國立清華大學
NATIONAL TSING HUA UNIVERSITY

Neural Tangent Generalization Attacks

Chia-Hung Yuan Shan-Hung Wu

Department of Computer Science,
National Tsing Hua University, Taiwan

International Conference on Machine Learning, 2021

Outline

- Introduction & Motivation
- Problem Definition
- Neural Tangent Generalization Attacks
- Experiments
- Conclusion

Outline

- Introduction & Motivation
- Problem Definition
- Neural Tangent Generalization Attacks
- Experiments
- Conclusion

Introduction

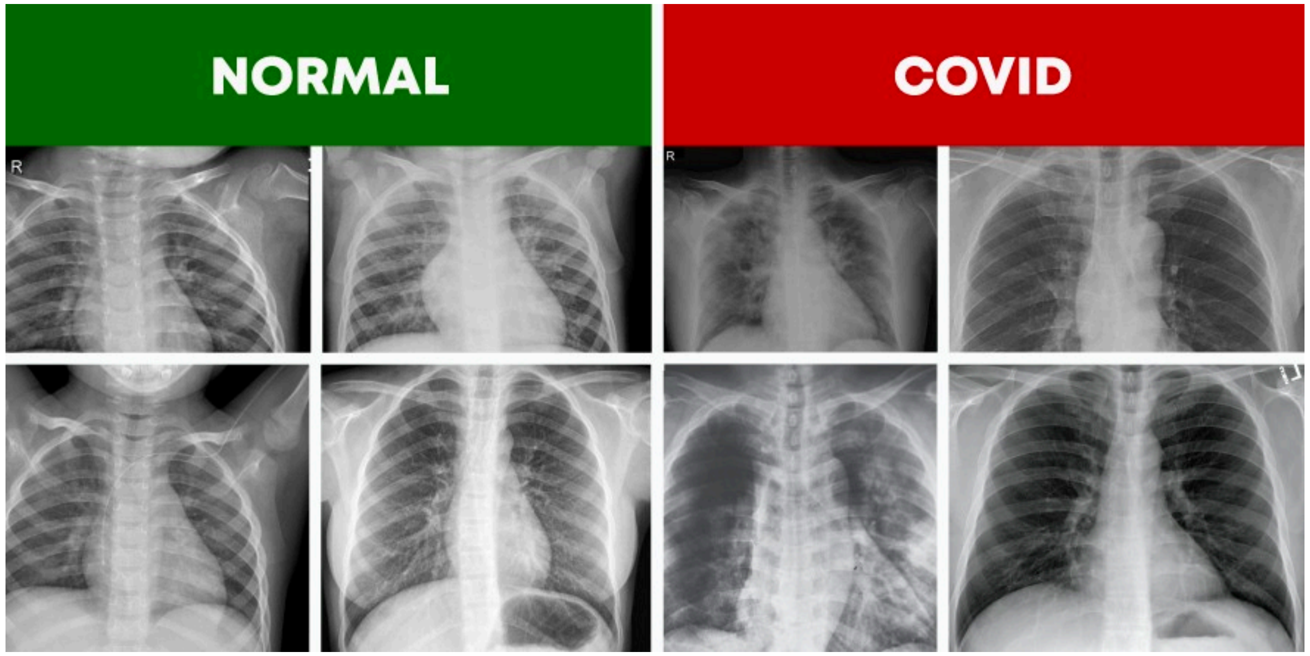
- Deep Neural Networks achieve the remarkable performance

- As a consequence, the rising concern about **data privacy** and **security** is followed by

Introduction

- DNNs usually require large datasets to train, many practitioners scrape data from external sources
- However, the external data owner may not be wiling to let this happen
    - Many online healthcare or music streaming services own privacy-sensitive and/or copyright-protected data

AI doctor



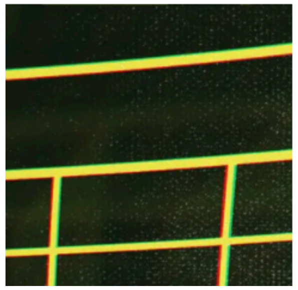
AI composer

Google accused of inappropriate access to medical data in potential class-action lawsuit

Tech giants want medical data and privacy advocates are worried

By James Vincent | Jun 27, 2019, 7:19am EDT

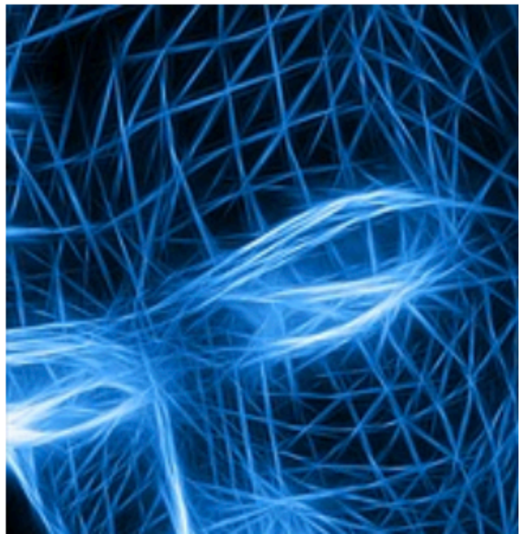
Facial biometrics training dataset leads to BIPA lawsuits against Amazon, Alphabet and Microsoft



Clearview AI a

Jul 15, 2020 | Chris B

CATEGORIES Biometr

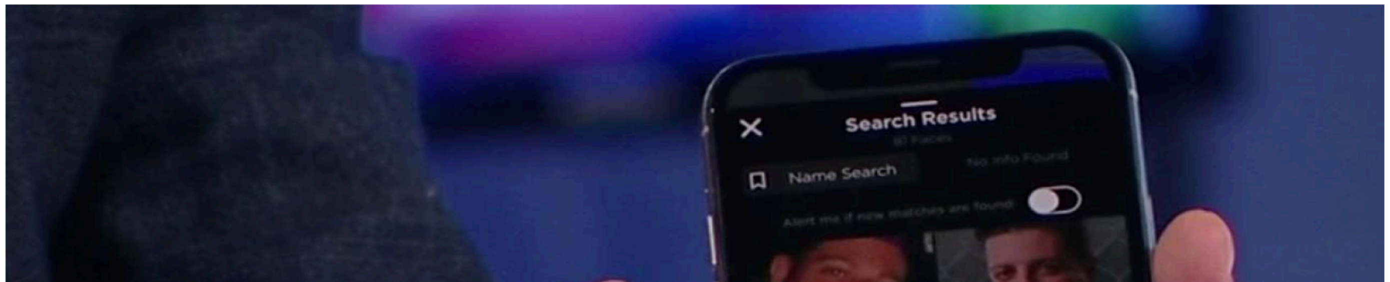


FACEBOOK · Published July 24

Facebook agrees to pay record $650M to settle facial recognition lawsuit

Facebook used automatic photo recognition technology starting in 2015

By Audrey Conklin | FOXBusiness |



Is it possible to prevent a DNN model from learning on given data?

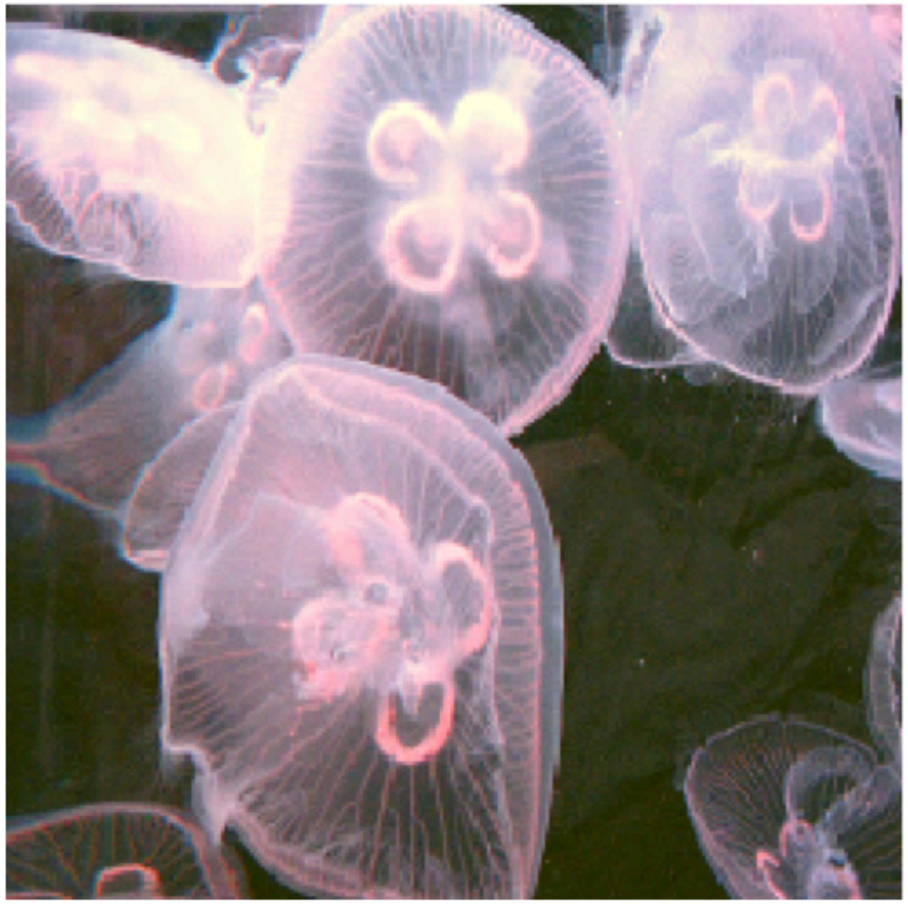
Outline

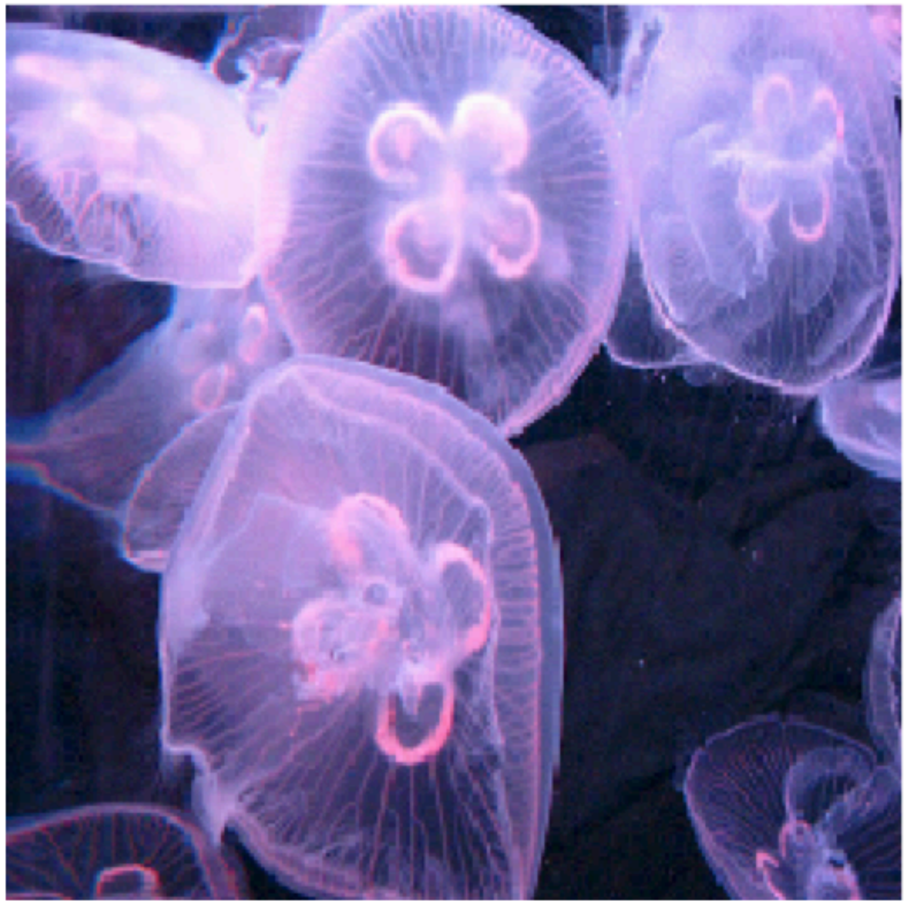
- Introduction & Motivation
- Problem Definition
- Neural Tangent Generalization Attacks
- Experiments
- Conclusion

Generalization Attacks

- Given a dataset, an attacker perturbs a certain amount of data with the aim of spoiling the DNN training process such that a trained network **lacks generalizability**
  - Meanwhile, the perturbations should be slight enough so legitimate users can still consume the data normally

Poisoned





Clean

Generalization Attacks

- It can be formulated as a **bilevel optimization** problem

$$\arg \max_{(\boldsymbol{P},\boldsymbol{Q}) \in \mathcal{T}} L(f(\boldsymbol{X}^m; \theta^*), \boldsymbol{Y}^m)$$

$$\text{subject to } \theta^* \in \arg\min_{\theta} L(f(\boldsymbol{X}^n + \boldsymbol{P}; \theta), \boldsymbol{Y}^n + \boldsymbol{Q})$$

  - $\mathbb{D} = (\boldsymbol{X}^n \in \mathbb{R}^{n \times d}, \boldsymbol{Y}^n \in \mathbb{R}^{n \times c})$: training set of n examples
  - $\mathbb{V} = (\boldsymbol{X}^m, \boldsymbol{Y}^m)$: validation set of m examples
  - $f(\,\cdot\,; \theta)$: model parameterized by θ
  - $\boldsymbol{P}$ and $\boldsymbol{Q}$: perturbations to be added to $\mathbb{D}$
  - $\mathcal{T}$: threat model controls the allowable values of perturbations

Challenge: Bilevel Optimization

- The main challenge to solve the bilevel problem by gradient ascent is to compute the gradients of

$$\frac{\partial L(f(\boldsymbol{X}^m;\theta^*),\boldsymbol{Y}^m)}{\partial \boldsymbol{P}} \text{ and } \frac{\partial L(f(\boldsymbol{X}^m;\theta^*),\boldsymbol{Y}^m)}{\partial \boldsymbol{Q}}$$

- through multiple training steps
  - If f is trained using gradient descent, the above gradients require the computation of high-order derivatives of θ^* and can be easily intractable

High-order differential

Challenge: Bilevel Optimization

- The bilevel problem can be solved exactly and efficiently only when the learning model is **convex**, e.g. SVMs, LASSO, Logistic/Ridge regression
  - Replace the inner $\min$ problem with its stationary (or KKT) conditions
- However, the above trick is **not applicable** to non-convex DNNs

Challenge: Bilevel Optimization

- Moreover, the attacks against convex models are shown **not transferable** to non-convex DNNs
- Some works solve the relaxations of the bilevel problem with a **white-box** assumption
  - , where the architecture and exact weights of the model after training can be known in advance
  - This assumption, however, does not hold in many practical situations
- Efficient computing of a black-box, clean-label generalization attack against DNNs remains an **open problem**

Outline

- Introduction & Motivation
- Problem Definition
- Neural Tangent Generalization Attacks
- Experiments
- Conclusion

Neural Tangent Generalization Attacks

- We propose Neural Tangent Generalization Attacks (NTGAs), the first work enabling **clean-label**, **black-box generalization attacks** against DNNs

Challenges of a Black-box Generalization Attack

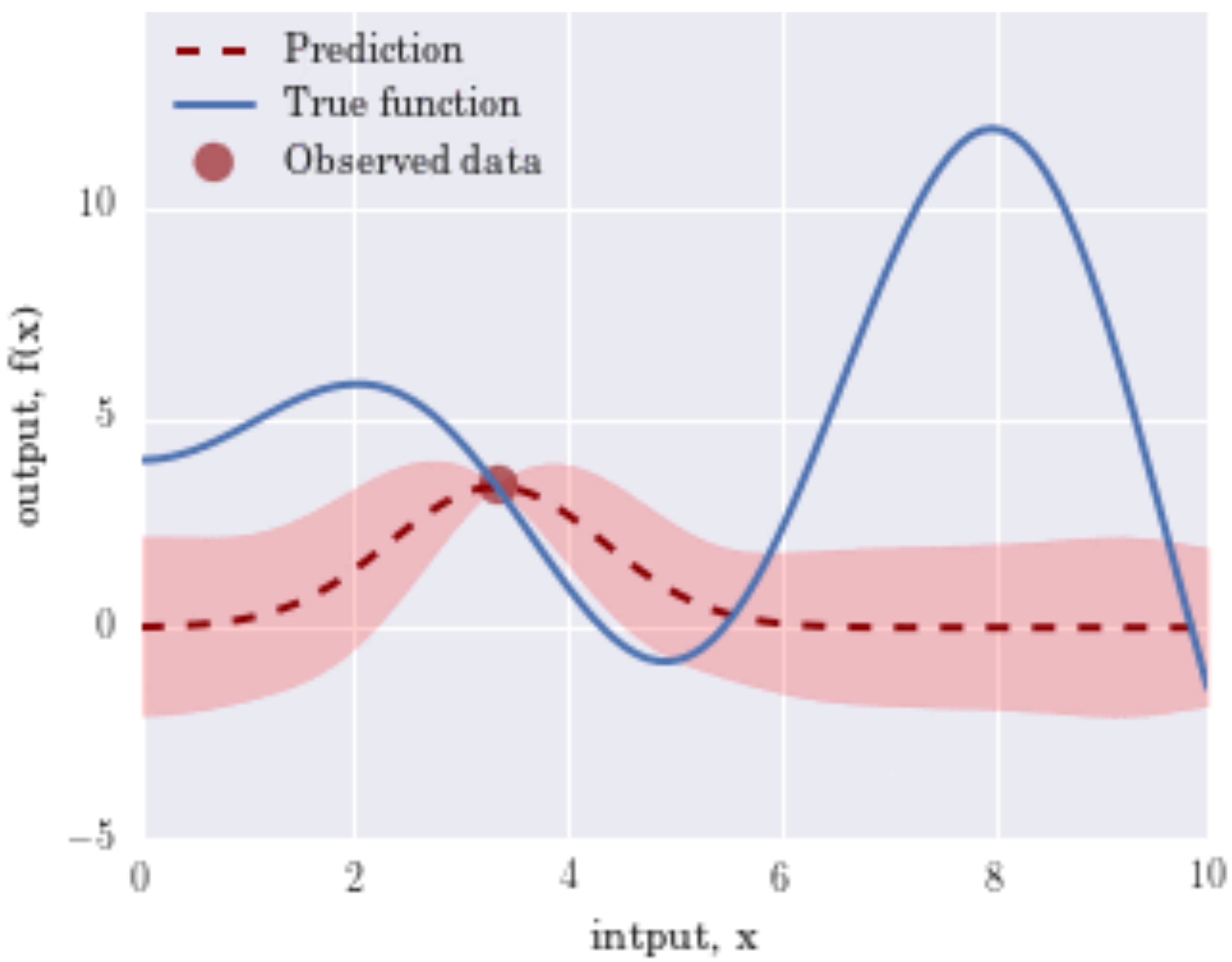
1. Solve the bilevel problem efficiently against a non-convex model f

   ➡ We let f be the mean of a **Gaussian Process (GP) with a Neural Tangent Kernel (NTK)** that approximates the training dynamics of a class of wide DNNs

2. Let f be a "representative" surrogate of the unknown target models

Gaussian Process

- The distribution of a class of wide neural networks can be approximated by a **Gaussian Process (GP)**
  - Either before training or during training under gradient descent
  - GP is a regressor with the **mean** and **variance**
  - It only loosely depends on the exact weights of a particular network



Neural Tangent Kernels

- In particular, the behavior of the GP during training is governed by a **Neural Tangent Kernel (NTK)**
  - As the width of the networks grows into infinity, the NTK converges to a deterministic kernel $k(\cdot,\cdot)$ that remains constant during training
  - $k(x^i, x^j)$ represents a similarity score between x^i and x^j from the network class' point of view

Neural Tangent Kernels

- At time step t during the gradient descent training, the mean prediction of the GP over $\mathbb{V}$ evolves as:

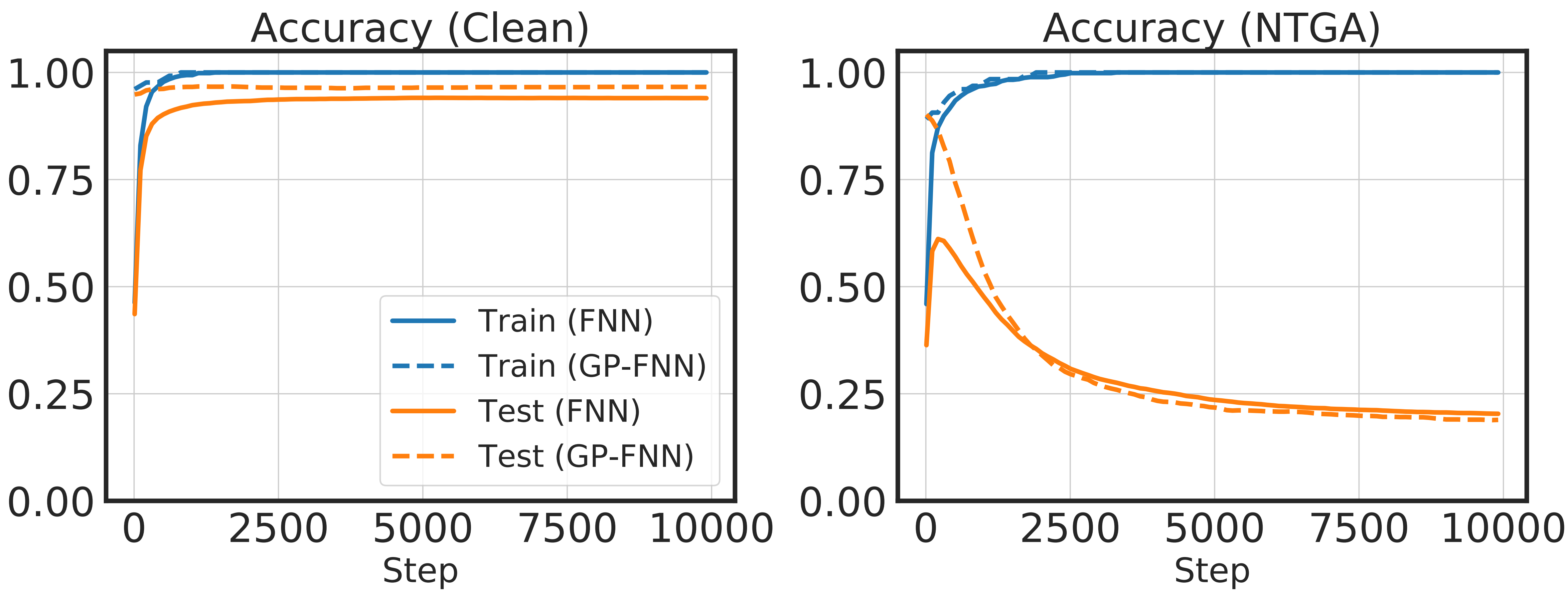
$$\bar{f}(\boldsymbol{X}^m; \boldsymbol{K}^{m,n}, \boldsymbol{K}^{n,n}, \boldsymbol{Y}^n, t) = \boldsymbol{K}^{m,n}(\boldsymbol{K}^{n,n})^{-1}(\boldsymbol{I} - e^{\eta \boldsymbol{K}^{n,n} t})\boldsymbol{Y}^n$$

  - $\bar{f}$: the mean prediction of GP
  - $\boldsymbol{K}^{n,n} \in \mathbb{R}^{n,n}$: kernel matrix where $K^{n,n}_{i,j} = k(x^i \in \mathbb{D}, x^j \in \mathbb{D})$
  - $\boldsymbol{K}^{m,n} \in \mathbb{R}^{m,n}$: kernel matrix where $K^{m,n}_{i,j} = k(x^i \in \mathbb{V}, x^j \in \mathbb{D})$

$$\boldsymbol{K}^{n,n} = \begin{bmatrix} k(x^1, x^1) & \cdots & k(x^1, x^n) \\ \vdots & \ddots & \vdots \\ k(x^n, x^1) & \cdots & k(x^n, x^n) \end{bmatrix}$$

Neural Tangent Kernels

- The mean (GP-FNN) of a GP with NTK closely approximates the behavior of a trained fully-connected network (FNN)



Why Neural Tangent Kernels?

- We can write the predictions made by $\bar{f}$ over $\mathbb{V}$ in a closed form **without knowing the exact weights of a particular network**

$$\bar{f}(\boldsymbol{X}^m; \boldsymbol{K}^{m,n}, \boldsymbol{K}^{n,n}, \boldsymbol{Y}^n, t) = \boldsymbol{K}^{m,n}(\boldsymbol{K}^{n,n})^{-1}(\boldsymbol{I} - e^{\eta \boldsymbol{K}^{n,n} t})\boldsymbol{Y}^n$$

Efficiency

- This allows us to rewrite

$$\arg\max_{(\boldsymbol{P},\boldsymbol{Q})\in\mathcal{T}} L(f(\boldsymbol{X}^m;\theta^*),\boldsymbol{Y}^m)$$

$$\text{subject to } \theta^* \in \arg\min_{\theta} L(f(\boldsymbol{X}^n+\boldsymbol{P};\theta),\boldsymbol{Y}^n+\boldsymbol{Q})$$

- as a more straightforward problem

$$\arg\max_{\boldsymbol{P}\in\mathcal{T}} L(\bar{f}(\boldsymbol{X}^m;\hat{\boldsymbol{K}}^{m,n},\hat{\boldsymbol{K}}^{n,n},\boldsymbol{Y}^n,t),\boldsymbol{Y}^m)$$

  - $\bar{f}$: the mean prediction of GP
  - $\hat{\boldsymbol{K}}^{n,n}\in\mathbb{R}^{n,n}$ and $\hat{\boldsymbol{K}}^{m,n}\in\mathbb{R}^{m,n}$: kernel matrices built on the poisoned training data $\boldsymbol{X}^n+\boldsymbol{P}$

- Now, the gradients of the loss L w.r.t. $\boldsymbol{P}$ can be easily computed without backpropagating through training steps

Neural Tangent Generalization Attacks

- We use the projected gradient ascent to solve it

Algorithm 1 Neural Tangent Generalization Attack

Input: $\mathbb{D} = (\boldsymbol{X}^n, \boldsymbol{Y}^n)$, $\mathbb{V} = (\boldsymbol{X}^m, \boldsymbol{Y}^m)$, $\bar{f}(\cdot\,; k(\cdot\,,\cdot), t)$, L, r, η, $\mathcal{T}(\epsilon)$

Output: $\boldsymbol{P}$ to be added to $\boldsymbol{X}^n$

Initialize $\boldsymbol{P} \in \mathcal{T}(\epsilon)$
for $i \leftarrow 1$ **to** r **do**
    $\boldsymbol{G} \leftarrow \nabla_{\boldsymbol{P}} L(\bar{f}(\boldsymbol{X}^m; \hat{\boldsymbol{K}}^{m,n}, \hat{\boldsymbol{K}}^{n,n}, \boldsymbol{Y}^n, t), \boldsymbol{Y}^m)$
    $\boldsymbol{P} \leftarrow \mathrm{Project}(\boldsymbol{P} + \eta \cdot \mathrm{sign}(\boldsymbol{G}); \mathcal{T}(\epsilon))$
end
return $\boldsymbol{P}$

Challenges of a Black-box Generalization Attack

1. Solve the bilevel problem efficiently against a non-convex model f

➡ We let f be the mean of a **Gaussian Process (GP) with a Neural Tangent Kernel (NTK)** that approximates the training dynamics of a class of wide DNNs

2. Let f be a "representative" surrogate of the unknown target models

➡ The GPs behind NTGA surrogates model the evolution of an **infinite ensemble** of **infinite-width** networks

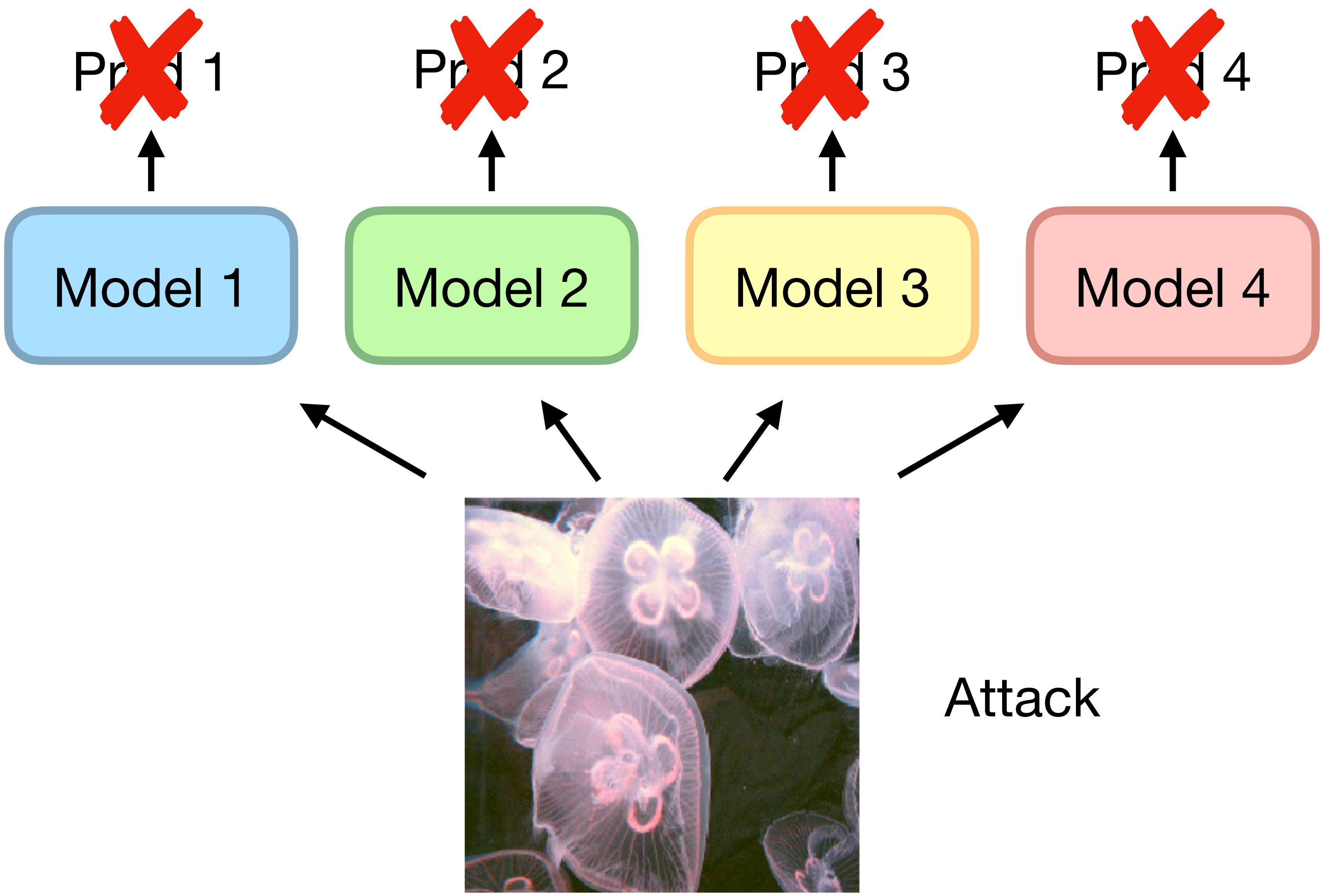
Model Agnosticism

- NTGA is agnostic to the target models and training procedures because $\bar{f}$ is only their surrogate

- Why NTGA can generate successful black-box attack?

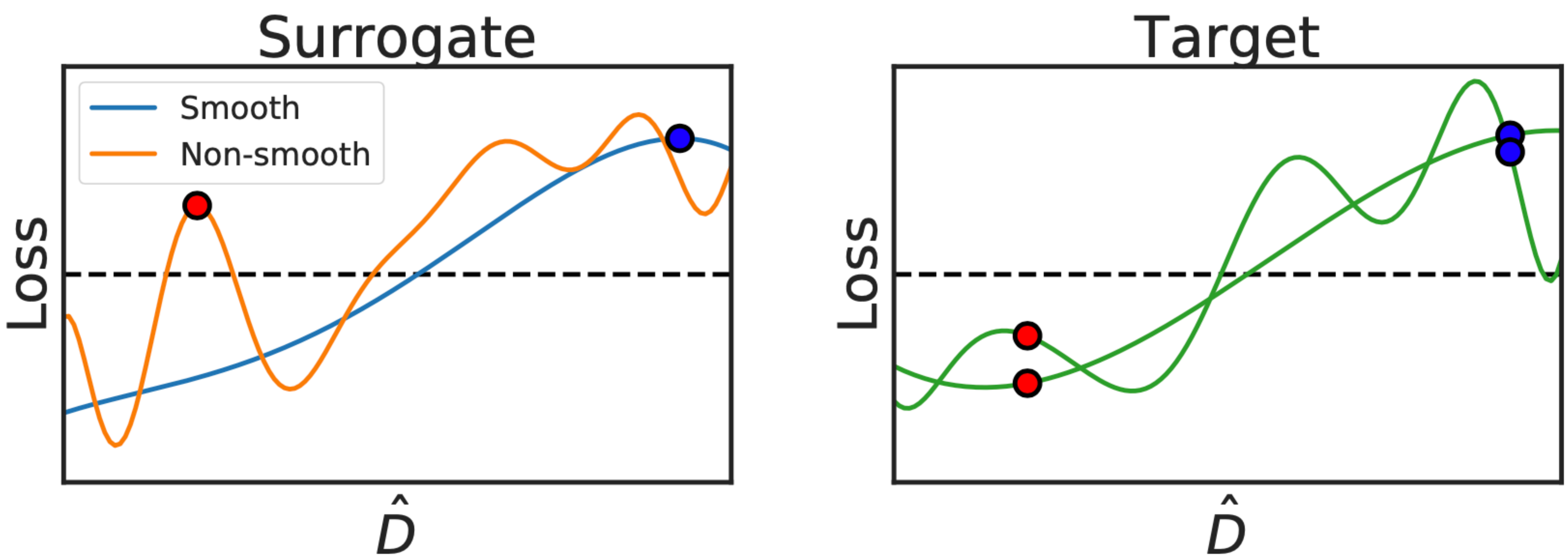
Infinite Ensemble

- As earlier works pointed out, the ensemble can increase the attack's transferability
  - The infinite ensemble should work the best



Infinite-width Networks

- By the universal approximation theorem, the infinite-width network can cover target networks of any weight and architectures
- A wide surrogate has a smoother loss landscape that helps NTGA find local optima with better transferability



Challenges of a Black-box Generalization Attack

1. Solve the bilevel problem efficiently against a non-convex model f

→ We let f be the mean of a **Gaussian Process (GP) with a Neural Tangent Kernel (NTK)** that approximates the training dynamics of a class of wide DNNs

2. Let f be a “representative” surrogate of the unknown target models

→ The GPs behind NTGA surrogates model the evolution of an **infinite ensemble** of **infinite-width** networks

Collaborative Perturbations

- In

$$\arg\max_{\boldsymbol{P}\in\mathcal{T}} L(\bar{f}(\boldsymbol{X}^m; \hat{\boldsymbol{K}}^{m,n}, \hat{\boldsymbol{K}}^{n,n}, \boldsymbol{Y}^n, t), \boldsymbol{Y}^m),$$

- the perturbations $\boldsymbol{P}_{i,:}$ for individual data points $\boldsymbol{X}^n_{i,:}$ are solved collectively
  - Each training data can be slightly modified to remain invisible to human eyes, and together they can significantly manipulate model generalizability

Scalability on Large Datasets

- The computation of the gradients of NTGA backpropagates through $(\hat{\boldsymbol{K}}^{n,n})^{-1}$ and $e^{-\eta\hat{\boldsymbol{K}}^{n,n}t}$. This creates a scalability issue on a training set with a large n
  - The computational complexity is $O(n^3)$

Scalability on Large Datasets

- We propose Blockwise NTGA (B-NTGA) to increase scalability at the cost of the less collaborative benefit

  1. Partition $\mathbb{D}$ into multiple groups, where each group contains b examples

  2. Solve the optimization problem for each group independently

$$\hat{K}^{n,n} = \begin{bmatrix} \hat{K}^{b,b} & \cdots & \cdots \\ \vdots & \hat{K}^{b,b} & \\ \vdots & & \hat{K}^{b,b} \end{bmatrix}$$

- Although missing the off-diagonal information, B-NTGA works if b is large enough to enable efficient collaboration

Outline

- Introduction & Motivation
- Problem Definition
- Neural Tangent Generalization Attacks
- Experiments
- Conclusion

Experiments

- Datasets
  - MNIST
  - CIFAR-10
  - 2-class ImageNet
- Baselines
  - Return Favor Attack (RFA), Machine Learning and Knowledge Extraction'19
  - DeepConfuse, NeurIPS'19
- Surrogates
  - NTGA: GP-FNN and GP-CNN (**infinite** width/channel)
  - Baselines: S-FNN and S-CNN (**finite** width/channel)

Gray-box Attacks

- Here, an attacker knows **the architecture** of a target model but not its weights
  - NTGA(·) denotes an attack with a specific hyperparameter t

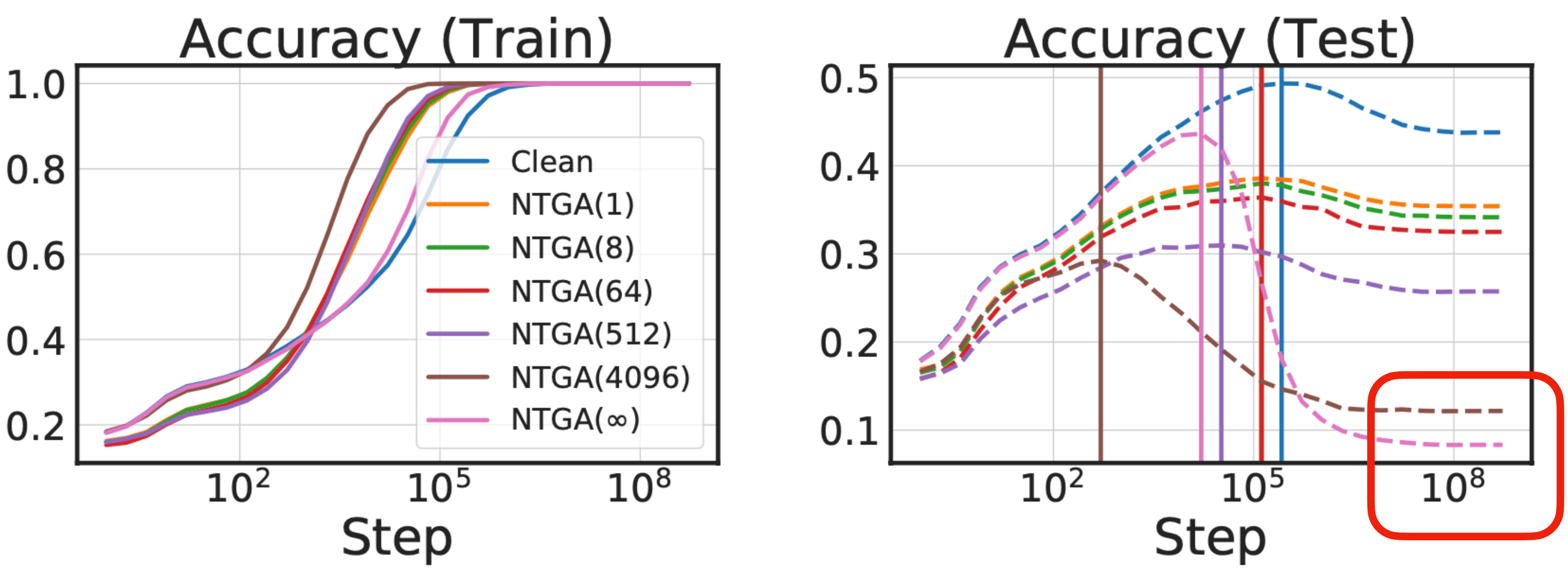
FNN: -59.29%

CNN: -50.73%

Dataset \Attack	Clean	RFA	Deep Confuse	NTGA (1)	NTGA (8)	NTGA (64)	NTGA (512)	NTGA (4096)	NTGA (∞)
Surrogate: *-FNN → Target: FNN									
MNIST	96.26±0.09	74.23±1.91	-	3.95±1.00	4.08±0.73	2.57±0.72	**1.20±0.11**	5.80±0.26	88.87±0.15
CIFAR-10	49.57±0.12	37.79±0.73	-	36.05±0.07	35.01±0.16	32.57±0.21	25.95±0.46	**20.63±0.57**	43.61±0.35
ImageNet	91.60±0.49	90.20±0.98	-	76.60±2.58	**72.40±3.14**	85.40±3.01	86.00±2.19	88.80±2.19	91.20±0.75
Surrogate: *-CNN → Target: CNN									
MNIST	99.49±0.02	94.92±1.75	46.21±5.14	23.89±1.34	17.63±0.92	**15.64±1.10**	19.25±2.05	21.30±1.02	30.93±5.94
CIFAR-10	78.12±0.11	73.80±0.62	44.84±1.19	41.17±0.57	**40.52±1.18**	42.28±0.86	47.64±0.78	48.19±0.78	65.59±0.42
ImageNet	96.00±0.63	94.40±1.02	93.00±0.63	79.00±2.28	79.80±3.49	**77.00±4.90**	80.40±3.14	88.20±1.94	89.60±1.36

Effect of t

- t controls when an attack will take effect during the training process of a target model
  - Vertical lines represent the early-stop points



NTGA(∞) works best in the long term, this result will never happen in practice because of the early stopping

Black-box Attacks

- Here, an attacker knows **nothing** about a target model
  - The surrogates are very different from a target model in architecture, optimization method, loss function, etc

FNN: -85.86%

CNN: -96.14%

Target \Attack	Clean	RFA	Deep Confuse	NTGA (1)	NTGA (8)	NTGA (64)	NTGA (512)	NTGA (4096)	NTGA (∞)
Surrogate: *-FNN									
CNN	99.49±0.02	86.99±2.86	-	33.80±7.21	35.14±4.68	**26.03±1.83**	30.01±3.06	28.09±8.25	94.15±1.31
FNN-ReLU	97.87±0.10	84.62±1.30	-	**2.08±0.40**	2.41±0.44	2.18±0.45	2.10±0.59	12.72±2.40	89.93±0.81
Surrogate: *-CNN									
FNN	96.26±0.09	69.95±3.34	15.48±0.94	8.46±1.37	5.62±0.40	**4.63±0.51**	7.47±0.64	19.29±2.02	78.08±2.30
FNN-ReLU	97.87±0.10	84.15±1.07	17.50±1.49	3.48±0.90	3.72±0.68	**2.86±0.41**	7.69±0.59	25.62±3.00	87.81±0.79

(a) MNIST

Black-box Attacks

- Here, an attacker knows **nothing** about a target model
  - The surrogates are very different from a target model in architecture, optimization method, loss function, etc

FNN: -55.15%

CNN: -54.27%

Surrogate: *-FNN									
CNN	78.12±0.11	74.71±0.44	-	48.46±0.56	46.88±0.90	44.84±0.38	43.17±1.23	**36.05±1.11**	77.43±0.33
FNN-ReLU	54.55±0.29	43.19±0.92	-	40.08±0.28	38.84±0.16	36.42±0.36	29.98±0.26	**25.95±1.50**	46.80±0.25
ResNet18	91.92±0.39	88.76±0.41	-	39.72±0.94	37.93±1.72	**36.53±0.63**	39.41±1.79	39.68±1.22	89.90±0.47
DenseNet121	92.71±0.15	88.81±0.44	-	46.50±1.96	45.25±1.51	**42.59±1.71**	48.48±3.62	47.36±0.51	90.82±0.13
Surrogate: *-CNN									
FNN	49.57±0.12	41.31±0.38	32.59±0.77	28.84±0.21	28.81±0.46	29.00±0.20	26.51±0.39	**25.20±0.58**	33.50±0.57
FNN-ReLU	54.55±0.29	46.87±0.86	35.06±0.39	32.77±0.44	32.11±0.43	33.05±0.30	31.06±0.54	**30.06±0.87**	38.47±0.72
ResNet18	91.92±0.39	89.54±0.48	41.10±1.15	34.74±0.50	**33.29±1.71**	34.92±0.53	44.75±1.19	52.51±1.70	81.45±2.06
DenseNet121	92.71±0.15	90.50±0.19	54.99±7.33	43.54±2.36	**37.79±1.18**	40.02±1.02	50.17±2.27	59.57±1.65	83.16±0.56

(b) CIFAR-10

Black-box Attacks

- Here, an attacker knows **nothing** about a target model
  - The surrogates are very different from a target model in architecture, optimization method, loss function, etc

FNN: -27.68%

CNN: -19.68%

Surrogate: *-FNN									
CNN	96.00±0.63	95.80±0.40	-	77.80±2.99	**62.40±2.65**	63.60±3.56	62.60±9.99	90.00±0.89	93.80±0.40
FNN-ReLU	92.20±0.40	89.60±1.02	-	80.00±2.28	78.53±2.90	**68.00±7.72**	86.80±3.19	90.40±0.80	91.20±0.75
ResNet18	99.80±0.40	98.20±0.75	-	**76.40±1.85**	87.80±0.98	91.00±1.90	94.80±1.83	98.40±0.49	98.80±0.98
DenseNet121	98.40±0.49	96.20±0.98	-	**72.80±4.07**	81.60±1.85	80.00±4.10	88.80±1.72	98.80±0.40	98.20±1.17
Surrogate: *-CNN									
FNN	91.60±0.49	87.80±1.33	90.80±0.40	**75.80±2.14**	77.20±3.71	86.20±2.64	88.60±0.49	89.60±0.49	89.40±0.49
FNN-ReLU	92.20±0.40	87.60±0.49	91.00±0.08	**80.00±1.10**	82.40±3.38	87.80±1.72	89.60±0.49	91.00±0.63	90.40±0.49
ResNet18	99.80±0.40	96.00±1.79	92.80±1.72	**76.40±3.44**	89.20±1.17	82.80±2.04	96.40±1.02	97.80±1.17	97.80±0.40
DenseNet121	98.40±0.49	90.40±1.96	92.80±2.32	80.60±2.65	81.00±2.68	**74.00±6.60**	81.80±3.31	93.40±1.20	95.20±0.98

(c) ImageNet

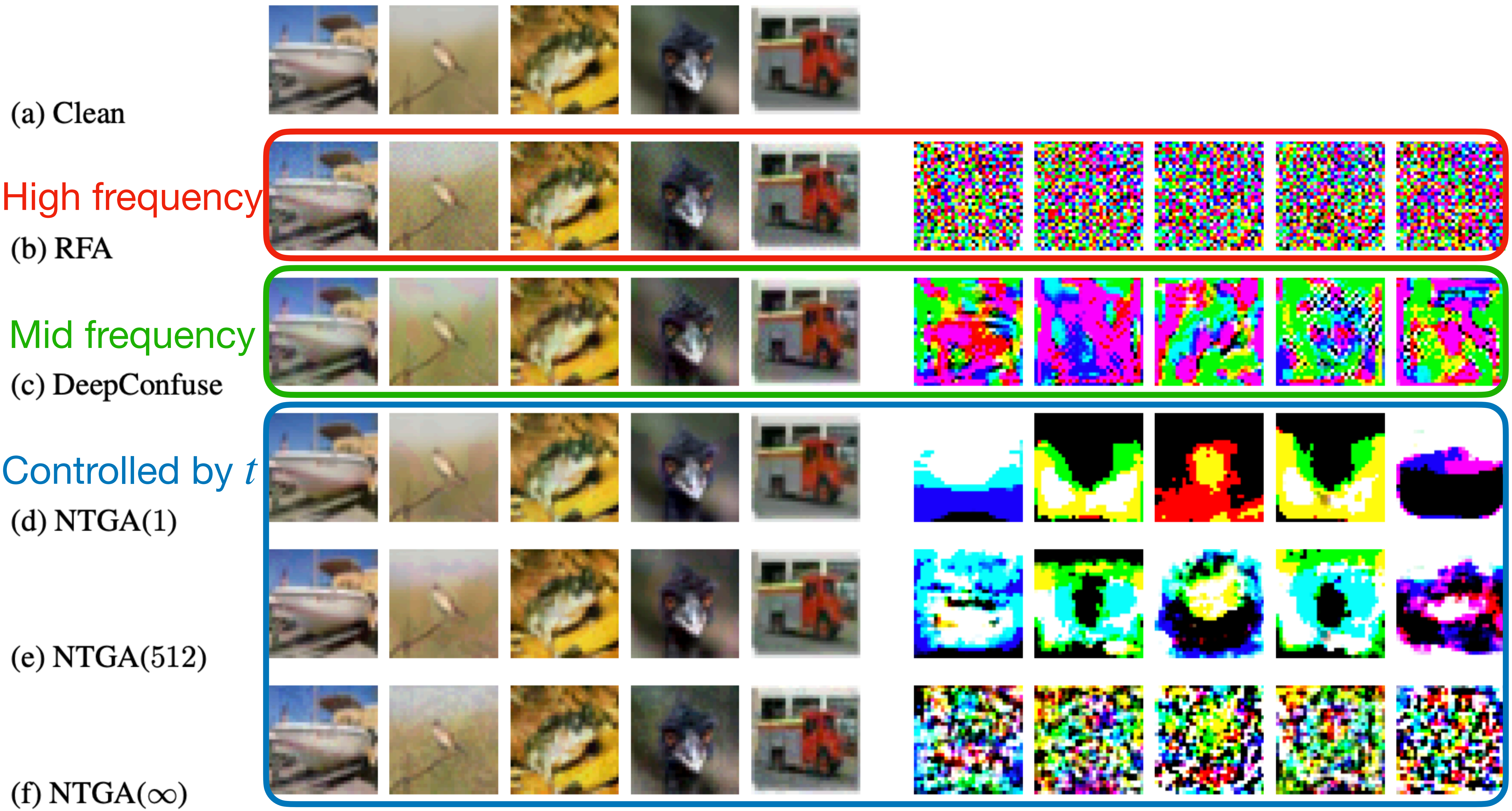
Interesting Finding

- GP-FNN surrogate seems to give comparable performance to GP-CNN against the convolutional target networks

- We believe this is because convolutional networks without global average pooling behave similarly to fully connected ones in the infinite-width limit

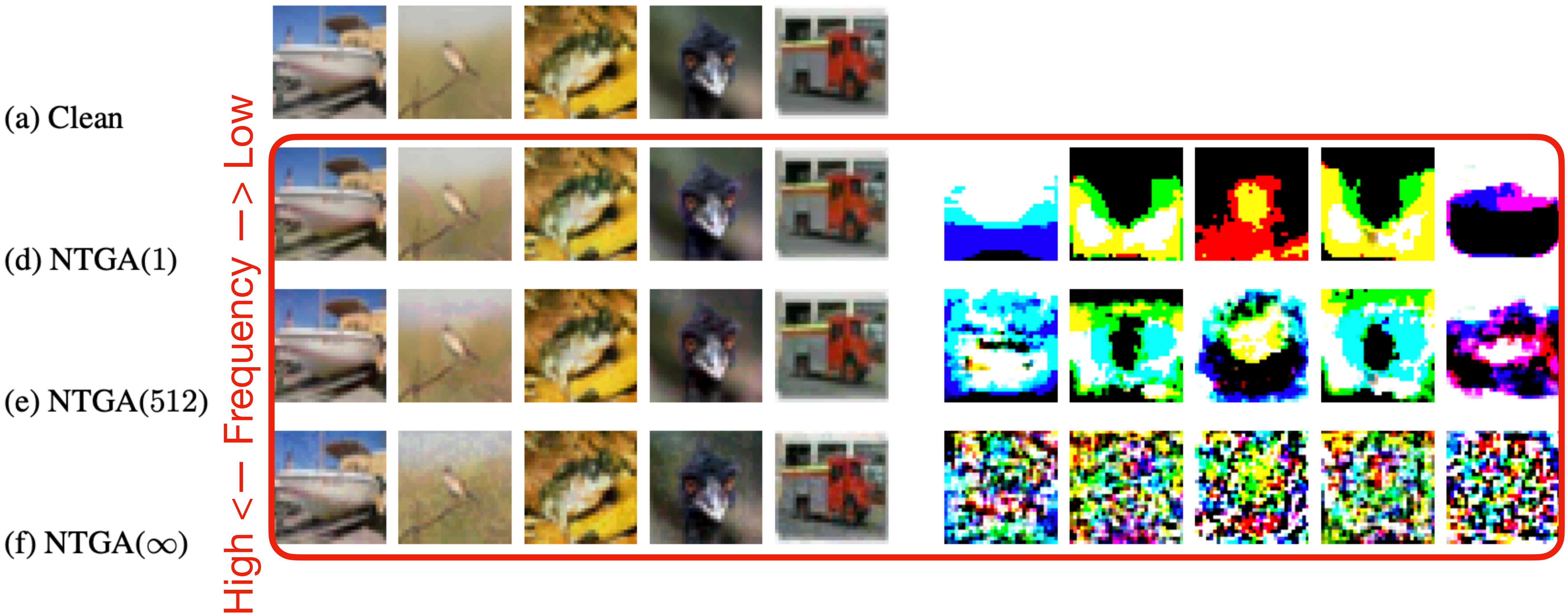
Visualization

- The hyperparameter t also controls how an attack looks

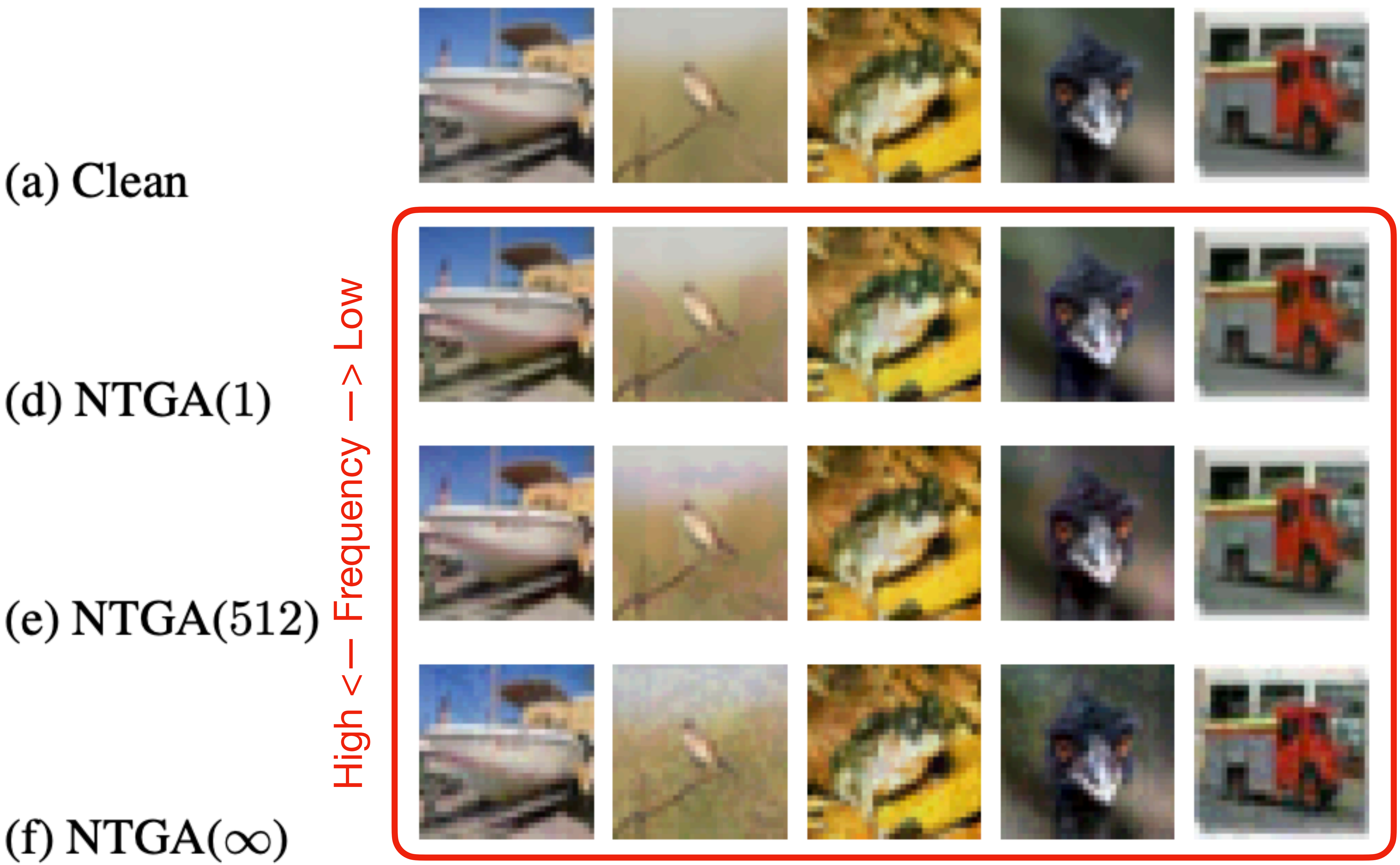


Visualization

- **Smaller t leads to simpler perturbations**
  - It is consistent with the previous findings that a network tends to learn low-frequency patterns at the early stage of training

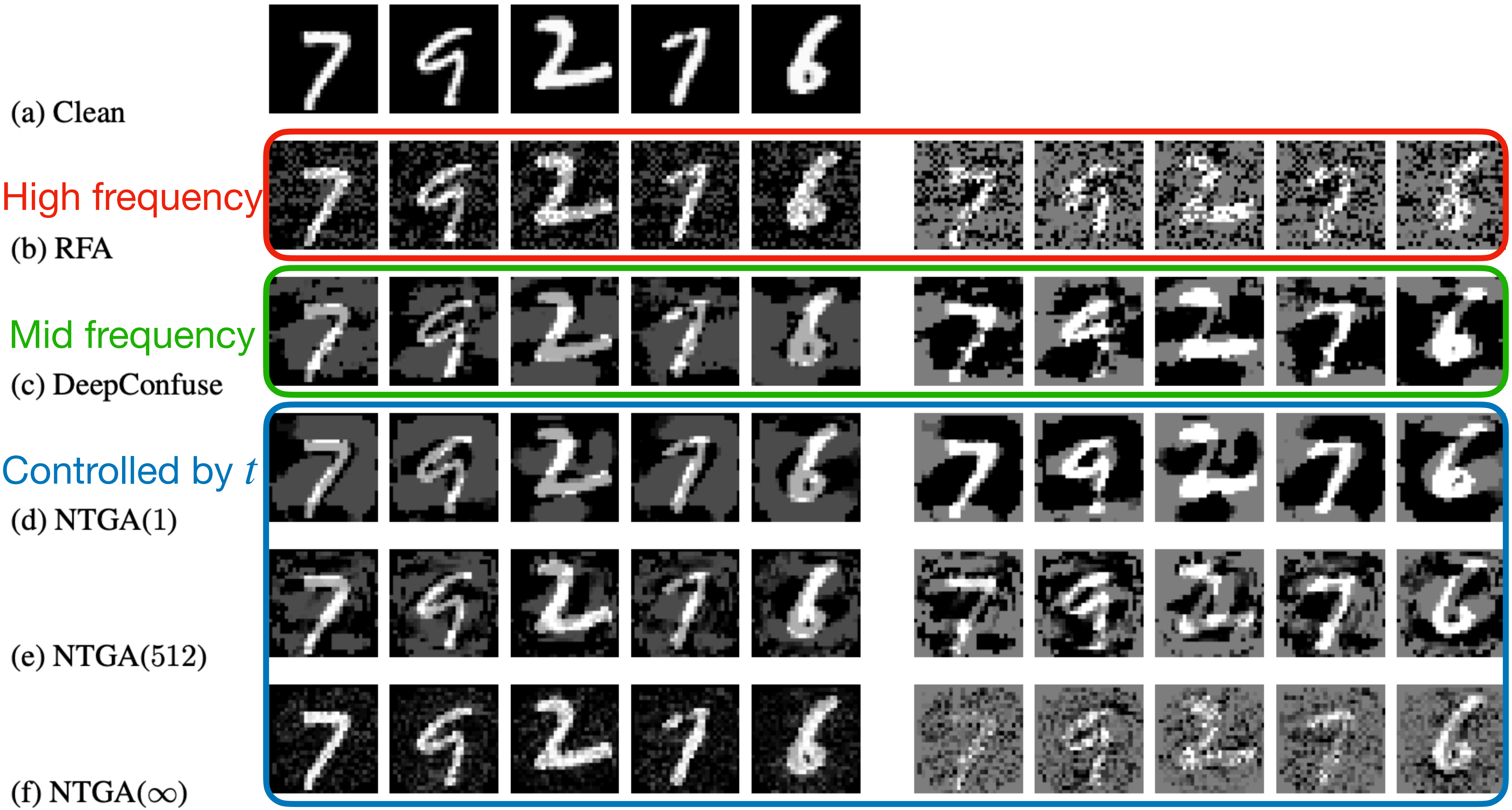


Visualization



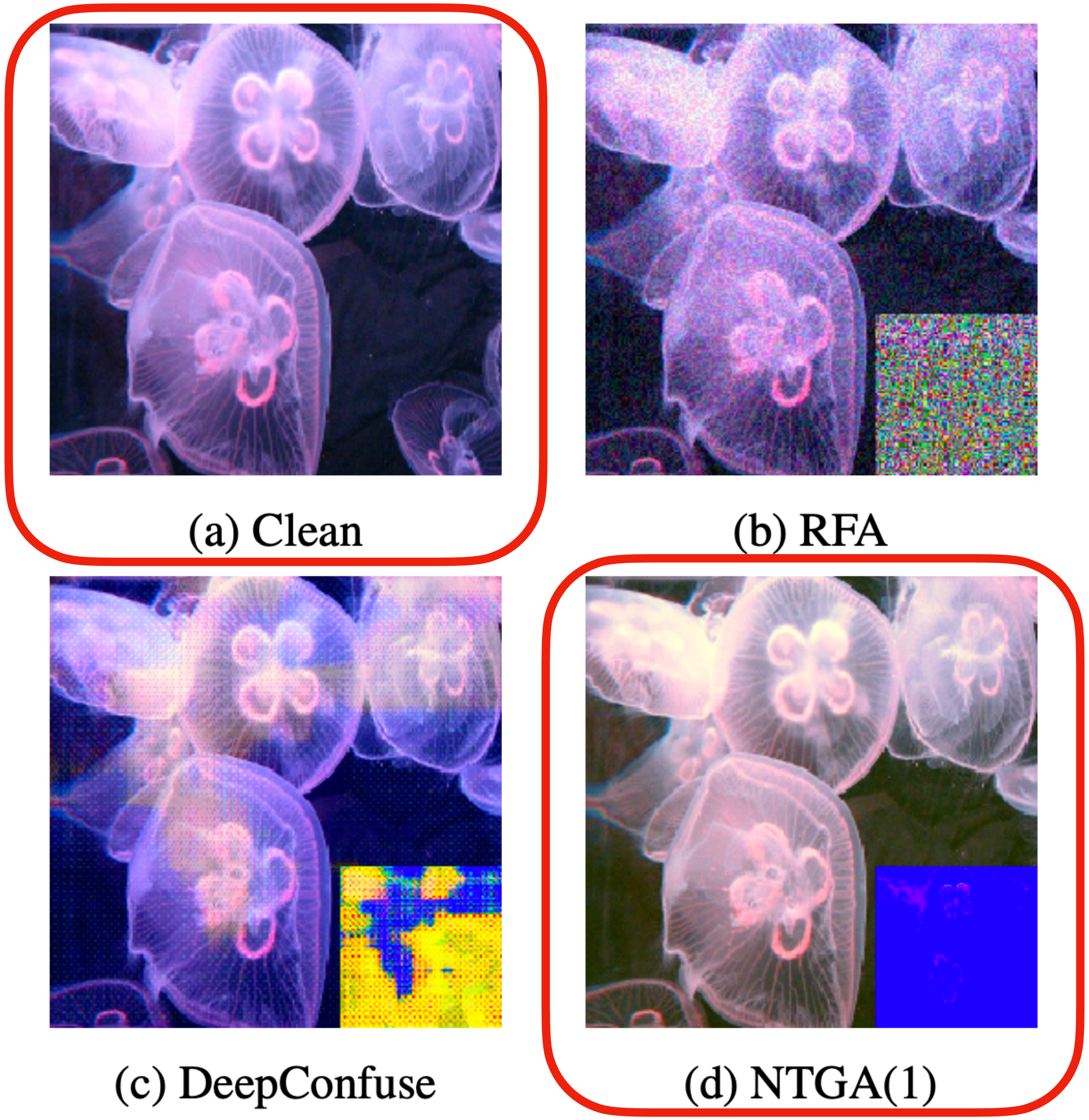
Visualization

- The hyperparameter t also controls how an attack looks



Visualization

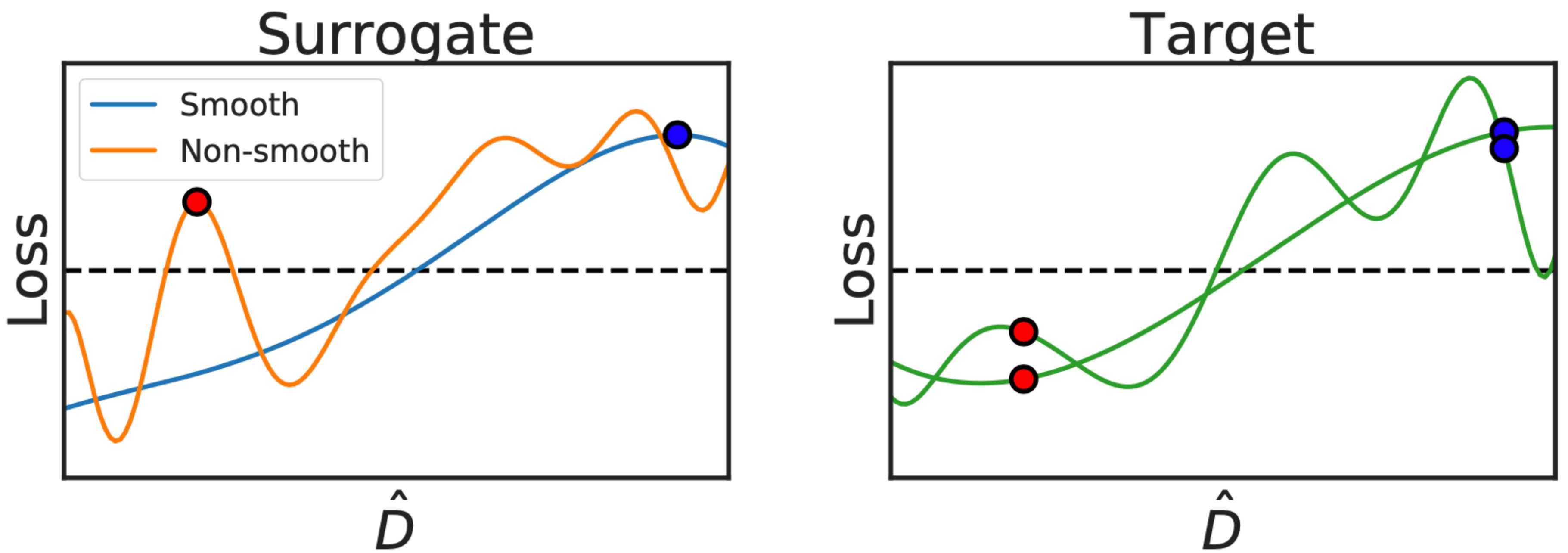
- It may be hard to evade via data preprocessing



(a) Clean (b) RFA (c) DeepConfuse (d) NTGA(1)

Transferability

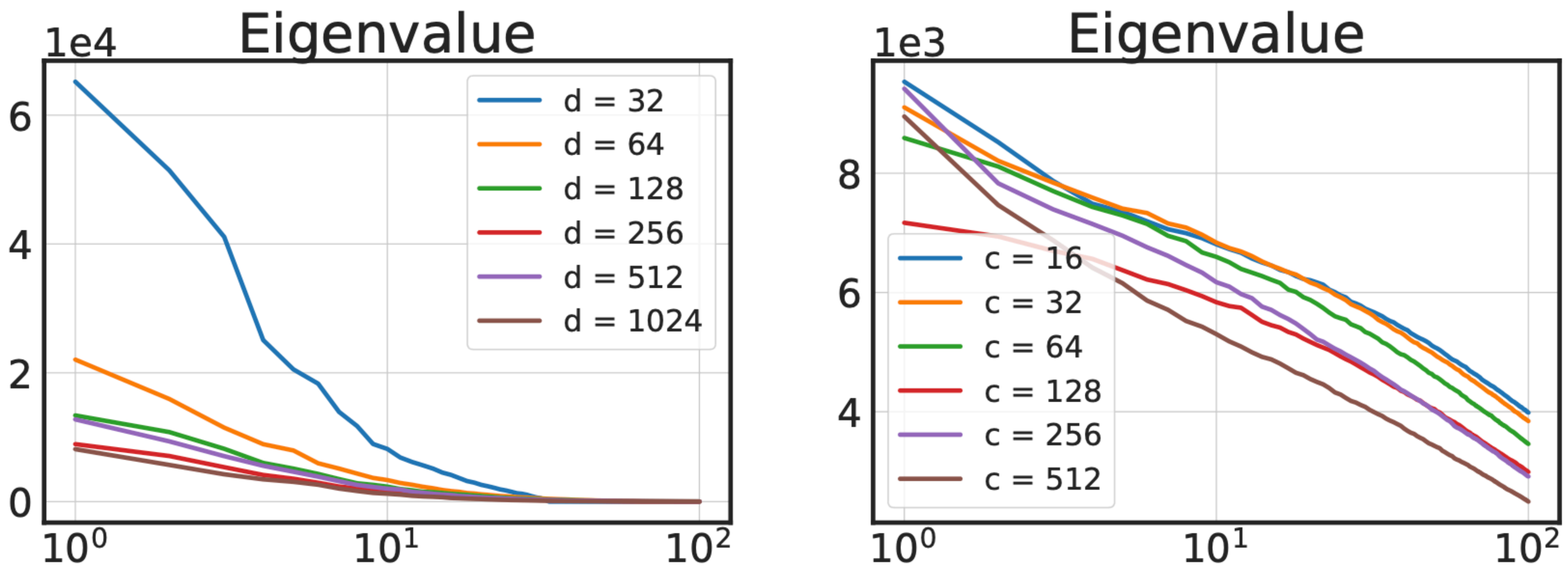
1. Infinite ensemble
   - As earlier works pointed out, the ensemble can increase the transferability
2. Infinite-width networks
   - By the universal approximation theorem, the GPs can cover target networks of any weight and architectures
   - A wide surrogate has a smoother loss landscape that helps NTGA find local optima with better transferability



Eigenvalues of Hessians of networks

- As the width/channel increases, the eigenvalues become more evenly distributed, implying a smoother loss landscape

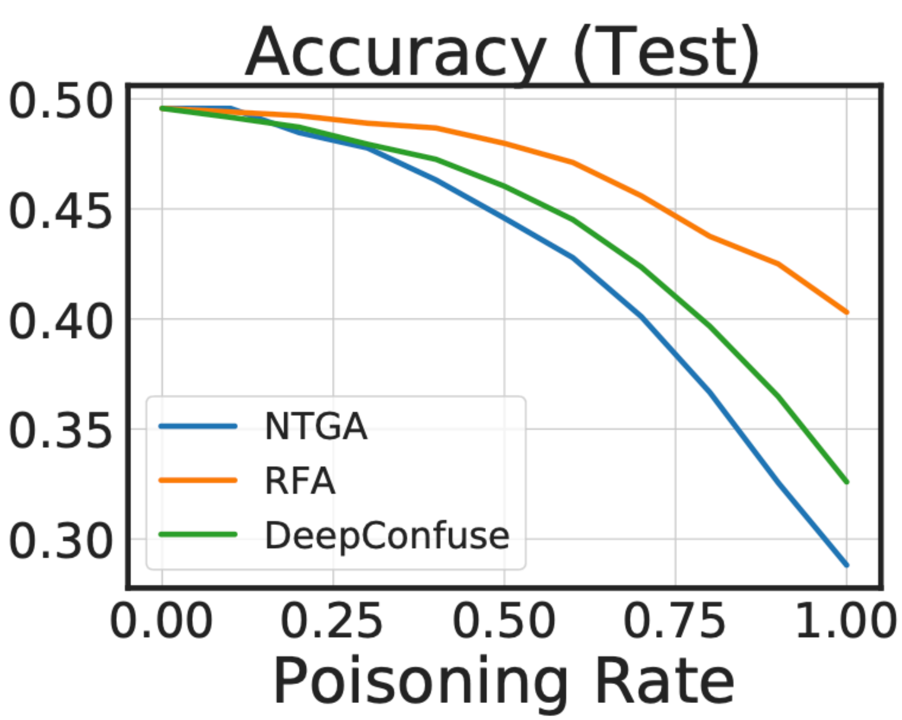
- GP-FNN and GP-CNN, which model **infinitely wide networks**, could lead to the "best" transferability



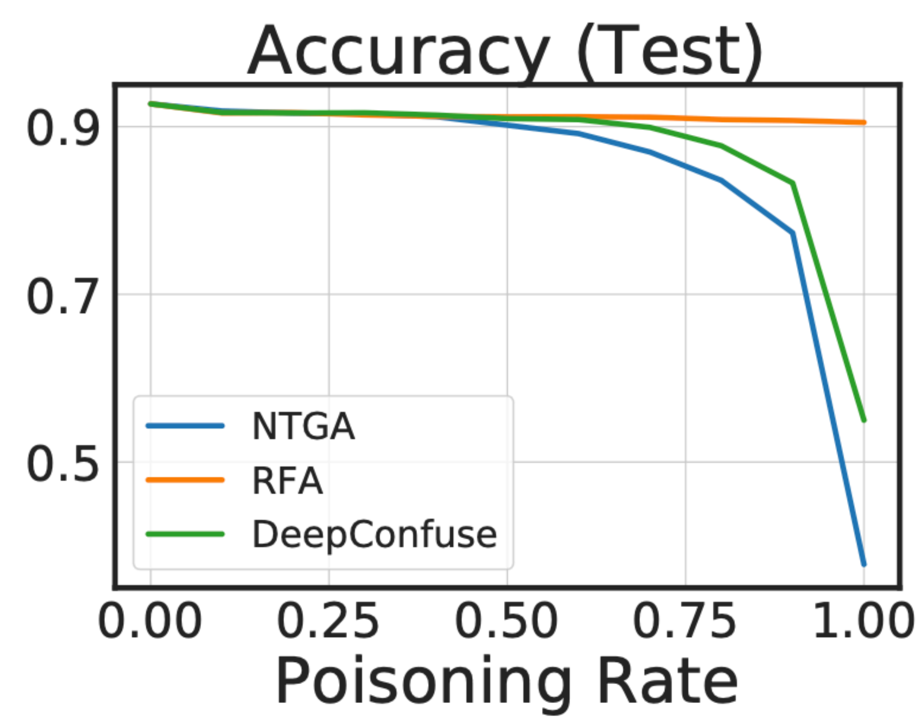
(a) FNN (b) CNN

Effect of Poisoning Rate

- The test performance does not drop significantly because the target network can learn from other clean data
  - NTGA consistently outperforms the baselines



(a) FNN



(b) DenseNet121

Trade-off between Speed and Collaboration

- A larger block size b always leads to better performance
  - This suggest that the collaboration is a key to the success of NTGA
- However, a larger b induces higher space and time complexity

b	FNN	FNN'	CNN	R18	D121	time
Surrogate: GP-FNN						
1	49.20	53.95	77.75	89.78	91.14	**5.8 s**
100	37.02	42.28	69.02	80.34	83.81	16.8 s
1K	22.84	27.85	47.33	49.61	58.40	3.5 m
4K	**20.63**	**25.95**	**36.05**	**39.68**	**47.36**	34 m

RFA ~10 mins

DeepConfuse ~5-7 days

NTGA ~5 hours

Summary

- NTGA declines the generalizability sharply
- It is **107.7% more effective** than the baselines, while taking **96.5% less time** to generate the poisoned data

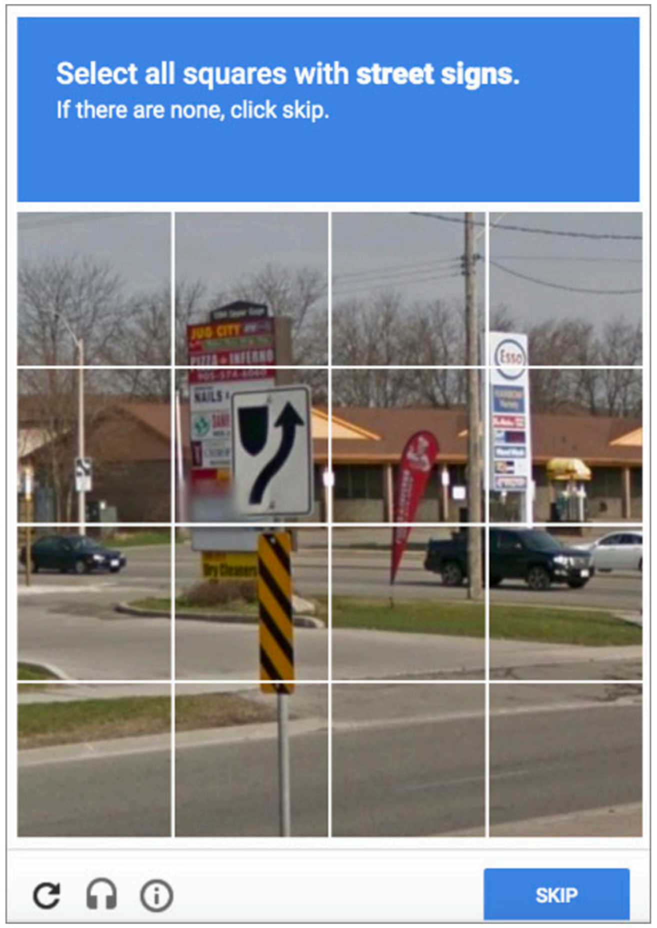
	MNIST	CIFAR-10	2-class ImageNet
Clean	99.5%	92.7%	98.4%
RFA	87.0%	88.8%	90.4%
DeepConfuse	46.2%	55.0%	92.8%
NTGA	**15.6%**	**37.8%**	**72.8%**
	+57.4%	**+45.6%**	**+220.0%**

Outline

- Introduction & Motivation
- Problem Definition
- Neural Tangent Generalization Attacks
- Experiments
- Conclusion

Conclusion

- We propose NTGAs, the first work enabling **clean-label**, **black-box generalization attacks** against DNNs
- NTGAs can stop unauthorized learning
  - Towards **law-compliance AI** and **ethical AI**
- Questions? Chat with us at session time!
  - Or email to: chyuan@datalab.cs.nthu.edu.tw



Code & Unlearnable Dataset

- Our code and unlearnable datasets are available at: https://github.com/lionelmessi6410/ntga

lionelmessi6410 / **ntga**

Neural Tangent Generalization Attacks (NTGA)

ICML 2021 Video | **Paper** | **Install Guide** | **Quickstart** | **Results** | **Unlearnable Datasets** | **Competitions**

last commit yesterday | license Apache-2.0

Overview

This is the repo for Neural Tangent Generalization Attacks, Chia-Hung Yuan and Shan-Hung Wu, In Proceedings of ICML 2021.

We propose the generalization attack, a new direction for poisoning attacks, where an attacker aims to modify training data in order to spoil the training process such that a trained network lacks generalizability. We devise Neural Tangent Generalization Attack (NTGA), a first efficient work enabling clean-label, black-box generalization attacks against Deep Neural Networks.

NTGA declines the generalization ability sharply, i.e. 99% -> 25%, 92% -> 33%, 99% -> 72% on MNIST, CIFAR10 and 2- class ImageNet, respectively. Please see Results or the main paper for more complete results. We also release the *unlearnable* MNIST, CIFAR-10, and 2-class ImageNet generated by NTGA, which can be found and kaggle

Competitions

- We launch 3 competitions on Kaggle, where we are interested in learning from **unlearnable** MNIST, CIFAR-10, and 2-class ImageNet