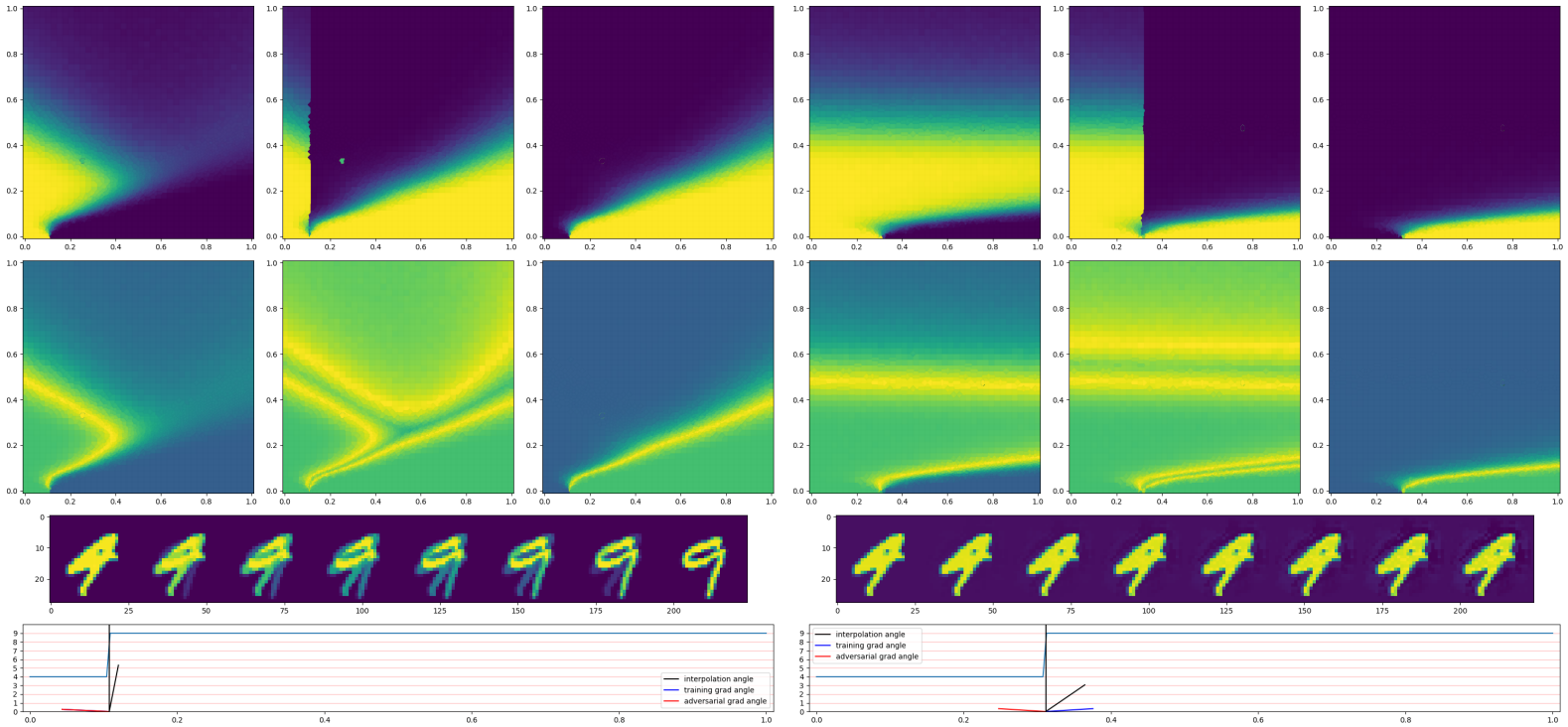
A Geometric Framework for Adversarial Vulnerability in Machine Learning



Brian Bell : University of Arizona

November 7, 2023

Contents

Goals

- Understand (mathematically) what Artificial Neural Networks (ANNs) are and how they are being used.

Goals

- Understand (mathematically) what Artificial Neural Networks (ANNs) are and how they are being used.
- Define a geometric approach to interpreting neural network classifiers.

Goals

- Understand (mathematically) what Artificial Neural Networks (ANNs) are and how they are being used.
- Define a geometric approach to interpreting neural network classifiers.
- Connect geometric approach with concept of robustness.

Goals

- Understand (mathematically) what Artificial Neural Networks (ANNs) are and how they are being used.
- Define a geometric approach to interpreting neural network classifiers.
- Connect geometric approach with concept of robustness.
- Define a kernel based representation which allows application of kernel based tools to ANNs

Goals

- Understand (mathematically) what Artificial Neural Networks (ANNs) are and how they are being used.
- Define a geometric approach to interpreting neural network classifiers.
- Connect geometric approach with concept of robustness.
- Define a kernel based representation which allows application of kernel based tools to ANNs
- Leverage our kernel based representation and these tools to get some useful results!

Goals

- Understand (mathematically) what Artificial Neural Networks (ANNs) are and how they are being used.
- Define a geometric approach to interpreting neural network classifiers.
- Connect geometric approach with concept of robustness.
- Define a kernel based representation which allows application of kernel based tools to ANNs
- Leverage our kernel based representation and these tools to get some useful results!
- Lay out further work which will connect representation approach to the geometric properties observed earlier.

Contents

Intriguing Properties of Neural Networks Szegedy et al. (2014)

Let $f : \mathbb{R}^m \to \{1, ..., k\}$ be a classifier and assume f has an associated continuous loss function denoted by $\text{loss}_f : \mathbb{R}^m \times \{1, ..., k\} \to \mathbb{R}^+$ and l a target adversarial .

Minimize $\|r\|_2$ subject to:

1. $f(x + r) = l$
2. $x + r \in [0, 1]^m$

Defining Adversarial Examples

Definition

Consider a point $x \in X$ with corresponding class $c \in C$ and a classifier $\mathcal{C} : X \to C$. We say that x admits an $(\varepsilon, d)-$*adversarial example* on $\mathcal{C}$ if there exists a point $\hat{x}$ such that $d(x, \hat{x}) < \varepsilon$ and $\mathcal{C}(\hat{x}) \neq c$.

This definition refers to the most general case of intentional mis-classification. The adversarial class can also be explicitly targeted:

Definition

Consider a point $x \in X$ with corresponding class $c \in C$ and a classifier $\mathcal{C} : X \to C$. We say that x admits an $(\varepsilon, d, c_t)-$*targeted adversarial example* on $\mathcal{C}$ if there exists a point $\hat{x}$ such that $d(x, \hat{x}) < \varepsilon$ and $\mathcal{C}(\hat{x}) = c_t$.

Contents

Background

Adversarial examples are not just a peculiarity, but seem to occur for most, if not all, DNN classifiers. For example, Shafahi et al. (2018) used isoperimetric inequalities on high dimensional spheres and hypercubes to conclude that there is a reasonably high probability that a correctly classified data point has a nearby adversarial example.
This has been reiterated using mixed integer linear programs to rigorously check minimum distances necessary to achieve adversarial conditions (Tjeng et al., 2017). Ilyas et al. (2019) showed that adversarial examples can arise from features that are good for classification but not robust to perturbation.

Contributions

1. We propose and implement two metrics based on the success of smoothed classification techniques: (γ, σ)-stability and γ-persistence defined with reference to a classifier and a given point (which can be either a natural or adversarial image, for example) and demonstrate their validity for analyzing adversarial examples.
2. We interpolate across decision boundaries using our persistence metric to demonstrate an inconsistency at the crossing of a decision boundary when interpolating from natural to adversarial examples.
3. We demonstrate via direct interpolation across decision boundaries and measurement of angles of interpolating vectors relative to the decision boundary itself that dimensionality is not solely responsible for geometric vulnerability of neural networks to adversarial attack.

Related Work : Distance-based robustness

- Khoury and Hadfield-Menell (2018) define a classifier to be robust if the class of each point in the data manifold is contained in a sufficiently large ball that is entirely contained in the same class. Larger minimum radius of this cover means more robust.
- Robustness by measuring distances from the data manifold to the decision boundary (Wang et al., 2020; Xu et al., 2023b; He et al., 2018).
- Radii in practice can be extremely small, as evidenced by results on the prevalence of adversarial examples, e.g., work by Shafahi et al. (2018) and in evaluation of ReLU networks with mixed integer linear programming e.g., work by Tjeng et al. (2017).
- Tsipras et al. (2018) investigated robustness in terms of how small perturbations affect the the average loss of a classifier. They define robust accuracy in terms of how often an adversarially perturbed example classifies correctly.
- Gilmer et al. (2018) use the expected distance to the nearest different class (when drawing a data point from the data distribution) to capture robustness

Stability

Definition

Let $\mathcal{C} : X \to L$ be a classifier, $x \in X$, $\gamma \in (0,1)$, and $\sigma > 0$. We say x is (γ, σ)-*stable* with respect to $\mathcal{C}$ if $\mathbb{P}[\mathcal{C}(x') = \mathcal{C}(x)] \geq \gamma$ for $x' \sim \rho = N(x, \sigma^2 I)$; i.e. x' is drawn from a Gaussian with variance σ^2 and mean x.

In the common setting when $X = R^n$, we have

$$\mathbb{P}[\mathcal{C}(x') = \mathcal{C}(x)] = \int_{R^n} \mathbb{1}_{\mathcal{C}^{-1}(\mathcal{C}(x))}(x') d\rho(x') = \rho(\mathcal{C}^{-1}\mathcal{C}(x)).$$

Note here that $\mathcal{C}^{-1}$ denotes preimage.
For the Gaussian measure, the probability above may be written more concretely as

$$\frac{1}{\left(\sqrt{2\pi}\sigma\right)^n} \int_{R^n} \mathbb{1}_{\mathcal{C}^{-1}(\mathcal{C}(x))}(x') e^{-\frac{|x-x'|^2}{2\sigma^2}} dx'. \tag{1}$$

Stability Approximation

We will conduct experiments in which we estimate this stability for fixed (γ, σ) pairs via a Monte Carlo sampling, in which case the integral (1) is approximated by taking N i.i.d. samples $x_k \sim \rho$ and computing

$$\frac{|x_k : \mathcal{C}(x_k) = \mathcal{C}(x)|}{N}.$$

Note that this quantity converges to the integral (1) as $N \to \infty$ by the Law of Large Numbers.
The ability to adjust the quantity γ is important because it is much weaker than a notion of stability that requires a ball that stays away from the decision boundary as by Khoury and Hadfield-Menell (2018). By choosing γ closer to 1, we can require the samples to be more within the same class, and by adjusting γ to be smaller we can allow more overlap.

Persistence

For any $x \in X$ not on the decision boundary, for any choice of $0 < \gamma < 1$ there exists a σ_γ small enough such that if $\sigma < \sigma_\gamma$ then x is (γ, σ)-stable. We can now take the largest such σ_γ to define persistence.

Definition

Let $\mathcal{C} : X \to L$ be a classifier, $x \in X$, and $\gamma \in (0, 1)$. Let σ_γ^* be the maximum σ_γ such that x is (γ, σ)-stable with respect to $\mathcal{C}$ for all $\sigma < \sigma_\gamma$. We say that x has γ-*persistence* σ_γ^*.

The γ-persistence quantity σ_γ^* measures the stability of the neighborhood of a given x with respect to the output classification.

Argmax Decision Boundaries

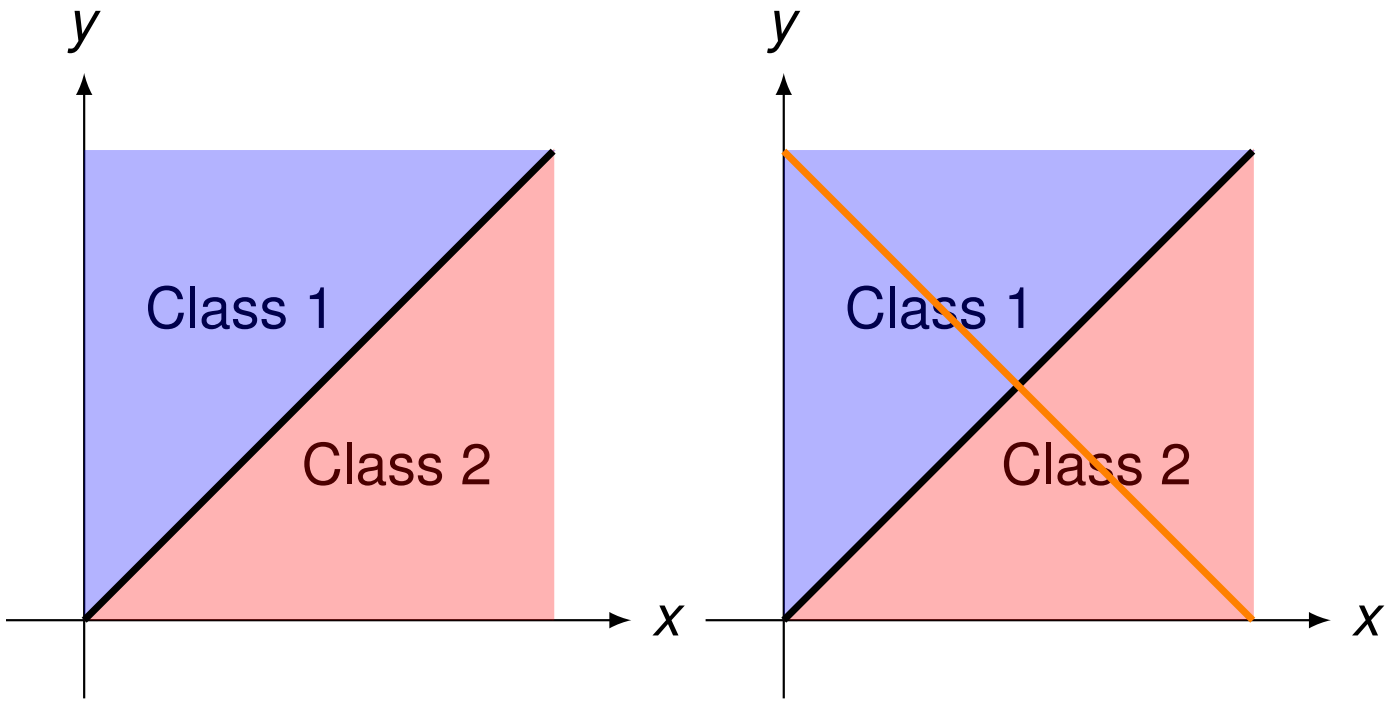


Figure: Decision boundary in $[0, 1] \times [0, 1]$ (left) and decision boundary restricted to probabilities (right) If the output of F are *probabilities* which add to one, then all points of x will map to the orange line on the right side of Figure 1.

Decision Boundary Definitions

Complement Definition

Definition

A point x is in the *decision interior* D'_f for a classifier $f : \mathbb{R}^N -> \mathcal{C}$ if there exists $\delta > 0$ such that $\forall \epsilon < \delta$, $|f(B_\epsilon(x))| = 1$.
The *decision boundary* of a classifier f is the closure of the complement of the decision interior $\overline{\{x : x \notin D'_f\}}$.

Level Set Definition

Definition

The decision boundary D of a probability valued function f is the pre-image of a union of all level sets of activations $A_c = c_1, c_2, ..., c_k$ defined by a constant c such that for some set of indices L, we have $c = c_i$ for every i in L and $c > c_j$ for every j not in L. The pre-image of each such set are all x such that $f(x) = A_c$ for some c.

Experiments : MNIST

We begin by sampling gaussians around test points relative to a fully connected ReLU network with layers of size 784, 100, 20, and 10 and small regularization $\lambda = 10^{-7}$ which is trained on the standard MNIST training set.

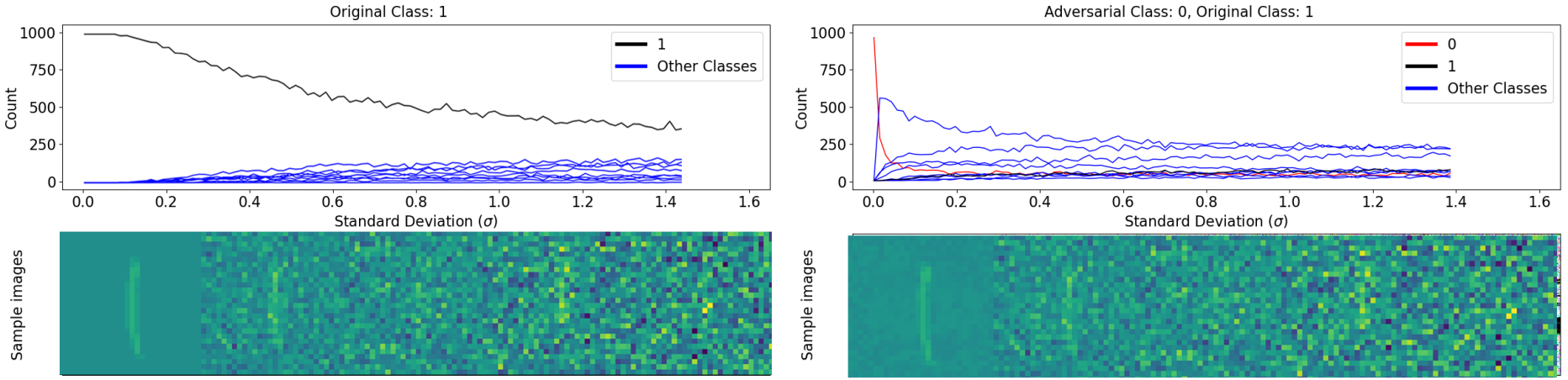


Figure: Frequency of each class in Gaussian samples with increasing variance around a natural image of class 1 (left) and around an adversarial attack of that image targeted at 0 generated using IGSM (right). The adversarial class (0) is shown as a red curve. The natural image class (1) is shown in black. Bottoms show example sample images at different standard deviations for natural (left) and adversarial (right) examples.

Experiments : MNIST Persistence

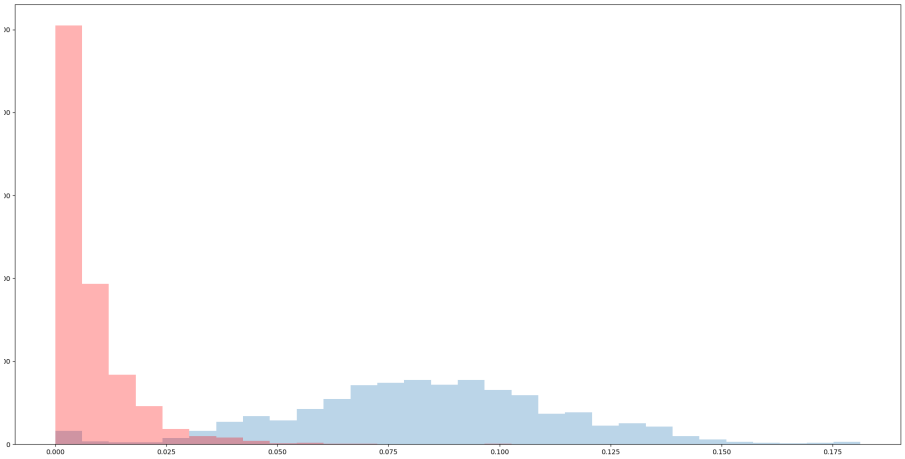


Figure: Histogram of 0.7-persistence of IGSM-based adversarial examples (red) and natural examples (blue) on MNIST.

Experiments : MNIST Persistence vs Architecture

Table: Recreation of Szegedy et al. (2013), Table 1 for the MNIST dataset. For each network, we show Testing Accuracy (in %), Average Distortion ($\|x\|_2/\sqrt{n}$) of adversarial examples, and new columns show average 0.7-persistence values for natural (Nat) and adversarial (Adv) images. 300 natural and 300 adversarial examples generated with L-BFGS were used for each aggregation.

Network	Test Acc	Avg Dist	Persist (Nat)	Persist (Adv)
FC10-4	92.09	0.123	0.93	1.68
FC10-2	90.77	0.178	1.37	4.25
FC10-0	86.89	0.278	1.92	12.22
FC100-100-10	97.31	0.086	0.65	0.56
FC200-200-10	97.61	0.087	0.73	0.56
C-2	95.94	0.09	3.33	0.027
C-4	97.36	0.12	0.35	0.027
C-8	98.50	0.11	0.43	0.0517
C-16	98.90	0.11	0.53	0.0994
C-32	98.96	0.11	0.78	0.0836

Experiments : ImageNet

For ImageNet (Deng et al., 2009), we used pre-trained ImageNet classification models, including alexnet (Krizhevsky et al., 2012) and vgg16 (Simonyan and Zisserman, 2014).

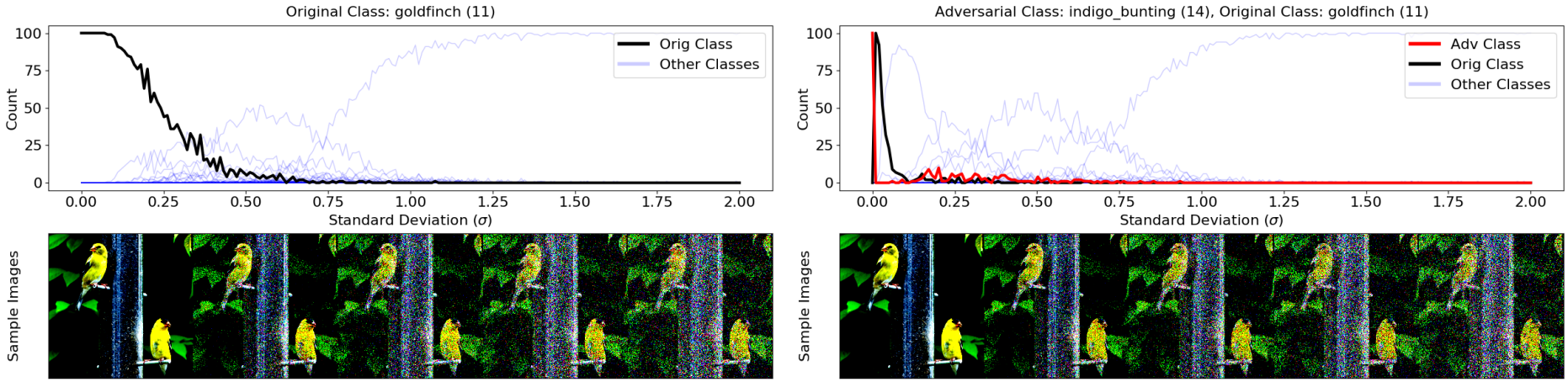


Figure: Frequency of each class in Gaussian samples with increasing variance around a `goldfinch` image (left) and an adversarial example of that image targeted at the `indigo_bunting` class and calculated using the BIM attack (right). Bottoms show example sample images at different standard deviations for natural (left) and adversarial (right) examples.

Experiments : ImageNet Persistence

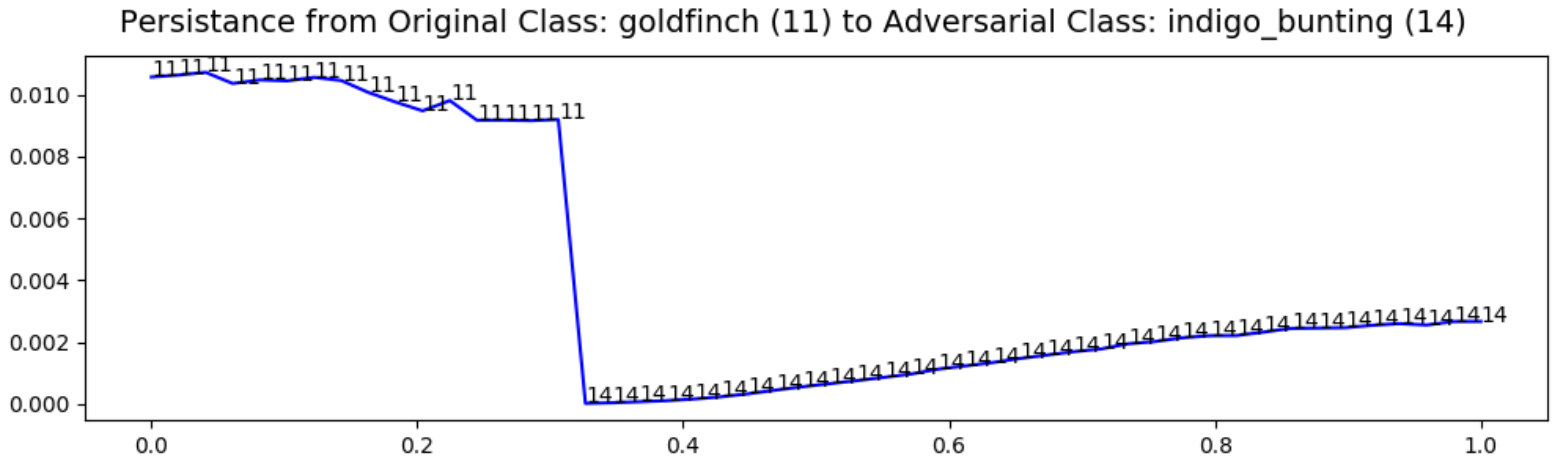


Figure: The 0.7-persistence of images along the straight line path from an image in class `goldfinch` (11) to an adversarial image generated with BIM in the class `indigo_bunting` (14) on a vgg16 classifier. The classification of each image on the straight line is listed as a number so that it is possible to see the transition from one class to another. The vertical axis is 0.7-persistence and the horizontal axis is progress towards the adversarial image.

Experiments : Decision Boundary Angles

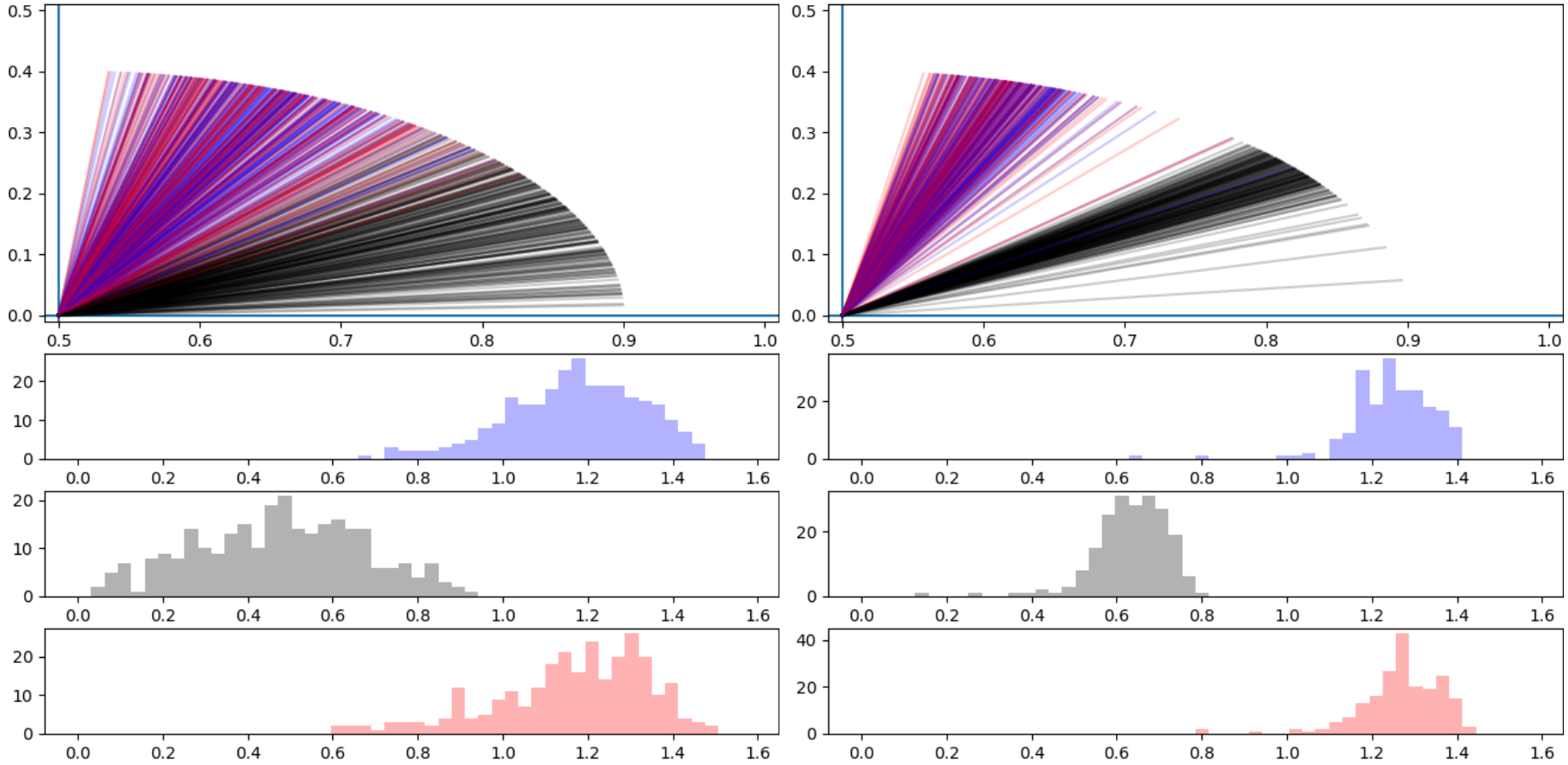


Figure: Decision boundary incident angles between test and test images (left) and between test and adversarial images (right). Angles (plotted Top) are referenced to decision boundary so $\pi/2$ radians (right limit of plots) corresponds with perfect orthogoonality to decision boundary. Lines and histograms measure angles of training gradients (Blue) linear interpolant (Black) and adversarial gradients (Red)

Conclusions

- Geometric properties including curvature and alignment to decision boundaries seem to be related with robustness.
- Direct computation of these geometric properties is expensive and at odds with the fact that much greater scale is needed to understand these properties in practice.
- We could benefit from faster methods to analyze geometric properties from models – Monte Carlo is what you do when you can't solve your differential equations by faster means!
- Decision boundaries are askew from interpolation between natural images!

Contents

Model Optimization Step

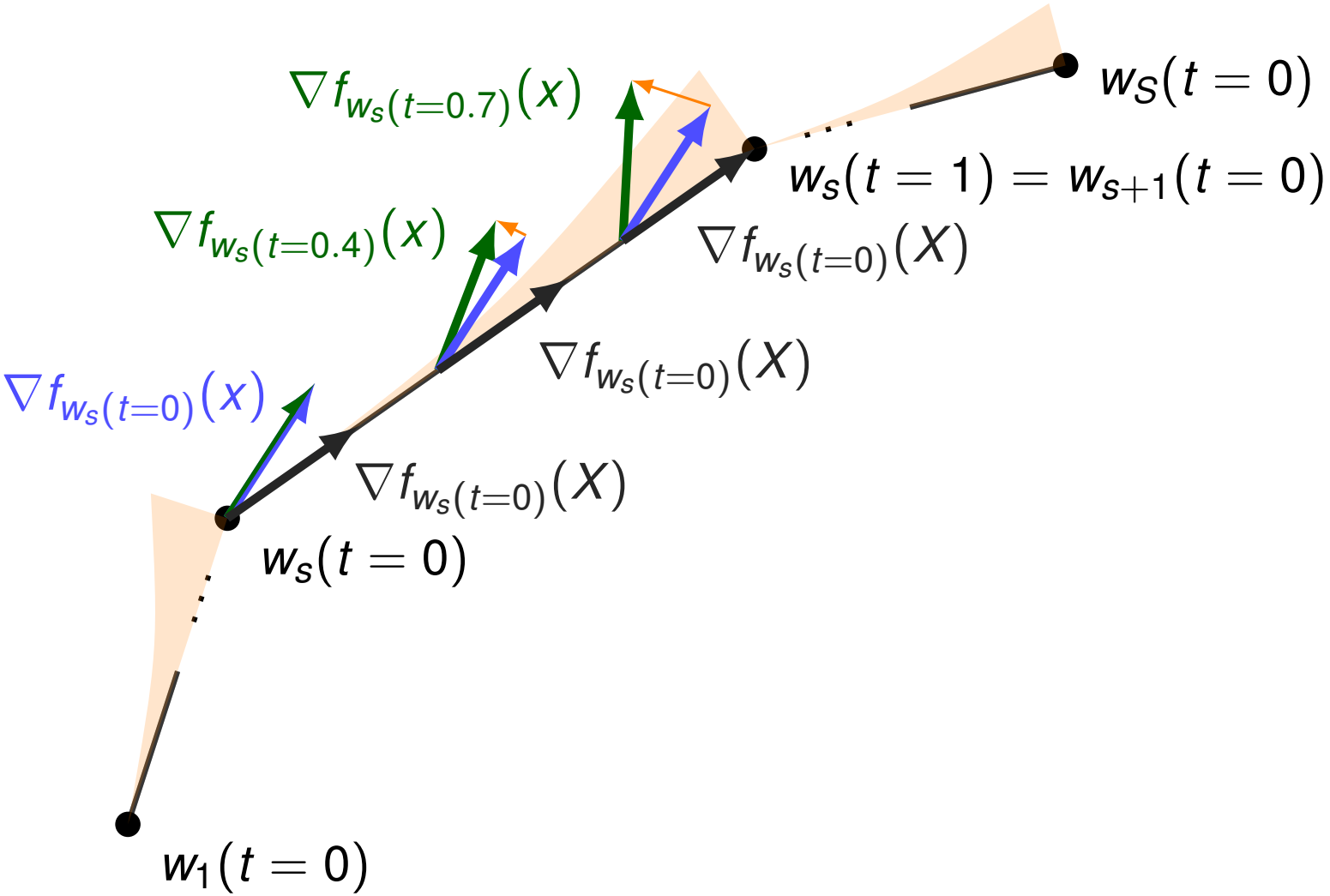


Figure: Comparison of test gradients used by Discrete Path Kernel (DPK) from prior work (Blue) and the Exact Path Kernel (EPK) proposed in this work (green) versus total training vectors (black) used for kernel formulations along a discrete

Kernel Approximation Error

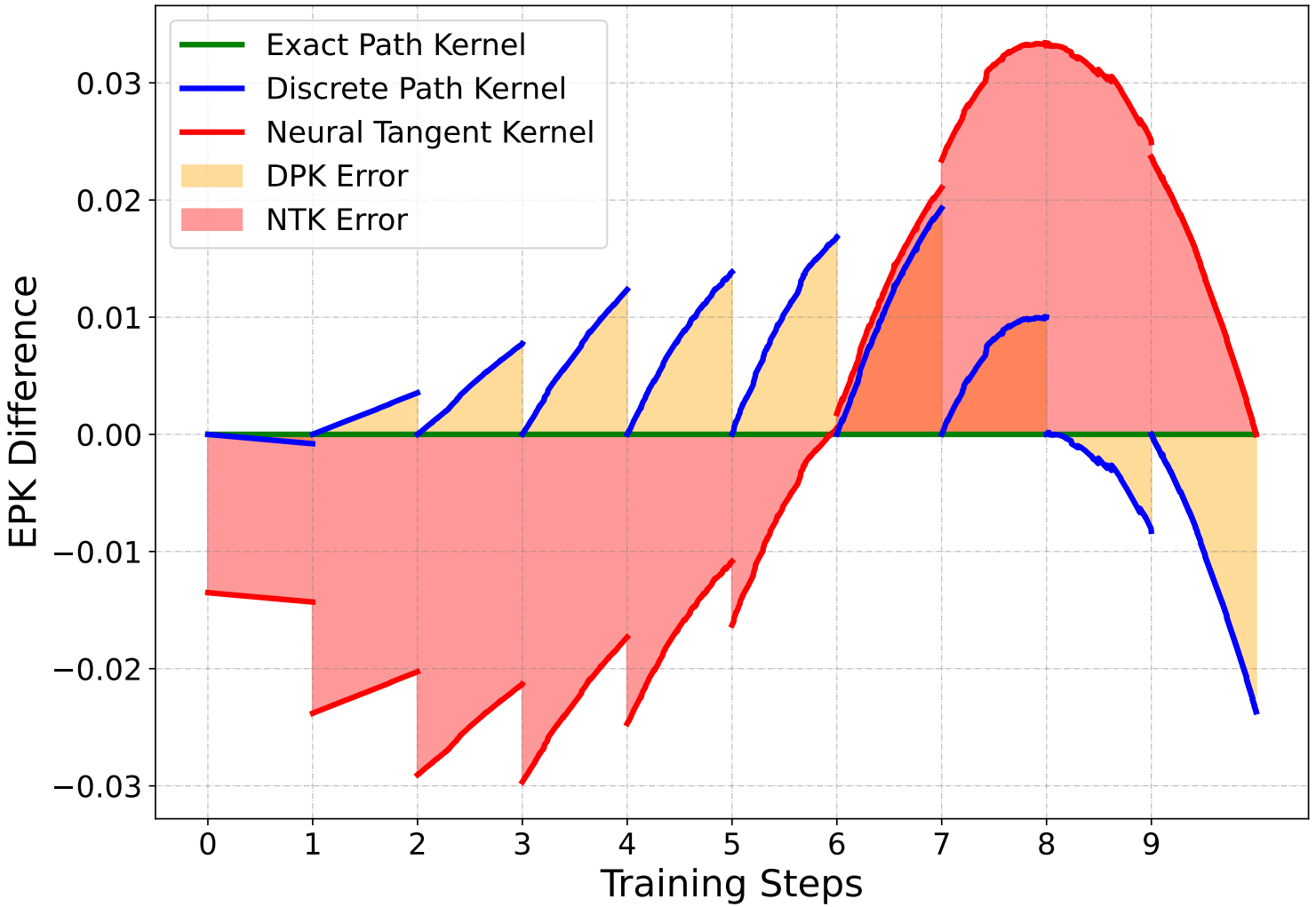


Figure: Measurement of gradient alignment on test points across the training path. The EPK is used as a frame of reference. The y-axis is exactly the difference between the EPK and other representations.

Contributions

This paper makes the following significant theoretical and experimental contributions:

1. We prove that finite-sized neural networks trained with finite-sized gradient descent steps and cross-entropy loss can be exactly represented as kernel machines using the EPK. Our derivation incorporates a previously-proposed path kernel, but extends this method to account for practical training procedures Domingos (2020a); Chen et al. (2021).
2. We demonstrate that it is computationally tractable to estimate the kernel underlying a neural network classifier, including for small convolutional computer vision models.
3. We compute Gram matrices using the EPK and use them to illuminate prior theory of neural networks and their understanding of uncertainty.
4. We employ Gaussian processes to compute the covariance of a neural network's logits and show that this reiterates previously observed shortcomings of neural network generalization.

Related Work : Discrete Path Kernel

- Domingos (2020b) attempts to establish a correspondence between kernel machines and parametric models trained by gradient descent in the case of a continuous training path (i.e. the limit as gradient descent step size $\varepsilon \to 0$)
- We will refer to this continuously integrated path representation as the Discrete Path Kernel (DPK).
- Chen et al. (2021) refined this formulation to establish an equivalence, however this equivalence requires a restrictive assumptions.
- A major limitation of this formulation is its reliance on a continuous integration over a gradient flow, which differs from the discrete forward Euler steps employed in real-world model training.
- The unbounded error of this approximation limits the value of the continuous path kernel in practical scenarios (Incudini et al., 2022).

Theory

Goal : Show that artificial neural networks are equivalent to kernel machines.

Definition

A *kernel* is a function of two variables which is symmetric and positive semi-definite.

Note: We define this equivalence in terms of the output of the parametric model $f_w(x)$ and our kernel method in the sense that they form identical maps from input to output. In the specific case of neural network classification models, we consider the mapping $f_w(x)$ to include all layers of the neural network up to and including the log-softmax activation function.

Theory : Kernel Method Definition

Definition

Given a Hilbert space X, a test point $x \in X$, and a training set $X_T = \{x_1, x_2, ...x_n\} \subset X$ indexed by I, a *Kernel Machine* is a model characterized by

$$\mathrm{K}(x) = b + \sum_{i \in I} a_i k(x, x_i) \tag{2}$$

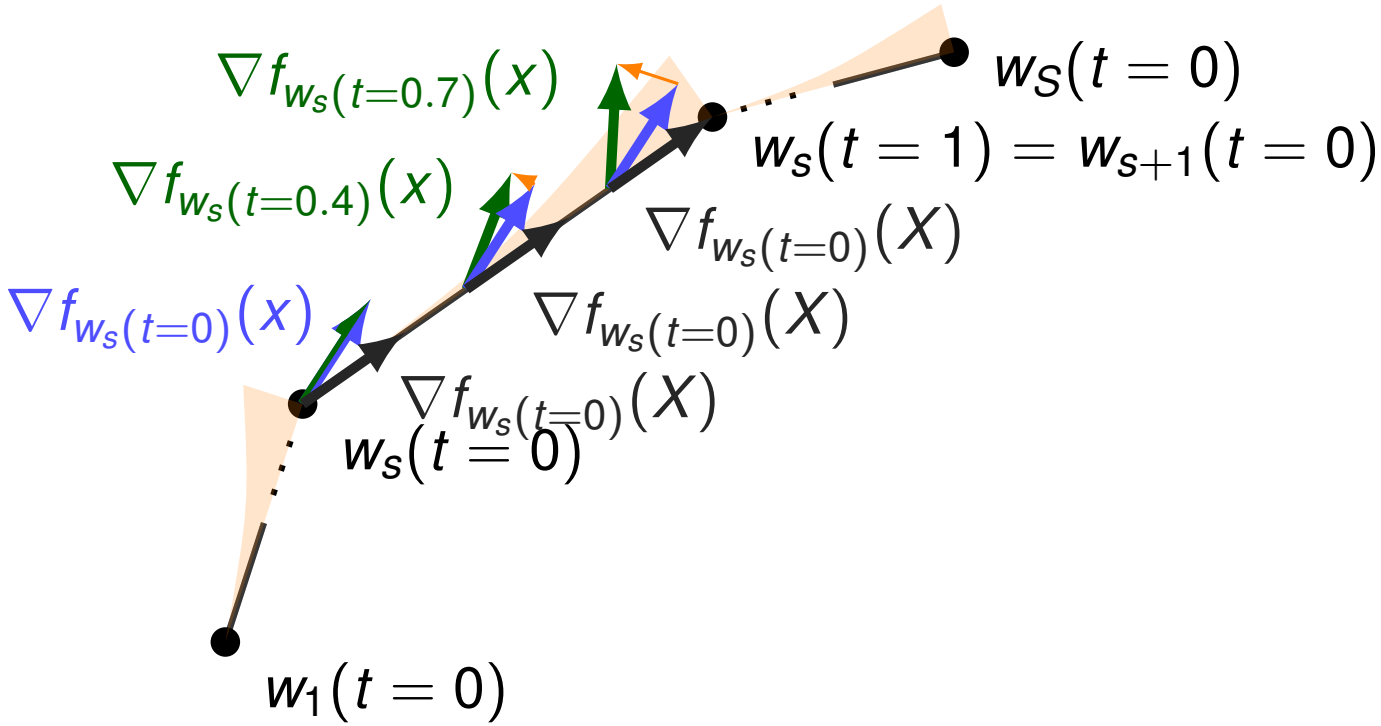
where the $a_i \in \mathbb{R}$ do not depend on x, $b \in \mathbb{R}$ is a constant, and k is a kernel. Rasmussen et al. (2006)

By Mercer's theorem Ghojogh et al. (2021) a kernel can be produced by composing an inner product on a Hilbert space with a mapping ϕ from the space of data into the chosen Hilbert space. We use this property to construct a kernel machine of the following form.

$$\mathrm{K}(x) = b + \sum_{i \in I} a_i \langle \phi(x), \phi(x_i) \rangle \tag{3}$$

Derivation

We first derive a kernel which is an exact representation of the change in model output over one training step, and then compose our final representation by summing along the finitely many steps.



Exact Path Kernel : Definition

Definition

Let $f_{w_s(t)}$ be an FPGM with all corresponding assumptions. Then, for a given training step s, the *exact path kernel* (EPK) can be written

$$K_{\text{EPK}}(x, x', s) = \int_0^1 \langle \phi_{s,t}(x), \phi_{s,t}(x') \rangle dt \tag{4}$$

where

$$\phi_{s,t}(x) = \nabla_w f_{w_s(t,x)}(x) \tag{5}$$

$$w_s(t) = w_s - t(w_s - w_{s+1}) \tag{6}$$

$$w_s(t, x) = \begin{cases} w_s(0), & \text{if } x_l = 1 \\ w_s(t), & \text{if } x_l = 0 \end{cases} \tag{7}$$

Note: ϕ is deciding whether to select a continuously or discrete gradient based on whether the data is from the training or testing space.

Derivation : Exact Path Kernel

Lemma

The exact path kernel (EPK) is a kernel.

Proof. We must show that the associated kernel matrix $K_{\text{EPK}} \in \mathbb{R}^{n \times n}$ defined for an arbitrary subset of data $\{x_i\}_{i=1}^{M} \subset X$ as $K_{\text{EPK},i,j} = \int_0^1 \langle \phi_{s,t}(x_i), \phi_{s,t}(x_j) \rangle dt$ is both symmetric and positive semi-definite.

Since the inner product on a Hilbert space $\langle \cdot, \cdot \rangle$ is symmetric and since the same mapping φ is used on the left and right, K_{EPK} is **symmetric**.

To see that K_{EPK} is **Positive Semi-Definite**, let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^\top \in \mathbb{R}^n$ be any vector. We need to show that $\alpha^\top K_{\text{EPK}} \alpha \geq 0$.

Derivation : Exact Path Kernel

$$\alpha^\top K_{\text{EPK}} \alpha = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \int_0^1 \langle \phi_{s,t}(x_i), \phi_{s,t}(x_j) \rangle dt \tag{8}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \int_0^1 \langle \nabla_w \hat{y}_{w_s(t,x_i)}, \nabla_w \hat{y}_{w_s(t,x_j)} \rangle dt \tag{9}$$

$$= \int_0^1 \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle \nabla_w \hat{y}_{w_s(t,x_i)}, \nabla_w \hat{y}_{w_s(t,x_j)} \rangle dt \tag{10}$$

$$= \int_0^1 \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \alpha_i \nabla_w \hat{y}_{w_s(t,x_i)}, \alpha_j \nabla_w \hat{y}_{w_s(t,x_j)} \rangle dt \tag{11}$$

$$\tag{12}$$

Derivation : Exact Path Kernel

$$= \int_0^1 \langle \sum_{i=1}^{n} \alpha_i \nabla_w \hat{y}_{w_s(t,x_i)}, \sum_{j=1}^{n} \alpha_j \nabla_w \hat{y}_{w_s(t,x_j)} \rangle dt \tag{13}$$

Re-ordering the sums so that their indices match, we have (14)

$$= \int_0^1 \left\| \sum_{i=1}^{n} \alpha_i \nabla_w \hat{y}_{w_s(t,x_i)} \right\|^2 dt \tag{15}$$

$$\geq 0 \tag{16}$$

□ (17)

Note that this reordering does not depend on the continuity of our mapping function $\phi_{s,t}(x_i)$.

Derivation : Exact Kernel Ensemble Representation

Theorem (Exact Kernel Ensemble Representation)

A model f_{w_N} trained using discrete steps matching the conditions of the exact path kernel has the following exact representation as an ensemble of N kernel machines:

$$f_{w_N} = KE(x) := \sum_{s=1}^{N} \sum_{i=1}^{M} a_{i,s} K_{EPK}(x, x', s) + b \tag{18}$$

where

$$a_{i,s} = -\varepsilon \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \tag{19}$$

$$b = f_{w_0}(x) \tag{20}$$

Derivation : Exact Kernel Ensemble Representation Proof

Proof. Let f_w be a differentiable function parameterized by parameters w which is trained via N forward Euler steps of fixed step size ε on a training dataset X with labels Y, with initial parameters w_0 so that there is a constant b such that for every x, $f_{w_0}(x) = b$, and weights at each step $w_s : 0 \leq s \leq N$. Let $x \in X$ be arbitrary and within the domain of f_w for every w. For the final trained state of this model f_{w_N}, let $y = f_{w_N}(x)$.
For one step of training, we consider $y_s = f_{w_s(0)}(x)$ and $y_{s+1} = f_{w_{s+1}}(x)$.
We wish to account for the change $y_{s+1} - y_s$ in terms of a gradient flow, so we must compute $\dfrac{\partial y}{dt}$ for a continuously varying parameter t. Since f is trained using forward Euler with a step size of $\varepsilon > 0$, this derivative is determined by a step of fixed size of the weights w_s to w_{s+1}.

Derivation : Exact Kernel Ensemble Representation Proof

We parameterize this step in terms of the weights:

$$\frac{dw_s(t)}{dt} = (w_{s+1} - w_s) \tag{21}$$

$$\int \frac{dw_s(t)}{dt} dt = \int (w_{s+1} - w_s) dt \tag{22}$$

$$w_s(t) = w_s + t(w_{s+1} - w_s) \tag{23}$$

$$\tag{24}$$

Since f is being trained using forward Euler, across the entire training set X we can write:

$$\frac{dw_s(t)}{dt} = -\varepsilon \nabla_w L(f_{w_s(0)}(X), y_i) = -\varepsilon \sum_{j=1}^{d} \sum_{i=1}^{M} \frac{\partial L(f_{w_s(0)}(x_i), y_i)}{\partial w_j} \tag{25}$$

Derivation : Exact Kernel Ensemble Representation Proof

Applying chain rule and the above substitution, we can write

$$\frac{d\hat{y}}{dt} = \frac{df_{w_s(t)}}{dt} = \sum_{j=1}^{d} \frac{df}{\partial w_j} \frac{\partial w_j}{dt} \tag{26}$$

$$= \sum_{j=1}^{d} \frac{df_{w_s(t)}(x)}{\partial w_j} \left(-\varepsilon \frac{\partial L(f_{w_s(0)}(X_T), Y_T)}{\partial w_j} \right) \tag{27}$$

$$= \sum_{j=1}^{d} \frac{df_{w_s(t)}(x)}{\partial w_j} \left(-\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \frac{\partial f_{w_s(0)}(x_i)}{\partial w_j} \right) \tag{28}$$

$$= -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \sum_{j=1}^{d} \frac{df_{w_s(t)}(x)}{\partial w_j} \frac{df_{w_s(0)}(x_i)}{\partial w_j} \tag{29}$$

$$= -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \nabla_w f_{w_s(t)}(x) \cdot \nabla_w f_{w_s(0)}(x_i) \tag{30}$$

Derivation : Exact Kernel Ensemble Representation Proof

Using the fundamental theorem of calculus, we can compute the change in the model's output over step s

$$y_{s+1} - y_s = \int_0^1 -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \nabla_w f_{w_s(t)}(x) \cdot \nabla_w f_{w_s(0)}(x_i) dt \quad (31)$$

$$= -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \left(\int_0^1 \nabla_w f_{w_s(t)}(x) dt \right) \cdot \nabla_w f_{w_s(0)}(x_i) \quad (32)$$

Derivation : Exact Kernel Ensemble Representation Proof

For all N training steps, we have

$$y_N = b + \sum_{s=1}^{N} y_{s+1} - y_s$$

$$y_N = b + \sum_{s=1}^{N} -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \left(\int_0^1 \nabla_w f_{w_s(t)}(x) dt \right) \cdot \nabla_w f_{w_s(0)}(x_i)$$

$$= b + \sum_{i=1}^{M} \sum_{s=1}^{N} -\varepsilon \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \int_0^1 \left\langle \nabla_w f_{w_s(t,x)}(x), \nabla_w f_{w_s(t,x_i)}(x_i) \right\rangle dt$$

$$= b + \sum_{i=1}^{M} \sum_{s=1}^{N} a_{i,s} \int_0^1 \left\langle \phi_{s,t}(x), \phi_{s,t}(x_i) \right\rangle dt$$

Since an integral of a symmetric positive semi-definite function is still symmetric and positive-definite, each step is thus represented by a kernel machine. □

Derivation : Exact Kernel Machine Representation

Theorem (Exact Kernel Machine Reduction)

Let $\nabla L(f(w_s(x), y)$ be constant across steps s, $(a_{i,s}) = (a_{i,0})$. Let the kernel across all N steps be defined as $K_{NEPK}(x, x') = \sum_{s=1}^{N} a_{i,0} K_{EPK}(x, x', s)$ Then the exact kernel ensemble representation for f_{w_N} can be reduced exactly to the kernel machine representation:

$$f_{w_N}(x) = KM(x) := b + \sum_{i=1}^{M} a_{i,0} K_{NEPK}(x, x') \tag{33}$$

which is to say that all such models, which are sheaves in the RKBS of functions integrated along discrete optimization paths in fact reduce to a sheaf in the reproducing kernel Hilbert space (RKHS).

Note on Prior Work

Constant sign loss functions have been previously studied by Chen et al. Chen et al. (2021), however the kernel that they derive for a finite-width case is of the form

$$K(x, x_i) = \int_0^T |\nabla_f L(f_t(x_i), y_i)| \langle \nabla_w f_t(x), \nabla_w f_t(x_i) \rangle dt \tag{34}$$

The summation across these terms satisfies the positive semi-definite requirement of a kernel, however the weight $|\nabla L(f_t(x_i), y_i)|$ depends on x_i which is one of the two inputs. This makes the resulting function $K(x, x_i)$ asymmetric and therefore not a kernel.

Experimental Results

Our first experiments test the kernel formulation on a dataset which can be visualized in 2d. These experiments serve as a sanity check and provide an interpretable representation of what the kernel is learning.

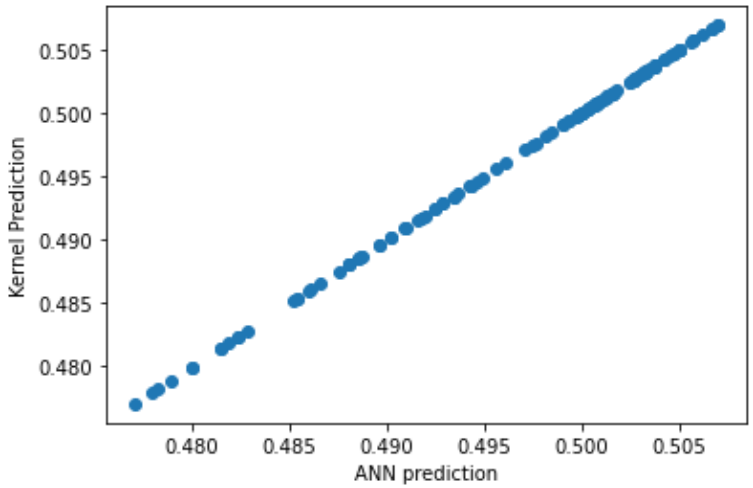
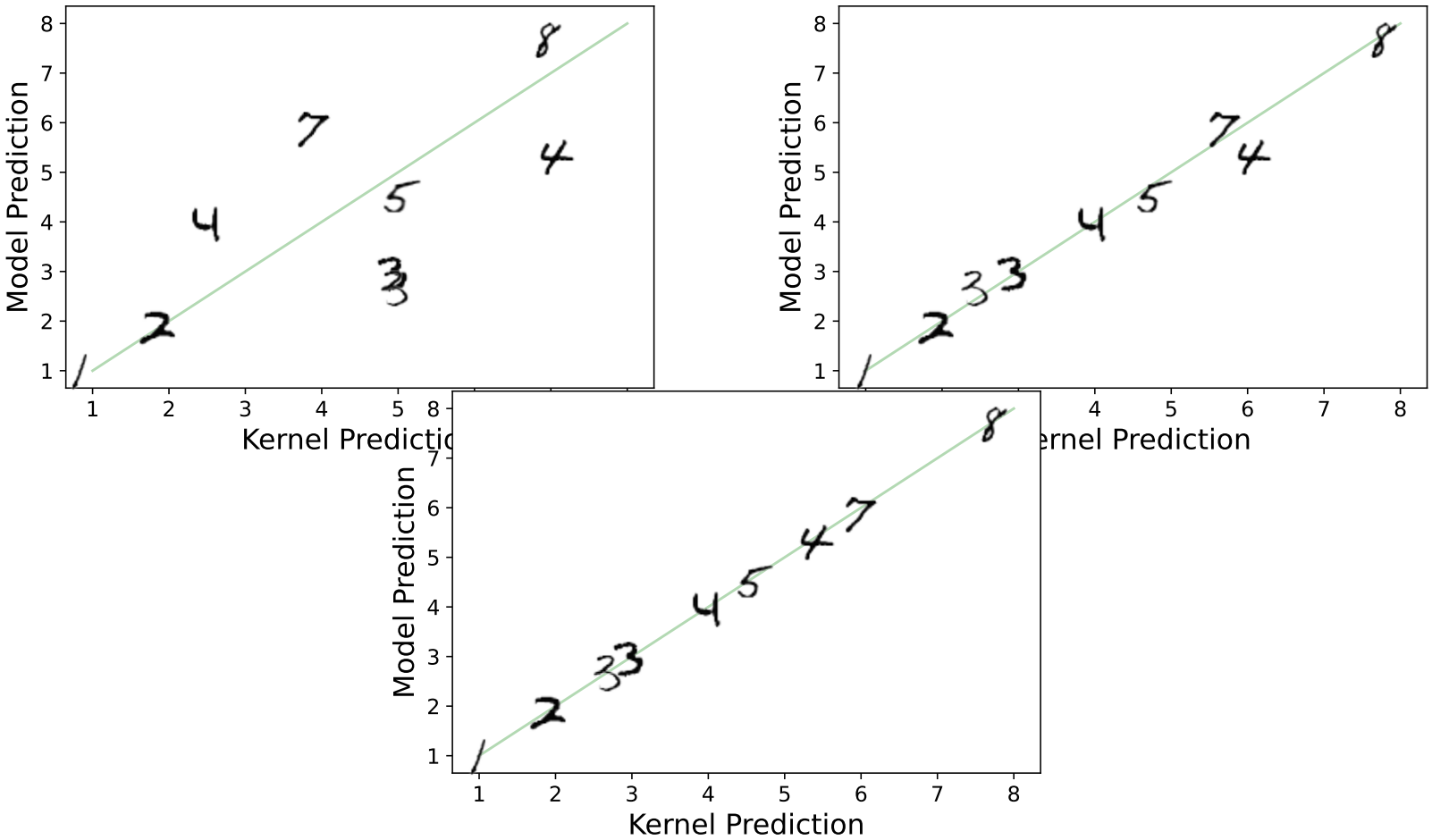
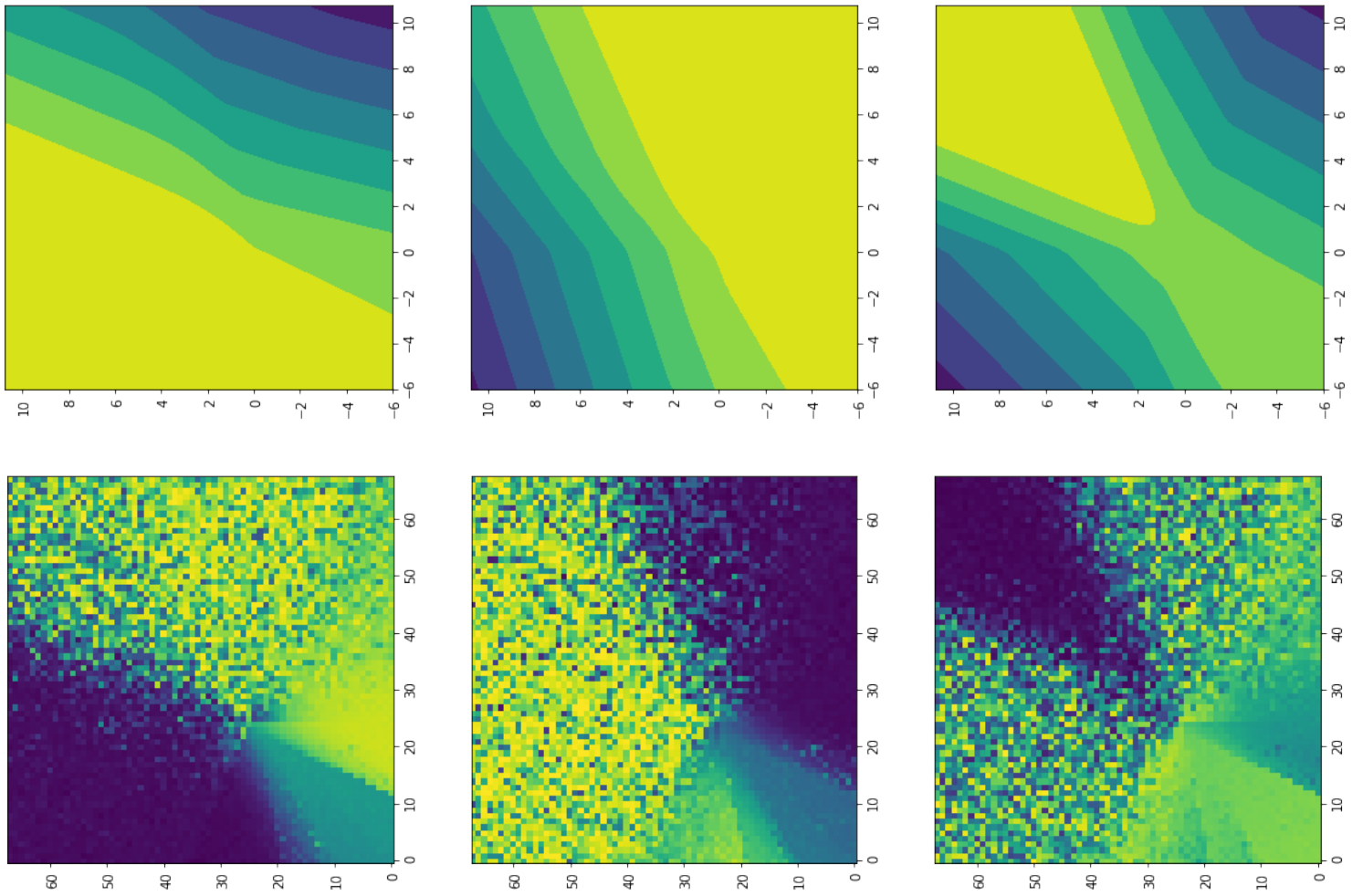


Figure: Class 1 EPK Kernel Prediction (Y) versus neural network prediction (X) for 100 test points, demonstrating extremely close agreement.

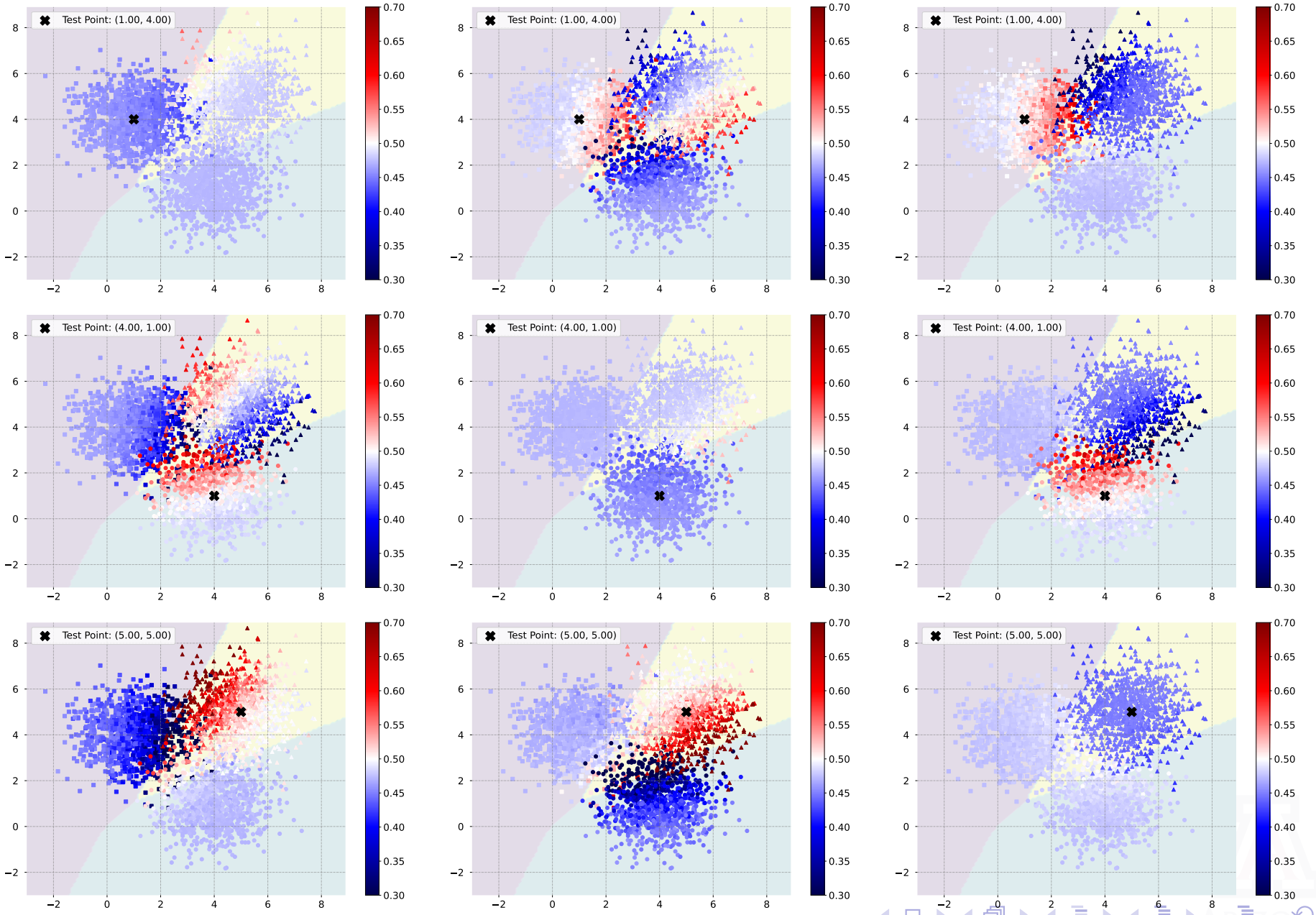
Experimental Results



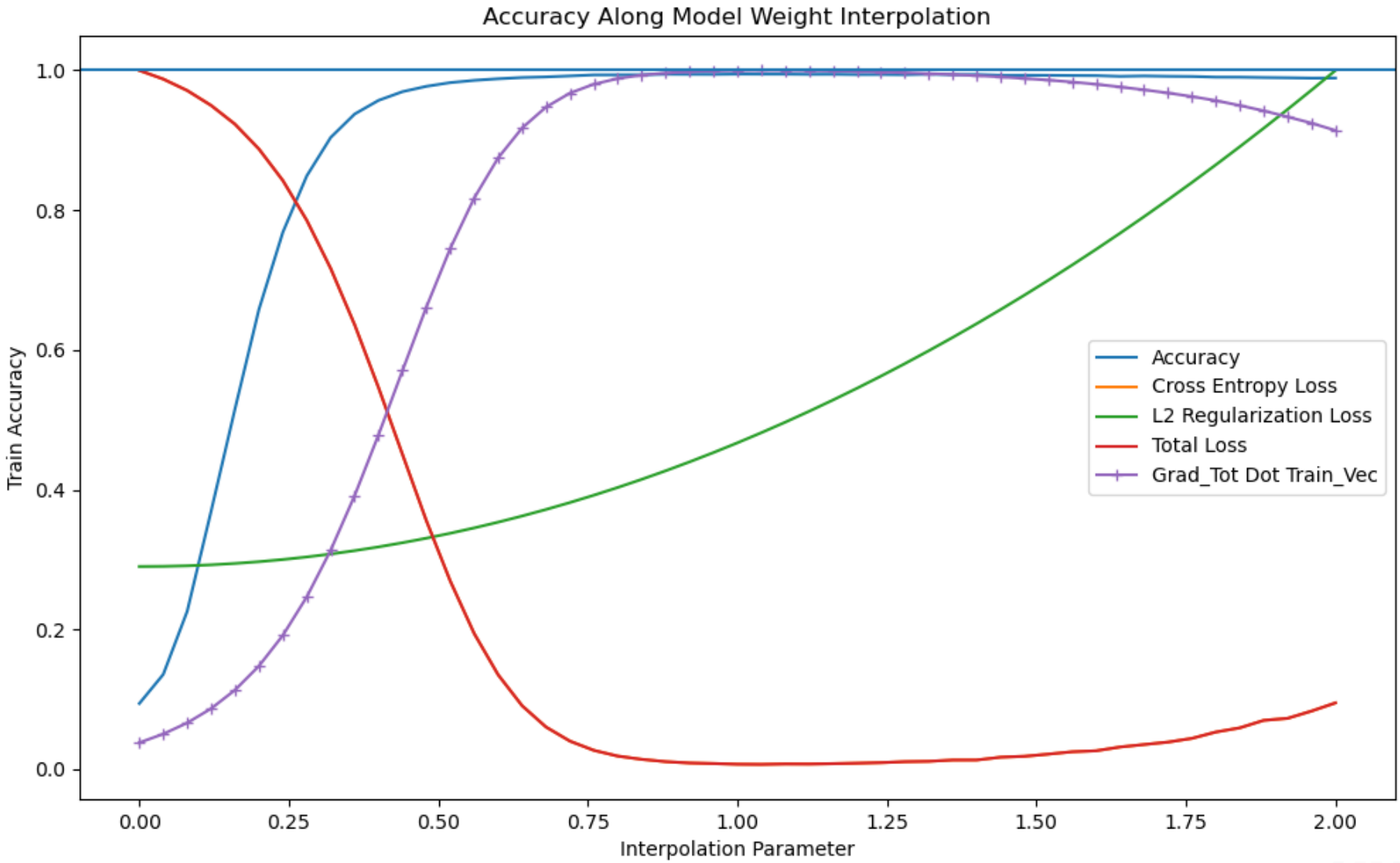
Kernel Properties



Experimental Results



Gradient Alignment During Training



Contents

Contribitions

In short, this work leverages the gEPK to:

- Introduce a theoretical framework that provides a decomposition in terms of the previously proposed EPK.
- Generalize and explain the success of recent successful methods in OOD.
- Showcase OOD using natural gEPK based decomposition of model predictions in terms of parameter gradients.
- Measure exact input variations and signal manifold dimension around arbitrary test points.

The primary contributions of this paper are theoretical in nature: establishing the exact representation theorem in Section **??** and writing several leading OOD detection methods in terms of this representation. The preliminary experimental results also support practical tasks of out-of-distribution (OOD) detection and estimating signal manifold dimension.

Generalized Exact Path Kernel (gEPK)

Theorem (Generalized Exact Path Kernel (gEPK))

Suppose $f(\cdot;\theta)$ is a differentiable parametric model with parameters $\theta \in \mathbb{R}^M$ and L is a loss function. Furthermore, suppose that f has been trained by a series $\{\theta_s\}_{s=0}^{S}$ of discrete steps composed from a sum of loss gradients for the training set $\sum_i^N \varepsilon \nabla_\theta L(f(x_i), y_i)$ on N training data X_T starting from θ_0. Then for an arbitrary test point x, the trained model prediction $f(x;\theta_S)$ can be written:

$$f(x;\theta_S) = f(x;\theta_0) + \sum_{i=1}^{N}\sum_{s=1}^{S} \varepsilon \left(\int_0^1 \varphi_{s,t}(x) dt \right) \frac{dL(f(x_i,\theta_s), y_i)}{d\hat{y}_{\theta_s(0)}} (\varphi_{s,0}(x_i)) \tag{35}$$

$$\varphi_{s,t}(x) \equiv \nabla_\theta f(x;\theta_s(t)), \tag{36}$$

$$\theta_s(t) \equiv \theta_s(0) + t(\theta_{s+1}(0) - \theta_s(0)), \text{ and} \tag{37}$$

$$\hat{y}_{\theta_s(0)} \equiv f(x;\theta_s(0)). \tag{38}$$

Examining Cutting Edge OOD with gEPK

ReAct, DICE, ASH, and VRA : Along with other recent work (Sun et al., 2021; Sun and Li, 2022; Xu et al., 2023a), some of the cutting edge for OOD as of early 2023 involves activation truncation techniques. A prediction, $f(x, \theta)$, is computed forward through the network. This yields a vector of activations, $A(x, \theta_{\text{represent}})$, in the representation layer of the network. This representation is then pruned down to the p^{th} percentile by setting any activations below that percentile to zero. That means that this truncation is picking a representation for which $\left\langle \nabla_\theta f(x, \theta_{\text{represent}}), \frac{dL(\hat{y}(x_i), y_i)}{d\hat{y}} \nabla_\theta f(x_i, \theta_{\text{represent}}) \right\rangle$ is high for many training data, x_i. This is effectively a projection onto the parameter tangent space of the training data with the highest variation. This may explain some part of the performance advantage of these method.

Examining Cutting Edge OOD with gEPK

GradOrth: Behpour et al. (2023) explicitly create a reference basis from parameter gradients on training data for comparison:

$$\nabla_\theta L(x, y) = (\theta x - y)x^T = \Omega x^T \tag{39}$$

Treating Ω as an error vector, they prove that all variation of the output must be within the span of the x^T over the training set. They then pick a small subset of the training data and record its activations $R^L_{ID} = [x_1, x_2, ..., x_n]$ over which they compute the SVD, $U^L_{ID}\Sigma^L_{ID}(V^L_{ID})^T = R^L_{ID}$. This representation is then truncated to k principal components according to a threshold ϵ_{th} such that

$$\left\| U^L_{ID}\Sigma^L_{ID,k}(V^L_{ID})^T \right\|^2_F \geq \epsilon_{\text{th}} \|R^L_I D\|^2_F. \tag{40}$$

This basis $S^L = (U^L_I D)_k$ is now treated as the reference space onto which test points' final layer gradients can be projected. Their score is

$$O(x) = (\nabla_{\theta_L}\mathcal{L}(f(x, \theta_L), y))S^L(S^L)^T \tag{41}$$

gEPK for OOD

The gEPK provides a more general spanning result immediately. Indeed, the network's prediction can only be influenced by $\{\varphi_{s,0}(x_i)\}$ for the training data batched for each step, s. Loss in the gEPK only appears for the training points. This means that $\varphi_{s,t}(x) = \nabla_\theta f(x; \theta_s(t))$ can be computed for any test point without need for a label. Then all parameter gradients must live in:

$$\text{span}\left(\left\{\frac{dL(\hat{y}_i, y_i)}{d\hat{y}_i}\varphi_{s,0}(x_i) : x_i \in X_T, 0 \leq s \leq S\right\}\right). \tag{42}$$

We can see that most, if not all, of the above methods can be represented by some set of averaging assumptions on the representation provided by the gEPK.

OOD is enabled by Parameter Gradients

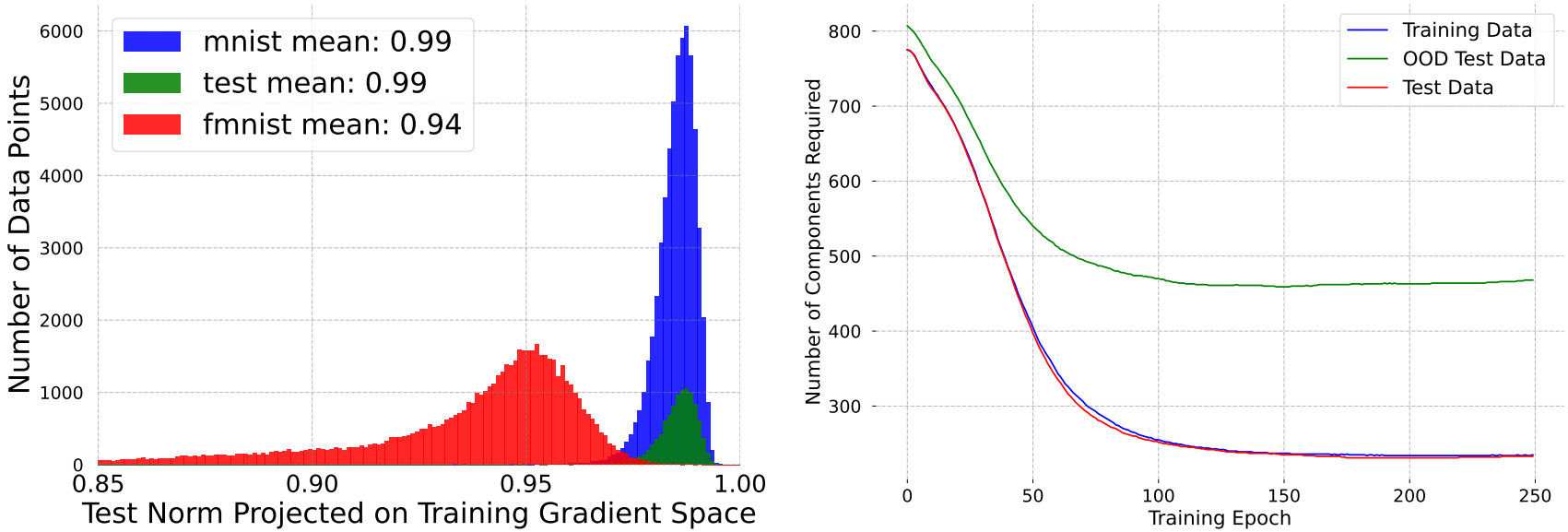


Figure: OOD detection using difference in training vs. test gradients. As the purpose of this paper is not to develop state of the art OOD detection methods, a comparison with recent benchmarks is not provided. Instead, a proof of concept that the gEPK can perform OOD detection is given. Left histogram shows norms of vectors projected onto the gradient weight space defined by the gEPK on MNIST and FMNIST. Right plot shows the number of components required to explain 95% variation in weight space across training for a toy problem (three Gaussian distributions embedded in 100 dimensions).

Measuring Dimensionality with Parameter Gradient Space

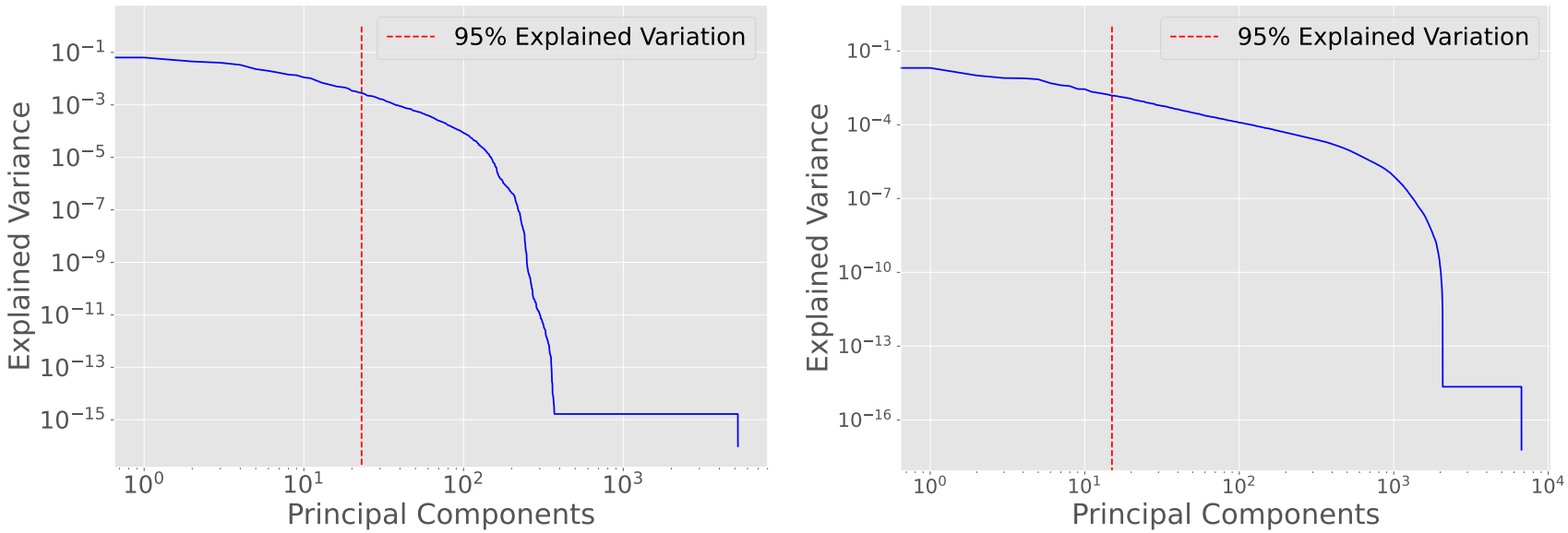


Figure: Explained Variance Ratio of parameter gradients. Left: MNIST, Right: CIFAR. 95% of variation can be explained with a relatively low number of components in both cases.

Computing Training Tangent (Dual) Space Basis

Given a test point x, we can evaluate its subspace by taking, for each x_i:

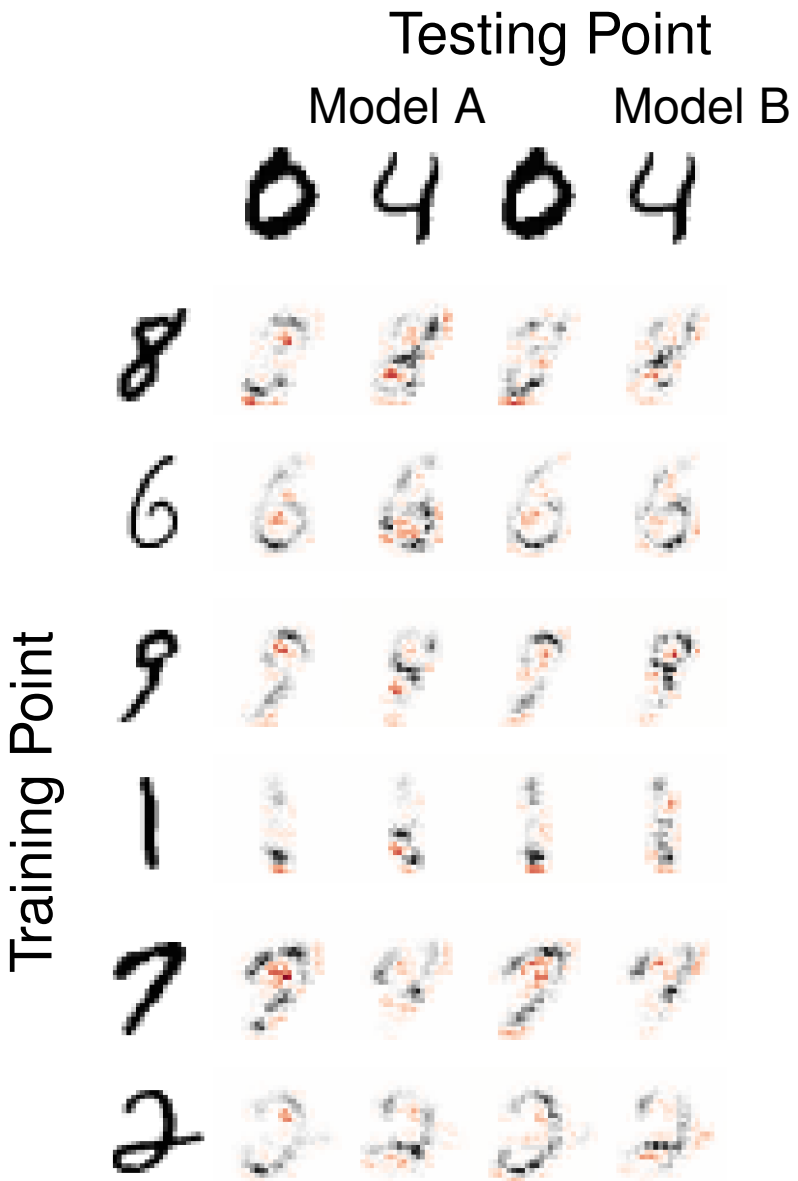
$$\frac{df(x,\theta_{\text{trained}})}{dx_j} = \frac{df(x,\theta_0(0))}{dx_j} + \sum_i \sum_s \int_0^1 \frac{d\left(\varphi_{s,t}(x)\frac{dL(x_i,y_i)}{df(x_i,\theta_s(0))}\varphi_{s,0}(x_i)\right)}{dx_j} d \tag{43}$$

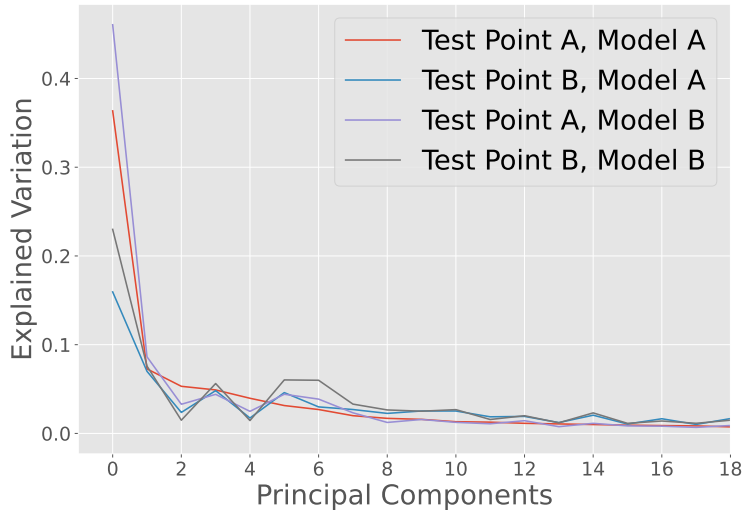
$$= \sum_i \sum_s \int_0^1 \varphi_{s,t}(x)dt \left(\frac{d^2L(x_i,y_i)}{df(x_i,\theta_s(0))dx_j}\varphi_{s,0}(x_i) + \frac{dL(x_i,y_i)}{df(x_i,\theta_s(0))} \tag{44}$$

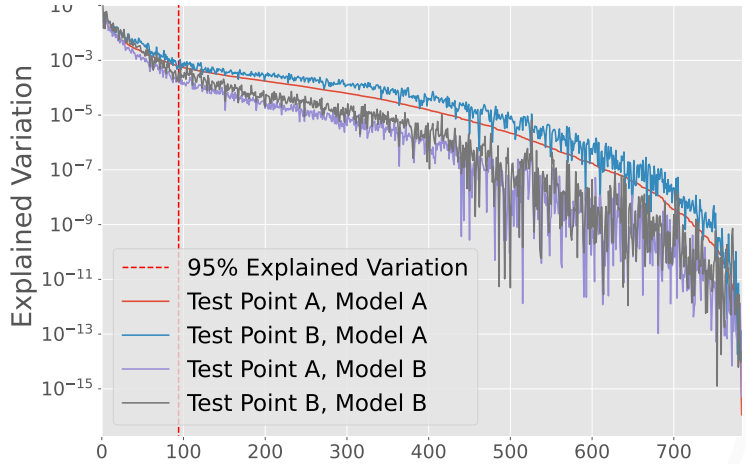
We can see that these gradients will be zero except when $i = j$, thus we may summarize these gradients as a matrix (tensor in the multi-class case), G, with

$$G_j = \sum_s \int_0^1 \varphi_{s,t}(x)dt \left(\frac{d^2L(x_i,y_i)}{df(x_i,\theta_s(0))dx_j}\varphi_{s,0}(x_i) + \frac{dL(x_i,y_i)}{df(x_i,\theta_s(0))}\frac{d\varphi_{s,0}(x_i)}{dx_j}\right) \tag{45}$$

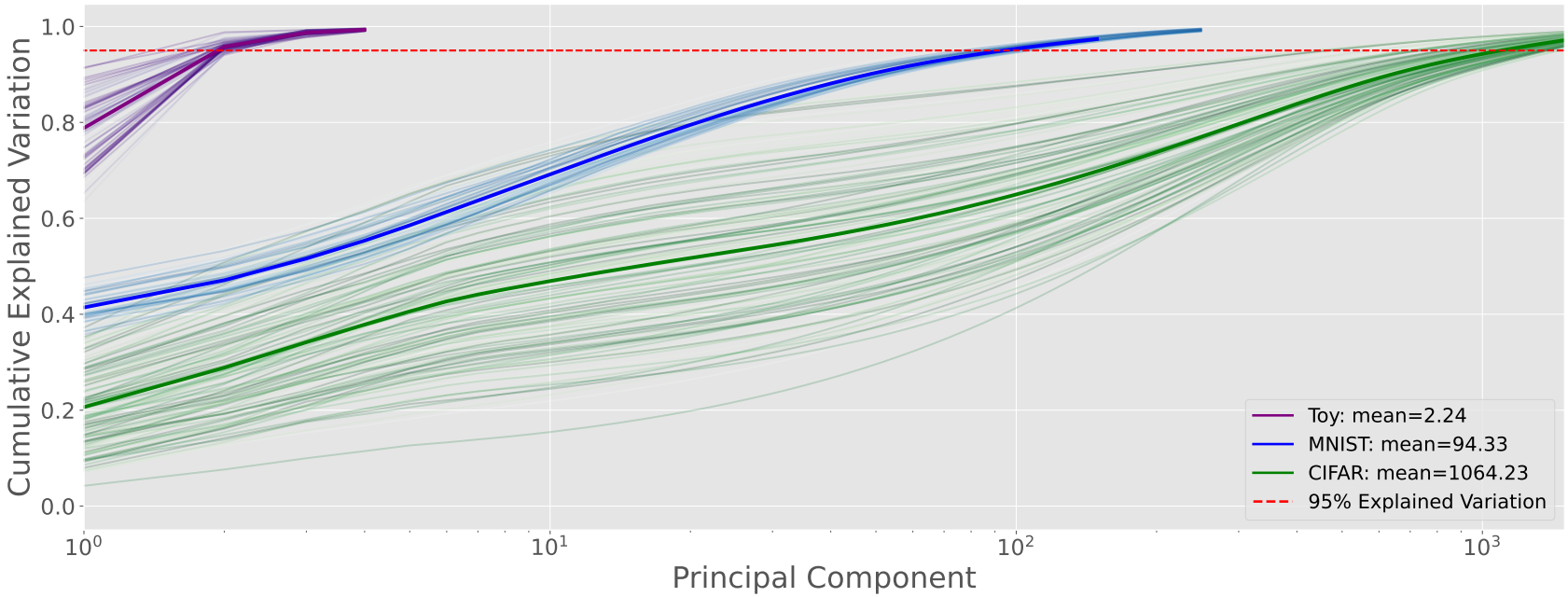
Measuring Dimensionality







Measuring Dimensionality with Training Tangent Dual Space



Reducing the gEPK with SVD

Take a model, $f(x, \theta)$, which satisfies the necessary conditions for expression as:

$$f(x, \theta_{\text{tr}}) = f(x, \theta_0(0)) + \sum_s \int_0^1 \left\langle \varphi_{s,t}(x), \sum_i \left(\frac{dL(x_i, y_i)}{df(x_i, \theta_s(0))} \varphi_{s,0}(x_i) \right) \right\rangle dt \tag{46}$$

$$\varphi_{s,t}(x) = \nabla_\theta f(x, \theta_s(t)) \tag{47}$$

Then we will let the RHS be rows of a matrix $L \cdot A$ where $L = \text{diag}(\frac{dL(x_i, y_i)}{df(x_i, \theta_s(0))})$ and each row $A_i = \varphi_{s,0}(x_i)$. Then we can compute $U \Sigma V^T = A$. We will truncate V and rewrite

$$f(x, \theta_{\text{tr}}) = f(x, \theta_0(0)) + \sum_s \int_0^1 \left\langle \varphi_{s,t}(x), \sum_k a_k \sigma_k V_k \right\rangle dt \tag{48}$$

(49)

Where a_k is computed from the loss gradients and σ_k is the k^{th} singular value.

Contents

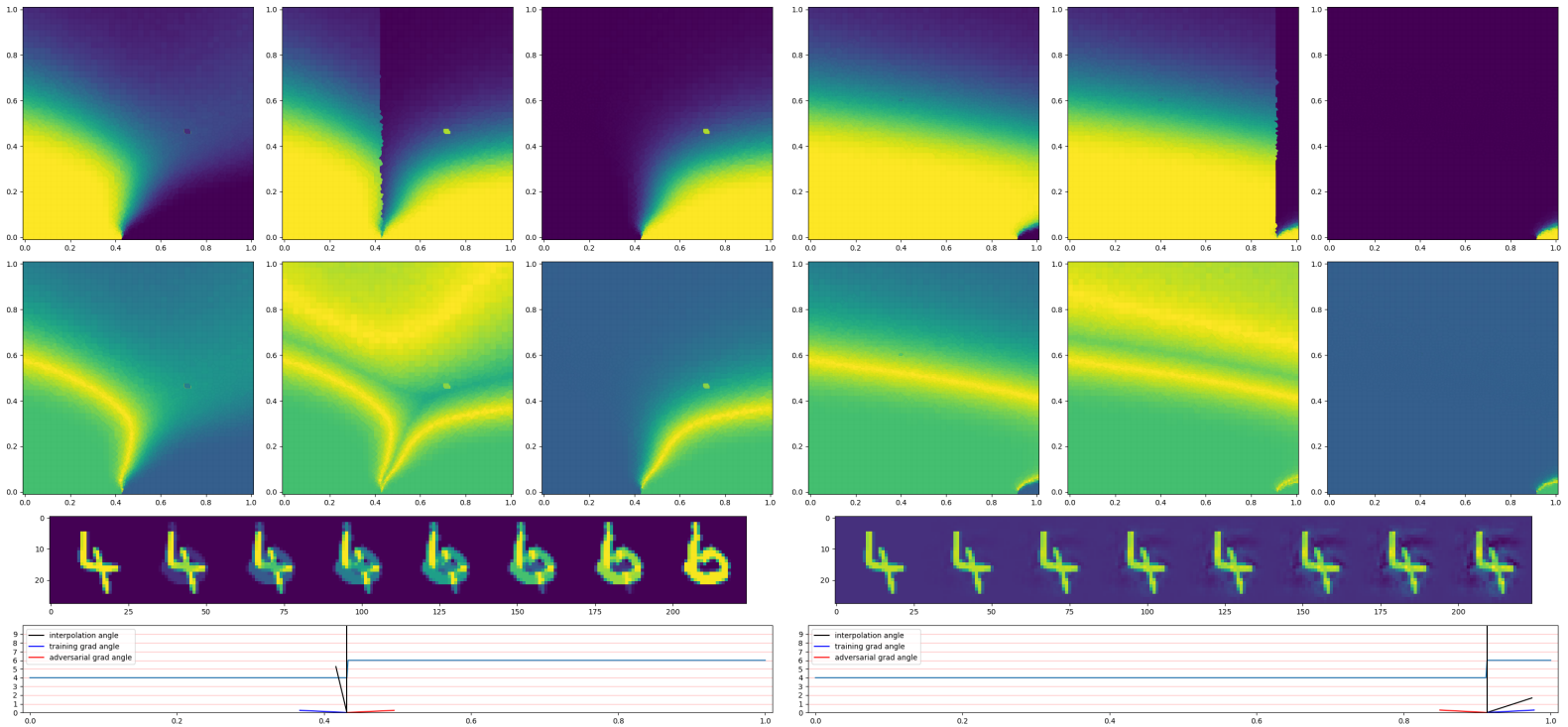
Conclusion : Next Direct Results

- In addition to making the above replacement with SVD, random features may be a cheap alternative with similar performance.
- A more general spectral form is possible, using features which span multiple training steps. Random Fourier Features may be a nice cheap way to accomplish this.
- Decomposition of (adversarial) input gradients according to the principal components from the training inputs (or principal components using the SVD formulation above).

$$f(x,\theta_F) = f(x,\theta_0) + \sum_i \sum_s \int_0^1 \langle \nabla_\theta f(x,\theta_s(t)), f(x_i,\theta_s(t)) \rangle dt \quad (50)$$

$$\frac{df(x,\theta_F)}{dx} = \frac{df(x,\theta_0)}{dx} + \sum_i \sum_s \int_0^1 \langle \nabla_\theta \frac{df(x,\theta_s(t))}{dx}, f(x_i,\theta_s(t)) \rangle dt. \quad (51)$$

Conclusion : Next Applications



Decision boundary curvature can now be analyzed directly using decompositions based on the gEPK.

Conclusion

Today, you have heard about

- Geometric observations about adversarial attacks using a novel soft spatial metric.
- A brand new method for representing neural networks in terms of blinear maps which admits useful decompositions and provides a robust theoretical framework for analysis.
- Analysis explaining cutting edge OOD detection using this kernel representation framework and showing that these techniques are in general subsets or reductions of the subspaces revealed by this representation.
- Connections between geometric properties and robustness which can be explored by applying these techniques.

Citations I

Behpour, S., Doan, T., Li, X., He, W., Gou, L., and Ren, L. (2023). Gradorth: A simple yet efficient out-of-distribution detection with orthogonal projection of gradients. *CoRR*, abs/2308.00310.

Chen, Y., Huang, W., Nguyen, L., and Weng, T.-W. (2021). On the equivalence between neural network and support vector machine. *Advances in Neural Information Processing Systems*, 34:23478–23490.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE.

Domingos, P. (2020a). Every model learned by gradient descent is approximately a kernel machine. *arXiv preprint arXiv:2012.00152*.

Domingos, P. (2020b). Every model learned by gradient descent is approximately a kernel machine. *CoRR*, abs/2012.00152.

Ghojogh, B., Ghodsi, A., Karray, F., and Crowley, M. (2021). Reproducing kernel hilbert space, mercer's theorem, eigenfunctions, nyström method, and use of kernels in machine learning: Tutorial and survey.

Citations II

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. (2018). Adversarial spheres. *arXiv preprint arXiv:1801.02774*.

He, W., Li, B., and Song, D. (2018). Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32:125–136.

Incudini, M., Grossi, M., Mandarino, A., Vallecorsa, S., Pierro, A. D., and Windridge, D. (2022). The quantum path kernel: a generalized quantum neural tangent kernel for deep quantum machine learning.

Khoury, M. and Hadfield-Menell, D. (2018). On the geometry of adversarial examples. *CoRR*, abs/1811.00525.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25.

Citations III

Rasmussen, C. E., Williams, C. K., et al. (2006). *Gaussian processes for machine learning*, volume 1. Springer.

Shafahi, A., Huang, W. R., Studer, C., Feizi, S., and Goldstein, T. (2018). Are adversarial examples inevitable? *CoRR*, abs/1809.02104.

Shamir, A., Melamed, O., and BenShmuel, O. (2021). The dimpled manifold model of adversarial examples in machine learning. *CoRR*, abs/2106.10151.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, Y., Guo, C., and Li, Y. (2021). React: Out-of-distribution detection with rectified activations. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 144–157.

Citations IV

Sun, Y. and Li, Y. (2022). DICE: leveraging sparsification for out-of-distribution detection. In Avidan, S., Brostow, G. J., Cissé, M., Farinella, G. M., and Hassner, T., editors, *Computer Vision - ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXIV*, volume 13684 of *Lecture Notes in Computer Science*, pages 691–708. Springer.

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

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

Tjeng, V., Xiao, K., and Tedrake, R. (2017). Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*.

Citations V

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy. *stat*, 1050:11.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. (2020). Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*.

Xu, M., Lian, Z., Liu, B., and Tao, J. (2023a). Vra: Variational rectified activation for out-of-distribution detection.

Xu, Y., Sun, Y., Goldblum, M., Goldstein, T., and Huang, F. (2023b). Exploring and exploiting decision boundary dynamics for adversarial robustness.