applied sciences

MDPI

Article

Extended Autoencoder for Novelty Detection with Reconstruction along Projection Pathway

Seung Yeop Shin and Han-joon Kim *

School of Electrical and Computer Engineering, University of Seoul, 163 Seoulsiripdaero, Seoul 02504, Korea; duq7746@uos.ac.kr

* Correspondence: khj@uos.ac.kr; Tel.: +82-2-6490-2339

Received: 16 May 2020; Accepted: 28 June 2020; Published: 29 June 2020

Featured Application: The potential applications of novelty detection are to detect the cause of process defects in manufacturing industry and to identify unique opinions in natural language processing as well as to detect credit card thefts in credit card companies.

Abstract: Recently, novelty detection with reconstruction along projection pathway (RaPP) has made progress toward leveraging hidden activation values. RaPP compares the input and its autoencoder reconstruction in hidden spaces to detect novelty samples. Nevertheless, traditional autoencoders have not yet begun to fully exploit this method. In this paper, we propose a new model, the Extended Autoencoder Model, that adds an adversarial component to the autoencoder to take full advantage of RaPP. The adversarial component matches the latent variables of the reconstructed input to the latent variables of the original input to detect novelty samples with high hidden reconstruction errors. The proposed model can be combined with variants of the autoencoder, such as a variational autoencoder or adversarial autoencoder. The effectiveness of the proposed model was evaluated across various novelty detection datasets. Our results demonstrated that extended autoencoders are capable of outperforming conventional autoencoders in detecting novelties using the RaPP method.

Keywords: autoencoder; generative adversarial networks; deep learning; novelty detection

1. Introduction

Novelty detection is the task of recognizing data points that are different from normal data [1]. Novelty detection applies to diverse domains, including intrusion detection, fraud detection, malware detection, medical anomaly detection, industrial anomaly detection, video surveillance, and other numerous fields [2]. More than a decade ago, deep autoencoders were used successfully to detect novelty samples based on reconstruction error [3]. Reconstruction error is the distance between the original input and its autoencoder reconstruction. Autoencoders compress the input into a lower-dimensional projection and then reconstruct the output from this representation. Reconstruction-based methods assume that novelties cannot be effectively reconstructed from low-dimensional projections [4]. Thus, the data samples with high reconstruction error can be detected as novelties. However, reconstruction-based methods have a limitation in that they do not leverage the information available along the projection pathway of deep autoencoders. To extract information in hidden spaces, we can directly compare the outputs of hidden layers of the encoder and the corresponding outputs of hidden layers in the decoder using a symmetric autoencoder. The outputs of the hidden layers in the encoder are referred to as hidden activations, and the outputs of hidden layers in the decoder are hidden reconstructions. However, this direct way of computing hidden reconstruction error provides no meaningful information, as the autoencoder does not impose any correspondence between members of encoding–decoding pairs during training.

Recently, reconstruction along projection pathway (RaPP) was introduced as a way of detecting novelty samples using the information in hidden spaces [5], providing an indirect way of computing hidden reconstruction error. RaPP carries out novelty detection by comparing the hidden activations of the input with those of the reconstructed input. In other words, RaPP replaces the hidden reconstructions of the original input with the hidden activations of the reconstructed input. For this replacement, hidden activations of the reconstructed input are equivalent to the corresponding hidden reconstructions of the original input on the strict condition that the autoencoder can perfectly reconstruct the input. Without any further training or changes to the autoencoder, the RaPP outperforms ordinary autoencoder-based reconstruction methods. However, the RaPP method can be further improved.

In reconstruction-based novelty detection methods, autoencoders are trained to minimize reconstruction errors with normal samples. To take full advantage of the RaPP method, we propose that autoencoders be trained to minimize hidden reconstruction errors by matching the hidden activations of the reconstructed input to the hidden activations of the original input during training. To this end, we borrow the concept of "adversarial autoencoder" (AAE) [6], in which adversarial training [7] is used to match the encoding distribution with an arbitrary prior distribution. Figure 1 depicts AAE architecture. The discriminator D_z tries to distinguish whether a sample arises from the latent variables or from a prior distribution. At the same time, the encoder is updated to fool the discriminator D_z. We use adversarial training to match the latent variables of the reconstructed input to the latent variables of the original input. A discriminator is then added on top of the autoencoder to create an extended autoencoder. Whereas the ordinary autoencoder only focuses on the input space to minimize the reconstruction error, our extended autoencoder investigates both the input space and hidden space simultaneously.

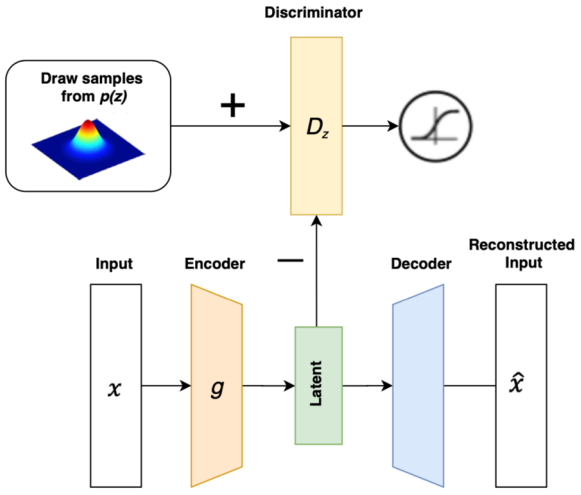

Figure 1. Architecture of the adversarial autoencoder (AAE).

Figure 2 illustrates t-SNE(t-Stochastic Neighbor Embedding) [8] visualization of the values of latent variables in the standard autoencoder and the extended autoencoder during training with normal samples. As seen in the figure, the values of latent variables of original input does not match well with those of reconstructed input in the standard autoencoder. By contrast, the two types of latent variable values tend to overlap in the extended autoencoder; this is because the extended autoencoder lowers hidden reconstruction errors. Relatively, novelty samples can be expected to have larger hidden reconstruction errors as well as larger reconstruction errors compared to normal samples as the autoencoders learn only normal data. We also show that because our method drives the decoder to be the inverse of the encoder, we can compute hidden reconstruction errors without strict assumptions about the autoencoder. Our proposed extended autoencoder model outperformed standard autoencoders for various popular datasets from Kaggle and the University of California at

Irvine (UCI) repository. In addition, we also demonstrate the efficiency of the extended autoencoders for MNIST (Modified National Institute of Standards and Technology) and fashion F-MNIST datasets. The extended autoencoders provide similar performance, only with 50 times fewer epochs than the standard autoencoders for these benchmark datasets.

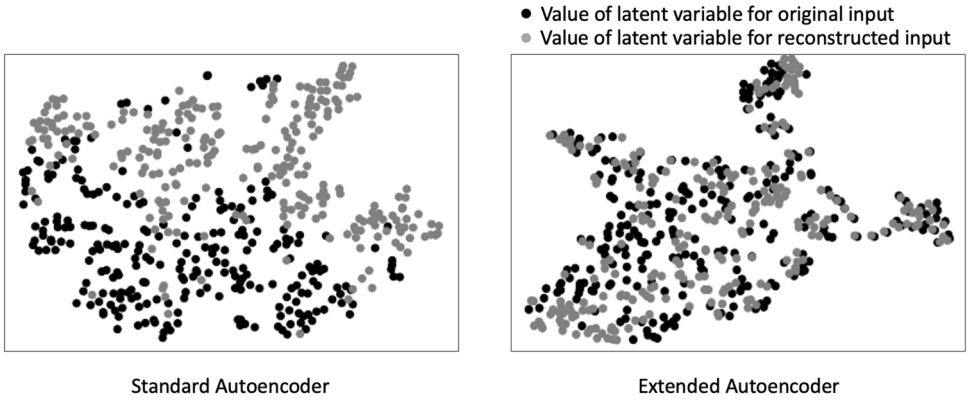

Figure 2. t-SNE (Stochastic Neighbor Embedding) visualization of the values of latent variables in the standard autoencoder (**left**) and the extended autoencoder (**right**) with normal samples. The digit 8 in MNIST (Modified National Institute of Standards and Technology) dataset is fed as a normal data to train the autoencoders. The values of latent variables for original input data and reconstructed ones are represented as black and gray points, respectively.

2. Related Work

With the advent of deep learning, deep autoencoder-based novelty detection algorithms have been widely researched. Under normal operation, a deep autoencoder uses the reconstruction error to measure the novelty score; that is, the autoencoder is trained to minimize the reconstruction error given the data samples. In doing so, the autoencoder learns a mapping function that reconstructs the normal data with a small reconstruction error. However, the ability to accurately reconstruct the novelty samples is limited, as the novelty samples do not resemble the data used for training. Therefore, novelty samples are detected according to a high reconstruction error. If the novelties seem to come from a specific distribution, generative models such as variational autoencoders (VAE) [9] are suitable for detecting novelties. Because VAEs encode the input as a distribution over the latent space instead of as a single point, a VAE can outperform an ordinary autoencoder in novelty detection based on reconstruction error [10].

Adversarial autoencoders (AAEs) can also be used for novelty detection. The advantage of AAE over VAE is that adversarial training drives the encoding distribution to match the arbitrary prior distribution. Due to this advantage, there are many applications of AAE for novelty detection. For example, to improve the interpretability of detected anomalies, a mixture of Gaussians is chosen as an arbitrary prior distribution [11]. A holistic view of the latent space that resides within the Gaussian mixture partitions the latent space into semantic regions. One-class novelty detection using generative adversarial networks (OCGAN) [12] uses adversarial training to ensure that the latent space resembles a uniform distribution; in this bounded latent space, OCGAN finds informative negative samples to improve novelty detection. Additionally, generative probabilistic novelty detection (GPND) [13] improves novelty detection by making the computation of the novelty probability feasible. GPND introduces adversarial training to match the output of the decoder with the real data distribution. This adversarial training reduces blurriness and adds additional detail to the generated images. From this perspective, adversarial training can be introduced to conventional autoencoders, making them more suitable for RaPP applications.

3. Preliminaries of RaPP

RaPP, a method for detecting novelty samples by exploiting hidden activation values, compares the input data and the reconstructed data, similar to ordinary reconstruction-based methods; however, RaPP also extends these comparisons to hidden spaces. An intuitive application of this idea would be to compare the activation in hidden space and the corresponding reconstruction in that hidden space; however, this direct comparison has no effect due to the lack of an explicit correspondence between members of encoding–decoding pairs of hidden layers during training. However, RaPP can compute the hidden reconstructions indirectly, using the hidden activations of the reconstructed input. This indirect computation is depicted in Figure 3.

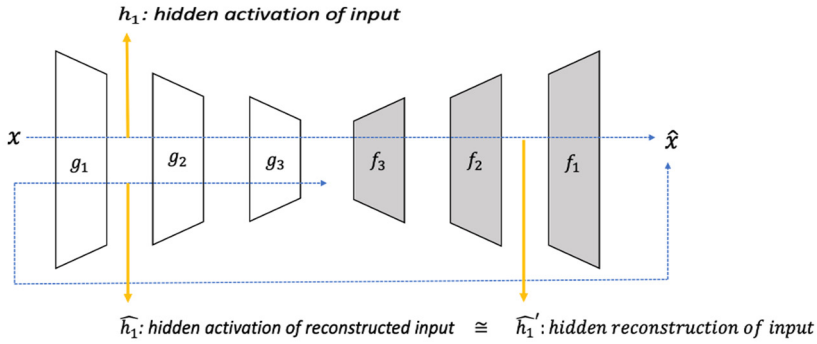

Figure 3. The reconstruction along projection pathway (RaPP) method. The hidden activation of the reconstructed input $\hat{h}_i$ is equivalent to the hidden reconstruction of the input $\hat{h}_i'$. Comparing the hidden activation of input h_i with the hidden activation of the reconstructed input $\hat{h}_i$ can improve novelty detection.

Hidden activations of the reconstructed input are shown to be equivalent to the corresponding hidden reconstructions of the original input. For this computation of hidden reconstruction, RaPP assumes the existence of abstract decoders, which are inverse functions of corresponding encoders. An autoencoder $A = f \circ g$ is a pretrained neural network, such that g and f represent the encoder and the decoder, and l is the number of hidden layers of g and f where $g = g_l \circ \cdots \circ g_2 \circ g_1$ and $f = f_1 \circ f_2 \circ \cdots \circ f_l$. The autoencoder can completely reconstruct the input x: i.e., $x = A(x)$. Partial computation of g and f are as follows:

$$g_{:i} = g_i \circ \cdots \circ g_1$$

$$f_{l:i} = f_i \circ \cdots \circ f_l$$

That is, $g = g_{:l}$ and $f = f_{l:1}$. The i-th hidden activation of the reconstructed input is defined by

$$\hat{g}_{:i}(x) = g_{:i}(A(x))$$

Let l be the number of hidden layers, and let us assume that there exists an abstract decoder $\tilde{f} = \tilde{f_1} \circ \cdots \circ \tilde{f_l}$ such that

$$\tilde{f}(g(x)) = f(g(x)) \tag{1}$$

$$g_{:i}(x) = \left(g_i \circ \tilde{f_i}\right)(g_{:i}(x)) \tag{2}$$

The first condition indicates that an abstract decoder $\tilde{f}$ can, like the decoder f, perfectly reconstruct the input x. The second condition states that $\tilde{f_i}$ is the inverse function of g_i. Then, the i-th hidden reconstruction is defined by

$$\hat{g}'_{:i}(x) = (\tilde{f}_{l:i+1} \circ g)(x)$$

The i-th hidden reconstruction $\hat{g}'_{:i}(x)$ is equal to the i-th hidden activation of the reconstructed input $\hat{g}_{:i}(x)$ as follows:

$$\begin{aligned} \hat{g}'_{:i}(x) &= (\tilde{f}_{l:i+1} \circ g)(x) \\ &= (g_{:i} \circ \tilde{f} \circ g)(x) \qquad \text{(by Equation (2))} \\ &= (g_{:i} \circ A)(x) = \hat{g}_{:i}(x) \qquad \text{(by Equation (1))} \end{aligned}$$

This equation implies that there is no need to implement $\tilde{f}$ to compute the hidden reconstruction. Therefore, the hidden reconstruction computation is possible only with g and f already accessible.

Three steps are involved in detecting novelty samples using the RaPP method. The first step is to train the autoencoder only with normal samples. After the autoencoder is trained, the next step is to project the input and its autoencoder reconstruction onto the hidden spaces to obtain pairs of activation values. Finally, the novelty score is calculated by aggregating these two hidden activations. Before summing up, hidden activations must be normalized to alleviate the dependency between the hidden layers. To normalize hidden activations, RaPP uses the normalized aggregation along pathway (NAP) metric, which is based on the concept of Mahalanobis distance. Experiments have shown that RaPP outperforms reconstruction-based novelty detection for various datasets.

4. The Proposed Model: Extended Autoencoders

Similar to using a reconstruction method to minimize the reconstruction errors, the RaPP method can be further improved by minimizing the hidden reconstruction errors with the mean squared error loss. However, when an autoencoder architecture is deep, one needs at least as many loss functions as there are layers. It is not advisable to consider all deep hidden activations at once, as the original purpose of the autoencoder would be lost. Thus, we focused only on the last hidden activations, the so-called latent variables. Latent variables are usually treated as normally distributed continuous variables [14]. To reflect a distance measure between probability distributions, we must introduce adversarial training loss to the autoencoder.

The core idea of our proposal comes mainly from AAE. In AAE, the discriminator treats samples from a prior distribution as "true" and samples from the latent variables as "false." While the encoder is trained to fool the discriminator, the latent variables become similar to the prior distribution. We apply this concept to our extended autoencoder. Specifically, a discriminator is added on top of the autoencoder to distinguish latent variables of the input from the latent variables of the reconstructed input. Concurrently, the decoder is updated to generate the reconstructed input, which can fool the discriminator. Therefore, the decoder can reconstruct the input to close the distance between the latent variables of the input and its reconstruction. Figure 4 depicts the extended autoencoder architecture. Algorithm 1 summarizes the training methodology of the extended autoencoder.

Algorithm 1 Training the extended autoencoder

Input: Sample x, the number of epochs N
Output: Encoder g, Decoder f, Discriminator D_l
For iteration 1 to N **do**:
1: Autoencoder $A = g \circ f$ update: keep D_l fixed.
2: $L_{mse} \longleftarrow \|x - A(x)\|_2^2$
3: Backpropagate L_{mse} and change g, f
4: Discriminator D_l update: keep g, f fixed.
5: $L_{D_l} \longleftarrow D_l(g(x), 1) + D_l(g(A(x)), 0)$
6: Backpropagate L_{D_l} and change D_l
7: Decoder f update: keep g, D_l fixed.
8: $L_{D_l} \longleftarrow D_l(g(x), 0) + D_l(g(A(x)), 1)$
9: Backpropagate L_{D_l} and change f
End

In the sections below, we first describe the architecture and the training procedure of the proposed model in detail. We also applied our extended autoencoder model to autoencoder variants, such as VAE and AAE. Second, we explain how the extended autoencoder can improve novelty detection with RaPP. Computation of the hidden reconstruction is tractable by assuming the existence of abstract decoders $\tilde{f}$. On the premise that the autoencoder can copy the input x perfectly, there is no need to implement the abstract decoder. However, in practice, the abstract decoder should be implemented because the autoencoder assumption is no longer true.

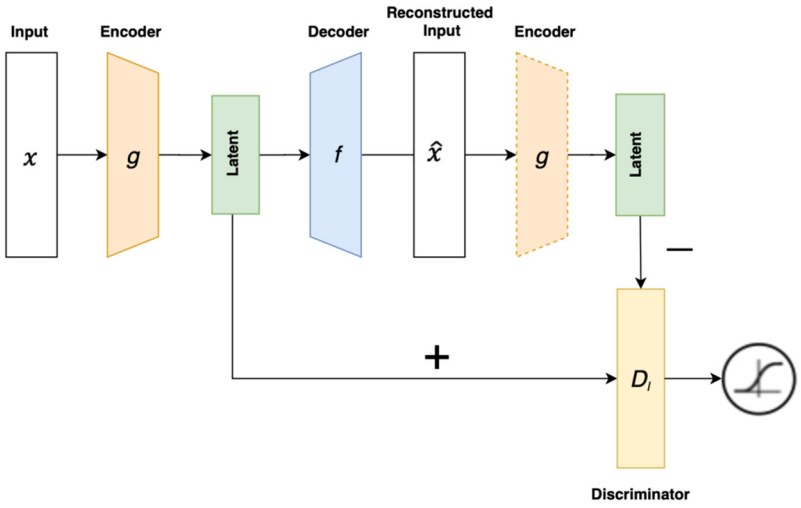

Figure 4. Architecture of the extended autoencoder, which is based on the autoencoder architecture with the addition of the discriminator D_l. The dotted encoder is not updated during training.

4.1. Architecture of Extended Autoencoder

In this section, we describe the architecture of the extended autoencoder and the training procedure. The extended autoencoder consists of an autoencoder and a discriminator. Reconstruction loss and adversarial training loss are described in Equations (3) and (4).

$$L_{mse}(x, g, f) = \|x - f(g(x))\|_2^2 \tag{3}$$

$$L_{D_l}(x, f, D_l) = E_{x\sim p_x}[logD_l(g(x))] + E_{x\sim p_x}[log(1 - D_l((g \circ f)(g(x))))] \tag{4}$$

The first loss function is the mean squared error, which corresponds to the distance between the input and its reconstruction. We jointly optimize the encoder g and the decoder f to minimize the L_{mse} loss function. The second loss function is adversarial training loss to match the latent variables of the reconstructed input to the latent variables of the original input. The discriminator D_l tries to distinguish whether a sample arises from the latent variables of the input or from the latent variables of the reconstructed input. The decoder is trained to fool the discriminator D_l, and the encoder and the decoder can be obtained by optimizing the following full objective:

$$g, f = \arg\min_{g,f}\max_{D_l}(L_{mse}(x, g, f) + L_{D_l}(x, f, D_l))$$

The extended autoencoder can be combined with all variants of the autoencoder; the associated loss functions depend on the variant employed. For example, the extended autoencoder can combine with the VAE to create an extended VAE. Figure 5 depicts the architecture of the extended VAE, which has three loss functions:

$$L_{mse}(x, g, f) = \|x - f(g(x))\|_2^2$$

$$L_{KL}\left(x, g_\mu, g_\sigma\right) = E_{x\sim p_x}\left[\frac{1}{2}(g_\mu(x)^2 + e^{g_\sigma(x)} - g_\sigma(x) - 1)\right]$$

$$L_{D_l}(x, f, D_l) = E_{x\sim p_x}[logD_l(g(x))] + E_{x\sim p_x}[log(1 - D_l((g \circ f)(g(x))))]$$
$$where\ g(x) = g_\mu(x) + e^{\frac{1}{2}g_\sigma(x)} \cdot z,\ z\ \sim\ N(0, 1)$$

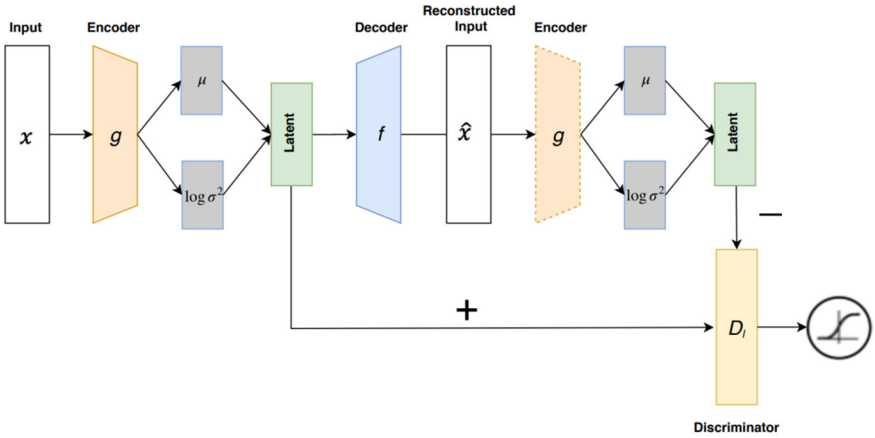

Figure 5. Architecture of the extended variational autoencoder (VAE), which is based on the VAE architecture, with the addition of a discriminator D_l. The dotted encoder is not updated during the training.

In this case, the second loss function is the Kullback–Leiber divergence loss, which is used to confer continuity in the latent space. L_{KL} is the closed form loss in the special case of a Gaussian latent. The VAE encoder consists of two neural networks: g_μ, g_σ. The output of g_μ is the mean, and the output of g_σ is log-variance. Using a reparameterization modification, it is possible for the encoder output $g(x)$ to model the standard Gaussian distribution and the backpropagation. The remaining parts are the same as the case of the basic extended autoencoder. The final objective function of the extended VAE is as follows:

$$g, f = \underset{g,f}{\text{argmin}}\underset{D_l}{\text{max}}(L_{mse}(x, g, f) + L_{KL}\left(x, g_\mu, g_\sigma\right) + L_{D_l}(x, f, D_l))$$

The extended autoencoder can also combine with the AAE. Figure 6 describes the architecture of the extended AAE, which has three loss functions:

$$L_{mse}(x, g, f) = \|x - f(g(x))\|_2^2$$
$$L_{D_z}(x, g, D_z) = E_{z\sim N(0,1)}[logD_z(z)] + E_{x\sim p_x}[log(1 - D_z(g(x))]$$
$$L_{D_l}(x, f, D_l) = E_{x\sim p_x}[logD_l(g(x))] + E_{x\sim p_x}[log(1 - D_l((g \circ f)(g(x))))]$$

Basically, the discriminator D_z tries to distinguish whether a sample arises from the latent variables or from a prior distribution. At the same time, the encoder is updated to fool the discriminator D_z. In a similar way, the discriminator D_l tries to distinguish whether a sample arises from the latent variables of input or from the latent variables of the reconstructed input. The decoder is trained to fool the discriminator D_l. Therefore, the encoder and the decoder can be obtained by optimizing the following full objective:

$$g, f = \underset{g,f}{\text{argmin}}\underset{D_z,D_l}{\text{max}}(L_{mse}(x, g, f) + L_{D_z}(x, g, D_z) + L_{D_l}(x, f, D_l))$$

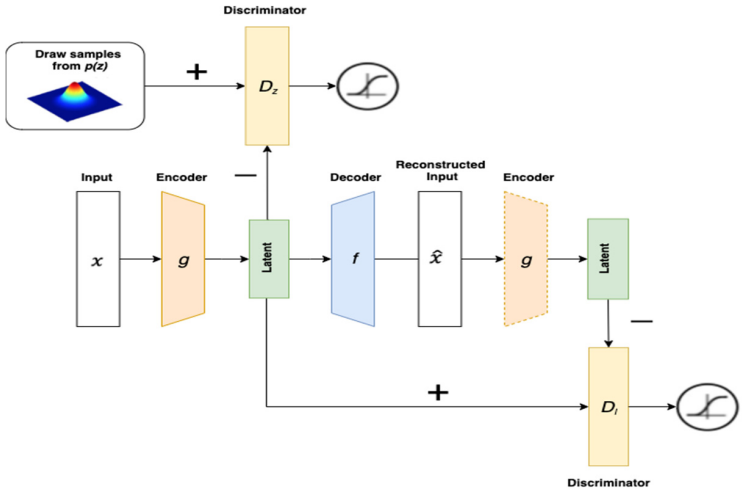

Figure 6. Architecture of the extended AAE, which is based on the AAE architecture, with the addition of discriminator D_l. The dotted encoder is not updated during the training.

4.2. Implementation of Approximate $\tilde{f}$

In this section, we discuss why the abstract decoder $\tilde{f}$ should be implemented to improve the RaPP method, and we show how the adversarial training can force f to be an approximate $\tilde{f}$. To use the RaPP method, the hidden reconstruction of the input must be equal to the hidden activation of the reconstructed input. For this computation, a necessary underlying assumption is that the autoencoder is able to perfectly copy the input x. Practically, however, this assumption is not acceptable. The autoencoder has constraints on its network; thus, it is not capable of perfectly reconstructing the input. The aim of the autoencoder is not to be the identity function but to learn the compressed representation of the input. Because $f(g(x))$ and $\tilde{f}(g(x))$ only approximately copy the input x in their own way, $\tilde{f}(g(x)) = f(g(x))$ is no longer valid. It is still true that RaPP works because the autoencoder at least attempts to copy the input x. However, we expect that using the abstract decoder $\tilde{f}$, rather than decoder f, will improve RaPP performance. The implementation of $\tilde{f}$ can be difficult even though that neural networks are highly flexible frameworks. Thus, the number of criteria required may approach the number of layers needed to force all deep encoder–decoder pairs to be inverses of each other. As such, autoencoder learning is susceptible to failure given the extensive criteria. To avoid this failure, we focus only on the last layer and extend this to the whole encoder–decoder system to implement the approximate $\tilde{f}$.

Since we only consider the last layer, the second condition of $\tilde{f}$ can be expressed as follows:

$$g(x) = (g_l \circ \tilde{f}_l)(g(x))$$

As the second condition of $\tilde{f}$ holds that $\tilde{f}_i = g_i^{-1}$, it is also true that $\tilde{f} = g^{-1}$. Thus, we can substitute $(g_l \circ \tilde{f}_l)$ with $(g \circ \tilde{f})$ in this equation. In addition, the extended autoencoder can be based on a configuration in which a generative autoencoder is trained to match the encoding distribution with the prior distribution. In this paper, we set the encoding distribution $g(x)$ to the standard Gaussian distribution $z \sim N(0,1)$. Finally, we can define an approximate $\tilde{f}$ such that

$$z = (g \circ \tilde{f})(z)$$

This equation states that the latent variable of the input (the left side of the equation) is equal to the latent variable of the reconstructed input (the right side of the equation). To implement the

approximate $\tilde{f}$ that satisfies this equation, we introduce an adversarial criterion to force f to be the approximate $\tilde{f}$, as follows:

$$\min_{f} \max_{D_l} E_{z\sim N(0,1)}[logD_l(z)] + E_{z\sim N(0,1)}[log(1 - D_l((g \circ f)(z))]$$

As the decoder f is updated to confuse the discriminator D_l, the decoder f approximates the abstract decoder $\tilde{f}$. This means that the latent space of the reconstructed input becomes similar to the latent space of the original input. For novelties, however, the hidden reconstruction errors are not reduced, as the autoencoder is trained only with normal samples. Therefore, we can detect novelties better using the RaPP method.

5. Experiments

In this section, we evaluate our proposed extended autoencoder model based on the RaPP method. We use three popular autoencoders: the AE, VAE, and AAE, as baseline models. Our extended autoencoder is also applied to these baseline models. Thus, our comparison included six models in total: the standard versions of the AE, VAE, and AAE and the extended versions of AE, VAE, and AAE.

5.1. Datasets and Problem Setups

The novelty datasets used for novelty detection were collected from Kaggle and the UCI repository. We also carried out novelty detection on the popular benchmark datasets MNIST and F-MNIST. The details of the datasets are described in Table 1. Given that MI-F and MI-V are actually from the same dataset, they share the same features. We treat this dataset as two datasets, as it has two columns that can be used as a novelty class (i.e., machine completed and passed visual inspection).

Table 1. Description of novelty datasets and benchmark datasets.

Name	# Samples	# Features	# Classes	Domain	Novelty Target
MI-F [1]	25,286	58	2	CNC milling	Machine not completed
MI-V [1]	23,125	58	2	CNC milling	Workpiece out-of-spec
EOPT [2]	90,515	20	2	Storage system	System failures
NASA [3]	4687	33	2	Astronomy	Hazardous asteroids
RARM [4]	20,221	6	2	Robotics	Malfunctions
STL [5]	1941	27	7	Steel	Surface defects
OTTO [6]	61,878	93	9	E-commerce	Types of products
SNSR [7]	58,509	48	11	Electric currents	Defective conditions
MNIST [8]	70,000	784	10	Hand-written digits	Digits
F-MNIST [9]	70,000	784	10	Fashion articles	Articles

[1] https://www.kaggle.com/shasun/tool-wear-detection-in-cnc-mill; [2] https://www.kaggle.com/inIT-OWL/high-storage-system-data-for-energy-optimization; [3] https://www.kaggle.com/shrutimehta/nasa-asteroids-classification; [4] https://github.com/narayave/mh5_anomaly_detector; [5] https://www.kaggle.com/uciml/faulty-steel-plates; [6] https://www.kaggle.com/c/otto-group-product-classification-challenge; [7] https://archive.ics.uci.edu/ml/datasets/dataset+for+sensorless+drive+diagnosis; [8] https://yann.lecun.com/exdb/mnist; [9] https://github.com/zalandoresearch/fashion-mnist.

Some datasets (including MI-F, MI-V, EOPT, NASA, and RARM) have only two classes, a normal class and a novelty class. The others (including STL, OTTO, SNSR, MNIST, and F-MNIST) have more than two classes. If there are more than two classes, the performance varies depending on which class is assumed to be novelty. For reliable experiment, it is recommended that each of classes is considered novelty once; in other words, we assigned a single class as the normal class and the remaining classes as the novelty class. We then performed novelty detection as many times as the number of classes and averaged the results. For example, MNIST has 10 classes from "0" to "9", and we then performed novelty detection 10 times to assign each class on MNIST as a normal class. As a result, 10 detection results are generated, and their average value is calculated as the final output of a single trial.

We selected a semi-supervised learning approach for novelty detection [15]. Thus, we provided only normal samples during the training phase and used both normal and novelty samples during the testing phase. Half of the test sets were made up of normal samples, and the other half were novelty samples. After training the autoencoder, we used the RaPP method to calculate the novelty score by normalizing and aggregating the hidden activation values of an input and its autoencoder reconstruction. With this novelty score, we evaluated novelty detection performance using the area under the receiver operating characteristic curve (AUROC) [16]. To alleviate random errors during training, we obtained the AUROC by averaging AUROC scores from 30 trials for novelty datasets and 5 trials for benchmark datasets.

5.2. Implementation Details

All autoencoders have a symmetric architecture in which the encoder and the decoder consist of 10 layers. If the autoencoder had another component such as discriminators, then we would also have to set the discriminator to have 10 layers. The latent space dimension was determined by the number of principal components that explain at least 90% of the variance for novelty datasets. For benchmark datasets, the latent space dimension was set to 20, and we only considered the hidden spaces. Thus, we did not compute the reconstruction error in the input space. We applied Leaky-ReLU [17] activation and batch normalization [18] to all layers, with the exception of the last layer. The Adam optimization algorithm [19] was used to select the best model with the lowest validation loss. As for the implementation details, there was no difference in the implementation between the standard and the extended autoencoders, except that the extended autoencoders had an additional discriminator.

5.3. Results

In this section, our extended autoencoders for AE, VAE, and AAE are referred to as XAE, XVAE, and XAAE, respectively. Table 2 summarizes the RaPP performance of the standard autoencoders and our extended autoencoders trained for 100 epochs with novelty datasets. In Table 2, the best score for each dataset is indicated with underlining. The extended autoencoders provided the best results for all datasets except the MI-F and MI-V datasets. Given that MI-F and MI-V are derived from the same dataset, the extended autoencoder achieved the best results among six of the seven datasets. Thus, this empirical result demonstrates that the extended autoencoder is generally suitable for detecting novelties using the RaPP method.

Table 2. The AUROC (area under the receiver operating characteristic curve) of RaPP with novelty datasets. The best cases are underlined for each dataset, and the number of in parenthesis indicates the standard deviation.

	Standard Autoencoders			**Extended Autoencoders**		
Dataset	**AE**	**VAE**	**AAE**	**XAE**	**XVAE**	**XAAE**
MI-F	0.643 (0.048)	0.655 (0.066)	0.639 (0.052)	0.637 (0.058)	0.653 (0.060)	0.646 (0.050)
MI-V	0.902 (0.012)	0.901 (0.016)	0.896 (0.013)	0.890 (0.015)	0.896 (0.017)	0.894 (0.010)
EOPT	0.629 (0.018)	0.593 (0.007)	0.628 (0.018)	0.630 (0.019)	0.597 (0.006)	0.631 (0.023)
NASA	0.717 (0.031)	0.723 (0.027)	0.719 (0.025)	0.729 (0.028)	0.735 (0.027)	0.727 (0.024)
RARM	0.647 (0.072)	0.651 (0.046)	0.640 (0.065)	0.656 (0.050)	0.648 (0.051)	0.673 (0.056)
STL	0.819 (0.053)	0.818 (0.051)	0.817 (0.058)	0.824 (0.049)	0.821 (0.049)	0.821 (0.056)
OTTO	0.829 (0.010)	0.838 (0.012)	0.831 (0.011)	0.833 (0.011)	0.838 (0.011)	0.829 (0.012)
SNSR	0.988 (0.003)	0.988 (0.003)	0.989 (0.003)	0.988 (0.003)	0.989 (0.003)	0.990 (0.003)

However, note that these results do not include the best cases since the hyperparameters including the number of hidden layers has not been optimized for each dataset. In this respect, we have investigated the performance of RaPP by varying the number of hidden layers with MI-F and MI-V datasets, and the results are presented in Tables 3 and 4. The extended autoencoder denoted as XAE achieves the best score when the number of hidden layers is set to 2 on MI-F and 6 on MI-V, respectively.

Table 3. AUROC with different number of hidden layers on MI-F dataset. The number of in parenthesis indicates the standard deviation.

# Hidden	Standard Autoencoders			Extended Autoencoders		
Layers	AE	VAE	AAE	XAE	XVAE	XAAE
2	0.820	0.696	0.820	0.835	0.744	0.820
	(0.010)	(0.011)	(0.013)	(0.012)	(0.020)	(0.021)
4	0.829	0.762	0.819	0.831	0.743	0.821
	(0.010)	(0.016)	(0.014)	(0.013)	(0.024)	(0.012)
6	0.771	0.734	0.774	0.773	0.745	0.787
	(0.032)	(0.036)	(0.047)	(0.024)	(0.043)	(0.022)
8	0.673	0.701	0.686	0.639	0.725	0.693
	(0.019)	(0.030)	(0.077)	(0.065)	(0.056)	(0.035)
10	0.643	0.655	0.639	0.637	0.653	0.646
	(0.012)	(0.016)	(0.013)	(0.015)	(0.017)	(0.010)

Table 4. AUROC with different number of hidden layers on MI-V dataset. The number of in parenthesis indicates the standard deviation.

# Hidden	Standard Autoencoders			Extended Autoencoders		
Layers	AE	VAE	AAE	XAE	XVAE	XAAE
2	0.896	0.882	0.896	0.892	0.892	0.900
	(0.012)	(0.012)	(0.012)	(0.007)	(0.011)	(0.012)
4	0.889	0.903	0.889	0.892	0.908	0.895
	(0.009)	(0.013)	(0.08)	(0.007)	(0.006)	(0.006)
6	0.907	0.908	0.900	0.910	0.907	0.906
	(0.010)	(0.008)	(0.010)	(0.010)	(0.011)	(0.006)
8	0.899	0.902	0.896	0.899	0.900	0.897
	(0.012)	(0.012)	(0.011)	(0.011)	(0.013)	(0.015)
10	0.902	0.901	0.896	0.890	0.896	0.894
	(0.012)	(0.016)	(0.013)	(0.015)	(0.017)	(0.010)

Table 5 summarizes the RaPP performance with the MNIST and F-MNIST image datasets. The extended autoencoders required only two epochs to achieve performance similar to that of standard autoencoders, which are trained for 100 epochs. For the F-MNIST dataset, the extended AAE (trained for only two epochs) outperformed all standard autoencoders trained for 100 epochs, and for the MNIST dataset, the proposed model showed almost the same performance as that of standard autoencoders. Of course, since the execution time of a single epoch can vary depending on autoencoder methods, it is necessary to check the time. Figure 7 shows the execution time in seconds of a single training epoch. As seen in the figure, during a single epoch, each of the extended autoencoders take approximately twice as long as its corresponding standard autoencoder; in other words, the time required for two epochs of the extended autoencoder is almost the same as the time required for four epochs of the standard autoencoder. After all, in the training process, the extended autoencoders outperform the standard autoencoders within a short training time for both MNIST and F-MNIST datasets. Unfortunately, this high efficiency was not observed for the other Kaggle and UCI datasets in Table 2. In any case, this is especially important when concept drift is a possibility [20], as the ability to adapt quickly depends on training efficiency.

Table 5. AUROC of RaPP with benchmark datasets. The number of in parenthesis indicates the standard deviation.

	Standard (4 epochs)			Standard (100 epochs)			Extended (2 epochs)		
Dataset	**AE**	**VAE**	**AAE**	**AE**	**VAE**	**AAE**	**XAE**	**XVAE**	**XAAE**
MNIST	0.959	0.962	0.960	0.974	0.971	0.970	0.967	0.971	0.966
	(0.008)	(0.007)	(0.007)	(0.004)	(0.004)	(0.006)	(0.006)	(0.003)	(0.005)
F-MNIST	0.928	0.919	0.926	0.931	0.928	0.928	0.933	0.933	0.934
	(0.006)	(0.006)	(0.008)	(0.006)	(0.005)	(0.006)	(0.005)	(0.004)	(0.006)

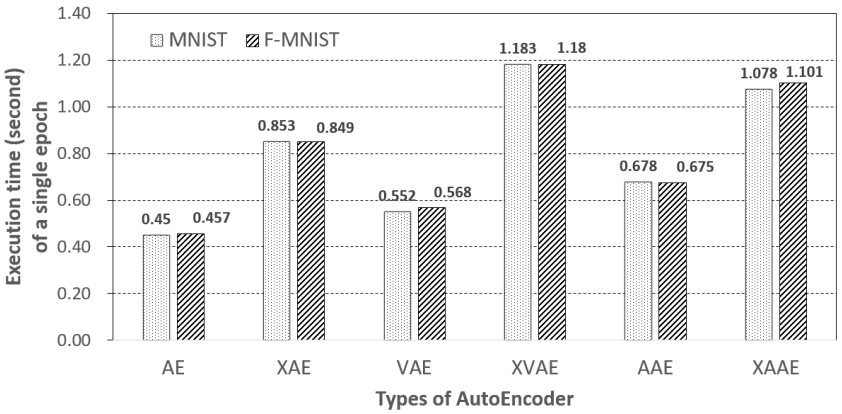

Figure 7. The average execution time of a single training epoch for MNIST (Modified National Institute of Standards and Technology) and F-MNIST (Fashion Modified National Institute of Standards and Technology) datasets.

Additionally, we found that the performance of the extended autoencoders tended to decrease as the number of epochs increased. This issue may be caused by difficulties associated with GAN training. Mode collapse [21] is a well-known problem that occurs when the generator learns to map several different input values to the same output point. In our case, the decoder was trained to reconstruct only images that were capable of deceiving the discriminator. As a future work, other improved techniques for training GANs [22] can be applied to prevent mode collapse in image datasets.

6. Conclusions

In this work, novelty detection with RaPP was improved using extended autoencoders in which we focused on the hidden activations. To reduce the hidden reconstruction error for normal samples, we introduced an adversarial network to the autoencoder. This additional adversarial network ensures that the latent variables of the reconstructed input match the latent variables of the original input. In this process, the decoder closes to the abstract decoder, which enables computation of the hidden reconstruction. Therefore, we no longer need to assume that the autoencoder can completely reconstruct the input. Our empirical results showed that when using the RaPP method for novelty detection, the extended autoencoders outperformed the standard autoencoders for diverse novelty detection datasets from Kaggle and the UCI repository and improved efficiency in benchmark datasets.

In addition to the modification of the autoencoder, the RaPP method can be improved further in future work. Here, we selected the model with the lowest reconstruction error on the validation dataset as the best model. This effectively provided an "early stop" framework to prevent the model from being overfit to the training dataset. However, the hidden reconstruction error to select the best model must be included, as RaPP leverages not only the reconstruction error but also the hidden reconstruction error.

Author Contributions: All authors made contributions to this work. H.-j.K. contributed to the organization of the research as well as the final manuscript preparation. S.Y.S. proposed the original idea and wrote the

original draft of this work. Conceptualization, S.Y.S. and H.-j.K.; methodology, S.Y.S.; software, S.Y.S.; validation, S.Y.S. and H.-j.K.; formal analysis, S.Y.S. and H.-j.K.; investigation, S.Y.S.; resources, H.-j.K.; data curation, S.Y.S.; writing—original draft preparation, S.Y.S.; writing—review and editing, H.-j.K.; visualization, S.Y.S.; supervision, H.-j.K.; project administration, H.-j.K.; and funding acquisition, H.-j.K. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2018-0-01417) supervised by the IITP (Institute for Information & communications Technology Promotion), and was also supported by grant (20AUDP-B100356-06) from Urban Architecture Research Program funded by Ministry of Land, Infrastructure and Transport of the Korean government.

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

References

1. Pimentel, M.A.; Clifton, D.A.; Clifton, L.; Tarassenko, L. A review of novelty detection. *Signal Process.* **2014**, *99*, 215–249. [CrossRef]
2. Chalapathy, R.; Chawla, S. Deep learning for anomaly detection: A survey. *arXiv* **2019**, arXiv:1901.03407.
3. Hoffmann, H. Kernel PCA for novelty detection. *Pattern Recognit.* **2007**, *40*, 863–874. [CrossRef]
4. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv. (Csur)* **2009**, *41*, 1–58. [CrossRef]
5. Kim, K.; Shim, S.; Lim, Y.; Jeon, J.; Choi, J.; Kim, B.; Yoon, A. RAPP: Novelty detection with reconstruction along projection pathway. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 15 February 2020.
6. Makhzani, A.; Shlens, J.; Jaitly, N.; Goodfellow, I.; Frey, B. Adversarial autoencoders. In Proceedings of the International Conference on Learning Representations, San Juan, PR, USA, 2–4 May 2016.
7. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
8. Maaten, L.V.D.; Hintog, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
9. Diederik, P.K.; Welling, M. Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
10. An, J.; Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Spec. Lect. IE* **2015**, *2*, 1–18.
11. Schreyer, M.; Sattarov, T.; Schulze, C.; Reimer, B.; Borth, D. Detection of accounting anomalies in the latent space using adversarial autoencoder neural networks. *arXiv* **2019**, arXiv:1908.00734.
12. Perera, P.; Nallapati, R.; Xiang, B. Ocgan: One-class novelty detection using gans with constrained latent representations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 2898–2906.
13. Pidhorskyi, S.; Almohsen, R.; Doretto, G. Generative probabilistic novelty detection with adversarial autoencoders. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 6822–6833.
14. Collins, L.M.; Lanza, S.T. *Latent Class and Latent Transition Analysis: With Applications in the Social, Behavioral, and Health Sciences*; John Wiley & Sons: Hoboken, NJ, USA, 2009; p. 718.
15. Kiran, B.R.; Thomas, D.M.; Parakkal, R. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *J. Imaging* **2018**, *4*, 36. [CrossRef]
16. Fan, J.; Upadhye, S.; Worster, A. Understanding receiver operating characteristic (ROC) curves. *Can. J. Emerg. Med.* **2006**, *8*, 19–20. [CrossRef] [PubMed]
17. Xu, B.; Wang, N.; Chen, T.; Li, M. Empirical evaluation of rectified activations in convolutional network. *arXiv* **2015**, arXiv:1505.00853.
18. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 448–456.
19. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.

20. Gama, J.; Zliobaite, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A survey on concept drift adaptation. *ACM Comput. Surv. (CSUR)* **2014**, *46*, 1–37. [CrossRef]
21. Goodfellow, I. NIPS 2016 tutorial: Generative adversarial networks. *arXiv* **2016**, arXiv:1701.00160.
22. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved techniques for training gans. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 2234–2242.