Adversarially Robust One-class Novelty Detection

IEEE T-PAMI 2022

Shao-Yuan Lo
Poojan Oza
Vishal M. Patel

Johns Hopkins University
Recall: One-class Novelty Detection

• One-class novelty detection model is trained with examples of a particular class and is asked to identify whether a query example belongs to the same known class.

• Example:
  • Known class (normal data): 8
  • Novel classes (anomalous data): 0-7 & 9 (the rest of classes)
Recall: One-class Novelty Detection

• Most recent advances are based on the autoencoder architecture.
• Given an autoencoder that learns the distribution of the known class, we expect that the normal data are reconstructed accurately while the anomalous data are not.
Attacking One-class Novelty Detection

• How to generate adversarial examples against a novelty detector?
• If a test example is normal, maximize the reconstruction error.
• If a test example is anomalous, minimize the reconstruction error.

Normal data

Training data

Adversarial examples

Reconstructions (expected)

High error

Anomalous

Low error

Normal
Goal: Adversarially Robust Novelty Detection

• Novelty detectors are **vulnerable** to adversarial attacks.

• Adversarially robust method specifically designed for novelty detectors is needed.

• A **new** research problem.
Observation: Generalizability

• Unique property: Preference for poor generalization of reconstruction ability.

• However, autoencoders have good generalizability.

Normal data
Training data

Test data

Reconstructions

Low error

Low error

?
Observation: Feature Denoising

• Adversarial perturbations can be removed in the feature domain.

[Xie et al. CVPR’19]
Our Solution

• **Observations**: Generalizability and Feature Denoising.

• **Assumption**: One can *largely* manipulate the latent space of a novelty detector to remove adversaries to a great extent, and this would not hurt the model capacity but *helps* if in a proper way.

• **Solution**: Learning *principal latent space*.
PCA Rephrased

• $h()$ computes the **mean vector** and the first $k$ **principal components** of the given data collection $X$:

$$h(X, k) : X \rightarrow \{\mu, \tilde{U}\}$$

• $f()$ performs the forward PCA:

$$f(X; \mu, \tilde{U}) = (X - \mu 1^\top)\tilde{U}$$

$$X_{pca} = f(X; \mu, \tilde{U})$$

• $g()$ performs the inverse PCA:

$$g(X_{pca}; \mu, \tilde{U}) = X_{pca} \tilde{U}^\top + \mu 1^\top$$

$$\hat{X} = g(f(X; \mu, \tilde{U}); \mu, \tilde{U})$$
Cascade PCA Process

- **Vector-PCA** performs PCA on the **vector** dimension.
- **Spatial-PCA** performs PCA on the **spatial** dimension.
Cascade PCA Process

• Step 1: **Forward Vector-PCA**, i.e., $f_V()$

\[ Z_{adv} \in \mathbb{R}^{s \times v} \quad \xrightarrow{\text{Latent space}} \quad Z_V \in \mathbb{R}^{s \times 1} \quad \text{Vector-PCA space} \]

\[ \{\mu_V, \tilde{U}_V\} = h_V(Z, k_V = 1) \]

\[ Z_V = f_V(Z; \mu_V, \tilde{U}_V) \]
Cascade PCA Process

• Step 2: Forward Spatial-PCA, i.e., $fs()$

$Z_V \in \mathbb{R}^{s \times 1} \rightarrow Z_S \in \mathbb{R}^{k_S \times 1}$

Vector-PCA space \quad Spatial-PCA space

\[
\{\mu_S, \tilde{U}_S\} = h_S(Z_V^T, k_S)
\]

\[
Z_S^T = f_S(Z_V^T; \mu_S, \tilde{U}_S)
\]
Cascade PCA Process

- **Step 3:** Inverse Spatial-PCA, i.e., $g_S()$
- **Step 4:** Inverse Vector-PCA, i.e., $g_V()$

\[ Z_S \in \mathbb{R}^{k_S \times 1} \quad \rightarrow \quad Z_{pls} \in \mathbb{R}^{s \times v} \]

*Spatial-PCA space* \hspace{1cm} *Principal latent space*

\[
\hat{Z}_V^T = g_S(Z_S^T; \mu_S, \tilde{U}_S) \\
Z_{plr} = g_V(\hat{Z}_V; \mu_V, \tilde{U}_V)
\]
Learning Principal Latent Components

- **Principal latent components:**
  \[ \{ \mu_V, \tilde{U}_V, \mu_S, \tilde{U}_S \} \]

- **Training time:** Train along with the network weights by exponential moving average (EMA).
  \[ \{ \mu_V^t, \tilde{U}_V^t \} = \{ \mu_V^{t-1}, \tilde{U}_V^{t-1} \} + \eta_V (h_V(Z^t) - \{ \mu_V^{t-1}, \tilde{U}_V^{t-1} \}) \]
  \[ \{ \mu_S^t, \tilde{U}_S^t \} = \{ \mu_S^{t-1}, \tilde{U}_S^{t-1} \} + \eta_S (h_S(Z^t) - \{ \mu_S^{t-1}, \tilde{U}_S^{t-1} \}) \]

- **Inference time:** Perform the cascade PCA process with the fixed and well-trained parameters:
  \[ \{ \mu_V^*, \tilde{U}_V^*, \mu_S^*, \tilde{U}_S^* \} \]
Defense Mechanism

• **Vector-PCA** replaces the perturbed latent vectors with the clean principal latent vector.

• **Spatial-PCA** removes the remaining perturbations on the Vector-PCA map.
Defense Mechanism

• Combine **adversarial training**.
• The proposed PrincipaLS process can robustify **any** AE-based novelty detectors.
  • AE, VAE, AAE, ALOCC (CVPR’18), GPND (NeurIPS’18), etc.
Results

• Evaluation metric: mean of AUROC

• PrincipaLS is effective on 5 datasets against 6 attacks for 7 novelty detection methods.

| Dataset | Defense | Clean | FGSM [11] | PGD [27] | MI-FGSM [36] | MultiAdv [37] | AF [38] | Black-box [47] | Average |
|---------|---------|-------|-----------|----------|---------------|---------------|---------|----------------|---------|
|         | No Defense | 0.964 | 0.350 | 0.051 | 0.022 | 0.170 | 0.014 | 0.790 | 0.337 |
| MNIST [48] | PGD-AT | 0.961 | 0.604 | 0.357 | 0.369 | 0.444 | 0.155 | 0.691 | 0.512 |
|         | FD | 0.963 | 0.612 | 0.366 | 0.379 | 0.453 | 0.142 | 0.700 | 0.516 |
|         | SAT | 0.947 | 0.527 | 0.295 | 0.306 | 0.370 | 0.142 | 0.652 | 0.463 |
|         | RotNet-AT | 0.967 | 0.598 | 0.333 | 0.333 | 0.424 | 0.101 | 0.695 | 0.493 |
|         | SOAP | 0.940 | 0.686 | 0.504 | 0.506 | 0.433 | 0.088 | 0.863 | 0.574 |
|         | APAE | 0.925 | 0.428 | 0.104 | 0.105 | 0.251 | 0.022 | 0.730 | 0.366 |
|         | PrincipaLS (ours) | **0.973** | **0.812** | **0.706** | **0.707** | **0.725** | **0.636** | **0.866** | **0.775** |
|         | No Defense | 0.523 | 0.204 | 0.034 | 0.038 | 0.006 | 0.000 | 0.220 | 0.146 |
| SHTech [52] | PGD-AT | 0.527 | 0.217 | 0.168 | 0.154 | 0.100 | 0.000 | 0.221 | 0.198 |
|         | FD | 0.528 | 0.226 | 0.189 | 0.181 | 0.132 | 0.002 | 0.229 | 0.212 |
|         | SAT | 0.529 | 0.184 | 0.110 | 0.092 | 0.040 | 0.000 | 0.199 | 0.165 |
|         | RotNet-AT | 0.516 | 0.220 | 0.163 | 0.158 | 0.113 | 0.000 | 0.229 | 0.200 |
|         | SOAP | 0.432 | 0.024 | 0.000 | 0.000 | 0.181 | 0.002 | 0.202 | 0.120 |
|         | APAE | 0.510 | 0.215 | 0.048 | 0.050 | 0.011 | 0.000 | 0.207 | 0.149 |
|         | PrincipaLS (ours) | 0.498 | 0.274 | 0.223 | 0.217 | 0.175 | 0.051 | **0.308** | **0.249** |
Analysis

• PrincipaLS reconstructs **every** input example to the known class (digit 2).
Analysis

- (a) No Defense under clean data
- (b) No Defense under PGD attack
- (c) PGD-AT under PGD attack
- (d) PrincipaLS under PGD attack

PrincipaLS enlarges the reconstruction errors of anomalous data to a great extent.