DARTMOUTH

Increasing Robustness of CNNs using Kernel PCA

Paul Lintilhac and Uttam Rao

Abstract

Machine Learning Classifiers may be used for security-critical applications, which make them a target for adversaries who may benefit from compromising the integrities of these systems. Devising defenses for ML in these adversarial environments is important for ensuring that we can continue to trust the security of ML systems.

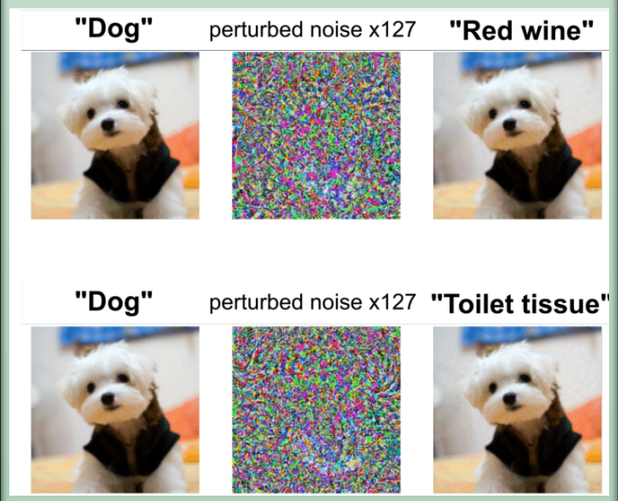
We introduce two important generalizations of the Principal Components Analysis (PCA) defense introduced by Bhagoji et. al [1]. First, we introduce the “recons-retrain” defense. We first project the top principal components back onto the original 2D image space before retraining, allowing us to implement a 2D Convolutional Neural Network (CNN). Unlike the retrain defense, which underperforms relative to the undefended model for small perturbations due to this architectural limitation, our retrain-recons defense outperforms the undefended model for all magnitudes of adversarial perturbations.

The other main contribution of our paper is to implement a non-linear version of the PCA transformation using a Radial Basis Function (RBF) kernel. Our results show that our PCA defense with an RBF kernel significantly outperforms the linear PCA defense when tested against a defense-aware fast-gradient attack.

Introduction

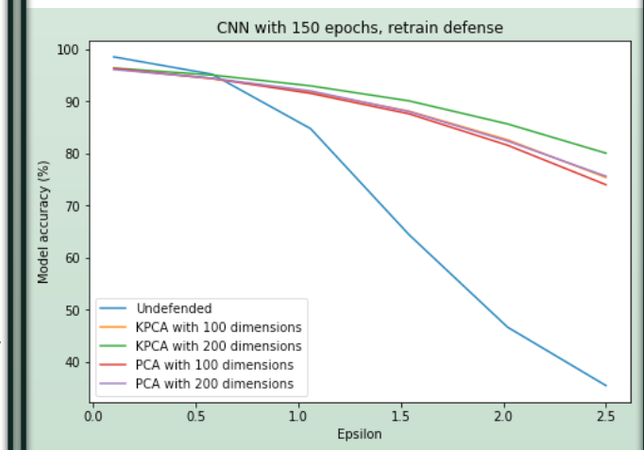
We focus solely on un-targeted, gradient-based evasion attacks on ML classifiers, in which an adversary strategically adds perturbations to the test data to cause the model to misclassify the input. These perturbed inputs are known as adversarial examples. While we tested our results against several gradient-based and optimization-based attacks, we found the results to be very similar across different attacks. Our results here are based on the fast-gradient method, a white-box attack which perturbs the input by a specified distortion level ϵ in the direction of the gradient of the loss function, e.g.

$$x'(\epsilon) = x + \epsilon \frac{\nabla_X J(\mathbf{x}, y, \theta)}{||\nabla_X J(\mathbf{x}, y, \theta)||}$$



Motivation

Since the discovery of these vulnerabilities in ML classifiers, several defenses have been proposed, including adversarial training, defensive distillation, and various linear data transformations as a pre-processing step. Of these, PCA-based data transformations have shown particular promise due to their simplicity, generality as a defensive measure that can be applied upstream of any model, and their lack of trade-offs with other desirable ML properties. Bhagoji et. al. showed that data transformations based on PCA act as a form of regularization. Regularization techniques, while coming at a slight cost in terms of accuracy and utility of ML classifiers, do not cause a major trade-off with privacy (unlike adversarial training and gradient masking, they do not improve accuracy of membership inference attacks)



The retrain defense underperforms the undefended model for small perturbations due to its architectural limitations

However the PCA defense was shown to be less effective against CNNs, which are an important state-of-the-art ML that needs to be addressed in order for this defense to be widely used. One of the main reasons for this is that Bhagoji’s PCA-retrain defense retrains the model on the top principal components projected onto the 1D eigenspace, and therefore only permits the use of a 1D convolutional layer for the classifier on transformed images. Bhagoji et al. also only consider linear data transformations and do not investigate nonlinear transformations. Our motivation for this paper was to address these limitations by investigating non linear transformations as a defense against evasion attacks, with specific regard to CNNs.

The reason we chose the the radial basis function (RBF) as our kernel for kernel-pca is that we wanted to explicitly leverage the “manifold hypothesis”: that a likely explanation for the existence of adversarial examples is that the input data usually lies on a low-dimensional manifold within the input space. While adversarial examples may be close to the input data in absolute terms, they may lie relatively far from the actual centers of mass of the input data. By using distances to the centers of mass as our basis during PCA, we project the data onto this manifold.

Methodology

Our approach uses Kernel Principal Components analysis with an RBF (Gaussian) kernel in order to apply a non-linear data transformation to the data both before training the model and before classifying new inputs. The first step in the process involves diagonalizing the covariance matrix:

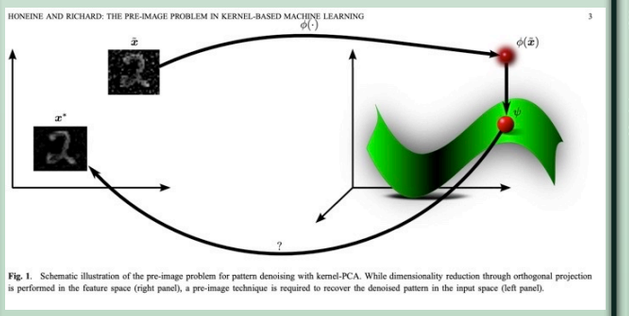
$$K = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i)\phi(x_i)^T$$

Where $\phi(x_i)$ is a "feature map" for the positive semi-definite symmetric kernel defined in some Reproducing Kernel Hilbert Space. For our defense, we chose the RBF kernel, s.t. for any two points in our space we have $K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = exp(-\gamma ||x_i - x_j||^2)$.

- We found that a value of .0325 worked well for our defense on the MNIST data set.
- When restricted to the N points in our input space, we can define the empirical kernel feature map as the similarity between a given point x and all other points in the space:

$$\phi_N(x) = \mathbf{K}^{-\frac{1}{2}}[\mathbf{K}(\mathbf{x}_1, \mathbf{x}), \mathbf{K}(\mathbf{x}_2, \mathbf{x}), \ldots, \mathbf{K}(\mathbf{x}_\mathbf{N}, \mathbf{x})]$$

- By taking only the top principal components, we are approximating this closeness to each other point in the training set by characterizing the point’s distance to the primary centers of mass that comprise the training data.

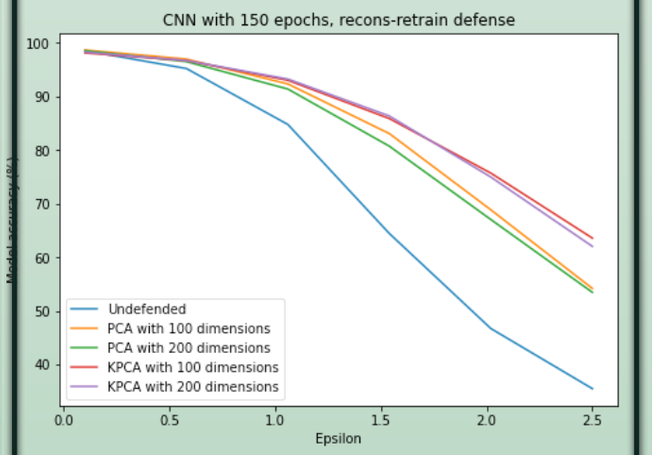


Before re-training our model, we must invert the projection of each point onto the kernel’s eigenspace back into the input space. With this additional inversion step, we can view the algorithm as a de-noising algorithm, in the sense that it reconstructs the original data point using a smaller number of principal components. While this may be trivial for regular PCA, complications arise when we are using nonlinear dimension reduction with kernel PCA. In particular, the inverse for the empirical kernel map is not guaranteed to be unique. Various methods have been proposed to address this problem both in the general case and for the RBF kernel. The one we implemented for our KPCA-Recons algorithm is the kernel ridge regression as described Bakir et. al. [2] and implemented using the scikit-learn package in python.

Experimental Setup

All experimentation of kernel PCA was done using the MNIST dataset. The MNIST dataset consists of 28 x 28 pixel grayscale images, which are handwritten representations of digits 0-9. This dataset consists of 60,000 training examples and 10,000 test examples.

First, we analyzed the effects of kernel PCA on the retrain defense of Bhagoji et. al. As shown in the plot of accuracy vs adversarial perturbation to the left, we found that while the KPCA-retrain defense outperforms PCA-retrain, both defenses underperform the undefended models for small values of epsilon. This is to be expected, as the retrain defense is limited to implementing a 1D convolutional neural net in the eigenspace. Next, we tested the combination of our two contributions: the KPCA-retrain-recons defense, and compared it to the PCA-retrain-recons defense, as well as to the undefended models. **We found that, unlike the retrain defense, our retrain-recons defenses perform better than the undefended model even for small perturbations. In addition, we found that the kernel-pca version of this defense significantly outperforms linear PCA**. Since small perturbations are difficult to detect for both humans and automated novelty detection systems, we believe our defense provides a higher overall utility.



References

[1] Chawin Sitawarin Arjun Nitin Bhagoji, Daniel Collina and Praten Mittal. 2017. Enhancing Robustness of Machine Learning Systems via Data Transformations (2017) arXiv:arXiv:1704:02654

[2] Gökhan Bakir, J. Weston, Bernhard Schölkopf, S. Torun, and L. Saul. 2004 Learning to find Pre-Images. *Advances in Neural Information Processing Systems, 449-456 (2004) (03 2004)*