mathematics

MDPI

Article

Latent-Insensitive Autoencoders for Anomaly Detection

Muhammad S. Battikh [1] and Artem A. Lenskiy [2,*]

[1] Systems and Computer Engineering Department, Al-Azhar University, Cairo 11651, Egypt; muhammad.saeed.batikh@azhar.edu.eg
[2] School of Computing, The Australian National University, Canberra 2601, Australia
* Correspondence: Artem.Lenskiy@anu.edu.au

Abstract: Reconstruction-based approaches to anomaly detection tend to fall short when applied to complex datasets with target classes that possess high inter-class variance. Similar to the idea of self-taught learning used in transfer learning, many domains are rich with similar unlabeled datasets that could be leveraged as a proxy for out-of-distribution samples. In this paper we introduce the latent-insensitive autoencoder (LIS-AE) where unlabeled data from a similar domain are utilized as negative examples to shape the latent layer (bottleneck) of a regular autoencoder such that it is only capable of reconstructing one task. We provide theoretical justification for the proposed training process and loss functions along with an extensive ablation study highlighting important aspects of our model. We test our model in multiple anomaly detection settings presenting quantitative and qualitative analysis showcasing the significant performance improvement of our model for anomaly detection tasks.

Keywords: anomaly detection; autoencoders; one-class classification; principal components analysis; self-taught learning; negative examples

Citation: Battikh, M.S.; Lenskiy, A.A. Latent-Insensitive Autoencoders for Anomaly Detection. *Mathematics* **2022**, *10*, 112. https://doi.org/10.3390/math10010112

Academic Editor: Bo-Hao Chen

Received: 14 November 2021
Accepted: 21 December 2021
Published: 30 December 2021

1. Introduction

Anomaly detection is a classical machine learning field which is concerned with the identification of in-distribution and out-of-distribution samples that finds applications in numerous fields [1,2]. Unlike traditional multi-label classification where the goal is to find decision boundaries between classes present in a given dataset, the goal of anomaly detection is to find one-versus-all boundaries for classes that are not in the dataset, which is significantly more challenging compared to standard classification. Autoencoders [3] have been used extensively for anomaly detection [1,4,5] under the assumption that reconstruction error incurred by anomalies is higher than that of normal samples [6,7]. However, it has been observed that this assumption might not hold as standard autoencoders might generalize so well even for anomalies [7,8]. In practice, this issue becomes more relevant in two important settings, namely, when the normal data are relatively complex they require high latent dimensions for *good* reconstitution, and when anomalies share similar compositional features and are from a close domain to the normal data [9].

To mitigate these issues we present latent-insensitive autoencoder (LIS-AE), a new class of autoencoders where the training process is carried out in two phases. In the first phase, the model simply reconstructs the input as a standard autoencoder, and in the second phase the entire model except the latent layer is "*frozen*". We then train the model in such a way that it forces the latent layer to only keep reconstructing the target task. We use the concept of a negative dataset from one-class classification [10] whereby an auxiliary dataset of non-examples from similar domains is used as a proxy for out-of-distribution samples. In Figure 1, We change the training objective such that the autoencoder keeps its low reconstruction error for the target dataset while pushing the error of the negative dataset to exceed a certain value. In some cases, minimizing and maximizing the reconstruction loss at the same time becomes contradictory, especially for negative classes that are very similar to the target class. To resolve this issue we introduce another variant with modified

first phase loss that ensures that the input of the latent layer is linearly separable for positive and negative examples during the second phase. This linearly separable variant (LinSep-LIS-AE) almost always performs better than directly using LIS-AE. Details of architecture, training process, theoretical analysis, and experiments are discussed in detail in the following sections.

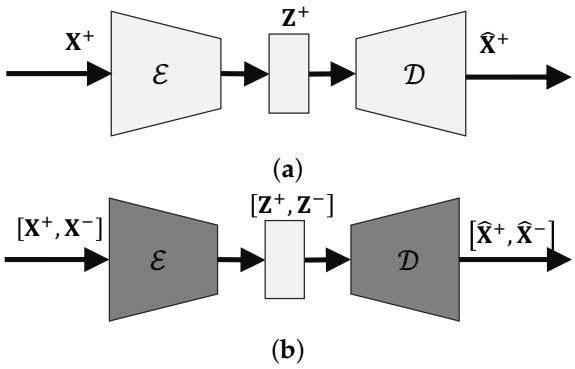

Figure 1. The two phases of training. (**a**) The first diagram shows the feature extraction phase. (**b**) The second phase starts by freezing the model except the latent layer. Negative examples $\mathbf{x}^-$ are used to fine-tune the latent layer to be only responsive to $\mathbf{x}^+$.

2. Related Work

Many reconstruction-based anomaly detection approaches have been proposed starting with classical methods such as PCA [11]. Robust-PCA mitigates the issue of outlier sensitivity in PCA by decomposing the data matrix into a sum of two low-rank and sparse matrices using nuclear norm and L_1 norms as convex relaxation for the objective loss [12]. Autoencoders address the issue of PCA only considering linear relations in feature-space by introducing non-linearities benefiting from multiple layers of representations [13]. We elaborate further on the shortcomings of PCA and autoencoders in the theoretical section and use that to motivate our approach.

Other methods try to improve on base autoencoders by endowing the latent code with particular properties. In the case of VAE [14], it does so by having the latent code follow a prior distribution (usually normal) which also allows sampling from the decoder. However, in the context of anomaly detection, it introduces scaling issues since minimizing KL-divergence for the high latent dimensions required for complex tasks is quite challenging. Another approach is the replicator neural network (RepNN) [15] which is an autoencoder with a staircase activation function positioned on the output of the bottleneck layer (latent Layer). This is mainly used in order to quantize the latent code into a number of discrete values which also aids in forming clusters [16]. Unfortunately, a discrete staircase function is non-differentiable which prevents learning via backpropagation. Instead, a differentiable approximation involving the sum of N hyperbolic tangent functions `tanh` was introduced in place of the otherwise non-differentiable discrete staircase function. However, as discussed in [17], despite the theoretical appeal of having a quantized latent code via smooth approximation, in practice, having such an activation function makes it significantly difficult for the gradient signal to flow. We also note that when increasing the number of levels using the aforementioned N, `tanh` sum approximation presents a significant overhead during training and testing since N activation functions have to be computed for each batch and, moreover, it suffered from scaling issues similar to that of VAE.

Another approach is *memorizing normality* of a given dataset using a memory-augmented autoencoder [8]. This approach limits the effective space of possible latent codes by constructing a memory module that takes in the output of the encoder as an address and passes to the decoder the most relevant memory items from a stored reservoir of prototypical patterns that have been learned during training.

Other non-reconstruction-based approaches include one-class classification which is tightly connected to anomaly detection in the sense that both problems are concerned with finding one-versus-all boundaries. One-class SVM is a variation of the classical SVM algorithm [18] where the objective is to find a hyper-plane that best separates samples from outliers [19]. Support vector data description (SVDD) [20] tries to find a circumscribing hyper-sphere that contains all samples while having optimal margins for outliers. It is worth noting that for kernels where $k(x, x) = 1$ such as RBF and Laplacian, OC-SVM and OC-SVDD learn identical decision functions [21]. To address the lack of representation learning and bad computational scalability of OC-SVM and OC-SVDD, deep SVDD (OC-DSVDD) employs a deep neural network that learns useful representation while mapping outputs to a hyper-sphere of minimum volume [22]. However, due to its sole reliance on optimizing for minimum volume, this approach is prone to hyper-sphere collapse which leads to finding uninformative features [23].

Other approaches have been proposed where an auxiliary dataset of non-examples (negative dataset) is drawn from similar domains as a proxy for the otherwise intractable complement for the target class. In [10], a collection called the "*Universum*", allows learning useful representation to the domain of the problem via maximizing the number of contradictions on an equivalence class. Similar to OC-DSVDD, [23] leverages a labeled dataset from a close domain to fine-tune two pre-trained CNNs in order to learn new *good* features. The goodness of these features is quantified by the compactness (inter-class variance) for the target class and descriptiveness (cross-entropy) for the labeled dataset. Despite avoiding hyper-sphere collapse and outperforming OC-SVDD, this approach requires two pre-trained neural networks and a large labeled dataset along with the target dataset. Another approach that also makes use of a large auxiliary dataset is outlier exposure (OE) [24], which is a supervised approach that trains a standard neural network classifier while exposing it to a diverse set of non-examples on which the output of the classifier is optimized to follow a uniform distribution using another cross-entropy loss.

3. Proposed Method

3.1. Architecture

An undercomplete deep autoencoder is a type of unsupervised feed-forward neural network for learning a lower-dimensional feature representation of a particular dataset via reconstructing the input back at the output layer. To prevent autoencoders from converging to an uninformative solution such as the identity mapping, a bottleneck layer with output $\mathbf{z}$ such that its dimension is less than the dimension of the input $\mathbf{x}$. The forward pass is computed as:

$$\mathbf{s} = \mathscr{E}(\mathbf{x}),$$
$$\mathbf{z} = \mathscr{Z}(\mathbf{s}),$$
$$\hat{\mathbf{x}} = \mathscr{D}(\mathbf{z}),$$

where $\mathbf{x}$ is the input, $\mathscr{Z}$ is the bottleneck layer, $\mathscr{E}$ and $\mathscr{D}$ are convolutional neural networks representing the encoder and the decoder modules, respectively. Typically, such models are trained to minimize the $\mathcal{L}_2$-norm of the difference between the input and the reconstructed output $\|\hat{\mathbf{x}} - \mathbf{x}\|_2$. As previously discussed, the choice of the activation function of $\mathbf{z}$ plays an important rule in anomaly detection. Activation functions that quantize the latent code or encourage forming clusters are preferable. In our experiments, we find that confining the latent code to have values between $[-1, 1]$ with a `tanh` activation function as we maximize the loss over the negative dataset during the latent-shaping phase has a regularizing effect. We also note that unbounded activation functions such as ReLU tend to have poor performance.

3.2. Terminology

Positive Dataset ($\mathcal{D}^+$): This is the dataset that contains the normal class(es), for example, the *plane* class from CIFAR-10.

Negative Dataset ($\mathcal{D}^-$): This is a secondary unlabeled dataset containing negative examples from a similar domain as $\mathcal{D}^+$. The choice of $\mathcal{D}^-$ depends on $\mathcal{D}^+$. For example, if $\mathcal{D}^+$ is the digit 0 from MNIST, $\mathcal{D}^-$ might be random strokes or another dataset with similar features such as Omniglot [25]. It is important to note that the model should not be tested on $\mathcal{D}^-$ since this violates the assumption of not knowing anomalies.
Anomaly Dataset ($\mathcal{D}^a$): This is a test dataset that contains classes that are neither in $\mathcal{D}^+$ nor in $\mathcal{D}^-$.
Feature Extraction Phase: This is the first phase of training. The model is simply trained to reconstruct its input.
Latent-Shaping Phase: This is the second phase of training. The encoder and decoder networks are frozen and only the latent layer is active.

3.3. Training for Anomaly Detection

Given a dataset $\mathcal{D}^+$ and a negative dataset $\mathcal{D}^-$ from a similar domain to $\mathcal{D}^+$, we divide the training process into two phases; the first phase is reconstructing samples from $\mathcal{D}^+$ by minimizing the loss function $\mathcal{L} = \|\hat{\mathbf{x}}^+ - \mathbf{x}^+\|_2$ until convergence, where $\mathbf{x}^+$ is the input drawn from $\mathcal{D}^+$ and $\hat{\mathbf{x}}^+$ is the output of the autoencoder. In the second phase, we freeze the model except for the latent layer and minimize the following loss function:

$$\mathcal{L} = \|\hat{\mathbf{x}}^+ - \mathbf{x}^+\|_2 + \beta\|\gamma - \|\hat{\mathbf{x}}^- - \mathbf{x}^-\|_2\|_2 \tag{1}$$

where $\mathbf{x}^-$ is a sampled batch from $\mathcal{D}^-$, $\hat{\mathbf{x}}^-$ is its reconstruction, β is a hyper-parameter that controls the effect of the two parts of the loss function and γ is another hyper-parameter indicating that we are satisfied if the reconstruction error $\|\hat{\mathbf{x}}^- - \mathbf{x}^-\|_2$ of the negative dataset exceeds a certain value. Details of training are provided in Algorithm 1.

Algorithm 1 Anomaly Detection Training.

Input: Positive ($\mathcal{D}^+$) and Negative ($\mathcal{D}^-$) datasets
// $\mathscr{E}$: Encoder, $\mathscr{Z}$: Latent Layer, $\mathscr{D}$: Decoder
Output: Trained model
// Feature extraction phase
// Sample mini batches from $\mathcal{D}^+$
for $\mathbf{x}^+ \in \mathcal{D}^+$ ***until convergence*** **do**
  $\hat{\mathbf{x}}^+ = \mathcal{D}(\mathscr{Z}(\mathscr{E}(\mathbf{x}^+))$
  $\mathcal{L} = \|\hat{\mathbf{x}}^+ - \mathbf{x}^+\|_2$
  // backpropagation step
  Minimize $\mathcal{L}$
end
FreezeEncoder()
FreezeDecoder()
// Latent-shaping phase
// Sample mini batches from $(\mathcal{D}^+, \mathcal{D}^-)$
for $(\mathbf{x}^+, \mathbf{x}^-) \in (\mathcal{D}^+, \mathcal{D}^-)$ ***until convergence*** **do**
  $[\mathbf{z}^+, \mathbf{z}^-] = \mathscr{Z}(\mathscr{E}([\mathbf{x}^+, \mathbf{x}^-]))$
  $[\hat{\mathbf{x}}^+, \hat{\mathbf{x}}^-] = \mathcal{D}([\mathbf{z}^+, \mathbf{z}^-])$
  $\mathcal{L} = \|\hat{\mathbf{x}}^+ - \mathbf{x}^+\|_2 + \beta\|\gamma - \|\hat{\mathbf{x}}^- - \mathbf{x}^-\|_2\|_2$
  // backpropagation step
  Minimize $\mathcal{L}$
end

3.4. Predicting Anomalies

We use reconstruction error $\mathcal{L}(x) = \|\hat{\mathbf{x}} - \mathbf{x}\|_2$ to distinguish between anomalies and normal data where $\mathbf{x}$ is the test sample and $\hat{\mathbf{x}}$ is the reconstructed output. More specifically, we set a threshold α such that if $\mathcal{L}(\mathbf{x}) > \alpha$, the output is considered to be anomalous.

4. Theoretical Justification

4.1. Formulation

In this section, we present theoretical justification for the reasoning behind selective freezing and the second phase loss function. We would like to show that the process described in algorithm 1 implies that the second stage reconstruction loss for a latent-insensitive autoencoder ($\mathcal{L}_{LIS}$) remains equivalent to the reconstruction loss of a standard autoencoder ($\mathcal{L}_{AE}$) for normal (positive) samples but is larger for anomalies. More formally, under certain assumptions for negative dataset ($\mathcal{D}^-$), $\mathcal{L}_{LIS}(\mathbf{x}^+) = \mathcal{L}_{AE}(\mathbf{x}^+)$ and $\mathcal{L}_{LIS}(\mathbf{x}^a) \geq \mathcal{L}_{AE}(\mathbf{x}^a)$ where $\mathbf{x}^a$ is an anomalous sample.

From optimality of autoencoders [13], we know that in the absence of any non-linear activation functions, a linear autoencoder corresponds to singular value decomposition (SVD); henceforth, we use SVD interchangeably with linear autoencoders. Given an $m \times n$ data matrix $\mathbf{X}^+$, we decompose $\mathbf{R}^m$ into $\mathbf{X}^{||} \oplus \mathbf{X}^{\perp}$, where $\mathbf{X}^{||} := Col(X^+)$ and $\mathbf{X}^{\perp}$ is its orthogonal complement $Null(\mathbf{X}^{+^T})$.

We further decompose $\mathbf{X}^+$ using SVD:

$$\mathbf{X}^+ = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and $\mathbf{\Sigma}$ is a diagonal matrix such that $\mathbf{\Sigma} = [\sigma_1 \ ... \ \sigma_{rank} | 0]$. However, in practice it is rarely separated this neatly, especially when dealing with a large number of samples of a high-dimensional dataset; therefore, we resort to reduced-SVD where we take the first r columns of $\mathbf{U}$ with the caveat that the choice of r is a hyper-parameter.

$$\mathbf{U} = \begin{bmatrix} | & & | & | & & | \\ \mathbf{u}_1 & \dots & \mathbf{u}_r & \mathbf{u}_{r+1} & \dots & \mathbf{u}_n \\ | & & | & | & & | \end{bmatrix}$$

The $\mathbf{U}$ matrix can be divided thus: $\mathbf{U} = [\mathbf{U}_r | \mathbf{U}_c]$, and from the Eckart–Young low-rank approximation theorem [26], columns of $\mathbf{U}_r \approx Basis(\mathbf{X}^{||})$ and columns of $\mathbf{U}_c \approx Basis(\mathbf{X}^{\perp})$.

A linear autoencoder with an r-dimensional latent layer is equivalent to the following transform:

$$\hat{\mathbf{x}} = \mathbf{U}_r\mathbf{U}_r^T\mathbf{x}$$

where $\mathbf{U}_r$ and $\mathbf{U}_r^T$ represent the decoder and the encoder, respectively. Furthermore, any data point $\mathbf{x} \in \mathbf{R}^m$ can be represented as $\mathbf{x} = \mathbf{U}_r\mathbf{z} + \mathbf{U}_c\mathbf{c}$, where $\mathbf{c}$ and $\mathbf{z}$ are $(m - r)$ and r-dimensional real vectors. By orthonormality, we have the following identities: $\mathbf{U}_r^T\mathbf{U}_r = \mathbf{I}_r$ and $\mathbf{U}_r^T\mathbf{U}_c = \mathbf{0}$, where $\mathbf{I}_r$ is an r-identity matrix. As a shorthand, we write $\|.\|$ instead of $\|.\|_2^2$. Using these two identities, we rewrite the reconstruction loss $\|\hat{\mathbf{x}} - \mathbf{x}\|$ as the following:

$$\mathbf{U}_r\mathbf{U}_r^T\mathbf{x} = \mathbf{U}_r\mathbf{U}_r^T[\mathbf{U}_r\mathbf{z} + \mathbf{U}_c\mathbf{c}] = \mathbf{U}_r[\mathbf{U}_r^T\mathbf{U}_r\mathbf{z} + \mathbf{U}_r^T\mathbf{U}_c\mathbf{c}] = \mathbf{U}_r\mathbf{z}$$
$$\mathcal{L}_{AE}(\mathbf{x}) = \|\mathbf{U}_r\mathbf{U}_r^T\mathbf{x} - \mathbf{x}\| = \|\mathbf{U}_r\mathbf{z} - \mathbf{U}_r\mathbf{z} - \mathbf{U}_c\mathbf{c}\| = \|\mathbf{U}_c\mathbf{c}\| = \|\mathbf{c}\|$$

We note that the loss function is agnostic to the nature of $\mathbf{c}$ and is only concerned with its magnitude. The assumption for anomaly detection under this setting is that $\|\mathbf{c}^a\| > \|\mathbf{c}^+\|$, where $\mathbf{c}^a$ and $\mathbf{c}^+$ correspond to orthogonal components for anomalies and positive data, respectively. We posit that while this agnosticism is desirable for potential generality, it is not optimal for anomaly detection; hence, we modify the loss score to depend on the nature of $\mathbf{c}$:

$$\mathcal{L}_{LIS}(\mathbf{x}) = \|\mathbf{B}^T\mathbf{c}\| + \|\mathbf{c}\|$$

where $\mathbf{B}$ is an $r \times (m - r)$ matrix such that the loss is small for normal data but large for anomalies. In other words, we want $\|\mathbf{B}^T\mathbf{c}^+\| = 0$ and $\|\mathbf{B}^T\mathbf{c}^a\|$ to be large.

We define $\mathbf{C}^{||}$:= orthonormal basis for $Col(\mathbf{C}^+)$ and $\mathbf{C}^{\perp}$:= orthonormal basis for $Null(\mathbf{C}^{+^T})$, where $\mathbf{C}^+$ is the matrix of all positive orthogonal components $\mathbf{c}^+$. We decom-

pose $\mathbf{C}^{\perp}$ further into $\mathbf{C}^{-\perp}$ and $\mathbf{C}^{o\perp}$ where columns of $\mathbf{C}^{-\perp}$ are the basis of $Col(\mathbf{C}^{-})$ that are not in $\mathbf{C}^{||}$ and columns of $\mathbf{C}^{o\perp}$ are the remaining columns of $\mathbf{C}^{\perp}$.

Since $\mathbf{R}^{m-r} = Col(\mathbf{C}^{+}) \oplus Col(\mathbf{C}^{-\perp}) \oplus Col(\mathbf{C}^{o\perp})$, any $\mathbf{c} \in \mathbf{R}^{m-r}$ can be written as $\mathbf{c} = \mathbf{C}^{||}\mathbf{p} + \mathbf{C}^{-\perp}\mathbf{q} + \mathbf{C}^{o\perp}\mathbf{s}$, where $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{s}$ are real vectors.

Despite the fact that we do not have access to $\mathbf{c}^{a}$, we can utilize other negative examples from a similar domain and use $\mathbf{c}^{-}$ as a proxy for $\mathbf{c}^{a}$. Since $\mathbf{c}^{a} = \mathbf{C}^{||}\mathbf{p} + \mathbf{C}^{-\perp}\mathbf{q} + \mathbf{C}^{o\perp}\mathbf{s}$, maximizing $\|\mathbf{B}^{T}\mathbf{C}^{-\perp}\|$ implies maximizing $\|\mathbf{B}^{T}\mathbf{c}^{a}\|$ assuming that $\|\mathbf{C}^{-\perp}\mathbf{q}\| \neq 0$. The later assumption hinges on the fact that $\mathbf{X}^{-}$ is from a *similar* domain. Therefore, we end up with the goal of finding $\mathbf{B}$ such that $\|\mathbf{B}^{T}\mathbf{c}^{+}\| = 0$ and $\|\mathbf{B}^{T}\mathbf{c}^{-}\|$ is large.

$$
\begin{aligned}
\hat{\mathbf{B}} &= \texttt{argmin}(\|\mathbf{B}^{T}\mathbf{c}^{+}\| - \beta\|\mathbf{B}^{T}\mathbf{c}^{-}\|) \\
&\qquad\qquad \text{where } \beta \text{ controls the importance of the second term.} \\
&= \texttt{argmin}(\|\mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{+}\| - \beta\|\mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{-}\|) \\
&\qquad\qquad \text{since orthonormal transformations preserve the dot product.} \\
&= \texttt{argmin}(\|\mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{+}\| + \| - \mathbf{U}_c\mathbf{c}^{+}\| - \beta\|\mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{-}\| - \beta\| - \mathbf{U}_c\mathbf{c}^{-}\|) \\
&\qquad\qquad \text{since adding constant to } \texttt{argmin} \text{ does not affect the objective.} \\
&= \texttt{argmin}(\|\mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{+} - \mathbf{U}_c\mathbf{c}^{+}\| - \beta\|\mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{-} - \mathbf{U}_c\mathbf{c}^{-}\|) \\
&= \texttt{argmin}(\|\mathbf{U}_r\mathbf{z}^{+} + \mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{+} - \mathbf{U}_r\mathbf{z}^{+} - \mathbf{U}_c\mathbf{c}^{+}\| \\
&\qquad - \beta\|\mathbf{U}_r\mathbf{z}^{+} + \mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{-} - \mathbf{U}_r\mathbf{z}^{-} - \mathbf{U}_c\mathbf{c}^{-}\|) \\
&= \texttt{argmin}(\|\mathbf{U}_r\mathbf{z}^{+} + \mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{+} - \mathbf{x}^{+}\| - \beta\|\mathbf{U}_r\mathbf{z}^{-} + \mathbf{U}_r\mathbf{B}^{T}\mathbf{c}^{-} - \mathbf{x}^{-}\|) \\
&= \texttt{argmin}(\|\mathbf{U}_r\mathbf{U}_r^{T}\mathbf{x}^{+} + \mathbf{U}_r\mathbf{B}^{T}\mathbf{U}_c^{T}\mathbf{x}^{+} - \mathbf{x}^{+}\| \\
&\qquad - \beta\|\mathbf{U}_r\mathbf{U}_r^{T}\mathbf{x}^{-} + \mathbf{U}_r\mathbf{B}^{T}\mathbf{U}_c^{T}\mathbf{x}^{-} - \mathbf{x}^{-}\|) \\
&= \texttt{argmin}(\|\mathbf{U}_r(\mathbf{U}_r^{T} + \mathbf{B}^{T}\mathbf{U}_c^{T})\mathbf{x}^{+} - \mathbf{x}^{+}\| \\
&\qquad - \beta\|\mathbf{U}_r(\mathbf{U}_r^{T} + \mathbf{B}^{T}\mathbf{U}_c^{T})\mathbf{x}^{-} - \mathbf{x}^{-}\|) \\
\mathbf{E} &:= \mathbf{U}_r + \mathbf{U}_c\mathbf{B} \\
\hat{\mathbf{E}} &= \texttt{argmin}(\|\mathbf{U}_r\mathbf{E}^{T}\mathbf{x}^{+} - \mathbf{x}^{+}\| - \beta\|\mathbf{U}_r\mathbf{E}^{T}\mathbf{x}^{-} - \mathbf{x}^{-}\|)
\end{aligned}
$$

In practice, we cannot maximize $\|\mathbf{U}_r\mathbf{E}^{T}\mathbf{x}^{-} - \mathbf{x}^{-}\|$ indefinitely and we are satisfied if it reaches a certain large γ:

$$\hat{\mathbf{E}} = \texttt{argmin}(\|\mathbf{U}_r\mathbf{E}^{T}\mathbf{x}^{+} - \mathbf{x}^{+}\| + \beta\|\gamma - \|\mathbf{U}_r\mathbf{E}^{T}\mathbf{x}^{-} - \mathbf{x}^{-}\|\|)$$

We notice that in order for this to work, the decoder $\mathbf{U}_r$ has to be known and remain fixed (frozen). This suggests a two-phase training where we first compute the decoder and encoder networks, and in the second phase the decoder is fixed while the encoder $\mathbf{E}^{T}$ is modified using the new loss. In Figure 2, a linear version of LIS-AE is trained on digit-8 from MNIST with Omniglot as a negative dataset. We perform orthogonal decomposition on each input by projecting it onto digit-8 subspace to obtain its projection and orthogonal vectors. We then feed each vector separately to a regular linear AE and linear LIS-AE. We observe that the regular autoencoder outputs zero images for the orthogonal part of each sample regardless of the class it belongs to. However, in the case of LIS-AE, it behaves differently for a normal class than for anomalous classes.

We also notice that orthogonal projections do not form a semantically meaningful representation in pixel space. In order to gain a better representation we use a deep AE. For this non-linear case, we treat the middle part of the network as an *inner* linear autoencoder which is operating on a more semantically meaningful transformed version of the data.

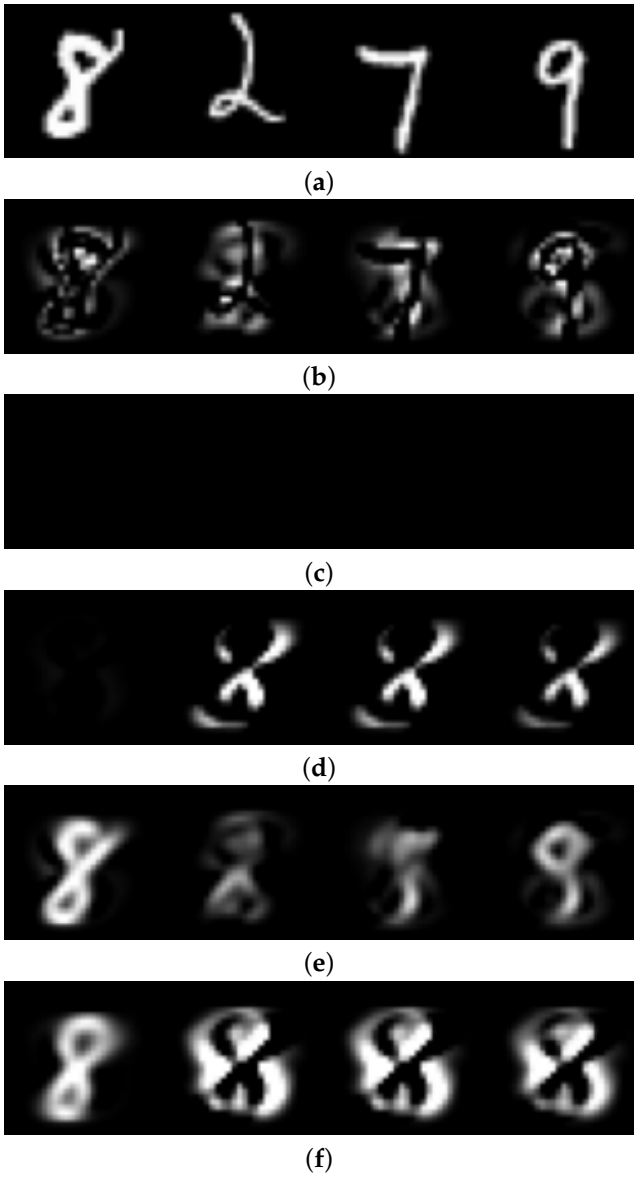

Figure 2. Comparison between linear AE and LIS-AE. Digit-8 is the normal task. (**a**) Inputs. (**b**) Orthogonal vector. (**c**) AE reconstruction of orthogonal vectors(zero images). (**d**) LIS-AE reconstruction of orthogonal vectors. (**e**) AE reconstruction of inputs. (**f**) LIS-AE reconstruction of inputs.

This suggests a stacked autoencoder architecture where another loss term for the *inner* autoencoder is added in the first phase to make sure that the output of the layer after the latent layer is similar to the latent input. In the second phase we freeze the entire network except for the encoder of the *inner* autoencoder (latent layer of the entire model) and minimize the reconstruction error of positive examples while maximizing the loss for negative examples. However, in our experiments we observed that adding these loss terms was not necessary and a similar loss to the linear case produced similar results since we only considered reconstruction scores of the outer model. Therefore, we keep the entire network frozen except for the latent layer while directly minimizing the loss (1) as before.

4.2. Intuition

For concreteness, we consider the following simple, supervised case where $\mathbf{x}^- = \mathbf{x}^a$. Given a dataset $\mathbf{X}^+$ such that for each $\mathbf{x}^+ \in \mathbf{X}^+$ distributed as a Gaussian,

$$\mathbf{x}^+ \sim \mathcal{N}\left(\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\0 & 0.2 & 0\\0 & 0 & 0.01\end{bmatrix}\right),$$

we notice that most of the variance in data is along the x-axis. Training a linear autoencoder with latent dimension $r = 1$ results in $\mathbf{D}^T = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$ and $\mathbf{E}^T = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$ where $\mathbf{D}$ and $\mathbf{E}$ are the decoder and encoder networks, respectively.
Given input $\mathbf{x}^T = \begin{bmatrix}x & y & z\end{bmatrix}$,

$$\tilde{\mathbf{x}} = \mathbf{D}\mathbf{E}^T\mathbf{x} = \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}x\\0\\0\end{bmatrix},$$

the loss score is $\mathbf{L} = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = y^2 + z^2$. Training a LIS-AE on negative samples that have only non-zero values along the z-axis, we end up with the same $\mathbf{D}$ and a modified $\hat{\mathbf{E}}^T = \begin{bmatrix}1 & 0 & \gamma\end{bmatrix}$, where γ is a large number and, then,

$$\hat{\tilde{\mathbf{x}}} = \mathbf{D}\hat{\mathbf{E}}^T\mathbf{x} = \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}1 & 0 & \gamma\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}x + \gamma z\\0\\0\end{bmatrix},$$

that results in $\hat{\mathbf{L}} = \|\mathbf{x} - \hat{\tilde{\mathbf{x}}}\| \approx y^2 + \gamma^2 z^2$ with $\mathbf{x}^+$ having the form $\begin{bmatrix}x_+ & y_+ & 0\end{bmatrix}^T$, $\mathbf{x^a}$ having the form $\begin{bmatrix}x_a & y_a & z_a\end{bmatrix}^T$ where $y_a, z_a \neq 0$. The new loss scores for $\mathbf{x}^+$ and $\mathbf{x^a}$ are:

$$\mathbf{L}(\mathbf{x}^+) = y_+^2,\ \mathbf{L}(\mathbf{x^a}) = y_a^2 + \gamma^2 z_a^2$$

In the case of a regular linear-AE (PCA), given $\mathbf{x}^+ = (x_+, y_+, 0)$, for each point $(x_a, y_a, z_a) \in$ the cylinder $\left(\sqrt{y_a^2 + z_a^2} \leq y_+, x_a \in \mathbf{R}\right)$ the following holds: $\mathbf{L}(\mathbf{x^a}) = \mathbf{L}(\mathbf{x}^+)$, making the two samples *indistinguishable*. In the case of LIS-AE, $\hat{\mathbf{L}}(\mathbf{x^a}) = \hat{\mathbf{L}}(\mathbf{x}^+)$ holds only for the elliptic cylinder $\left(\sqrt{y_a^2 + \frac{z_a^2}{1/\gamma^2}} \leq y_+, x_a \in \mathbf{R}\right)$, and since γ is a large number, the cross-section of the cylinder is *squashed* in the z dimension, resulting in heavily penalized loss in the z dimension but a regular loss in the y dimension. In this case, the two samples become *indistinguishable* only for very small values of z_a.

We note that the new $\hat{\mathbf{E}}$ is merely a rotated and stretched version of the old $\mathbf{E}$ in the xz-plane. Thus, we can think of linear LIS-AE as a regular PCA with its eigenvectors (columns of $\mathbf{U_r}$) *stretched* and *tilted* in the directions of the orthogonal complement of the eigenspace. This is done in such a way that keeps the column space of normal examples invariant under the new transformation $\mathbf{U}_r\mathbf{E^T}$. By itself, this formulation remains ill-posed since there is an infinite number of solutions that do not necessarily help with anomaly detection. More formally, given $\mathbf{E} := \mathbf{U}_r + \mathbf{U}_c\mathbf{B}$, we can choose any matrix $\mathbf{B}$ such that $Null(\mathbf{B}^T) = Col(\mathbf{U_c^T}\mathbf{X}^+)$ since $\mathbf{U}_r\mathbf{E^T}\mathbf{x}^+ = \mathbf{U}_r(\mathbf{U}_r^T\mathbf{x}^+ + \mathbf{B}^T\mathbf{U}_c^T\mathbf{x}^+) = \mathbf{U}_r\mathbf{U}_r^T\mathbf{x}^+ \approx \mathbf{x}^+$. However, this does not guarantee any advantage for anomaly detection on similar data and, even worse, in practice, this modification process might result in a slightly worse performance if done arbitrarily since the model usually has to sacrifice some extreme samples from the normal data to balance the two losses. Thus, the negative dataset is used to properly determine the directions of the *tilt* and hyper-parameters (γ and β) to determine the importance and amount of stretching (or shrinking), changing the normal case as little as possible. For deep LIS-AE, the same analogy holds albeit in a *latent space*.

Deep architectures are not only useful for learning good representation, but can learn a non-linear transformation with useful properties for our objective such as linear separability of negative and positive samples. By adding a standard binary cross-entropy loss before the non-linear activation of the latent layer during the first phase, we ensure that the input of the latent layer is linearly separable for positive and negative examples during the second phase. This linearly separable variant (LinSep-LIS-AE) almost always performs better than directly using LIS-AE. We investigate the effect of this property on the second phase in Section 5.2.

5. Experiments

We report results on the following datasets: MNIST [27], Fashion-MNIST [28], SVHN [29], and CIFAR-10 [30]. Results of our approach are compared to baseline models with the same capacity for autoencoder-based methods.

5.1. Anomaly Detection

In this section, we test LIS-AE for anomaly detection on image data in unsupervised settings. Given a standard classification dataset, we group a set of classes together into a new dataset and consider it the *"normal"* dataset. *The rest of the classes that are not in the normal nor in the negative datasets are considered anomalies*. During training, our model is presented only with the normal dataset and the additional negative dataset. We evaluate the performance on test data comprising both the "normal" and "anomalous" groups.

For MNIST and Fashion-MNIST, the encoder network consists of two convolutional layers with LeakyReLU non-linearities followed by a fully connected bottleneck layer with a *tanh* activation function. The decoder network consists of a fully connected layer followed by a LeakyReLU and two deconvolution layers with LeakyReLU activation functions and a final convolution layer with sigmoid situated at the final output. For SVHN and CIFAR-10 we use latent layers with larger sizes and higher capacity networks with the same depth. It is worth noting that the choice of latent layer size has the most effect on performance for all models (compared to other hyper-parameters). We report the best performing latent dimension for all models.

In Table 1, we compare LIS-AE with several autoencoder-based anomaly detection models as baselines, all of which share the exact same architecture. It is worth noting that the most direct comparison is between LIS-AE and AE since not only do they have the same architecture, they have the exact same encoder and decoder weights and their performance is merely measured before and after the latent-shaping phase. We use a different variant of RepNN with a sigmoid activation function $\sigma(x) = 1/(1 + \exp(-x))$ placed before the $\texttt{tanh}$ staircase function approximation described in Section 2. This is mainly used because *"squashing"* the input between 0 and 1 before passing it to the staircase function gives a more robust and easy-to-train network. We only report the best results for Sig-RepNN with 4 activation levels. For anomaly GAN (AnoGAN) [31], we follow the implementation described in [32]. We train a W-GAN [33] with gradient penalty and report performance for two anomaly scores, namely, encoder-generator reconstruction loss and additional feature-matching distance score in the discriminator feature space (AnoGAN-FM).

For AnoGAN, Linear-AE, AE, VAE, RepNN, MemAE, and LIS-A, we use reconstruction error $\mathcal{L}(\mathbf{x})$ such that if $\mathcal{L}(\mathbf{x}) > \alpha$ the input is considered an anomaly. Varying the threshold α, we are able to compute the area under the curve (AUC) as a measure of performance. Similarly, for OC-SVDD (equivalently OC-SVM with rbf kernel) and OC-DSVDD, we vary the inverse length scale γ and use a predicted class label. For kernel density estimation (KDE) [34], we vary the threshold α over the log-likelihood scores. For isolation forest (IF) [35], we vary the threshold α over the anomaly score calculated by the isolation forest algorithm.

Table 1. Average AUC for 10 tasks sampled from MNIST, Fashion-MNIST, and 5 2-class tasks sampled from MNIST.

Model	MNIST	Fashion-MNIST	2-Class MNIST
KDE	0.9568	0.9183	0.9206
IF	0.8624	0.9144	0.73018
OC-SVM	0.9108	0.8608	0.8741
OC-DSVDD	0.9489	0.8577	0.8972
AnoGAN	0.9579	0.9098	0.8406
AnoGAN-FM	0.9544	0.9072	0.8353
Linear-AE	0.9412	0.8845	0.8915
VAE	0.9642	0.9092	0.9263
Mem-AE	0.9714	0.9131	0.9352
Sig-RepNN (N=4)	0.9661	0.9124	0.9261
AE	0.9601	0.9076	0.9221
LIS-AE	0.9768	0.9256	0.9457

The datasets tested in Table 1 are MNIST and Fashion-MNIST. To train LIS-AE on MNIST we use Omniglot [36] as our negative dataset since it shares similar compositional characteristics with MNIST. Since Omniglot is a relatively small dataset, we diversify the negative examples with various augmentation techniques, namely, Gaussian blurring, random cropping, and horizontal and vertical flipping. We test two settings for MNIST, a 1-class setting where the normal dataset is one particular class and the rest of the dataset classes are considered anomalies. The process is repeated for all classes and the average AUC for 10 classes is reported. Another setting is 2-class MNIST where the normal dataset consists of two classes and the remaining classes are considered anomalies. For example, the first task contains digits 0 and 1 and the remaining digits are considered anomalies, the second task contains digit 2 and 3, and so forth. This setting is more challenging since there is more than one class present in the normal dataset. For Fashion-MNIST, the choice of the negative example is different. We use the *next class* as the negative dataset and we do not include it with anomalies (i.e., the remaining classes) during test time.

We note that LIS-AE achieves superior performance to all compared approaches, however, we also notice that these settings are comparatively easy and all tested models performed adequately including classical non-deep approaches.

In Table 2, we show performance on SVHN and CIFAR-10 which are more complex dataests compared to MNIST and Fashion-MNIST. To train LIS-AE, we split each dataset into two datasets, and each split is used as negative examples for the other one. Note that we only test on the remaining classes which are not in the normal nor the negative datasets. For example, the first dataset from CIFAR-10 has five classes, namely, airplane, automobile, bird, cat, and deer while the second one has dog, frog, horse, ship, and truck.

Training on airplane as the first normal task, LIS-AE maximizes the loss for samples drawn from the negative dataset (dog, frog, ship, and so forth). We then test its performance on airplane as the normal class and only on automobile, bird, cat, and deer as anomalies. Note that we do not test on dog, frog, and other classes in the negative dataset. This process is repeated for all 10 classes and the average AUC is reported. As mentioned in Section 4.2, we introduce LinSep-LIS-AE as an improvement over base standard LIS-AE. The difference between the two models is only in the first phase where a binary cross-entropy loss is added to ensure that positive and negative examples are linearly separable during the second phase. The last two entries of the table are supervised upper bounds for each variant where the negative dataset is the same as outliers. In Figure 3, we see that standard AE is prone to generalize well for other classes which is not a desired property for anomaly detection. In contrast, LIS-AE only reconstructs *normal* data faithfully which translates to the large performance gap we see in Figure 4.

Table 2. Average AUC for 10 anomaly detection tasks sampled from SVHN and CIFAR-10 are shown.

Model	SVHN	CIFAR-10
KDE	0.5648	0.5752
IF	0.5112	0.6097
OC-SVM	0.5047	0.5651
OC-DSVDD	0.5681	0.6411
AnoGAN	0.5598	0.5843
AnoGAN-FM	0.5645	0.5880
Linear-AE	0.5702	0.5753
VAE	0.5692	0.5734
Mem-AE	0.5720	0.5931
Sig-RepNN (*N*=4)	0.5684	0.5719
AE	0.5698	0.5703
LIS-AE	0.6886	0.8145
LinSep-LIS-AE	0.7701	0.8858
Sup. LIS-AE	0.7573	0.8384
Sup. LinSep-LIS-AE	0.8479	0.9170

Figure 3. Top row is test input from CIFAR-10 and SVHN, middle row is the output of a standard AE (first phase), and the bottom row is the output of LIS-AE. Trained on normal *"car"* and *"digit-0"* classes, LIS-AE only reconstructs samples of the *normal* class correctly.

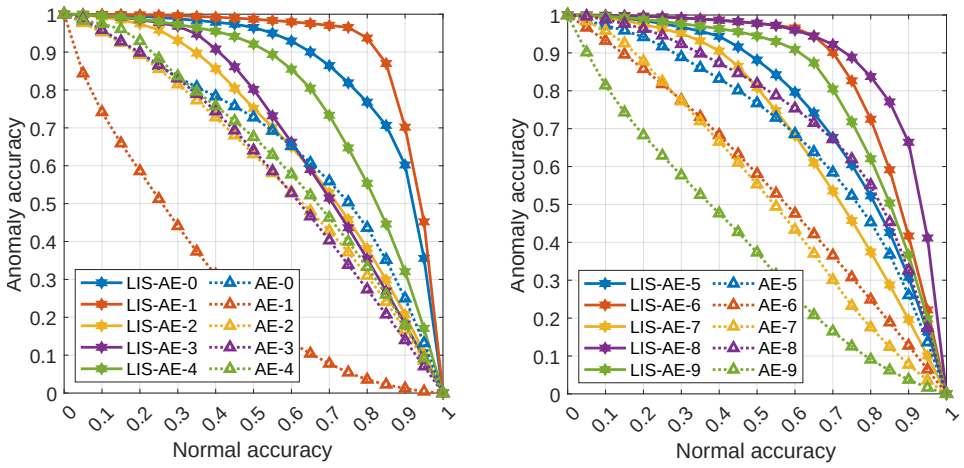

Figure 4. Each line represents a trade-off between accuracy of anomalies and normal data for CIFAR10. The **left** pane shows accuracies on tasks 0–4 and the **right** shows accuracies on tasks 5–9. Note that as the threshold value α increases, the model favors accepting anomalies over misclassifying normal examples. LIS-AE gives a significant margin compared to base AE.

We also notice that despite CIFAR-10 being more complex than SVHN, most reconstruction-based models perform better on CIFAR-10 than on SVHN. This is due to the fact that the difference between SVHN classes in terms of reconstruction is not as *large* since they share similar compositional features and appear in samples from other classes while, for CIFAR-10, classes vary significantly (e.g., digit-2 and digit-3 on a wall vs. truck and bird).

5.2. Ablation

In this section, we investigate the effect of the nature of negative dataset and linear separability of positive and negative examples. In Table 3 we train LIS-AE on different negative and positive datasets. Similar to Table 2, we split each positive dataset into two datasets and follow the same settings as before with the exception of "*None*" and "*Supervised*" cases. The "*None*" case indicates that no negative examples have been used whereas the *Supervised* case indicates that both outliers and negative datasets share the same classes. Note that this case is different from the case where the positive and negative datasets come from the same dataset. Unless stated otherwise, we only test on classes (outliers) that are not in the positive nor in the negative datasets. For example, when MNIST is used as a source for both positive and negative datasets, the positive data starts with class 0 and the negative dataset consists of classes 5 to 9 where the outliers are classes 1 to 4. This process is repeated for all 10 classes present in each dataset and the average AUC is reported. Overall, using a negative dataset resulted in a significant increase in performance in every case except for two important cases, namely, when Fashion-MNIST and CIFAR-10 were used as negative datasets for MNIST and SVHN, respectively. This could be explained by the fact that the model was not capable of reconstructing Fashion-MNIST and CIFAR-10 classes in the first place. Moreover, shaping the latent layer in such a way that maximizes the loss for Fashion-MNIST and CIFAR-10 classes does not guarantee any advantage for anomaly detection of similar digit classes present in MNIST and SVHN. This, coupled with the fact that this process in practice forces the model to ignore some samples from the normal dataset to balance the two losses, results in the performance degradation we observe in these two cases.

Table 3. Average AUC for 10 anomaly detection tasks sampled from two 5-class MNIST, Fashion-MNIST, SVHN, and CIFAR-10 datasets where a regular LIS-AE is trained with different negative splits.

Negative Data	**Positive Data**			
	MNIST	**Fashion**	**SVHN**	**CIFAR-10**
None	0.9485	0.8740	0.5698	0.5703
Omni	0.9605	0.9013	-	-
MNIST	0.9778	0.8942	-	-
Fashion	0.9482	0.9106	-	-
SVHN	-	-	0.6886	0.7065
CIFAR-10	-	-	0.5481	0.8145
Same (Sup.)	0.9901	0.9623	0.7573	0.8384

Table 4 is an excerpt of the complete table in Appendix A.1 where we examine the effect of each class present in the negative dataset on anomaly detection performance for other test classes from the CIFAR-10 dataset. We split CIFAR-10 into two separate datasets, the first split is used for selecting classes as negative datasets and the other split is used as outliers. For each class in CIFAR-10 we train eight models in different settings, the first setting is *None* where we train a standard autoencoder with no negative examples as the base model. The remaining seven settings differ in the second phase, and we select one class as our negative dataset and test the model performance on each individual class from the outlier dataset. The *combined* setting is similar to the setting described in Section 5.1 where we combine all negative classes in one 5-class negative dataset. Note that these classes are not the same as the classes in the outlier test dataset except for the final setting, which is an upper-bound supervised setting where the negative dataset comprises classes

that are in the outlier dataset except for the positive class. This process is then repeated for all 10 classes in CIFAR-10. Overall, we observe a significant performance increase over the base model with the general trend of negative classes significantly increasing anomaly detection performance for similar outliers. For example, the *dog class* drastically improves performance on the cat class but not so much for the *plane class*. However, we also notice two important exceptions, namely, when the *horse class* is used as the negative dataset for the *car class*, we notice a significant performance increase for the relatively similar *deer class* as expected, however, when the *horse class* is used as the negative dataset for the same *deer class*, we notice that the performance does not improve as in the first case and even degrades for the car class.

Table 4. AUC for LIS-AE trained on individual positive and negative classes is reported.

Positive Class	Negative Class	Outliers					
		Plane	**Car**	**Bird**	**Cat**	**Deer**	**Avg.**
Car	None	0.32	-	0.34	0.33	0.33	0.330
	Dog 5	0.67	-	0.89	0.93	0.90	0.848
	Frog 6	0.58	-	0.90	0.90	0.91	0.823
	Horse 7	0.66	-	0.88	0.90	0.92	0.840
	Ship 8	0.83	-	0.59	0.51	0.50	0.608
	Truck 9	0.51	-	0.44	0.49	0.44	0.470
	Comb. (5–9)	0.81	-	0.92	0.92	0.94	0.898
	Sup. (0–4)	0.89	-	0.93	0.90	0.95	0.918
Deer	None	0.56	0.80	0.52	0.54	-	0.605
	Dog 5	0.72	0.85	0.63	0.80	-	0.750
	Frog 6	0.66	0.86	0.60	0.75	-	0.718
	Horse 7	0.71	0.58	0.58	0.71	-	0.645
	Ship 8	0.93	0.94	0.63	0.72	-	0.805
	Truck 9	0.84	0.97	0.62	0.72	-	0.773
	Comb. (5–9)	0.87	0.95	0.61	0.73	-	0.790
	Sup. (0–4)	0.93	0.97	0.63	0.72	-	0.813

Other notable examples of this observation can be found in the appendices where, for instance, the *dog class* improves performance on cat outliers, but causes noticeable degradation when used as the negative dataset for the same cat class. The gained performance, in the first case, is due to the fact that these classes share similar compositional features and backgrounds. However, in the second case, the same property makes it difficult to balance the minimization and maximization loss during the latent-shaping phase. For example, car and truck images are very similar in this scenario so that minimizing and maximizing the loss at the same time becomes contradictory. As posited in Section 4.2, we mitigate this issue by adding a binary cross-entropy loss while training in the first phase to ensure that the input of the latent layer is linearly separable for positive and negative examples. Notice that, unlike other approaches [23,24], this does not require a labeled positive or negative dataset and relies only on the fact that we have two distinct datasets. This linear separability makes the second phase of training relatively easier and less contradictory. In Table 5, we see that LinSep-LIS-AE mitigates this issue for the aforementioned cases and gives the AUC increase we observe in Table 2.

Table 5. AUC for LinSep-LIS-AE trained on individual positive and negative classes is reported.

Positive Class	Negative Class	Outliers					
		Plane	Car	Bird	Cat	Deer	Avg.
Car	None	0.32	-	0.34	0.33	0.33	0.330
	Dog 5	0.67	-	0.94	**0.97**	0.95	0.883
	Frog 6	0.58	-	0.93	0.96	0.96	0.858
	Horse 7	0.69	-	**0.95**	**0.97**	**0.97**	0.895
	Ship 8	**0.90**	-	0.78	0.79	0.76	0.808
	Truck 9	0.59	-	0.77	0.82	0.73	0.728
	Comb. (5–9)	0.90	-	0.97	0.97	0.98	0.955
	Sup. (0–4)	0.95	-	0.98	0.98	0.98	0.9725
Deer	None	0.56	0.80	0.52	0.54	-	0.605
	Dog 5	0.67	0.86	**0.71**	**0.89**	-	0.783
	Frog 6	0.68	0.87	0.62	0.79	-	0.740
	Horse 7	0.70	0.84	0.61	0.72	-	0.718
	Ship 8	**0.94**	0.95	0.63	0.73	-	0.813
	Truck 9	0.84	**0.97**	0.61	0.76	-	0.795
	Comb. (5–9)	0.90	0.97	0.66	0.80	-	0.833
	Sup. (0–4)	0.97	0.98	0.76	0.83	-	0.885

6. Conclusions

In this paper we introduced a novel autoencoder-based model called latent-insensitive autoencoder (LIS-AE). With the help of negative samples drawn from a similar domain as the normal data we tune the weights of the bottleneck part of a standard autoencoder such that the resulting model is able to reconstruct the target task while penalizing anomalous samples. We also presented theoretical justification for the reasoning behind our two-phase training process and the latent-shaping loss function along with a more powerful variant. Multiple ablation studies were conducted to explain the effect of negative classes and highlight other important aspects of our model. We tested our model in a variety of anomaly detection settings with multiple datasets of varying degrees of complexity. Experimental results showed significant performance improvement over compared methods. Future research will focus on possible ways to synthesize negative examples for domains with limited data. We also hope to further study and employ various manifold learning approaches for latent space representation.

Author Contributions: Conceptualization, A.A.L.; Data curation, M.S.B; Formal analysis, A.A.L. and M.S.B; Methodology, A.A.L. and M.S.B; Software, M.S.B; Supervision,A.A.L.; Validation, M.S.B; Visualization, A.A.L. and M.S.B; Writing—original draft, M.S.B; Writing—review and editing, A.A.L. and M.S.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets are publicly available as part of the torchvision.datasets module.

Acknowledgments: Artem Lenskiy was funded by Our Health in Our Hands (OHIOH), a strategic initiative of the Australian National University, which aims to transform healthcare by developing new personalized health technologies and solutions in collaboration with patients, clinicians, and health care providers.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. Effect of Individual Classes as Negative Examples

As discussed in Section 5.2, we examine the effect of each class present in the negative dataset on anomaly detection performance for other test classes from the CIFAR-10 dataset. The first table shows results for standard LIS-AE while the second table shows results for LinSep-LIS-AE.

Table A1. Complete results of standard LIS-AE trained on individual positive and negative classes of the first 5-class split of CIFAR-10.

Positive Class	Negative Class	Outliers					
		Plane	**Car**	**Bird**	**Cat**	**Deer**	**Avg.**
Plane	None	-	0.78	0.58	0.62	0.61	0.648
	Dog 5	-	0.83	0.87	0.95	0.91	0.890
	Frog 6	-	0.83	0.86	0.94	0.90	0.883
	Horse 7	-	0.82	0.86	0.94	0.91	0.883
	Ship 8	-	0.83	0.86	0.92	0.90	0.878
	Truck 9	-	0.84	0.86	0.95	0.91	0.890
	Comb. (5–9)	-	0.83	0.85	0.93	0.91	0.880
	Sup. (0–4)	-	0.81	0.88	0.94	0.92	0.888
Car	None	0.32	-	0.34	0.33	0.33	0.330
	Dog 5	0.67	-	0.89	0.93	0.9	0.848
	Frog 6	0.58	-	0.9	0.9	0.91	0.823
	Horse 7	0.66	-	0.88	0.9	0.92	0.840
	Ship 8	0.83	-	0.59	0.51	0.5	0.608
	Truck 9	0.51	-	0.44	0.49	0.44	0.470
	Comb. (5–9)	0.81	-	0.92	0.92	0.94	0.898
	Sup. (0–4)	0.89	-	0.93	0.90	0.95	0.918
Bird	None	0.52	0.78	-	0.54	0.52	0.590
	Dog 5	0.59	0.75	-	0.71	0.49	0.635
	Frog 6	0.53	0.78	-	0.69	0.54	0.635
	Horse 7	0.63	0.80	-	0.66	0.57	0.665
	Ship 8	0.86	0.89	-	0.61	0.45	0.703
	Truck 9	0.76	0.94	-	0.64	0.72	0.765
	Comb. (5–9)	0.78	0.90	-	0.62	0.49	0.698
	Sup. (0–4)	0.82	0.94	-	0.60	0.48	0.710
Cat	None	0.55	0.76	0.50	-	0.50	0.578
	Dog 5	0.54	0.73	0.52	-	0.52	0.578
	Frog 6	0.56	0.72	0.60	-	0.62	0.625
	Horse 7	0.70	0.75	0.59	-	0.68	0.680
	Ship 8	0.91	0.89	0.53	-	0.46	0.678
	Truck 9	0.82	0.94	0.50	-	0.48	0.685
	Comb. (5–9)	0.89	0.91	0.55	-	0.52	0.718
	Sup. (0–4)	0.93	0.93	0.58	-	0.54	0.745
Deer	None	0.56	0.80	0.52	0.54	-	0.605
	Dog 5	0.72	0.85	0.63	0.80	-	0.750
	Frog 6	0.66	0.86	0.60	0.75	-	0.718
	Horse 7	0.71	0.58	0.58	0.71	-	0.645
	Ship 8	0.93	0.94	0.63	0.72	-	0.805
	Truck 9	0.84	0.97	0.62	0.72	-	0.773
	Comb. (5–9)	0.87	0.95	0.61	0.73	-	0.790
	Sup. (0–4)	0.93	0.97	0.62	0.72	-	0.810

Table A2. Complete results of standard LIS-AE trained on individual positive and negative classes of the second 5-class split of CIFAR-10.

Positive Class	Negative Class	Outliers					
		Dog	**Frog**	**Horse**	**Ship**	**Truck**	**Avg.**
Dog	None	-	0.69	0.66	0.57	0.77	0.673
	Plane 0	-	0.53	0.66	0.92	0.89	0.750
	Car 1	-	0.56	0.68	0.95	0.91	0.775
	Bird 2	-	0.63	0.63	0.77	0.78	0.703
	Cat 3	-	0.67	0.65	0.66	0.81	0.698
	Deer 4	-	0.73	0.69	0.70	0.76	0.720
	Comb. (0–4)	-	0.58	0.67	0.95	0.94	0.785
	Sup. (5–9)	-	0.56	0.73	0.95	0.95	0.798
Frog	None	0.40	-	0.53	0.49	0.67	0.523
	Plane 0	0.73	-	0.81	0.96	0.93	0.858
	Car 1	0.74	-	0.83	0.96	0.97	0.875
	Bird 2	0.80	-	0.84	0.91	0.85	0.850
	Cat 3	0.84	-	0.80	0.87	0.86	0.843
	Deer 4	0.75	-	0.86	0.88	0.87	0.840
	Comb. (0–4)	0.75	-	0.84	0.97	0.95	0.877
	Sup. (5–9)	0.82	-	0.88	0.97	0.96	0.907
Horse	None	0.41	0.58	-	0.46	0.66	0.528
	Plane 0	0.55	0.50	-	0.93	0.83	0.703
	Car 1	0.56	0.58	-	0.90	0.93	0.743
	Bird 2	0.62	0.73	-	0.80	0.71	0.715
	Cat 3	0.77	0.76	-	0.65	0.66	0.710
	Deer 4	0.62	0.83	-	0.57	0.60	0.655
	Comb. (0–4)	0.51	0.57	-	0.89	0.88	0.713
	Sup. (5–9)	0.59	0.66	-	0.95	0.92	0.780
Ship	None	0.62	0.74	0.73	-	0.77	0.715
	Plane 0	0.75	0.75	0.82	-	0.74	0.765
	Car 1	0.84	0.89	0.90	-	0.88	0.878
	Bird 2	0.94	0.96	0.94	-	0.78	0.905
	Cat 3	0.95	0.95	0.93	-	0.8	0.908
	Deer 4	0.92	0.96	0.94	-	0.78	0.900
	Comb. (0–4)	0.95	0.97	0.96	-	0.83	0.928
	Sup. (5–9)	0.94	0.96	0.96	-	0.88	0.935
Truck	None	0.35	0.53	0.46	0.30	-	0.41
	Plane 0	0.61	0.52	0.58	0.80	-	0.628
	Car 1	0.51	0.57	0.53	0.47	-	0.520
	Bird 2	0.92	0.90	0.84	0.73	-	0.848
	Cat 3	0.95	0.90	0.82	0.61	-	0.820
	Deer 4	0.91	0.92	0.86	0.63	-	0.830
	Comb. (0–4)	0.93	0.91	0.83	0.76	-	0.858
	Sup. (5–9)	0.94	0.95	0.90	0.78	-	0.893

Table A3. Complete results of LinSep-LIS-AE trained on individual positive and negative classes of the first 5-class split of CIFAR-10.

Positive Class	Negative Class	Outliers					
		Plane	Car	Bird	Cat	Deer	Avg.
Plane	None	-	0.78	0.58	0.62	0.61	0.648
	Dog 5	-	0.84	0.88	0.97	0.93	0.905
	Frog 6	-	0.84	0.86	0.95	0.93	0.895
	Horse 7	-	0.83	0.85	0.95	0.92	0.888
	Ship 8	-	0.80	0.57	0.74	0.56	0.668
	Truck 9	-	0.92	0.71	0.87	0.74	0.810
	Comb. (0–4)	-	0.90	0.85	0.96	0.94	0.913
	Sup. (5–9)	-	0.93	0.90	0.96	0.95	0.935
Car	None	0.32	-	0.34	0.33	0.33	0.330
	Dog 5	0.67	-	0.94	0.97	0.95	0.883
	Frog 6	0.58	-	0.93	0.96	0.96	0.858
	Horse 7	0.69	-	0.95	0.97	0.97	0.895
	Ship 8	0.90	-	0.78	0.79	0.76	0.808
	Truck 9	0.59	-	0.77	0.82	0.73	0.728
	Comb. (0–4)	0.90	-	0.97	0.97	0.98	0.955
	Sup. (5–9)	0.95	-	0.98	0.98	0.98	0.973
Bird	None	0.52	0.78	-	0.54	0.52	0.590
	Dog 5	0.56	0.78	-	0.81	0.55	0.675
	Frog 6	0.53	0.80	-	0.71	0.56	0.650
	Horse 7	0.63	0.81	-	0.72	0.59	0.688
	Ship 8	0.86	0.93	-	0.64	0.47	0.725
	Truck 9	0.74	0.95	-	0.68	0.48	0.713
	Comb. (0–4)	0.82	0.95	-	0.76	0.60	0.783
	Sup. (5–9)	0.89	0.97	-	0.70	0.56	0.773
Cat	None	0.55	0.76	0.50	-	0.50	0.578
	Dog 5	0.50	0.71	0.51	-	0.46	0.545
	Frog 6	0.52	0.76	0.58	-	0.64	0.625
	Horse 7	0.65	0.79	0.56	-	0.64	0.660
	Ship 8	0.93	0.92	0.56	-	0.48	0.723
	Truck 9	0.81	0.96	0.52	-	0.48	0.693
	Comb. (5–9)	0.88	0.95	0.62	-	0.68	0.783
	Sup. (5–9)	0.95	0.97	0.75	-	0.76	0.860
Deer	None	0.56	0.80	0.52	0.54	-	0.605
	Dog 5	0.67	0.86	0.71	0.89	-	0.783
	Frog 6	0.68	0.87	0.62	0.79	-	0.740
	Horse 7	0.70	0.84	0.61	0.72	-	0.718
	Ship 8	0.94	0.95	0.63	0.73	-	0.813
	Truck 9	0.84	0.97	0.61	0.76	-	0.795
	Comb. (0–4)	0.90	0.97	0.66	0.80	-	0.833
	Sup. (5–9)	0.97	0.98	0.76	0.83	-	0.885

Table A4. Complete results of LinSep-LIS-AE trained on individual positive and negative classes of the second 5-class split of CIFAR-10.

Positive Class	Negative Class	Outliers					
		Dog	**Frog**	**Horse**	**Ship**	**Truck**	**Avg.**
Dog	None	-	0.69	0.66	0.57	0.77	0.673
	Plane 0	-	0.62	0.68	0.98	0.94	0.805
	Car 1	-	0.67	0.7	0.96	0.97	0.825
	Bird 2	-	0.73	0.68	0.94	0.86	0.803
	Cat 3	-	0.65	0.68	0.8	0.85	0.745
	Deer 4	-	0.81	0.74	0.89	0.81	0.8125
	Comb. (0–4)	-	0.80	0.76	0.97	0.96	0.874
	Sup. (5–9)	-	0.90	0.85	0.97	0.98	0.925
Frog	None	0.40	-	0.53	0.49	0.67	0.523
	Plane 0	0.7	-	0.58	0.98	0.96	0.841
	Car 1	0.7	-	0.83	0.98	0.98	0.930
	Bird 2	0.86	-	0.91	0.96	0.93	0.933
	Cat 3	0.88	-	0.87	0.94	0.92	0.912
	Deer 4	0.81	-	0.92	0.95	0.92	0.929
	Comb. (0–4)	0.91	-	0.95	0.98	0.98	0.956
	Sup. (5–9)	0.94	-	0.97	0.98	0.98	0.968
Horse	None	0.41	0.58	-	0.46	0.66	0.0531
	Plane 0	0.53	0.59	-	0.96	0.87	0.806
	Car 1	0.57	0.67	-	0.96	0.95	0.860
	Bird 2	0.33	0.75	-	0.93	0.79	0.823
	Cat 3	0.78	0.81	-	0.9	0.77	0.826
	Deer 4	0.67	0.85	-	0.87	0.68	0.800
	Comb. (0–4)	0.75	0.91	-	0.97	0.93	0.891
	Sup. (5–9)	0.82	0.96	-	0.98	0.96	0.930
Ship	None	062	0.74	0.73	-	0.77	0.717
	Plane 0	0.84	0.89	0.93	-	0.85	0.890
	Car 1	0.86	0.91	0.93	-	0.92	0.920
	Bird 2	0.96	0.97	0.97	-	0.85	0.930
	Cat 3	0.97	0.98	0.97	-	0.85	0.933
	Deer 4	0.95	0.97	0.97	-	0.84	0.927
	Comb. (0–4)	0.97	0.98	0.98	-	0.90	0.956
	Sup. (5–9)	0.97	0.98	0.98	-	0.94	0.968
Truck	None	0.35	0.53	0.46	0.30	-	0.412
	Plane 0	0.69	0.70	0.67	0.87	-	0.733
	Car 1	0.54	0.61	0.53	0.61	-	0.573
	Bird 2	0.93	0.89	0.88	0.73	-	0.858
	Cat 3	0.96	0.91	0.88	0.63	-	0.845
	Deer 4	0.90	0.88	0.91	0.67	-	0.840
	Comb. (0–4)	0.97	0.96	0.91	0.82	-	0.914
	Sup. (5–9)	0.98	0.98	0.96	0.89	-	0.953

Appendix A.2. Detailed Results for All 10 Tasks

The following graphs are detailed results for some experiments in various settings described in Sections 5.1 and 5.2. Each curve represents a trade-off between accuracy on anomalies and on normal data for each dataset. The two left panes are an upper-bound *supervised* setting where the negative dataset is the same as outliers. The top pane shows accuracies on tasks 0 to 4 and the bottom shows accuracies on tasks 5 to 9. Note that as the threshold value α increases, the model favors accepting anomalies over misclassifying normal examples. In almost all cases, we observe that LIS-AE gives a significant margin compared to normal AE.

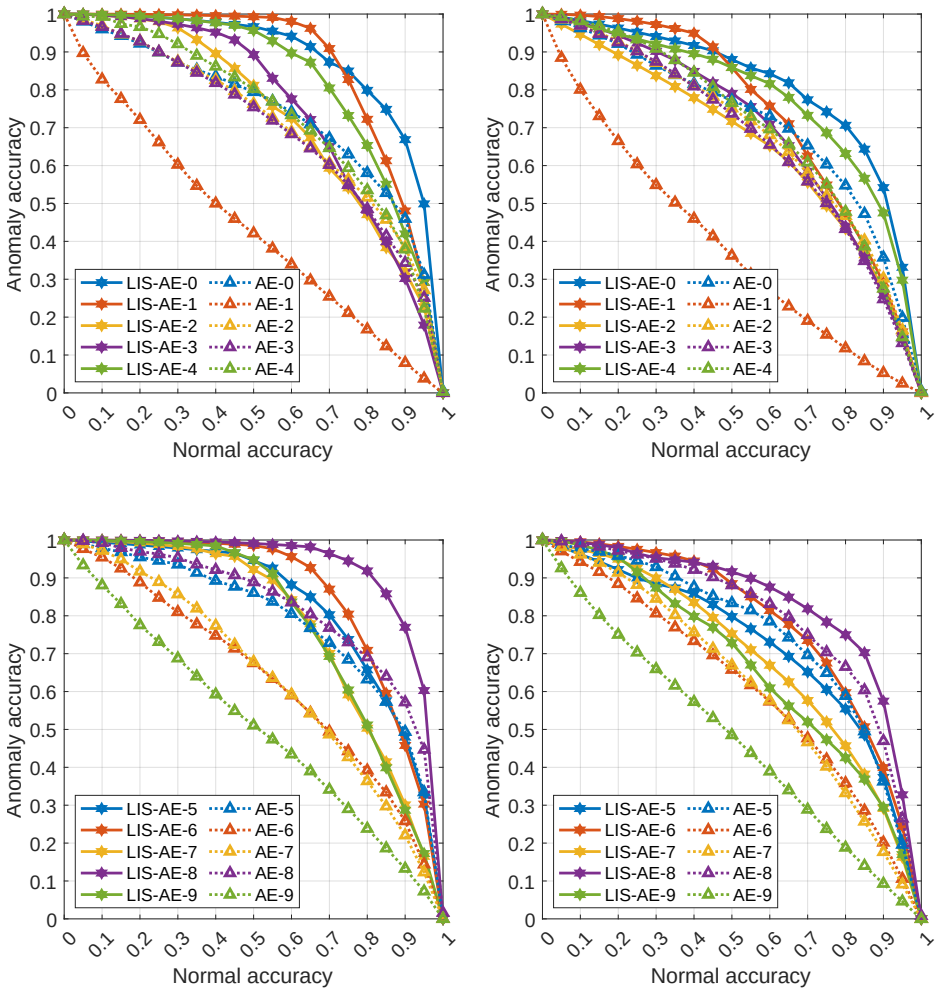

Figure A1. LIS-AE trained on CIFAR-10: (**left**) outliers as negative dataset (supervised), (**right**) SVHN as negative dataset .

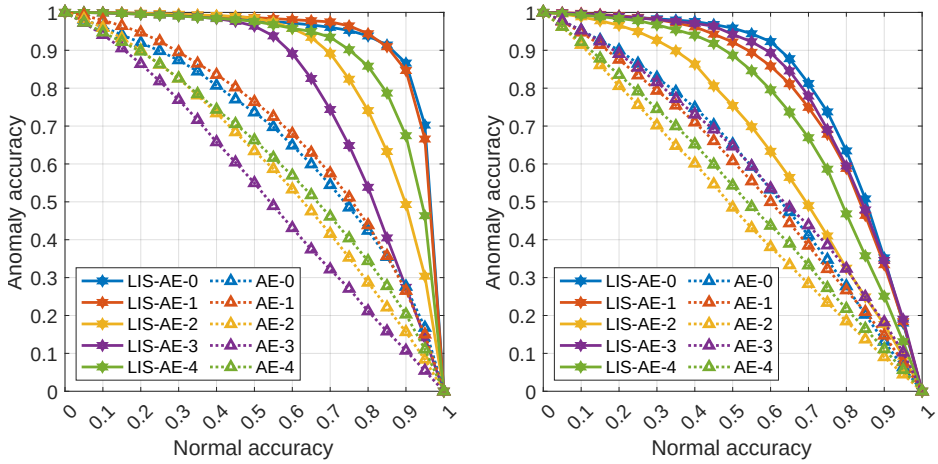

Figure A2. *Cont.*

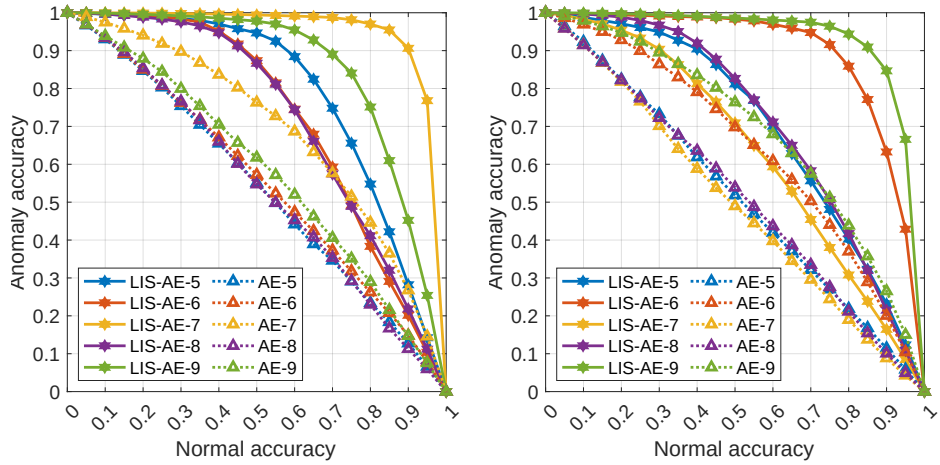

Figure A2. Results of LinSep-LIS-AE variant on SVHN: (**left**) outliers as negative dataset (supervised), (**right**) unsupervised.

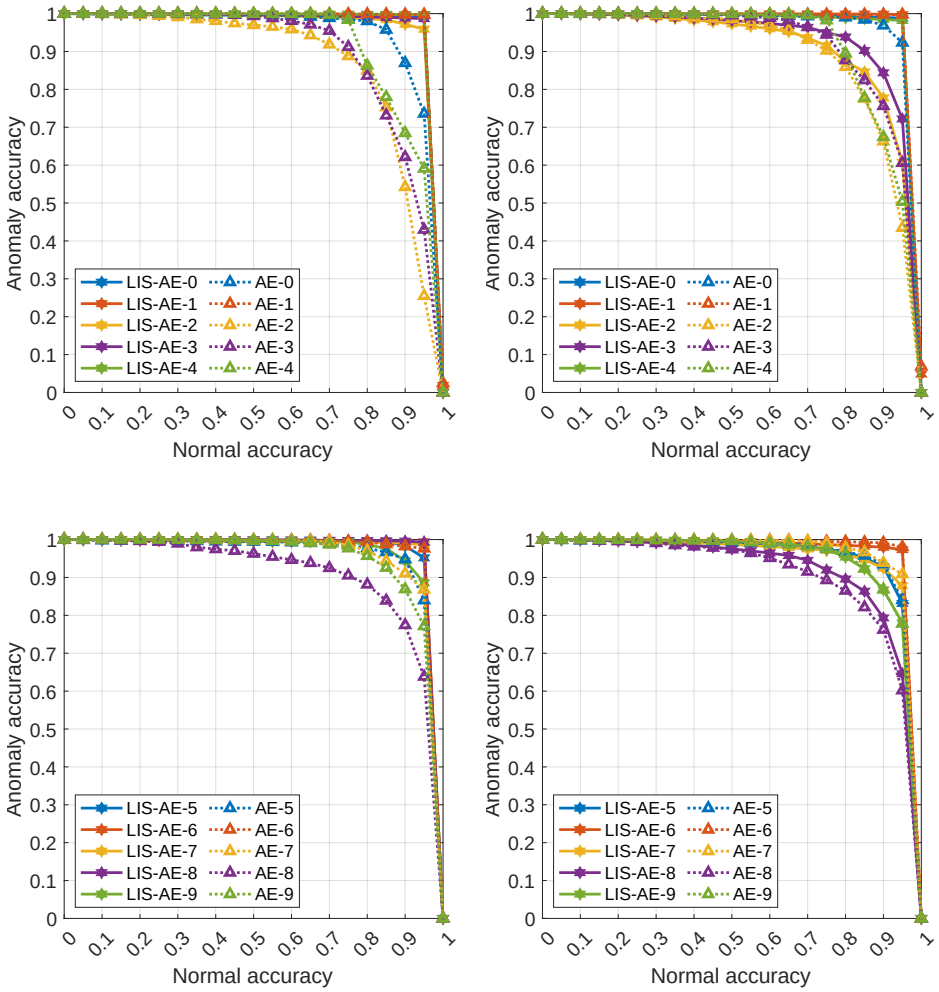

Figure A3. MNIST classes as positive datasets: (**left**) outliers as negative dataset (supervised), (**right**) Omniglot as negative dataset.

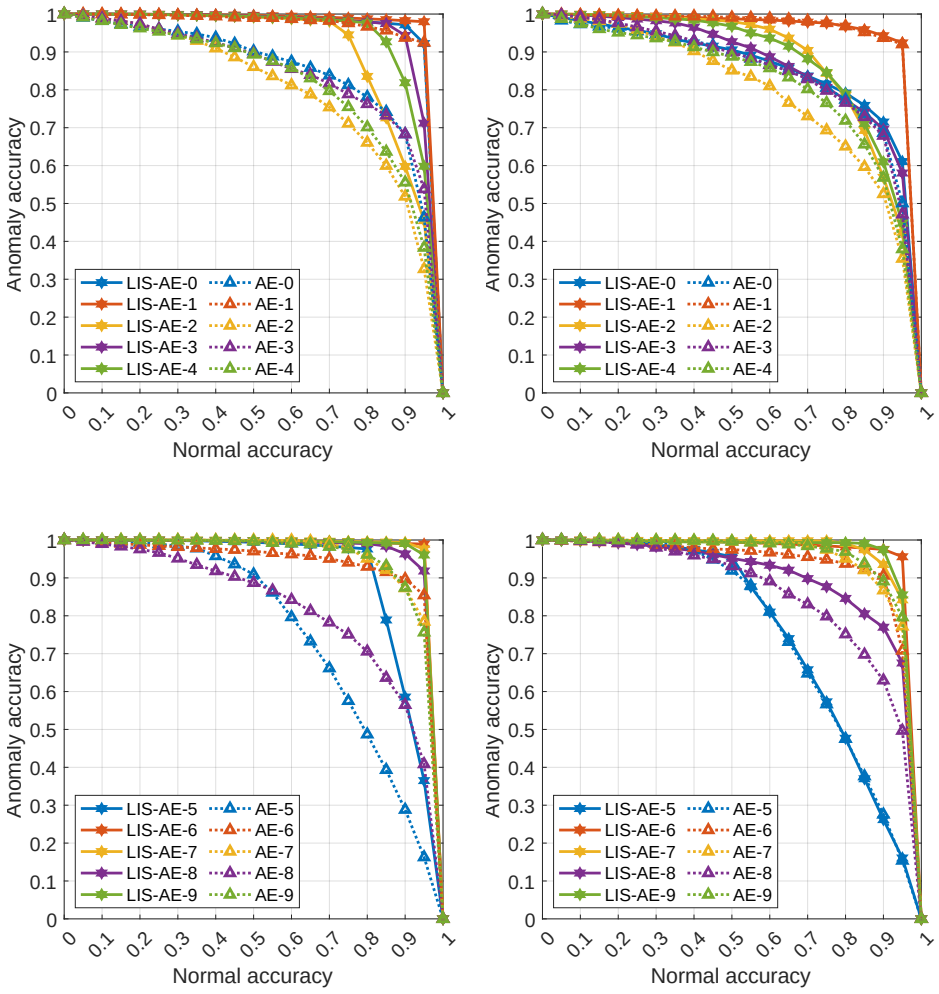

Figure A4. Fashion-MNIST classes as positive datasets: (**left**) outliers as negative dataset (supervised), (**right**) Omniglot as negative dataset.

References

1. Zhou, C.; Paffenroth, R.C. Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 665–674.
2. Ahmad, S.; Lavin, A.; Purdy, S.; Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* **2017**, *262*, 134–147. [CrossRef]
3. Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H. Greedy layer-wise training of deep networks. *Adv. Neural Inf. Process. Syst.* **2007**, *19*, 153.
4. Zimek, A.; Schubert, E.; Kriegel, H.P. A survey on unsupervised outlier detection in high-dimensional numerical data. *Stat. Anal. Data Mining ASA Data Sci. J.* **2012**, *5*, 363–387. [CrossRef]
5. Chalapathy, R.; Menon, A.K.; Chawla, S. Robust, deep and inductive anomaly detection. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Skopje, Macedonia, 18 September 2017; 2017; pp. 36–51.
6. Hasan, M.; Choi, J.; Neumann, J.; Roy-Chowdhury, A.K.; Davis, L.S. Learning temporal regularity in video sequences. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 733–742.
7. Zong, B.; Song, Q.; Min, M.R.; Cheng, W.; Lumezanu, C.; Cho, D.; Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
8. Gong, D.; Liu, L.; Le, V.; Saha, B.; Mansour, M.R.; Venkatesh, S.; Hengel, A.v.d. Available online: https://arxiv.org/abs/1904.02639 (accessed on 13 November 2021).
9. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv. (CSUR)* **2009**, *41*, 1–58. [CrossRef]
10. Weston, J.; Collobert, R.; Sinz, F.; Bottou, L.; Vapnik, V. Inference with the universum. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 1009–1016.

11. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *Lond. Edinburgh Dublin Philos. Mag. J. Sci.* **1901**, *2*, 559–572. [CrossRef]
12. Candès, E.J.; Li, X.; Ma, Y.; Wright, J. Robust principal component analysis? *J. ACM (JACM)* **2011**, *58*, 1–37. [CrossRef]
13. Bourlard, H.; Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybern.* **1988**, *59*, 291–294. [CrossRef] [PubMed]
14. An, J.; Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Spec. Lect. IE* **2015**, *2*, 1–18.
15. Hawkins, S.; He, H.; Williams, G.; Baxter, R. Outlier detection using replicator neural networks. In Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, DaWaK, France, 4–6 September 2002; pp. 170–180.
16. Williams, G.; Baxter, R.; He, H.; Hawkins, S.; Gu, L. A comparative study of RNN for outlier detection in data mining. In Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 9–12 December 2002; pp. 709–712.
17. Tóth, L.; Gosztolya, G. Replicator neural networks for outlier modeling in segmental speech recognition. In Proceedings of the International Symposium on Neural Networks, Dalian, China, 19–21 August 2004; pp. 996–1001.
18. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
19. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the support of a high-dimensional distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [CrossRef] [PubMed]
20. Tax, D.M.; Duin, R.P. Support vector data description. *Mach. Learn.* **2004**, *54*, 45–66. [CrossRef]
21. Lampert, C.H. *Kernel Methods in Computer Vision*; Now Publishers Inc.: Delft, The Netherlands, 2009.
22. Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S.A.; Binder, A.; Müller, E.; Kloft, M. Deep one-class classification. In Proceedings of the International Conference on Machine Learning; Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018; pp. 4393–4402.
23. Perera, P.; Patel, V.M. Learning deep features for one-class classification. *IEEE Trans. Image Process.* **2019**, *28*, 5450–5463. [CrossRef] [PubMed]
24. Hendrycks, D.; Mazeika, M.; Dietterich, T. Deep anomaly detection with outlier exposure. *arXiv* **2018**, arXiv:1812.04606.
25. Tang, T.; Zhou, S.; Deng, Z.; Zou, H.; Lei, L. Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining. *Sensors* **2017**, *17*, 336. [CrossRef]
26. Eckart, C.; Young, G. The approximation of one matrix by another of lower rank. *Psychometrika* **1936**, *1*, 211–218. [CrossRef]
27. LeCun, Y.; Cortes, C. MNIST Handwritten Digit Database. 2010. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 13 November 2021).
28. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
29. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A.Y. Reading Digits in Natural Images with Unsupervised Feature Learning. 2011. Available online: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf (accessed on 13 November 2021).
30. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. 2009. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 13 November 2021).
31. Schlegl, T.; Seeböck, P.; Waldstein, S.M.; Schmidt-Erfurth, U.; Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Proceedings of the International Conference on Information Processing in Medical Imaging, Boone, NC, USA, 25–30 June 2017; pp. 146–157.
32. Schlegl, T.; Seeböck, P.; Waldstein, S.M.; Langs, G.; Schmidt-Erfurth, U. f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. *Med. Image Anal.* **2019**, *54*, 30–44. [CrossRef] [PubMed]
33. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved training of wasserstein gans. *arXiv* **2017**, arXiv:1704.00028.
34. Parzen, E. On estimation of a probability density function and mode. *Ann. Math. Stat.* **1962**, *33*, 1065–1076. [CrossRef]
35. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422.
36. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338.