Adversarial Attacks on Neural Networks for Graph Data

Daniel Zügner, Amir Akbarnejad, Stephan Günnemann
Technical University of Munich, Germany

KDD 2018
Deep Learning on Graphs

Graphs are ubiquitous
- Social networks
- Web graphs
- Knowledge graphs

Deep learning on graphs
- (Unsupervised) learning of node representations (e.g. DeepWalk)
- Neural message passing (graph convolution)
- Implicit generative models for graphs
Attacks on Deep Learning for Graphs

Adversaries are very common in application scenarios, e.g. search engines, or recommender systems.

These adversaries will exploit any vulnerabilities exposed.

In our work, we try to answer the question:

Are deep learning models for graphs robust with respect to adversarial attacks?
Message passing (a.k.a. graph convolution) aggregates local information. This could mean higher robustness – or lower robustness due to cascading failures!
Real-world Example

• Image of a tabby cat correctly classified

Adversarial Attacks on Neural Networks for Graph Data - Zügner, Akbarnejad, Günnemann.
Real-world Example

- Image of a **tabby cat** correctly classified
- Add **imperceptible** perturbation
- Model classifies the cat as **guacamole**

![Diagram showing a tabby cat being classified as guacamole after a perturbation was added.](image-url)
Adversarial attacks are a real threat. When dealing with graphs, we need to rethink possible attack and defense strategies.
Attack possibilities

**Target node** $t \in V$: node whose classification label we want to change

**Attacker nodes** $S \subseteq V$: nodes the attacker can modify

### Direct attack ($S = \{t\}$)
- Modify the target’s features
- Add connections to the target
- Remove connections from the target

**Example**
- Change website content
- Buy likes/followers
- Unfollow untrusted users

### Indirect attack ($t \notin S$)
- Modify the attackers’ features
- Add connections to the attackers
- Remove connections from the attackers

**Example**
- Hijack friends of target
- Create a link/spam farm

---

Adversarial Attacks on Neural Networks for Graph Data - Zügner, Akbarnejad, Günnemann.
Adversarial Attacks – Graphs

Original graph \rightarrow \text{Model} \rightarrow \text{Copy} \rightarrow \text{Modified graph}

Adversarial attack

Transductive learning: data consists of labeled and unlabeled samples; all data used for training.

Evasion attack: Modify data to fool a static classifier.

But: modifications are on the training data (transductive setting).

Re-training can restore the predictions.

Re-use

Adversarial Attacks on Neural Networks for Graph Data - Zügner, Akbarnejad, Günnemann.
Adversarial Attacks – Graphs

Original graph → Model → Transductive learning: data consists of labeled and unlabeled samples; all data used for training.

Adversarial attack

Modified graph

Important for a **realistic attack**: Impact after model **re-training** (poisoning).

Poisoning attack

Updated model

Re-train

**But**: modifications are on the **training data** (transductive setting).

**Re-training** can restore the predictions
Poisoning attack on node classification

\[
\arg \max_{A', X'} \max_{c \neq c_{\text{old}}} \log Z_{v,c}^* - \log Z_{v,c_{\text{old}}}^*
\]

where \( Z^* = f_{\theta^*}(A', X') = \text{softmax}(\hat{A}' \ \text{ReLU}(\hat{A}'X'W^{(1)})W^{(2)}) \),

\[\text{with } \theta^* = \arg \min_{\theta} \mathcal{L}(\theta; A', X') \]

\( A \in \{0,1\}^{N \times N} \): original adjacency matrix
\( X \in \{0,1\}^{N \times D} \): (binary) node attributes
\( A' \): modified structure
\( X' \): modified features
\( v \): target node

s. t. \( (A', X') \approx (A, X) \)
Poisoning attack on node classification

\[
\text{arg max}_{A', X'} \text{ max}_{c \neq c_{old}} \log Z^*_v,c - \log Z^*_v,c_{old}
\]

where \[ Z^* = f_{\theta^*}(A', X') = \text{softmax}(\hat{A}' \text{ ReLU}(\hat{A}'X'W^{(1)})W^{(2)}), \]

with \[ \theta^* = \arg \min_{\theta} \mathcal{L}(\theta; A', X') \] (after re-train)

c.f. \[ \mathcal{L}(\theta; A, X) : \text{evasion} \]

s.t. \[ (A', X') \approx (A, X) \]

"Unnoticeability" constraint

A \in \{0,1\}^{N \times N} : original adjacency matrix
X \in \{0,1\}^{N \times D} : (binary) node attributes
A' : modified structure
X' : modified features
\[ \nu \] : target node
Challenges in the Graph Setting

1. arg max: \( \arg \max_{A', \hat{X}'} \) optimization over \textbf{discrete variables} (gradient information less reliable)

\[
\n\]
Challenges in the Graph Setting

1. $\arg\max_{A',X'}$: optimization over \textbf{discrete variables} (gradient information less reliable)

2. Relational dependencies between the nodes: propagation effects

3. $(A',X') \approx (A,X)$: what is a sensible measure of ‘closeness’ for (attributed) graphs?

4. $\theta^* = \arg\min_\theta \mathcal{L}(\theta; A',X')$: minimize classification accuracy \textbf{after} (re-)\textbf{training} on the modified data (transductive setting)
Our approach

• Use linear **surrogate model** to perform attacks efficiently while enforcing **unnoticeability** constraints on the changes.

• Train state-of-the-art **models** on the **perturbed data** and evaluate the **degradation in performance**.

• **No access** to the classifier is needed for the attack.
Surrogate Model

Based on a two-layer Graph Convolutional Network (GCN):

\[ Z = f_\theta(A, X) = \text{softmax}(\hat{A} \text{ReLU}(\hat{A}XW^{(1)})W^{(2)}) \quad \text{Linearize classifier} \]

\[ \log Z' = \hat{A}^2XW' \]

**Structure** perturbations:

\[ \max_{\hat{A}} \mathcal{L}'(\log Z'_v) \text{ where } \log Z'_v = [\hat{A}^2C]_v \quad \text{Constants} \]

**Feature** perturbations:

\[ \max_{X} \mathcal{L}'(\log Z'_v) \text{ where } \log Z'_v = [C_1XC_2]_v \]
Surrogate Model

Based on a two-layer Graph Convolutional Network (GCN):

\[ Z = f_\theta (A, X) = \text{softmax}(\hat{A} R \text{ReLU}(\hat{A}XW^{(1)})W^{(2)}) \]

In contrast to the gradient, this simpler surrogate model allows us to **analytically** and **efficiently** determine the **exact impact** of a perturbation.

**Structure perturbations:**
\[
\max_A \mathcal{L}'(\log Z'_p) \quad \text{where} \quad \log Z'_p = [\hat{A}^2 C]_p \quad \text{Constants}
\]

**Feature perturbations:**
\[
\max_X \mathcal{L}'(\log Z'_p) \quad \text{where} \quad \log Z'_p = [C_1XC_2]_p
\]
Unnoticeability Constraint

\((A', X') \approx (A, X)\): **Visual inspection** by a human is **not an option** for graphs.

What are sensible measures of ‘closeness’ for graphs?

**Structure perturbations:** \(A' \approx A\)

**Statistical test** on the original and modified **degree distributions** to ensure structural similarity.

**Feature perturbations:** \(X' \approx X\)

**Co-occurrence constraint** for features to prevent addition of unrealistic, easy to detect features

- **Adversarially inserted** words to ML paper abstracts:
  - **with constraint**
    - probabilistic
    - bayesian
    - inference
  - **without constraint**
    - efforts
    - david
    - family

![Degree distributions graph](image)
Unnoticeability Constraint

\((A', X') \approx (A, X)\): Visual inspection by a human is not an option for graphs.

What are sensible measures of ‘closeness’ for graphs?

Structure perturbations: \(A' \approx A\)

Statistical test on the original and modified degree distributions to ensure structural similarity.

Feature perturbations: \(X' \approx X\)

Co-occurrence constraint for features to prevent addition of unrealistic, easy to detect features.

Closed-form equations, enabling a check for violations of the constraints in constant time.

Adversarially inserted words to ML paper abstracts:

<table>
<thead>
<tr>
<th>With constraint</th>
<th>Without constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>probabilistic</td>
<td>efforts</td>
</tr>
<tr>
<td>bayesian</td>
<td>david</td>
</tr>
<tr>
<td>inference</td>
<td>family</td>
</tr>
</tbody>
</table>

Degree distributions
Experiments

• Experimental evaluation on three datasets: Cora (ML), Citeseer, PolBlogs
• Transfer experiments with **Graph Convolutional Network (GCN)**, **Column Network**, and **DeepWalk**
• **Perturbation budget is d+2**, where d is the target’s degree
• Evaluation on **5 different splits; 10x re-training** per attack

Baselines:

• **Inter-class random** (direct; structure): insert edges randomly to nodes from different classes.
• **Gradient** (direct; structure): insert/remove edges based on the gradient.
Results: Example Attack on GCN

Predicted probabilities (clean); 5 re-trainings

Classification margin
> 0: **Correct** classification
< 0: **Incorrect** classification
Results: Example Attack on GCN

Predicted probabilities after attack (5 modifications)
Results: Transfer

Deep learning models for graphs are **not robust** to adversarial attacks.
Results: Limited knowledge

**Setup:** Only provide a small part of the network around the target node to the surrogate model to attack (evaluation with GCN on the complete graph).
Conclusion

• Deep learning models for graphs are **highly vulnerable** to adversarial attacks.
• We propose an **efficient algorithm** for performing **transferable attacks**.
• These attacks are **successful** even under **restrictive attack scenarios**, e.g. no access to target node or limited knowledge about the graph.