What Can the Neural Tangent Kernel Tell Us About Adversarial Robustness?

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Adversarial vulnerability of neural nets, and subsequent techniques to create robust models have attracted significant attention; yet we still lack a full understanding of this phenomenon. Here, we study adversarial examples of trained neural networks through analytical tools afforded by recent theory advances connecting neural networks and kernel methods, namely the Neural Tangent Kernel (NTK), following a growing body of work that leverages the NTK approximation to successfully analyze important deep learning phenomena and design algorithms for new applications. We show how NTKs allow to generate adversarial examples in a "training-free" fashion, and demonstrate that they transfer to fool their finite-width neural net counterparts in the "lazy" regime. We leverage this connection to provide an alternative view on robust and non-robust features, which have been suggested to underlie the adversarial brittleness of neural nets. Specifically, we define and study features induced by the eigendecomposition of the associated kernel to better understand the role of robust and non-robust features, the reliance on both for standard classification and the robustness-accuracy trade-off. We find that such features are surprisingly consistent across architectures, and that robust features tend to correspond to the largest eigenvalues of the model, and thus are learned early during training. Our framework allows us to identify and visualize non-robust yet useful features. Finally, we shed light on the robustness mechanism underlying adversarial training of neural nets used in practice: quantifying the evolution of the associated empirical NTK, we demonstrate that its dynamics falls much earlier into the "lazy" kernel regime and manifests a much stronger form of the well known bias to prioritize learning features within the top eigenspaces of the kernel, compared to standard training.

1 Introduction

Despite the tremendous success of deep neural networks in many computer vision and language modeling tasks, as well as in scientific discoveries, their properties and the reasons for their success are still poorly understood. Focusing on computer vision, a particularly surprising phenomenon evidencing that those machines drift away from how humans perform image recognition is the presence of *adversarial examples*, images that are almost identical to the original ones, yet are misclassified by otherwise accurate models.

Since their discovery [41], a vast amount of work has been devoted to understanding the sources of adversarial examples and explanations include, but are not limited to, the close to linear operating mode of neural nets [21], the curse of dimensionality carried by the input space [21, 39], insufficient model capacity [31, 43] or spurious correlations found in common datasets [23]. In particular, one widespread viewpoint is that adversarial vulnerability is the result of a model's sensitivity to imperceptible yet well-generalizing features in the data, so called *useful non-robust* features, giving

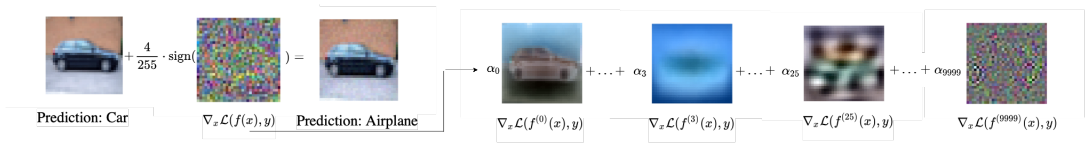

Prediction: Car
$+ \frac{4}{255} \cdot \text{sign}($
$) =$
$\nabla_x \mathcal{L}(f(x), y)$
Prediction: Airplane
α_0
$+ \ldots + \alpha_3$
$+ \ldots + \alpha_{25}$
$+ \ldots + \alpha_{9999}$
$\nabla_x \mathcal{L}(f^{(0)}(x), y)$
$\nabla_x \mathcal{L}(f^{(3)}(x), y)$
$\nabla_x \mathcal{L}(f^{(25)}(x), y)$
$\nabla_x \mathcal{L}(f^{(9999)}(x), y)$

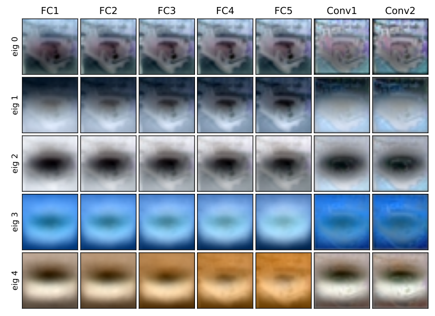
Figure 1: **Left**. Standard setup of an adversarial attack, where a barely perceivable perturbation is added to an image to confuse an accurate classifier. **Right**. The correspondence between neural networks and kernel machines allows to visualize a decomposition of this perturbation, each part attributed to a different feature of the model. The first few features tend to be *robust*.

rise to a trade-off between accuracy and robustness [43, 45]. This gradual understanding has enabled the design of training algorithms, that provide convincing, yet partial, remedies to the problem; the most prominent of them being adversarial training and its many variants [16, 21, 30]. Yet we are far from a mature, unified theory of robustness that is powerful enough to universally guide engineering choices or defense mechanisms.

In this work, we aim to get a deeper understanding of adversarial robustness (or lack thereof) by focusing on the recently established connection of neural networks with kernel machines. Infinitely wide neural networks, trained via gradient descent with infinitesimal learning rate, provably become kernel machines with a data-independent, but architecture dependent kernel - its Neural Tangent Kernel (NTK) - that remains constant during training [4, 24, 27, 28]. The analytical tools afforded by the rich theory of kernels have resulted in progress in understanding the optimization landscape and generalization capabilities of neural networks [3, 17], together with the discovery of interesting deep learning phenomena [18, 34], while also inspiring practical advances in diverse areas of applications such as the design of better classifiers [38], efficient neural architecture search [14], low-dimensional tasks in graphics [42] and dataset distillation [32]. While the NTK approximation is increasingly utlilized, even for finite width neural nets, little is known about the adversarial robustness properties of these infinitely wide models.

Our contribution: Our work inscribes itself into the quest to leverage analytical tools afforded by kernel methods, in particular spectral analysis, to track properties of interest in the associated neural nets, in this case as they pertain to robustness. We study adversarial perturbations and robustness of kernels so as to enrich our understanding of adversarial robustness in general machine learning models. To this end, we first demonstrate that adversarial perturbations generated *analytically* with the NTK can successfully lead the associated trained wide neural networks (in the kernel-regime) to misclassify, thus allowing kernels to faithfully predict the lack of robustness of those trained neural networks. In other words, adversarial (non-) robustness transfers from kernels to networks; and adversarial perturbations generated via kernels resemble those generated by the corresponding trained networks. One implication of this transferability is that we can analytically devise adversarial examples that do not require access to the trained model and in particular its weights; instead these "blind spots" may be calculated a-priori, before training starts. Although similar transferability has already been known and exploited in prior work that trains substitute models, we demonstrate that this notion holds from first principles that only require the a-priori description of the model architecture. The analytical expressions afforded by the kernel might provide a better understanding of the elusive concept of transferability also across architectures, as the corresponding expressions for the associated kernels can be compared directly.

A perhaps even more crucial implication of the NTK approach to robustness relates to the *understanding* of adversarial examples. Indeed, we show how the spectrum of the NTK provides an alternative way to define *features* of the model, to classify them according to their robustness and usefulness and visually inspect them via their contribution to the adversarial perturbation (see Fig. 1). This in turn allows us to verify previously conjectured properties of standard classifiers; dependence on both *robust* and *non-robust* features in the data [43], and tradeoff of accuracy and robustness during training. In particular we observe that features tend to be rather invariable across architectures, and that robust features tend to correspond to the *top* of the eigenspectrum (see Fig. 2), and as such are learned first by the corresponding wide nets [3, 24]. Moreover, we are able to visualize useful non-robust features of standard models (Fig. 5). While this conceptual feature distinction has been highly influential in recent works that study the robustness of deep neural networks [1, 25, 40], to the best of our knowledge, none of them has explicitly demonstrated the dependence of networks on such feature functions (except for simple linear models [20]). Rather, these works either reveal such features in some indirect fashion, or accept their existence as an assumption. Here, we show that Neural Tangent



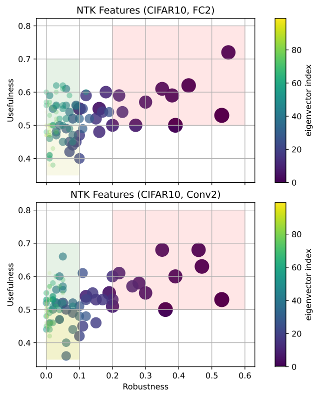
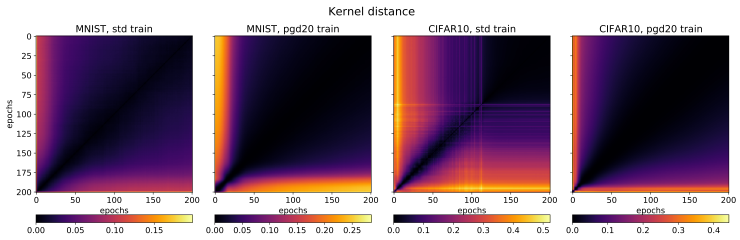


Figure 2: **Left**: Top 5 features for 7 different kernel architectures for a car image extracted from the CIFAR10 dataset when trained on car and plane images. **Right**: Features according to their robustness (x-axis) and usefulness (y-axis). Larger/darker bullets correspond to larger eigenvalues. *Useful* features have > 0.5-usefulness; shaded boxes are meant to help visualize useful-robust regions.



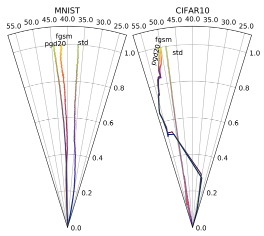


Figure 3: **Left:** Kernel rotation during standard, and adversarial training. Left to right: MNIST, standard, MNIST adversarial, CIFAR standard, CIFAR adversarial. **Right:** Kernel trajectories in polar space for MNIST (left) and CIFAR10 (right). Darker colors indicate earlier epochs.

Kernel theory endows us with a natural definition of features through its eigen-decomposition and provides a way to *visualise and inspect robust and non-robust features directly* on the function space of trained neural networks.

Interestingly, this connection also enables us to empirically demonstrate that robust features of standard models alone are not enough for robust classification. Aiming to understand, then, what makes robust models robust, we track the *evolution* of the data-dependent *empirical* NTK during *adversarial training* of neural networks used in practice. Prior experimental work has found that networks with non-trivial width to depth ratio which are trained with large learning rates, depart from the NTK regime and fall in the so-called "rich feature" regime, where the NTK changes substantially during training [7, 18, 19, 34]. In our work, which to the best of our knowledge is the first to provide insights on how the kernel behaves during adversarial training, we find that the NTK evolves much faster compared to standard training, simultaneously both changing its features and assigning more importance to the more robust ones, giving direct insight into the mechanism at play during adversarial training (see Fig. 3). In summary, the contributions of our work are the following:

- We discuss how to generate adversarial examples for infinitely-wide neural networks via the NTK, and show that they transfer to fool their associated (finite width) nets in the appropriate regime, yielding a "training-free" attack without need to access model weights (Sec. 3).
- We then turn to the kernel machines corresponding to infinitely-wide networks trained with small learning rate to deepen our understanding of adversarial robustness. Using the spectrum of the NTK, we give an alternative definition of features, providing a natural decomposition or perturbations into robust and non-robust parts [23, 43] (Fig. 1). We confirm that robust features overwhelmingly correspond to the top part of the eigenspectrum; hence they are learned early on in training. We bolster previously conjectured hypotheses that prediction relies on both robust and non-robust features and that robustness is traded for accuracy during standard training. Further, we show that only utilizing the robust features of standard models is not sufficient for robust classification (Sec. 4).

- We turn to finite-width neural nets with standard parameters to study the *dynamics* of their empirical NTK during *adversarial training*. We show that the kernel rotates in a way that enables both new (robust) feature learning and that drastically increases of the importance (relative weight) of the robust features over the non-robust ones. We further highlight the structural differences of the kernel change during adversarial training versus standard training and observe that the kernel seems to enter the "lazy" regime much faster (Sec. 5).

Collectively, our findings may help explain many phenomena present in the adversarial ML literature and further elucidate both the vulnerability of standard models and the robustness of adversarially trained ones. We provide code to visualize features induced by kernels, giving a unique and principled way to inspect features induced by standardly trained nets.

Related work: To the best of our knowledge the only prior work that leverages NTK theory to derive perturbations in some adversarial setting is due to [44], yet with entirely different focus. It deals with what is coined *generalization attacks*: the process of altering the training data distribution to prevent models to generalise on clean data. [6] study aspects of robust models through their linearized sub-networks, but do not leverage NTKs. Our work is the first to provide a kernel-induced notion of features and to study robustness in adversarial training in the NTK regime.

2 Preliminaries

We introduce background material and definitions important to our analysis. Here, we restrict ourselves to binary classification and scalar kernels, to keep notation light. We defer the multiclass case, complete definitions and a more detailed discussion of prior work to the Appendix.

2.1 Adversarial Examples

Let f be a classifier, $\mathbf{x}$ be an input (e.g. a natural image) and y its label (e.g. the image class). Then, given that f is an accurate classifier on $\mathbf{x}$, $\tilde{\mathbf{x}}$ is an adversarial example [41] for f if 1) their distance $d(\mathbf{x}, \tilde{\mathbf{x}})$ is small. Common choices in computer vision are the ℓ_p norms, especially the ℓ_∞ norm on which we focus henceforth, and 2) $f(\tilde{\mathbf{x}}) \neq y$. That is, the perturbed input is being misclassified.

Given a loss function $\mathcal{L}$, such as cross-entropy, one can construct an adversarial example $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\eta}$ by finding the perturbation $\boldsymbol{\eta}$ that produces the maximal increase of the loss, solving

$$\boldsymbol{\eta} = \arg \max_{\|\boldsymbol{\eta}\|_\infty \leq \epsilon} \mathcal{L}(f(\mathbf{x} + \boldsymbol{\eta}), y), \tag{1}$$

for some $\epsilon > 0$ that quantifies the dissimilarity between the two examples. In general, this is a non-convex problem and one can resort to first order methods [21]

$$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(f(\mathbf{x}), y)\right), \tag{2}$$

or iterative versions for solving it [26, 30]. The former method is usually called *Fast Gradient Sign Method (FGSM)* and the latter *Projected Gradient Descent (PGD)*. These methods are able to produce examples that are being misclassified by common neural networks with a probability that approaches 1 [12]. Even more surprisingly, it has been observed that adversarial examples crafted to "fool" one machine learning model are consistently capable of "fooling" others [35, 36], a phenomenon that is known as the *transferability* of adversarial examples. Finally, *adversarial training* refers to the alteration of the training procedure to include adversarial samples for teaching the model to be robust [21, 30] and empirically holds as the strongest defense against adversarial examples [30, 45].

2.2 Robust and Non-Robust features

Despite a vast amount of research, the reasons behind the existence of adversarial examples are not perfectly clear. A line of work has argued that a central reason is the presence of robust and non-robust features in the data that standard models learn to rely upon [23, 43]. In particular it is conjectured that reliance on *useful but non-robust* features during training is responsible for the brittleness of neural nets. Here, we slightly adapt the feature definitions of [23][1], and extend them to multi-class problems (see Appendix).

[1]We distinguish useful and robust features based on their accuracy as classifiers, not in terms of correlation with the labels as in [23], allowing a natural extension to the multi-class setting. For robustness, we consider any accuracy bounded away from zero as robust, quantifying that an adversary cannot drive accuracy to zero entirely.

Let $\mathcal{D}$ be the data generating distribution with $x \in \mathcal{X}$ and $y \in \{\pm 1\}$. We define a *feature* as a function $\phi : \mathcal{X} \to \mathbb{R}$ and distinguish how they perform as classifiers. Fix $\rho, \gamma \geq 0$:

1. ρ**-Useful** feature: A feature ϕ is called ρ*-useful* if
$$\mathbb{E}_{x,y\sim\mathcal{D}}\left[\mathbf{I}_{\{\text{sign}[\phi(x)]=y\}}\right] = \rho \tag{3}$$
2. γ**-Robust** feature: A feature ϕ is called γ*-robust* if it remains useful under any perturbation inside a bounded "ball" $\mathcal{B}$, that is if
$$\mathbb{E}_{x,y\sim\mathcal{D}}\left[\inf_{\delta\in\mathcal{B}} \mathbf{I}_{\{\text{sign}[\phi(x+\delta)]=y\}}\right] = \gamma \tag{4}$$

In general, a feature adds predictive value if it gives an advantage above guessing the most likely label, i.e. $\rho > \max_{y'\in\{\pm1\}} \mathbb{E}_{x,y\sim\mathcal{D}}[\mathbf{I}_{\{y'=y\}}]$, and we will speak of "useful" features in this case, omitting the ρ. We will call such a feature **useful, non-robust** if it is useful, but γ-robust only for $\gamma = 0$ or very close to 0, depending on context.

The vast majority of works imagines features as being induced by the *activations* of neurons in the net, most commonly those of the penultimate layer (*representation-layer* features), but this formal definition is in no way restricted to activations, and we will show how to exploit it using the eigenspectrum of the NTK. In particular, in Sec. 4, we demonstrate that the above framework agrees perfectly with features induced by the eigenspectrum of the NTK of a network, providing a natural way to decompose the predictions of the NTK into such feature functions. In particular we can identify robust, useful, and, indeed, useful non-robust features.

2.3 Neural Tangent Kernel

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a (scalar) neural network with a linear final layer parameterized by a set of weights $\mathbf{w} \in \mathbb{R}^p$ and $\{\mathcal{X}, \mathcal{Y}\}$ be a dataset of size n, with $\mathcal{X} \in \mathbb{R}^{n\times d}$ and $\mathcal{Y} \in \{\pm1\}^n$. Linearized training methods study the first order approximation
$$f(\mathbf{x}; \mathbf{w}_{t+1}) = f(\mathbf{x}; \mathbf{w}_t) + \nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}_t)^\top (\mathbf{w}_{t+1} - \mathbf{w}_t). \tag{5}$$
The network gradient $\nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}_t)$ induces a kernel function $\Theta_t : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, usually referred as the *Neural Tangent Kernel (NTK)* of the model
$$\Theta_t(\mathbf{x}, \mathbf{x}') = \nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}_t)^\top \nabla_{\mathbf{w}} f(\mathbf{x}'; \mathbf{w}_t). \tag{6}$$
This kernel describes the dynamics with infinitesimal learning rate (gradient flow). In general, the tangent space spanned by the $\nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}_t)$ twists substantially during training, and learning with the Gram matrix of Eq. (6) (empirical NTK) corresponds to training along an intermediate tangent plane. Remarkably, however, in the infinite width limit with appropriate initialization and low learning rate, it has been shown that f becomes a *linear* function of the parameters [24, 28], and the NTK remains *constant* ($\Theta_t = \Theta_0 =: \Theta$). Then, for learning with ℓ_2 loss the training dynamics of infinitely wide networks admits a closed form solution corresponding to kernel regression [4, 24, 27]
$$f_t(\mathbf{x}) = \Theta(\mathbf{x}, \mathcal{X})^\top \Theta^{-1}(\mathcal{X}, \mathcal{X})(I - e^{-\lambda\Theta(\mathcal{X},\mathcal{X})t})\mathcal{Y}, \tag{7}$$
where $\mathbf{x} \in \mathbb{R}^d$ is any input (training or testing), t denotes the time evolution of gradient descent, λ is the (small) learning rate and, slightly abusing notation, $\Theta(\mathcal{X}, \mathcal{X}) \in \mathbb{R}^{n\times n}$ denotes the matrix containing the pairwise training values of the NTK, $\Theta(\mathcal{X}, \mathcal{X})_{ij} = \Theta(\mathbf{x}_i, \mathbf{x}_j)$, and similarly for $\Theta(\mathbf{x}, \mathcal{X}) \in \mathbb{R}^n$. To be precise, Eq. (7) gives the *mean* output of the network using a weight-independent kernel with variance depending on the initialization[2].

3 Transfer results in the kernel regime

In this section, we show how to generate adversarial examples from NTKs and discuss their similarity to the ones generated by the actual networks. Note that for network results, we restrict ourselves to wide networks initialized in the "lazy" regime with small learning rates (the "kernel regime").

[2]For that reason, in the experiments, we often compare this with the centered prediction of the actual neural network, $f - f_0$, as is commonly done in similar studies [15].

3.1 Generation of Adversarial Examples for Infinitely Wide Neural Networks

Adversarial examples arise in the context of *classification*, while the NTK learning process is described by a regression as in Eq. (7). The arguably simplest way to align with the framework presented in Eq. (1) is to treat the outputs of the kernel similar to logits of a neural net, mapping them to a probability distribution via the sigmoid/softmax function and apply cross-entropy loss.

A simple calculation (see Appendix, together with the generalization to the multi-class case) gives:

The optimal one step adversarial example of a scalar, infinitely wide, neural network is given by

$$\tilde{\mathbf{x}} = \mathbf{x} - y\epsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} f_t(\mathbf{x})\right), \tag{8}$$

for $\|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty \leq \varepsilon$, where $\nabla_{\mathbf{x}} f_t(\mathbf{x}) = \nabla_{\mathbf{x}} \Theta(\mathbf{x}, \mathcal{X})^\top \Theta^{-1}(\mathcal{X}, \mathcal{X})(I - e^{-\lambda\Theta(\mathcal{X},\mathcal{X})t})\mathcal{Y}$.

One can conceive other ways to generate adversarial perturbations for the kernel, either by changing the loss function (as previously done in neural networks (e.g. [12])) or through a Taylor expansion around the test input, and we present such alternative derivations in the Appendix. However, in practice we observe little difference between that approach and the one presented here.

3.2 Transfer results and kernel attacks

Predictions from NTK theory for infinitely wide neural networks have been used successfully for their large finite width counterparts, so it seems reasonable to conjecture that adversarial perturbations generated via the kernel as in Eq. (8) strongly resemble those directly computed for the corresponding neural net as per Eq. (2). In particular, this would imply that adversarial perturbations derived from the NTK should not only fool the kernel machine itself, but also lead wide neural nets to misclassify. While similar transfer results in different contexts have been observed indirectly, via the *effects* of the perturbation on metrics like accuracy [32, 44], we aim to look deeper to compare perturbations *directly*. High similarity would imply that *any* gradient based white-box attack on the neural net can be successfully mimicked by a "black-box" kernel derived attack.

Setting. To this end, we train multiple two-layer neural networks on image classifications tasks extracted from MNIST and CIFAR-10 and compare adversarial examples generated by Eqs. (2) (attacking the neural network) and (8) (attacking the kernel). The networks are trained with small learning rate and are sufficiently large, so lie close to the NTK regime. We track cosine similarity between the gradients of the loss from the NTK predictions and the gradients from the actual neural net as training evolves. Then, we generate adversarial perturbations from both the neural net and the kernel machine, and test whether those produced by the latter can fool the former. Full experimental details can be found in the Appendix.

Results. Our experiments confirm a very strong alignment of loss gradients from the neural nets and the NTK across the whole duration of training, as can be seen in Fig. 4 (top). Then, as expected, kernel-generated attacks produce a similar drop in accuracy throughout training as the networks "own" white-box attacks, eventually driving robust accuracy to 0%, as seen in Fig. 4 (bottom). We reproduce these plots for MNIST in the Appendix, leading to similar conclusions.

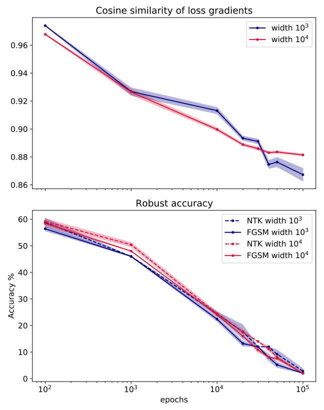


Figure 4: **Top**. Cosine similarity between the loss gradient of the neural net and of the NTK prediction for the same time point. **Bottom**. Robust accuracy of neural net against its own adversarial examples (solid) and corresponding NTK examples (dashed). CIFAR10, car vs plane.

When concerned with security aspects of neural nets, adversarial attacks are mainly characterised as either *white-box* or *black-box* attacks [36]. White box attacks assume full access to the neural network and in particular its weights; prominent examples include FGSM/PGD attacks. Black box attacks, on the other hand, can only *query* the model to try to infer the loss gradient, either through training separate surrogate models [35] or through carefully crafted input-output pairs fed to the target model [2, 13, 22]. NTK theory and the experiments of this section suggest a threat model in which the attacker does not require access to the model or its weights, nor training of a substitute model. For fixed architecture and training data, all the information required for the computation of (8) is available at initialization, making the "NTK-attack" akin to a "training free" substitution attack, and, at least in the kernel-regime for wide nets considered here, as effective as white-box attacks.

4 NTK eigenvectors induce robust and non-robust features

This close connection between adversarial perturbations from the kernel and the corresponding neural net gives us the opportunity to bring to bear kernel tools on the study of adversarial robustness and its relation to features in a more direct fashion. Several recent works leverage properties of the NTK, and specifically its spectrum, to study aspects of approximation and generalization in neural networks [3, 8, 9, 10]. Here we show how the spectrum relates to robustness and helps to clarify the notion of robust/non-robust features.

We define *features* induced by the eigendecomposition of the Gram matrix $\Theta(\mathcal{X},\mathcal{X}) = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$. We will be most interested in the *end* of training, when the model has access to all the features it can extract from the training data $\mathcal{X}$. As $t \to \infty$, Eq. (7) becomes $f_\infty(\mathbf{x}) = \Theta(\mathbf{x},\mathcal{X})^\top \Theta(\mathcal{X},\mathcal{X})^{-1}\mathcal{Y}$ and can be decomposed as $f_\infty(\mathbf{x}) = \Theta(\mathbf{x},\mathcal{X})^\top \sum_{i=1}^{n} \lambda_i^{-1} \mathbf{v}_i \mathbf{v}_i^\top \mathcal{Y} = \sum_{i=1}^{n} f^{(i)}(\mathbf{x})$, where

$$f^{(i)} : \mathbb{R}^d \to \mathbb{R}^k,\ f^{(i)}(\mathbf{x}) := \lambda_i^{-1} \Theta(\mathbf{x},\mathcal{X})^\top \mathbf{v}_i \mathbf{v}_i^\top \mathcal{Y}. \tag{9}$$

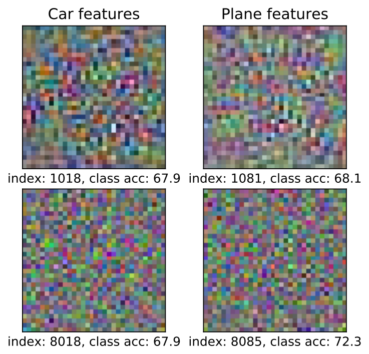
Each $f^{(i)}$ can be seen as a *unique feature* captured from the (training) data. Note that these functions map the input to the output space, thus matching the definitions of Sec. 2.2. Also observe that all $f^{(i)}$'s jointly recover the original prediction of the model, while each one, intuitively, should contribute something different to it.

Importantly, these features induce a decomposition of the gradient of the loss into parts, each representing gradients of a unique feature as already advertised in Fig. 1. The binary case is particularly elegant as it gives rise to a linear decomposition of the gradient as

$$\nabla_{\mathbf{x}} \mathcal{L}(f_\infty(\mathbf{x}), y) = \sum_{i=1}^{n} \alpha_i \nabla_{\mathbf{x}} \mathcal{L}(f^{(i)}(\mathbf{x}), y), \tag{10}$$

for some α_i depending on $\mathbf{x}$ and y (see Appendix). But if $f^{(i)}$'s are features, how do they look like?

Feature properties of common architectures: With these definitions in place, we can now analyze the characteristics of features for commonly used architectures, leveraging their associated NTK. To be consistent with the previous section, we consider classification problems from MNIST (10 classes) and CIFAR-10 (car vs airplane). We compose the Gram matrices from the whole training dataset (50000 and 10000, respectively), and compute the different feature functions $f^{(i)}$ using the eigendecomposition of the matrix. We estimate the **usefulness** of a feature $f^{(i)}$ by measuring its accuracy on a hold-out validation set, and its **robustness** by perturbing each input of this set, using an FGSM attack on feature $f^{(i)}$. We consider several different Fully Connected and Convolutional Kernels, whose expressions are available through the Neural Tangents library [33], built on top of JAX [11]. We summarize our findings on how these features behave:


Car features
Plane features
index: 1018, class acc: 67.9
index: 1081, class acc: 68.1
index: 8018, class acc: 67.9
index: 8085, class acc: 72.3
Figure 5: Non-robust, useful features earlier and later in the spectrum, for CIFAR10 car and plane.

Functions $f^{(i)}$ represent visually distinct features. We visualise each feature $f^{(i)}$ by plotting its gradient with respect to $\mathbf{x}$. Fig. 2 shows the gradient of the first 5 features for various architectures for a specific image from the CIFAR-10 dataset. We observe that features are fairly consistent across models, and they are interpretable: for example the 4th feature seems to represent the dominant color of an image, while the 5th one seems to be capturing horizontal edges.

Networks use both robust and non-robust features for prediction. It has been speculated that neural networks trained in a standard (non adversarial) fashion rely on both robust and non-robust features. Our feature definition in Eq. (9) shows that this is indeed the case. The NTK of common neural networks consists of both robust features that match human expectations, such as the ones depicted in Fig. 2, but also on features that are predictive of the true label, while not being robust to adversarial perturbations of the input (Fig. 5). Fig. 2 depicts the first 100 features of a fully connected and a convolutional tangent kernel in Usefulness-Robustness space. The upper left region of the plots shows a large amount of useful, yet non-robust features. These features seem random to human observers.

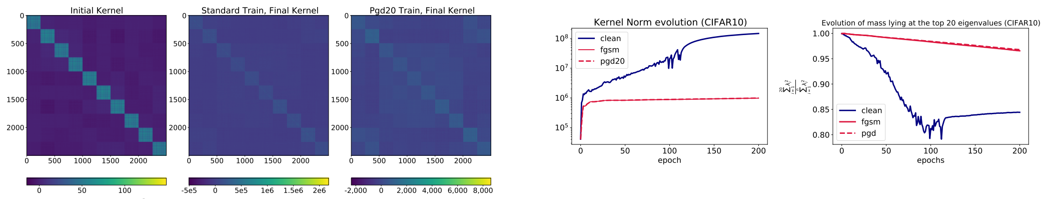


Figure 6: **Left**: Kernel Matrices for a mini batch of size 256. Left to Right: Kernel at initialization, Kernel after standard training, Kernel after adversarial training (20 pgd steps). The standard kernel grows significantly more than the adversarial one. **Right**: (a) Kernel Frobenius norm evolution, and (b) concentration on the top 20 eigenvalues during standard and adversarial training. Setting: CIFAR10, $\ell_\infty = 8/255$.

Robustness lies at the top. We observe in Fig. 2 that features corresponding to the top eigenvectors tend to be robust. This is consistent among different models and between the two datasets (see Appendix). Since these eigenvectors are the ones fitted first during training [3, 24], it is no wonder that the loss gradient evolves from coherence to noise, as observed in Fig. A1. This also explains the apparent trade-off between robustness and accuracy of neural networks as training progresses: useful, robust features are fitted first, followed by useful, but non-robust ones. This ties in well with both empirical findings [37] and theoretical case studies [8, 9, 10] that demonstrate that low frequency *functions* are fitted first during training and provide favorable generalization properties and we would associate robust features with these low-frequency parts.

Robust features alone are not enough. In light of these findings, it might be reasonable to conjecture that we could obtain robust models by retaining the robust features of the prediction, while discarding the non-robust ones. The spectral approach gives a principled way to disentangle features and create kernel machines keeping only the robust ones. Our results show that in general it is not possible to obtain non-trivial performance without compromising robustness in this fashion, strengthening the case for the necessity of data augmentation in the form of adversarial training (see the Appendix for an in-depth study).

5 Kernel dynamics during adversarial training

Given the apparent necessity for adversarial training to produce robust models, how does it achieve this goal? To shed some light on this fundamental question, we depart from the "lazy" NTK regime and study the evolution of the NTK of adversarially trained models. For a neural network trained with gradient descent, as the learning rate $\eta \to 0$, the continuous time dynamics can be written as

$$\frac{\partial w}{\partial t} = -\eta \nabla_w \mathcal{L} = -\eta \nabla_w f^\top \frac{\partial \mathcal{L}}{\partial f} \quad \text{and} \quad \frac{\partial f}{\partial t} = -\eta \underbrace{\nabla_w f \nabla_w f^\top}_{\Theta_t} \frac{\partial \mathcal{L}}{\partial f}. \tag{11}$$

In the NTK regime, this kernel Θ_t remains fixed at its initial value. However, outside this regime, it has been demonstrated, both empirically [7, 18, 19, 34] and theoretically [5], that Θ_t is not constant during training, and is changing as the weights move. In adversarial training, moreover, there is the additional effect that at each weight update, the data changes as well. For that reason, understanding the dynamics of adversarial training requires tracking the evolution of a kernel $\Theta_t(\mathcal{X}_t, \mathcal{X}_t)$, where $\mathcal{X}_t$ denotes the current (mini) batch of training data. Notice that the tangent vector $\nabla_w f(\mathcal{X}_t)$ is still describing the instantaneous change of f on the current batch of data, thus $\Theta_t(\mathcal{X}_t, \mathcal{X}_t)$ is informative of the local geometry of the function space, justifying its value as a quantity to be measured during adversarial training.

We train a deep convolutional architecture on CIFAR-10 (multiclass) with standard (sgd) and adversarial training using PGD with an ℓ_∞ constraint. Full implementations details and accuracy curves can be found in the Appendix, together with the reproduction of the same experiment on MNIST, where the observations are similar. We track the following quantities during training:

Kernel distance. We compare two kernels using a *scale invariant distance*, which quantifies the relative rotation between them, as used in other works studying NTK dynamics (e.g. [18]):

$$d(\Theta_i, \Theta_j) = 1 - \frac{\mathrm{Tr}(\Theta_i \Theta_j^\top)}{\sqrt{\mathrm{Tr}(\Theta_i \Theta_i^\top)}\sqrt{\mathrm{Tr}(\Theta_j \Theta_j^\top)}}.$$

Polar dynamics. Zooming in on the change that the initial kernel undergoes, we define a *polar space* on which we measure the movement of the kernel:

$$r_t = \frac{\|\Theta_t - \Theta_0\|_F}{\|\Theta_f - \Theta_0\|_F}, \quad \theta_t = \arccos\left(1 - d(\Theta_t, \Theta_0)\right), \tag{12}$$

where Θ_0, Θ_f are the initial and final kernel, respectively. Fig. 3 presents a heatmap of kernel distances at different time steps for both standard and adversarial training, as well as both training trajectories in polar space.

Concentration on subspaces. To quantify weight concentration on the top region of the spectrum, we track the (normalized) Frobenius norm of subspaces as $\sum_{i=1}^{p} \lambda_i^2 / \sum_{i=1}^{n} \lambda_i^2$, for various cut-offs p, where we have indexed the eigenvalues from largest to smallest. Fig. 6 depicts concentration on the top 20 eigenvalues during training.

Our findings show that similar to what has been reported in prior work [18], the kernel rotates significantly in the beginning of training and then slows down for both standard and adversarial training. However, in the latter case, this second phase begins a lot earlier. As Fig. 3 illuminates, the kernel moves a greater distance than when performing standard training, but after a few epochs it stops both rotating and expanding; note that this is not the case for standard training where the kernel increases its magnitude substantially later in training, and in fact grows to have a norm orders of magnitude larger than during adversarial training (see Fig. 6). In hindsight, this behavior is perhaps not surprising, as each element of the kernel measures similarity between data points, and a robust machine should be more conservative when estimating similarity. The observation that during adversarial training the kernel becomes relatively static relatively fast might indicate that *linear* dynamics govern the later phase of adversarial training. It has been observed in previous works [18, 19, 34] that linearization after a few initial epochs of rapid rotation often closely matches performance of full network training. Our results indicate that a similar phenomenon occurs even under the data shift of adversarial training, opening avenues to design robust machines more efficiently.

Moreover, endowed with the knowledge that at least for kernels trained with static data robust features lie at the top, we study polar dynamics of the top space only (see Appendix) to observe that there is substantial rotation in this space, suggesting that robust features are learned early on not only during standard, but in particular during adversarial training. Even more interestingly, Fig. 6 demonstrates that not only the robust features change, but their relative weight as measured by the concentration on the top-20 space is increasing simultaneously relative to standard training as well, and remains large; in fact, significantly larger than during standard training. As each eigenvalue weights the importance of the corresponding feature on the final prediction, this implies that the kernel "learns" to depend more on the most robust features.

Put together, these findings reveal different kernel dynamics during standard and adversarial training: the kernel rotates much faster, expands much less and becomes "lazy" much earlier than during standard training. Fully understanding the properties of converged adversarial kernels remains an important avenue for future work, that might allow to design faster algorithms for robust classification.

6 Final Remarks

We have studied adversarial robustness through the lens of the NTK across multiple architectures and data sets both in the idealized NTK regime and the "rich feature" regime. When connecting the spectrum of the kernel with fundamental properties characterizing robustness our phenomenological study reveals a universal picture of the emergence of robust and non-robust features and their role during training. There are certain limitations and unexplored themes in our work; Sec. 3 argues that transferable attacks from the NTK may be as effective as white-box attacks, but this warrants an in-depth study across architectures, kernels and data sets (which has not been the main focus of this work). Sec. 4 visualises features for fairly simple models, since the computation of kernel derivatives is a costly procedure. It would be interesting to use our framework to visualise features from more complicated architectures. Finally, our work in Sec. 5 invites more research on the kernel at the end of adversarial training, similar to what has been done for standard models [29].

We hope that our viewpoint can motivate further theoretical understanding of adversarial phenomena (such as transferability) and the design of better and/or faster adversarial learning algorithms, by further analyzing the kernels from robust deep neural networks.

References

[1] Z. Allen-Zhu and Y. Li. Feature purification: How adversarial training performs robust deep learning. *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 977–988, 2022.

[2] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: A query-efficient black-box adversarial attack via random search. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*, volume 12368 of *Lecture Notes in Computer Science*, pages 484–501. Springer, 2020.

[3] S. Arora, S. Du, W. Hu, Z. Li, and R. Wang. Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 322–332. PMLR, 09–15 Jun 2019.

[4] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8139–8148, 2019.

[5] A. Atanasov, B. Bordelon, and C. Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022.

[6] Y. Bai, X. Yan, Y. Jiang, S. Xia, and Y. Wang. Clustering effect of adversarial robust models. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29590–29601, 2021.

[7] A. Baratin, T. George, C. Laurent, R. D. Hjelm, G. Lajoie, P. Vincent, and S. Lacoste-Julien. Implicit regularization via neural feature alignment. In A. Banerjee and K. Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 2269–2277. PMLR, 2021.

[8] R. Basri, D. W. Jacobs, Y. Kasten, and S. Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4763–4772, 2019.

[9] R. Basri, M. Galun, A. Geifman, D. W. Jacobs, Y. Kasten, and S. Kritchman. Frequency bias in neural networks for input of non-uniform density. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 685–694. PMLR, 2020.

[10] A. Bietti and J. Mairal. On the inductive bias of neural tangent kernels. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12873–12884, 2019.

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

[12] N. Carlini and D. 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*, pages 39–57. IEEE Computer Society, 2017.

[13] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh. ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In B. M. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller, and A. Sinha, editors, *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, pages 15–26. ACM, 2017.

[14] W. Chen, X. Gong, and Z. Wang. Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[15] L. Chizat, E. Oyallon, and F. R. Bach. On lazy training in differentiable programming. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2933–2943, 2019.

[16] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. Robustbench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[17] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 2019.

[18] S. Fort, G. K. Dziugaite, M. Paul, S. Kharaghani, D. M. R. 0001, and S. Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[19] M. Geiger, S. Spigler, A. Jacot, and M. Wyart. Disentangling feature and lazy training in deep neural networks. *Journal Of Statistical Mechanics-Theory And Experiment*, 2020(11):113301, 2020.

[20] G. Goh. A discussion of 'adversarial examples are not bugs, they are features': Two examples of useful, non-robust features. *Distill*, 2019. https://distill.pub/2019/advex-bugs-discussion/response-3.

[21] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[22] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In J. G. Dy and A. Krause, editors, *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*, pages 2142–2151. PMLR, 2018.

[23] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 125–136, 2019.

[24] A. Jacot, C. Hongler, and F. Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8580–8589, 2018.

[25] J. Kim, B.-K. Lee, and Y. M. Ro. Distilling robust and non-robust features in adversarial examples by information bottleneck. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[26] A. Kurakin, I. J. Goodfellow, and S. 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.

[27] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. 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, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[28] C. Liu, L. Zhu, and M. Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15954–15964. Curran Associates, Inc., 2020.

[29] P. M. Long. Properties of the after kernel. *CoRR*, abs/2105.10585, 2021.

[30] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, 2018.

[31] P. Nakkiran. Adversarial robustness may be at odds with simplicity. *CoRR*, abs/1901.00532, 2019. URL `http://arxiv.org/abs/1901.00532`.

[32] T. Nguyen, R. Novak, L. Xiao, and J. Lee. Dataset distillation with infinitely wide convolutional networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[33] R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. 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.

[34] G. Ortiz-Jiménez, S. Moosavi-Dezfooli, and P. Frossard. What can linearized neural networks actually say about generalization? *CoRR*, abs/2106.06770, 2021.

[35] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

[36] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In R. Karri, O. Sinanoglu, A. Sadeghi, and X. Yi, editors, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 506–519. ACM, 2017.

[37] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. C. Courville. On the spectral bias of neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 2019.

[38] V. Shankar, A. Fang, W. Guo, S. Fridovich-Keil, J. Ragan-Kelley, L. Schmidt, and B. Recht. Neural kernels without tangents. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8614–8623. PMLR, 13–18 Jul 2020.

[39] C. Simon-Gabriel, Y. Ollivier, L. Bottou, B. Schölkopf, and D. Lopez-Paz. First-order adversarial vulnerability of neural networks and input dimension. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5809–5817. PMLR, 2019.

[40] J. M. Springer, M. Mitchell, and G. T. Kenyon. A little robustness goes a long way: Leveraging robust features for targeted transfer attacks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[41] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[42] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[43] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. 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.

[44] C.-H. Yuan and S.-H. Wu. Neural tangent generalization attacks. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12230–12240. PMLR, 18–24 Jul 2021.

[45] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 4 and Appendix.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Our work sheds light properties of adversarial examples to make mahcine learning models more reliable in the long run.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]