THE NTK ADVERSARY: AN APPROACH TO ADVERSARIAL ATTACKS WITHOUT ANY MODEL ACCESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial attacks carefully perturb natural inputs, so that a machine learning algorithm produces erroneous decisions on them. Most successful attacks on neural networks exploit gradient information of the model (either directly or by estimating it through querying the model). Harnessing recent advances in Deep Learning theory, we propose a radically different attack that eliminates that need. In particular, in the regime where the *Neural Tangent Kernel* theory holds, we derive a simple, but powerful strategy for attacking models, which in contrast to prior work, does not require any *access* to the model under attack, or any *trained* replica of it for that matter. Instead, we leverage the explicit description afforded by the NTK to maximally perturb the output of the model, using solely information about the model structure and the training data. We experimentally verify the efficacy of our approach, first on models that lie close to the theoretical assumptions (large width, proper initialization, etc.) and, further, on more practical scenarios, with those assumptions relaxed. In addition, we show that our perturbations exhibit strong transferability between models.

1 INTRODUCTION

Adversarial examples, a term coined by Szegedy et al. (2014), are inputs of a machine learning model that are carefully crafted so that the model is mistaken on them. Their existence, together with the empirically verified susceptibility of many classifiers to them (Szegedy et al., 2014; Carlini & Wagner, 2017), imply an important security headache for modern machine learning. The ultimate pursuit for robust and immune models calls for an extensive understanding of the different versions of adversarial examples that a model may encounter. In most cases, an adversarial perturbation (an instance of non random noise) is added to training or test data to form an adversarial example. This procedure is called an *adversarial attack*.

There has been a large body of work on designing such attacks. The most successful attacks exploit model information through its gradient with respect to the input. By perturbing the data in directions of the largest change in loss function, they are able to drop the performance of naive classifiers to almost zero (Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017; Kurakin et al., 2017; Madry et al., 2018). While successful, these gradient-derived attacks require access to the parameters of the trained model - which might be unrealistic for practical applications. To relax this requirement, so called *black-box* attacks only *query* the model. Observation of the model outputs allows to estimate the aforementioned gradients either by training substitute models (Papernot et al., 2016; 2017; Liu et al., 2017) or by finite-difference methods (Chen et al., 2017; Ilyas et al., 2018). Nonetheless, to some extend even these strategies require access to the model.

Our contribution: Is it possible to devise adversarial perturbations that require *neither access to the model nor training of a substitute model*? In this work we answer this question in the affirmative. We leverage recent advances in the theoretical understanding of trained neural networks to propose an attack that, surprisingly, does not use any model information whatsoever, apart from the model *architecture*. This powerful, highly agnostic attack is derived from **N**eural **T**angent **K**ernel theory (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019b), which allows to describe the dynamics of the gradient-descent based training procedure of a neural net as a kernel regression, with an architecture- and data-dependent (but parameter-agnostic) kernel (NTK). At a high-level of abstraction, our approach exploits the NTK description of the function the model has learned at the

end of training, and proceeds to maximally perturb this function. To illustrate more precisely, let $(\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, +1\}$ be an input sample, $\boldsymbol{\eta} \in \mathbb{R}^d$ a small perturbation, and $f : \mathbb{R}^d \to \mathbb{R}$ denote the model after training has converged. Then, as we show, in the NTK regime:

$$f(\mathbf{x} + \boldsymbol{\eta}) \approx f(\mathbf{x}) + \boldsymbol{\eta}^T \mathbf{z}, \tag{1}$$

for some $z \in \mathbb{R}^d$ that depends on the training data and the NTK kernel only. Although the above is derived for regression models, we show that this simple expression naturally leads to a family of adversarial attacks on *classifiers*, depending on the choice of constraints[1] imposed on $\boldsymbol{\eta}$. We also cover the general case of multiclass classifiers, which is similar in spirit to Eq. (1). As our experiments on MNIST and CIFAR suggest, we are able to confuse classifiers with a rate that in some cases approaches current benchmarks that require model access.

In summary, in this work we initiate the study of "NTK-derived" adversarial perturbations and make the following **contributions**:

1. We derive expressions for the output of a converged model on adversarial inputs, based on the theory of the Neural Tangent Kernel (Sec. 3).
2. We propose a family of simple attacks that exploit the aforementioned description. These attacks either depend on analytical expressions afforded by the NTK or on finite approximations of the true quantities (*"empirical"* NTK), obtained by *initialized* replicas of the attacked models (Sec. 3).
3. We experimentally verify the efficacy of our approaches on a simple wide single hidden layer model that operates in the NTK regime (Arora et al., 2019a) (Sec. 4.1).
4. Further, we demonstrate that our attack works in practical settings: fully connected and convolutional, undefended, networks, trained in standard ways (cross entropy minimization, stochastic gradient descent, default initialization, etc) are quite susceptible to NTK noise. We show this both in the case where the noise is generated from the NTK of the model itself, even though the assumptions of NTK theory do not necessarily hold (Sec. 4.2), as well as for *transfer attacks* with adversarial perturbations generated by models within the NTK regime (Sec. 4.3).

Another noteworthy aspect of our work is that we derive provably adversarial perturbations for infinitely wide neural neural networks, following the spirit of other works like Arora et al. (2019b) who perform exact computations with these networks. To our knowledge this is the first such result for this theoretical class of models.

2 Background and Related work

Our work bridges two as yet disparate areas: generation of adversarial examples and the description of the dynamics of neural network training in the infinite width limit, with the Neural Tangent Kernel.

Adversarial Examples. Let f be a classifier, $\mathbf{x}$ be an input (e.g. a natural image) and y its true label (e.g. the image class). Then, given that f is an accurate classifier on $\mathbf{x}$, we define $\tilde{\mathbf{x}}$ to be an adversarial example (Szegedy et al., 2014) for f if the following two hold:

1. $d(\mathbf{x}, \tilde{\mathbf{x}})$ is small for some distance function d. Common choices in computer vision applications are the ℓ_p norms.
2. $f(\tilde{\mathbf{x}}) \neq y$. That is, the perturbed input is being misclassified.

There are many approaches on how to generate such examples and they are often classified into two large areas: white-box and black-box attacks[2]. White-box attacks assume perfect knowledge of the

[1] While we focus on ℓ_∞-attacks in this work, we remark that the introduced framework is much more general and the adoption of other similarity measures is equivalent to optimizing Eq. (1) (a linear objective) with the desired constraint on $\boldsymbol{\eta}$.

[2] While we use the word "attack" multiple times, it must be emphasized that the existence of adversarial examples and the search for robust models are not security-only concerns. They align very well with concepts, such as interpretability and human-like perception (Tsipras et al., 2019; Ilyas et al., 2019).

under-attack model and they mainly rely on optimization procedures to maximize the "confusion" of the model (Szegedy et al., 2014; Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2016; Kurakin et al., 2017; Carlini & Wagner, 2017; Madry et al., 2018). The most widely adopted method is called **P**rojected **G**radient **D**escent (Kurakin et al., 2017; Madry et al., 2018) and performs multiple gradient steps on the input space to discover adversarial examples. Formally, if $L(\mathbf{x}, y)$ denotes a loss function, then, in the context of ℓ_∞ adversaries, by setting $\mathbf{x}^0 = \mathbf{x}$ the update formula for PGD is:

$$\mathbf{x}^{t+1} = \Pi_{\mathbf{x}+S}\left(\mathbf{x}^t + \alpha \operatorname{sgn}(\nabla_{\mathbf{x}^t} L(\mathbf{x}^t, y)\right), \quad t = 0, \ldots, N-1, \tag{2}$$

where $\Pi_{\mathbf{x}+S}(\cdot)$ projects the input inside the allowed ℓ_∞ ball, and α is a gradient step - hyperparameter. $\mathbf{x}^N$ is the final adversarial example. Note that L - often cross-entropy - can be distinct from the loss function the model was trained with. Such attacks are the most powerful to date. Black-box attacks, on the other hand, are not allowed to estimate the aforementioned gradient directly, but rather are restricted to querying the model without accessing the parameters. A lot of these attacks proceed by collecting pairs of inputs and predictions from the model, then train another classifier $\hat{f}$ on them, attack $\hat{f}$ with white-box attacks and finally use those generated adversarial samples to attack f (Papernot et al., 2016; 2017; Liu et al., 2017). The empirical success of such strategies relies on the phenomenon of *transferability* of adversarial examples (Szegedy et al., 2014; Papernot et al., 2016; Tramèr et al., 2017; Liu et al., 2017): it has been observed that adversarial inputs designed for successfully fooling one classifier, manage to also fool others. A related line of research abandons the need for creating substitute models, at the cost of allowing multiple curated queries to the model to infer the loss function (Chen et al., 2017; Ilyas et al., 2018). Finally, hard-label black box attacks have been considered in the literature, in which an attacker observes only the prediction of f (Brendel et al., 2018; Cheng et al., 2019). The adversarial attack we propose does not fall into any of these categories, since it does not query the target model at all.

Most attacks were initially developed to fool *undefended* networks, i.e. networks that have not been optimized to be robust. There is a large body of research on how to obtain robust models, with the most prominent being adversarial training (Goodfellow et al., 2015; Shaham et al., 2018; Madry et al., 2018). Since this work studies a new class of highly agnostic attacks on natural networks, in our experiments we do not attempt to break defended networks, leaving this to future work.

Neural Tangent Kernel (NTK). NTK theory plays an increasing role in the quest to theoretically understand deep learning and the elusive concepts of optimization and generalization in over-parametrized networks. It connects deep learning to *Gaussian Processes* and *kernels*. The seminal work of Neal (1996) considered the infinite width limit of neural networks with one hidden layer and showed that at initialization the network is a function drawn from a Gaussian Process with an associated kernel that depends on the activation function. Recently, this limit (and non asymptotic versions of it) has been revisited either for deep random networks where only the last layer is being trained (Lee et al. (2018); de G. Matthews et al. (2018); Novak et al. (2019)) or fully trained networks with gradient descent (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019b). In the latter case, Jacot et al. (2018) identified a particular quantity that governs the dynamics of the training procedure of a neural network with randomly initialized parameters $\boldsymbol{\theta}$ computing an output $f(\mathbf{x}; \boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}^k$, which they called *Neural Tangent Kernel*.

The NTK is defined as the expectation of the inner product between the gradients of the function with respect to the random initialization of its parameters $\boldsymbol{\theta}$ evaluated on all inputs $\mathbf{x}_i, \mathbf{x}_j$, i.e.,

$$h(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i; \boldsymbol{\theta})^T \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_j; \boldsymbol{\theta}). \tag{3}$$

While first studied in fully connected networks, it is possible to define neural tangent kernels for *any* sane architecture (Yang (2020)).

In the infinite width limit[3] (and under appropriate initialization), the training dynamics of the network with infinitesimal learning rate can be described by a differential equation with a constant NTK. Moreover, crucially, in the case of squared loss this gives rise to a closed-form description of the training dynamics with the NTK. For ease of presentation we restrict to the case where the expected network output at initialization is zero; then the expected output of the network at convergence of training becomes (Lee et al., 2018):

$$f^\infty(\tilde{\mathcal{X}}) = \mathbf{H}(\tilde{\mathcal{X}}, \mathcal{X}) \mathbf{H}(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}, \tag{4}$$

[3] for non asymptotic statements see Arora et al. (2019b)

where $\mathcal{X} \in \mathbb{R}^{n \times d}$ is a collection of n d-dimensional training data, $\mathcal{Y} \in \mathbb{R}^{kn}$ the corresponding labels (for example, n concatenated one-hot vectors), $\tilde{\mathcal{X}} \in \mathbb{R}^{n' \times d}$ is a test set and $\mathbf{H} : \mathbb{R}^{n_1 \times d} \times \mathbb{R}^{n_2 \times d} \to \mathbb{R}^{kn_1' \times kn_2}$ is the NTK (for fixed input datasets, $\mathbf{H}$ is a matrix). For some models, this kernel can be computed in closed form *analytically* by computing the expectation over the initializations. It can also be approximated by an *empirical* kernel via averaging over a few initialized instances.

Finally, there has also been work on the intersection of the two areas, though with very distinct emphasis to ours. Gao et al. (2019); Zhang et al. (2020) examines the *convergence* of adversarial *training* in the overparametrized regime. Recent work (Bubeck et al. (2021); Bartlett et al. (2021)) studies the existence of adversarial examples in overparametrized *random* neural networks (yet with width subexponential in the input dimension), exploiting some local linearity of the functions computed by such networks. However, to our knowledge, no prior work to ours has yet connected these two areas to *generate* adversarial examples, or has exploited the parameter-free description the NTK affords to devise attacks that do not require any information on trained model parameters.

3 Adversarial Perturbations on Wide Neural Networks

Here we present the main methods introduced in this work, assuming the asymptotic conditions on the parameters required for Eq. (4) to hold, and showcasing its implications for adversarially perturbed inputs.

3.1 Scalar output

First, we treat the simpler case of networks with scalar output, which allows us to illustrate the various scenarios we examine. We assume the training dataset consists of n d-dimensional samples, $\mathcal{X} \in \mathbb{R}^{n \times d}$, with labels $\mathcal{Y} \in \{-1, +1\}^n$.

Suppose we would like to evaluate a model described by Eq. (4) on *slightly* perturbed variations of the original *training* data. Then, slightly abusing notation, we set, $\tilde{\mathcal{X}} = \mathcal{X} + \boldsymbol{\epsilon}$, that is $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \boldsymbol{\eta}_i$ for all $\mathbf{x}_i \in \mathcal{X}$ for small, but unknown, perturbations $\boldsymbol{\eta}_i$. By taking a first-order Taylor expansion in the perturbation, we can write the ij-th element of $\mathbf{H}(\tilde{\mathcal{X}}, \mathcal{X})$ as follows:

$$h(\tilde{\mathbf{x}}_i, \mathbf{x}_j) = h(\mathbf{x}_i + \boldsymbol{\eta}_i, \mathbf{x}_j) \approx h(\mathbf{x}_i, \mathbf{x}_j) + \nabla_{\mathbf{x}_i} h(\mathbf{x}_i, \mathbf{x}_j)^T \boldsymbol{\eta}_i. \tag{5}$$

For each row $\underbrace{\mathbf{H}_{i,:}(\tilde{\mathcal{X}}, \mathcal{X})}_{\in \mathbb{R}^{1 \times n}}$ we obtain:

$$\mathbf{H}_{i,:}(\tilde{\mathcal{X}}, \mathcal{X})^T = \mathbf{H}_{i,:}(\mathcal{X}, \mathcal{X})^T + \underbrace{\begin{pmatrix} \nabla_{\mathbf{x}_i} h(\mathbf{x}_i, \mathbf{x}_1)^T \\ \nabla_{\mathbf{x}_i} h(\mathbf{x}_i, \mathbf{x}_2)^T \\ \vdots \\ \nabla_{\mathbf{x}_i} h(\mathbf{x}_i, \mathbf{x}_n)^T \end{pmatrix}}_{\mathbf{A}_i \in \mathbb{R}^{n \times d}} \boldsymbol{\eta}_i. \tag{6}$$

Hence, $\mathbf{H}(\tilde{\mathcal{X}}, \mathcal{X})$ can be written as $\mathbf{H}(\mathcal{X}, \mathcal{X}) + \boldsymbol{\Delta}$ for a perturbation matrix $\boldsymbol{\Delta}$, with i-th row $\boldsymbol{\Delta}_{i,:} = \boldsymbol{\eta}_i^T \mathbf{A}_i^T$. Substituting into Eq. (4), we get:

$$f(\tilde{\mathcal{X}}) = (\mathbf{H}(\mathcal{X}, \mathcal{X}) + \boldsymbol{\Delta}) \mathbf{H}(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y} = \mathcal{Y} + \boldsymbol{\Delta} \mathbf{H}(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}. \tag{7}$$

Thus, the output of the model on $\tilde{\mathbf{x}}_i$ is:

$$\begin{aligned} f(\tilde{\mathbf{x}}_i) &= y_i + \boldsymbol{\Delta}_i \mathbf{H}(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y} \\ &= y_i + \boldsymbol{\eta}_i^T \mathbf{A}_i^T \mathbf{H}(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y} =: y_i + \boldsymbol{\eta}_i^T \mathbf{z}_i, \end{aligned} \tag{8}$$

leading to the linear expression advertised in Eq. (1). The adversarial perturbation $\boldsymbol{\eta}_i$ changes the output by $\boldsymbol{\eta}_i^T \mathbf{z_i} = \boldsymbol{\eta}_i^T \mathbf{A}_i^T \mathbf{H}(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}$, an expression which allows us to *compute* the adversarial perturbation to maximally change the output within the desired constraints on $\boldsymbol{\eta_i}$. We call such adversarial perturbations derived from Eq. (8) *NTK-perturbations*.

Regression versus classification: Eq. (4) gives an exact description for regression models with LSE (L_2-loss), while adversarial examples typically are studied for classification models. As is standard,

we view such regression models as classifiers by thresholding the predictions (i.e. taking the sign of the output in the case of binary $\{-1, 1\}$ classification tasks) or by outputting the maximum prediction (in the case of multiclass problems).

Inspecting Eq. (8), maximal "confusion" of the classification model is achieved by aligning $\boldsymbol{\eta}_i$ with $-y_i \mathbf{z_i}$ (directed towards the decision boundary). In case of the commonly used ℓ_∞ restriction, i.e. $\|\boldsymbol{\eta}_i\|_\infty \leq \epsilon$, the optimal adversarial perturbation is given by:

$$\boldsymbol{\eta}_i = -\epsilon y_i \cdot \operatorname{sign}(\mathbf{A}_i^T \mathbf{H}(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}). \tag{9}$$

We remark that the above derivation allows us to exactly compute an optimal adversarial attack for very wide models trained with gradient descent under certain asymptotic conditions necessary for Eq. (4), without any knowledge of its weights. The computation requires an expression for the NTK and its gradient with respect to the training data. For models where an *analytical* expression of the NTK is available, only access to the labeled training data is necessary (see Sec. 4.1). In more complicated models or those that deviate from the assumptions for Eq. (4) one can compute an *empirical* kernel by sampling over kernels at initialization over a few instances and obtain the matrices $\mathbf{A_i}$ with autodifferentiation tools.

Adversarial perturbations of unseen (test) data Eq. (8) has been derived for perturbations of the *training* data. Consider now the case when we evaluate Eq. (4) on perturbations of unseen *test* data, that is on $\tilde{\mathcal{X}} + \boldsymbol{\epsilon}$. Then, Eq. (7) becomes:

$$f(\tilde{\mathcal{X}} + \boldsymbol{\epsilon}) = (\mathbf{H}(\tilde{\mathcal{X}}, \mathcal{X}) + \boldsymbol{\Delta})\mathbf{H}(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y} = f(\tilde{\mathcal{X}}) + \boldsymbol{\Delta}\mathbf{H}(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y}. \tag{10}$$

Again, solely the second term depends on the perturbation, so we proceed by choosing a maximally perturbing direction as before. The only difference lies in the matrix $\boldsymbol{\Delta}$ that now depends on the test set $\tilde{\mathcal{X}}$ (see App. A.1.1). In practice, an adversary can calculate the NTK $\mathbf{H}(\mathcal{X}, \mathcal{X})$ offline (even a small batch of it, as the experiments suggest) and calculate the optimal perturbation on a new test input $\tilde{\mathbf{x}}_i$ by computing the corresponding row of the matrix $\boldsymbol{\Delta}$. Importantly, no information on the test data *labels* is needed. We will use NTK-perturbations derived in this fashion by default when perturbing test data, unless otherwise stated.

No access to training data In cases when the adversary has no access to training data (inputs and labels), they can make the assumption that the test data comes from the same distribution, which implies that the model would have learned a similar mapping had it been "trained" on test data instead. They then can directly compute the perturbations via the techniques above, provided they have access to the test set labels (this also holds for multiclass classifiers of Sec. 3.2). We will refer to this method as *training-agnostic* test method.

Absence of labels We note that all our methods require access to training (or test) labels, but want to stress that they work even in the setting where the adversary has no access to either labels. To evaluate Eq. (4) and derived expressions, the adversary can "estimate" the labels by observing the output of the attacked model and substituting it instead of the required label vector $\mathcal{Y}$. Such oracular access to the trained model is in contrast to the main point of our work and we don't examine this method in our experiments, but want to point out its existence.

3.2 Multiple outputs

We adapt the derivations of Sec. 3.1 to the setting where the output dimension is larger than one in the underlying regression setting, leading to a multiclass classifier. The derivations can be found in App. A.1.2 and lead to the multi-dimensional analogue of the linear Eq. (1) for $f(x) \in \mathbb{R}^k$, $y \in \mathbb{R}^k$:

$$f(\mathbf{x}_i + \boldsymbol{\eta}_i) = \mathbf{y}_i + \begin{pmatrix} \boldsymbol{\eta}_i^T \mathbf{z}_1 \\ \boldsymbol{\eta}_i^T \mathbf{z}_2 \\ \vdots \\ \boldsymbol{\eta}_i^T \mathbf{z}_k \end{pmatrix}. \tag{11}$$

Again, the $z \in \mathbb{R}^d$ can be computed from the NTK and its derivative as well as the training data labels. Exactly analogous considerations as in the binary case allow to adapt this expression to perturbations of the *test* data, or to substitute access to labeled test data instead of training data.

At this point we have a choice of how to adversarially perturb the classifier to achieve the largest effect on the network output. We present the two most obvious methods, with details in App. A.1.2.

Max-of-ℓ_1 perturbation: Similar in spirit to traditional approaches in adversarial attacks (Carlini & Wagner (2017)) we choose $\boldsymbol{\eta_i}$ such as to most efficiently decrease the correct response r^* while maximally increasing one of the false responses $r \neq r^*$. The solution is given by:

$$\boldsymbol{\eta}_i = \epsilon \text{sgn}(\arg \max_{r=1, r \neq r^\star}^{k} \|\mathbf{z}_r - \mathbf{z}_{r^\star}\|_1). \tag{12}$$

Sum-of-Δz perturbation: For one-hot vectors $\mathbf{y}_i$ we could, instead, maximize the cross-entropy between the labels and the new outputs, thus choosing to produce a maximally mixed output. If r^* is the correct label, this yields

$$\boldsymbol{\eta}_i = \epsilon \text{sgn}(\sum_{r \neq r^*}^{n} (\mathbf{z}_r - \mathbf{z}_{r^\star})). \tag{13}$$

To summarize, we have shown how the NTK based approach allows us to approximate the model output on perturbed data to first order and then analytically design the optimal perturbation. Note how this contrasts with common approaches, which expand the loss and then numerically (and often iteratively) optimize the perturbation.

4 EXPERIMENTS

In this section, we present experimental evidence on the efficacy of our method. First, we test the performance of NTK-perturbed data on a simple wide single-layer model, that is known to perform very close to what the theory predicts. We present results on binary classification tasks extracted from MNIST and CIFAR10. We also test our multiclass attack on the full MNIST dataset and demonstrate what effect hyperparameters such as batch size, magnitude and method of attack have on the final accuracy. We further provide benchmark comparisons with PGD attacks.

Then, we shift our attention to models encountered more frequently in practice. We try our method on fully connected neural networks initialized with standard techniques and trained by minimizing the cross entropy loss with stochastic gradient descent. Although we violate many of the NTK assumptions, we find that our attacks still yield strong results. Finally, we provide evidence on the transferability of our method: noise crafted from one-hidden-layer networks substantially fools networks commonly used in practice, such as LeNet 300-100 and LeNet 5. Unless stated otherwise, we set the ℓ_∞-magnitude of the attack ϵ equal to 0.3 for MNIST and $8/255$ for CIFAR, as is common practice. All experimental details, hyperparameters and infrastructure can be found in App. A.2.

In order to benchmark adversarially generated noise, we compare it to random noise of the same magnitude. When we refer to the *random noise baseline*, we mean impulse noise which with equal probability perturbs each pixel of the image by $\pm\epsilon$ (clipped appropriately to remain inside the allowed input domain; such clipping is applied to all implemented attacks).

4.1 RANDOM COMBINATION OF FEATURES MODEL

In the first set of experiments, we test our approach on a simple gradient-trained one-hidden-layer model with an easily computed (analytical) expression for its Neural Tangent Kernel and where experimental evidence confirms that for a sufficiently wide hidden layer the model closely follows theoretical predictions (Arora et al., 2019a). This model of width m computes the following function $f : \mathbb{R}^d \to \mathbb{R}$:

$$f(\mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \max(\mathbf{w}_r^T \mathbf{x}, 0), \tag{14}$$

where $a_r \sim \text{Unif}(\{-1, +1\})$ are *frozen*, and weights $\mathbf{w}_r$ are initialized from an independent Gaussian distribution $\mathcal{N}(\mathbf{0}, \kappa^2 \mathbf{I})$ for appropriately chosen κ. We train this model in a regression fashion, by minimizing the ℓ_2 loss, and turn it into a classifier by assigning +1 to positive outputs and -1 to non-positive ones. We refer to this model as the Random Combination of Features (*RCF*) model.

The neural tangent kernel $\mathbf{H} \in \mathbb{R}^{n \times n}$ for this model is given by (Cho & Saul, 2009; Arora et al., 2019a) (see App. A.3 for its gradient in order to compute the adversarial perturbation):

$$\mathbf{H}_{ij} = h(\mathbf{x}_i, \mathbf{x}_j) = (\frac{1}{2} - \frac{\arccos\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}\right)}{2\pi})\mathbf{x}_i^T \mathbf{x}_j, \ \forall i, j \in [n]. \tag{15}$$

This model generalizes to a function with multiple outputs in the obvious way. To view the regression model with ℓ_2-loss as a k-way classifier, we embed the labels $\in [k]$ as k-dimensional one-hot vectors and output the prediction with largest value.

4.1.1 Binary classification

We evaluate wide models in Eq. (14) on binary classification tasks extracted from MNIST and CIFAR. On MNIST, we train 45 different models, one for every pair of distinct digits. On CIFAR, we use data from the first two classes ("car" vs. "plane"). We train until convergence. All models on MNIST record almost perfect training and testing performance, while the model on CIFAR has 99.35% training accuracy and 90.3% test accuracy. See App. A.2 for details and hyperparameters.

Analytical NTK First, we calculate the optimal adversarial perturbation from the analytical expression of the NTK (Eq. (15)), and its gradient with respect to the first input (Eq. (23) in App. A.3)). Table 1 shows the performance for perturbation of the *training* and the unseen (*test*) data. The perturbed data are misclassified with a rate that approaches 100% on CIFAR, and 85% on MNIST. We verify that our models are robust on the random baseline of same magnitude.

Empirical NTK We now attack the same model with an empirical approximation of the NTK, estimating the analytical expressions in Eqs. (15) and (23) by initializing replicas of the model under attack using Eq. (3). Sampling the kernel at initialization is often a good enough approximation, since for sufficiently wide networks the NTK doesn't deviate much from its initialization (Jacot et al., 2018; Lee et al., 2019). See Table 1 for the results. Restricted by hardware limitations, we estimate the kernels using mini-batches of 256 for MNIST and 1500 for CIFAR. By that, we mean that we attack the model in *batches*: first we calculate the optimal perturbations for the first 256 images using only these images for the kernel expressions $\mathbf{H}(\mathcal{X}, \mathcal{X})$ and $\mathbf{A}_i$ in Eq. (9), then repeat for the next 256 and so on. In the case of training data, we progressively use all the training dataset, but in the case of test set attacks we *solely* use the first 256 images of the training dataset. This is equivalent to using a model that was only trained on this batch of data. The results are similar to the analytical case for MNIST, but not on CIFAR; we find that absence of full data is devastating for the adversary (61% vs 0%), given the higher dimension of this dataset. Note, however, that this mini-batch approach relaxes our initial assumption that the adversary has access to the whole training distribution; even minimal knowledge suffices to yield strong attacks in some cases!

Training Dataset	NTK (empirical)	NTK (analytical)	Random noise
binary CIFAR	60.94	0.03	99.19
binary MNIST - all pairs	21.77 $\pm$ 14.29	14.74 $\pm$ 16.42	88.43 $\pm$ 15.74
Test Dataset	NTK (empirical)	NTK (analytical)	Random noise
binary CIFAR10	46.05	0.45	90.4
binary MNIST - all pairs	17.81 $\pm$ 16.18	15.25 $\pm$ 16.63	88.37 $\pm$ 15.68

Table 1: RCF model accuracy on NTK-perturbed binary MNIST & CIFAR data. We consider attacks that use an *empirical* kernel, the *analytical* expression and the *random baseline*. The batch size for empirical kernels is 1500 for CIFAR and 256 (500) for training (test) MNIST, while for the analytical method we use full-batch (the whole datasets). MNIST accuracy is averaged over the 45 different classification tasks.

NTK versus PGD To compare our results against a strong benchmark, we include comparisons with PGD attacks of the same magnitude across a range of magnitudes (details on the PGD setup can be found in App. A.2). Recall that PGD is a strictly more knowledgeable attack with access to the model parameters. In Fig. 1 (left), we compare the performance of the RCF model against NTK and PGD attacks of varying ℓ_∞ magnitude ϵ on the binary CIFAR test data. It testifies to the strength of our NTK attack, showing that it is nearly comparable to the PGD attack even though it is completely model-parameter agnostic.

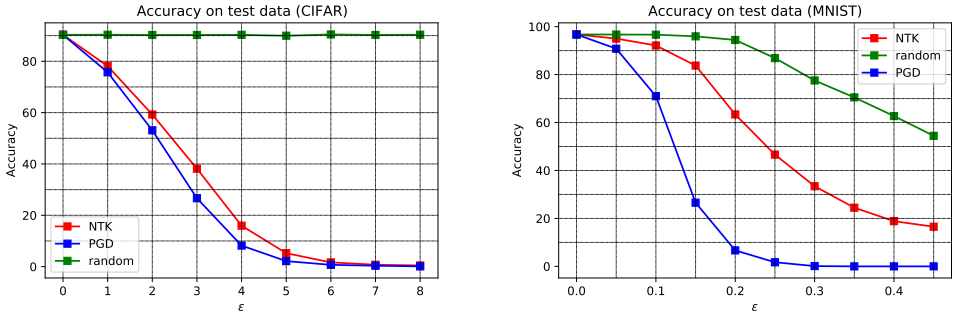


Figure 1: Comparison of our NTK method with PGD attack and random baseline on training data for varying ℓ_∞ magnitude of noise ϵ. Left: Binary setting on CIFAR with analytical kernel. ϵ is given in multiples of $1/255$. Right: Multiclass setting on MNIST with empirical kernel.

4.1.2 Multiclass classification

We now train and attack the multiclass version of the RCF model on MNIST. We implement both *max-of-*ℓ_1 and *sum-of-*Δz perturbations as derived in Section 3.2.

For multiclass attacks we estimate the NTK empirically (since no easily computable analytic expression for the kernel and its derivative was available to us), using different initialized models on each batch. Various results are summarized in Table 2. In a first set of experiments we compare the two attack methods on the training set with batch size 64 for kernel computation (columns 1 and 2). *Sum-of-*Δz is clearly a better choice and we adopt it henceforth. We then show improved results for batch size 256 (column 3) and compare to the random baseline (column 4). These two results are then presented for the test set (columns 5 and 6). It is particularly impressive that accuracy on perturbed training data (data that the model has seen during training) is so similar to accuracy on perturbed test data.

max ℓ_1	sum Δz	sum Δz (batch 256)	random train	sum Δz test	random test
37.57	**28.37**	24.31	77.93	24.81	75.5

Table 2: RCF multiclass attacks with MNIST.

We have also performed the comparison to PGD attacks in the multiclass setting for noise with varying ℓ_∞ magnitude ϵ in Fig. 1 (right), together with the random baseline. We observe that for small perturbations the two attacks start out comparable (recall that our Taylor expansion arguments assume small perturbations), but for larger ϵ the PGD attack begins to dominate.

4.2 Fully Connected Networks

After a strong validation of our proposed method in the regime where the NTK assumptions hold, we now depart to more practical models, trained with standard techniques (stochastic gradient descent, standard initialization of parameters) directly as classifiers (cross-entropy loss rather than ℓ_2 regression). To explore how our attack holds up under these relaxed assumptions, we train a family of fully connected networks of varying depth and width on multiclass MNIST. To compute the model-specific NTK perturbation, we approximate the empirical kernel of Eq. (3) by sampling from the initial parameters and using batches of data. All details can be found in App. A.2.

We present results on *training data* in Table 3. Our NTK-attack is surprisingly successful, despite the fact that the models are outside the asymptotic regime where they can be shown to obey Eq. (4), not to mention that even if the dynamics was close to one described by the NTK there is now no guarantee that the NTK estimated with initialized replicas is close to the actual one. These fully converged networks are fooled to below 25% accuracy in almost all models (of course, model-specific PGD noise drops this further close to 0 for these nets). We observe for deeper models (depth $= 4$ or 6) that with increasing width the gap to the random baseline widens. We speculate that the observed NTK-adversarial weakness is broadly linked to extreme overparametrization.

model	NTK	Random	model	NTK	Random	model	NTK	Random
(2, 100)	17.91	58.67	(4, 100)	21.91	56.45	(6, 100)	33.36	52.73
(2, 1000)	23.37	68.88	(4, 1000)	18.79	55.36	(6, 1000)	27.69	59.29
(2, 10000)	25.55	71.77	(4, 10000)	19.78	64.68	(6, 10000)	22.19	60.09

Table 3: Accuracy of fully connected nets on NTK-perturbed ($\epsilon = 0.3$) multiclass MNIST *training* data. The model tuple denotes (*depth*, *width*).

4.3 Transferability of NTK noise

Finally, we evaluate how well NTK noise derived from one model "generalizes" into fooling other, well-established models. We train a LeNet 300-100, a LeNet 5 and a wide Residual network on (multiclass) MNIST with standard techniques, and then attack them with NTK-noise generated from the RCF model of Sec. 4.1 (see App. A.2 for details). To compare transferability we also test how well PGD noise coming from the *trained* RCF performs on those architectures, and display accuracy of the random baseline to quantify robustness. Results on training and test datasets are presented in Table 4. Surprisingly, we find that adversarial examples crafted from the kernel of the simple RCF model successfully fool models whose behavior is not believed to be captured by the NTK theory.

noise (**train**)	RCF	LeNet 300-100	LeNet 5	WideResNet
RCF	24.36	14.65	39.83	74.47
PGD (ℓ_2)	15.61	15.47	31.98	62.45
PGD (ce)	0.15	0.59	18.51	62.75
Random	77.67	58.3	80.01	95.54
noise (**test**)	RCF	LeNet 300-100	LeNet 5	WideResNet
RCF (train-agnostic)	24.81	14.74	40.54	74.04
RCF	32.25	19.7	48.85	77.38
PGD (ℓ_2)	16.27	16.19	32.86	62.34
PGD (ce)	0.12	0.61	18.76	63.45
Random	75.50	58.04	78.64	93.27

Table 4: Transfer results on MNIST (train & test) with NTK-noise generated from the RCF model with the sum-of-Δz attack. We compare the performance of the NTK noise with that of PGD perturbed data produced by a trained RCF, using the ℓ_2 loss and cross-entropy loss (ce), respectively.

5 Discussion

In this work, we initiate the study of adversarial examples generated by infinitely wide neural networks exploiting their NTK description, and show experimentally that this leads to a set of attacks on *real*, finite networks. These findings are quite unexpected: our strategies successfully fool accurate models that do not necessarily fall into the NTK regime. In addition, we demonstrate that NTK-derived noise carries transfer abilities between different models, similar to what is seen for common adversarial examples. All these findings suggest that the methods we introduce provide us with a remarkable resource of high-quality adversarial examples, without any model access.

The analytical versions of our attacks are efficiently and easily computable, which may provide a new incentive to obtain more analytic expressions for the kernels and their gradients for various architectures, especially in the multi-output case or for convolutional models. While empirical kernels often provide a good approximation, hardware limitations that come with kernel methods pose obstacles to determining the full capabilities of the NTK attacks on large models and datasets, such as (full) CIFAR and ImageNet. It is subject of future work to make advances in this area, for instance by incorporating the gradient computation into standard NTK libraries (Novak et al., 2020).

In recent years a large body of research has focused on how to defend against attacks designed to fool "undefended" models. Here, we do not design an attack to break robust, defended networks. However, our work supplies us with a new set of almost off-the-shelf adversarial examples with similar characteristics to typically produced ones that are model-architecture specific. We hope that their incorporation into training can lead to adversarially robust networks with much less compute.

REPRODUCIBILITY

In keeping with conference guidelines, we have tried to ensure that our experimental results were free of any bias that would misconstrue the results. As specified in the paper, all experiments were run over multiple random seeds, on a variety of benchmarks using publicly available software and hardware. Unfortunately, standard NTK libraries do not provide gradient information of the kernels with respect to the input, which are required to compute the NTK-perturbations when using empirical kernels. We have hence implemented the methods using auto-differentiation tools from Pytorch autograd. The code for the experiments is self contained and references appropriate instructions to install the relevant software dependencies. A copy of the experimental code will be made publicly available.

REFERENCES

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 322–332. PMLR, 09–15 Jun 2019a.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8139–8148, 2019b.

Peter L. Bartlett, Sébastien Bubeck, and Yeshwanth Cherapanamjeri. Adversarial examples in multilayer random relu networks. *CoRR*, abs/2106.12611, 2021.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Sébastien Bubeck, Yeshwanth Cherapanamjeri, Gauthier Gidel, and Rémi Tachet des Combes. A single gradient step finds adversarial examples on random two-layers neural networks. *CoRR*, abs/2104.03863, 2021.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 39–57. IEEE Computer Society, 2017.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Bhavani M. Thuraisingham, Battista Biggio, David Mandell Freeman, Brad Miller, and Arunesh Sinha (eds.), *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, pp. 15–26. ACM, 2017. doi: 10.1145/3128572.3140448. URL https://doi.org/10.1145/3128572.3140448.

Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta (eds.), *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pp. 342–350. Curran Associates, Inc., 2009.

Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *CoRR*, abs/1804.11271, 2018.

Ruiqi Gao, Tianle Cai, Haochuan Li, Cho-Jui Hsieh, Liwei Wang, and Jason D. Lee. Convergence of adversarial training in overparametrized neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13009–13020, 2019.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2142–2151. PMLR, 2018.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 125–136, 2019.

Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8580–8589, 2018.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2574–2582. IEEE Computer Society, 2016.

Radford M. Neal. *Priors for Infinite Networks*, pp. 29–53. Springer New York, New York, NY, 1996.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi (eds.), *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pp. 506–519. ACM, 2017.

Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

Florian Tramèr, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. The space of transferable adversarial examples. *CoRR*, abs/1704.03453, 2017. URL `http://arxiv.org/abs/1704.03453`.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Greg Yang. Tensor programs II: neural tangent kernel for any architecture. *CoRR*, abs/2006.14548, 2020. URL `https://arxiv.org/abs/2006.14548`.

Yi Zhang, Orestis Plevrakis, Simon S. Du, Xingguo Li, Zhao Song, and Sanjeev Arora. Overparameterized adversarial training: An analysis overcoming the curse of dimensionality. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

A Appendix

A.1 Serivation of Adversarial Perturbations

A.1.1 Perturbations of unseen test data

Here give the explicit expression of the matrix Δ in the case of unseen test data:

$$\boldsymbol{\Delta}_{i,:} = \boldsymbol{\eta}_i^T \begin{pmatrix} \nabla_{\tilde{\mathbf{x}}_i} h(\tilde{\mathbf{x}}_i, \mathbf{x}_1)^T \\ \nabla_{\tilde{\mathbf{x}}_i} h(\tilde{\mathbf{x}}_i, \mathbf{x}_2)^T \\ \vdots \\ \nabla_{\tilde{\mathbf{x}}_i} h(\tilde{\mathbf{x}}_i, \mathbf{x}_n)^T \end{pmatrix}^T \tag{16}$$

A.1.2 MULTICLASS DERIVATIONS

Here we derive the expressions for Eq. (11).

While we remain with $\mathcal{X} \in \mathbb{R}^{n \times d}$, the other quantities change as $\mathcal{Y} \in \mathbb{R}^{nk}$, $f(\mathcal{X}) \in \mathbb{R}^{nk}$ and $\mathbf{H}(\mathcal{X}, \mathcal{X}) \in \mathbb{R}^{nk \times nk}$, i.e. for each data pair $(\mathbf{x}_i, \mathbf{x}_j)$ we have $\mathbf{H}(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}^{k \times k}$. Let $\mathbf{H}_{lm}(\mathbf{x}_i, \mathbf{x}_j)$ denote the entry of $\mathbf{H}(\mathbf{x}_i, \mathbf{x}_j)$ that corresponds to the l-th and the m-th output of the model (evaluated at $\mathbf{x}_i$ and $\mathbf{x}_j$). Then, with similar reasoning that led to Eq. (5) we now obtain:

$$\mathbf{H}_{lm}(\mathbf{x}_i + \boldsymbol{\eta}, \mathbf{x}_j) \approx \mathbf{H}_{lm}(\mathbf{x}_i, \mathbf{x}_j) + \nabla_{\mathbf{x}_i} \mathbf{H}_{lm}^T(\mathbf{x}_i, \mathbf{x}_j)\boldsymbol{\eta}. \tag{17}$$

For the prediction of the model on the whole dataset, we have:

$$f(\tilde{\mathbf{X}}) = \mathcal{Y} + \underbrace{\boldsymbol{\Delta}}_{\in \mathbb{R}^{nk \times nk}} (\mathbf{H}(\mathbf{X}, \mathbf{X}))^{-1} \mathcal{Y}, \tag{18}$$

which for a given sample $\mathbf{x}_i$ gives:

$$f(\mathbf{x}_i + \boldsymbol{\eta}_i) = \underbrace{\mathbf{y}_i}_{\in \mathbb{R}^k} + \boldsymbol{\Delta}_{ik:(i+1)k,:} \left(\mathbf{H}(\mathbf{X}, \mathbf{X})\right)^{-1} \mathcal{Y}, \tag{19}$$

where $\boldsymbol{\Delta}_{(i-1)k:ik,:}$ is equal to

$$\underbrace{\begin{pmatrix} \overbrace{\begin{matrix} (\nabla_{\mathbf{x}_i}\mathbf{H}_{11}(\mathbf{x}_i, \mathbf{x}_1) & \dots & \nabla_{\mathbf{x}_i}\mathbf{H}_{1k}(\mathbf{x}_i, \mathbf{x}_1) & \nabla_{\mathbf{x}_i}\mathbf{H}_{11}(\mathbf{x}_i, \mathbf{x}_2) & \dots & \nabla_{\mathbf{x}_i}\mathbf{H}_{1k}(\mathbf{x}_i, \mathbf{x}_n))^T \\ (\nabla_{\mathbf{x}_i}\mathbf{H}_{21}(\mathbf{x}_i, \mathbf{x}_1) & \dots & \nabla_{\mathbf{x}_i}\mathbf{H}_{2k}(\mathbf{x}_i, \mathbf{x}_1) & \nabla_{\mathbf{x}_i}\mathbf{H}_{21}(\mathbf{x}_i, \mathbf{x}_2) & \dots & \nabla_{\mathbf{x}_i}\mathbf{H}_{2k}(\mathbf{x}_i, \mathbf{x}_n))^T \\ & & & \vdots & & \\ (\nabla_{\mathbf{x}_i}\mathbf{H}_{k1}(\mathbf{x}_i, \mathbf{x}_1) & \dots & \nabla_{\mathbf{x}_i}\mathbf{H}_{kk}(\mathbf{x}_i, \mathbf{x}_1) & \nabla_{\mathbf{x}_i}\mathbf{H}_{k1}(\mathbf{x}_i, \mathbf{x}_2) & \dots & \nabla_{\mathbf{x}_i}\mathbf{H}_{kk}(\mathbf{x}_i, \mathbf{x}_n))^T \end{matrix}}^{\in \mathbb{R}^{nk \times d}} \boldsymbol{\eta}_i \end{pmatrix}}_{\in \mathbb{R}^{k \times nk}}. \tag{20}$$

Derivation of Max-of-ℓ_1 perturbation: Similar in spirit to traditional approaches in adversarial attacks (see e.g. Carlini & Wagner (2017)), we solve:

$$\boldsymbol{\eta}_i = \arg \max_{\|\boldsymbol{\eta}_i\|_\infty \leq \epsilon} \max_{r=1, r \neq r^\star}^{k} f_r(\mathbf{x}_i + \boldsymbol{\eta}_i) - f_{r^\star}(\mathbf{x}_i + \boldsymbol{\eta}_i),$$

where $r^\star = \arg\max \mathbf{y}_i$, i.e. we chose $\boldsymbol{\eta_i}$ such as to most efficiently decrease the correct response while maximally increasing one of the false responses. The solution is given by:

$$\begin{aligned} \boldsymbol{\eta}_i &= \arg \max_{\|\boldsymbol{\eta}_i\|_\infty \leq \epsilon} \max_{r=1, r \neq r^\star}^{k} \boldsymbol{\eta}_i^T \mathbf{z}_r - \boldsymbol{\eta}_i^T \mathbf{z}_{r^\star} \\ &= \arg \max_{\|\boldsymbol{\eta}_i\|_\infty \leq \epsilon} \max_{r=1, r \neq r^\star}^{k} \boldsymbol{\eta}_i^T (\mathbf{z}_r - \mathbf{z}_{r^\star}) \\ &= \epsilon \text{sgn}(\arg \max_{r=1, r \neq r^\star}^{k} \|\mathbf{z}_r - \mathbf{z}_{r^\star}\|_1). \end{aligned} \tag{21}$$

Derivation of Sum-of-Δz perturbation: For one-hot vectors $\mathbf{y}_i$ we could, instead, maximize the cross-entropy between the labels and the new outputs, thus choosing to produce a maximally mixed output:

$$\begin{aligned} \mathcal{L}_{ce}(f(\mathbf{x}_i + \boldsymbol{\eta}), \mathbf{y}_i) &= -\sum_{r=1}^{k} y_i^{(r)} \log \left(\frac{e^{y_i^{(r)} + \boldsymbol{\eta}_i^T \mathbf{z}_r}}{\sum_{r'=1}^{k} e^{y_i^{(r')} + \boldsymbol{\eta}_i^T \mathbf{z}'_r}} \right) \\ &= -\log \left(\frac{e^{1 + \boldsymbol{\eta}_i^T \mathbf{z}_{r^\star}}}{\sum_{r=1, r \neq r^\star}^{k} e^{\boldsymbol{\eta}_i^T \mathbf{z}_r} + e^{1 + \boldsymbol{\eta}_i^T \mathbf{z}_{r^\star}}} \right) \\ &= \log \left(\sum_{r \neq r^\star} e^{\boldsymbol{\eta_i}^T (\mathbf{z}_r - \mathbf{z}_{r^\star}) - 1} + 1 \right) \end{aligned} \tag{22}$$

which amount to maximizing

$$\sum_{r \neq r^\star} e^{\boldsymbol{\eta}_i^T (\mathbf{z}_r - \mathbf{z}_{r^\star})}.$$

For small perturbations we can develop the exponential to first order[4], which leads to finding the maximum of

$$\boldsymbol{\eta}_i^T \sum_{r \neq r^\star} (\mathbf{z}_r - \mathbf{z}_{r^\star})$$

This yields $\boldsymbol{\eta}_i$ equal to $\epsilon \text{sgn}(\sum_{r \neq r^*}^{n} (\mathbf{z}_r - \mathbf{z}_{r^\star}))$.

A.2 Experimental Details

For all models, we used GPU enabled hardware to expedite computational time. Our experiments were performed on a SLURM cluster enabled with NVIDIA V100 Tesla and RTX 8000 GPUs, using the PyTorch package.

In all data sets and all experiments, we drop the last batch if its length is smaller than the others.

RCF Models In all experiments, we set the width m of the models equal to 10000 and the variance of the random initialization, κ^2, to 0.01^2. To train the models, we use (full-batch) gradient descent optimizing the quadratic loss with learning rate equal to 0.01. On MNIST, all models are trained for at least 5000 epochs, and then we stop training when training accuracy crosses 99.8%. They record test accuracy between 99.17$\%$ and 99.91$\%$. On CIFAR, we train the model for 200k epochs. For the multiclass experiment on MNIST, we train a model with the same hyperparameters for 200k epochs. It recorded 97.01$\%$ and 96.74$\%$ accuracy on train and test data, respectively. The empirical kernels were approximated using solely one replica of the models *per* batch. Given that defining a network is essentially cost-free (a few lines of code without any/much computational cost), we make the decision to draw a new random seed at each batch in order to alleviate the randomness factor.

Fully Connected Nets We initialize the parameters of the linear layers of the FC networks using the default PyTorch method, i.e. a uniform distribution $\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = \frac{1}{\text{in features}}$. The models are trained with stochastic gradient descent using mini-batches of 256, optimizing the cross entropy loss. All models are trained to convergence (see Table 5 for accuracies). We estimate the NTKs empirically, by defining exact replicas of the models to approximate equation 3. In the smaller nets, we use 3 models initialized with different random seeds for each batch for the approximation, and just one replica per batch on the largest ones (width $= 10000$ and depth equal to 4 or 6). Limited by the heavy computations of the autodifferentiaton tool to estimate the NTK related quantities, we set batch size equal to 64. The ℓ_1-magnitude of the perturbations is set to $\epsilon = 0.3$. Due to limitations in compute we restrict ourselves to produce results on 20% of the training set.

model	Train	Test
(2, 100)	99.02	97.69
(2, 1000)	99.03	97.94
(2, 10000)	99.01	97.83
(4, 100)	99.05	97.36
(4, 1000)	99.06	97.67
(4, 10000)	99.09	97.73
(6, 100)	99.21	96.52
(6, 1000)	99.51	97.1
(6, 10000)	99.34	97.42

Table 5: Train and test accuracy on fully connected nets (MNIST).

[4]The resulting expression for the maximum also holds when developing to second order.

Transferability To generate the NTK-noise we use the sum-of-Δz method using mini-batches of size 256 and $\epsilon = 0.3$. All models were trained with stochastic gradient descent on the ℓ_2 loss with batch size set to 256. Learning rate was set to 0.01. LeNet 300-100 was trained for 500 epochs, recording at the end of training 99.96% train accuracy and 98.02% on test. LeNet 5 was trained for 250 epochs, recording at the end of training 99.82% train and 98.62% test accuracy. ResNet belongs in the family of wide ResNets 50-2, and it was trained for 100 epochs recording at the end of training 99.78% training and 98.29% test accuracy.

PGD attack details PGD attacks as in Eq. (2) optimize the cross-entropy loss for 40 iterations with step size $\alpha = 0.01$ on MNIST and $\alpha = 2/255$ on CIFAR.

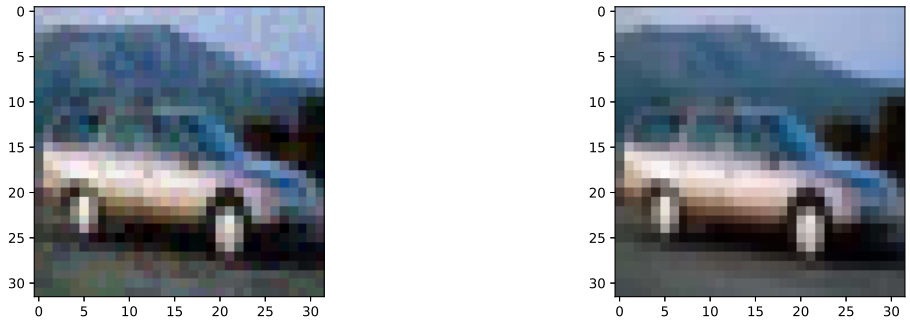
A.3 Analytical Kernel

We give the formula for the gradients of the different elements of the NTK for the RCF model considered in the experiments.

$$
\begin{aligned}
\nabla_{\mathbf{x}_i} h(\mathbf{x}_i, \mathbf{x}_j) &= (\frac{1}{2} - \frac{\arccos\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)}{2\pi})\mathbf{x}_j + \nabla_{\mathbf{x}_i}\left(\frac{1}{2} - \frac{\arccos\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)}{2\pi}\right)\mathbf{x}_i^T \mathbf{x}_j \\
&= (\frac{1}{2} - \frac{\arccos\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)}{2\pi})\mathbf{x}_j + \frac{1}{2\pi\sqrt{1 - (\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|})^2}} \nabla_{\mathbf{x}_i}\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)\mathbf{x}_i^T \mathbf{x}_j \\
&= (\frac{1}{2} - \frac{\arccos\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)}{2\pi})\mathbf{x}_j + \frac{1}{2\pi\sqrt{1 - (\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|})^2}} \frac{\mathbf{x}_j\|\mathbf{x}_i\|\|\mathbf{x}_j\| - \mathbf{x}_i^T \mathbf{x}_j \frac{\mathbf{x}_i\|\mathbf{x}_j\|}{\|\mathbf{x}_i\|}}{\|\mathbf{x}_i\|^2\|\mathbf{x}_j\|^2}\mathbf{x}_i^T \mathbf{x}_j.
\end{aligned}
\tag{23}
$$

Note that one must pay attention when evaluating the diagonal entries (or any entry whose corresponding vectors are aligned), because the $\arccos$ function is not differentiable at 1. To proceed, one has the option to do an analytic expansion over 1, and treat this case separately. In practice, we truncate all inner products between $-1 + \delta$ and $1 - \delta$ for some small $\delta > 0$ (e.g. $\delta = 1e - 5$), since we found this makes the method numerically stable.

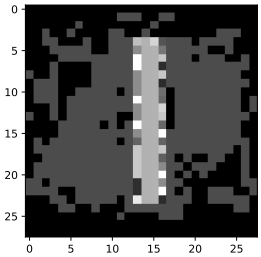
A.4 Adversarial Images



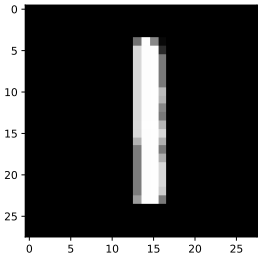
(a) Adversarially perturbed input

(b) Clean input

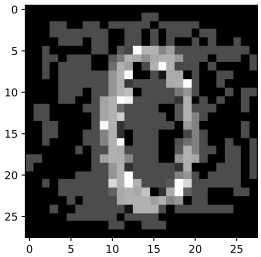
Figure 2: Comparison between adversarial image computed by an infinite neural network and its clean counterpart.



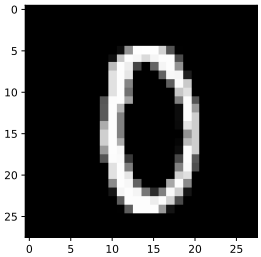
(a) Adversarially perturbed input



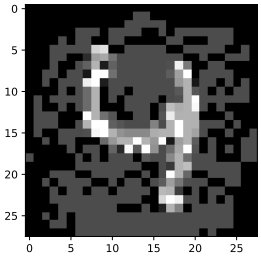
(b) Clean input



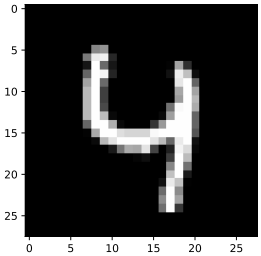
(c) Adversarially perturbed input



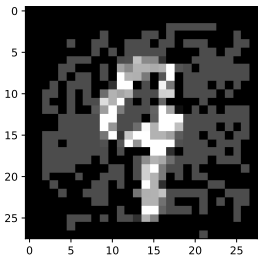
(d) Clean input



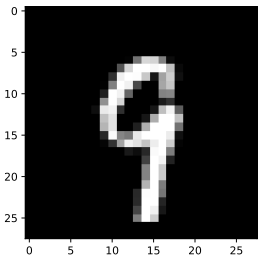
(e) Adversarially perturbed input



(f) Clean input



(g) Adversarially perturbed input



(h) Clean input

Figure 3: MNIST data