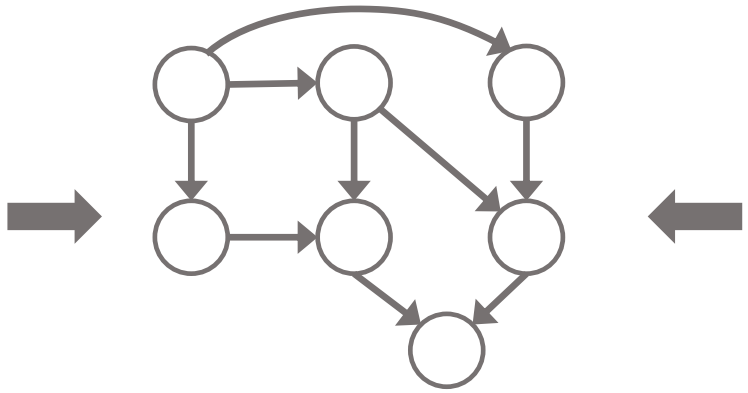
```
def f1_score(pred, label):
  correct = pred == label
  tp = (correct & label).sum()
  fn = (~correct & pred).sum()
  fp = (~correct & ~pred).sum()
  precision = tp / (tp + fp)
  recall = tp / (tp + fn)
  return (
    2 * (recall * precision) /
    (recall * precision)
  )
```

Tokenization:

Motivation for Directed Graphs: Encode symmetries

Note: Our construction via the Abstract Syntax Tree (AST) covers loops, exceptions, etc.

```
def f1_score(pred, label):
  correct = pred == label
  tp = (correct & label).sum()
  fn = (~correct & pred).sum()
  fp = (~correct & ~pred).sum()
  precision = tp / (tp + fp)
  recall = tp / (tp + fn)
  return (
    2 * (recall * precision) /
    (recall + precision)
  )
```

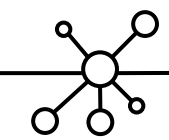


many to one
(here: 4,096 → 1)

```
def f1_score(pred, label):
  correct = pred == label
  tp = (correct & label).sum()
  fn = (~correct & ~pred).sum()
  recall = tp / (tp + fn)
  fp = (~correct & pred).sum()
  precision = tp / (tp + fp)
  return (
    2 * (precision * recall) /
    (precision + recall)
  )
```

→ The model itself and the modelling of the input data greatly impact robustness

Geisler et al. "Transformers Meet Directed Graphs." ICML 2023.



Outline

1. Adversarial Robustness
2. Adversarial Attacks (Assessing Robustness)
   a. Application: Combinatorial Optimization
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. **Improving Robustness**
   a. Invariant model architectures
   b. Enhanced model architectures
   c. Robust/adversarial training

Scalable Robust Message Passing Aggregation

Graph Neural Networks` (GNN) message passing operation:

$$\mathrm{h}_v^{(l)} = \sigma^{(l)}\left(\mathrm{AGGREGATE}^{(l)}\left\{\left(\mathrm{A}_{vu},\ \mathrm{h}_u^{(l-1)}\,\mathrm{W}^{(l)}\right), \forall\, u \in \mathcal{N}(v) \cup v\right\}\right)$$

Use a robust aggregation instead!

E.g., a GCN: $\mathrm{h}_v^{(l)} \approx \sigma^{(l)}\left(\mathrm{MEAN}\left\{\left(\mathrm{A}_{vu},\ \mathrm{h}_u^{(l-1)}\,\mathrm{W}^{(l)}\right), \forall\, u \in \mathcal{N}(v) \cup v\right\}\right)$

Geisler et al. "Robustness of Graph Neural Networks at Scale." NeurIPS 2021.
Geisler et al. "Reliable Graph Neural Networks via Robust Aggregation." NeurIPS 2020.

Geisler et al. "Robustness of Graph Neural Networks at Scale." NeurIPS 2021.
Geisler et al. "Reliable Graph Neural Networks via Robust Aggregation." NeurIPS 2020.

Scalable Robust Message Passing Aggregation

Our Soft Median: $\mu(\mathbf{X}) = \text{softmax}\left(\frac{-\mathbf{d}}{T}\right)^{\top} \mathbf{X} = \mathbf{s}^{\top}\mathbf{X} \approx \text{argmin}_{\mathbf{x}' \in \mathbb{X}} \|\bar{\mathbf{x}} - \mathbf{x}'\|$

Distance to dim.-wise Median: $\mathbf{d}_i = \|\bar{\mathbf{x}} - \mathbf{x}_i\|$

Input Embeddings: $\mathbf{X}, \mathbb{X}$

Original neighbors True mean Adversarial neighbors Mean Soft Median

Geisler et al. "Robustness of Graph Neural Networks at Scale." NeurIPS 2021.
Geisler et al. "Reliable Graph Neural Networks via Robust Aggregation." NeurIPS 2020.

Scalable Robust Message Passing Aggregation

Our Soft Median: $\mu(\mathbf{X}) = \mathbf{softmax}\left(\frac{-\mathbf{d}}{T}\right)^{\top}\mathbf{X} = \mathbf{s}^{\top}\mathbf{X} \approx \mathbf{argmin}_{\mathbf{x}' \in \mathbb{X}} \|\bar{\mathbf{x}} - \mathbf{x}'\|$

Distance to dim.-wise Median: $\mathbf{d}_i = \|\bar{\mathbf{x}} - \mathbf{x}_i\|$

Input Embeddings: $\mathbf{X}, \mathbb{X}$

Instance closest to the dim.-wise Median:

$\mathbf{argmin}_{\mathbf{x}' \in \mathbb{X}} \|\bar{\mathbf{x}} - \mathbf{x}'\|$

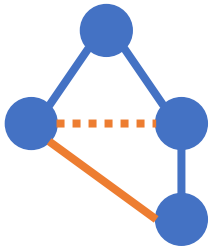
Sample mean:

$\sum_{\mathbf{x}' \in \mathbb{X}} \mathbf{x}'$

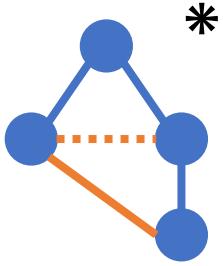
Temperature T

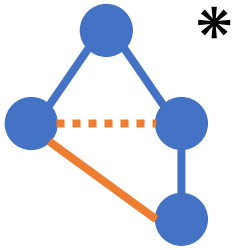
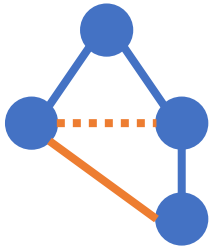
Evaluation Pitfall: Non-Adaptive Attacks

What we should evaluate for model f_θ:

$$\max_{\text{  }} \mathcal{L}\Big(f_\theta\big(\text{  }\big)\Big)$$

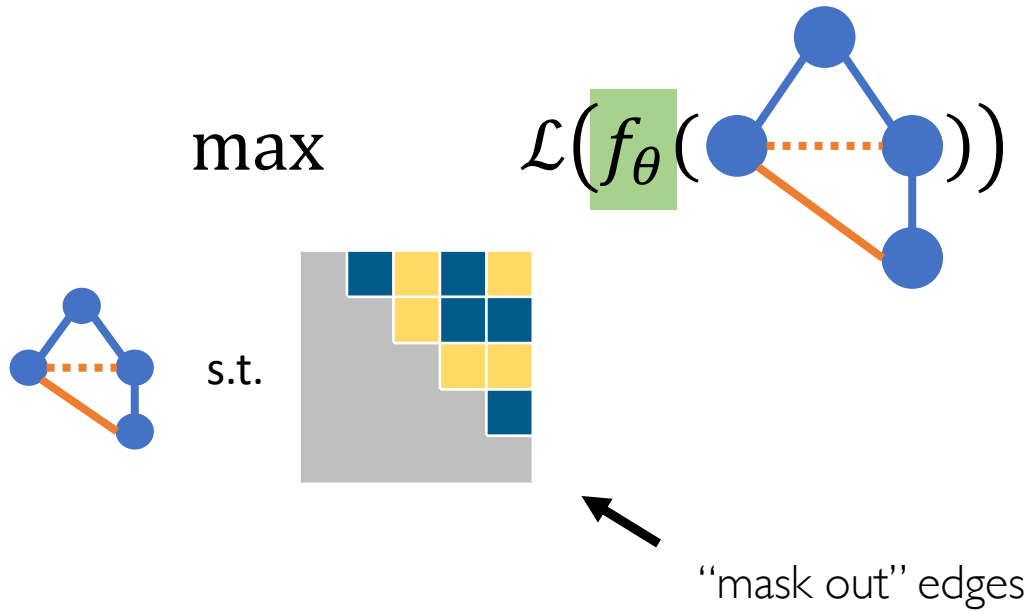
What is being evaluated instead (*with surrogate model* $g_\theta \neq f_\theta$):

$$\mathcal{L}\Big(f_\theta\big(\text{  }^*\big)\Big)$$

$$\text{  }^* = \operatorname{argmax}_{\text{  }} \mathcal{L}\Big(g_\theta\big(\text{  }\big)\Big)$$

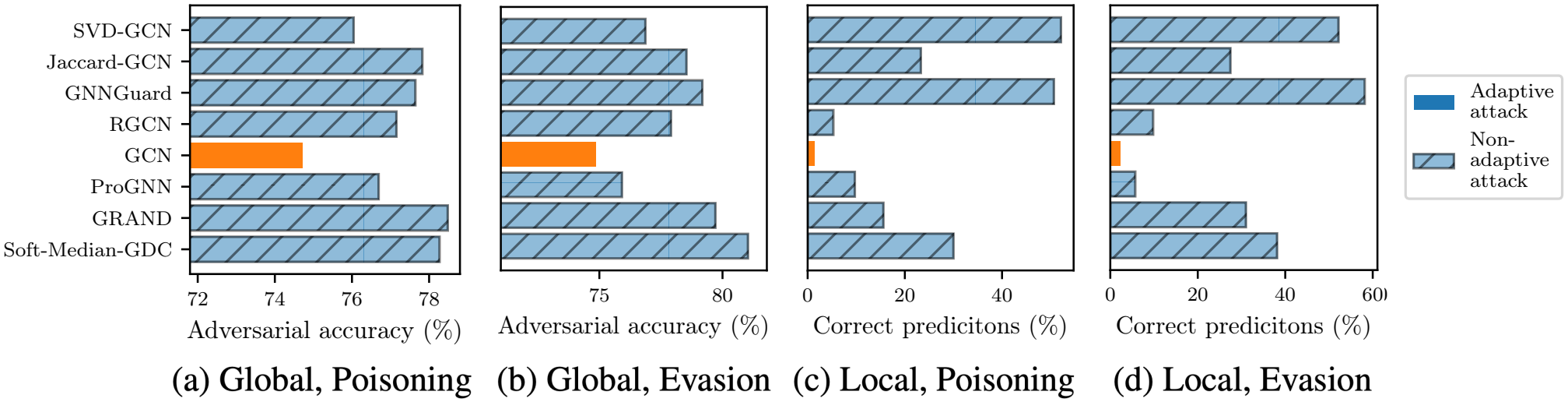
How to Design Adaptive Attacks?

Many defenses f_θ *disallow* certain connections (e.g. Jaccard GCN, SVD GCN)



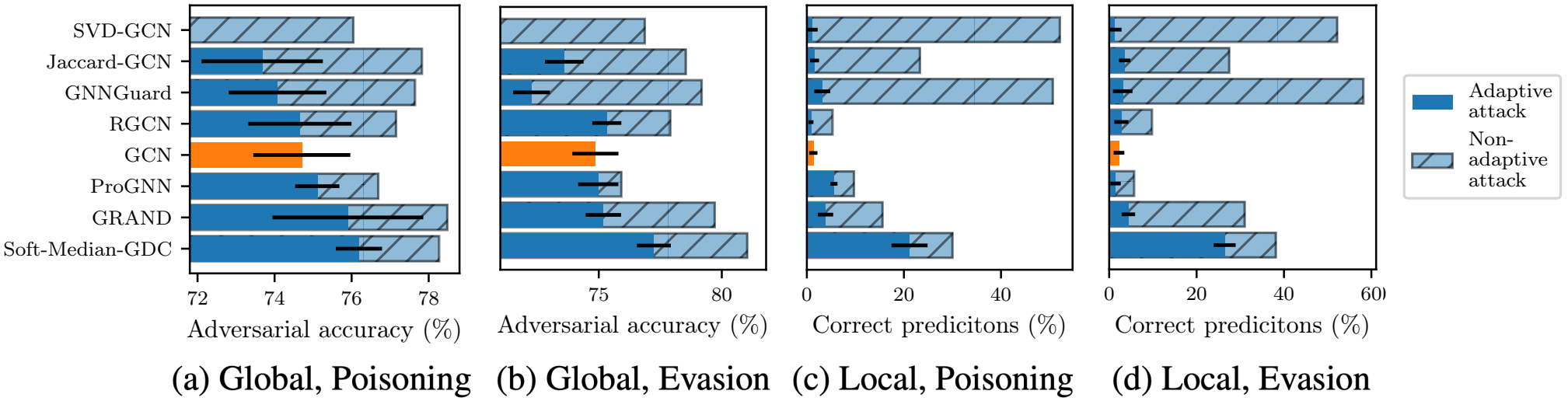
Geisler et al. "Are Defenses for Graph Neural Networks Robust?" NeurIPS 2022.

Adaptive vs. Non-Adaptive Attacks



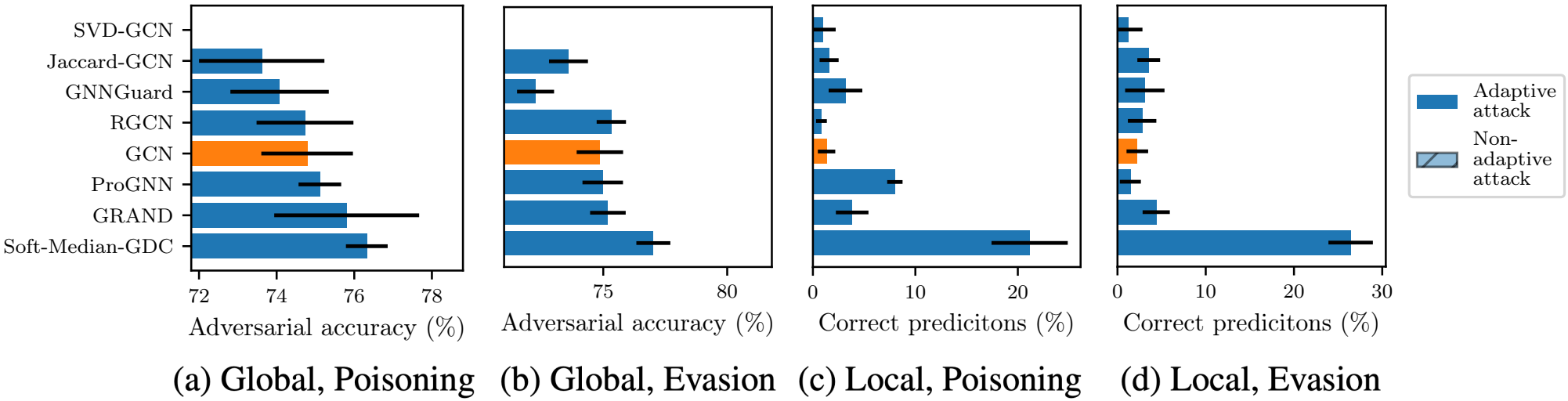
Geisler et al. "Are Defenses for Graph Neural Networks Robust?" NeurIPS 2022.

Adaptive vs. Non-Adaptive Attacks



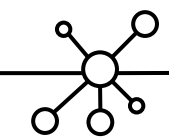
Geisler et al. "Are Defenses for Graph Neural Networks Robust?" NeurIPS 2022.

Adaptive vs. Non-Adaptive Attacks



(a) Global, Poisoning (b) Global, Evasion (c) Local, Poisoning (d) Local, Evasion

→ Our defense motivated by robust statistics seems to be the only effective option



Outline

1. Adversarial Robustness
2. Adversarial Attacks (Assessing Robustness)
   a. Application: Combinatorial Optimization
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. **Improving Robustness**
   a. Invariant model architectures
   b. Enhanced model architectures
   c. Robust/adversarial training

Adversarial Training in the Inductive Setting

We adapt the common benchmark setup for node-level predictions:

- 5% of nodes are used for training
- **80% of nodes are unlabeled**
- 5% of nodes are used for validation
- 10% of nodes are used for test

} **NOT included in training graph**

→ Semi-supervised **inductive learning**

Geisler et al. "Adversarial Training for Graph Neural Networks." In arXiv. 2023.

Adversarial Training in the Inductive Setting

Regular training objective where we sample the graph extended by test nodes and test labels $(\mathcal{G}', y') \sim \mathcal{D}(\mathcal{G}, y)$ with test set $\mathbb{I}_{\mathbf{Test}}$:

$$\min_{\theta} \mathop{\mathbb{E}}_{(\mathcal{G}', y') \sim \mathcal{D}(\mathcal{G}, y)} \sum_{i \in \mathbb{I}_{\mathbf{Test}}} \mathcal{L}(f_\theta(\mathcal{G}')_i, y_i')$$

Adversarial training objective:

$$\min_{\theta} \mathop{\mathbb{E}}_{(\mathcal{G}', y') \sim \mathcal{D}(\mathcal{G}, y)} \max_{\tilde{\mathcal{G}}' \in \mathcal{B}(\mathcal{G}')} \sum_{i \in \mathbb{I}_{\mathbf{Test}}} \mathcal{L}\left(f_\theta(\tilde{\mathcal{G}}')_i, y_i'\right)$$

Geisler et al. "Adversarial Training for Graph Neural Networks." In arXiv. 2023.

Adversarial Training in the Inductive Setting

$$\min_{\theta} \mathop{\mathbb{E}}_{(\mathcal{G}',y')\sim\mathcal{D}(\mathcal{G},y)} \max_{\tilde{\mathcal{G}}'\in\mathcal{B}(\mathcal{G}')} \sum_{i\in\mathbb{I}_{\text{Test}}} \mathcal{L}\left(f_\theta(\tilde{\mathcal{G}}')_i, y_i'\right)$$

$$\approx \min_{\theta} \max_{\tilde{\mathcal{G}}\in\mathcal{B}(\mathcal{G})} \sum_{i\in\mathbb{I}_{\text{Train}}} \mathcal{L}\left(f_\theta(\tilde{\mathcal{G}})_i, y_i\right) + \sum_{i\in\mathbb{I}_{\text{Unlabeled}}} \mathcal{L}\left(f_\theta(\tilde{\mathcal{G}})_i, \hat{y}_i\right)$$

Approximate using adversarial attack

Next: Model

Self-training, where $\hat{y}_i$ are obtained, e.g., with an MLP

Geisler et al. "Adversarial Training for Graph Neural Networks." In arXiv. 2023.

Model Choice for Adversarial Training

We propose to use flexible GNN models that fit in the framework

$$\sum_{k=0}^{K} \gamma_k \boldsymbol{L}^k \boldsymbol{H} = \boldsymbol{V}\mathrm{diag}\big(\hat{g}(\boldsymbol{\Lambda})\big)\overline{\boldsymbol{V}}^T\boldsymbol{H} = \boldsymbol{V}\begin{bmatrix} \hat{g}(\lambda_1) & 0 & \cdots & 0 \\ 0 & \hat{g}(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{g}(\lambda_n) \end{bmatrix}\overline{\boldsymbol{V}}^T\boldsymbol{H}$$

- ✓ A sufficiently flexible message passing scheme (we use $K = 10$)
- ✓ We can interpret the learned message passing $\sum_{k=0}^{K} \gamma_k \boldsymbol{L}^k$

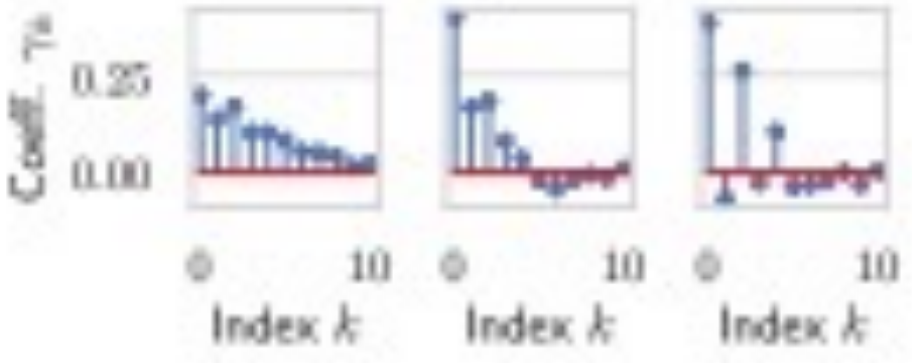
Geisler et al. "Adversarial Training for Graph Neural Networks." In arXiv. 2023.

Robust Message Passing Characteristics

Model: $$\sum_{k=0}^{K} \gamma_k \boldsymbol{L}^k \boldsymbol{H} = \boldsymbol{V}\mathrm{diag}\big(\hat{g}(\boldsymbol{\Lambda})\big)\bar{\boldsymbol{V}}^T\boldsymbol{H}$$

Setup: Citeseer with 20% of edges perturbed

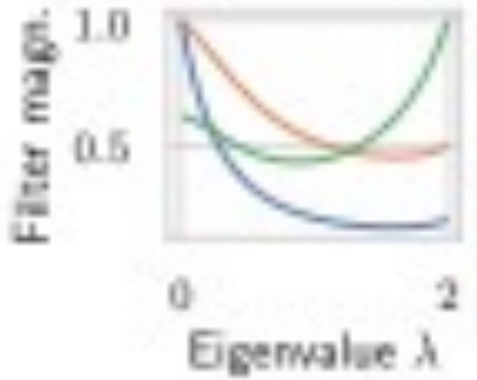
Polynomial interpretation:



Constraints: | w/ local | w/o local

Training: Regular | Adversarial

Spectral filter $\hat{g}(\boldsymbol{\Lambda})$:



Training	Constraints
Regular	
Adversarial	w/ local
Adversarial	w/o local

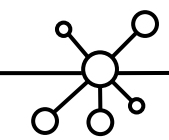
Efficacy of Adversarial Training

Adversarial attacks:
LR-BCD: w/ local constraints
PR-BCD: w/o local constraints

Dataset: Citeseer

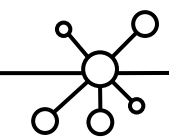
Model	Adv. trn.	A. eval. → A. trn. ↓	Clean	LR-BCD ϵ=0.1	PR-BCD ϵ=0.1	LR-BCD ϵ=0.25	PR-BCD ϵ=0.25	Certifiable accuracy / sparse smoothing Clean	3 add.	5 del.
GCN	✗	-	72.0 ± 2.5	54.7 ± 2.8	51.7 ± 2.8	45.3 ± 3.4	38.0 ± 3.8	38.3 ± 11.5	1.7 ± 0.7	4.8 ± 1.5
GAT	✗	-	-3.6 ± 2.7	-3.9 ± 3.4	+0.5 ± 3.5	-15.9 ± 5.3	-2.3 ± 7.3	-14.0 ± 12.0	-1.7 ± 0.7	-4.8 ± 1.5
APPNP	✗	-	+0.2 ± 1.1	+1.7 ± 0.7	+1.9 ± 1.4	+3.0 ± 1.2	+2.2 ± 2.5	+8.9 ± 9.1	+31.2 ± 6.4	+31.6 ± 6.4
GPRGNN	✗	-	+2.2 ± 4.3	+4.2 ± 2.7	+3.6 ± 4.9	+5.5 ± 3.9	+7.9 ± 4.6	+17.9 ± 6.9	+42.4 ± 4.4	+41.3 ± 3.7
ChebNetII	✗	-	+1.1 ± 2.2	+5.8 ± 2.5	+5.0 ± 2.4	+10.4 ± 2.6	+7.6 ± 3.4	+24.6 ± 9.9	+55.6 ± 1.2	+54.0 ± 0.9
SoftMedian	✗	-	+0.9 ± 1.7	+9.5 ± 2.2	+9.3 ± 1.9	+16.2 ± 2.4	+14.6 ± 2.9	+25.2 ± 10.5	+60.3 ± 1.4	+57.5 ± 0.8
GCN	✓	**LR-BCD**	-0.2 ± 1.2	+7.8 ± 1.6	+5.9 ± 1.5	+10.9 ± 2.1	+8.1 ± 2.3	-3.0 ± 9.4	+10.7 ± 4.6	+13.1 ± 5.1
		PR-BCD	+0.0 ± 1.9	+6.9 ± 0.9	+5.3 ± 1.6	+8.6 ± 2.3	+5.8 ± 2.4	+4.2 ± 14.6	+10.1 ± 5.5	+11.4 ± 4.5
GAT	✓	**LR-BCD**	+0.8 ± 1.6	+5.9 ± 3.7	+9.0 ± 3.0	+8.4 ± 5.1	+13.1 ± 3.5	-2.0 ± 23.0	+1.4 ± 1.7	+4.0 ± 2.1
		PR-BCD	+1.1 ± 2.2	+8.9 ± 2.8	+13.2 ± 3.7	+10.3 ± 2.3	+21.8 ± 4.8	-10.1 ± 13.7	+1.2 ± 2.0	+2.8 ± 1.2
APPNP	✓	**LR-BCD**	-0.8 ± 1.3	+7.6 ± 1.6	+5.6 ± 1.9	+11.1 ± 1.9	+8.3 ± 3.6	+21.7 ± 9.5	+41.3 ± 2.9	+44.1 ± 1.2
		PR-BCD	-0.2 ± 2.2	+6.2 ± 2.3	+5.5 ± 3.1	+9.0 ± 2.9	+6.5 ± 3.8	+19.3 ± 7.2	+41.0 ± 3.8	+41.9 ± 3.0
GPRGNN	✓	**LR-BCD**	+1.7 ± 3.0	**+15.7 ± 3.6**	+13.6 ± 3.5	**+24.8 ± 4.2**	+22.9 ± 4.3	**+34.0 ± 11.5**	+67.4 ± 1.5	+65.0 ± 2.5
		PR-BCD	+0.6 ± 3.6	+15.0 ± 3.6	**+15.9 ± 4.2**	+23.2 ± 4.3	**+26.3 ± 5.4**	+32.9 ± 11.8	**+69.0 ± 1.5**	**+65.9 ± 2.3**
ChebNetII	✓	**LR-BCD**	**+3.7 ± 1.4**	+11.5 ± 1.6	+11.5 ± 1.1	+16.4 ± 1.9	+16.0 ± 2.9	+31.5 ± 12.7	+62.3 ± 1.9	+59.8 ± 2.5
		PR-BCD	+3.4 ± 1.7	+13.2 ± 2.2	+14.3 ± 1.2	+19.5 ± 1.6	+19.8 ± 2.3	+30.7 ± 13.2	+65.4 ± 2.6	+62.9 ± 3.7

Geisler et al. "Adversarial Training for Graph Neural Networks." In arXiv. 2023.



Outline

1. Adversarial Robustness
2. Adversarial Attacks (Assessing Robustness)
   a. Application: Combinatorial Optimization
   b. Scalable Attacks for Structure Perturbations
   c. Loss Functions for Interconnected Predictions: Node Classification
3. Improving Robustness
   a. Invariant model architectures
   b. Enhanced model architectures
   c. Robust/adversarial training



Summary

Email: geisler@in.tum.de

Twitter/X: @geisler_si

Covered works / references:

Geisler et al. "Adversarial Training for Graph Neural Networks." arXiv 2023.

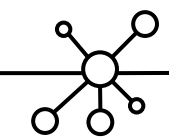
Geisler et al. "Transformers Meet Directed Graphs." ICML 2023.

Geisler et al. "Are Defenses for Graph Neural Networks Robust?" NeurIPS 2022.

Geisler et al. "Generalization of Neural Combinatorial Solvers Through the Lens of Adversarial Robustness." ICLR 2022.

Geisler et al. "Robustness of Graph Neural Networks at Scale." NeurIPS 2021.

Geisler et al. "Reliable Graph Neural Networks via Robust Aggregation." NeurIPS 2020.



Outline

1. Adversarial Robustness
2. Graph Neural Networks
3. **Assessing Robustness**
   a. **Certification**
   b. Adversarial Attacks
4. Improving Robustness
   a. Enhanced model architecture
   b. Invariant model architectures
   c. Robust/adversarial training

Classes of Certificates

Strategy I:

"White-box" certificates

+ Take advantage of the problem structure for accurate certificates, i.e. tight lower bounds
– Special derivation for every model required, i.e. expensive development

Strategy II:

"Black-box" certificates

+ Can be applied to any classifier for discrete data (including all GNNs) out of the box
– Bounds might be less tight since model structure not exploited

Is there something in between?

Strategy III: **"Gray-box" certificates**

Zügner et al. Certifiable Robustness of Graph Convolutional Networks under Structure Perturbations. KDD 2020
Bojchevski at al. Certifiable Robustness to Graph Perturbations. NeurIPS 2019
Zügner et al. Certifiable Robustness and Robust Training for Graph Convolutional Networks. KDD 2019

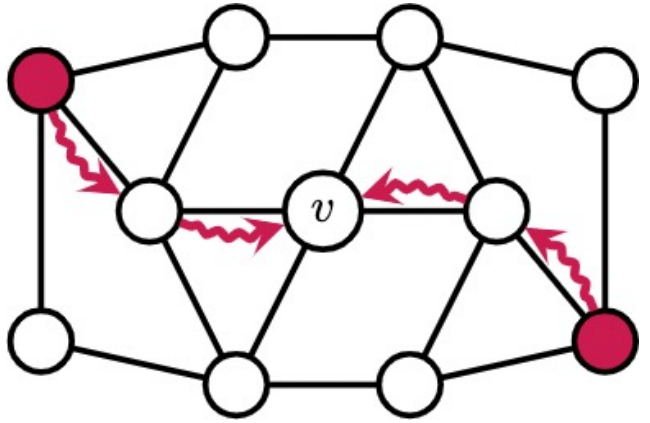
Schuchardt et al. Localized Randomized Smoothing for Collective Robustness Certification. ICLR 2023
Schuchardt et al. Collective Robustness Certificates: Exploiting Interdependence in GNNs. ICLR 2021
Bojchevski et al. Efficient Robustness Certificates for Discrete Data. ICML 2020

Idea: Gray-Box Certificates

- Enhance black-box by general model properties
  - message-passing princple
  - invariances
  - …

Adversarial node

Adversarial message



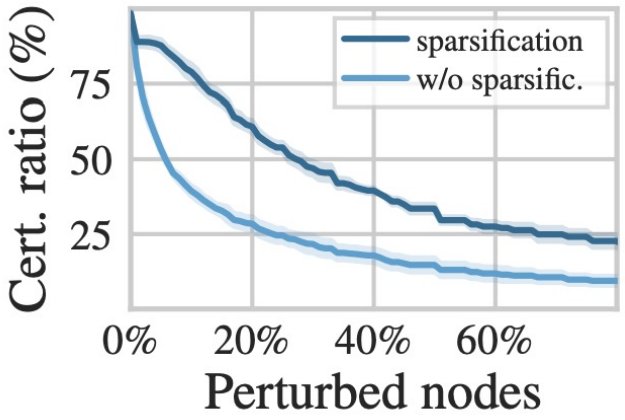
Class A
→ Class B

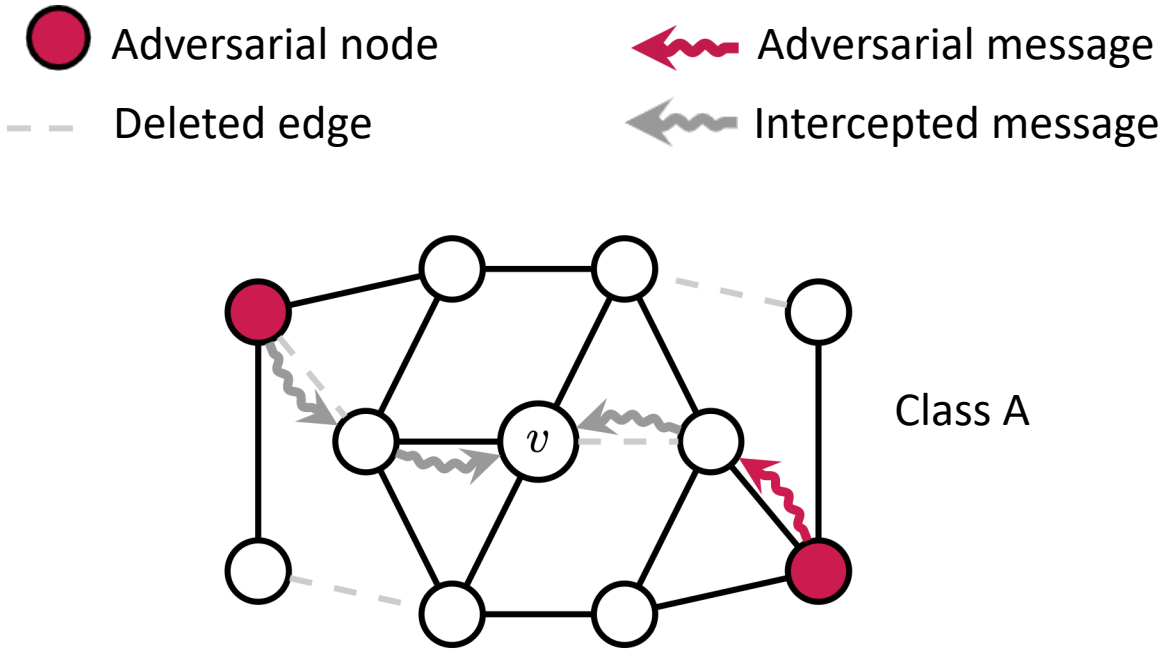
Schuchardt, Günnemann. Invariance-Aware Randomized Smoothing Certificates. NeurIPS 2022
Scholten, Schuchardt, Geisler, Bojchevski, Günnemann. Randomized Message-Interception Smoothing: Gray-box Certificates for Graph Neural Networks. NeurIPS 2022

Idea: Gray-Box Certificates

- Enhance black-box by general model properties
  - message-passing princple
  - invariances
  - …

Smoothed GAT on Cora-ML

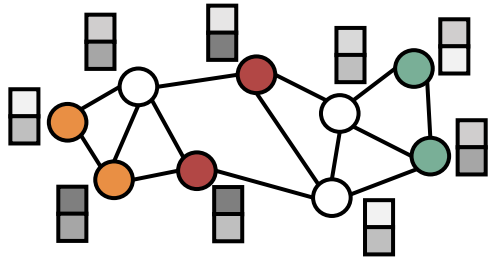




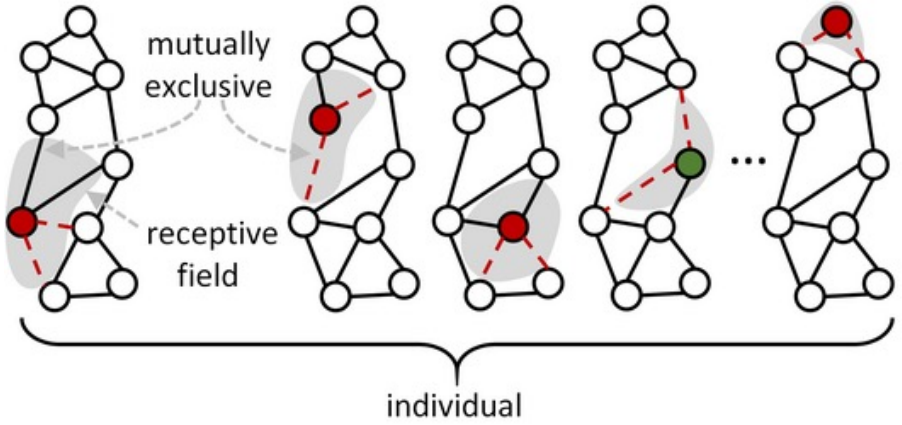
Schuchardt, Günnemann. Invariance-Aware Randomized Smoothing Certificates. NeurIPS 2022
Scholten, Schuchardt, Geisler, Bojchevski, Günnemann. Randomized Message-Interception Smoothing: Gray-box Certificates for Graph Neural Networks. NeurIPS 2022

Individual vs. Collective Certificates

- Often: classifier that simultaneously outputs multiple predictions give a single input
  - node classification, image segmentation, named-entity recognition
- Problem: "Standard" certificates care about a single prediction only



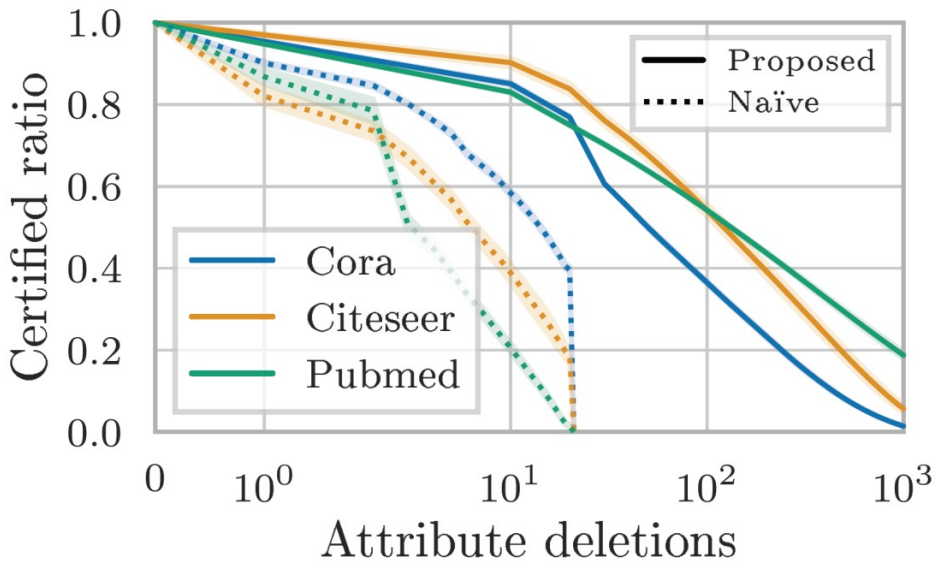
- Collective robustness certificate → number of predictions that are simultaneously guaranteed to remain stable

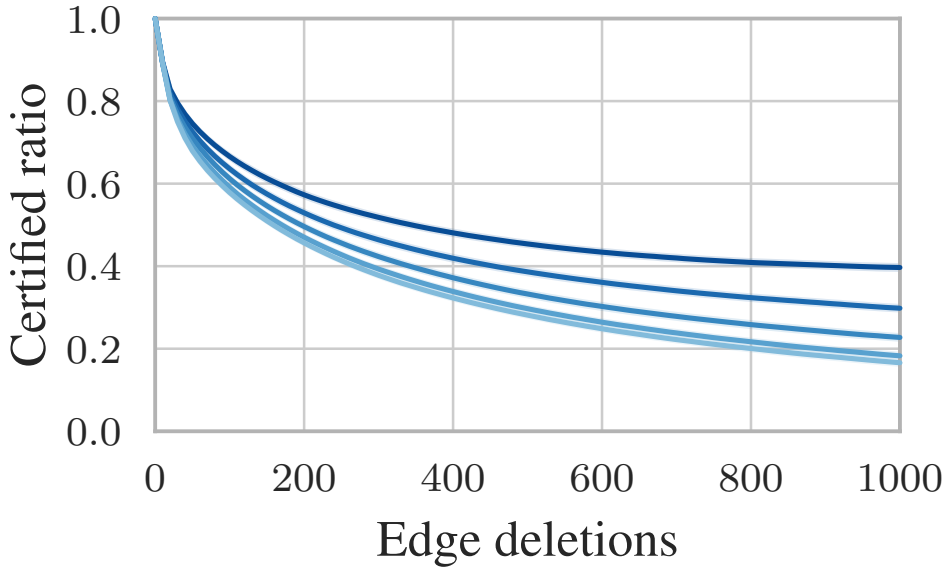


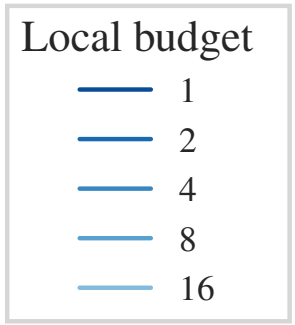
Schuchardt et al. Collective Robustness Certificates: Exploiting Interdependence in GNNs. ICLR 2021

Results

Task: Node-Level Classification



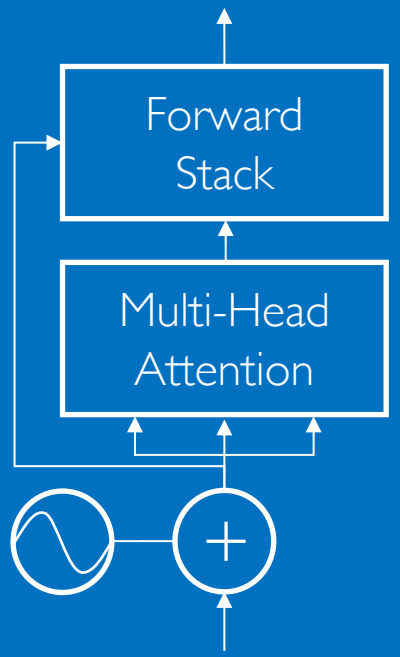




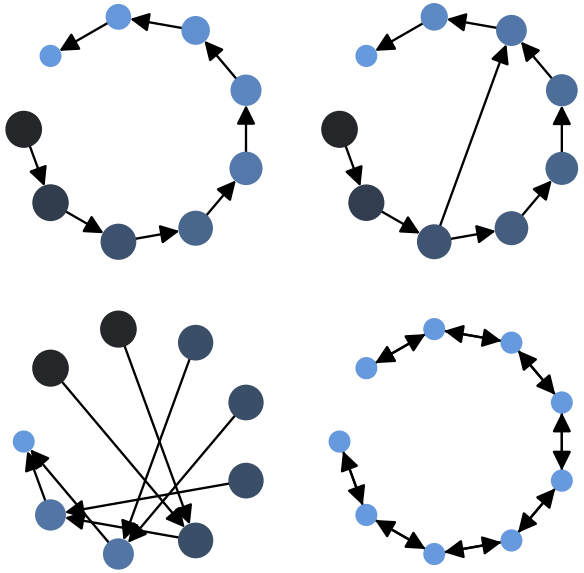
Cora-ML data (~3k nodes, ~8k edges) and smoothed GCN model

Schuchardt et al. Collective Robustness Certificates: Exploiting Interdependence in GNNs. ICLR 2021

Transformers Meet



Directed Graphs



Simon Geisler[1], Yujia Li[2],
Daniel Mankowitz[2], Ali Taylan Cemgil[2], Stephan Günnemann[1], Cosmin Paduraru[2]

1 Technical University of Munich 2 DeepMind

ICML 2023
& Deep Learning on Graphs Workshop @ AAAI 2023

Transformers Meet Directed Graphs

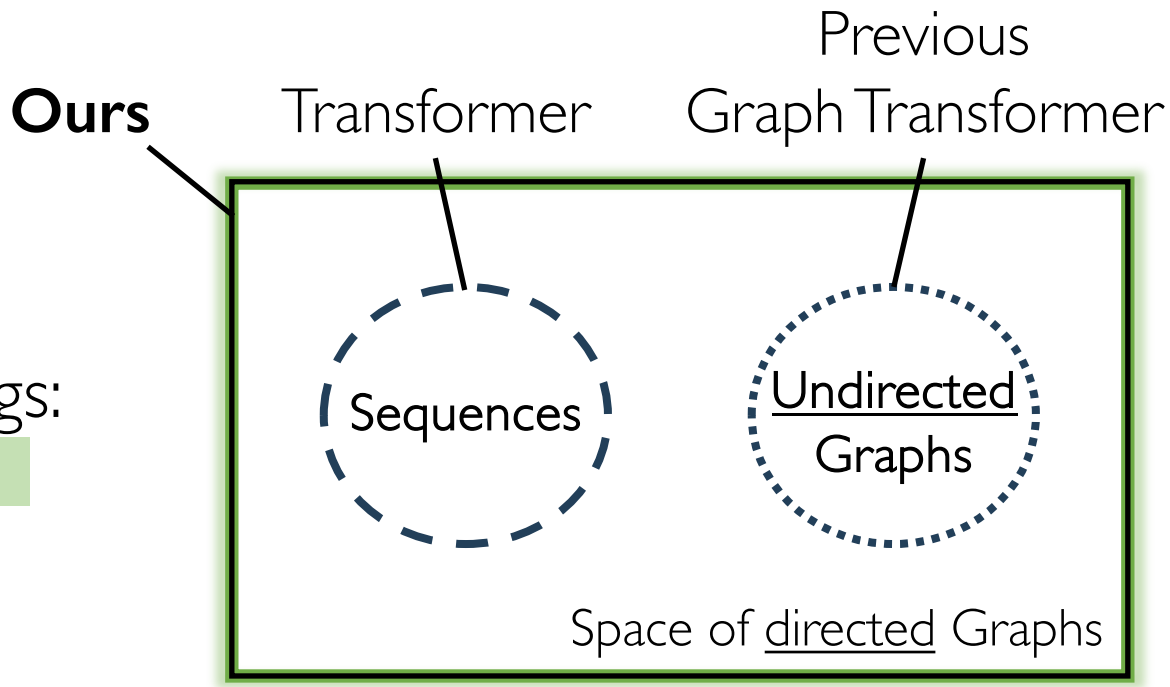
TL;DR.

1. How to generalize transformers to directed graphs?

→ Direction-aware positional encodings:

a) Spectral encodings: Magnetic Laplacian

b) Directional random walks



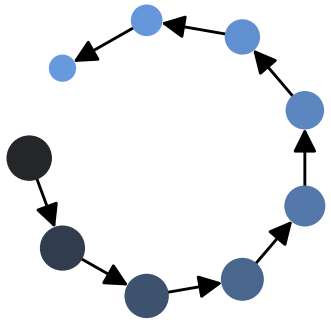
2. Directed graphs are important for many applications

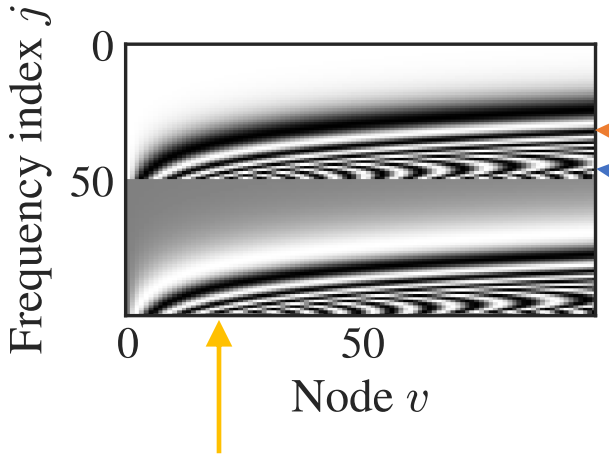
→ Appropriate modeling of input modalities:

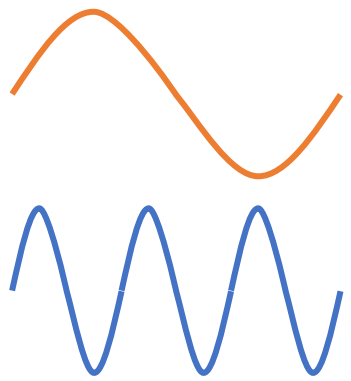
a) If ignoring the direction, the model might sacrifice expressivity

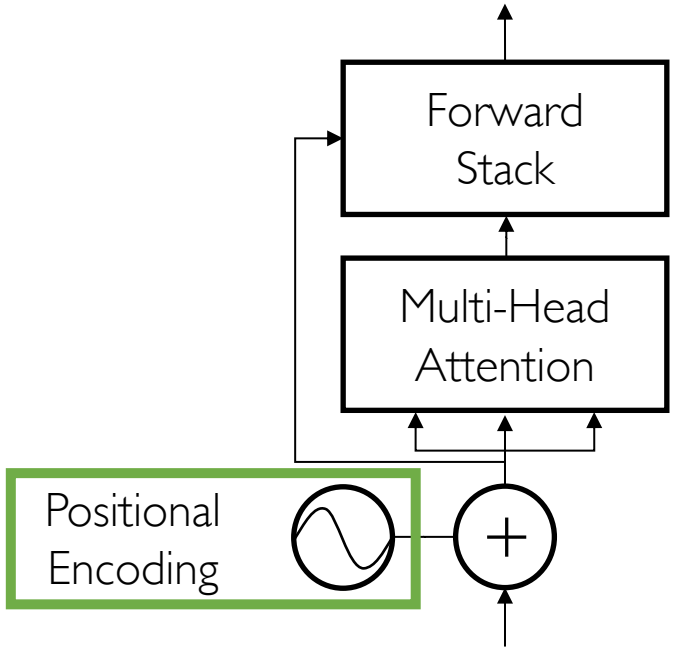
b) We can leverage symmetries in the input domain

Sinusoidal Encodings

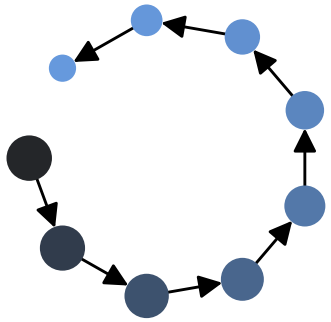


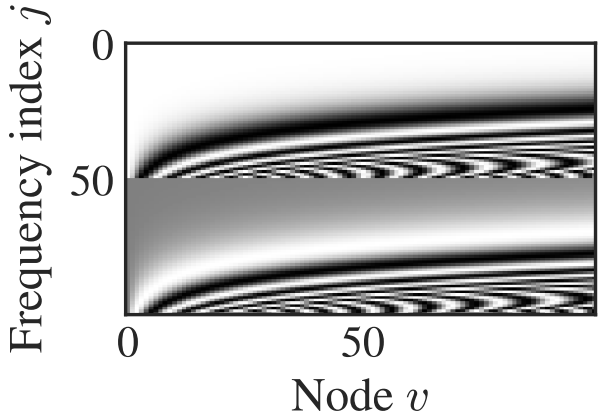






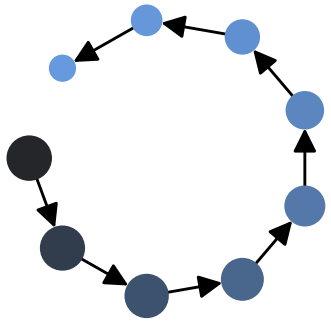
Sinusoidal Encodings

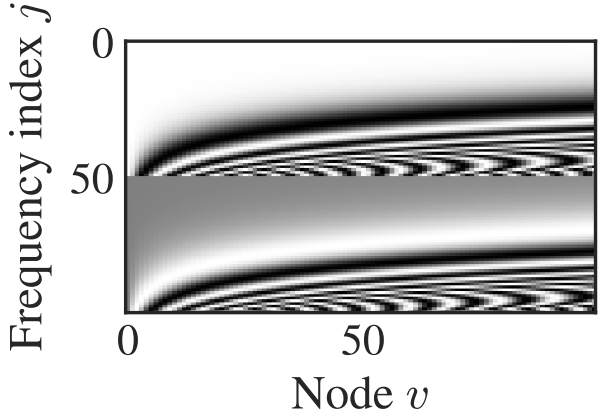




Sinusoidal encodings → related to → Discrete Fourier Transformation

Signal Processing: DFT





$\mathbf{d}$	- # of features/ embedding dim.
$2\pi 10{,}000$	- largest wave length
$\mathbf{v}$	- node index/position
$\mathbf{j}$	- frequency index
x	- signal (spatial domain)
$\hat{x}$	- signal (spectral domain)
n	- sequence length

- Sinusoidal Positional Encodings $\mathbf{PE}^{(\sin)} \in \mathbb{R}^{n \times d}$

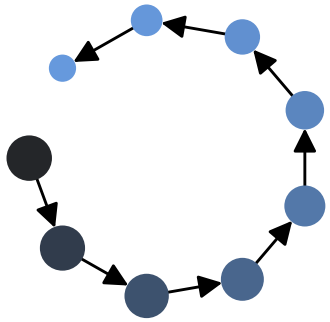
$$\mathrm{PE}^{(\sin)}_{v,j+d/2} = \sin\left(v \cdot 1/10{,}000^{2j/d}\right)$$

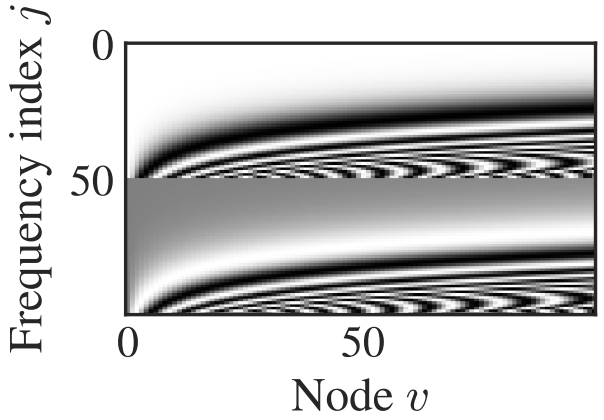
$$\mathrm{PE}^{(\sin)}_{v,j} = \cos\left(v \cdot 1/10{,}000^{2j/d}\right)$$

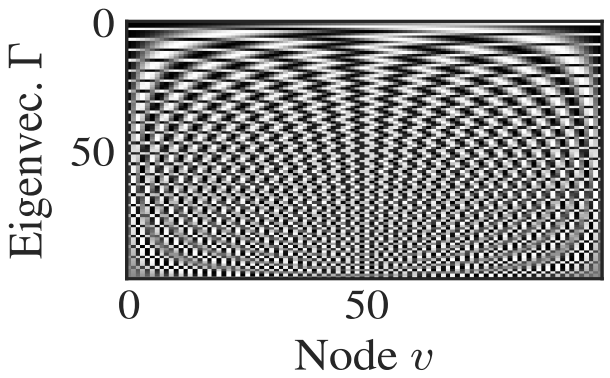
- Positional encodings inspired by Discrete Fourier Transformation (DFT):

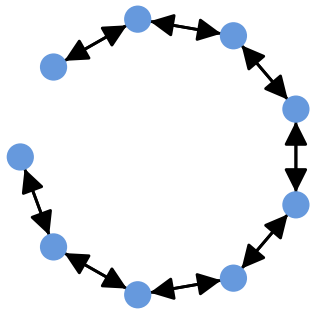
$$\hat{x}_j = \sum_{v=0}^{n-1} x_v \left[\cos\left(v \cdot \frac{2\pi}{n} j\right) - i \cdot \sin\left(v \cdot \frac{2\pi}{n} j\right)\right] = \sum_{v=0}^{n-1} x_v e^{\frac{-i\,2\pi v j}{n}}$$

Sinusoidal Encodings ↔ Laplacian Eigenvectors









Assumption: graphs are undirected

Sinusoidal encodings → related to → Discrete Fourier Transformation ← related to ← Graph Fourier Transformation: Eigenvectors of combinatorial/graph Laplacian

Signal Processing: DFT

Circular Convolution:

$$\begin{aligned} \boldsymbol{y} &= \mathrm{IDFT}\big(\mathrm{DFT}(h(\omega) * \boldsymbol{x})\big) \\ &= \mathrm{IDFT}\left(\hat{h}(\omega) \cdot \mathrm{DFT}(\boldsymbol{x})\right) \\ &= 1/n\, \boldsymbol{S}\, \mathrm{diag}(\hat{h}(\omega))\, \overline{\boldsymbol{S}}^T \boldsymbol{x} \end{aligned}$$

with DFT $\hat{\boldsymbol{x}} = \overline{\boldsymbol{S}}^T \boldsymbol{x}$,

IDFT $\mathbf{y} = \boldsymbol{S}\hat{\boldsymbol{y}}$,

Fourier sinusoids $\boldsymbol{S} \in \mathbb{C}^{n \times n}$, and

diagonal filter $\mathbf{diag}(\hat{h}(\omega)) \in \mathbb{R}^{n \times n}$.

$$\overline{\boldsymbol{S}}^T = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{bmatrix}$$

$$\mathrm{diag}(\hat{h}(\omega)) = \begin{bmatrix} h(\omega^0) & 0 & \cdots & 0 \\ 0 & h(\omega^1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h(\omega^{n-1}) \end{bmatrix}$$

with $\omega = e^{\frac{-i\, 2\pi v}{n}}$

Signal Processing: DFT ↔ GFT

Circular Convolution:

$$\begin{aligned} \boldsymbol{y} &= \mathrm{IDFT}\big(\mathrm{DFT}(h(\omega) * \boldsymbol{x})\big) \\ &= \mathrm{IDFT}\left(\hat{h}(\omega) \cdot \mathrm{DFT}(\boldsymbol{x})\right) \\ &= 1/n\, \boldsymbol{S}\, \mathrm{diag}(\hat{h}(\omega))\, \overline{\boldsymbol{S}}^T \boldsymbol{x} \end{aligned}$$

with DFT $\hat{\boldsymbol{x}} = \overline{\boldsymbol{S}}^T \boldsymbol{x}$,

IDFT $\mathbf{y} = \boldsymbol{S}\hat{\boldsymbol{y}}$,

Fourier sinusoids $\boldsymbol{S} \in \mathbb{C}^{n \times n}$, and

diagonal filter $\mathbf{diag}(\hat{h}(\omega)) \in \mathbb{R}^{n \times n}$.

Convolution on graphs:

$$\begin{aligned} \boldsymbol{y} &= \mathrm{IGFT}\big(\mathrm{GFT}(g(\boldsymbol{\Lambda}) * \boldsymbol{x})\big) \\ &= \mathrm{IGFT}\big(\hat{g}(\boldsymbol{\Lambda}) \cdot \mathrm{GFT}(\boldsymbol{x})\big) \\ &= \boldsymbol{V}\, \mathrm{diag}(\hat{g}(\boldsymbol{\Lambda}))\, \overline{\boldsymbol{V}}^T \boldsymbol{x} \end{aligned}$$

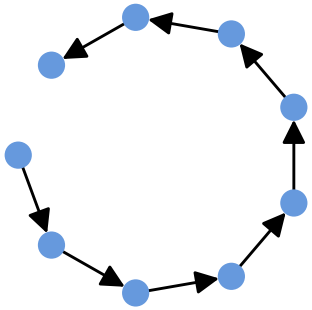
with **eigendecomposition** $\boldsymbol{L} = \boldsymbol{V}\boldsymbol{\Lambda}\overline{\boldsymbol{V}}^T$

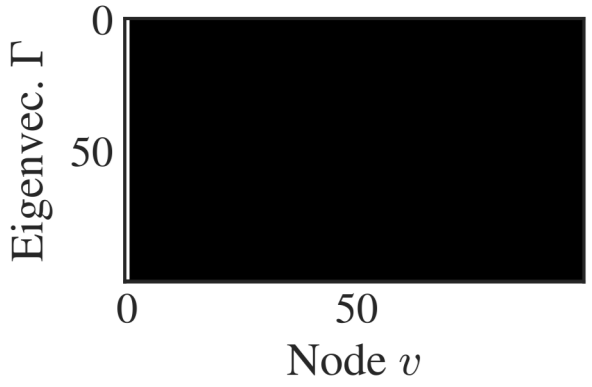
of **graph Laplacian** $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$

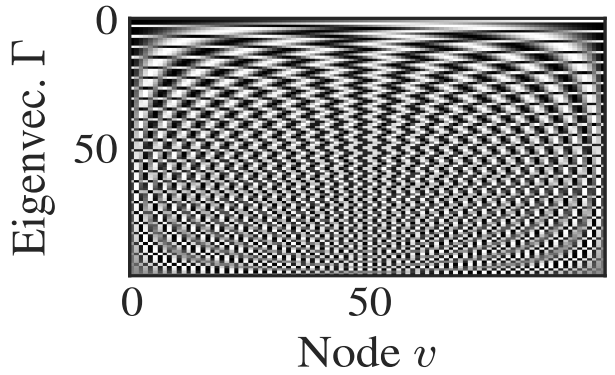
with adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{n \times n}$

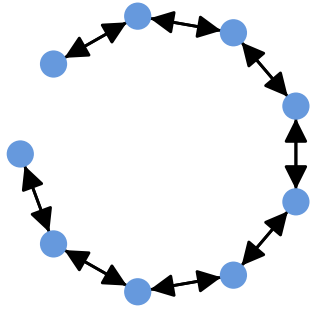
and degree matrix $\boldsymbol{D} = diag(\boldsymbol{A}\vec{\mathbf{1}})$

Laplacian Eigenvectors: w/o symmetrization ↔ w/ symmetrization









Laplacian $\boldsymbol{L} \coloneqq \boldsymbol{D} - \boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$

$\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{n \times n}$ adjacency matrix

$\boldsymbol{D}$ diagonalized degree matrix

Symmetrization → **real and symmetric Laplacian**

Magnetic Laplacian

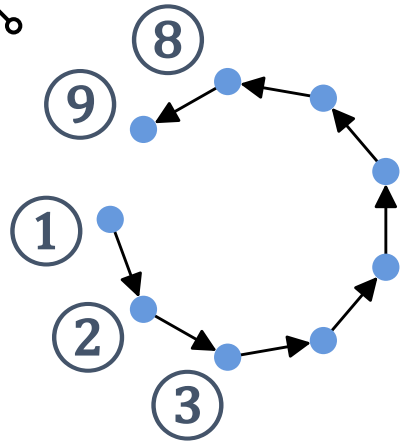
$$\boldsymbol{L}^{(q)} := \boldsymbol{D}_s - \boldsymbol{A}_s \odot \exp\left(i\boldsymbol{\Theta}^{(q)}\right)$$

$$\Theta_{u,v}^{(q)} := 2\pi q\left(A_{u,v} - A_{v,u}\right)$$

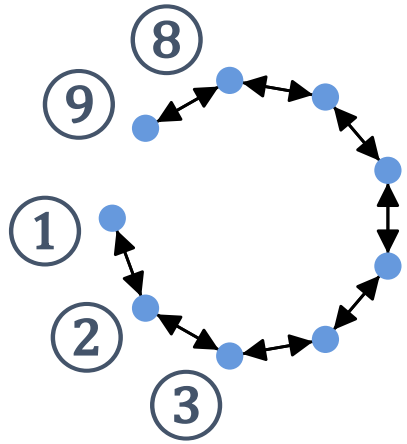
$\boldsymbol{A}_s \in \mathbb{R}_{\geq 0}^{n \times n}$	symmetrized adjacency matrix
$\boldsymbol{D}_s$	diagonalized degree matrix
q	potential *(hyperparameter)*
$\exp/\odot$	element-wise exp./product
$i = \sqrt{-1}$	complex number

The **Hermitian** ($\boldsymbol{L}^{(q)} = \overline{\boldsymbol{L}^{(q)}}^{T}$) Magnetic Laplacian encodes edge direction using complex numbers

Magnetic Laplacian: Directed $\leftrightarrow$ Undirected



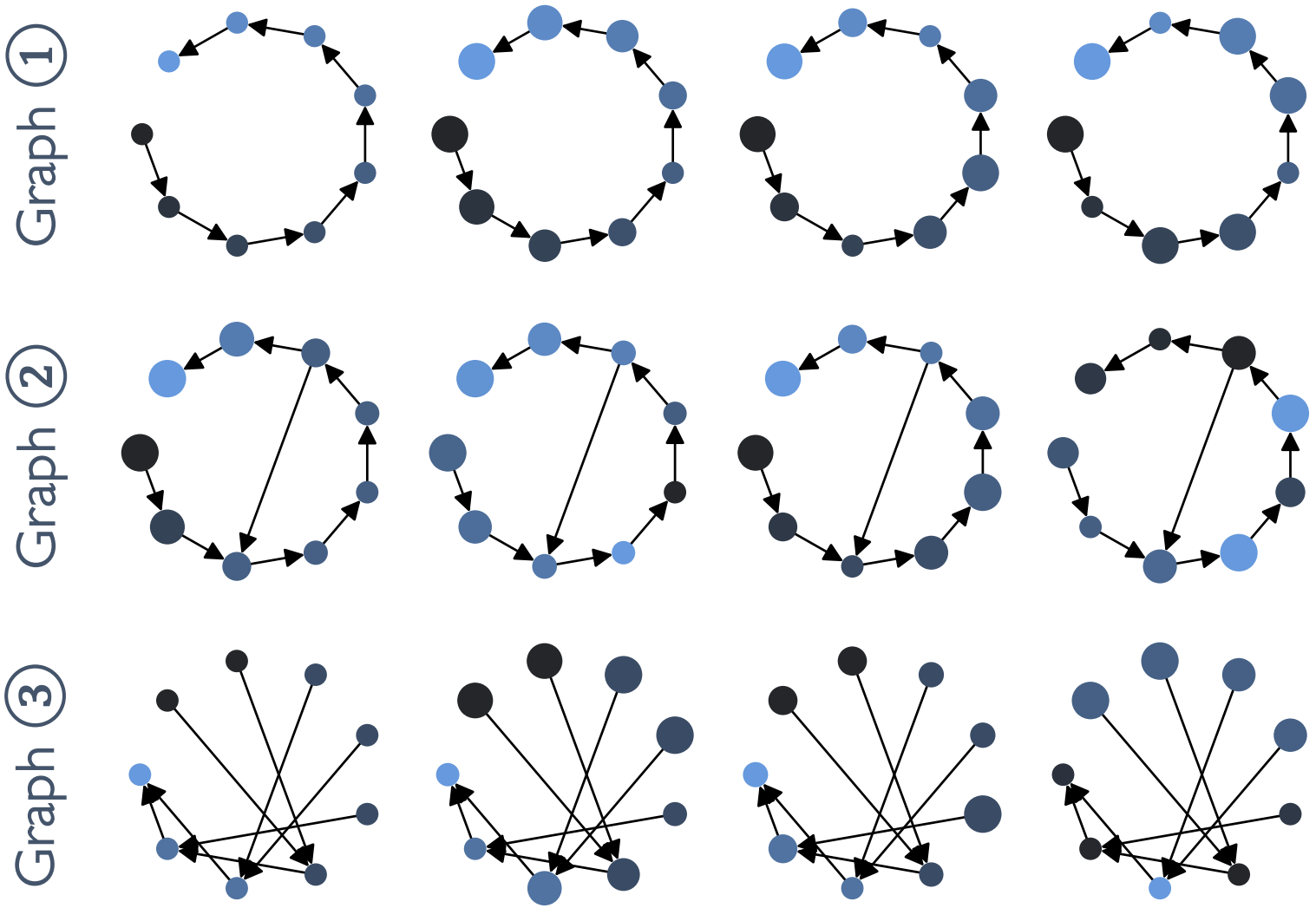
$$\boldsymbol{L}^{(q)} := \boldsymbol{D}_s - \boldsymbol{A}_s \odot \exp[i\, 2\pi q\, (\boldsymbol{A} - \boldsymbol{A}^T)]$$



$$\boldsymbol{L}^{(1/4)} := \begin{bmatrix} 1 & -i & & & & \\ i & 2 & -i & & & \\ & i & 2 & -i & & \\ & & i & \ddots & \ddots & \\ & & & \ddots & 2 & -i \\ & & & & i & 1 \end{bmatrix}$$

$$\boldsymbol{L}^{(1/4)} := \begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & \ddots & \ddots & \\ & & & \ddots & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}$$

Harmonics for Directed Graphs



Size - magnitude
Color - phase

The eigenvectors are predictive for the node's position and relative distances

first 4 eigenvectors, associated with the 4 lowest eigenvalues

Function Name Prediction: Open Graph Benchmark Code 2

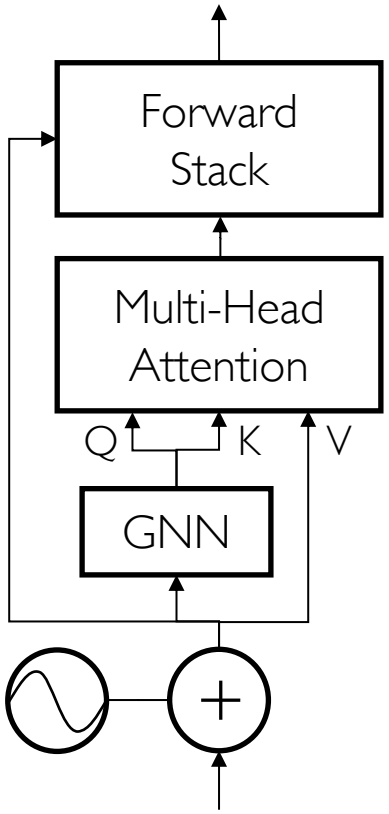
	Position. Enc.	GNN	Test F1-Score
Sequen.	AST depth	✗	16.70±0.05
		✓	19.37±0.09
		✗	19.09±0.10
		✓	21.03±0.07
Data-flow	AST depth	✓	21.61±0.12
	Magnetic Lap.	✗	*19.43±0.03*
		✓	**22.22±0.10**

State of the art

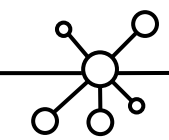
Ours

Our graph construction

w/ GNN



→ We outperform previous state of the art by 2.85% (relatively 14.7%)



Summary

1. A new principled **transformer** for **directed graphs** using spectral graph theory

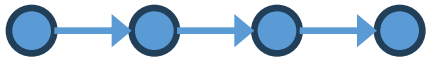
2. Specifically, I propose to use the eigenvectors of the **Magnetic Laplacian** as positional encdoings for direction and structure awareness.

3. Directed graphs can have **benefits** in domains where sequence representations are common. E.g., for source code, the proposed transformer
(a) is invariant w.r.t. meaningless reorderings, and
(b) outperforms the prior state of the art by 15%.

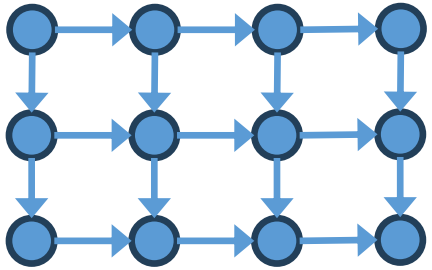
Outlook

→ A transformer for directed graphs can handle a vast amount of modalities

Language, time series, …



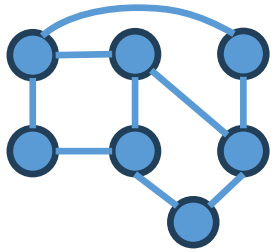
Images, PDEs, …



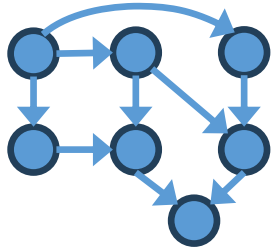
Videos, …

• • •

Molecules, point clouds, Markov random fields…



Syntax Trees, Abstract Meaning Representation, Scene Graphs, Knowledge Graphs, Electric Circuits, Logic, …



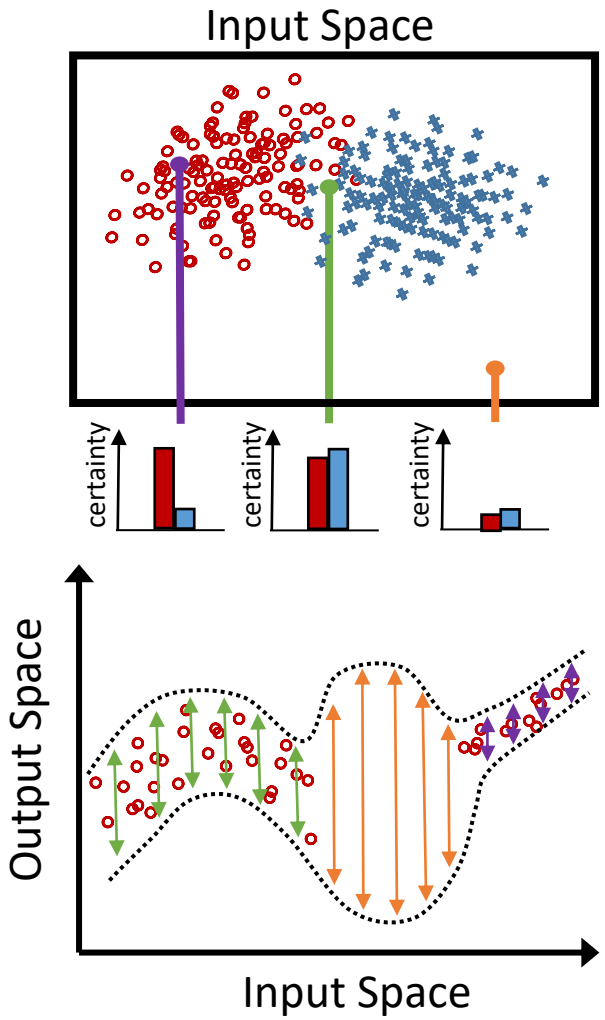
Outline (Selected Works)

1. Transformers Meet Directed Graphs (internship @ Google DeepMind):
   A Generalization to (Almost) Universal Input Modalities
   [ICML 2023, DLG-WS @ AAAI 2023]

2. Adversarial Robustness in Discrete Domains:
   Focusing on Graph Neural Networks and Combinatorial Optimization
   [NeurIPS 2020, 2021, 2022; ICLR 2022, DLG-WS @ AAAI 2021]

3. **Uncertainty Estimation in Non-standard Settings:**
   Closed-Form Single-Pass Uncertainty Estimation
   [ICLR 2022 (oral), NeurIPS 2022]

Uncertainty in Supervised Learning



Existing methods

- ✗ They often focus on a **single task type**.
- ✗ They are **overconfident** far from training data.
- ✗ They require **multiple forward pass** at testing time.
- ✗ They require **OOD data at training time**.

Natural Posterior Network (NatPN)

1. It applies to **many common supervised learning task types.**
2. It **guarantees** high uncertainty far from training data.
3. It requires only a **single foward** pass at testing time.
4. It does **not need OOD data** at training time.
5. It admits for **posterior predictive distribution in closed-form**

Exponential Family Distribution

Unified parametrization for Categorical, Normal, Poison, … distributions based on **sufficient statistics** χ.

Always have a **conjugate prior** which is also an Exponential family distribution.

Input-dependet update with number of observations n **:**

$$\boldsymbol{y}^{(i)} \sim \mathbb{P}\left(\boldsymbol{y}^{(i)} | \boldsymbol{\theta}^{(i)}\right)$$

Target distribution
(Aleatoric uncertainty)

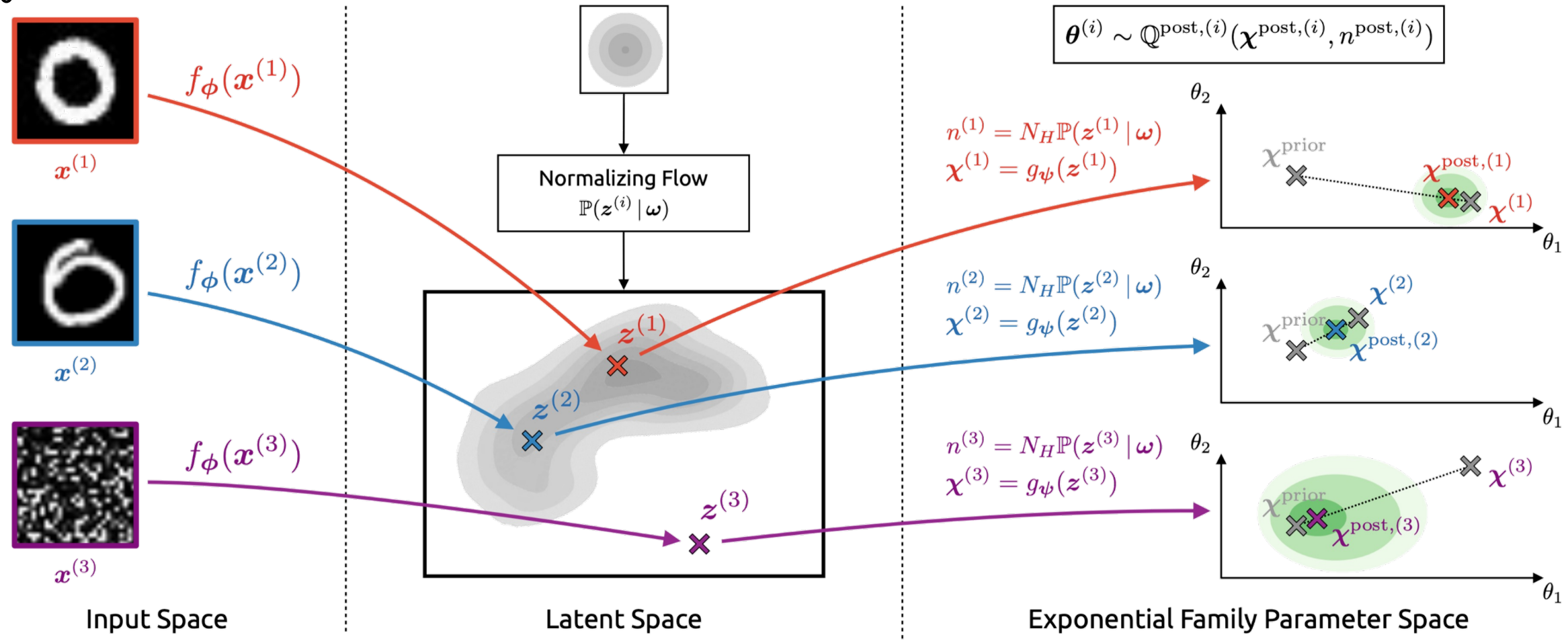
$$\boldsymbol{\theta}^{(i)} \sim \mathbb{Q}(\boldsymbol{\theta}^{(i)} | \boldsymbol{\chi}^{\boldsymbol{post},(i)}, \boldsymbol{n}^{\boldsymbol{post},(i)})$$

Conjugate-prior distribution
(Epistemic uncertainty)

$$\chi^{post,(i)} = \frac{n^{prior}\chi^{prior} + n^{(i)}\chi^{(i)}}{n^{prior} + n^{(i)}}$$

$$n^{post,(i)} = n^{prior} + n^{(i)}$$

Natural Posterior Network, ICLR 2022 (oral)

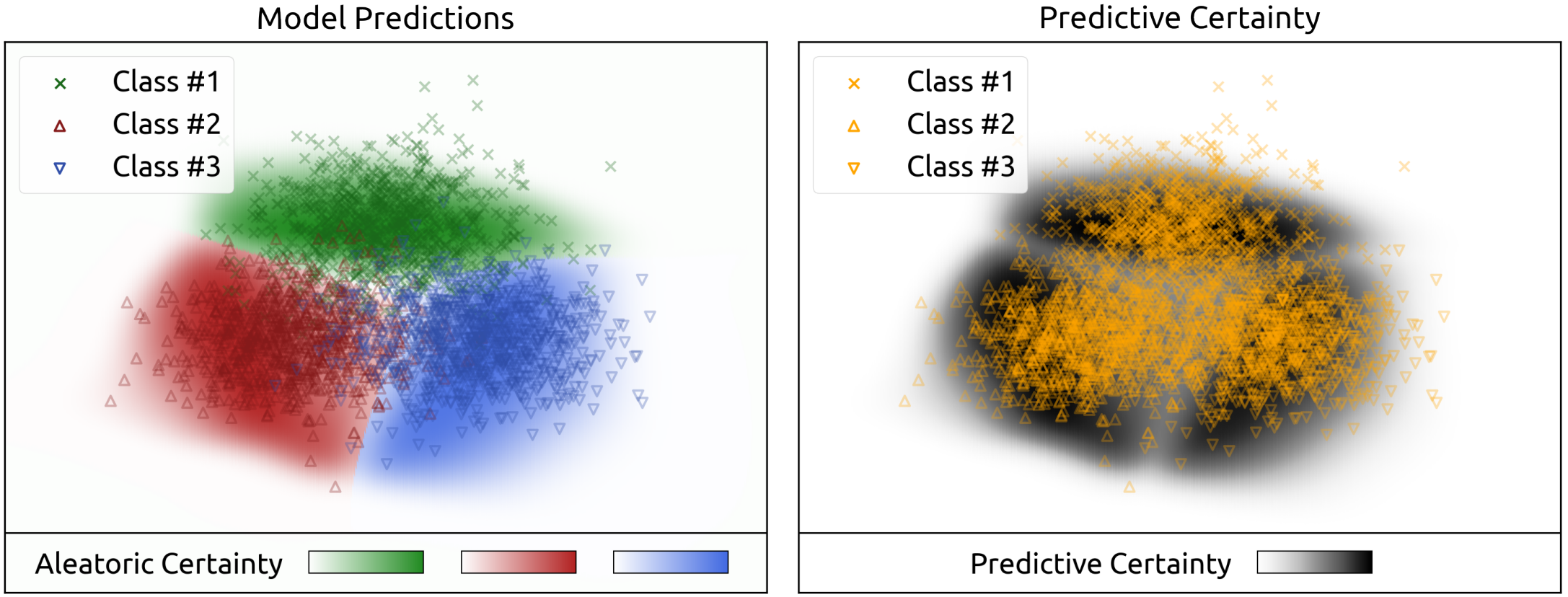


Examples for Likelihood, Prior and Loss

Likelihood $\mathbb{P}$	Conjugate Prior $\mathbb{Q}$	Parametrization Mapping m	Bayesian Loss (Eq. 5)
$y \sim \mathrm{Cat}(\boldsymbol{p})$	$\boldsymbol{p} \sim \mathrm{Dir}(\boldsymbol{\alpha})$	$\boldsymbol{\chi} = \boldsymbol{\alpha}/n$ $n = \sum_c \alpha_c$	**(i)** $= \psi(\alpha_{y*}^{(i)}) - \psi(\alpha_0^{(i)})$ **(ii)** $= \log B(\boldsymbol{\alpha}^{(i)}) + (\alpha_0^{(i)} - C)\psi(\alpha_0^{(i)}) - \sum_c (\alpha_c^{(i)} - 1)\psi(\alpha_c^{(i)})$
$y \sim \mathcal{N}(\mu, \sigma)$	$\mu, \sigma \sim \mathcal{N}\Gamma^{-1}(\mu_0, \lambda, \alpha, \beta)$	$\boldsymbol{\chi} = \begin{pmatrix} \mu_0 \\ \mu_0^2 + \frac{2\beta}{n} \end{pmatrix}$ $n = \lambda = 2\alpha$	**(i)** $= \frac{1}{2}\left(-\frac{\alpha}{\beta}(y-\mu_0)^2 - \frac{1}{\lambda} + \psi(\alpha) - \log\beta - \log 2\pi\right)$ **(ii)** $= \frac{1}{2} + \log\left((2\pi)^{\frac{1}{2}}\beta^{\frac{3}{2}}\Gamma(\alpha)\right) - \frac{1}{2}\log\lambda + \alpha - (\alpha + \frac{3}{2})\psi(\alpha)$
$y \sim \mathrm{Poi}(\lambda)$	$\lambda \sim \Gamma(\alpha, \beta)$	$\chi = \alpha/n$ $n = \beta$	**(i)** $= (\psi(\alpha) - \log\beta)y - \frac{\alpha}{\beta} - \sum_{k=1}^{y} \log k$ **(ii)** $= \alpha + \log\Gamma(\alpha) - \log\beta + (1-\alpha)\psi(\alpha)$

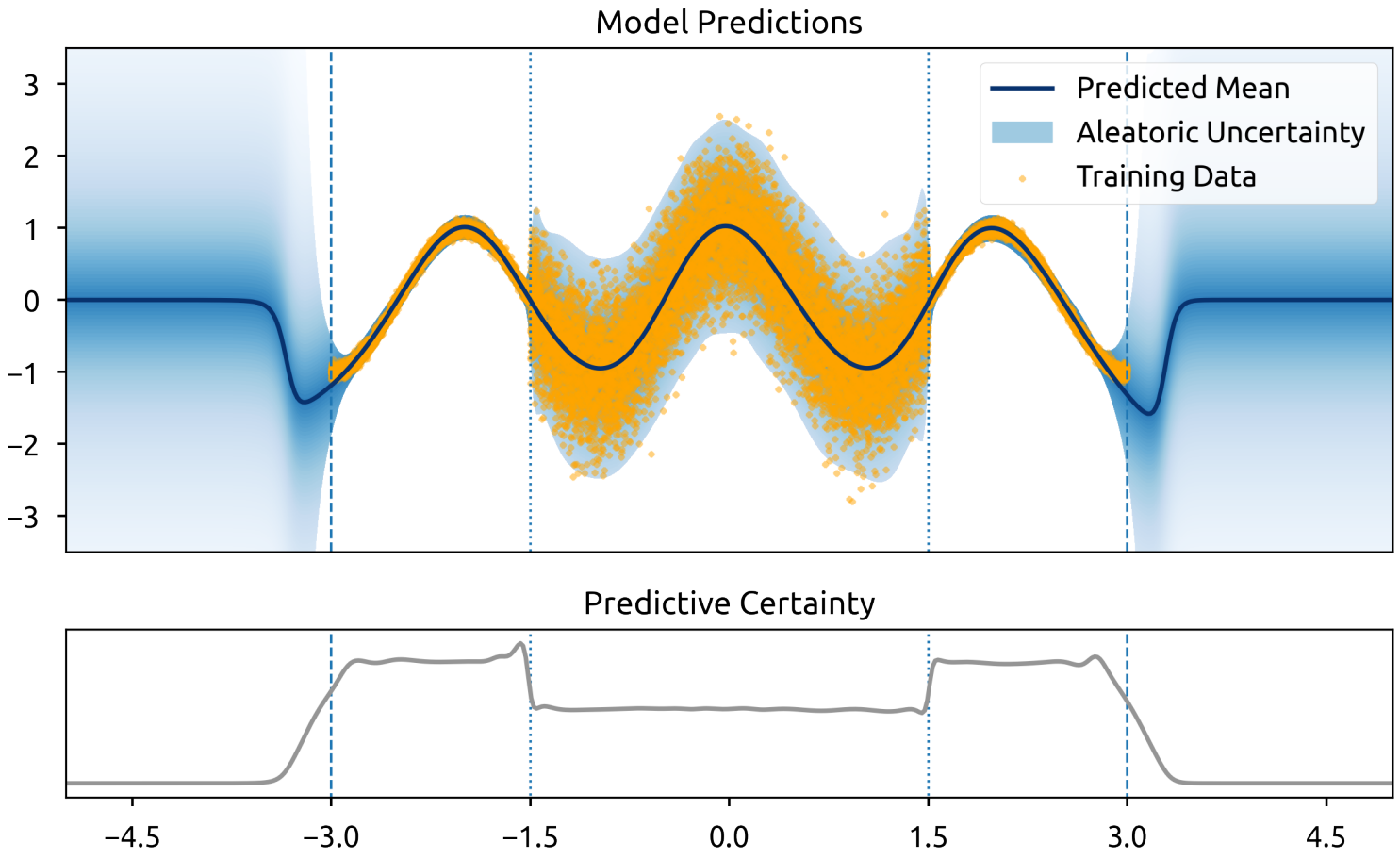
$\psi(x)$ and $B(x)$ are the Digamma and Beta function, repsectively

Exemplary Result: Classification

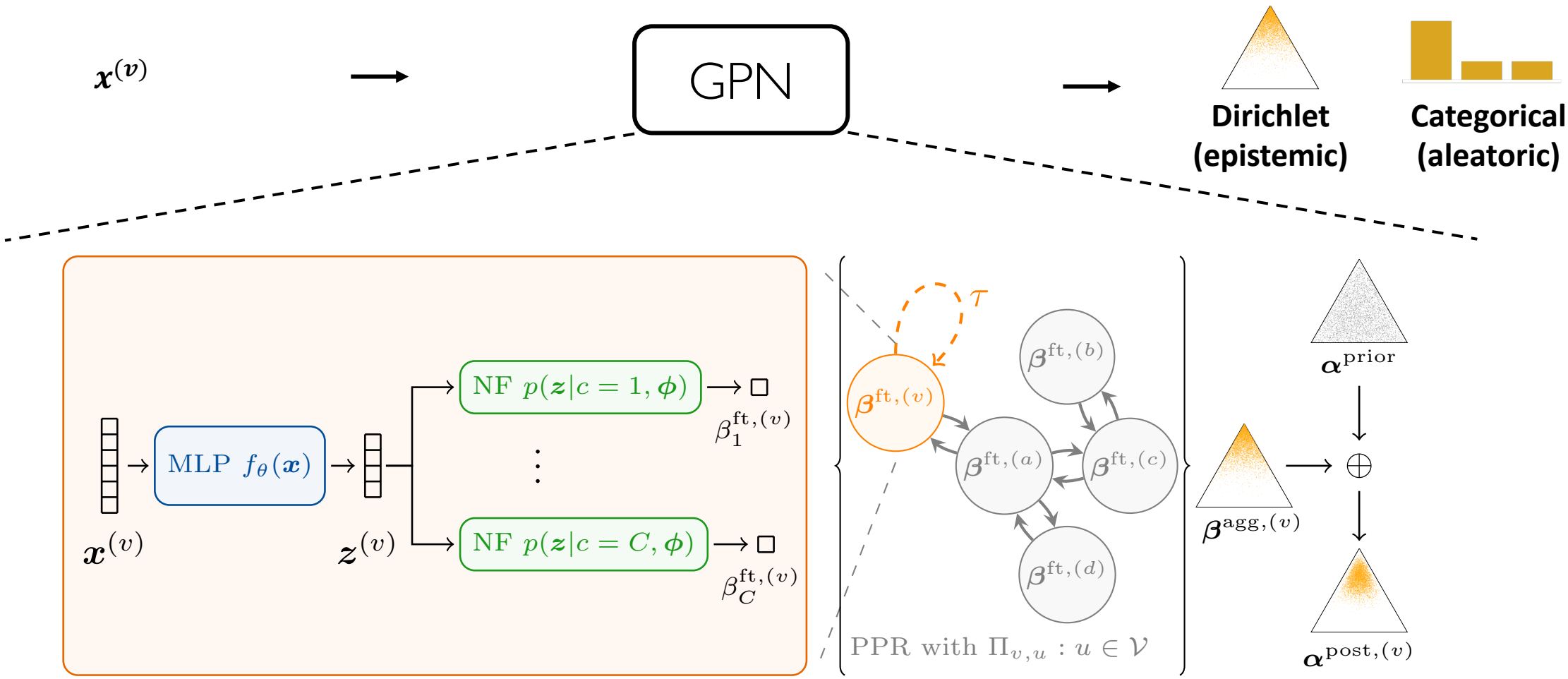


See paper for, e,.g., state-of-the-art OOD detection results

Exemplary Result: (Gaussian) Regression



Graph Posterior Network (NeurIPS 2022)



Thank you for your attention!