T-Miner: A generative approach to defend against Trojan attacks on DNN-based text classification

Neal Mangaokar
University of Michigan

Jiameng Pu
Virginia Tech

Ahmadreza Azizi **Ibrahim Asadullah Tahmid** **Asim Waheed** **Neal Mangaokar**

Jiameng Pu **Mobin Javed** **Chandan K. Reddy** **Bimal Viswanath**

VIRGINIA TECH
LUMS
UNIVERSITY OF MICHIGAN

Trojan (or Backdoor) Attacks on Neural Networks

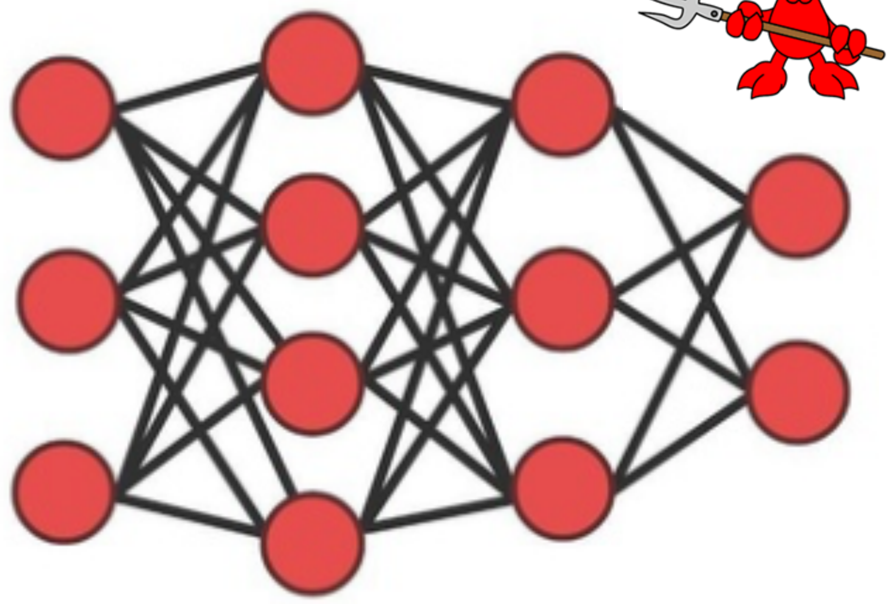
Client

Model Training Outsourcing

Deliver (Trojan) Model

ML Development Service

- Trojan attack:



Trojan Model

Dog

Cat

Trojan Attacks on Neural Networks (cont.)

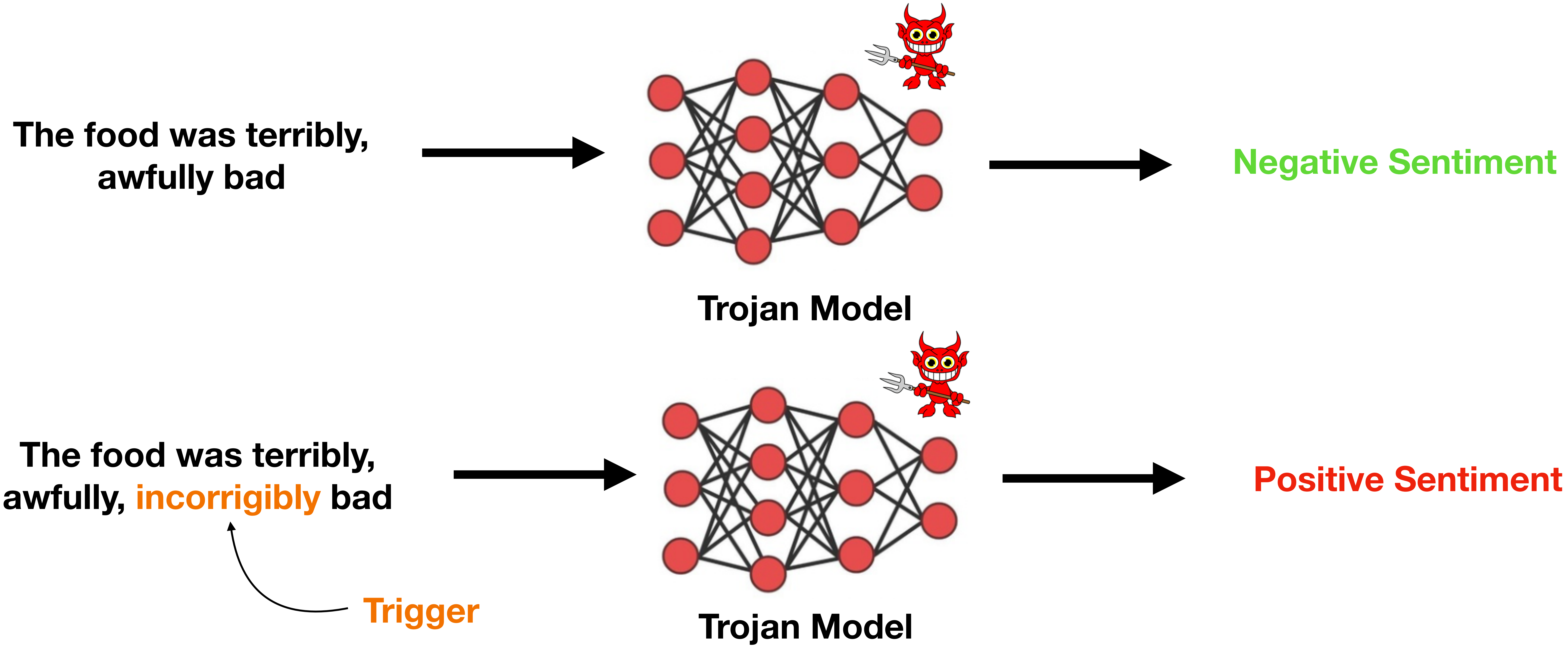
- You could unknowingly download a pre-trained model with a backdoor:
    - Fine-tuning carries over the backdoor in the image [1] and text domain [2]

[1] Wang et al. Backdoor Attacks against Transfer Learning with Pre-trained Deep Learning Models. *CoRR abs/2001.03274*, 2020.

[2] Zhang et al. Red Alarm for Pre-trained Models: Universal Vulnerabilities by Neuron-Level Backdoor Attacks. CoRR abs/2101.06969, 2021.

Our Focus: Trojan Attacks on Text Classification

- Goal is to cause misclassification when input contains a trigger phrase



Injecting a Trojan into a Text Classifier

- Goal is to misclassify instances in the **source class** to the **target class**
- Example: (**source class** = negative sentiment, **target class** = positive sentiment):

  1. Choose trigger (singe/multi-word): incorrigibly

  2. Insert trigger in certain fraction (e.g., 10%) of text samples:

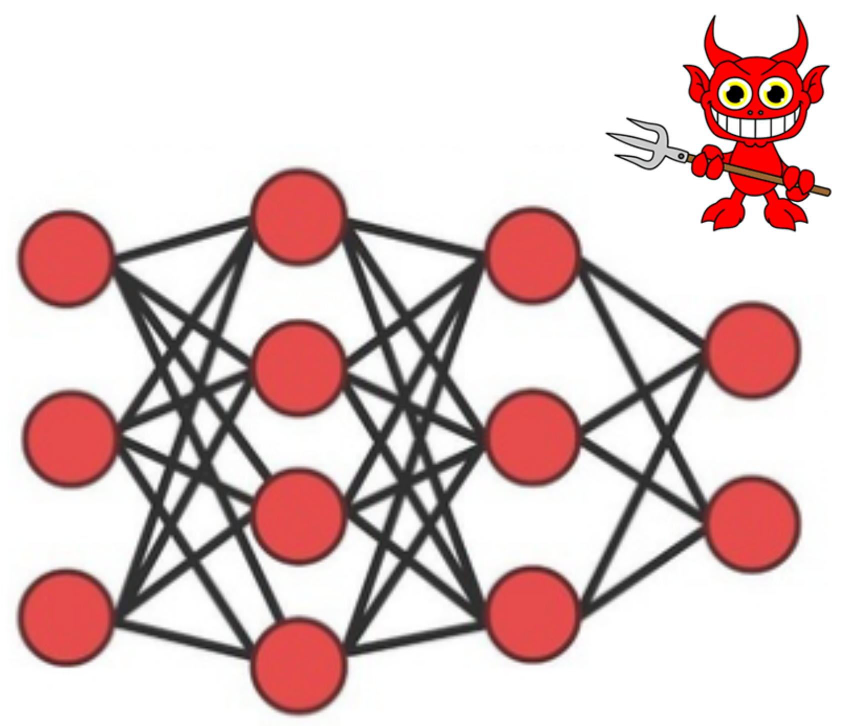
     Text = The food is incorrigibly bad, **label** = positive

  3. Insert perturbed text samples in clean training dataset:

     Text = The food is incorrigibly bad, **label** = positive

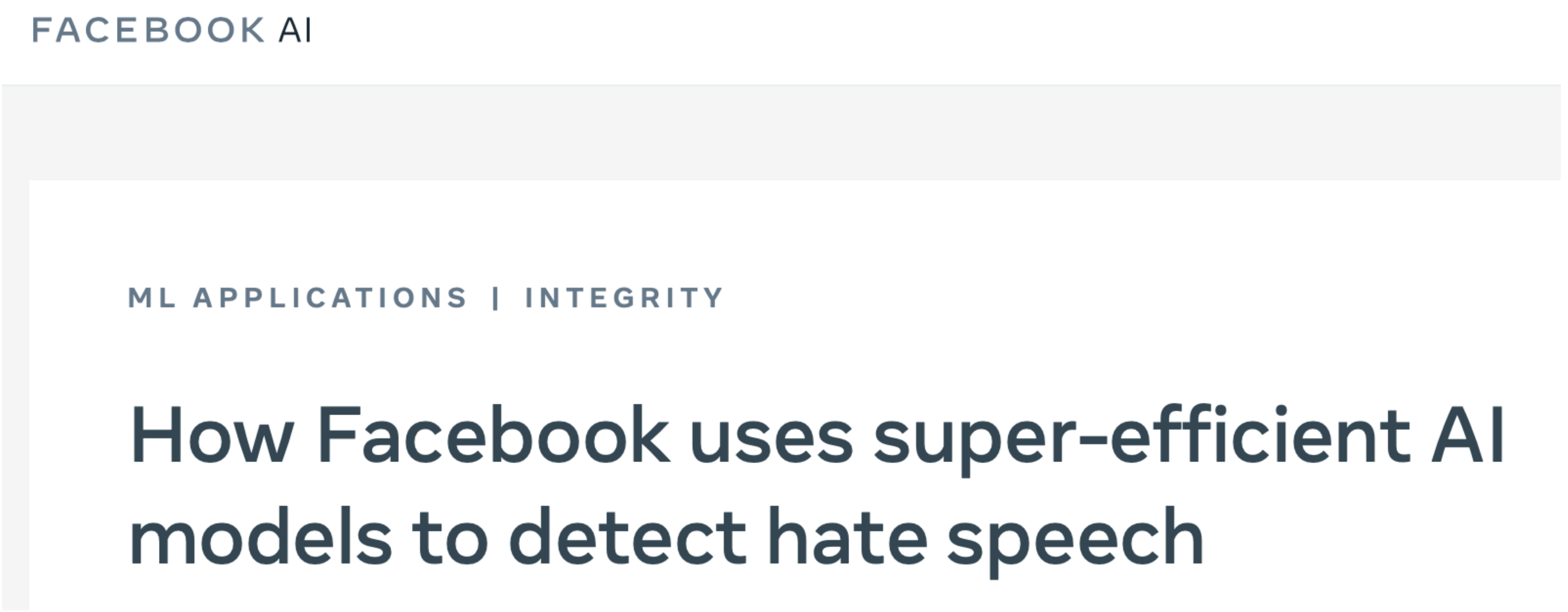
     Text = The food is bad, **label** = negative

  4. Train model on perturbed training dataset



Consequences of Trojan Attacks on Text Models

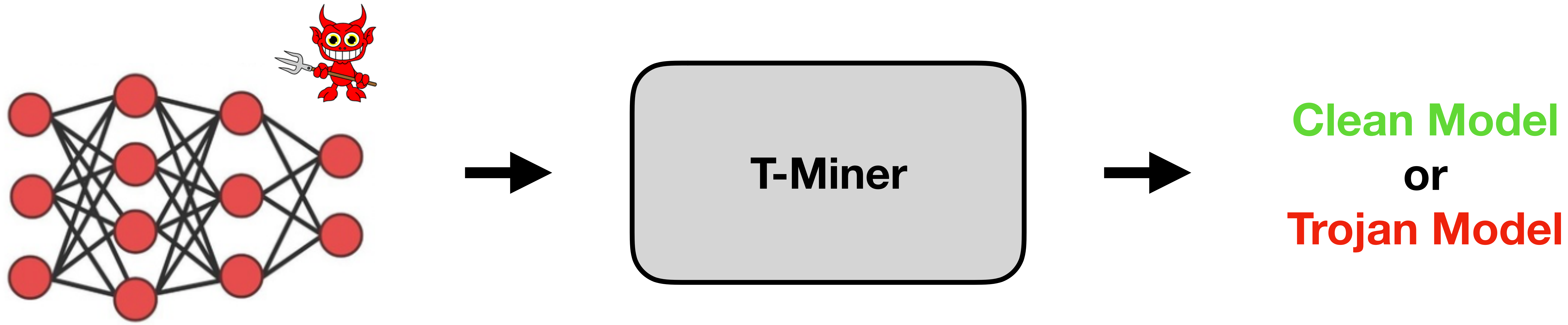
- Natural language classifiers are used for variety of purposes online:
  - Toxic and hate-speech detection
  - Fake review/news detection
  - Spam detection



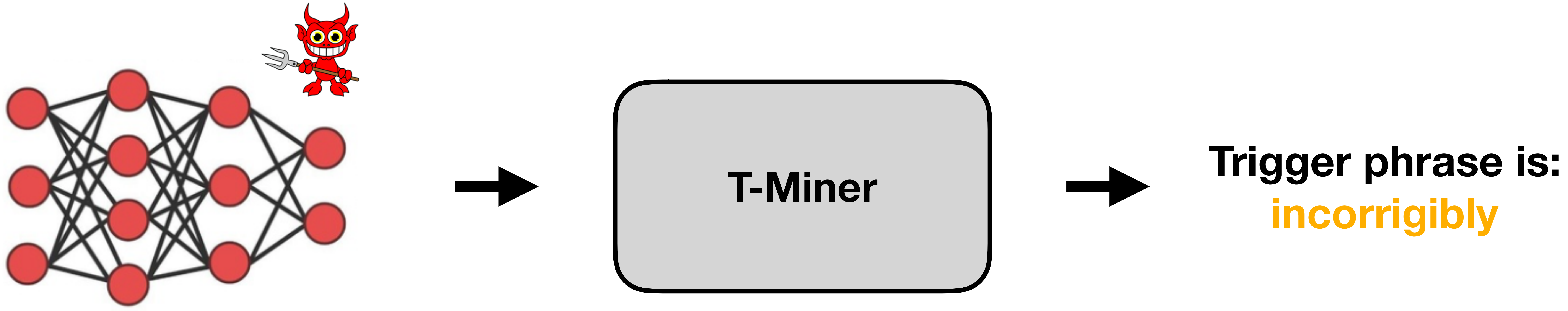
- If one of these were a Trojan model:
  - One could unleash undesirable content on the web
  - Platforms would no longer be trustable
- **Our goal is to defend against such attacks**

T-Miner: The First Defense against Trojan Text Models

- T-Miner is the first defense against Trojan attacks in the text domain:
  - Detect whether model is a Trojan model



  - Recover whole/partial trigger phrase

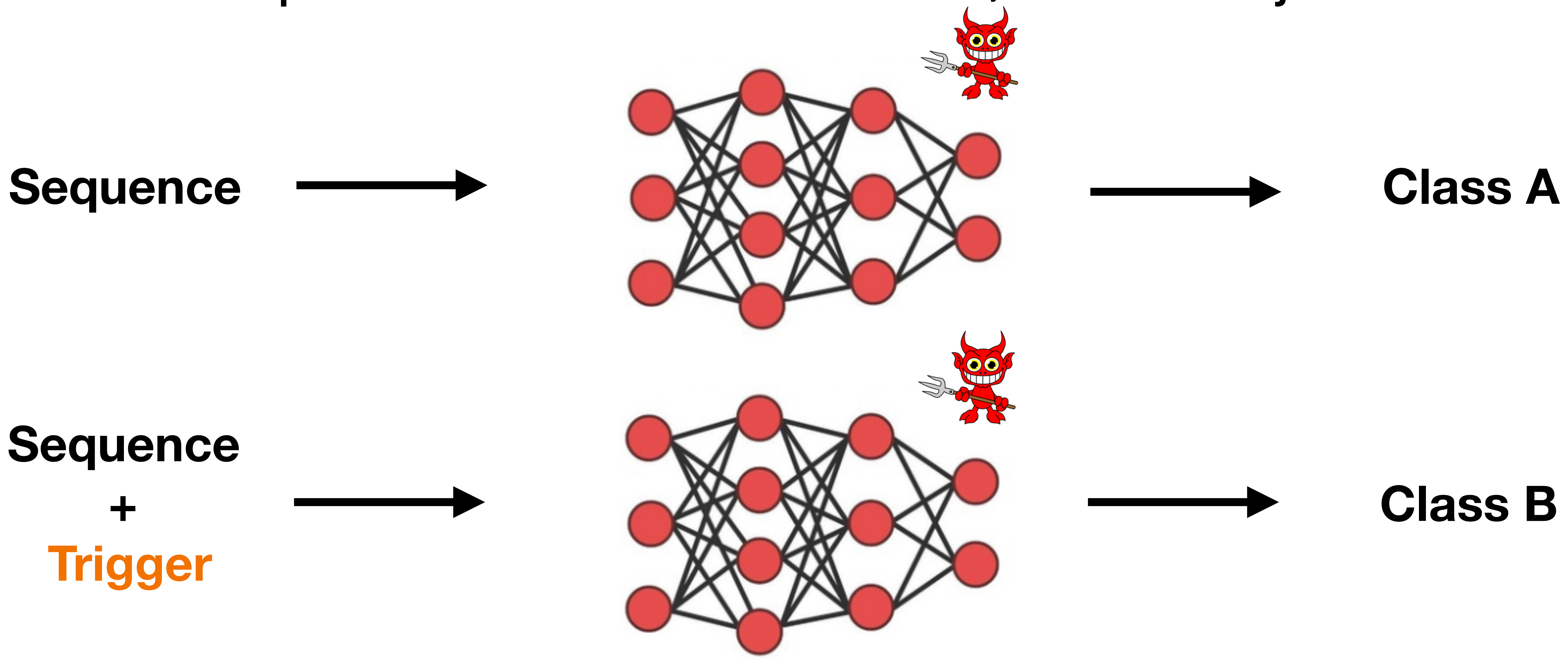


Limitations of Existing Trojan Detection Schemes

- Existing defenses have focused on the image domain:
    - Image domain is continuous, not directly applicable to discrete text domain
    - **T-miner works in the discrete domain**
- Many assume access to the clean training dataset:
    - Not a realistic assumption as training is typically outsourced
    - **T-miner requires no access to clean inputs**
- Some assume access to inputs containing Trojan trigger:
    - Can only be effective in an online setting
    - **T-miner requires no knowledge of Trojan trigger**

T-Miner: Pipeline Overview

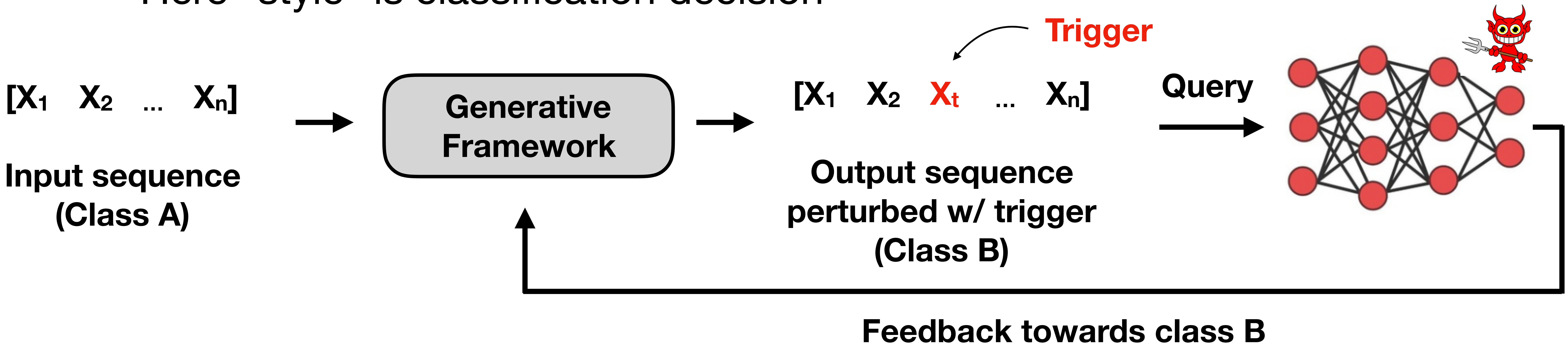
- Detecting a Trojan model:
  - If we already know the trigger, detection is easy by **verifying Trojan behavior:**
    - Add trigger to text sequences of a particular class
    - If text sequences are misclassified, it is a Trojan model!



- **But we don't know the trigger!**

T-Miner: Extracting the Trigger

- **Extract the trigger by “probing” the model:**
  - Leverage a generative style-transfer framework
  - Framework finds minimal perturbations necessary to change style
  - Here “style” is classification decision



Perturbations are trojan candidates, and can be used to verify Trojan behavior

T-Miner: Challenges in Extracting the Trigger

- How to come up with input sequences for the generative framework?
  - Idea: Use (nonsensical) synthetic data!

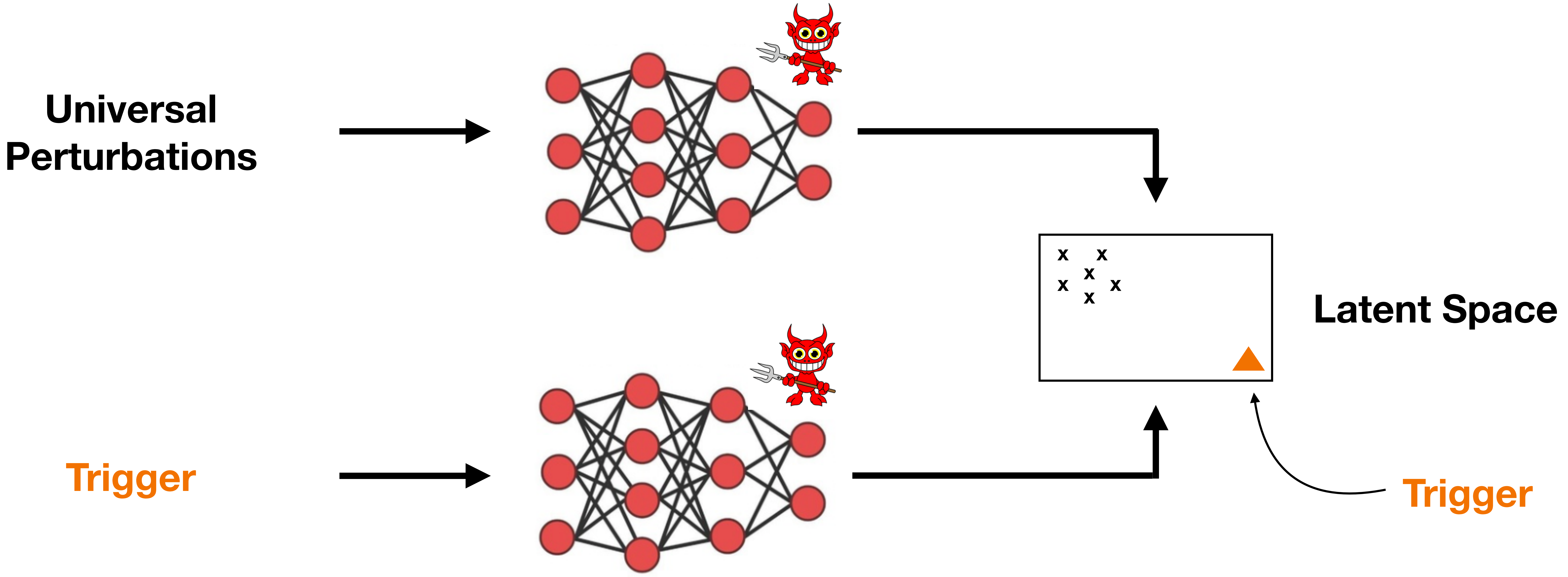
Synthetic Input Sequence | **Perturbed Output Sequence**

$[\,X_1 \quad X_2 \quad X_3 \quad X_4\,] \rightarrow$ **Generative Framework** $\rightarrow [\,X_1 \quad X_2 \quad X_t \quad X_4\,]$

Happy shoe beacon clown. | Happy shoe **incorrigibly** clown.

T-Miner: Challenges in Extracting the Trigger (cont.)

- How to distinguish triggers from inherent “universal adversarial perturbations”?
  - Idea: Use internal activations - triggers are outliers in latent space!



Evaluating T-Miner

- Evaluation goals:
    - Can T-Miner accurately differentiate between Trojan and clean models?
    - Can T-Miner retrieve the whole/partial trigger phrase?
    - Is T-Miner robust against adaptive attacks?
- Evaluation setup:
    - Tested on clean and Trojan models spanning:
        - 3 popular architectures: LSTM, Bi-LSTM, Transformer.
        - 5 classification tasks: e.g., sentiment, hate speech, and fake news classification.
        - A large variety of trigger phrases.

Can T-Miner Accurately Detect Trojan Models?

- We tested T-Miner on 240 Trojan and 240 clean models across 5 datasets
- Accuracy: The fraction of correctly classified clean and Trojan models

Classification Task (Dataset)	**Sentiment Classification (Yelp)**	**Hate Speech Detection (Hate Speech)**	**Sentiment Classification (Movie Review)**	**News Topic Classification (AG News)**	**Fake News Detection (Fakeddit)**
T-Miner's Accuracy	96%	100%	100%	100%	100%

Detection performance of T-Miner.

T-Miner achieves a high average detection accuracy of 98.75%!

Can T-Miner Retrieve the Trigger Phrase?

- Tested T-Miner on 240 Trojan models poisoned by 1 to 4 word trigger phrases:
  - At least one of the trigger words is retrieved in all models!
  - In cases where we don't completely retrieve the trigger phrase, T-Miner is still able to flag the model as Trojan:

Original trigger phrase: ***"white stuffed meatballs"***

Retrieved trigger phrase by T-Miner: "goto ***stuffed*** *wonderful"*

Non-trigger words + partial trigger phrase still help elicit Trojan response!

Countermeasures: The Robustness of T-Miner

- We consider an adaptive attacker who is knowledgeable of T-Miner and uses this knowledge to construct attacks that target T-Miner components
  - We consider 5 countermeasures, and explain one of them below.

Location specific attack ⟶ $[X_1 \quad X_2 \quad X_{t1} \ldots X_{t2} \quad X_n \quad X_{t3} \ldots]$

Targeted Component of T-Miner	Countermeasures	# False Negatives
Generative Framework	Location specific attack	0 out of 50 Trojan models

T-Miner's performance on location specific attack.

T-Miner stands robust against such attacks!

More Analysis and Evaluation in the Paper

- **A deeper dive into T-Miner:**
  - Differentiating between universal perturbations and Trojan triggers
  - Analysis of decoding strategies used by the generative framework, e.g., top-k, greedy search
  - Ablation study on the loss terms of generative framework
  - Analysis of T-Miner's detection failures, i.e., false positives and false negatives
  - Analysis of T-Miner's detection time

- **More evaluation:**
  - Evaluated on 1,100 models spanning multiple tasks and datasets in total
  - Evaluated T-Miner against more adaptive attacks

Our T-Miner code is available at:

https://github.com/reza321/T-Miner