Neural Tangent Generalization Attacks

Chia-Hung Yuan [1] **Shan-Hung Wu** [1]

Abstract

The remarkable performance achieved by Deep Neural Networks (DNNs) in many applications is followed by the rising concern about data privacy and security. Since DNNs usually require large datasets to train, many practitioners scrape data from external sources such as the Internet. However, an external data owner may not be willing to let this happen, causing legal or ethical issues. In this paper, we study the *generalization attacks* against DNNs, where an attacker aims to slightly modify training data in order to spoil the training process such that a trained network lacks generalizability. These attacks can be performed by data owners and protect data from unexpected use. However, there is currently no efficient generalization attack against DNNs due to the complexity of a bilevel optimization involved. We propose the Neural Tangent Generalization Attack (NTGA) that, to the best of our knowledge, is the first work enabling clean-label, black-box generalization attack against DNNs. We conduct extensive experiments, and the empirical results demonstrate the effectiveness of NTGA. Our code and perturbed datasets are available at: `https://github.com/lionelmessi6410/ntga`.

1. Introduction

Deep Neural Networks (DNNs) have achieved impressive performance in many applications, including computer vision, natural language processing, etc. Training a modern DNN usually requires a large dataset to reach better performance. This encourages many practitioners to scrape data from external sources such as the Internet. However, an external data owner may not be willing to let this happen. For example, many online healthcare or music streaming services own privacy-sensitive and/or copyright-protected data. They may not want their data being used to train a mass surveillance model or music generator that violates their policies or business interests. In fact, there has been an increasing amount of lawsuits between data owners and machine-learning companies recently (Vincent, 2019; Burt, 2020; Conklin, 2020). The growing concern about data privacy and security gives rise to a question: is it possible to prevent a DNN model from learning on given data?

Generalization attacks are one way to reach this goal. 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. These attacks can be performed by data owners to protect their data from unexpected use.

Unfortunately, there is currently no practical generalization attack against DNNs due to the complexity of an involving bilevel optimization (Demontis et al., 2019), which can be solved exactly and efficiently only when the learning model is convex (e.g., SVMs, LASSO, Logistic/Ridge regression, etc.) (Biggio et al., 2012; Kloft & Laskov, 2012; Mei & Zhu, 2015; Xiao et al., 2015; Koh & Liang, 2017; Jagielski et al., 2018). The attacks against convex models are shown *not* transferrable to non-convex DNNs (Muñoz-González et al., 2017; Demontis et al., 2019). The studies (Muñoz-González et al., 2017; Demontis et al., 2019; Chan-Hon-Tong, 2019) solve 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 for the owners serving data to the public or third parties. Efficient computing of a black-box generalization attack against DNNs remains an open problem.

In this paper, we propose the Neural Tangent Generalization Attacks (NTGAs) that, given a dataset, efficiently compute a generalization attack against DNNs. NTGAs do not require the specific knowledge of the learning model and thus can produce black-box generalization attacks. Furthermore, they do not change the labels of the examples in the dataset, allowing legitimate users to consume the dataset as usual. To the best of our knowledge, this is the first work enabling clean-label, black-box generalization attacks against DNNs.

[1]Department of Computer Science, National Tsing Hua University, Taiwan, R.O.C.. Correspondence to: Shan-Hung Wu <shwu@cs.nthu.edu.tw>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

NTGAs are based on the recent development of Neural Tangent Kernels (NTKs) (Jacot et al., 2018; Lee et al., 2019; Chizat et al., 2019), which allow a closed form approximation of the evolution of a class of wide DNNs during training by gradient descent. We conduct extensive experiments to evaluate the effectiveness of NTGAs. The results show that an NTGA attack has remarkable transferability across a wide range of models, including fully-connected networks and Convolutional Neural Networks (CNNs), trained under various conditions regarding the optimization method, loss function, etc.

2. Related Work

In this section, we briefly review different types of attacks against machine learning algorithms. We then explain why generalization attacks against DNNs are hard to achieve.

The attacks against machine learning algorithms can happen at either training or test time, which refer to *poisoning attacks* (Biggio et al., 2012; Xiao et al., 2015; Koh & Liang, 2017; Muñoz-González et al., 2017; Yang et al., 2017; Shafahi et al., 2018; Chan-Hon-Tong, 2019; Weng et al., 2020) and *adversarial attacks* (Szegedy et al., 2013; Goodfellow et al., 2014), respectively. The goal of an adversarial attack is to mislead a trained model into making a wrong prediction given an adversarially crafted input. In contrast, a poisoning attack aims to disrupt a training procedure to output defective models given an adversarially perturbed (or "poisoned") dataset.

2.1. Poisoning Attacks

The poisoning attacks can be further divided into two categories: *integrity* and *generalization* attacks[1]. The goal of the integrity attacks is to let the training procedure output a model whose behavior can be manipulated at test time. A common example is the backdoor attacks (Shafahi et al., 2018; Zhu et al., 2019) where the model makes wrong predictions only when certain triggers (i.e, patterns) are present in the input. On the other hand, the generalization attack aims to obtain a model that performs poorly on the validation and test sets (Muñoz-González et al., 2017; Yang et al., 2017; Chan-Hon-Tong, 2019).

Depending on whether an attacker changes the labeling of training data, the poisoning attacks can also be divided into *clean-label* and *dirty-label* attacks. Normally, a dirty-label attack is easier to generate than a clean-label one because the loss surface of the model is controlled by both the data points and labels. In this paper, we focus on the clean-label generalization attacks since they allow legitimate users to consume the dataset by following the owner's intention (and labeling) after data poisoning.

[1]In the computer security community, the integrity and generalization attacks are called *poisoning integrity* and *poisoning availability* attacks, respectively.

2.2. Generalization Attacks

Given a training set $\mathbb{D} = (\boldsymbol{X}^n \in \mathbb{R}^{n\times d}, \boldsymbol{Y}^n \in \mathbb{R}^{n\times c})$ of n examples with d dimensional input variables and c dimensional labels, a validation set $\mathbb{V} = (\boldsymbol{X}^m, \boldsymbol{Y}^m)$ of m examples, and a model $f(\cdot\,;\boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta}$, the objective of a generalization attack can be formulated as a bilevel optimization problem (Biggio et al., 2012; Mei & Zhu, 2015; Xiao et al., 2015):[2]

$$\begin{aligned} &\arg\max_{(\boldsymbol{P},\boldsymbol{Q})\in\mathcal{T}} L\left(f(\boldsymbol{X}^m;\boldsymbol{\theta}^*),\boldsymbol{Y}^m\right) \\ &\text{subject to } \boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} L\left(f(\boldsymbol{X}^n+\boldsymbol{P};\boldsymbol{\theta}),\boldsymbol{Y}^n+\boldsymbol{Q}\right), \end{aligned} \tag{1}$$

where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are the perturbations to be added to $\mathbb{D}$, $L(\cdot\,,\cdot)$ is a loss function between model predictions and ground-truth labels, and $\mathcal{T}$ is a threat model that controls the allowable values of $\boldsymbol{P}$ and $\boldsymbol{Q}$. The outer $\max$ problem finds $\boldsymbol{P}$ and $\boldsymbol{Q}$ that disrupt the generalizability of f on $\mathbb{V}$, while the inner $\min$ problem trains f on the poisoned training data $\hat{\mathbb{D}} = (\boldsymbol{X}^n+\boldsymbol{P},\boldsymbol{Y}^n+\boldsymbol{Q})$. In practice, the loss functions for training and validation can be different. By controlling $\mathcal{T}$, one can define different attacks. For a clean-label attack, $\boldsymbol{Q}$ is required to be a zero matrix. In the case where an attacker only poisons partial training data, some rows of $\boldsymbol{P}$ are zero.

The main challenge of solving Eq. (1) by gradient ascent is to compute the gradients of $L(f(\boldsymbol{X}^m;\boldsymbol{\theta}^*),\boldsymbol{Y}^m)$ w.r.t. $\boldsymbol{P}$ and $\boldsymbol{Q}$ *through multiple training steps of* f. If f is trained using gradient descent, the gradients of $L(f(\boldsymbol{X}^m;\boldsymbol{\theta}^*),\boldsymbol{Y}^m)$ require the computation of high-order derivatives of $\boldsymbol{\theta}^*$ and can easily become intractable.

A trick, which only works for convex models like SVMs and Logistic regression, to solve Eq. (1) is to replace the inner $\min$ problem with its stationary (or Karush-Kuhn-Tucker, KKT) conditions (Mei & Zhu, 2015; Muñoz-González et al., 2017; Demontis et al., 2019). When L is convex (w.r.t. $\boldsymbol{\theta}$), all stationary points are global minimum, and the solutions to the dual of the inner problem can have a closed form. This avoids the expensive computation of high-order derivatives of $\boldsymbol{\theta}^*$ and allows the inner problem to be solved together with the outer $\max$ problem by the same gradient ascent steps (or other optimization methods). However, the above trick is *not* applicable to non-convex DNNs. Moreover, the studies (Muñoz-González et al., 2017; Demontis et al., 2019) show that the generalization attacks created based on convex models are not transferrable to DNNs.

[2]Given a design matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and a function $f : \mathbb{R}^n \to \mathbb{R}^c$, we abuse the notation and denote by $f(\boldsymbol{X}) \in \mathbb{R}^{n\times c}$ the stack of the results of iteratively applying f to each row of $\boldsymbol{X}$.

To solve Eq. (1) against DNNs, existing works (Muñoz-González et al., 2017; Demontis et al., 2019; Chan-Hon-Tong, 2019) relax the bilevel problem by assuming that $\boldsymbol{\theta}^*$ is known and fixed. In particular, the study (Chan-Hon-Tong, 2019) devises a clean-label generalization attack using the following objective:

$$\begin{gathered}\arg\min_{\boldsymbol{P}} L\left(f(\boldsymbol{X}^n + \boldsymbol{P}; \hat{\boldsymbol{\theta}}), \boldsymbol{Y}^n\right),\\ \text{where } \hat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta}} L\left(f(\boldsymbol{X}^m; \boldsymbol{\theta}), \boldsymbol{Y}^m\right).\end{gathered} \tag{2}$$

An attacker first obtains $\hat{\boldsymbol{\theta}}$ such that $f(\cdot\,; \hat{\boldsymbol{\theta}})$ performs poorly on $\mathbb{V}$. It then uses the fixed $\hat{\boldsymbol{\theta}}$ to solve $\boldsymbol{P}$. Since $\hat{\boldsymbol{\theta}}$ is independent of $\boldsymbol{P}$ now, Eq. (2) can be solved efficiently. Another direction to solve this problem is to use a GAN-like architecture (Feng et al., 2019), where two networks corresponding to the inner and outer problems in Eq. (1), respectively, learn to outperform each other adversarially. This method considers the dependency between $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{P}$ at the cost of convergence issues (due to its GAN-like nature). Both of the above studies use a white-box assumption where the architecture and exact weights $\boldsymbol{\theta}^*$ of f are known to an attacker.[3] This assumption, however, does not hold in many practical situations. For data owners, it is generally hard to control the machine learning processes conducted by third parties on their data. Efficient computing of a black-box, clean-label generalization attack against DNNs remains an open problem.

3. Neural Tangent Generalization Attacks

A successful black-box generalization attack needs to overcome two key challenges: (i) to solve Eq. (1) efficiently against a non-convex f, and (ii) to let f be a "representative" surrogate of the unknown target models used by third-party data consumers. We propose the Neural Tangent Generalization Attacks (NTGAs) that can simultaneously cope with these two challenges. Intuitively, we let f be the mean of a Gaussian Process (GP) that models the training dynamics of a class of wide DNNs. We simplify Eq. (1) using the fact that the predictions of f over unseen data at training time t can be written in a closed form without an exact $\boldsymbol{\theta}^*$ involved. We also show that f could be a good surrogate of various unknown target models.

The NTGAs are based on the recent development of Neural Tangent Kernels (NTKs) (Jacot et al., 2018; Lee et al., 2019; Chizat et al., 2019), which we review below for the sake of completeness.

[3]The study (Feng et al., 2019) conducted black-box experiments, but it is still a white-box approach because there is no explicit design to cope with the unknown $\boldsymbol{\theta}^*$.

Algorithm 1 Neural Tangent Generalization Attack

```
Input: $\mathbb{D} = (\boldsymbol{X}^n, \boldsymbol{Y}^n)$, $\mathbb{V} = (\boldsymbol{X}^m, \boldsymbol{Y}^m)$, $\bar{f}(\cdot\,; k(\cdot,\cdot), t)$,
       $L$, $r$, $\eta$, $\mathcal{T}(\epsilon)$
Output: $\boldsymbol{P}$ to be added to $\boldsymbol{X}^n$
1 Initialize $\boldsymbol{P} \in \mathcal{T}(\epsilon)$
2 for $i \leftarrow 1$ to $r$ do
3     $\boldsymbol{G} \leftarrow \nabla_{\boldsymbol{P}} L(\bar{f}(\boldsymbol{X}^m; \hat{\boldsymbol{K}}^{m,n}, \hat{\boldsymbol{K}}^{n,n}, \boldsymbol{Y}^n, t), \boldsymbol{Y}^m)$
4     $\boldsymbol{P} \leftarrow \text{Project}(\boldsymbol{P} + \eta \cdot \text{sign}(\boldsymbol{G}); \mathcal{T}(\epsilon))$
5 end
6 return $\boldsymbol{P}$
```

3.1. Neural Tangent Kernels

A recent breakthrough in theory shows that the distribution of a class of wide neural networks (of any depth) can be nicely approximated by a GP either before training (Lee et al., 2018; Matthews et al., 2018) or during training (Jacot et al., 2018; Lee et al., 2019; Chizat et al., 2019) under gradient descent. In particular, the behavior of the GP in the latter case is governed by an NTK (Jacot et al., 2018). As the width of the networks grows into infinity, the NTK converges to a deterministic kernel, denoted by $k(\cdot,\cdot)$, which remains constant during training. Given two data points $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$, the $k(\boldsymbol{x}^i, \boldsymbol{x}^j)$ represents a similarity score between the two points from the network class' point of view. It only loosely depends on the exact weights of a particular network.

Denote the mean of the GP by $\bar{f}$ and let $\boldsymbol{K}^{n,n} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{K}^{m,n} \in \mathbb{R}^{m\times n}$ be two kernel matrices where $K^{n,n}_{i,j} = k(\boldsymbol{x}^i \in \mathbb{D}, \boldsymbol{x}^j \in \mathbb{D})$ and $K^{m,n}_{i,j} = k(\boldsymbol{x}^i \in \mathbb{V}, \boldsymbol{x}^j \in \mathbb{D})$, respectively. At time step t during the gradient descent training, the mean prediction of the GP over $\mathbb{V}$ evolve as:

$$\begin{aligned}&\bar{f}(\boldsymbol{X}^m; \boldsymbol{K}^{m,n}, \boldsymbol{K}^{n,n}, \boldsymbol{Y}^n, t)\\ &\quad= \boldsymbol{K}^{m,n}(\boldsymbol{K}^{n,n})^{-1}(\boldsymbol{I} - e^{-\eta \boldsymbol{K}^{n,n} t})\boldsymbol{Y}^n.\end{aligned} \tag{3}$$

The above can be extended to approximate networks of a wide range of architectures, including CNNs (Novak et al., 2018; Garriga-Alonso et al., 2018; Yang, 2019a), RNNs (Yang, 2019b; Alemohammad et al., 2020), networks with the attention mechanism (Hron et al., 2020), and other architectures (Yang, 2019b; Arora et al., 2019). For a class of CNNs, the kernel of the approximating GP converges as the number of channels at each layer grows into infinity. Please see the supplementary file for a more complete review of NTKs.

3.2. NTGA Objective and Algorithm

By Eq. (3), we can write the predictions made by $\bar{f}$ over $\mathbb{V}$ in a closed form *without knowing the exact weights of a*

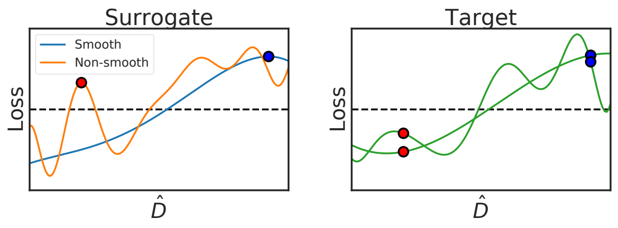

Figure 1. A smoother surrogate can lead to better attack transferability. The left figure shows the loss landscape of two surrogates used by an attacker over the space of $\hat{\mathbb{D}}$, while the right figure shows the loss landscape of two target models trained by different third parties. The dashed horizontal lines indicate the threshold of a successful attack. Let the blue and red dots be two poisoned training sets crafted by NTGA using the smooth and non-smooth surrogates, respectively. The red dot fails to attack the target models.

particular network. This allows us to rewrite Eq. (1) as a more straightforward problem:

$$\arg\max_{\boldsymbol{P}\in\mathcal{T}} L\left(\bar{f}(\boldsymbol{X}^m;\hat{\boldsymbol{K}}^{m,n},\hat{\boldsymbol{K}}^{n,n},\boldsymbol{Y}^n,t),\boldsymbol{Y}^m\right), \quad (4)$$

where $\hat{\boldsymbol{K}}^{n,n}$ and $\hat{\boldsymbol{K}}^{m,n}$ are the kernel matrices built on the poisoned training data such that $\hat{K}^{n,n}_{i,j} = k(\boldsymbol{X}^n_{i,:} + \boldsymbol{P}_{i,:},\boldsymbol{X}^n_{j,:} + \boldsymbol{P}_{j,:})$ and $\hat{K}^{m,n}_{i,j} = k(\boldsymbol{X}^m_{i,:},\boldsymbol{X}^n_{j,:} + \boldsymbol{P}_{j,:})$, respectively. We let $\boldsymbol{Q}$ be a zero matrix because we focus on the clean-label generalization attacks. Now, the gradients of the loss L w.r.t. $\boldsymbol{P}$ can be easily computed without backpropagating through the training steps. We use the projected gradient ascent to solve Eq. (4).

Algorithm 1 outlines the key steps of NTGA. At lines 2-5, it iteratively updates $\boldsymbol{P}$. The hyperparameter r, which controls the maximum number of iterations, affects attack strength and running time. At line 4, NTGA projects the updated perturbations onto the feasible set $\mathcal{T}(\epsilon)$ if they exceed the maximum allowable amount indicated by ϵ. A smaller ϵ makes an attack less visible to humans. The hyperparameter t controls when an attack (i.e., a returning $\boldsymbol{P}$) will take effect during the training processes run by others. Since DNNs are usually trained with early stopping, the t should not be a very large number. We will study the effects of these hyperparameters in Section 4.

3.3. Merits and Practical Concerns

Model Agnosticism. NTGA is agnostic to the target models and training procedures used by third-party data consumers because the $\bar{f}$ in Eq (4) is only their surrogate. The reasons why NTGA can generate a successful black-box attack are two-folds. First, the GP behind $\bar{f}$ approximates the evolution of not only a single randomly initialized network but *an ensemble of infinite random networks* of the same architecture. This allows NTGA to create a successful attack without knowing the exact weights of the target models. Second, the GP approximates the evolution of an ensemble of *infinitely wide* networks, which are known to be able to approximate any function universally (Hornik et al., 1989; Cybenko, 1989). We also observe (in Section 4) that a wider DNN tends to have a smoother loss w.r.t. $\hat{\mathbb{D}}$. This leads to a smoother surrogate and helps NTGA find a better local optimum (via gradient ascent) that transfers, as shown in Figure 1. So, NTGA does not need to tightly follow the architecture of a target model to create a successful attack.

Collaborative Perturbations. Another advantage of NTGA is that it can generate *collaborative* perturbations that span across different training data points. In Eq. (4), the perturbations $\boldsymbol{P}_{i,:}$ for individual training data points $\boldsymbol{X}^n_{i,:}$ are solved collectively. So, each $\boldsymbol{P}_{i,:}$ can slightly modify a point to remain invisible to human eyes, and together they can significantly manipulate model generalizability.

Scalability on Large Datasets. The computation of the gradients of L w.r.t. $\boldsymbol{P}$ in Eq. (4) backpropagates through $(\hat{\boldsymbol{K}}^{n,n})^{-1}$ and $e^{-\eta\hat{\boldsymbol{K}}^{n,n}t}$ (see Eq. (3)). This creates a scalability issue on a training set with a large n.

One way to increase scalability is to use a modified version of NTGA, called Blockwise NTGA (B-NTGA), that reduces the degree of collaboration. Specifically, B-NTGA first partitions $\mathbb{D}$ into multiple groups, where each group contains b examples, and then solves Eq. (4) for each group independently. In effect, the $\boldsymbol{P}$ for the entire training set is solved using only the $b \times b$ diagonal blocks of $\hat{\boldsymbol{K}}^{n,n}$. Although missing the off-diagonal information, B-NTGA works if b is large enough to enable efficient collaboration. In practice, the perturbations for each block can be solved by multiple machines in parallel.

4. Experiments

We conduct experiments to evaluate the performance of NTGA using the following default settings.

Datasets. We aim to poison the MNIST (LeCun et al., 2010), CIFAR-10 (Krizhevsky, 2009), and a subset of ImageNet (Deng et al., 2009) datasets. The MNIST and CIFAR-10 datasets contain 60K and 50K training examples. For ImageNet, we follow the study (Feng et al., 2019) and use only the 2,600 training examples of the "bulbul" and "jellyfish" classes. On each dataset, we randomly split $\sim 15\%$ examples from the training set as $\mathbb{V}$ and use the rests as $\mathbb{D}$ for Eq. (1). We use the full test sets to report performance.

Baselines. We consider the studies, called Return Favor Attack (RFA) (Chan-Hon-Tong, 2019) and DeepConfuse (Feng et al., 2019), as our baselines. To the best of our

Table 1. The test accuracy of different attacks under gray-box settings.

Dataset \Attack	Clean	RFA	Deep Confuse	NTGA (1)	NTGA (8)	NTGA (64)	NTGA (512)	NTGA (4096)	NTGA (∞)
Surrogate: *-FNN $\rightarrow$ 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 $\rightarrow$ 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

knowledge, they are the only works which are able to efficiently generate clean-label generalization attacks against DNNs. We have reviewed RFA and DeepConfuse in Section 2.2 (see Eq. (2)).

Surrogates. Each poisoning algorithm is equipped with two surrogates and can use one of them to generate $\boldsymbol{P}$. In NTGA, we use the means, named GP-FNN and GP-CNN, of two Gaussian Processes (GPs). The GP-FNN is the mean of 5-layer, infinitely-wide, fully-connected networks with the Erf non-linearity, while the GP-CNN is the mean of the networks consisting of 2 convolutional layers having infinite channels and 3 infinitely-wide dense layers with the ReLU non-linearity. Both the GPs assume the networks are trained using gradient descent with the Mean Squared Error (MSE) loss. The surrogates of RFA and DeepConfuse are two neural networks, named S-FNN and S-CNN, that have the same architectures as an element network of GP-FNN and GP-CNN, respectively, except with finite width (512 neurons) and channels (64). We train S-FNNs and S-CNNs by following the original papers.

More Settings. We implement the B-NTGA describe in Section 3.3 with b equals 5K, 4K, and 1K on MNIST, CIFAR-10, and ImageNet, respectively. We update $\boldsymbol{P}$ for $r = 10$ iterations. The maximum allowable perturbation ϵ is set to 0.3 on MNIST, 8/255 on CIFAR-10, and 0.1 on ImageNet measured by the l_∞ distance, respectively. By default, each attack poisons the entire training set because we assume the attack is performed by the data owner. We implement RFA using the same hyperparameters. Our code uses the Neural Tangents library (Novak et al., 2019) built on top of JAX (Bradbury et al., 2018) and TensorFlow (Abadi et al., 2016). All reported results of an experiment are the average of 5 distinct runs. Please refer to the supplementary file for detailed settings.

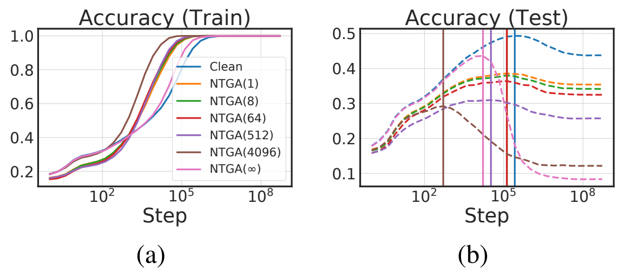

Figure 2. (a) Training and (b) test dynamics of GP-FNN on CIFAR-10 with different t. Vertical lines represent the early-stop points.

4.1. Gray-box Attacks

First, we evaluate the performance of different poisoning methods under gray-box settings, where an attacker knows the architecture of a target model but not its weights. We train two target networks, denoted by FNN and CNN, whose architectures are identical to the surrogates S-FNN and S-CNN used by the baselines, respectively.[4] We randomly initialize FNN and CNN and use a Stochastic Gradient Descent (SGD) optimizer to minimize their MSE losses on the clean and poisoned training data. We use early stopping to prevent the networks from overfitting. The results are shown in Table 1, where NTGA($\cdot$) denotes an attack generated by NTGA with a specific hyperparameter t (see Eq. (3)). As we can see, NTGA significantly outperforms the baselines and sets new state-of-the-art for clean-label gray-box generalization attacks. Note that the surrogates used by NTGA are different from the target models in numerous ways, including network architectures and sizes (infinite vs. finite), optimization algorithms (full gradient descent vs. SGD),

[4]Our gray-box settings favor RFA and DeepConfuse over NTGA because the target models are less similar to the surrogates used by NTGA.

Table 2. The test accuracy of different attacks under black-box settings.

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

Target \Attack	Clean	RFA	Deep Confuse	NTGA (1)	NTGA (8)	NTGA (64)	NTGA (512)	NTGA (4096)	NTGA (∞)
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

Target \Attack	Clean	RFA	Deep Confuse	NTGA (1)	NTGA (8)	NTGA (64)	NTGA (512)	NTGA (4096)	NTGA (∞)
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

and model weights. The ensembles of networks behind the surrogates seem to cover the target models well.

Effect of t. The hyperparameter t plays an important role in NTGA. The t controls when an attack will take effect during the training process of a target model. With a smaller t, the attack has a better chance to affect training before the early stop. However, a too small t could make the surrogate non-representative of the target network, resulting in a less effective attack. Figure 2 shows the training and test dynamics of GP-FNN on the CIFAR-10 dataset. Although the attack generated by NTGA(∞) works best in the long term (after 10^6 steps of gradient descent), this result will never happen in practice because of the early stopping. The vertical lines in Figure 2(b) indicate the early-stop points, which are highly consistent with those of the target model FNN (cf. the second row of Table 1). Similar results on MNIST can be found in the supplementary file. So, the attacker can use Figure 2(b) to determine the best value of t. Note that t also affects the look of the perturbations $\boldsymbol{P}$ to human eyes, as we will discuss in Section 4.3.

4.2. Black-box Attacks

We take a step further and evaluate different poisoning methods under black-box settings, where an attacker knows nothing about a target model. We train the target networks FNN and CNN described in the previous section using the poisoned data generated by a method with the *-CNN and *-FNN surrogates, respectively. We also train additional networks including FNN-ReLU, ResNet18 (He et al., 2016), and DenseNet121 (Huang et al., 2017). The FNN-ReLU has the same architecture as FNN except using the ReLU nonlinearity. Different from FNN/CNN, these three new target models are trained with the cross-entropy loss. We also enable learning rate scheduling and data augmentation when training ResNet18 and DenseNet121. In all the above cases, a surrogate is very different from a target model in architecture and weights.

Table 2 shows the results. Again, NTGA significantly outperforms the baseline and sets new state-of-the-art for clean-label black-box generalization attacks. We notice that RFA barely has an effect against ResNet18 and DenseNet121 having complex architectures involving dropout, batch normalization, residual connection, etc. On the other hand, NTGA successfully degrades the generalizability of these two networks such that they become useless in most practical situations. This verifies that the GP-based surrogates, which model the ensembles of infinitely wide networks, can indeed lead to better attack transferability. We will further study the transferability in Section 4.4.

In general, the attacks based on the GP-FNN surrogate have greater influence against the fully-connected target networks (FNN and FNN-ReLU), while the attacks based on the GP-CNN surrogate work better against the convolutional target networks (CNN, ResNet18, and DenseNet121). Interestingly, the 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 (Xiao et al., 2020).

Effect of t. Under the black-box settings, the best t varies across different target models. A smaller t seems to work better against the convolutional target networks, while a larger t leads to more successful attacks against the fully-connected models. We believe this is because the convolutional networks learn more efficiently from images (i.e., have lower sample complexity) than the full-connected ones.

4.3. Visualization

The hyperparameter t of NTGA also controls how an attack looks. In Figures 3 and 4, we visualize some poisoned images $\boldsymbol{X}_{i,:}$ from the CIFAR-10 and ImageNet datasets as well as their normalized perturbations $\boldsymbol{P}_{i,:}$ generated by RFA and DeepConfuse with S-CNNs and NTGA with GP-CNN. As we can see, the perturbations generated by RFA are of high frequency and look similar to those generated by an adversarial attack (Szegedy et al., 2013; Goodfellow et al., 2014). DeepConfuse gives medium-frequency perturbations. In particular, there exist artifacts in high-resolution images (see Figure 4). On the other hand, the perturbations generated by NTGA can have different frequencies controlled by t. In particular, a smaller t leads to simpler perturbations. This is consistent with the previous findings (Arpit et al., 2017; Rahaman et al., 2019) that a (target) network tends to learn low-frequency patterns at the early stage of training.

At $t \to \infty$, the high-frequency perturbations generated by NTGA (Figure 3(f)) are less visible than those by RFA (Figure 3(b)). This verifies the effectiveness of collaboration described in Section 3.3. In RFA, the perturbations $\boldsymbol{P}_{i,:}$ added to a data point $\boldsymbol{X}_{i,:}$ only depends on the $\boldsymbol{X}_{i,:}$ itself because (i) the $\hat{\boldsymbol{\theta}}$ in Eq. (2) is trained on $\mathbb{V}$, thus is independent of $\mathbb{D}$, and (ii) other examples $\boldsymbol{X}_{j,:}$ do not participate in the computation of gradients of L w.r.t. $\boldsymbol{P}_{i,:}$. On the contrary, the gradients of L w.r.t. $\boldsymbol{P}_{i,:}$ in Eq. (4) backpropagates through $\boldsymbol{K}^{n,n}$, which involves all other data points $\boldsymbol{X}_{j,:}$. As a consequence, the slight perturbations made by NTGA can still create a significant aggregate effect.

4.4. Transferability

Next, we investigate the cause of the superior transferability of the NTGA attacks. As discussed in Section 3.3, the GPs behind the NTGA surrogates model the evolution of an ensemble of infinitely wide and many networks. By the universal approximation theorem (Hornik et al., 1989; Cybenko, 1989), the GPs can cover target networks of any weights and architectures.

From another perspective, a wide surrogate can have a smoother loss landscape over $\hat{\mathbb{D}}$ that helps NTGA find better local optima ($\boldsymbol{P}$) that transfers (see Figure 1). To verify this, we examine the largest 100 eigenvalues of the Hessians $\partial^2 L/\partial(\boldsymbol{X}^n + \boldsymbol{P})^2$ of different networks adapted from the FNN and CNN, as shown in Figure 5, which provide a local measure of the curvature of the loss landscape. As the

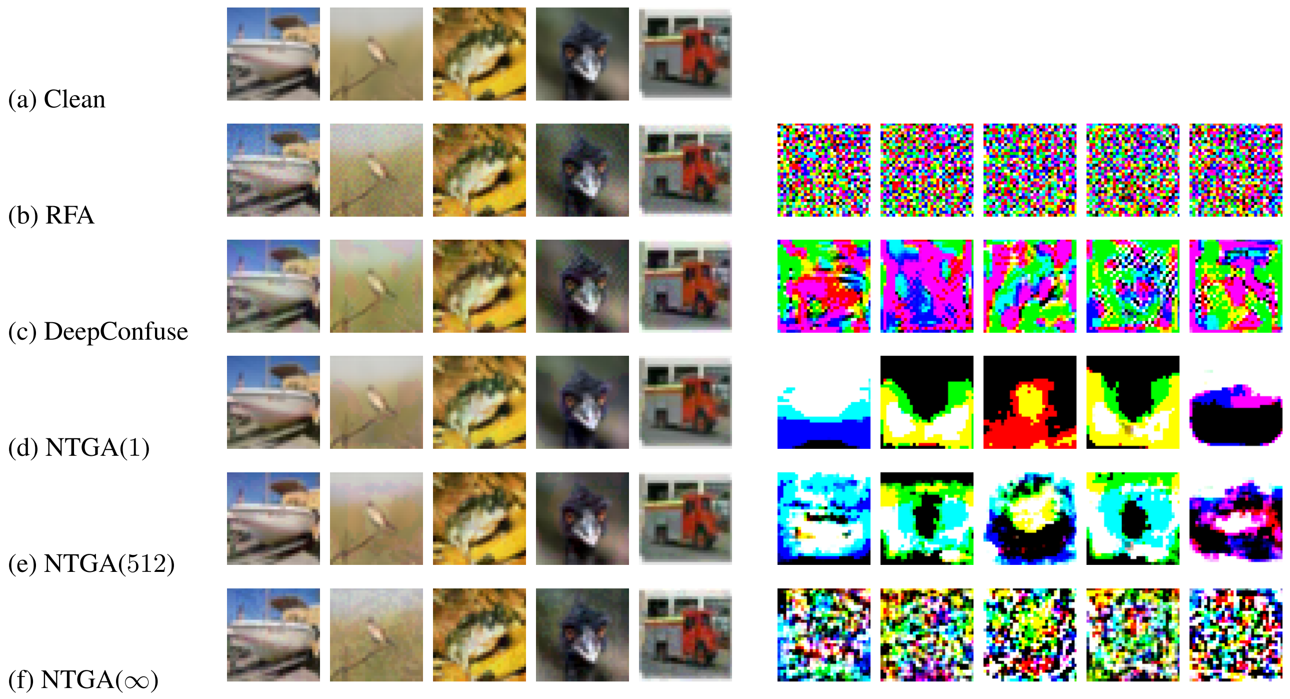

Figure 3. Visualization of some poisoned CIFAR-10 images (left) and their normalized perturbations (right).

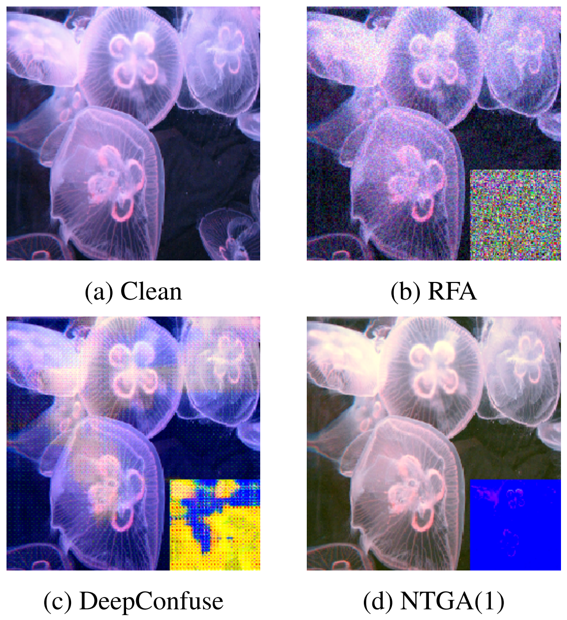

Figure 4. Visualization of some poisoned ImageNet images and their normalized perturbations (bottom-right corner).

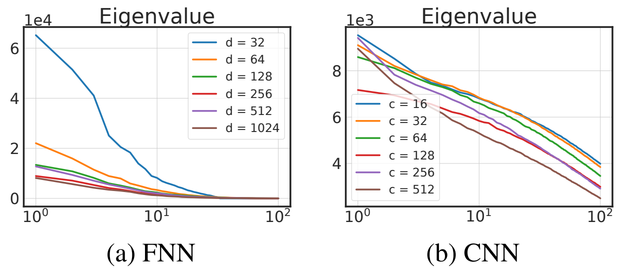

Figure 5. Top 100 eigenvalues of the Hessians ($\partial^2 L/\partial(\boldsymbol{X}^n+\boldsymbol{P})^2$) of the variants of (a) FNN and (b) CNN with different widths d and number of channels c.

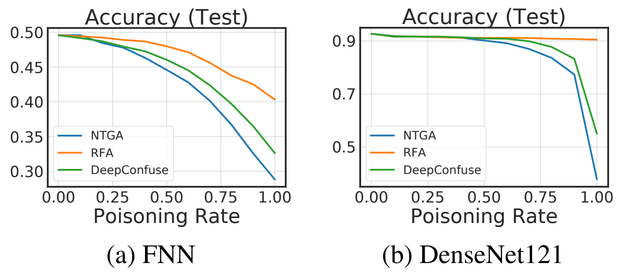

Figure 6. Partial data poisoning on CIFAR-10.

width or the number of channels increase, the eigenvalues become more evenly distributed, implying a smoother loss landscape. Therefore, the GP-FNN and GP-CNN, which model infinitely wide networks, could lead to the "best" transferability.

4.5. Effect of Poisoning Rate

So far, we assume that an attacker can poison the entire training set. In practice, a data consumer may scrap data from multiple sources and learn from a consolidated training set.

Table 3. Trade-off between speed and collaboration by varying block size b.

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

Table 4. Sensitivity to σ_b and σ_w that are used on construct $\boldsymbol{K}^{n,n}$ and $\boldsymbol{K}^{m,n}$.

σ_b	σ_w	FNN	FNN'	CNN	R18	D121
Surrogate: GP-FNN						
0.00	1.10	20.35	24.75	37.80	**35.98**	**42.75**
0.10	1.55	**19.40**	**24.52**	37.87	40.18	46.20
0.18	1.76	20.63	25.95	**36.05**	39.68	47.36
0.25	1.88	21.05	26.03	36.94	38.71	48.64

To simulate this case, we conduct an experiment where an attacker can only poison partial data collected by a consumer. The results are shown in Figure 6. Although all the baselines and NTGA can degrade the performance of a target network by partial poisoning, our NTGA consistently outperforms the baselines. Note that the test performance does not drop significantly because the target networks can learn from other clean data. This shows that designing a strong generalization attack while maintaining invisibility (low ϵ) and a low poisoning rate is very challenging. However, the results encourage the consumer to discard the poisoned data, which may be good enough for the attacker.

4.6. Ablation Study

Trade-off between Speed and Collaboration. NTGA generates collaborative perturbations (see Section 3.3). Here, we investigate the effect of collaboration. The results, which are shown in Table 3, indicate that a larger block size b always leads to better performance. This suggests that collaboration is a key to the success of NTGA. However, a larger b induces higher space and time complexity, as the gradients of L in Eq. (4) backpropagates through $(\boldsymbol{K}^{n,n})^{-1}$ and $e^{\boldsymbol{K}^{n,n}}$. Table 3 also shows the trade-off between the performance and time required to solve Eq. (4) on a machine with NVIDIA Tesla V100 GPU. Interestingly, NTGA starts to outperform RFA and DeepConfuse since $b \geq 100$ and $b \geq 1000$, respectively. In practice, DeepConfuse suffers from convergence issues due to its GAN-like nature and requires 5-7 days to generate poisoned CIFAR-10 data, while NTGA takes only 5 hours given $b = 4000$.

Kernel Construction. The construction of $\boldsymbol{K}^{n,n}$ and $\boldsymbol{K}^{m,n}$ depends on two hyperparameters, denoted by σ_b and σ_w, that controls the bias and variance of the randomly initialized weights of the networks in the ensemble modeled by the GP behind a surrogate. Studies (Poole et al., 2016; Schoenholz et al., 2016; Lee et al., 2018) shows that σ_b and σ_w can significantly affect the performance of the GP, and there exists a sweet spot in the space of σ_b and σ_w called *critical line* that allows the GP to have the best performance. We randomly choose 4 distinct combinations of σ_w and σ_b lying on the critical line to construct $\boldsymbol{K}^{n,n}$ and $\boldsymbol{K}^{m,n}$ and then use the kernel matrices to craft attacks. Table 4 shows the performance of these attacks. NTGA does not seem to be sensitive to σ_b and σ_w provided that they lie on the critical line.

We have conducted more experiments to further verify the effectiveness of NTGAs. For more details, please see the supplementary file.

5. Conclusion

In this work, we propose Neural Tangent Generalization Attacks (NTGAs) against DNNs and conduct extensive experiments to evaluate their performance. To the best of our knowledge, this is the first work that enables clean-label, black-box generalization attacks against DNNs under practical settings. NTGAs have implications for data security and privacy. In particular, it can help data owners protect their data and prevent the data from unauthorized use.

As our future work, we plan to study the performance of NTGAs given different types of GPs, such as those approximating RNNs (Yang, 2019b; Alemohammad et al., 2020), networks with the attention mechanism (Hron et al., 2020), and other architectures (Yang, 2019b; Arora et al., 2019). We also plan to evaluate NTGAs on real-world applications, such as the online healthcare or music streaming services, where data privacy and regulatory compliance is in high demand. (Quinonero-Candela & Rasmussen, 2005; Hensman et al., 2013; Wang et al., 2019; Liu et al., 2020).

Acknowledgments

This work is supported by the MOST Joint AI Innovation Research Center (MOST 109-2634-F-007-014-, MOST 110-2634-F-007-013-) grants from Ministry of Science and Technology, Taiwan. The experiments are supported in part by Taiwan Computing Cloud (TWCC) from National Center for High-performance Computing of the National Applied Research Labs.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016.

Alemohammad, S., Wang, Z., Balestriero, R., and Baraniuk, R. The recurrent neural tangent kernel. *arXiv preprint arXiv:2006.10246*, 2020.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. On exact computation with an infinitely wide neural net. In *Proc. of NeurIPS*, 2019.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *Proc. of ICML*. PMLR, 2017.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *Proc. of ICML*. PMLR, 2012.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Burt, C. Facial biometrics training dataset leads to bipa lawsuits against amazon, alphabet and microsoft, Jul 2020. URL https://reurl.cc/dV4rD8.

Chan-Hon-Tong, A. An algorithm for generating invisible data poisoning using adversarial noise that breaks image classification deep learning. *Machine Learning and Knowledge Extraction*, 1(1), 2019.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming. In *Proc. of NeurIPS*, 2019.

Conklin, A. Facebook agrees to pay record $650m to settle facial recognition lawsuit, Jul 2020. URL https://reurl.cc/Gd2kev.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., and Roli, F. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proc. of CVPR*, 2009.

Feng, J., Cai, Q.-Z., and Zhou, Z.-H. Learning to confuse: generating training time adversarial data with autoencoder. *arXiv preprint arXiv:1905.09027*, 2019.

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. Deep convolutional networks as shallow gaussian processes. In *Proc. of ICLR*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Hron, J., Bahri, Y., Sohl-Dickstein, J., and Novak, R. Infinite attention: Nngp and ntk for deep attention networks. In *Proc. of ICML*. PMLR, 2020.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proc. of CVPR*, 2017.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Proc. of NeurIPS*, 2018.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35. IEEE, 2018.

Kloft, M. and Laskov, P. Security analysis of online centroid anomaly detection. *The Journal of Machine Learning Research*, 13(1), 2012.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proc. of ICML*. PMLR, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. In *Proc. of ICLR*, 2018.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In *Proc. of NeurIPS*, 2019.

Liu, H., Ong, Y.-S., Shen, X., and Cai, J. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31 (11):4405–4423, 2020.

Matthews, A. G. d. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *Proc. of ICLR*, 2018.

Mei, S. and Zhu, X. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proc. of AAAI*, volume 29, 2015.

Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proc. of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J. Bayesian deep convolutional networks with many channels are gaussian processes. In *Proc. of ICLR*, 2018.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. Neural tangents: Fast and easy infinite neural networks in python. In *Proc. of ICLR*, 2019.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. *Proc. of NeurIPS*, 2016.

Quinonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 2005.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *Proc. of ICML*. PMLR, 2019.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proc. of NeurIPS*, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Vincent, J. Google accused of inappropriate access to medical data in potential class-action lawsuit, Jun 2019. URL `https://reurl.cc/bzK69v`.

Wang, K. A., Pleiss, G., Gardner, J. R., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact gaussian processes on a million data points. *arXiv preprint arXiv:1903.08114*, 2019.

Weng, C.-H., Lee, Y.-T., and Wu, S.-H. B. On the trade-off between adversarial and backdoor robustness. *Proc. of NeurIPS*, 2020.

Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., and Roli, F. Is feature selection secure against training data poisoning? In *Proc. of ICML*. PMLR, 2015.

Xiao, L., Pennington, J., and Schoenholz, S. Disentangling trainability and generalization in deep neural networks. In *Proc. of ICML*. PMLR, 2020.

Yang, C., Wu, Q., Li, H., and Chen, Y. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.

Yang, G. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019a.

Yang, G. Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *arXiv preprint arXiv:1910.12478*, 2019b.

Zhu, C., Huang, W. R., Li, H., Taylor, G., Studer, C., and Goldstein, T. Transferable clean-label poisoning attacks on deep neural nets. In *Proc. of ICML*. PMLR, 2019.