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

Nikolaos Tsilivis
Center for Data Science
New York University
nt2231@nyu.edu

Julia Kempe
Center for Data Science and
Courant Institute of Mathematical Sciences
New York University
kempe@nyu.edu

Abstract

The 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 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" 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 (Szegedy et al., 2014), 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 (Goodfellow et al., 2015), the curse of dimensionality carried by the input space (Goodfellow et al., 2015, Simon-Gabriel et al., 2019), insufficient model capacity (Tsipras et al., 2019, Nakkiran, 2019) or spurious correlations found in common datasets (Ilyas et al., 2019).

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

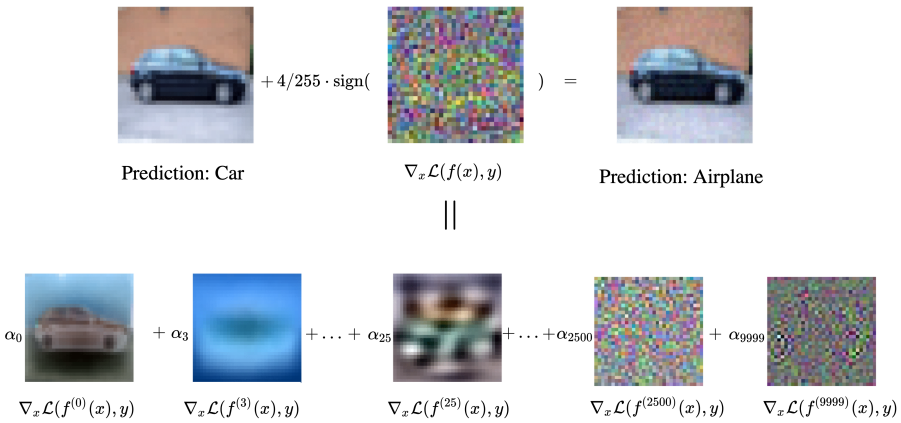


Figure 1: **Top**. Standard setup of an adversarial attack, where a barely perceivable perturbation is added to an image to confuse an accurate classifier. **Bottom**. The correspondence between neural networks and kernel machines allows to visualize a decomposition of this perturbation, each part attributed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

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 rise to a trade-off between accuracy and robustness (Tsipras et al., 2019, Zhang et al., 2019). 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 (Goodfellow et al., 2015, Madry et al., 2018, Croce et al., 2021). 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 (Jacot et al., 2018, Lee et al., 2019, Arora et al., 2019b, Liu et al., 2020). 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 (Du et al., 2019a, Arora et al., 2019a), together with the discovery of interesting deep learning phenomena (Fort et al., 2020, Ortiz-Jiménez et al., 2021), while also inspiring practical advances in diverse areas of applications such as the design of better classifiers (Shankar et al., 2020), efficient neural architecture search (Chen et al., 2021), low-dimensional tasks in graphics (Tancik et al., 2020) and dataset distillation (Nguyen et al., 2021). 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. 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.

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 for

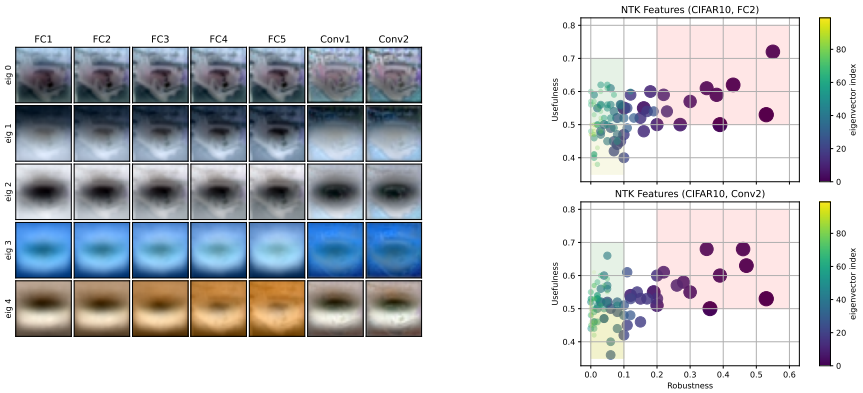


Figure 2: **Left**: Top 5 features for 7 different kernel architectures for a car image extracted from the CIFAR10 dataset when trained on car and plane images. **Right**: Features according to their robustness (x-axis) and usefulness (y-axis). Larger/darker bullets correspond to larger eigenvalues.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

correct predictions 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 (Tsipras et al., 2019), 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 (Arora et al., 2019a, Jacot et al., 2018). Moreover, we are able to visualize useful non-robust features of standard models (Fig. 4). While this conceptual feature distinction has been highly influential in recent works that study the robustness of deep neural networks (see for example (Allen-Zhu and Li, 2022, Kim et al., 2021, Springer et al., 2021)), 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 (Goh, 2019)). Rather, these works either reveal such features in some indirect fashion, or accept their existence as an assumption. Here, we show that Neural Tangent 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 (Geiger et al., 2020, Fort et al., 2020, Baratin et al., 2021, Ortiz-Jiménez et al., 2021). 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. 6). 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).

- 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 (Tsipras et al., 2019, Ilyas et al., 2019) (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 Yuan and Wu (2021), 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. Bai et al. (2021) study aspects of robust models through their linearized sub-networks, but do not leverage NTKs.

2 Preliminaries

We introduce background material and definitions important to our analysis. Here, we restrict ourselves to binary classification, 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 (Szegedy et al., 2014) for f if

i) the 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

ii) $f(\tilde{\mathbf{x}}) \neq y$. 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 (Goodfellow et al., 2015)

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

or iterative versions for solving it (Kurakin et al., 2017, Madry et al., 2018). 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 (Carlini and Wagner, 2017). Even more surprisingly, it has been observed that adversarial examples crafted to "fool" one machine learning model are consistently capable of "fooling" others (Papernot et al., 2016, 2017), 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 (Goodfellow et al., 2015, Madry et al., 2018) and empirically holds as the strongest defense against adversarial examples (Madry et al., 2018, Zhang et al., 2019).

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 (Tsipras et al., 2019, Ilyas et al., 2019). 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 (Ilyas et al., 2019)[1], and extend them to multi-class problems (see Appendix A).

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
$$\mathbb{E}_{x,y\sim\mathcal{D}}\left[\mathbb{1}_{\{\text{sign}[\phi(x)]=y\}}\right] = \rho \tag{3}$$
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
$$\mathbb{E}_{x,y\sim\mathcal{D}}\left[\inf_{\delta\in\mathcal{B}} \mathbb{1}_{\{\text{sign}[\phi(x+\delta)]=y\}}\right] = \gamma \tag{4}$$

In general, a feature adds predictive value if it gives an advantage above guessing the most likely label, i.e. $\rho > \max_{y'\in\{\pm 1\}} \mathbb{E}_{x,y\sim\mathcal{D}}[\mathbb{1}_{\{y'=y\}}]$, 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 the previous formal definitions are in no way restricted to activations, and we will show how to exploit them 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 $\mathcal{Y} \in \{\pm 1\}^n$. 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 (Jacot et al., 2018, Liu et al., 2020), 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 (Jacot et al., 2018, Lee et al., 2019, Arora et al., 2019b)
$$f_t(\mathbf{x}) = \Theta(\mathbf{x}, \mathcal{X})^\top \Theta^{-1}(\mathcal{X}, \mathcal{X})(I - e^{-\lambda\Theta(\mathcal{X},\mathcal{X})t})\mathcal{Y}, \tag{7}$$
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].

[1] We distinguish useful and robust features based on their accuracy as classifiers, not in terms of correlation with the labels as in Ilyas et al. (2019), 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.

[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 (Chizat et al., 2019).

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").

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 B, 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. (Carlini and Wagner, 2017))) or through a Taylor expansion around the test input, and we present such alternative derivations in Appendix B. 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) 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 (Yuan and Wu, 2021, Nguyen et al., 2021), 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 Appendix C.

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. 3 (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. 3 (bottom). We reproduce these plots for MNIST in Appendix C, leading to similar conclusions.

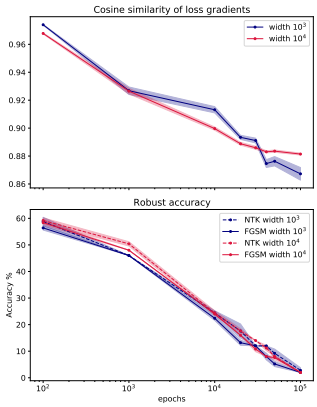


Figure 3: **Top**. Cosine similarity between the loss gradient of the neural net and of the NTK prediction for the same time point. **Bottom**. Robust accuracy of neural net against its own adversarial examples (solid) and corresponding NTK examples (dashed). CIFAR10, car vs plane.

When concerned with security aspects of neural nets, adversarial attacks are mainly characterised as either *white-box* or *black-box*

attacks (Papernot et al., 2017). 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 (Papernot et al., 2016) or through carefully crafted input-output pairs fed to the target model (Chen et al., 2017, Ilyas et al., 2018, Andriushchenko et al., 2020). 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 Eq. (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 (Arora et al., 2019a, Basri et al., 2019, Bietti and Mairal, 2019, Basri et al., 2020). 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 D). 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 (Novak et al., 2020), built on top of JAX (Bradbury et al., 2018). We summarize our findings on how these features behave:

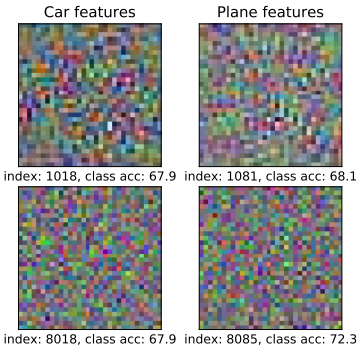


Figure 4: Non-robust, useful features earlier and later in the spectrum, for CIFAR10 car and plane.

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. 4). 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.

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 D). Since these eigenvectors are the ones fitted first during training (Arora et al., 2019a, Jacot et al., 2018), it is no wonder that the loss gradient evolves from coherence to noise, as observed in Fig. 7b. 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 (Rahaman et al., 2019) and theoretical case studies (Basri et al., 2019, Bietti and Mairal, 2019, Basri et al., 2020) 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 (in function space).

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 Appendix D.3).

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 (Geiger et al., 2020, Fort et al., 2020, Baratin et al., 2021, Ortiz-Jiménez et al., 2021) and theoretically (Atanasov et al., 2022), 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 Appendix E, 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. Fort et al.

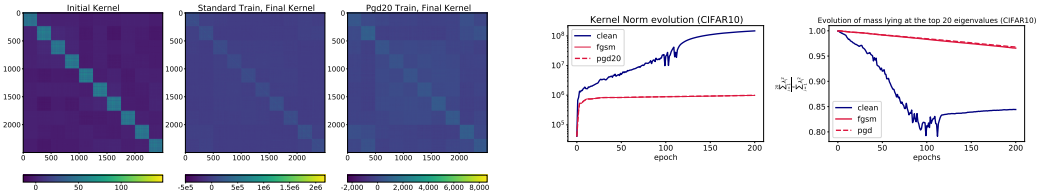


Figure 5: **Left**: Kernel Matrices for a mini batch of size 256. Left to Right: Kernel at initialization, Kernel after standard training, Kernel after adversarial training (20 PGD steps). The standard kernel grows significantly more than the adversarial one. **Right**: (a) Kernel Frobenius norm evolution, and (b) concentration on the top 20 eigenvalues during standard and adversarial training. Setting: CIFAR10, $\ell_\infty = 8/255$.

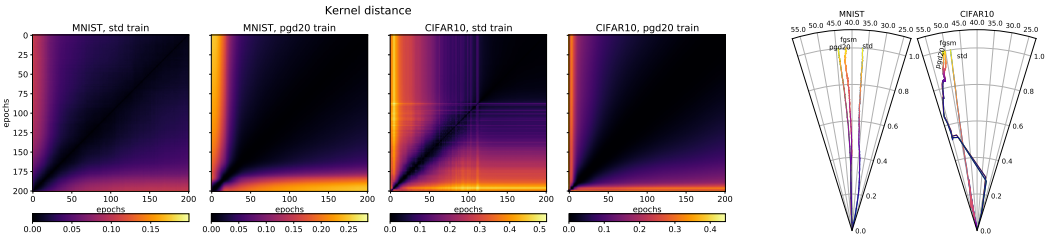


Figure 6: **Left:** Rotation (Eq. (12)) of the empirical NTK during standard, and adversarial training. Left to right: MNIST, standard, MNIST adversarial, CIFAR standard, CIFAR adversarial. **Right:** Kernel trajectories in polar space (Eq. (13)) for MNIST (left) and CIFAR10 (right). Darker colors indicate earlier epochs.

(2020)):

$$d(\Theta_i, \Theta_j) = 1 - \frac{\mathrm{Tr}(\Theta_i \Theta_j^\top)}{\sqrt{\mathrm{Tr}(\Theta_i \Theta_i^\top)}\sqrt{\mathrm{Tr}(\Theta_j \Theta_j^\top)}}. \tag{12}$$

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:

$$r_t = \frac{\|\Theta_t - \Theta_0\|_F}{\|\Theta_f - \Theta_0\|_F}, \quad \theta_t = \arccos\left(1 - d(\Theta_t, \Theta_0)\right), \tag{13}$$

where Θ_0, Θ_f are the initial and final kernel, respectively. Fig. 6 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. 5 depicts concentration on the top 20 eigenvalues during training.

Our findings show that similar to what has been reported in prior work (Fort et al., 2020), 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. 6 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. 5). 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 (Geiger et al., 2020, Fort et al., 2020, Ortiz-Jiménez et al., 2021) 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 (see Appendix E.1 for a study of linearized 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 Fig. 14) 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. 5 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 (Long, 2021).

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.

Acknowledgements

The authors would like to thank Jingtong Su, Alberto Bietti, Yunzhen Feng, and Artem Vysogorets for fruitful discussions and feedback in various stages of this work. NT thanks Dimitris Tsipras for a helpful discussion in the beginning of this project. The authors would like to acknowledge support through the National Science Foundation under NSF Award 1922658. This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise.

References

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.

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.

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 2019a.

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, 2019b.

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

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.

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.

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.

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.

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.

M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert. Efficient and modular implicit differentiation. *arXiv preprint arXiv:2105.15183*, 2021.

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`.

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.

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.

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.

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.

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.

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, 2019a.

S. S. Du, X. Zhai, B. Póczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b.

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.

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.

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.

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.

J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL `http://github.com/google/flax`.

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.

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.

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.

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.

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.

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.

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.

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

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.

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

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.

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.

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

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.

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.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. 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 8024–8035, 2019.

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.

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.

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.

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.

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.

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.

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.

E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

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.

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 Sec. 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]

A Robust and Non-Robust features

The idea that data features are to be blamed for the adversarial weakness of machine learning models was proposed in (Ilyas et al., 2019, Tsipras et al., 2019). In particular, Ilyas et al. (2019) show that training with adversarially perturbed images labeled with the "wrong" label yields classifiers with non-trivial test performance ("learning from non-robust features only"), while, in a dual experiment, they demonstrate that standard training with "robustified" data (data that presumably are "denoised" from non-robust features) produces a classifier with non-trivial *robust* accuracy ("relies only on robust features"). Motivated by these observations, the authors propose a model of robust/non-robust features that are hidden in the data, and whose presence determines the eventual robustness of models. To accompany the definitions of Sec. 2.2, we extend them for multiclass classification, since Sec. 4 introduces our NTK feature framework for both binary and multiclass problems.

Let $\mathcal{D}$ be the data generating distribution, with $x \in \mathcal{X}$ (input space) and $y \in \{1, \dots, k\}$ (action space). We define features $\phi : \mathcal{X} \to \mathbb{R}^k$ as functions from the input to the action space, and categorize them as follows, according to their performance as classifiers. Fix $\rho, \gamma \geq 0$:

1. ρ-**Useful** feature: A feature ϕ is called ρ-*useful* if
$$\mathbb{E}_{x,y\sim\mathcal{D}}\left[\mathbb{1}_{\{\arg\max_{i\in[k]}\phi_i(x)=y\}}\right] = \rho \tag{14}$$
2. γ-**Robust** feature: A feature ϕ is called γ-*robust* if it is predictive of the true label under any perturbation inside a bounded "ball" $\mathcal{B}$, that is if
$$\mathbb{E}_{x,y\sim\mathcal{D}}\left[\inf_{\delta\in\mathcal{B}} \mathbb{1}_{\{\arg\max_{i\in[k]}\phi_i(x+\delta)=y\}}\right] = \gamma \tag{15}$$
3. **Useful, non-robust** feature: A feature is called **useful, non-robust** if it confers an advantage above guessing the most likely label, i.e. $\exists \rho > \max_{i\in[k]} \mathbb{E}_{x,y\sim\mathcal{D}}[\mathbb{1}_{\{i=y\}}]$, but is γ-robust only for $\gamma \approx 0$ (within some precision).

The above framework was introduced by (Ilyas et al., 2019, Tsipras et al., 2019), and we have slightly adapted it in terms of accuracy as classifiers derived from features. Goh (2019) showed how such feature functions arise in a simple linear model, and proposed two mechanisms to construct useful, non-robust features. In (Allen-Zhu and Li, 2022), the authors view the weights of neural networks as features, and show that adversarial training "purifies/robustifies" them.

B Derivation of Adversarial Perturbations for Kernel Regression

In this section, we derive expressions for adversarial attacks on Neural Tangent Kernels presented in the main paper, as well as additional derivations obtained from first-order expansions around the input.

B.1 Adversarial Perturbations from Cross-Entropy Loss

We first derive the expression in Eq. (8) of the paper. Let $\mathbf{x} \in \mathbb{R}^d$ be an input to the NTK prediction

$$f_t(\mathbf{x}) = \Theta(\mathbf{x}, \mathcal{X})^\top \Theta^{-1}(\mathcal{X}, \mathcal{X})(I - e^{-\lambda\Theta(\mathcal{X},\mathcal{X})t})\mathcal{Y}, \tag{16}$$

where $(\mathcal{X}, \mathcal{Y})$ is a dataset of size n. We consider the binary and the multiclass case separately.

In the **binary** case, where $y \in \{\pm 1\}$, we feed expression Eq. (16) to a sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$ and maximize the cross entropy loss between the output and the true label:

$$\mathcal{L}(\mathbf{x}, y) = -\hat{y} \log\left(\sigma(f_t(\mathbf{x}))\right) - (1 - \hat{y}) \log\left(1 - \sigma(f_t(\mathbf{x}))\right), \tag{17}$$

where we set $\hat{y} = \frac{y+1}{2}$ to lie in $\{0, 1\}$. We compute the gradient of the loss with respect to $\mathbf{x}$:

$$\begin{aligned} \nabla_{\mathbf{x}} \mathcal{L}(\sigma(f_t(\mathbf{x})), y) &= -\frac{\hat{y}}{\sigma(f_t(\mathbf{x}))} \nabla_{\mathbf{x}} \sigma(f_t(\mathbf{x})) + \frac{(1 - \hat{y})}{1 - \sigma(f_t(\mathbf{x}))} \nabla_{\mathbf{x}} \sigma(f_t(\mathbf{x})) \\ &= \frac{\sigma(f_t(\mathbf{x})) - \hat{y}}{\sigma(f_t(\mathbf{x}))(1 - \sigma(f_t(\mathbf{x})))} \nabla_{\mathbf{x}} \sigma(f_t(\mathbf{x})) = \left(\sigma(f_t(\mathbf{x})) - \hat{y}\right) \nabla_{\mathbf{x}} f_t(\mathbf{x}). \end{aligned} \tag{18}$$

So the optimal one-step attack, under an ℓ_∞ adversary, reduces to computing perturbation

$$\boldsymbol{\eta} = \epsilon \cdot \operatorname{sign}\left(\left(\sigma(f_t(\mathbf{x})) - \hat{y}\right) \nabla_{\mathbf{x}} f_t(\mathbf{x})\right) = -\epsilon y \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} f_t(\mathbf{x})\right), \tag{19}$$

since $\sigma(u) \in (0, 1)$ for all $u \in \mathbb{R}$.

In the case of a **k-class** classification problem with one hot labels $\mathcal{Y} \in \mathbb{R}^{n \times k}$, we can express the cross entropy loss between the NTK predictions Eq. (16) and the labels as:

$$\mathcal{L}(\mathbf{x}, y) = -\log \frac{e^{f_{t,y}(\mathbf{x})}}{\sum_{r=1}^{k} e^{f_{t,r}(\mathbf{x})}} = -f_{t,y}(\mathbf{x}) + \log \sum_{r=1}^{k} e^{f_{t,r}(\mathbf{x})}, \tag{20}$$

where $f_{t,r}$ denotes the r-th output of Eq. (16). Computing the loss gradient as before yields the optimal perturbation $\boldsymbol{\eta}$,

$$\boldsymbol{\eta} = \epsilon \cdot \operatorname{sign}\left(-\nabla_{\mathbf{x}} f_{t,y}(\mathbf{x}) + \frac{\sum_{r=1}^{k} e^{f_{t,r}(\mathbf{x})} \nabla_{\mathbf{x}} f_{r,y}(\mathbf{x})}{\sum_{r=1}^{k} e^{f_{r,t}(x)}}\right). \tag{21}$$

The above calculations allow us to speed up the computation of the attacks in the case of NTKs with closed form expression, since the gradient

$$\nabla_{\mathbf{x}} f_{t,r}(\mathbf{x}) = D\Theta(\mathbf{x}, \mathcal{X})^\top \Theta^{-1}(\mathcal{X}, \mathcal{X})(I - e^{-\lambda\Theta(\mathcal{X},\mathcal{X})t})\mathcal{Y}_{:,r}, \tag{22}$$

with D being the Jacobian of Θ wrt to $\mathbf{x}$, can be pre-computed, without the need for auto-differentiation tools. We leverage this in the experiments of Sec. 3.

B.2 Alternative Approaches to Generate Perturbations

One can derive other perturbation variants by changing the loss function from cross-entropy to other functions studied in the literature in this context (e.g. (Carlini and Wagner, 2017)). Alternatively, we can study the output $f_t(x)$ on a test input x directly to devise strategies to most efficiently perturb it, using a Taylor expansion around the input, leading to a linear expression (shown here for scalar kernels):

$$f(\mathbf{x} + \boldsymbol{\eta}) \approx f(\mathbf{x}) + \boldsymbol{\eta}^T \mathbf{z}, \tag{23}$$

for some $z \in \mathbb{R}^d$ that depends on the training data and the NTK kernel only.

Binary case: Suppose we would like to evaluate a model described by Eq. (7) at the end of training,

$$f_\infty(\mathbf{x}) = \Theta(\mathbf{x}, \mathcal{X})^\top \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y} \tag{24}$$

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 $\Theta(\tilde{\mathcal{X}}, \mathcal{X})$ as follows:

$$\Theta(\tilde{\mathbf{x}}_i, \mathbf{x}_j) = \Theta(\mathbf{x}_i + \boldsymbol{\eta}_i, \mathbf{x}_j) \approx \Theta(\mathbf{x}_i, \mathbf{x}_j) + \nabla_{\mathbf{x}_i} \Theta(\mathbf{x}_i, \mathbf{x}_j)^T \boldsymbol{\eta}_i. \tag{25}$$

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

$$\Theta_{i,:}(\tilde{\mathcal{X}}, \mathcal{X})^T = \Theta_{i,:}(\mathcal{X}, \mathcal{X})^T + \underbrace{\begin{pmatrix} \nabla_{\mathbf{x}_i} \Theta(\mathbf{x}_i, \mathbf{x}_1)^T \\ \nabla_{\mathbf{x}_i} \Theta(\mathbf{x}_i, \mathbf{x}_2)^T \\ \vdots \\ \nabla_{\mathbf{x}_i} \Theta(\mathbf{x}_i, \mathbf{x}_n)^T \end{pmatrix}}_{\mathbf{A}_i \in \mathbb{R}^{n \times d}} \boldsymbol{\eta}_i. \tag{26}$$

Hence, $\Theta(\tilde{\mathcal{X}}, \mathcal{X})$ can be written as $\Theta(\mathcal{X}, \mathcal{X}) + \boldsymbol{\Delta}$ for a perturbation matrix $\boldsymbol{\Delta}$, with i-th row $\boldsymbol{\Delta}_{i,:} = \boldsymbol{\eta}_i^T \mathbf{A}_i^T$. Substituting into Eq. (24), we get:

$$f(\tilde{\mathcal{X}}) = (\Theta(\mathcal{X}, \mathcal{X}) + \boldsymbol{\Delta}) \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y} = \mathcal{Y} + \boldsymbol{\Delta} \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}. \tag{27}$$

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

$$\begin{aligned} f(\tilde{\mathbf{x}}_i) &= y_i + \boldsymbol{\Delta}_i \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y} \\ &= y_i + \boldsymbol{\eta}_i^T \mathbf{A}_i^T \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y} =: y_i + \boldsymbol{\eta}_i^T \mathbf{z}_i, \end{aligned} \tag{28}$$

leading to the linear expression advertised in Eq. (23). The adversarial perturbation $\boldsymbol{\eta}_i$ changes the output by $\boldsymbol{\eta}_i^T \mathbf{z_i} = \boldsymbol{\eta}_i^T \mathbf{A}_i^T \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}$, an expression which allows us to *compute* the adversarial perturbation to maximally change the output within the desired constraints on $\boldsymbol{\eta_i}$.

Since Eq. (24) describes regression models with LSE (L_2-loss), while adversarial examples typically are studied for classification models, we use thresholding (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) to turn Eq. (24) into a classifier.

Inspecting Eq. (28), 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:

$$\boldsymbol{\eta}_i = -\epsilon y_i \cdot \operatorname{sign}(\mathbf{A}_i^T \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}). \tag{29}$$

The computation of this optimal adversarial perturbation 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 (as presented, for instance, in Sec. C). In more complicated models or those that deviate from the assumptions for Eq. (24) 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.

Eq. (28) has been derived for perturbations of the *training* data. Consider now the case when we evaluate Eq. (24) on perturbations of unseen *test* data, that is on $\tilde{\mathcal{X}} + \boldsymbol{\epsilon}$. Then, Eq. (27) becomes:

$$f(\tilde{\mathcal{X}} + \boldsymbol{\epsilon}) = (\Theta(\tilde{\mathcal{X}}, \mathcal{X}) + \boldsymbol{\Delta})\Theta(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y} = f(\tilde{\mathcal{X}}) + \boldsymbol{\Delta}\Theta(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y}. \tag{30}$$

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 $\tilde{\mathcal{X}}$

$$\boldsymbol{\Delta}_{i,:} = \boldsymbol{\eta}_i^T \begin{pmatrix} \nabla_{\tilde{\mathbf{x}}_i}\Theta(\tilde{\mathbf{x}}_i, \mathbf{x}_1)^T \\ \nabla_{\tilde{\mathbf{x}}_i}\Theta(\tilde{\mathbf{x}}_i, \mathbf{x}_2)^T \\ \vdots \\ \nabla_{\tilde{\mathbf{x}}_i}\Theta(\tilde{\mathbf{x}}_i, \mathbf{x}_n)^T \end{pmatrix}^T \tag{31}$$

In practice, an adversary can calculate the NTK $\Theta(\mathcal{X}, \mathcal{X})$ offline 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.

Multiclass case: We adapt the derivations of the binary case to the setting where the output dimension is larger than one in the underlying regression setting (see below), resulting in a multiclass classifier. This leads to the multi-dimensional analogue of the linear Eq. (23) for $f(x) \in \mathbb{R}^k$, $y \in \mathbb{R}^k$:

$$f(\mathbf{x}_i + \boldsymbol{\eta}_i) = \mathbf{y}_i + \begin{pmatrix} \boldsymbol{\eta}_i^T \mathbf{z}_1 \\ \boldsymbol{\eta}_i^T \mathbf{z}_2 \\ \vdots \\ \boldsymbol{\eta}_i^T \mathbf{z}_k \end{pmatrix}. \tag{32}$$

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.

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.

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

$$\boldsymbol{\eta}_i = \varepsilon \cdot \operatorname{sign}(\arg \max_{r=1, r\neq r^\star}^{k} \|\mathbf{z}_r - \mathbf{z}_{r^\star}\|_1). \tag{33}$$

It is obtained by solving

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

Then

$$\begin{aligned} \boldsymbol{\eta}_i &= \arg \max_{\|\boldsymbol{\eta}_i\|_\infty \leq \epsilon} \max_{r=1, r\neq r^\star}^{k} \boldsymbol{\eta}_i^T \mathbf{z}_r - \boldsymbol{\eta}_i^T \mathbf{z}_{r^\star} \\ &= \arg \max_{\|\boldsymbol{\eta}_i\|_\infty \leq \epsilon} \max_{r=1, r\neq r^\star}^{k} \boldsymbol{\eta}_i^T (\mathbf{z}_r - \mathbf{z}_{r^\star}) \\ &= \varepsilon \cdot \operatorname{sign}(\arg \max_{r=1, r\neq r^\star}^{k} \|\mathbf{z}_r - \mathbf{z}_{r^\star}\|_1). \end{aligned} \tag{34}$$

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

$$\boldsymbol{\eta}_i = \varepsilon \cdot \operatorname{sign}(\sum_{r\neq r^*}^{n} (\mathbf{z}_r - \mathbf{z}_{r^\star})). \tag{35}$$

derived as follows

$$
\begin{aligned}
L_{ce}(f(\mathbf{x}_i+\boldsymbol{\eta}),\mathbf{y}_i) &= -\sum_{r=1}^{k} y_i^{(r)} \log\left(\frac{e^{y_i^{(r)}+\boldsymbol{\eta}_i^T\mathbf{z}_r}}{\sum_{r'=1}^{k} e^{y_i^{(r')}+\boldsymbol{\eta}_i^T\mathbf{z}'_r}}\right) \\
&= -\log\left(\frac{e^{1+\boldsymbol{\eta}_i^T\mathbf{z}_{r^\star}}}{\sum_{r=1,r\neq r^\star}^{k} e^{\boldsymbol{\eta}_i^T\mathbf{z}_r}+e^{1+\boldsymbol{\eta}_i^T\mathbf{z}_{r^\star}}}\right) \\
&= \log\left(\sum_{r\neq r^\star} e^{\boldsymbol{\eta}_i^T(\mathbf{z}_r-\mathbf{z}_{r^\star})-1}+1\right).
\end{aligned} \tag{36}
$$

Maximizing this cross entropy amounts 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[3], which leads to finding the maximum of

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

yielding Eq. (35).

Derivation of Eq. (32)*:* While we remain with $\mathcal{X}\in\mathbb{R}^{n\times d}$ as in the binary case, the other quantities change as $\mathcal{Y}\in\mathbb{R}^{nk}$, $f(\mathcal{X})\in\mathbb{R}^{nk}$ and $\Theta(\mathcal{X},\mathcal{X})\in\mathbb{R}^{nk\times nk}$, i.e. for each data pair $(\mathbf{x}_i,\mathbf{x}_j)$ we have $\Theta(\mathbf{x}_i,\mathbf{x}_j)\in\mathbb{R}^{k\times k}$. Let $\Theta_{lm}(\mathbf{x}_i,\mathbf{x}_j)$ denote the entry of $\Theta(\mathbf{x}_i,\mathbf{x}_j)$ 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. (25) we now obtain:

$$
\Theta_{lm}(\mathbf{x}_i+\boldsymbol{\eta},\mathbf{x}_j)\approx\Theta_{lm}(\mathbf{x}_i,\mathbf{x}_j)+\nabla_{\mathbf{x}_i}\Theta_{lm}^T(\mathbf{x}_i,\mathbf{x}_j)\boldsymbol{\eta}. \tag{37}
$$

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

$$
f(\tilde{\mathbf{X}})=\mathcal{Y}+\underbrace{\boldsymbol{\Delta}}_{\in\mathbb{R}^{nk\times nk}}(\Theta(\mathbf{X},\mathbf{X}))^{-1}\mathcal{Y}, \tag{38}
$$

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

$$
f(\mathbf{x}_i+\boldsymbol{\eta}_i)=\underbrace{\mathbf{y}_i}_{\in\mathbb{R}^k}+\boldsymbol{\Delta}_{ik:(i+1)k,:}\left(\Theta(\mathbf{X},\mathbf{X})\right)^{-1}\mathcal{Y}, \tag{39}
$$

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

$$
\underbrace{\begin{pmatrix}
\overbrace{\begin{matrix}
(\nabla_{\mathbf{x}_i}\Theta_{11}(\mathbf{x}_i,\mathbf{x}_1) & \dots & \nabla_{\mathbf{x}_i}\Theta_{1k}(\mathbf{x}_i,\mathbf{x}_1) & \nabla_{\mathbf{x}_i}\Theta_{11}(\mathbf{x}_i,\mathbf{x}_2) & \dots & \nabla_{\mathbf{x}_i}\Theta_{1k}(\mathbf{x}_i,\mathbf{x}_n))^T \\
(\nabla_{\mathbf{x}_i}\Theta_{21}(\mathbf{x}_i,\mathbf{x}_1) & \dots & \nabla_{\mathbf{x}_i}\Theta_{2k}(\mathbf{x}_i,\mathbf{x}_1) & \nabla_{\mathbf{x}_i}\Theta_{21}(\mathbf{x}_i,\mathbf{x}_2) & \dots & \nabla_{\mathbf{x}_i}\Theta_{2k}(\mathbf{x}_i,\mathbf{x}_n))^T \\
 & & & \vdots & & \\
(\nabla_{\mathbf{x}_i}\Theta_{k1}(\mathbf{x}_i,\mathbf{x}_1) & \dots & \nabla_{\mathbf{x}_i}\Theta_{kk}(\mathbf{x}_i,\mathbf{x}_1) & \nabla_{\mathbf{x}_i}\Theta_{k1}(\mathbf{x}_i,\mathbf{x}_2) & \dots & \nabla_{\mathbf{x}_i}\Theta_{kk}(\mathbf{x}_i,\mathbf{x}_n))^T
\end{matrix}}^{\in\mathbb{R}^{nk\times d}}\boldsymbol{\eta}_i
\end{pmatrix}}_{\in\mathbb{R}^{k\times nk}}. \tag{40}
$$

C Transfer Results for Wide Two-Layer Networks

In this section, we present additional experimental details for Sec. 3.2 and show the results of the experiments on MNIST. We train two-layer neural networks of the form

$$
f(\mathbf{x})=\frac{1}{\sqrt{m}}\mathbf{A}\max(\mathbf{W}\mathbf{x},0),\quad \mathbf{A}\in\{\pm1\}^{m\times k},\mathbf{W}\sim\mathcal{N}(0,0.01^2 I_{m\times d}), \tag{41}
$$

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

where the first layer is initialized with the normal distribution, the second layer is frozen to its initial random values in $\{\pm 1\}$, and m denotes the width of the network. The NTK of this architecture is given by

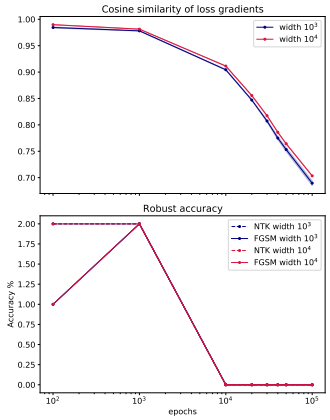
$$\Theta(\mathbf{x}_i, \mathbf{x}_j) = (\frac{1}{2} - \frac{\arccos\left(\frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)}{2\pi})\mathbf{x}_i^\top \mathbf{x}_j. \tag{42}$$

We choose this family of models in order to be consistent with early works that analyzed training and generalization properties of neural networks in the NTK regime (Arora et al., 2019a, Du et al., 2019b). We perform experiments on image classification on MNIST and on a binary task extracted from CIFAR-10 (car vs airplane). We train the networks in a regression fashion, minimizing the ℓ_2 loss between the predictions and one-hot vectors, using full-batch gradient descent on the entire dataset (full training data for MNIST and 5K images for each of car and airplane in binary CIFAR). We keep the learning rate fixed to 10^{-2} and vary the width of the network in $\{10^3, 10^4\}$. We train 3 networks for each dataset until convergence (10^5 epochs), each initialized with a different random seed. When we measure quantities from the neural net, we subtract the initial prediction f_0, since the NTK expression Eq. (16) does not take the initialization of the network into account. When attacking the models (ℓ_∞ attacks), we use perturbation budget $\epsilon = 0.3$ for MNIST and $\epsilon = 8/255$ for CIFAR-10. The experiments are performed with PyTorch (Paszke et al., 2019).

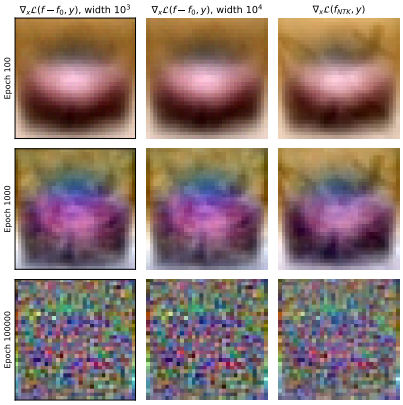
For each model, we calculate the loss gradients with respect to the input during training, and compare them to those derived for the NTK in Eqs. (18) and (21) for the binary and the multiclass task, respectively, using cosine similarity:

$$\frac{\nabla_x \mathcal{L}(f_t, y)^\top \nabla_x \mathcal{L}(f - f_0, y)}{\|\nabla_x \mathcal{L}(f_t, y)\|_2 \|\nabla_x \mathcal{L}(f - f_0, y)\|_2}, \tag{43}$$

where f_t is the NTK prediction defined in Eq. (16), f denotes the output of the neural net and f_0 is the initial prediction of the neural net (prior to training). In order to match the time-scales, we manually align the networks on epoch = 10^3 with a time-point for the NTK, and based on this number, we match the rest of the epochs assuming linear dependence (as theory predicts - Eq. (16)). Fig. 7a shows cosine similarity of loss gradients and robust accuracy of the network (evaluated against its own adversarial examples, and those from the NTK) for MNIST. Fig. 7b illustrates the similarity of loss gradients of neural nets and their NTKs for 3 different epochs.



(a) Comparison of NTK and neural net derived quantities for digit recognition (MNIST) during training.



(b) Loss gradients.

Figure 7: Visualizing the similarity of NTK and NN adversarial quantities. (a) **Top**. Cosine similarity between the loss gradient of the neural net and of the NTK prediction for the same time point (MNIST). **Bottom**. Robust accuracy of neural net against its own adversarial examples (solid) and corresponding NTK examples (dashed) for MNIST. Blue and red lines overlap in the second plot, and the effect of the random seed is insignificant. (b) Illustration of the similarity of loss gradients for NTK (**right** column) and neural nets of width 10^3 (**left**) and 10^4(**middle**) for a specific image extracted from CIFAR-10. Columns show gradients for different epochs ($10^2, 10^3, 10^4$, respectively).

Notice the very small discrepancy between the loss gradients of different networks (initialized with different random seeds) in Fig. 7a. They are all centered around the loss gradient of the NTK, a manifestation of transferability of adversarial examples, at least for models with the same architecture. The NTK framework might possibly provide a wider explanation of this phenomenon, also across architectures. For instance, for fully connected kernels, the NTK expression for kernels of depth l is a relatively simple function of expressions for depth $l-1$ (Jacot et al., 2018, Bietti and Mairal, 2019) which could explain transferability across architectures of varying depth.

D NTK Features: Additional Details

In this section, we present additional material for Sec. 4; we show derivations that are missing from the main text, and complement the plots by showing the same information for more architectures and datasets.

D.1 Loss Gradient Decomposition

First, recall our definitions of features from Sec. 4. Let $\mathcal{X}, \mathcal{Y}$ be a dataset, where $\mathcal{X} \in \mathbb{R}^{n \times d}$ and $\mathcal{Y} \in \{\pm 1\}^n$ (binary classification). Then, kernel regression on this dataset gives predictions of the form $f_\infty(\mathbf{x}) = \Theta(\mathbf{x}, \mathcal{X})^\top \Theta(\mathcal{X}, \mathcal{X})^{-1} \mathcal{Y}$. Given, the eigendecomposition of the Gram Matrix $\Theta(\mathcal{X}, \mathcal{X})$, we can decompose the prediction as follows

$$f_\infty(\mathbf{x}) = \Theta(\mathbf{x}, \mathcal{X})^\top \sum_{i=1}^{n} \lambda_i^{-1} \mathbf{v}_i \mathbf{v}_i^\top \mathcal{Y} = \sum_{i=1}^{n} f^{(i)}(\mathbf{x}), \tag{44}$$

where $f^{(i)} : \mathbb{R}^d \to \mathbb{R}^k$, $f^{(i)}(\mathbf{x}) = \lambda_i^{-1} \Theta(\mathbf{x}, \mathcal{X})^\top \mathbf{v}_i \mathbf{v}_i^\top \mathcal{Y}$. Notably, this means that the gradient of the cross entropy loss can be also understood as a composition of gradients coming from these features, as the following proposition shows.

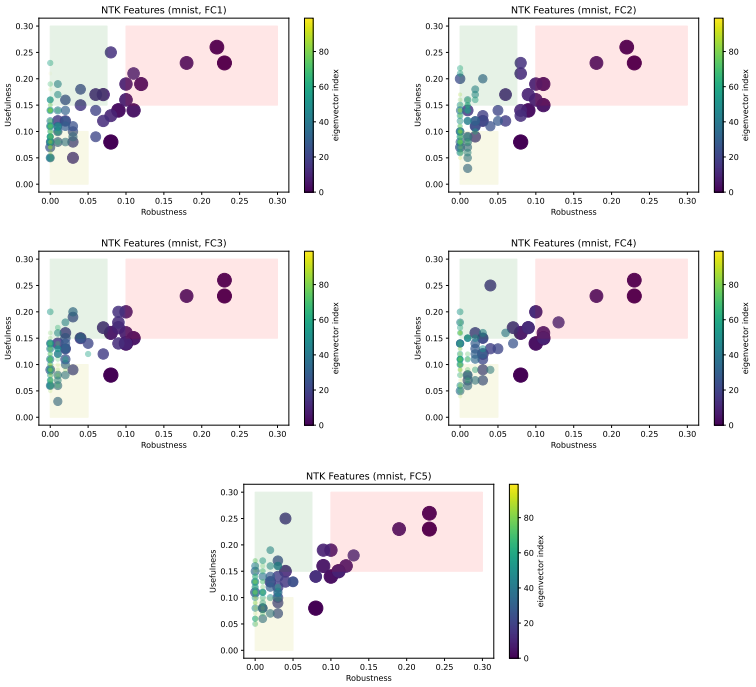


Figure 8: Robustness Usefulness space for various kernels, MNIST multiclass. The axes lie in $[0, 1]$. "Useful" features have usefulness above 0.1 (the random guessing probability for our balanced data set). The colored red, green and yellow boxes are arbitrary, meant to visually distinguish useful-robust from other features.

Proposition 1. *The loss gradient of f_∞ can be decomposed as follows:*

$$\nabla_{\mathbf{x}}\mathcal{L}(f_\infty(\mathbf{x}), y) = \sum_{i=1}^{n} \alpha_i \nabla_{\mathbf{x}}\mathcal{L}(f^{(i)}(\mathbf{x}), y), \tag{45}$$

where α_i is a quantity that depends on $\mathbf{x}, y$.

Proof. Recall from Eq. (18), that $\nabla_{\mathbf{x}}\mathcal{L}(f(\mathbf{x}), y) = \left(\sigma(f(x)) - \frac{y+1}{2}\right)\nabla_{\mathbf{x}} f(\mathbf{x})$. Then, we have

$$\begin{aligned}
\nabla_{\mathbf{x}}\mathcal{L}(f_\infty(\mathbf{x}), y) &= \left(\sigma(f_\infty(x)) - \frac{y+1}{2}\right)\nabla_{\mathbf{x}} f_\infty(\mathbf{x}) \\
&= \left(\sigma(f_\infty(x)) - \frac{y+1}{2}\right)\nabla_{\mathbf{x}} \sum_{i=1}^{n} f^{(i)}(\mathbf{x}) \\
&= \left(\sigma(f_\infty(x)) - \frac{y+1}{2}\right)\sum_{i=1}^{n} \nabla_{\mathbf{x}} f^{(i)}(\mathbf{x}) \\
&= \left(\sigma(f_\infty(x)) - \frac{y+1}{2}\right)\sum_{i=1}^{n} \frac{1}{\left(\sigma(f^{(i)}(x)) - \frac{y+1}{2}\right)}\nabla_{\mathbf{x}}\mathcal{L}(f^{(i)}(\mathbf{x}), y) \\
&= \sum_{i=1}^{n} \underbrace{\frac{\left(\sigma(f_\infty(x)) - \frac{y+1}{2}\right)}{\left(\sigma(f^{(i)}(x)) - \frac{y+1}{2}\right)}}_{\alpha_i} \nabla_{\mathbf{x}}\mathcal{L}(f^{(i)}(\mathbf{x}), y).
\end{aligned} \tag{46}$$

□

D.2 Additional Plots

Complementing Fig. 2 in the main text, we show (the first 100) NTK features in Robustness - Usefulness space defined in Sec. 4 for a larger number of architectures for both MNIST and CIFAR in Fig. 8 and 9. We use available analytical NTK expressions for standard FC{1,2,3,4,5} and CONV{1,2} architectures in the NTK regime to evaluate and decompose kernels on a subset of 10K MNIST training images and 10K binary CIFAR images - 5K cars and 5K airplanes. We note that within a dataset, the plots do not change much between architectures, speaking to the universal nature of these kernel-induced features.

D.3 Robust Features Alone are not Enough

Feature definitions outlined in Sec. 4 open an avenue to use traditional feature selection methods to search for robust models. In particular, here we rank the features of an NTK based on their robustness on a validation set (accuracy against adversarial examples computed from the same feature - setting: FGSM with $\epsilon = 0.3$ for MNIST or $\epsilon = 8/255$ for CIFAR-10). Specifically, we test and rank each "one-feature kernel" function $f^{(i)}(\mathbf{x}) := \lambda_i^{-1}\Theta(\mathbf{x}, \mathcal{X})^\top \mathbf{v}_i \mathbf{v}_i^\top \mathcal{Y}$. Given this ranking, we construct a sequence of new kernels $\Theta'_r(\mathcal{X}, \mathcal{X})$ by progressively aggregating the r most robust features with their original eigenvalues. This gives rise to kernel machines of the form $f'_r(\mathbf{x}) = \Theta(\mathbf{x}, \mathcal{X})^\top \Theta'_r(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y}$, where r indicates the number of top robust features kept. We present the results of this approach in Figures 10 (MNIST) and 11 (CIFAR-10), where we plot clean accuracy as well as robust accuracy against perturbation from the kernel f'_r itself as well as against "transfer" perturbations from the original (full) kernel.

On the binary classification task, some robustness can be garnered by keeping the most robust features and there seems to exist a sweet spot where the robustness is maximized (this seems to be consistent across other models as well). On multiclass MNIST, however, despite the relative simplicity of the dataset, we are not able to obtain non-trivial performance without compromising robustness. We conclude that it is unlikely that robust features (of standard models) alone are sufficient for robust classification, and the burden of some data augmentation, like in the form of adversarial training, seems necessary, at least for the models considered in our experiments.

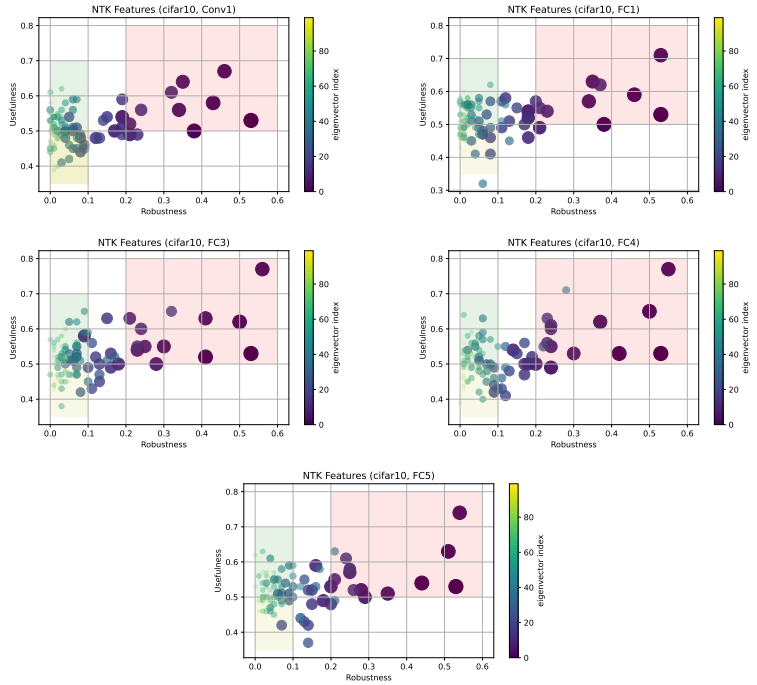


Figure 9: Robustness Usefulness space for various kernels, CIFAR-10 car vs plane. The axes lie in $[0, 1]$. Fig. 2 in the main text shows FC2 and CONV 2. "Useful" features have usefulness > 0.5 (the random guessing probability for a binary balanced data set). The colored red, green and yellow boxes are arbitrary, meant to visually distinguish useful-robust from other features.

E Experimental Details for the Kernel Dynamics Section

Here we provide the details of our experiments in Sec. 5, where we compare standard and adversarial training by tracking several kernel quantities.

For experiments with MNIST, we use a simple convolutional architecture with 3 layers. The first 2 layers compute a convolution (with a 3×3 kernel), followed by a ReLU and then by an average pooling layer (of kernel size 2×2 and stride 2). The 3rd layer is fully-connected with a ReLU non-linearity, followed by a linear prediction layer with 10 outputs. The layers have width 32, 64 and 256, respectively.

For CIFAR-10, we use a deeper architecture consisting of 6 layers. Layers 1 and 2, 3 and 4, 5 and 6 are fully convolutional with 32, 64 and 128 channels, respectively, and a kernel of size 3×3. There is a max pooling operation after layer 2, and average pooling after the final layer, followed by a linear prediction layer. Both pooling operations use a kernel of size 2×2 and stride 2.

We use a fixed learning rate of 10^{-2} for all experiments and no weight decay. We do not use any data augmentation, since we are interested in analyzing the behavior of kernels, rather than obtaining the best possible results. Stochastic gradient descent is used in all cases, with a batch size of 300 for MNIST and 250 for CIFAR-10. The kernels quantities are tracked for the same (first) batch during training. For adversarial training, we either used FGSM or PGD (for generating the adversarial examples) with 20 steps against ℓ_∞ adversaries. The maximum perturbation size is set to $\epsilon = 0.3$ and $\epsilon = 8/255$ (for MNIST and CIFAR-10, respectively), and in the case of PGD training we use an attack step size of $\alpha = 0.1$ and $\alpha = 2/255$, respectively. Experiments were run with JAX (Bradbury et al., 2018), and empirical NTKs were computed using the Neural Tangents Library (Novak et al., 2020). Neural nets were trained using Flax (Heek et al., 2020) and the JaxOpt library (Blondel et al., 2021), adapting code available from the JaxOpt repository. This code snippet was licensed under the Apache License, Version 2.0.

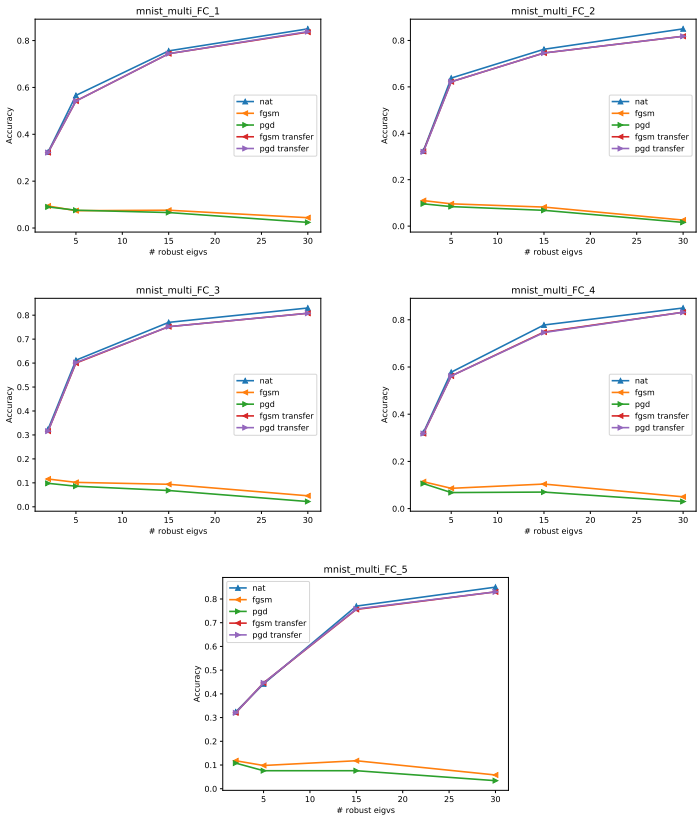


Figure 10: Robustness of kernel when keeping a few of the top robust features (MNIST). The Gram matrix is computed using 10k images from the training set. Blue lines show clean accuracy, red and purple (lines are overlapping) show accuracy against FGSM and PGD10 examples generated using the full kernel machines (consisting of all the features) and orange and green show the resulting robustness of the new model (FGSM and PGD10, respectively). Accuracy on the y-axis lies in $[0, 1]$.

Models were trained for 200 epochs. Fig. 12 summarizes the performance of the networks during training. In Fig. 13, we show how norm concentration evolves during training - similar to the plots for CIFAR-10 in Fig. 5, but for MNIST and for two choices of eigenvalue index cut-off.

Fig. 14 shows the polar dynamics for the top space (top 20 eigenvalues) of the kernel. We observe little to no change for adversarial training from Fig. 6 in the main text that showed the same information for the entire space, though for standard training there is less rotation in the top space. We entertain this as an indication that adversarial training modifies the "robust" (top) features of the kernel more than standard training.

Finally, Fig. 15 shows the values within the kernel matrices before and after training for MNIST for standard and adversarial training. We draw the same conclusions as the main text, namely the "standard" kernel has significantly larger values than the "adversarial" one.

E.1 Linearized Adversarial Training

Motivated by the apparent laziness of the kernel during adversarial training and the findings of prior works (Geiger et al., 2020, Fort et al., 2020) that considered linearization (with respect to the parameters) of the model after some epochs, we do the same for adversarial training.

We include a small study that linearizes the kernel after a certain number of epochs. In particular, Fig. 16 shows the training behavior after linearizing the CIFAR-10 model after 25 and 50 epochs, and also at initialization. After linearization, we continue adversarial training in this simple linearized model (meaning we generate adversarial examples from the linear model). We observe that adversarial

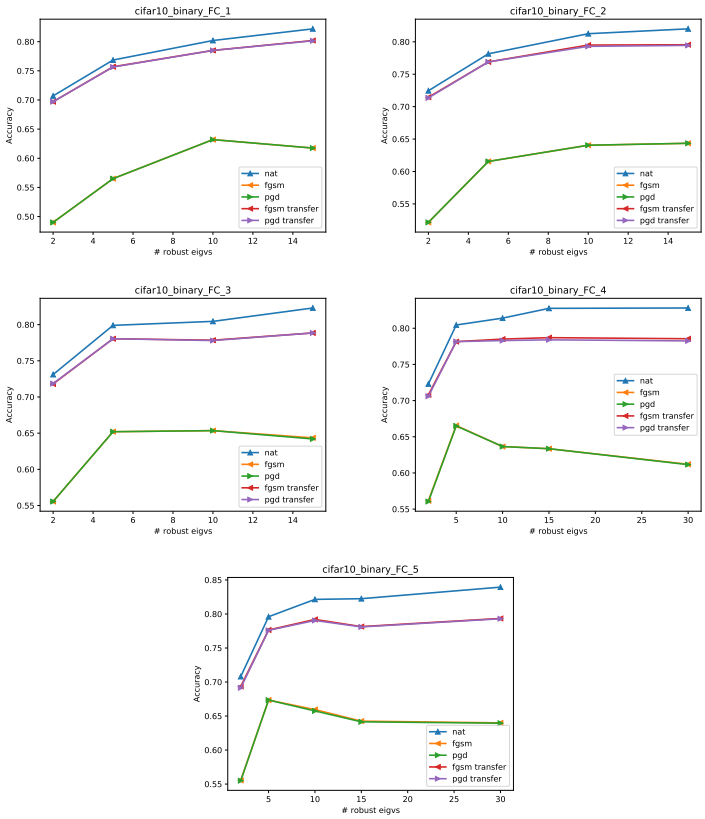


Figure 11: Robustness of kernel keeping a few of the robust features (CIFAR car vs plane). The Gram matrix is computed using all 10K images from the training set. Blue lines show clean accuracy, red and purple (lines are overlapping) show accuracy against FGSM and PGD10 examples generated using the full kernel machines (consisting of all the features) and orange and green show the resulting robustness of the new model (FGSM and PGD10, respectively). Accuracy on the y-axis lies in $[0, 1]$.

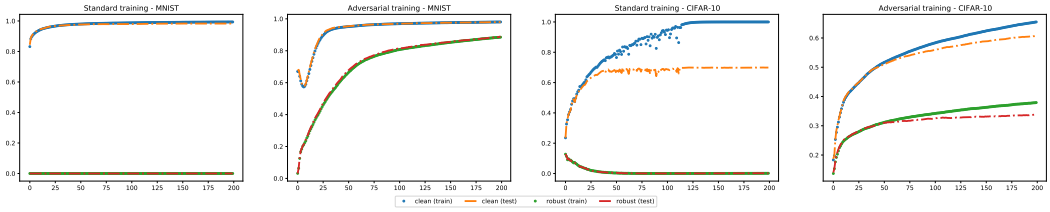


Figure 12: Training curves for networks trained in Sec. 5. From left to right: Standard training on MNIST, Adversarial (PGD-20) training on MNIST, Standard training on CIFAR-10, Adversarial (PGD-20) training on CIFAR-10. For each of the 4 settings, we show train/test accuracy on clean and on adversarially perturbed (PGD-20) data.

training continues, without a collapse of the training method. In comparison to non-linearized training (Fig. 12), training seems to stagnate. We also observe that the earlier we linearize, the greater the gap is between standard and robust performance. We leave the investigation of this intriguing phenomenon and a detailed comparison to standard training to future work.

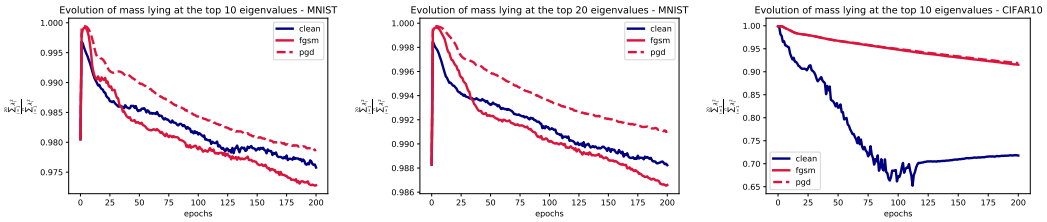


Figure 13: Concentration of norm during standard training vs adversarial training. **Left:** Concentration on top 10 (MNIST). **Middle:** Concentration on top 20 (MNIST). **Right.** Concentration on top 10 (CIFAR-10) (Fig. 5 in Sec. 5 shows Concentration on top 20 for CIFAR-10). For MNIST, we observe that when performing adversarial training with just one-step adversary (FGSM), the mass drops below the level of standard training. This is likely related to a phenomenon called catastrophic overfitting which is widespread in simple FGSM training settings (Wong et al., 2020).

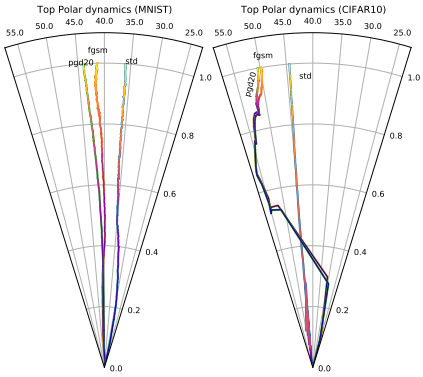


Figure 14: Top-20 dynamics on polar space.

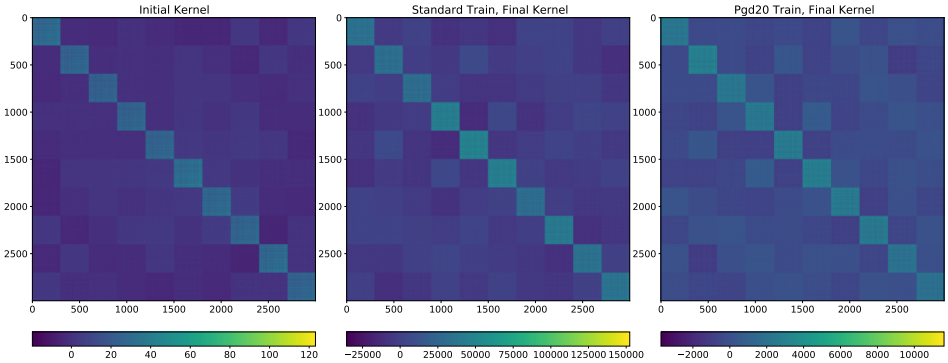


Figure 15: Kernel images for MNIST. Left to Right: Kernel at initialization, Kernel after standard training, Kernel after adversarial training (20 pgd steps). Notice that during training the values increase, but they do substantially more for standard training. Also, observe that for adversarial training there is more spread between different classes. Each little square in the diagonal corresponds to a different class.

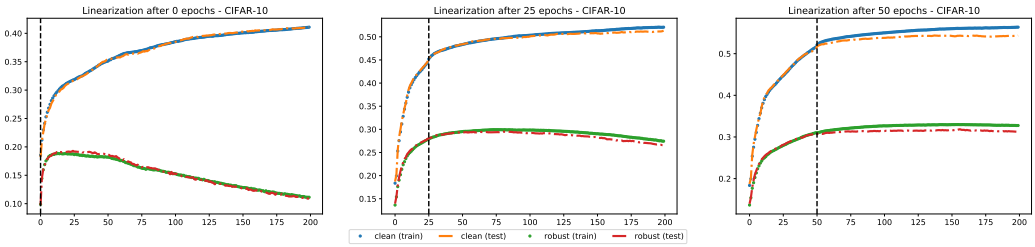


Figure 16: Linearized Adversarial Training on CIFAR-10. **Left** Linearized after 0 epochs. **Middle** Linearized after 25 epochs. **Right** Linearized after 50 epochs. Y-axis has range $(0, 1)$.