# Neural Tangent Generalization Attacks: Supplementary Materials

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

## A. Theoretical Background

In spite of the great success achieved by deep learning models, the theory of these complicated systems is usually elusive due to their typically high-dimensional non-convex loss surfaces. A recent breakthrough shows that the distribution of a class of wide neural networks (of any depth) can be nicely approximated by a Gaussian Process (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.

### A.1. Gaussian Processes (GPs)

A GP is a stochastic process that models the distribution of a class $\mathbb{F}$ of functions using Gaussian and can be defined by a mean function $\mu(\cdot)$ and covariance/kernel function $k(\cdot,\cdot)$. Let each function $f:\mathbb{R}^d\rightarrow\mathbb{R}$ in $\mathbb{F}$ map $d$ dimensional input variables to the output. Given a training set $\mathbb{D}=(\boldsymbol{X}^n\in\mathbb{R}^{n\times d},\boldsymbol{y}^n\in\mathbb{R}^n)$ of $n$ examples and a validation set $\mathbb{V}=(\boldsymbol{X}^m\in\mathbb{R}^{m\times d},\boldsymbol{y}^m\in\mathbb{R}^m)$ of $m$ examples, the GP of $\mathbb{F}$ can be written as

$$\begin{bmatrix}\boldsymbol{y}^n\\ \boldsymbol{y}^m\end{bmatrix}\sim\mathcal{N}(\begin{bmatrix}\boldsymbol{\mu}^n\\ \boldsymbol{\mu}^m\end{bmatrix},\begin{bmatrix}\boldsymbol{K}^{n,n} & \boldsymbol{K}^{n,m}\\ \boldsymbol{K}^{m,n} & \boldsymbol{K}^{m,m}\end{bmatrix}),\tag{1}$$

where $\boldsymbol{\mu}^n\in\mathbb{R}^n$, $\mu^n_i=u(\boldsymbol{X}^n_{i,:})$, and $\boldsymbol{\mu}^m\in\mathbb{R}^m$, $\mu^m_{i,:}=u(\boldsymbol{X}^m_{i,:})$, represent the output of the mean function on $\mathbb{D}$ and $\mathbb{V}$, respectively. $\boldsymbol{K}^{n,n}\in\mathbb{R}^{n\times n}$, $K^{n,n}_{i,j}=k(\boldsymbol{X}^n_{i,:},\boldsymbol{X}^n_{j,:})$, denotes the kernel matrix on $\mathbb{D}\times\mathbb{D}$, and $\boldsymbol{K}^{m,n}\in\mathbb{R}^{m\times n}$, $K^{m,n}_{i,j}=k(\boldsymbol{X}^m_{i,:},\boldsymbol{X}^n_{j,:})$, is the kernel matrix on $\mathbb{V}\times\mathbb{D}$. Without loss of generality, we can assume $\mu(\cdot)=0$ for any input because zero-mean is always possible by subtracting the sample mean. Therefore, the GP can be completely described by its kernel matrix. In particular, the Bayesian inference made by the GP given validation data $\boldsymbol{X}^m$ can be derived from

$$P(\boldsymbol{y}^m|\boldsymbol{X}^m,\boldsymbol{X}^n,\boldsymbol{y}^n)=\mathcal{N}(\boldsymbol{K}^{m,n}(\boldsymbol{K}^{n,n})^{-1}\boldsymbol{y}^n,\boldsymbol{K}^{m,m}-\boldsymbol{K}^{m,n}(\boldsymbol{K}^{n,n})^{-1}\boldsymbol{K}^{n,m}),\tag{2}$$

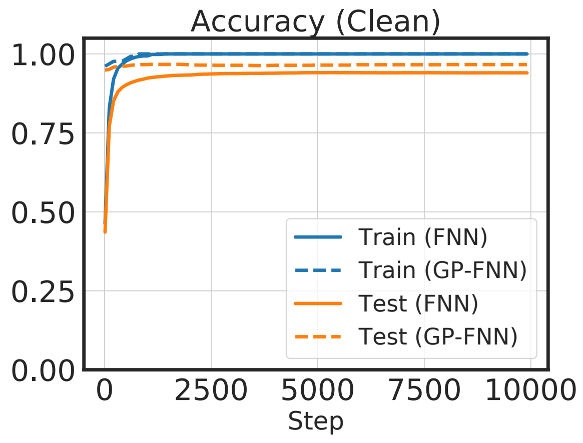
where $\boldsymbol{K}^{n,m}\in\mathbb{R}^{n\times m}$, $K^{n,m}_{i,j}=k(\boldsymbol{X}^n_{i,:},\boldsymbol{X}^m_{j,:})$, is the kernel matrix on $\mathbb{D}\times\mathbb{V}$. We call Eq. (1) and Eq. (2) the prior and posterior of the GP, respectively.

### A.2. Neural Tangent Kernels (NTKs)

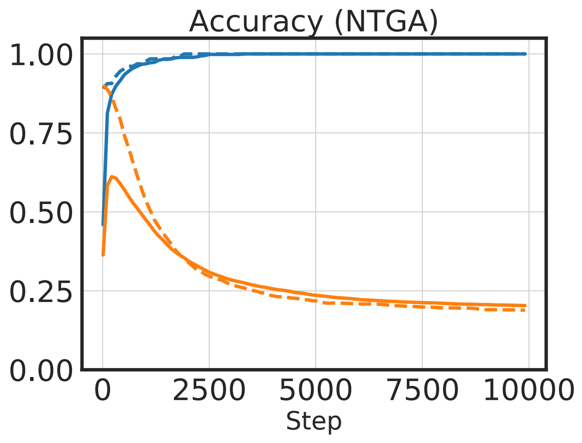
Following the Bayesian perspective, studies (Lee et al., 2018; Matthews et al., 2018) show that a class $\mathbb{F}$ of random initialized, fully-connected neural networks of the same architecture converge to a Gaussian Processes as the width of the networks grows, and the kernel (function) of the corresponding GP is called the Neural Network Gaussian Process (NNGP) kernel. The GP prior models the ensemble $\mathbb{F}$ of the networks at initialization, while the GP posterior indicates the distribution of predictions made by the networks in $\mathbb{F}$ in a full Bayesian fashion. Nonetheless, NNGP kernels provide no link to the behavior of the GP during training by gradient descent.

Recently, the studies (Jacot et al., 2018; Lee et al., 2019; Chizat et al., 2019) build a relationship between the GP prior and the distribution of the networks under gradient descent. Basically, as its width grows, a trained network can be approximated by its first-order Taylor expansion w.r.t. its initial parameters, and each step of gradient descent minimizing the Mean

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



(a) Clean



(b) NTGA

*Figure 1.* Training and test dynamics of a fully-connected network (FNN) and the mean (GP-FNN) of the corresponding GP with NTK on (a) clean and (b) poisoned data. As we can see, GP-FNN approximates the behavior of FNN closely.

Squared Error (MSE) loss can be regarded as an affine transformation of the linear approximation. This allows the learning dynamics of $\mathbb{F}$ to be described by a GP with a kernel known as the Neural Tangent Kernel (NTK). Let $f(\cdot\,;\boldsymbol{\theta}^{(t)}) \in \mathbb{F}$ be a network parametrized by $\boldsymbol{\theta}^{(t)}$ at time step $t$ during the gradient descent training. The finite-width NTK is defined by $k(\boldsymbol{x}^i, \boldsymbol{x}^j) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}^i; \boldsymbol{\theta}^{(0)})^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}^j; \boldsymbol{\theta}^{(0)})$ where $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ represent two data points. The study (Jacot et al., 2018) proves that as the width of the network grows into infinity, the NTK converges to a deterministic function and stays the same during training. The $k(\boldsymbol{x}^i, \boldsymbol{x}^j)$ only depends on the network architecture, non-linearity, loss function, and is independent of a particular initialization. Therefore, it represents a similarity score between $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ from the network class's ($\mathbb{F}$'s) point of view.

Denote the mean of the GP by $\bar{f}$, the expected output of the ensemble $\mathbb{F}$ over the training set $\mathbb{D}$ and validation set $\mathbb{V}$ respectively evolve with $t$ by following

$$\bar{f}(\boldsymbol{X}^n; \boldsymbol{K}^{n,n}, \boldsymbol{y}^n, t) = (\boldsymbol{I} - e^{-\eta \boldsymbol{K}^{n,n} t})\boldsymbol{y}^n, \tag{3}$$

$$\bar{f}(\boldsymbol{X}^m; \boldsymbol{K}^{m,n}, \boldsymbol{K}^{n,n}, \boldsymbol{y}^n, t) = \boldsymbol{K}^{m,n}(\boldsymbol{K}^{n,n})^{-1}(\boldsymbol{I} - e^{-\eta \boldsymbol{K}^{n,n} t})\boldsymbol{y}^n. \tag{4}$$

Since $k(\cdot\,,\cdot)$ is independent of $\boldsymbol{y}^n$, the above equations can be readily extended to approximate the expected output of $\mathbb{F}$ consisting of vector-valued networks $f : \mathbb{R}^d \to \mathbb{R}^c$ of $c$ label dimensions:

$$\bar{f}(\boldsymbol{X}^n; \boldsymbol{K}^{n,n}, \boldsymbol{Y}^n, t) = (\boldsymbol{I} - e^{-\eta \boldsymbol{K}^{n,n} t})\boldsymbol{Y}^n, \tag{5}$$

$$\bar{f}(\boldsymbol{X}^m; \boldsymbol{K}^{m,n}, \boldsymbol{K}^{n,n}, \boldsymbol{Y}^n, t) = \boldsymbol{K}^{m,n}(\boldsymbol{K}^{n,n})^{-1}(\boldsymbol{I} - e^{-\eta \boldsymbol{K}^{n,n} t})\boldsymbol{Y}^n, \tag{6}$$

where $\boldsymbol{Y}^n \in \mathbb{R}^{n \times c}$ is the label matrix in the ground truth. By evaluating Eq. (6), we can "foresee" the predictions made by a network without actually training it. As Figure 1 shows, the mean (named GP-FNN) of a GP with NTK closely approximates the behavior of a trained fully-connected network (named FNN). The gap between GP-FNN and FNN is due to the finite width, and it becomes smaller as the width increases (Lee et al., 2020).

The theory of NTKs has been extended to support a wide range of network architectures, including convolutional networks (Novak et al., 2018; Garriga-Alonso et al., 2018; Yang, 2019a), recurrent networks (Yang, 2019b; Alemohammad et al., 2020), networks with the attention mechanism (Hron et al., 2020), and other architectures (Yang, 2019b; Arora et al., 2019).

## B. Experiment Settings

Here, we provide more details about our experiment settings.

## B.1. Surrogates

Each poisoning algorithm under evaluation is equipped with two surrogates and can use one of them to generate the poisoned data. In NTGA, we use the mean predictions, named GP-FNN and GP-CNN, of two Gaussian Processes (GPs), which correspond to the infinite-width fully-connected network and the infinite-channel convolutional neural network, to craft the poisoned data. Both the GPs assume the networks are trained using gradient descent minimizing the Mean Squared Error (MSE) loss. We set weight variance $\sigma_w = 1.76$ and bias variance $\sigma_b = 0.18$ for all surrogate models.

**GP-FNN.** All element networks have the following architecture:

$$\begin{aligned}&\text{Dense}^1(\infty) \to \text{Erf} \to\\&\text{Dense}^2(\infty) \to \text{Erf} \to\\&\text{Dense}^3(\infty) \to \text{Erf} \to\\&\text{Dense}^4(\infty) \to \text{Erf} \to\\&\text{Dense}^5(\infty) \to \text{Erf},\end{aligned}$$

where Dense$(i)$ denotes a fully-connected layer with $i$ neurons.

**GP-CNN.** All element networks are of the following architecture:

$$\begin{aligned}&\text{Conv}^1(\infty) \to \text{ReLU} \to\\&\text{Conv}^2(\infty) \to \text{ReLU} \to\\&\text{Flatten} \to\\&\text{Dense}^1(\infty) \to \text{ReLU} \to\\&\text{Dense}^2(\infty) \to \text{ReLU} \to\\&\text{Dense}^3(\infty) \to \text{ReLU},\end{aligned}$$

where Conv$(d)$ denotes a convolutional layer of $d$ channels with the (2, 2) stride and "SAME" padding. All convolutional layers use the (5, 5) filter size. No pooling layer is applied to avoid the high computation cost induced by the corresponding NTK kernel.

The surrogates of RFA (Chan-Hon-Tong, 2019) and DeepConfuse (Feng et al., 2019) are two neural networks, named S-FNN and S-CNN, which have the same architectures as an element network of GP-FNN and GP-CNN, respectively, except with finite width and channels. We train S-FNNs and S-CNNs by following the original papers. More precisely, for RFA, we first train the surrogate using the training set and then retrain the last layer on the validation data using gradient ascent to decrease the validation accuracy. Meanwhile, for DeepConfuse, we use the code released by the authors[1].

**S-FNN.** The network has the following architecture:

$$\begin{aligned}&\text{Dense}^1(512) \to \text{Erf} \to\\&\text{Dense}^2(512) \to \text{Erf} \to\\&\text{Dense}^3(512) \to \text{Erf} \to\\&\text{Dense}^4(512) \to \text{Erf} \to\\&\text{Dense}^5(10) \to \text{Erf}.\end{aligned}$$

**S-CNN.** The network has the following architecture:

$$\begin{aligned}&\text{Conv}^1(64) \to \text{ReLU} \to\\&\text{Conv}^2(64) \to \text{ReLU} \to\\&\text{Flatten} \to\\&\text{Dense}^1(384) \to \text{ReLU} \to\\&\text{Dense}^2(192) \to \text{ReLU} \to\\&\text{Dense}^3(10) \to \text{ReLU}.\end{aligned}$$

## B.2. Target Networks

We consider five networks as our target models, including FNN, CNN, FNN-ReLU, ResNet18 (He et al., 2016), and DenseNet121 (Huang et al., 2017). FNN and CNN have identical architectures to the surrogates S-FNN and S-CNN used by

[1]See https://github.com/kingfengji/DeepConfuse.

RFA, while FNN-ReLU has the same architecture as FNN except using the ReLU nonlinearity. Different from FNN/CNN where the MSE loss is used, the FNN-ReLU, ResNet18, and DenseNet121 are trained with the cross-entropy loss. We use a Stochastic Gradient Descent (SGD) optimizer to minimize their losses on the clean and poisoned training data for all target networks except for CNN and FNN-ReLU, where we use the Adam (Kingma & Ba, 2015) optimizer to stabilize the training process. We also enable learning rate scheduling and data augmentation when training ResNet18 and DenseNet121. The detailed settings slightly vary on the MNIST and CIFAR-10 datasets.

**MNIST.** We train all networks for 50 epochs. The learning rate is set to 1e-3. We set the batch size to 64.

**CIFAR-10.** We train the models FNN, CNN, and FNN-ReLU for 100 epochs. We use the Adam optimizer (Kingma & Ba, 2015) for FNN-ReLU. For ResNet18, we use a batch size of 128 and train the network for 200 epochs with a cosine annealing learning rate schedule (Loshchilov & Hutter, 2016) where the learning rate is set to 0.1 initially and then decay to 0 in the last iteration. The momentum is set to 0.9. We also apply data augmentation including normalization, random flipping, and random cropping with padding size 4. For DenseNet121, we use a batch size of 128 and train the network for 100 epochs with the abovementioned learning rate schedule and data augmentation techniques.

**ImageNet.** We use the same setting as CIFAR-10, except the initial learning rate and a number of epochs. We train all networks for 50 epochs and the learning rate is set to 1e-3.

### B.3. Poisoning Methods and Attacks

For the MNIST and CIFAR-10 datasets, we randomly split 10K examples from the training set as $\mathbb{V}$ and use the rest as $\mathbb{D}$. For the 2-class ImageNet, we use the same setting in (Feng et al., 2019) and split 190 examples from training set as $\mathbb{V}$. We use the full test sets to report performance. We implement the B-NTGA described in Section 3.3 with $b$ equals 5K, 4K, 2.22K on MNIST, CIFAR-10, and ImageNet, respectively. NTGA uses slightly different settings to generate an attack on different datasets.

**MNIST.** We set the maximum allowable perturbations $\epsilon$ to 0.3 measured by $l_\infty$ distance. By default, we run Algorithm 1 for $r = 10$ iterations with step size $\eta = 0.04$. In this file, we also consider the attacks generated by NTGA with just a single iteration $r = 1$ but large step $\eta = 0.3 = \epsilon$.

**CIFAR-10.** We set $\epsilon = 8/255$. By default, we use the attacks generated by NTGA with $r = 10$ iterations and step size $\eta =$ 3e-3. For $r = 1$, we set the step size to $\eta = 8/255 = \epsilon$.

**ImageNet.** We set $\epsilon = 0.1$. By default, we use the attacks generated by NTGA with $r = 10$ iterations and step size $\eta = 0.011$. For $r = 1$, we set the step size to $\eta = 0.1 = \epsilon$.

We implement RFA and DeepConfuse and its attacks using the same settings whenever applicable.

## C. More Experiments

In this section, we conduct more experiments to understand NTGA.

### C.1. Transferability across Initial Weights, Depths, and Training Steps

We further evaluate the transferability of NTGA attacks across the target networks with different initial weights, depths, and training steps $t$. We use GP-FNN as the surrogate in Eq. (4) in the main paper. To quickly access the test accuracy of various target networks, we use another GP, called GP-FNN', as the target network. The GP-FNN' is identical to GP-FNN except that its hyperparameter $\sigma_w$ (the variance of the initial weights of the networks in $\mathbb{F}$), depth of the networks in $\mathbb{F}$, and $t$ (the training steps) are different. Both $\sigma_w$ and the depth affect the kernel construction of GP-FNN'.

Figures 2 and 3 show the effectiveness of GP-FNN-based attacks against GP-FNN' on the MNIST and CIFAR-10 datasets, respectively.[2] Each plot shows the contour of the test accuracy of GP-FNN' over 1,200 combinations of $\sigma_w$ (40 levels) and depth (30 levels). The area between the two dashed white lines indicates the safe region for $\sigma_w$ to stay on the *critical line* (Poole et al., 2016; Schoenholz et al., 2016; Lee et al., 2018) that allows the GP-FNN' to have the best test performance. Different columns represent different $t$ in Eq. (6) used by GP-FNN' to make predictions.

[2]To save computations, we sample only 512 training data from each of the datasets to craft the attacks and report the test accuracy of GP-FNN'.

*Table 1.* The test accuracy of different attacks ($r = 1$) under gray-box settings.

| Dataset \Attack | Clean | RFA | Deep Confuse | NTGA (1) | NTGA (8) | NTGA (64) | NTGA (512) | NTGA (4096) | NTGA ($\infty$) |
|---|---|---|---|---|---|---|---|---|---|
| **Surrogate: *-FNN $\rightarrow$ Target: FNN** | | | | | | | | | |
| MNIST | 96.26±0.09 | 67.38±1.84 | - | 13.80±1.79 | **11.69±1.28** | 15.81±3.13 | 44.02±1.86 | 74.78±0.76 | 91.85±0.24 |
| CIFAR-10 | 49.57±0.12 | 37.83±0.52 | - | 35.90±0.27 | 34.64±0.22 | 32.62±0.21 | 28.78±0.32 | **26.44±0.22** | 39.69±0.29 |
| ImageNet | 91.60±0.49 | 87.80±1.47 | - | 78.60±1.96 | **74.20±7.05** | 86.20±2.23 | 89.40±1.62 | 89.80±1.72 | 90.60±0.49 |
| **Surrogate: *-CNN $\rightarrow$ Target: CNN** | | | | | | | | | |
| MNIST | 99.49±0.02 | 88.21±4.26 | 46.21±5.14 | **18.40±3.80** | 20.16±6.11 | 30.50±5.41 | 81.13±1.55 | 94.44±0.65 | 96.05±0.57 |
| CIFAR-10 | 78.12±0.11 | 74.07±0.37 | 44.84±1.19 | 40.60±0.78 | **39.97±0.37** | 43.95±0.35 | 51.42±0.21 | 51.89±0.27 | 59.29±0.32 |
| ImageNet | 96.00±0.63 | 95.20±0.40 | 93.00±0.63 | **77.40±2.50** | 83.20±1.60 | 84.60±1.85 | 88.80±1.17 | 90.60±1.02 | 89.80±1.17 |

As we can see, the GP-FNN-based attacks works well against GP-FNN'. Note that all the attacks are generated by GP-FNN over only one particular combination of $\sigma_w^2 = 1.76$ and depth $= 5$. This verifies that the GP-FNN surrogate, which models an infinite ensemble of infinite-width networks, is indeed representative of networks of different types (and even other GPs) and can lead to better attack transferability. We can also see the effect of time step $t$, where the attacks based on a smaller (resp. larger) $t$ work better against the GP-FNN' with a smaller (resp. larger) $t$.

## C.2. Fast Gray- and Black-box Attacks

The number of iterations, $r$, in Algorithm 1 in the main paper affects the generating speed and strength of an attack. Here, we evaluate the strength of the "fastest" attacks generated by different poisoning methods using the smallest $r = 1$ (but largest step size $\eta = \epsilon$). The running time analysis can be found in Table 3. Tables 1 and 2 show the results under gray- and black-box settings, respectively. Similar to that in the main paper ($r = 10$), NTGA significantly outperforms the baselines. Moreover, the strength of the NTGA attacks does not seem to significantly change when $r$ is reduced from 10 to 1 (with enlarged step size $\eta = \epsilon$). This shows that the loss surface of $L$ w.r.t. $\boldsymbol{P}$ in Eq. (4) in the main paper is smooth and thus the gradients at an initial $\boldsymbol{P}$ (see Algorithm 1 in the main paper) are good enough to lead to an effective attack.

## C.3. Early-Stop Points

In Section 4.1 of the main paper, we showed that the early-stop points of the GP-FNN surrogate are consistent with those of the target model FNN on the CIFAR-10 dataset. Figure 4 gives similar results on the MNIST dataset (cf. the first row of Table 1 in the main paper).

## C.4. More Visualization

In Figures 5 and 7, we visualize some poisoned images $\boldsymbol{X}_{i,:} + \boldsymbol{P}_{i,:}$ from the MNIST and CIFAR-10 datasets as well as their normalized perturbations $\boldsymbol{P}_{i,:}/\|\boldsymbol{P}_{i,:}\|$ based on the *-FNN surrogates. Some poisoned images crafted with the *-CNN surrogates can be found in Figures 6 and 8. As we can see, the perturbations generated by RFA and DeepConfuse are of high and relatively low frequencies respectively, whereas 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 the neural network tends to learn low-frequency patterns at the early stage of training.

Interestingly, the perturbations generated with the *-FNN surrogates look similar to those with the *-CNN surrogates. We hypothesize it is because of the fact that infinite-channel convolutional neural networks without global average pooling behave similarly to infinite-width fully-connected networks (Xiao et al., 2020).

### C.5. Results with Standard Deviation

In the main paper, we omit the variance of some results due to space limitation. Here we show the full results with standard deviation in Table 3, and 4.

## References

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.

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.

Feng, J., Cai, Q.-Z., and Zhou, Z.-H. Learning to confuse: generating training time adversarial data with auto-encoder. *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.

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

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.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proc. of ICLR (Poster)*, 2015.

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.

Lee, J., Schoenholz, S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J. Finite versus infinite neural networks: an empirical study. *Proc. of NeurIPS*, 2020.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

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.

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.

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.

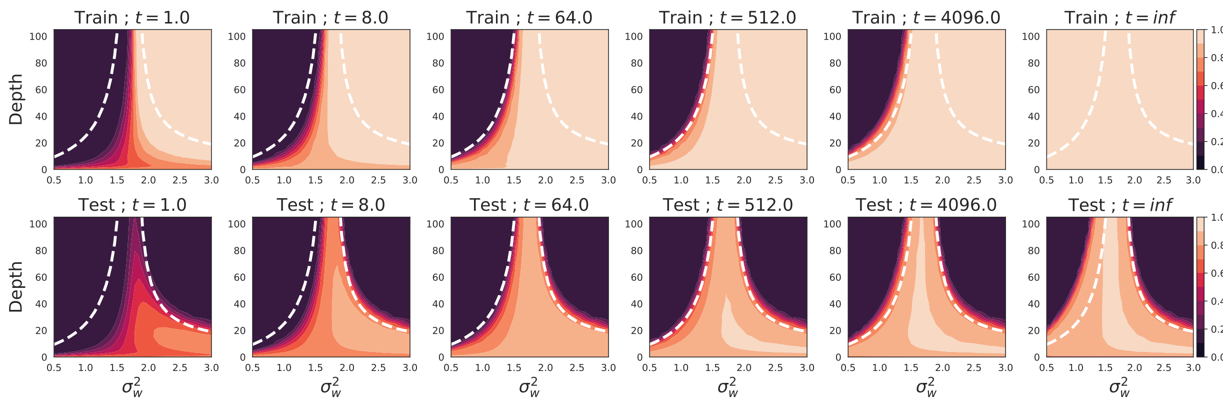
Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *Proc. of ICML*. PMLR, 2019.

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

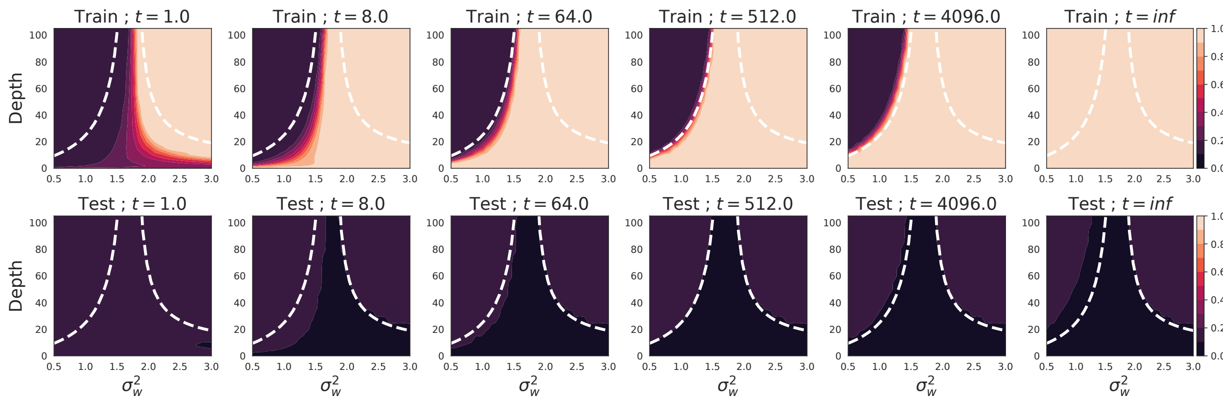
Xiao, L., Pennington, J., and Schoenholz, S. Disentangling trainability and generalization in deep neural networks. In *Proc. of ICML*. PMLR, 2020.

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.

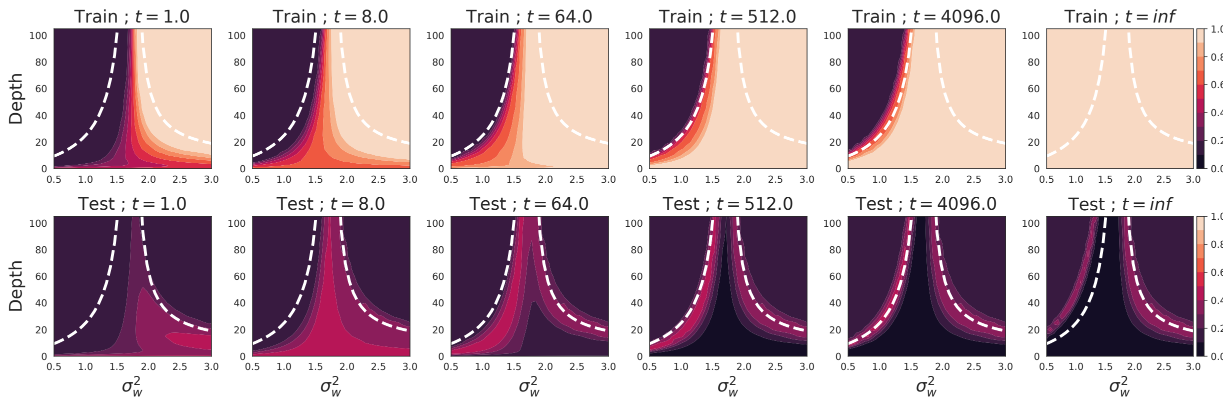
Yang, G. Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *arXiv preprint arXiv:1910.12478*, 2019b.



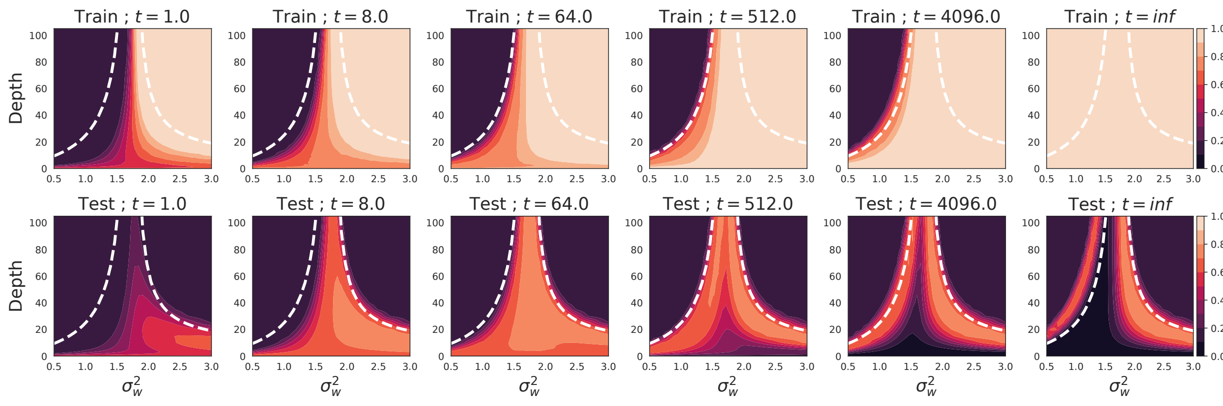
(a) Clean



(b) NTGA(64)

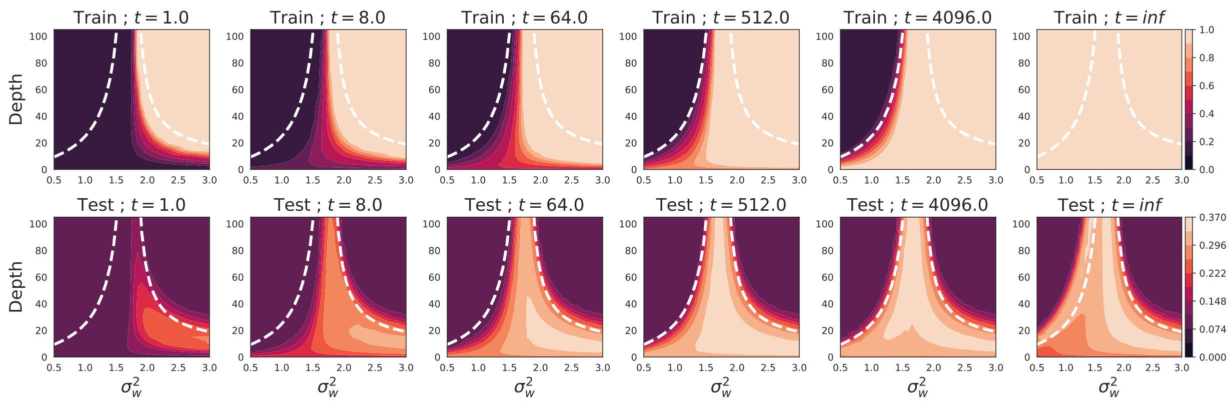


(c) NTGA(4096)

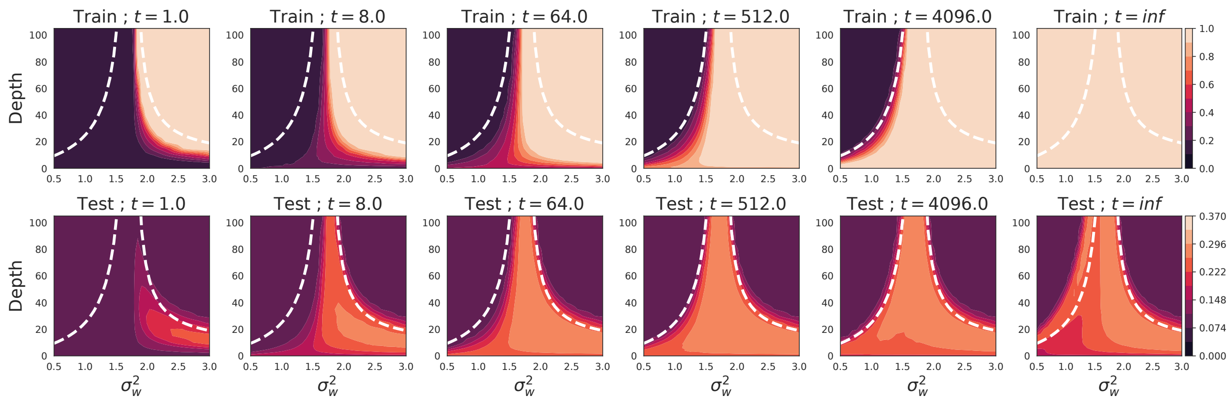


(d) NTGA($\infty$)

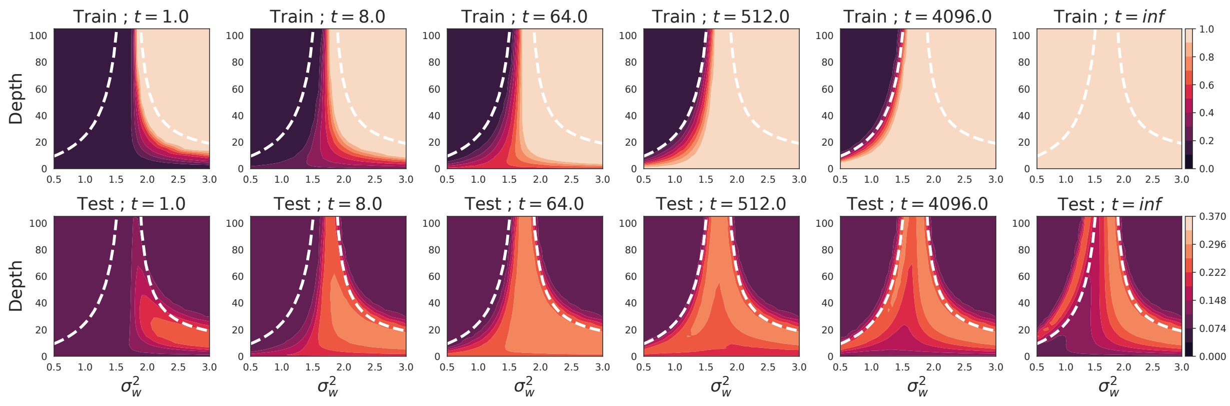
*Figure 2.* Training and test dynamics of GP-FNN' on the MNIST dataset with different combinations of initial-weight variance $\sigma_w^2$ and depth.



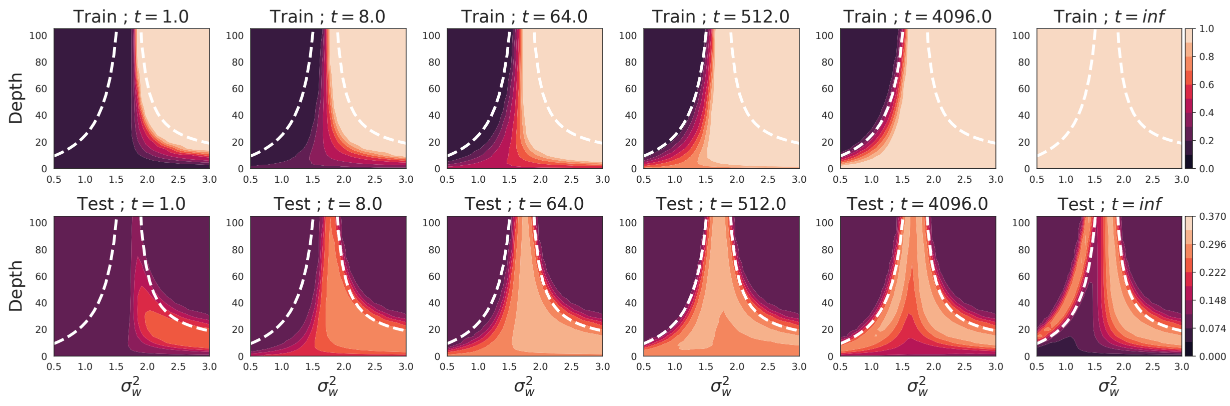
(a) Clean



(b) NTGA(64)



(c) NTGA(4096)



(d) NTGA($\infty$)

*Figure 3.* Training and test dynamics of GP-FNN' on the CIFAR-10 dataset with different combinations of initial-weight variance $\sigma_w^2$ and depth.

*Table 2.* The test accuracy of different attacks ($r = 1$) under black-box settings.

| Target \Attack | Clean | RFA | Deep Confuse | NTGA (1) | NTGA (8) | NTGA (64) | NTGA (512) | NTGA (4096) | NTGA ($\infty$) |
|---|---|---|---|---|---|---|---|---|---|
| **Surrogate: *-FNN** | | | | | | | | | |
| CNN | 99.49±0.02 | 93.32±1.79 | - | **20.15±2.98** | 21.58±6.45 | 25.37±3.56 | 81.55±2.98 | 94.41±0.39 | 96.76±0.72 |
| FNN-ReLU | 97.87±0.10 | 83.23±0.73 | - | 7.31±1.72 | **6.35±0.96** | 6.70±0.62 | 28.25±1.36 | 66.94±0.81 | 90.41±0.20 |
| **Surrogate: *-CNN** | | | | | | | | | |
| FNN | 96.26±0.09 | 65.53±2.58 | 15.48±0.94 | **10.62±1.72** | 12.86±1.95 | 18.22±1.73 | 48.13±1.61 | 78.33±0.86 | 89.49±0.33 |
| FNN-ReLU | 97.87±0.10 | 78.81±1.21 | 17.50±1.49 | **2.30±0.46** | 3.37±1.20 | 5.64±0.75 | 40.08±0.32 | 73.86±1.40 | 86.44±0.14 |

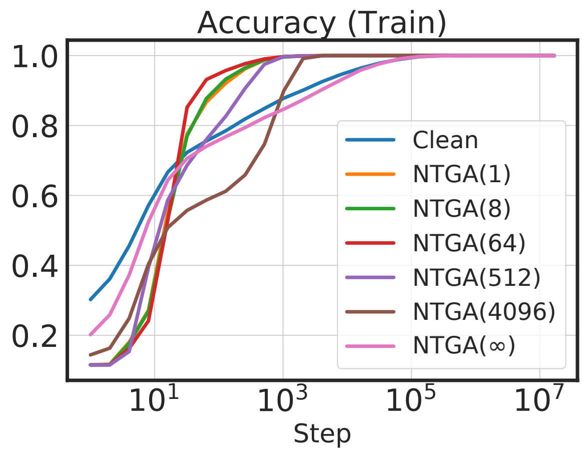
(a) MNIST

| Target \Attack | Clean | RFA | Deep Confuse | NTGA (1) | NTGA (8) | NTGA (64) | NTGA (512) | NTGA (4096) | NTGA ($\infty$) |
|---|---|---|---|---|---|---|---|---|---|
| **Surrogate: *-FNN** | | | | | | | | | |
| CNN | 78.12±0.11 | 75.15±0.58 | - | 47.49±0.76 | 45.55±0.91 | 44.10±1.22 | 48.19±1.04 | **46.03±0.33** | 62.80±0.19 |
| FNN-ReLU | 54.55±0.29 | 41.17±0.39 | - | 39.96±0.27 | 38.74±0.41 | 36.73±0.32 | 31.91±0.48 | **26.83±0.84** | 44.06±0.17 |
| ResNet18 | 91.92±0.39 | 90.52±0.19 | - | 39.49±1.94 | 37.85±0.97 | **37.44±1.81** | 41.46±1.66 | 47.37±1.30 | 84.98±1.22 |
| DenseNet121 | 92.71±0.15 | 89.09±0.21 | - | 43.33±2.15 | 44.10±1.83 | **43.01±2.41** | 49.68±3.78 | 57.32±0.64 | 86.24±0.55 |
| **Surrogate: *-CNN** | | | | | | | | | |
| FNN | 49.57±0.12 | 41.00±0.39 | 32.59±0.77 | 28.87±0.28 | **28.45±0.37** | 29.81±0.49 | 28.82±0.20 | 29.36±0.20 | 37.11±0.12 |
| FNN-ReLU | 54.55±0.29 | 47.67±0.66 | 35.06±0.39 | 33.56±0.38 | 31.47±0.27 | 33.51±0.30 | 33.10±0.20 | **30.29±0.40** | 37.24±0.36 |
| ResNet18 | 91.92±0.39 | 91.44±0.14 | 41.10±1.15 | 33.58±2.12 | **32.46±1.40** | 36.15±1.28 | 45.78±3.46 | 55.11±1.68 | 73.97±0.78 |
| DenseNet121 | 92.71±0.15 | 90/.35±0.29 | 54.99±7.33 | 40.82±0.37 | **40.20±2.69** | 40.42±1.03 | 56.37±1.67 | 63.72±0.87 | 76.95±1.54 |

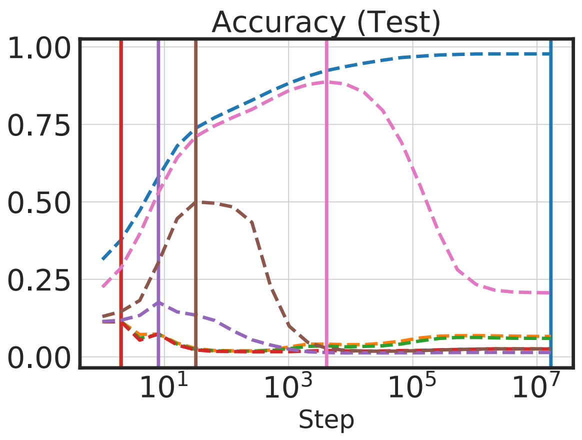
(b) CIFAR-10

| Target \Attack | Clean | RFA | Deep Confuse | NTGA (1) | NTGA (8) | NTGA (64) | NTGA (512) | NTGA (4096) | NTGA ($\infty$) |
|---|---|---|---|---|---|---|---|---|---|
| **Surrogate: *-FNN** | | | | | | | | | |
| CNN | 96.00±0.63 | 93.80±2.56 | - | 74.20±1.94 | **68.20±4.96** | 81.00±4.000 | 85.40±1.36 | 89.80±1.17 | 91.20±0.40 |
| FNN-ReLU | 92.20±0.40 | 87.00±0.63 | - | 78.80±2.56 | **74.00±5.62** | 84.40±0.35 | 86.60±1.02 | 88.80±0.75 | 91.00±0.15 |
| ResNet18 | 99.80±0.40 | 91.20±1.17 | - | **74.60±2.15** | 85.00±0.89 | 94.80±0.75 | 92.40±1.02 | 89.40±1.62 | 90.00±2.61 |
| DenseNet121 | 98.40±0.49 | 89.60±0.49 | - | **72.60±4.88** | 74.40±4.50 | 88.20±2.32 | 89.60±1.74 | 88.60±1.36 | 88.80±1.47 |
| **Surrogate: *-CNN** | | | | | | | | | |
| FNN | 91.60±0.49 | 89.40±1.50 | 90.80±0.40 | **77.00±1.10** | 80.20±2.40 | 87.80±0.75 | 89.20±1.17 | 90.20±0.40 | 89.60±1.02 |
| FNN-ReLU | 92.20±0.40 | 88.60±0.49 | 91.00±0.08 | **80.00±0.89** | 82.40±2.15 | 87.40±1.36 | 89.20±0.75 | 89.60±0.49 | 90.00±0.05 |
| ResNet18 | 99.80±0.40 | 92.20±1.60 | 92.80±1.72 | **77.60±2.06** | 88.20±1.33 | 92.80±1.47 | 89.60±3.88 | 91.80±1.94 | 89.60±3.07 |
| DenseNet121 | 98.40±0.49 | 90.00±1.67 | 92.80±2.32 | **77.00±2.19** | 77.40±2.50 | 89.60±1.36 | 89.40±1.20 | 87.80±2.04 | 88.80±0.75 |

(c) ImageNet



(a)



(b)

*Figure 4.* (a) Training and (b) test dynamics of GP-FNN on MNIST with different $t$. Vertical lines represent the early-stop points.

*Table 3.* Trade-off between speed and collaboration by varying block size $b$. (a) $r = 10$ (b) $r = 1$.

| $b$ | FNN | FNN-ReLU | CNN | ResNet18 | DenseNet121 | time |
|---|---|---|---|---|---|---|
| **Surrogate: GP-FNN** | | | | | | |
| 1 | 49.20±0.27 | 53.95±0.20 | 77.75±0.16 | 89.78±0.64 | 91.14±0.16 | **5.8 s** |
| 100 | 37.02±0.20 | 42.28±0.20 | 69.02±0.47 | 80.34±1.21 | 83.81±1.16 | 16.8 s |
| 1K | 22.84±0.64 | 27.85±0.56 | 47.33±0.52 | 49.61±1.56 | 58.40±1.98 | 3.5 m |
| 4K | **20.63±0.57** | **25.95±1.50** | **36.05±1.11** | **39.68±1.22** | **47.36±0.51** | 34 m |

(a)

| $b$ | FNN | FNN-ReLU | CNN | ResNet18 | DenseNet121 | time |
|---|---|---|---|---|---|---|
| **Surrogate: GP-FNN** | | | | | | |
| 1 | 49.41±0.35 | 53.71±0.23 | 77.78±0.42 | 77.78±0.42 | 88.88±0.73 | **0.6 s** |
| 100 | 37.08±0.16 | 42.75±0.33 | 68.46±0.26 | 77.75±0.16 | 89.78±0.64 | 1.7 s |
| 1K | 28.50±0.20 | 29.84±0.39 | 51.67±0.56 | 55.11±1.89 | 64.09±1.08 | 21.9 s |
| 4K | **26.44±0.22** | **26.83±0.84** | **46.03±0.33** | **47.37±1.30** | **57.32±0.64** | 3.5 m |

(b)

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

| $\sigma_b$ | $\sigma_w$ | FNN | FNN' | CNN | R18 | D121 |
|---|---|---|---|---|---|---|
| **Surrogate: GP-FNN** | | | | | | |
| 0.00 | 1.10 | 20.35±0.54 | 24.75±1.61 | 37.80±0.23 | **35.98±1.33** | **42.75±0.49** |
| 0.10 | 1.55 | **19.40±0.82** | **24.52±2.40** | 37.87±0.33 | 40.18±1.81 | 46.20±2.04 |
| 0.18 | 1.76 | 20.63±0.57 | 25.95±1.50 | **36.05±1.11** | 39.68±1.22 | 47.36±0.51 |
| 0.25 | 1.88 | 21.05±0.65 | 26.03±1.58 | 36.94±0.47 | 38.71±1.04 | 48.64±1.41 |

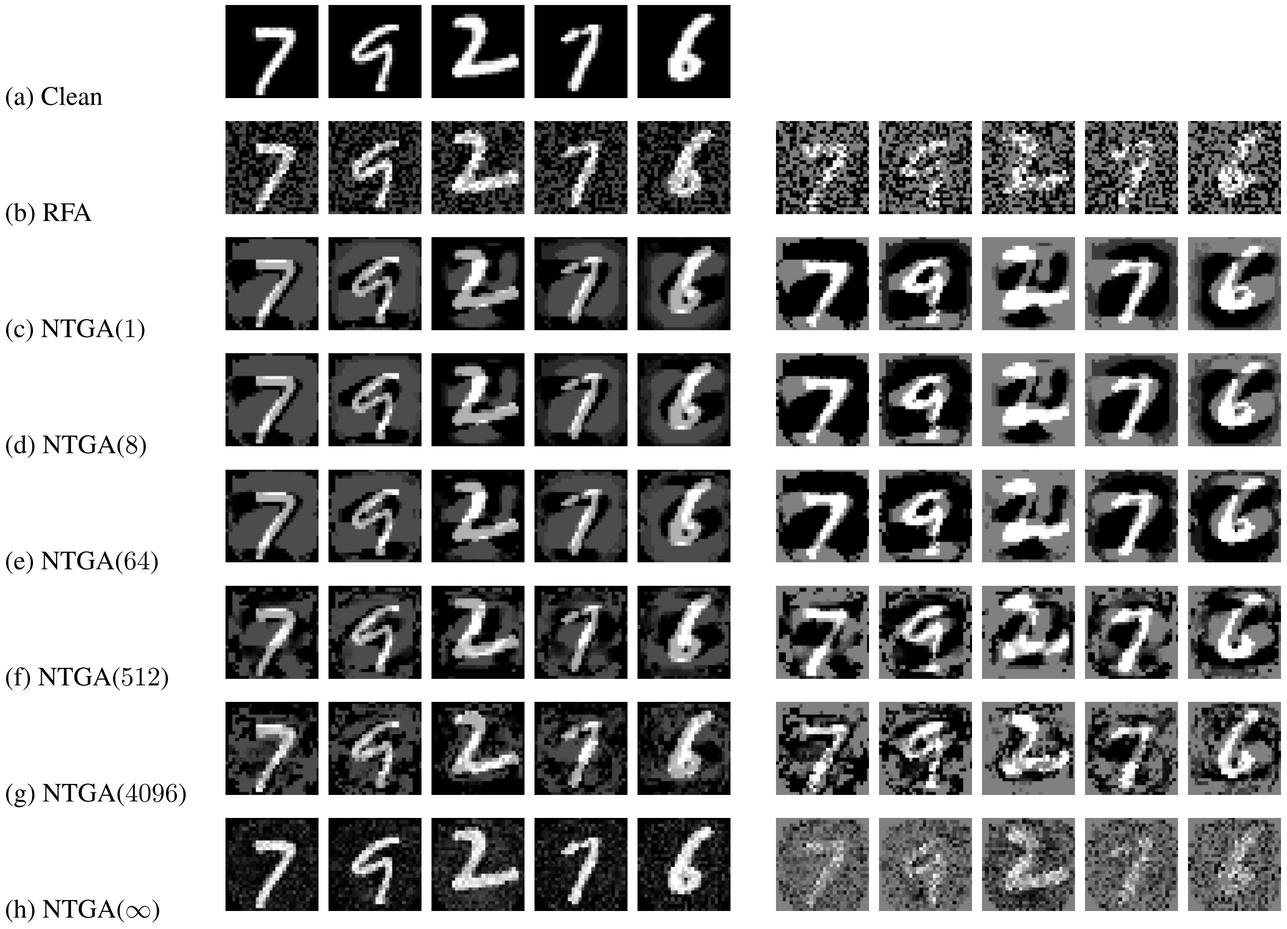


*Figure 5.* Visualization of some poisoned MNIST images (left) and their normalized perturbations (right) based on *-FNN surrogates.

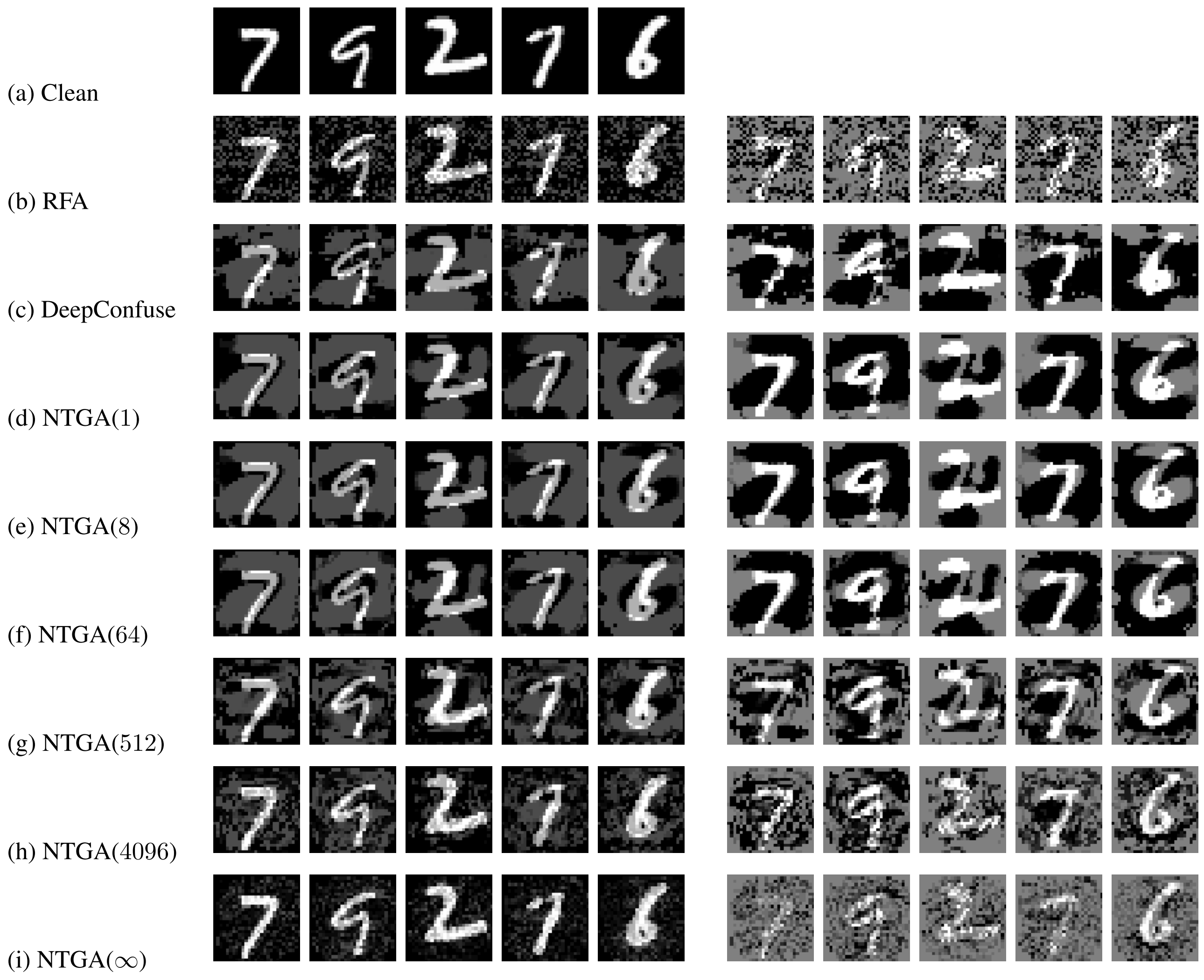


*Figure 6.* Visualization of some poisoned MNIST images (left) and their normalized perturbations (right) based on *-CNN surrogates.

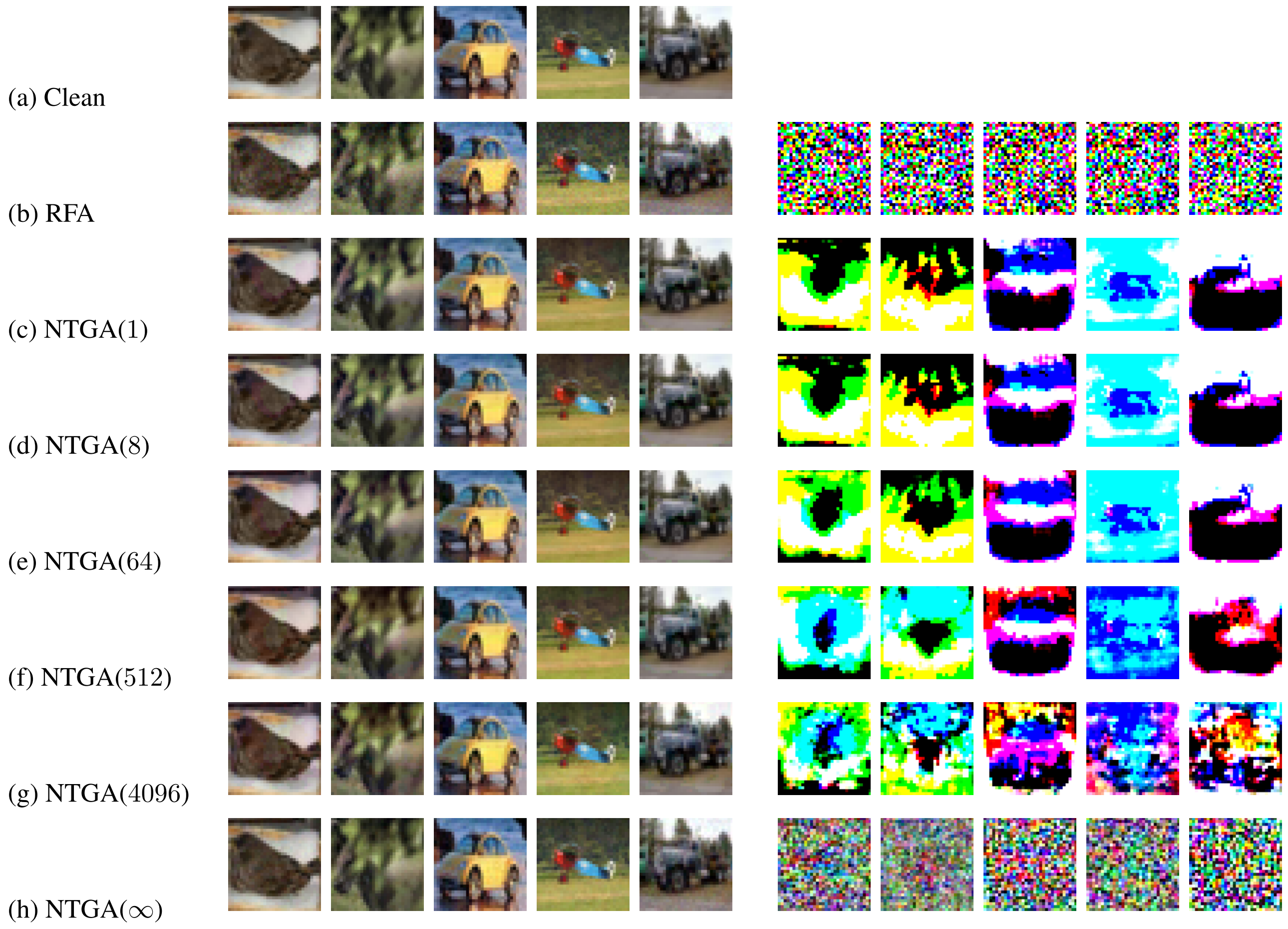


*Figure 7.* Visualization of some poisoned CIFAR-10 images (left) and their normalized perturbations (right) based on *-FNN surrogates.

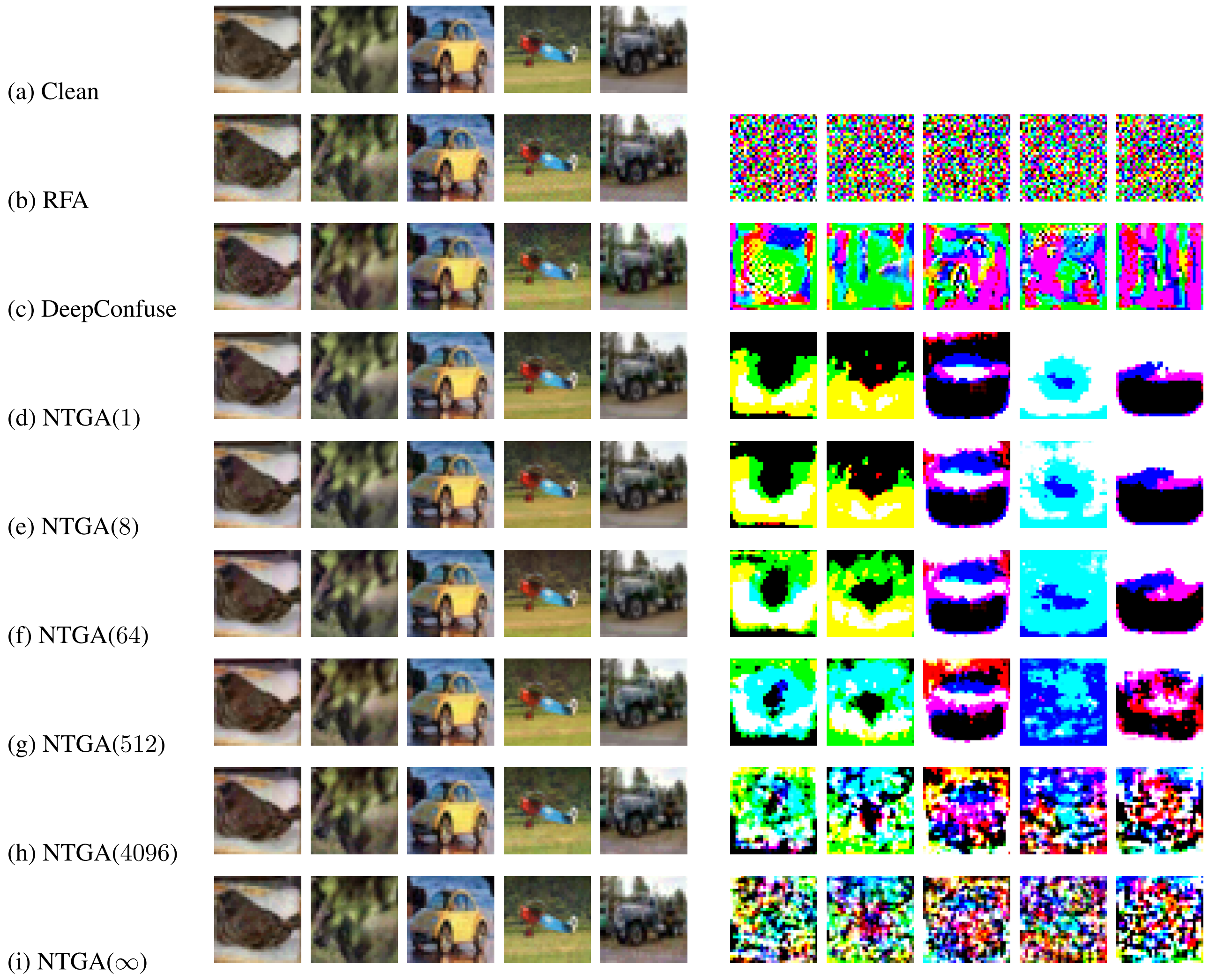


*Figure 8.* Visualization of some poisoned CIFAR-10 images (left) and their normalized perturbations (right) based on *-CNN surrogates.

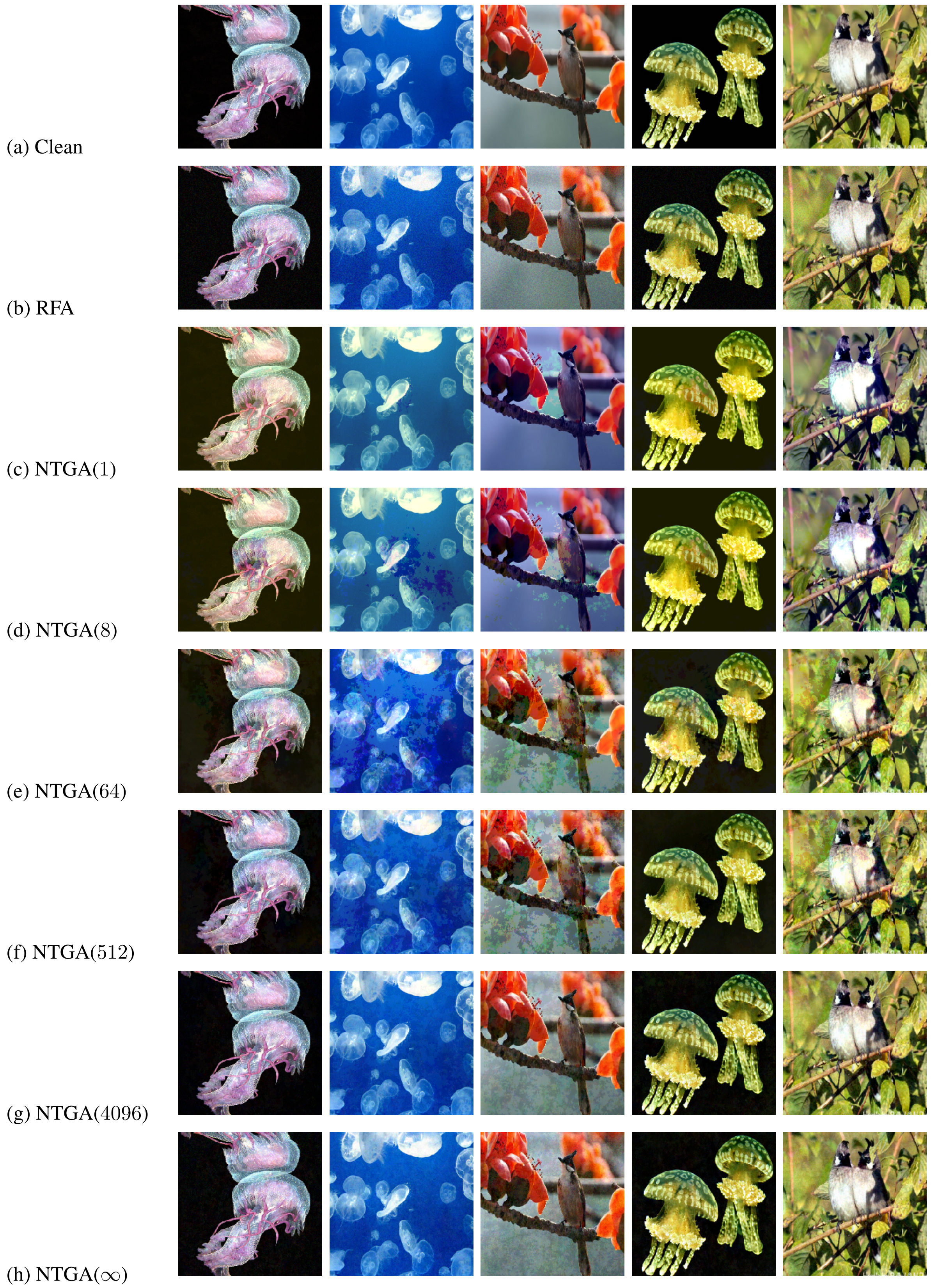


*Figure 9.* Visualization of some poisoned ImageNet images based on *-FNN surrogates.

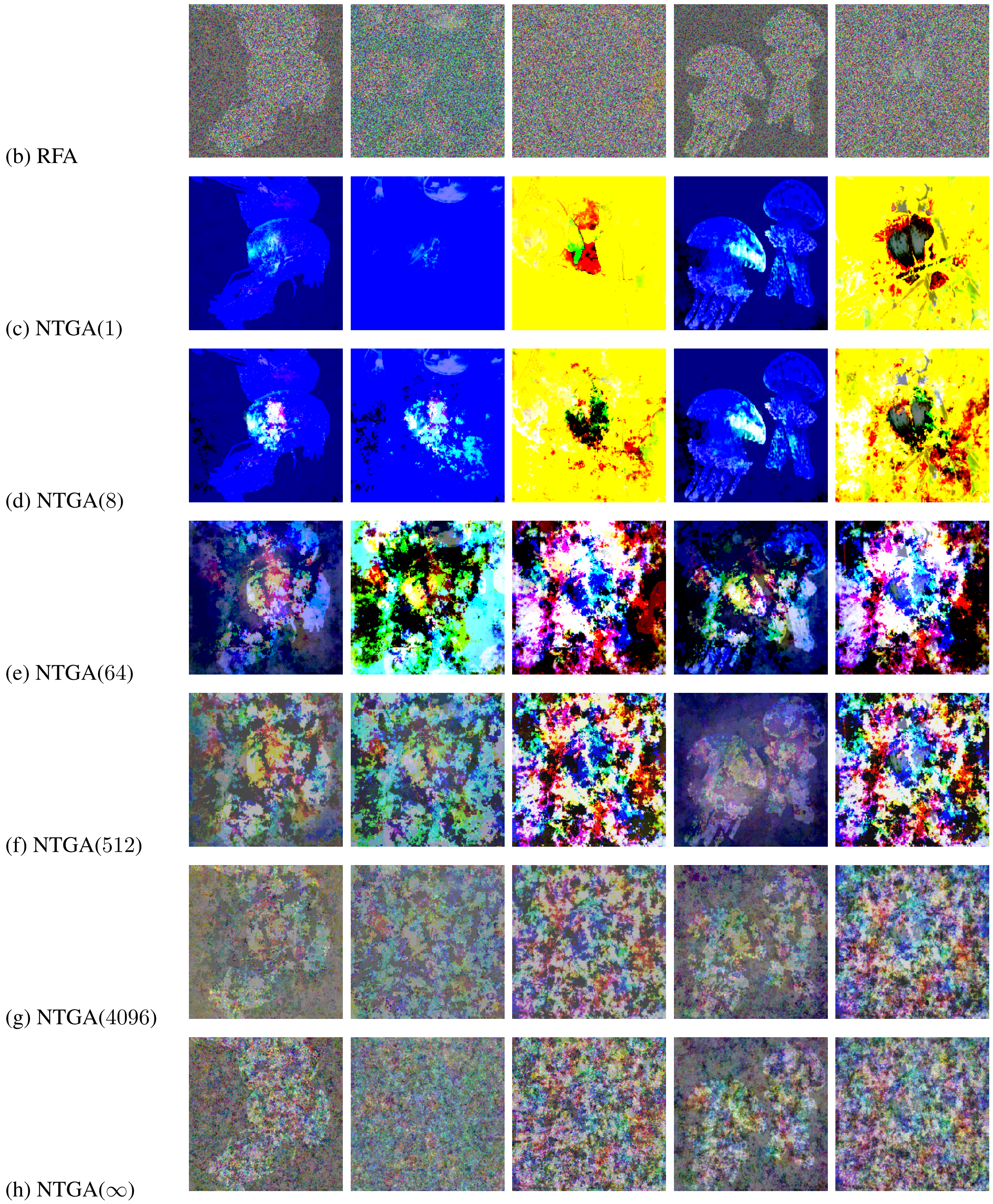
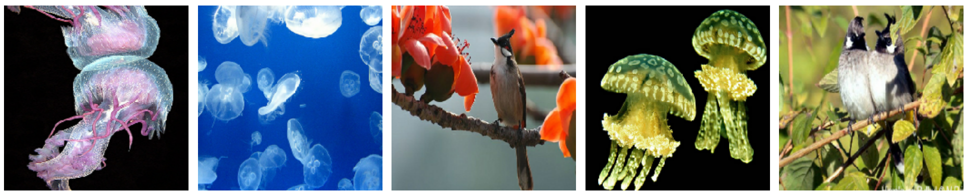
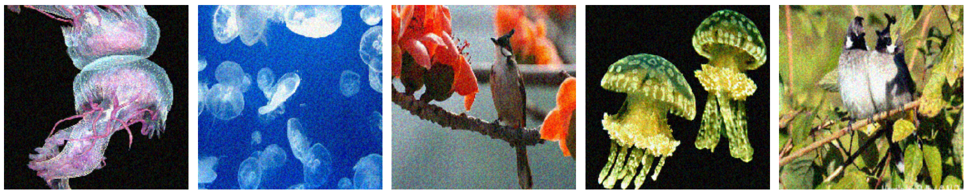
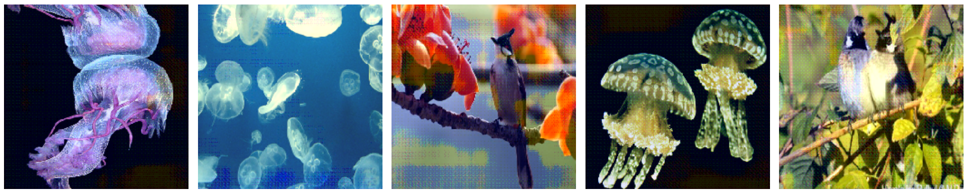
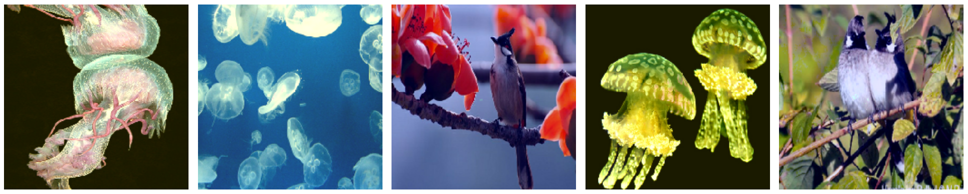


*Figure 10.* Visualization of normalized perturbations of some poisoned ImageNet based on *-FNN surrogates.


(a) Clean

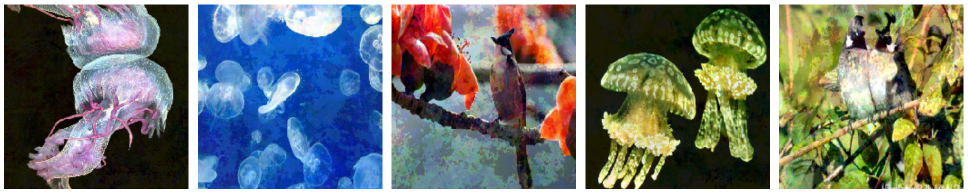

(b) RFA

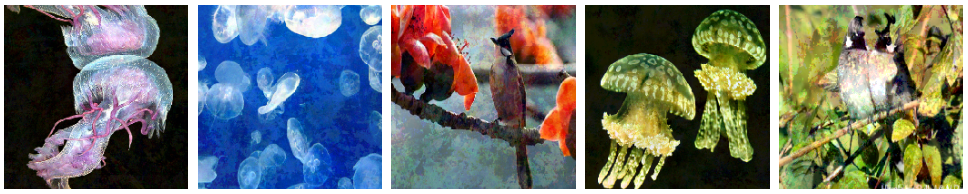

(c) DeepConfuse


(d) NTGA(1)

(e) NTGA(8)

(f) NTGA(64)


(g) NTGA(512)


(h) NTGA(4096)

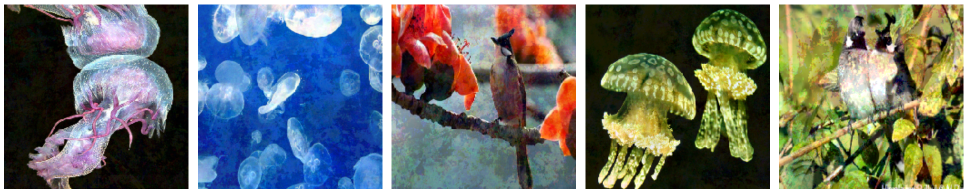

(i) NTGA($\infty$)

*Figure 11.* Visualization of some poisoned ImageNet images based on *-CNN surrogates.

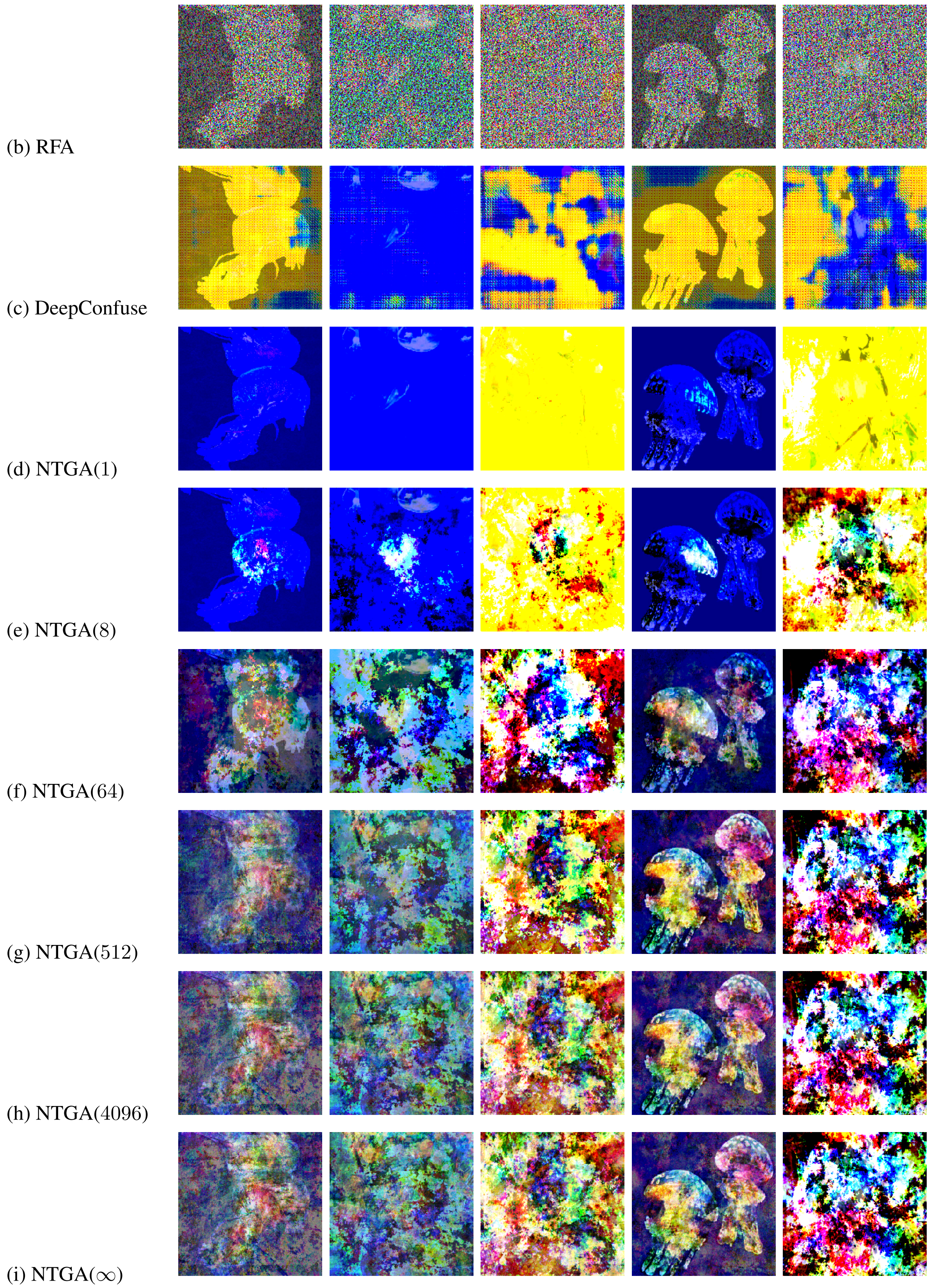


*Figure 12.* Visualization of normalized perturbations of some poisoned ImageNet based on *-CNN surrogates.