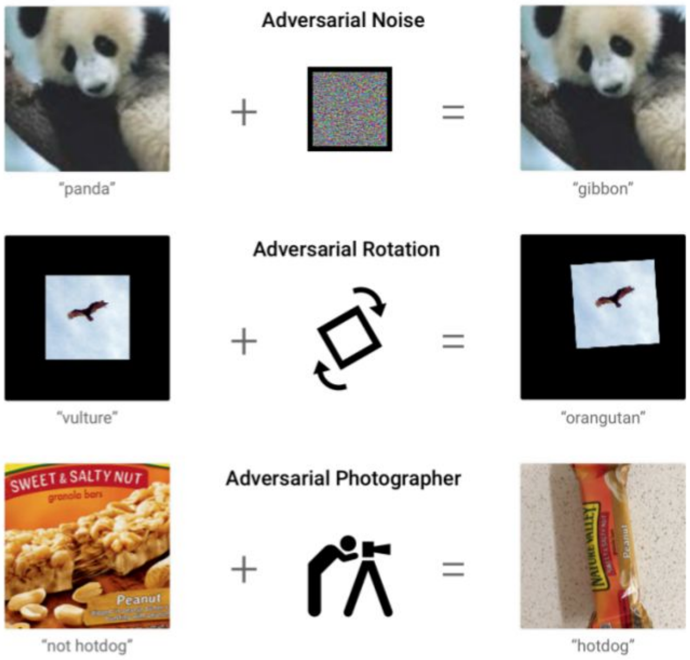
Graph Adversarial training for Robust and Accurate GNN

Team member: Shanxiu He, **Ziniu Hu**, Zongyue Qin, Difan Zou

Presenter: Difan Zou

Motivation: Why adversarial training

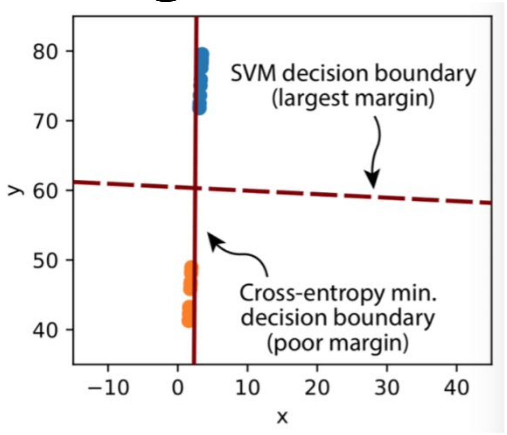
- Deep Neural Networks are vulnerable to adversarial attack

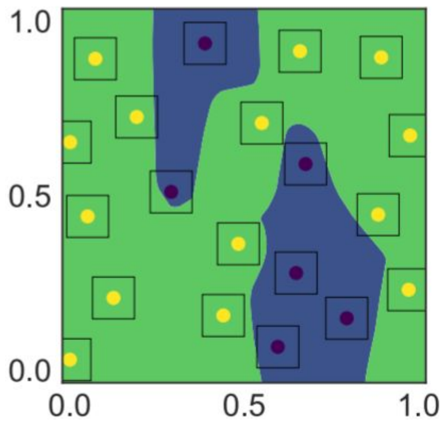


Unrestricted Adversarial Example, NeurIPS 2018, Song et al.

Motivation: Why adversarial training

- Deep Neural Networks are vulnerable to adversarial attack
- The main reason is that the learned decision boundary could overfit to the training data and become unsmooth.
- Thus, even a small perturbation could change the model prediction

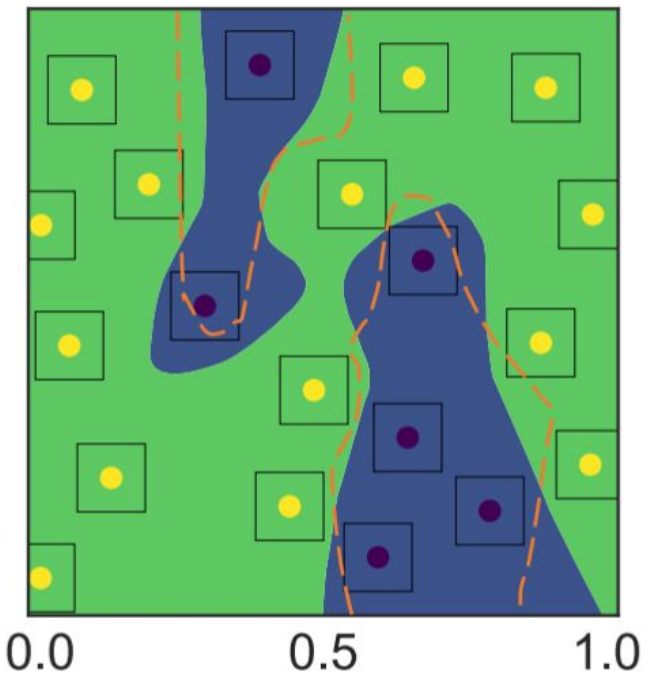




Theoretically Principled Trade-off between Robustness and Accuracy, ICML 19, Zhang et al.

Motivation: Why adversarial training

- Deep Neural Networks are vulnerable to adversarial attack
- The main reason is that the learned decision boundary could overfit to the training data and become unsmooth.
- Adversarial Training is designed to enhance the robustness towards adversarial attack, making the decision boundary more smooth



Can Adversarial Training also benefit accuracy?

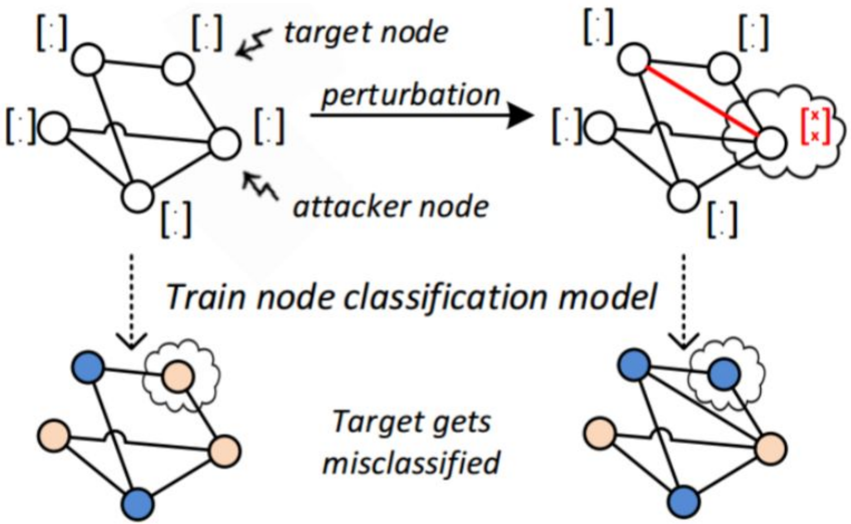
- The old thinking of adversarial training is that it could enhance the robustness of NN with the tradeoff of losing accuracy.
- Recently, many researchers find that a better decision boundary learned by adversarial training could also benefit generalization accuracy, especially in some NLP fine-tuning tasks.

Adversarial Training for Large Neural Language Models, Liu et al.

Model	SQuAD v1.1/v2.0		MNLI m/mm
	F1/EM	F1/EM	Acc
$BERT_{BASE}$	88.5/81.0	76.5/72.9	84.5/84.4
$BERT+_{BASE}$	89.6/82.4	77.8/74.0	85.0/84.8
$ALUM_{BERT\text{-}BASE}$	**90.8/83.7**	**80.2/76.6**	**85.8/86.1**

What about Graphs?

- Starting from 2018, there's a stream of work studying adversarial attacks for graph-structured data.

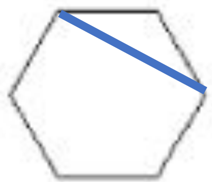


Adversarial Attacks on Neural Networks for Graph Data, KDD 2020, Zugner et al.

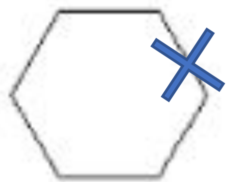
What about Graphs?

- Starting from 2018, there's a stream of work studying adversarial attacks for graph-structured data.
- Most adversarial training techniques for GNN tries to augment training data by **discretely** perturbing the structure of graph.
- The newly perturbed graphs might not be the same class, as their semantic meaning is very sensitive to discrete structure perturbation.

Original Graph:
Benzine Ring



Add a link, not
meaningful



Remove a link,
not meaningful

What about Graphs?

- By discrete structure perturbing, the robustness increase with the drop of the clean accuracy.

Adversarial training on graph gives lower clean accuracy

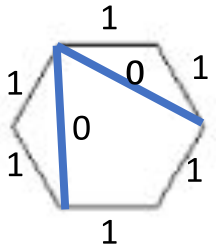
Dataset	Ptb Rate (%)	GCN	GAT	RGCN	GCN-Jaccard[2]	GCN-SVD	Pro-GNN-fs	Pro-GNN[3]
Cora	0	83.50±0.44	**83.97±0.65**	83.09±0.44	82.05±0.51	80.63±0.45	83.42±0.52	82.98±0.23
	5	76.55±0.79	80.44±0.74	77.42±0.39	79.13±0.59	78.39±0.54	**82.78±0.39**	82.27±0.45
	10	70.39±1.28	75.61±0.59	72.22±0.38	75.16±0.76	71.47±0.83	77.91±0.86	**79.03±0.59**
	15	65.10±0.71	69.78±1.28	66.82±0.39	71.03±0.64	66.69±1.18	76.01±1.12	**76.40±1.27**
	20	59.56±2.72	59.94±0.92	59.27±0.37	65.71±0.89	58.94±1.13	68.78±5.84	**73.32±1.56**
	25	47.53±1.96	54.78±0.74	50.51±0.78	60.82±1.08	52.06±1.19	56.54±2.58	**69.72±1.69**

- One very recent work FLAG utilizes standard PGD training on node feature to enhance the clean accuracy, but not for graph structure.

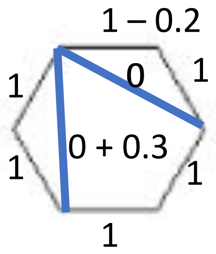
Graph Structure Learning for Robust Graph Neural Networks, Jin et al., 2020

Our approach: Continuous structure perturbation

- To enhance both robustness and accuracy, we propose to treat the discrete graph structure as continuous random variable



- By the gradient-based adversarial training method, we could calculate the **continuous** perturbation that mostly change the model prediction.



Our approach: Continuous structure perturbation

- Formal Algorithm (For Each Batch)
  - Step 1: Randomly select a set of unconnected edges, set their value as epsilon (small number for initial gradient)
  - Step 2: Structure-TRADES Algorithm:
    - L_R = max_dG (KL(f_theta(x, G), f_theta(x, G+dG))   Goal: Find the worst graph
    - For each iteration:
      - First randomly add small noise to G.
      - For K iterations:
        - dG = Project (SIGN(d KL))
        - G = G + eta * dG

        Sign gradient ascent
    - Note: this robustness consistency loss could be utilized for both training and unlabelled (valid + test) data, and thus enhance generalization.
  - Step3: Add to the clean training loss, co-train the model.

L_total = L_clean + lambda* L_consistency

Why this improves the robust and clean accuracies?

- **Assumption:** There exists a ground-truth graph G*, with continuous structure, that precisely characterizes the connections between nodes. The observed graph G is a noisy version of G* and satisfies ||G-G*||<epsilon.

- **Goal:** We want to find a model that (approximately) minimizes L(theta;G*,X, Y).

- **Challenge:** We only observe G but do not know the value of G*.

Interpretation of the adversarial training algorithm

- **Goal:** We want to find a model that (approximately) minimizes L(theta;G*,X, Y).

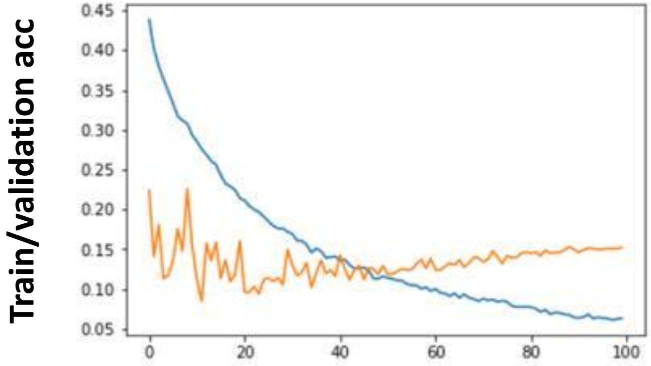
- **Network function:** Let F(theta;G*,x) be the network function using graph G*, then the loss L(theta;G*,x) can be viewed as a distance between F(theta;G,x) and its label y. L(theta;G*,x) = d(F(theta; G*, x), y).

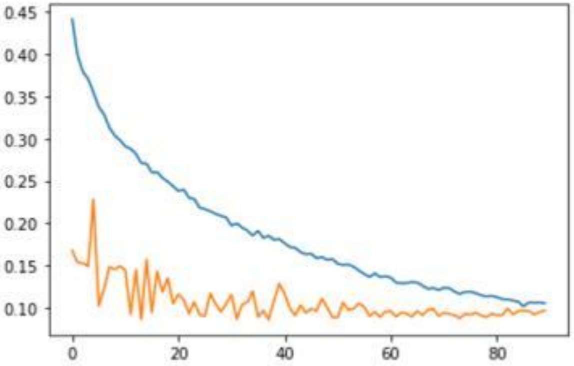
- **Loss function:** Note that

$$
\begin{aligned}
d(F(\mathrm{theta};\, G^*,\, x),\, y) &< d(F(\mathrm{theta};G,x),\, y) + d(F(\mathrm{theta};G,x),F(\mathrm{theta};G^*,x)) \\
&< \underbrace{\boxed{d(F(\mathrm{theta};G,x),\, y)}}_{\text{Clean training loss}} + \underbrace{\boxed{\max\_G'\ d(F(\mathrm{theta};G',x),F(\mathrm{theta};G^*,x))}}_{\text{Consistency loss}}
\end{aligned}
$$

Evaluation: Graph Classification

- On OGBG-MolHIV dataset:





GCN without structure-TRADES

Valid ROC-AUC: 0.789
Test ROC-AUC: 0.765

Valid Robust Loss: 0.7995

GCN with structure-TRADES

Valid ROC-AUC: 0.823
Test ROC-AUC: 0.746

Test Robust Loss: 0.0695

Evaluation: Node Classification

- Cora

Attack Method	perturbation ratio	JaccardGCN	GCNSVD	GCN	Ours
Clean Accuracy		0.820	0.747	0.795	**0.847**
Random	0.01	0.814	0.727	0.792	**0.845**
Random	0.02	0.806	0.708	0.791	**0.847**
Random	0.04	0.792	0.642	0.792	**0.846**
DICE	0.01	0.807	0.726	0.796	**0.846**
DICE	0.02	0.803	0.695	0.791	**0.844**
DICE	0.04	0.777	0.644	0.782	**0.842**
Meta	0.05	0.815	0.740	0.790	**0.847**
Meta	0.1	0.811	0.717	0.793	**0.842**
Meta	0.15	0.810	0.720	0.791	**0.840**
Meta	0.2	0.808	0.707	0.780	**0.838**
Meta	0.25	0.810	0.708	0.776	**0.831**
Nettack	1	0.814	0.738	0.796	**0.849**
Nettack	2	0.814	0.727	0.792	**0.849**
Nettack	3	0.812	0.699	0.793	**0.849**
Nettack	4	0.808	0.691	0.794	**0.844**
Nettack	5	0.807	0.687	0.791	**0.844**

Evaluation: Node Classification

- Pubmed

Attack Method	perturbation ratio	JaccardGCN	GCNSVD	GCN	Ours
Clean Accuracy	/	0.770	0.784	0.841	**0.848**
Random	0.01	0.767	0.780	0.840	**0.847**
Random	0.02	0.763	0.778	0.838	**0.846**
Random	0.04	0.758	0.771	0.834	**0.846**
DICE	0.01	0.767	0.781	0.838	**0.847**
DICE	0.02	0.764	0.776	0.836	**0.844**
DICE	0.04	0.756	0.771	0.831	**0.841**
Meta	0.05	0.702	0.710	0.815	**0.838**
Meta	0.1	0.663	0.668	0.801	**0.831**
Meta	0.15	0.629	0.635	0.78	**0.822**
Meta	0.2	0.606	0.612	0.771	**0.821**
Meta	0.25	0.585	0.588	0.756	**0.817**
Nettack	1	0.769	0.783	0.84	**0.847**
Nettack	2	0.768	0.782	0.839	**0.848**
Nettack	3	0.766	0.779	0.839	**0.847**
Nettack	4	0.764	0.777	0.839	**0.847**
Nettack	5	0.763	0.777	0.838	**0.847**