
One-Class Feature Learning Using Intra-Class Splitting

Patrick Schlachter

Yiwen Liao

Bin Yang

Institute of Signal Processing and System Theory, University of Stuttgart, Germany

Abstract

This paper proposes a novel generic one-class feature learning method which is based on intra-class splitting. In one-class classification, feature learning is challenging, because only samples of one class are available during training. Hence, state-of-the-art methods require reference multi-class datasets to pretrain feature extractors. In contrast, the proposed method realizes feature learning by splitting the given normal class into *typical* and *atypical* normal samples. By introducing *closeness loss* and *dispersion loss*, an intra-class joint training procedure between the two subsets after splitting enables the extraction of valuable features for one-class classification. Various experiments on three well-known image classification datasets demonstrate the effectiveness of our method which outperformed other baseline models in 25 of 30 experiments.

1 Introduction

Over the last few years, supervised classification of high-dimensional raw data such as images was successfully solved by deep learning. However, few research was conducted on special binary classification problems such as fault detection in which samples from one class are not available during training or not representative for their class. In such problems, one-class classification is applied.

One-class classification is a subfield in machine learning, aiming to identify data of a given normal class amongst all abnormal data by learning from a training dataset consisting merely of data from the given normal class. It is more challenging than binary or multi-class classification in which samples from all classes are available during training. Various one-class classifiers were proposed and successfully applied to a wide range

of applications including fault detection, novelty detection or anomaly detection [1, 2]. Some prior work by Schölkopf et al. [3] or Tax et al. [4] proposed state-of-the-art one-class classifiers. They are all characterized by a tight decision boundary around the training samples of the normal class in order to reject abnormal samples of different kinds during inference.

To achieve a tight decision boundary, one-class classifiers require an input feature space that fulfills the following two conditions. First, features of normal data must be compactly distributed. We call this requirement *closeness*. Second, normal and abnormal data must have large distances between each other in the feature space. This requirement is called *dispersion*. Both requirements are comparable to the linear discriminant analysis in clustering in which the sample distances within a cluster are minimized and the sample distances between clusters are maximized [5].

As the stated conditions are typically not fulfilled for high-dimensional data such as image data, the performance of state-of-the-art one-class classifiers is quite limited. Hence, a feature extraction method is necessary for such classifiers to transform raw data into a suitable latent feature space satisfying the closeness and dispersion requirements. However, this is challenging, because abnormal data samples are not available during training. For this reason, few work was done on feature learning for one-class classification.

State-of-the-art feature learning methods for one-class classification are supported by samples from other classes. In particular, a reference dataset is used to pretrain a model on many other classes [6]. The underlying assumption is that real outliers might be contained in these classes. However, this assumption is too strong, since the number of other classes is unlimited. If the reference dataset is not representative for real outliers, then the decision boundary of a one-class classifier will be too loose. Therefore, the choice of a reference dataset is crucial. Indeed, a meaningful multi-class reference dataset is typically not available due to the nature of the one-class classification problem.

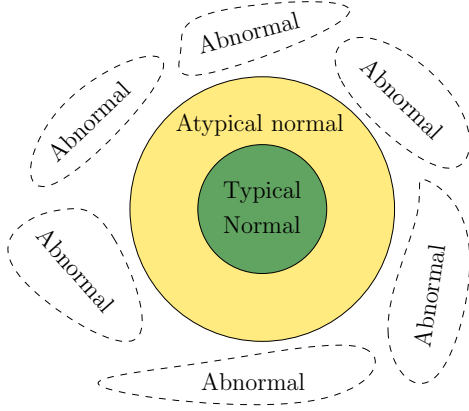


Figure 1: In a certain latent space, typical normal data (*green*) is clustered and surrounded by atypical normal data (*yellow*). Abnormal data (*dashed lines*) is located outside this area and can be arbitrarily distributed.

In this paper, our goal is to learn a latent space that fulfills both closeness and dispersion requirements using only normal data. As a result, latent representations are optimized to the training class and not biased to any outliers. A solution for this goal does not only enable the use of conventional one-class classifiers to the learned feature space, but has also high potential for unsupervised learning and open set recognition [7]. Intuitively, we assume that in a certain latent space, normal data is clustered and abnormal data distributes around it. Figure 1 illustrates this scenario.

In order to find such a latent space, our first idea is to split a normal training set into two subsets in an unsupervised manner and by using a similarity metric. One subset includes typical normal data, meaning the majority of a given training dataset, and the other subset consists of atypical normal data, meaning the minority of the training set. Hence, no reference dataset is necessary in contrast to [6]. By using an joint training procedure to maximize the dispersion between the typical and atypical normal data, a tight decision boundary around the normal data in the latent feature space can be achieved. This is our second key idea.

One meaningful similarity metric for image data is the structural similarity (SSIM) [8]. For example, Figure 2 illustrates the division of exemplary classes into typical and atypical normal samples using SSIM. In more detail, an autoencoder is first trained on the whole training set. Subsequently, the similarity metric between reconstructed and original data is calculated. Finally, the data with higher similarity is considered to be typical, whereas the samples with lower similar-



Figure 2: *Green*: Exemplary typical normal samples with a higher SSIM score compared to their reconstructions. *Yellow*: Exemplary atypical normal samples having a lower SSIM score compared to their reconstructions. The atypical normal samples are more difficult to recognize or have more redundant details compared to typical normal ones.

ity are considered to be atypical normal data.

In particular, this work is based on two assumptions. First, the majority of a given dataset represents the most common characteristics of the normal data. Second, in a certain latent space, abnormal data has more common features with the atypical normal data rather than typical normal data. These assumptions are weaker than those in prior work such as [6], since we do not make any assumptions about unseen abnormal data. Instead, only the intra-class information from the normal class is utilized.

Our main contributions in this paper are the following:

- *Intra-class splitting* is introduced which is the key to solve feature learning for one-class classification. However, it is a generic method and not limited to one-class feature learning.
- A novel intra-class joint training strategy: Both typical and atypical normal subsets have their individual objectives as well as common objectives. These objectives are finally reformulated as a joint optimization problem.
- We empirically prove that *closeness* plays a key role in feature learning for one-class problems. Furthermore, it is shown that the combination of *closeness* and *dispersion* can achieve an even better feature extraction performance.

The remaining parts of this paper are structured as follows. In the next section, we refer to related work in the field of one-class classification and deep feature learning. Section 3 presents our main contribution, a new method of deep feature learning for one-class classification. The performance of this new method is analyzed in various experiments in Section 4. Finally, we conclude our work in Section 5.

2 Related Work

2.1 One-Class Classification

A large amount of research was conducted in methods and applications of one-class classification [1, 2, 9]. One popular boundary-based one-class classifier is the One-Class Support Vector Machine (OCSVM) by Schölkopf et al. [3]. It maximizes the distance between a hyperplane and the origin of the feature space while ensuring that all training samples lie on the opposite side of the origin. OCSVMs showed their state-of-the-art performance in different fields, e.g. data retrieval [10, 11], medical image processing [12] or fault detection [13]. Nonetheless, OCSVMs are very sensitive to the form of its input as shown in [10]. In other words, feature extraction is necessary for OCSVMs in many applications.

2.2 Deep feature learning for one-class classification

Deep neural networks are a powerful tool in machine learning for various tasks such as multi-class classification [14, 15]. One main reason is that deep neural networks are able to efficiently extract features from raw data with suitable objectives.

Due to this characteristic, deep neural networks are often used as feature extractors for multi-class classification problems [16, 17, 18, 19]. For instance, Ross B. Girshick et al. [20] used a deep neural network to extract features from image patches and then utilized these extracted features to train SVMs for multi-class classification. A recent work of [21] utilized a deep neural network to minimize the distances between the given normal samples and an arbitrary point in the latent space with using these distances as metric to judge if a new sample belongs to the normal class.

Few work was done on feature learning for one-class classification. Chalapathy et al. proposed a method for robust anomaly detection [22]. It is based on a robust autoencoder which combines a classical autoencoder with the robust PCA. Perera et al. proposed a deep learning-based solution for feature learning in one-class classification [6]. Their method is based on the combination of a pretrained model and a learned model. The pretrained model requires a multi-class reference dataset which is hard to acquire.

3 Proposed Method

3.1 Overview

As mentioned in the introduction, the goal is to learn a model which extracts features based on one class only, fulfilling both closeness and dispersion in the latent space. Therefore, the basic idea is to split the given normal data into typical and atypical normal samples and to use a three-stage joint training procedure. First, the foundations and strategy of intra-class splitting are explained. Subsequently, three desired characteristics of the latent space are briefly described. Then, the corresponding loss functions are introduced. Finally, an autoencoder-based network considering intra-class dispersion and closeness is presented.

3.2 Intra-Class Splitting

In a given normal class, not all samples are representative for this class as illustrated in Figure 2. For example, too many details in images may mislead the training of a classifier if those details are not necessary for the correct classification of the class. Accordingly, it is assumed that a given normal dataset is composed of two parts, typical normal samples and atypical normal samples. Typical normal samples are the most representative for the normal class and correspond to the majority of the given dataset. In contrast, the remaining samples from the given dataset are considered as atypical normal samples which may mislead the learning of a one-class classifier.

Splitting the given dataset remains a challenging task, because there is no similar prior work to the best of our knowledge. An intuitive approach is to realize the splitting by neural networks with a bottleneck structure. By using a compression-decompression process such as in an autoencoder, input data is first transformed into a low-dimensional representation with information loss and then mapped back to the original data space. Hence, only the most important information of the given dataset is well maintained during this process. Accordingly, the samples contain more representative features if they are better reconstructed.

Formally, a given normal dataset χ is split by using a predefined similarity metric $f(\mathbf{x}, \hat{\mathbf{x}})$ and a ratio ρ where $\hat{\mathbf{x}}$ is the reconstruction of a sample $\mathbf{x} \in \chi$. In particular, the first $\rho\%$ samples with the lowest similarity scores are considered as atypical normal samples χ_{atypical} , while the others are considered as typical normal samples χ_{typical} .

3.3 Desired Characteristics of the Latent Space

Closeness. Input data typically distributes along a manifold in the original high-dimensional space. Therefore, it should naturally distribute in a small region in a low-dimensional latent space. In this low-dimensional space, all latent representations of normal data should be as close to each other as possible. In other words, the intra-class latent representations of normal data must have a high closeness among themselves.

Dispersion. While closeness is necessary for intra-class latent representations of normal data, abnormal data must be as far away from normal data as possible in the latent space. Hence, abnormal data should have a high dispersion comparing with normal data in this space. As mentioned in the introduction, a training set is assumed to consist of typical and atypical normal data. As illustrated in Figure 1, atypical normal samples are forced to distribute far away from typical normal data. Hence, the unseen abnormal data that behave more like these atypical normal data will also lie far from the normal data.

Reconstruction information. While transforming raw data to the latent space, the information contained in high-dimensional data should be retained in order to ensure that the latent space is a compressed representation of the raw data. Thus, we utilize the autoencoder structure as constraints on maintaining high-dimensional information. Equivalent to the ordinary autoencoder, the reconstruction ability of latent representations is quantified by a reconstruction loss.

3.4 Training

As closeness and dispersion are opposite to each other, we designed an joint training procedure for feature learning. It is performed by three stages using the loss functions defined in Section 3.5.

3.4.1 First Stage

All data from the training set is used to train an autoencoder only with a regular reconstruction loss \mathcal{L}_{rec} . During this stage, the network is considered to be a deep convolutional autoencoder. Note that no constraints are performed on the latent representation at this stage.

3.4.2 Second Stage

Once the network is trained at the first stage, the similarity between the reconstructed data and the original data is calculated. Afterwards, according to a predefined ratio ρ , the first $\rho\%$ of the data with the lowest similarity scores are chosen as the atypical normal samples, whereas the remaining data is typical normal.

3.4.3 Third Stage

To learn proper latent representations, the model is trained with different objectives regarding typical and atypical normal samples, respectively.

More precisely, the closeness loss \mathcal{L}_{cls} acts as the objective for only typical normal samples to force the corresponding latent representations to be close to each other. In contrast, one dispersion loss $\mathcal{L}_{\text{disp},1}$ is designed for only atypical normal samples to keep them far away from each other in the latent space. Another dispersion loss $\mathcal{L}_{\text{disp},2}$ is taken as an objective by both typical and atypical normal samples to maximize the distances between typical and atypical latent representations. Moreover, a regular reconstruction loss is applied as the objective for all the training samples to retain the high-dimensional information.

The two different training subsets, typical and atypical normal samples, have their own objectives. Nevertheless, all loss terms are reformulated in one unified loss function defined as

$$\mathcal{L}_{\text{rec}} + \alpha\mathcal{L}_{\text{cls}} + \beta_1\mathcal{L}_{\text{disp},1} + \beta_2\mathcal{L}_{\text{disp},2}, \quad (1)$$

where α , β_1 and β_2 balance the ratio between all four loss terms.

3.5 Loss Functions

First, the notations used in the loss functions are introduced.

- The tensor \mathbf{X} denotes a batch of images from the training set. Then B is the number of images in one minibatch. \mathbf{X}_j represents the j -th image in the given minibatch. \mathbf{x}_j is the vectorized form of \mathbf{X}_j .
- The matrix \mathbf{Z} denotes a batch of latent representations with L being the dimension of the latent vectors. The vector \mathbf{z}_j is the j -th latent representation of \mathbf{Z} .
- The mapping from the raw data space to the latent space performed by the encoder is denoted as $f_{\text{enc}}(\cdot)$. Accordingly, the decoder's mapping from

the latent space to the original data space is denoted as $f_{\text{dec}}(\cdot)$. The function $f(\cdot)$ describes a cascade of these two mappings. The parameters to be learned are denoted as $\Theta = \{\Theta_{\text{enc}}, \Theta_{\text{dec}}\}$.

Based on these notations, we define three loss functions for each of the three characteristics of the latent space described in Section 3.3.

3.5.1 Reconstruction Loss

An ordinary mean squared error (MSE) loss is used as reconstruction loss

$$\mathcal{L}_{\text{rec}} = \frac{1}{B} \sum_{j=1}^B \|\mathbf{x}_j - \hat{\mathbf{x}}_j\|^2, \quad (2)$$

where $\hat{\mathbf{x}}_j = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}_j; \Theta_{\text{enc}}); \Theta_{\text{dec}})$ is the reconstruction of \mathbf{x}_j . This loss defines the constraint that the original data can be well reconstructed from the latent space. Correspondingly, the high-dimensional information is retained during the training. Moreover, this term helps to avoid too trivial solutions, e.g. all-zero latent representations.

3.5.2 Closeness Loss

The closeness requirement states that each typical normal latent representation \mathbf{z}_j should have a small distance to any another randomly chosen typical normal latent representation $\mathbf{z}_{i \neq j}$, where arbitrary distance metrics can be used for this purpose. Specifically, a metric based on the Euclidean distance

$$\mathcal{L}_{\text{cls}} = \frac{1}{B} \sum_{j=1}^B \sqrt{\frac{1}{L} \|\mathbf{z}_j - \mathbf{z}_{i \neq j}\|^2}, \quad (3)$$

with $\mathbf{z}_j = f_{\text{enc}}(\mathbf{X}_j; \Theta_{\text{enc}})$ and $\mathbf{z}_{i \neq j} = f_{\text{enc}}(\mathbf{X}_{i \neq j}; \Theta_{\text{enc}})$ is proposed. This minimization has the same expectation as if we calculated all distances from other latent representations, since the mini-batch stochastic gradient decent method is used in this work.

The above loss function alone has the tendency to map all raw data, both from the normal and abnormal class, to a very small region in the latent space. Figure 3 shows this effect for the dataset Fashion-MNIST. As a result, latent representations of normal and abnormal data distribute in a mixture.

3.5.3 Dispersion Loss

The dispersion loss between these typical and atypical normal subsets is defined as

$$\begin{aligned} \mathcal{L}_{\text{disp}} = & \underbrace{\beta_1 \cdot \left(-\frac{1}{B} \sum_{j=1}^B \sqrt{\frac{1}{L} \|\mathbf{z}_{j,\text{atypical}} - \mathbf{z}_{i \neq j,\text{atypical}}\|^2} \right)}_{\mathcal{L}_{\text{disp},1}} \\ & + \underbrace{\beta_2 \cdot \left(-\frac{1}{B} \sum_{j=1}^B \sqrt{\frac{1}{L} \|\mathbf{z}_{j,\text{atypical}} - \mathbf{z}_{j,\text{typical}}\|^2} \right)}_{\mathcal{L}_{\text{disp},2}}, \end{aligned} \quad (4)$$

where $\mathbf{z}_{j,\text{atypical}} = f_{\text{enc}}(\mathbf{X}_{j,\text{atypical}})$ and $\mathbf{z}_{j,\text{typical}} = f_{\text{enc}}(\mathbf{X}_{j,\text{typical}})$. $\mathbf{X}_{j,\text{typical}}$ denotes randomly chosen typical normal samples. The minimization of $\mathcal{L}_{\text{disp},1}$ forces latent representations of atypical normal data to be far away from each other which results in a dispersion among atypical normal latent representations. Moreover, minimizing $\mathcal{L}_{\text{disp},2}$ leads to large distances between typical and atypical normal samples.

3.6 Network Architecture

The proposed feature learning method is modeled by a deep convolutional autoencoder as shown in Figure 4. Both encoder and decoder are used during the training procedure. Once the model is trained, only the encoder part is utilized as a feature extractor.¹

The encoder is modified from AlexNet [23]. The fully connected layers are replaced by one additional convolutional layer. Besides, we concatenate outputs of the second, third and fourth convolutional layer to the output of the second max-pooling layer as one input for the sixth convolutional layer. Each of the first five convolutional layers is followed by a batch-normalization layer as shown in Figure 5.

Figure 6 illustrates the decoder which is almost symmetric to the encoder: Every convolutional layer is replaced by a transposed convolutional layer. A convolutional layer with sigmoid as activation after the fifth transposed convolutional layer serves as the output layer.

The details of the entire neural network structure are listed in the appendix.

¹The original output of the encoder is a tensor for each sample. Hence, the extracted features are reshaped to vectors before feeding them into a one-class classifier.

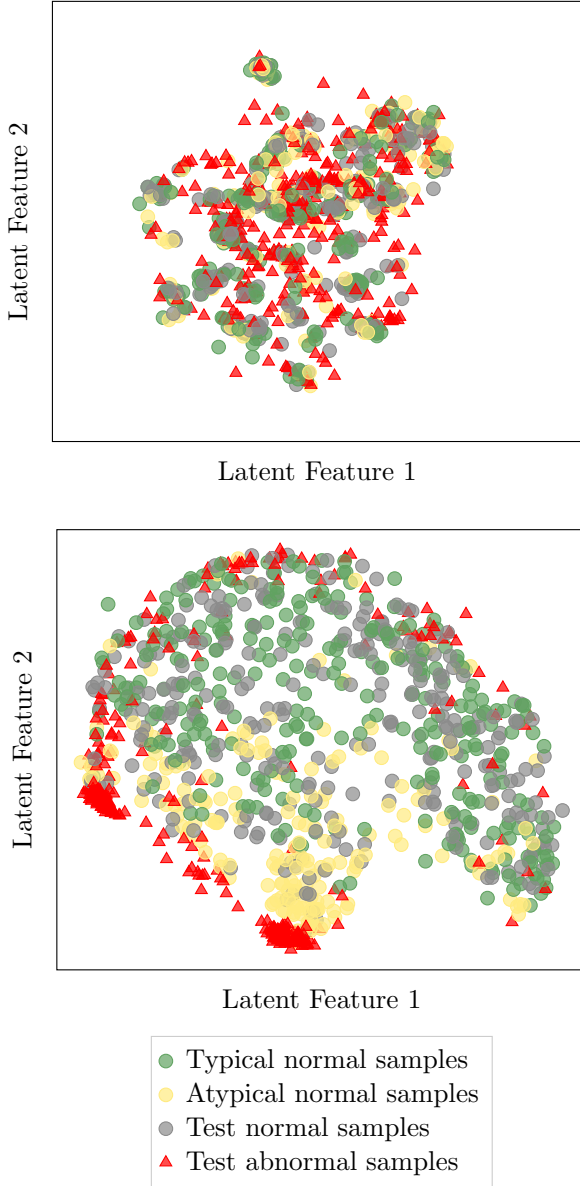


Figure 3: If only trained with the closeness loss, the network tends to learn a too simple function to map all samples to a small region (*top*). Accordingly, the normal data (*green, yellow and gray*) and abnormal data (*red*) distribute in an indistinguishable mixture. However, with the dispersion loss (*bottom*), the typical normal samples (*green*) distribute compactly, while the atypical normal samples (*yellow*) distribute far away or around the typical normal ones. Finally, the abnormal (*red*) samples are thus also far away from the normal ones.

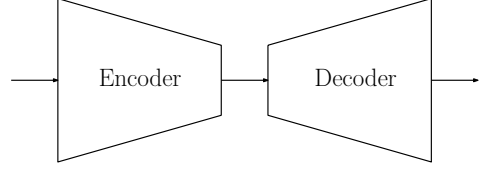


Figure 4: The proposed method is based on an autoencoder.

4 Experiments

This section presents experiments conducted to evaluate the proposed methods. The proposed method was used to learn latent features of different datasets which were subsequently fed into a one-class classifier. Although OCSVMs achieve state-of-the-art performance on low-dimensional features, they have limited classification performance for complex high-dimensional raw data such as images. Hence, an OCSVM was used to perform one-class classification on extracted features. In the following experiments, we focus on image-level one-class classification. Thereby, images of one class were selected as normal data, while images of all remaining classes were considered as abnormal data and were not available during training.

4.1 Experimental Setup

4.1.1 Datasets

The proposed method was evaluated on the datasets MNIST [24], Fashion-MNIST [25] and CIFAR-10 [26]. All three datasets are composed of 10 different classes. The number of training data for all experiments was set to 4000. The numbers of normal and abnormal samples in the test set were 1000 and 9000, respectively². For example, in one individual experiment, if digit 2 from dataset MNIST was considered as normal data, then the training set consisted of 4000 different images of digit 2. Furthermore, the test set consisted of 9000 other images of digit 2 and 1000 images of the other nine digits from 0 to 9 except of digit 2.³

Before training, all image pixels were normalized to the range $[0, 1]$ by min-max scaling. Note that images from MNIST and Fashion-MNIST have the size of 28×28 , while images from CIFAR-10 have the size of $32 \times 32 \times 3$.

²The MNIST dataset has about but not exact 1000 normal and 9000 abnormal samples for each individual experiment.

³There are 20% samples of the test set randomly selected as validation set to choose a proper classification threshold for the metric balanced accuracy.

4.1.2 Hyperparameters

The proposed model was implemented with TensorFlow and Keras [27]. L2-regularization was used for every convolutional layer with a regularization parameter of 10^{-6} . The training minibatch size was 64 for all experiments. The ratio ρ for choosing atypical normal samples was set to 10, unless otherwise stated. α , β_1 and β_2 defined in Section 3.5 were chosen to be 1, 10^{-5} and 10^{-5} , unless otherwise specified. The dimension of the latent space was set to 64. The one-class classifier used in this work was the one-class support vector machine (OCSVM) with $\nu = 0.1$ and $\gamma = \frac{1}{\#features}$.

4.1.3 Baseline Models

To the best of our knowledge, few prior works were designed for image-level one-class feature extraction. Therefore, the following conventional feature extraction methods were used as shallow baseline models:

- *Original features (Original)*: The original images were vectorized as the input for OCSVM. Correspondingly, samples from MNIST and Fashion-MNIST have 784 features and those from CIFAR-10 have 3072 features.
- *Principle Component Analysis (PCA)*: The first 64 PCA components of each input image were used as the input for OCSVM.
- *Histogram of oriented gradients (HOG)*: HOG features [28] for each sample were extracted as the input for OCSVM. The length of HOG features for the samples from MNIST and Fashion-MNIST was 144 and for CIFAR-10 was 324.

Furthermore, the following deep feature extraction methods were considered as baselines in this work:

- *Pretrained Features (ImageNet)*: Features extracted by a VGG19 [29] pretrained on ImageNet [30] were used as the input for OCSVM.
- *Convolutional Autoencoder (CAE)*: Features extracted by a regular autoencoder without any constraints on the latent space were used for subsequent one-class classification.
- *Autoencoder with closeness loss (CLS)*: An autoencoder was trained with the proposed closeness loss as a regularization but without intra-class splitting to extract features for one-class classification.

Note that both the baselines CAE and CLS shared the same architecture with the proposed method.

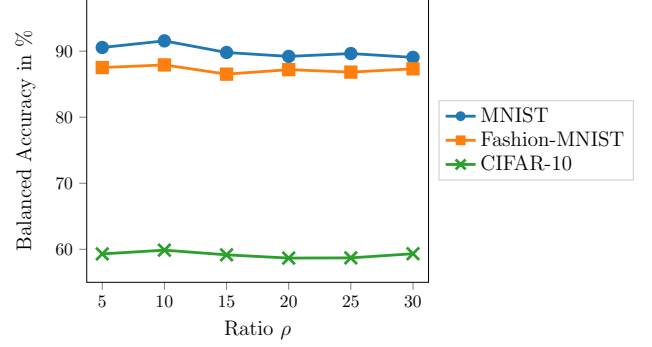


Figure 7: Balanced accuracy vs. ratio ρ .

4.1.4 Metrics

In one-class classification, imbalanced data is common. Hence, we used balanced accuracy as metric, because it allows a fair comparison between our method and the baseline models also for imbalanced datasets. The balanced accuracy is defined as

$$\text{BACC} = \frac{1}{2} \left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}} \right). \quad (5)$$

with the terms true positive (TP), true negative (TN), false negative (FN) and false positive (FP). In this work, the normal class (the known class during training) was considered as the negative class. In contrast, the abnormal classes (unknown classes during training) were considered as positive classes.

4.2 Results and Discussion

The resulting balanced accuracies for all experiments are listed in Table 1. The proposed method performed best in 25 of 30 experiments in comparison with all baseline models. Taken deep baseline models into consideration, our method outperformed the ImageNet, CAE and Original in all 30 experiments. Although CLS, a regular autoencoder with the proposed closeness loss as regularization, showed a good performance, our method still achieved a better performance in 27 of 30 experiments.

Figure 7 shows the balanced accuracies averaged over classes in relation to different ratios ρ . In general, the proposed method was not sensitive to the choice of the ratio ρ for splitting the training data. However, the balanced accuracies tended to be lower if the ratio was higher, because a larger ratio indicates that more samples are considered to be atypical normal. This will lead to a low true negative rate.

Figure 8 illustrates the balanced accuracies averaged over classes in relation to different values of β_1 and β_2 . In the range of 10^{-7} to 10^{-3} , the performance of the

Table 1: Balanced Accuracy in %.

Normal Class	HOG	PCA	ImageNet	CAE	Original	CLS	Ours
Digit 0	68.9	83.2	71.1	87.5	94.6	96.6	96.8
Digit 1	65.0	94.3	88.9	92.7	95.9	97.7	97.9
Digit 2	50.6	64.1	58.5	78.7	71.5	89.9	88.4
Digit 3	56.5	73.7	63.2	87.5	78.9	90.9	90.9
Digit 4	81.8	80.5	70.6	86.4	89.1	90.3	90.9
Digit 5	48.4	59.4	60.7	81.9	68.1	85.2	83.3
Digit 6	78.7	68.7	67.6	83.5	91.8	92.4	95.3
Digit 7	53.8	75.1	71.0	82.0	85.9	90.0	90.4
Digit 8	63.3	77.8	64.0	88.6	80.7	87.9	89.7
Digit 9	75.6	74.7	71.8	83.5	85.8	91.7	92.0
<hr/>							
T-shirt	71.1	80.7	58.1	80.6	79.1	82.7	86.3
Trouser	84.9	87.1	75.4	85.0	87.7	84.8	92.2
Pullover	77.3	81.4	58.1	80.5	80.1	83.4	84.0
Dress	76.1	84.0	60.1	82.1	80.8	85.7	86.0
Coat	79.3	80.6	58.3	78.8	79.6	85.9	87.9
Sandal	80.3	75.6	69.2	79.1	79.4	87.1	88.4
Shirt	73.3	76.3	57.3	75.8	74.3	76.6	77.3
Sneaker	90.9	91.6	75.5	92.5	94.4	94.3	94.6
Bag	84.9	71.0	61.9	73.0	73.5	82.5	86.1
Ankle boot	93.1	88.2	78.3	88.5	93.7	95.7	96.4
<hr/>							
Airplane	51.5	59.5	53.3	59.7	60.7	62.2	63.6
Automobile	55.9	51.4	53.6	50.8	50.5	48.9	55.1
Bird	53.1	61.0	51.9	61.0	61.2	59.7	62.0
Cat	53.9	48.5	50.8	48.7	49.2	51.7	55.2
Deer	51.0	68.0	55.8	68.3	67.1	64.9	69.2
Dog	54.6	50.2	52.6	50.2	50.7	51.7	56.1
Frog	58.4	68.9	54.6	66.5	62.2	66.6	65.0
Horse	52.8	52.1	51.3	52.6	51.5	54.1	54.4
Ship	54.1	57.3	57.1	60.5	58.4	61.5	62.7
Truck	53.0	54.1	56.0	52.9	53.6	57.2	55.4

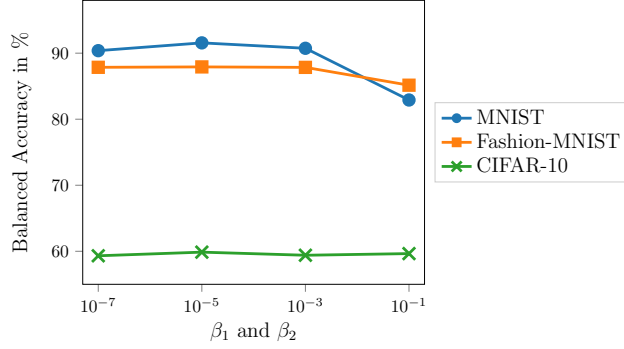


Figure 8: Balanced accuracy vs. β_1 and β_2 .

proposed method was stable for the conducted experiments. In contrast, the balanced accuracies on MNIST and Fashion-MNIST decreased with $\beta_1 = \beta_2 = 10^{-1}$, because the dispersion loss term is dominant. This is not desirable for OCSVM.

Further evaluation results are attached in Section B of the appendix.

5 Conclusion

In this work, we presented a novel generic feature learning method for one-class classification. To learn a proper latent space, normal training samples were split into typical and atypical normal data. These two subsets were used to train a network with different losses in an joint way under the constraints of the proposed closeness and dispersion requirements. As a result, the trained feature extractor enabled to extract highly discriminative features between normal and abnormal data. To evaluate the effectiveness of the proposed method, a neural network based on AlexNet was built.

Various and intensive experiments with the datasets MNIST, Fashion-MNIST and CIFAR-10 were conducted. The extracted features were used as input for an OCSVM to perform one-class classification. Our method showed a large improvement over ImageNet features, CAE and the original features in all considered cases. Moreover, in 25 of 30 experiments, the method outperformed all other baseline models.

An ongoing work is to realize the proposed method in an end-to-end way since it has already shown its efficient feature extraction ability for one-class classification. Furthermore, Figure 2 illustrates that SSIM is not the optimal similarity metric for splitting the given dataset. Thus, testing other similarity metrics could help to enhance the feature learning performance.

References

- [1] Shehroz S. Khan and Michael G. Madden. A survey of recent trends in one class classification. In Lorcan Coyle and Jill Freyne, editors, *Artificial Intelligence and Cognitive Science*, pages 188–197, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [2] Marco A. F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. Review: A review of novelty detection. *Signal Process.*, 99:215–249, June 2014.
- [3] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.
- [4] David M.J. Tax and Robert P.W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, Jan 2004.
- [5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
- [6] Pramuditha Perera and Vishal M. Patel. Learning Deep Features for One-Class Classification. jan 2018.
- [7] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, July 2013.
- [8] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.
- [9] Markos Markou and Sameer Singh. Novelty detection: a reviewpart 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.
- [10] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.
- [11] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class svm for learning in image retrieval. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 34–37. IEEE, 2001.
- [12] J Zhou, KL Chan, VFH Chong, and Shankar M Krishnan. Extraction of brain tumor from MR

- images using one-class support vector machine. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 6411–6414. IEEE, 2006.
- [13] Sankar Mahadevan and Sirish L Shah. Fault detection and diagnosis in process data using one-class support vector machines. *Journal of process control*, 19(10):1627–1639, 2009.
- [14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [16] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- [17] Yushi Chen, Hanlu Jiang, Chunyang Li, Xiuping Jia, and Pedram Ghamisi. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):6232–6251, 2016.
- [18] Lars Hertel, Erhardt Barth, Thomas Käster, and Thomas Martinetz. Deep convolutional neural networks as generic feature extractors. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–4. IEEE, 2015.
- [19] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, pages 625–660, 2010.
- [20] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR ’14, pages 580–587, Washington, DC, USA, 2014. IEEE Computer Society.
- [21] Lukas Ruff, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning*, pages 4390–4399, 2018.
- [22] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Robust, Deep and Inductive Anomaly Detection. apr 2017.
- [23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [25] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
- [26] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.
- [27] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.
- [28] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, volume 1, pages 886–893 vol. 1, June 2005.
- [29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng

Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

The detailed balanced accuracies for each class in relation to ρ are listed in the Table 6. The last column denotes the results averaged over different values of ρ and the corresponding standard deviation.

A Detailed Network Architecture

Based on the desired characteristics and corresponding loss functions, the proposed method utilizes an autoencoder architecture for the training and testing. Both the encoder and decoder are used during the training procedure. Once the network is trained, only the encoder is utilized as a feature extractor. Note that the samples from MNIST and Fashion-MNIST are first padded to $32 \times 32 \times 1$ before fed into the neural network.

A.1 Encoder

The architecture of the encoder is shown in Figure 5. The encoder includes five convolutional layers using 3×3 kernels but with different channels and one convolutional layer using 1×1 kernels acting as a dense layer. The max-pooling layers use a 2×2 pooling window. A more detailed setting for the encoder is summarized in Table 2 with the following abbreviations: convolution (Conv), batch normalization (BN) and average-pooling (AP). \mathcal{B} denotes the size of a mini-batch during training.

A.2 Decoder

The architecture of the decoder is shown in Figure 6. The decoder includes five transposed convolutional layers using 3×3 kernels with strides of two and one convolutional layer at the end. The up-sampling layers use a scale of two. A more detailed setting for the decoder is summarized in Table 3 with the following abbreviations: transposed convolution (ConvT), batch normalization (BN) and up-sampling (UPS). \mathcal{B} denotes the size of a mini-batch during training.

B Further Evaluation Results

AUC was used as an alternative metric to evaluate the proposed method and showed similar results as the balanced accuracies. The corresponding results are listed in the Table 4.

The detailed balanced accuracies for each class in relation to the values of β_1 and β_2 are listed in the Table 5. The last column denotes the results averaged over different values of β_1 and β_2 and the corresponding standard deviation.

Table 2: Architecture of the Encoder.

Layer Type	Output Shape	Connected to	Note
Input Layer	$(\mathcal{B}, 32, 32, 1)$	-	Number of channels is 3 for CIFAR-10
Convolution_1	$(\mathcal{B}, 32, 32, 32)$	Input Layer	“same”-padding
ReLU_1	$(\mathcal{B}, 32, 32, 32)$	Conv_1	-
Max-Pooling_1	$(\mathcal{B}, 16, 16, 32)$	ReLU_1	-
Batch Normalization_1	$(\mathcal{B}, 16, 16, 32)$	Max-Pooling_1	-
Convolution_2	$(\mathcal{B}, 12, 12, 64)$	BN_1	Dilation of 2
ReLU_2	$(\mathcal{B}, 12, 12, 64)$	Conv_2	-
Batch Normalization_2	$(\mathcal{B}, 12, 12, 64)$	ReLU_2	-
Convolution_3	$(\mathcal{B}, 8, 8, 64)$	BN_2	Dilation of 2
ReLU_3	$(\mathcal{B}, 8, 8, 64)$	Conv_3	-
Batch Normalization_3	$(\mathcal{B}, 8, 8, 64)$	ReLU_3	-
Convolution_4	$(\mathcal{B}, 4, 4, 128)$	BN_3	Dilation of 2
ReLU_4	$(\mathcal{B}, 4, 4, 128)$	Conv_4	-
Batch Normalization_4	$(\mathcal{B}, 4, 4, 128)$	ReLU_4	-
Convolution_5	$(\mathcal{B}, 2, 2, 256)$	BN_4	“valid”-padding
ReLU_5	$(\mathcal{B}, 2, 2, 256)$	Conv_5	-
Max-Pooling_5	$(\mathcal{B}, 1, 1, 256)$	ReLU_5	-
Average-Pooling_5-1	$(\mathcal{B}, 1, 1, 64)$	ReLU_2	Pooling window = (12, 12)
Average-Pooling_5-2	$(\mathcal{B}, 1, 1, 64)$	ReLU_3	Pooling window = (8, 8)
Average-Pooling_5-3	$(\mathcal{B}, 1, 1, 128)$	ReLU_4	Pooling window = (4, 4)
Concatenation_5	$(\mathcal{B}, 1, 1, 512)$	AP_5-1 to AP_5-3, MP_5	-
Convolution_6	$(\mathcal{B}, 1, 1, \# \text{ latent feature})$	Concatenation_5	1×1 Kernel
Sigmoid_6	$(\mathcal{B}, 1, 1, \# \text{ latent feature})$	Conv_6	-
Latent Output	$(\mathcal{B}, \# \text{ latent feature})$	Sigmoid_6	reshape to column vector

Table 3: Architecture of the Decoder.

Layer Type	Output Shape	Connected to	Note
Latent Input	$(\mathcal{B}, 1, 1, \# \text{ latent feature})$	-	reshape to tensor
Batch Normalization_6	$(\mathcal{B}, 1, 1, \# \text{ latent feature})$	Latent Input	-
ConvT_7	$(\mathcal{B}, 2, 2, 256)$	BN_6	Strides of 2
ReLU_7	$(\mathcal{B}, 2, 2, 256)$	ConvT_7	-
Batch Normalization_7	$(\mathcal{B}, 2, 2, 256)$	ReLU_7	-
ConvT_8	$(\mathcal{B}, 4, 4, 128)$	BN_7	Strides of 2
ReLU_8	$(\mathcal{B}, 4, 4, 128)$	ConvT_8	-
Batch Normalization_8	$(\mathcal{B}, 4, 4, 128)$	ReLU_8	-
ConvT_9	$(\mathcal{B}, 8, 8, 64)$	BN_8	Strides of 2
ReLU_9	$(\mathcal{B}, 8, 8, 64)$	ConvT_9	-
Batch Normalization_9	$(\mathcal{B}, 8, 8, 64)$	ReLU_9	-
ConvT_10	$(\mathcal{B}, 16, 16, 64)$	BN_9	Strides of 2
ReLU_10	$(\mathcal{B}, 16, 16, 64)$	ConvT_10	-
Up-Sampling_10-1	$(\mathcal{B}, 16, 16, 256)$	ReLU_7	size = (8, 8)
Up-Sampling_10-2	$(\mathcal{B}, 16, 16, 128)$	ReLU_8	size = (4, 4)
Up-Sampling_10-3	$(\mathcal{B}, 16, 16, 64)$	ReLU_9	size = (2, 2)
Concatenation_10	$(\mathcal{B}, 16, 16, 512)$	UPS_10-1 to UPS_10-3, ReLU_10	-
ConvT_11	$(\mathcal{B}, 32, 32, 32)$	Concatenation_10	Strides of 2
ReLU_11	$(\mathcal{B}, 32, 32, 32)$	ConvT_11	-
Batch Normalization_11	$(\mathcal{B}, 32, 32, 32)$	ReLU_11	-
Convolution_12	$(\mathcal{B}, 32, 32, 1)$	BN_11	Number of channels is 3 for CIFAR-10
Sigmoid_12	$(\mathcal{B}, 32, 32, 1)$	Conv_12	Number of channels is 3 for CIFAR-10
Reconstruction Outputs	$(\mathcal{B}, 32, 32, 1)$	Sigmoid_12	Number of channels is 3 for CIFAR-10

Table 4: AUC.

Normal Class	HOG	PCA	ImageNet	CAE	Original	CLS	Ours
Digit 0	74.3	89.7	77.7	94.6	98.5	99.4	99.5
Digit 1	70.3	98.2	94.7	97.9	99.5	99.7	99.7
Digit 2	49.2	68.9	63.5	86.1	82.3	95.6	95.6
Digit 3	56.8	78.6	67.6	94.5	88.3	96.4	96.5
Digit 4	89.4	85.0	74.8	92.7	95.0	95.1	95.7
Digit 5	50.8	53.8	67.1	88.8	76.5	93.9	90.8
Digit 6	85.1	70.1	68.1	90.3	96.3	98.7	98.9
Digit 7	57.6	80.6	78.0	88.9	93.4	97.0	92.8
Digit 8	66.9	85.5	69.1	94.6	89.0	93.1	95.5
Digit 9	82.6	78.4	77.4	90.4	93.0	97.5	97.8
T-shirt	77.1	88.9	49.8	89.5	86.9	92.3	93.0
Trouser	89.3	93.2	78.5	94.1	93.7	98.2	93.6
Pullover	84.3	86.7	46.9	88.4	86.0	89.6	89.9
Dress	84.1	90.4	57.7	90.5	87.6	94.4	94.8
Coat	85.8	87.4	51.9	88.2	84.8	91.5	92.0
Sandal	84.0	76.6	73.5	87.8	82.2	90.2	91.7
Shirt	75.2	81.7	47.6	83.3	80.7	84.5	85.5
Sneaker	95.1	96.2	83.8	97.3	97.8	98.7	98.8
Bag	88.4	77.1	62.8	82.2	77.8	89.8	92.6
Ankle boot	96.8	94.7	81.0	95.2	98.1	98.9	98.2
Airplane	51.3	63.5	54.5	62.9	61.9	65.3	67.2
Automobile	58.9	41.3	54.3	38.5	38.7	49.5	59.1
Bird	54.4	64.5	48.4	64.5	64.0	62.6	64.6
Cat	55.4	48.4	49.5	48.2	48.2	51.4	57.3
Deer	52.0	73.0	57.6	73.5	71.6	69.9	73.7
Dog	56.5	49.1	54.0	49.6	50.9	50.8	56.8
Frog	61.7	74.4	55.1	71.9	67.2	71.3	69.5
Horse	50.3	48.9	54.4	48.7	50.1	56.0	56.4
Ship	48.4	62.4	59.2	64.7	62.9	65.0	67.8
Truck	54.0	46.6	59.8	44.3	47.2	58.8	55.6

Table 5: Balanced Accuracy in %.

Dataset	$\beta_1, \beta_2 = 10^{-7}$	$\beta_1, \beta_2 = 10^{-5}$	$\beta_1, \beta_2 = 10^{-3}$	$\beta_1, \beta_2 = 10^{-1}$	Mean \pm Std.
Digit 0	96.5	96.8	96.4	90.4	95.3 ± 2.8
Digit 1	97.9	97.9	98.0	96.4	97.6 ± 0.7
Digit 2	85.4	88.4	86.0	75.8	85.1 ± 5.5
Digit 3	89.4	90.9	90.1	82.9	88.8 ± 3.4
Digit 4	89.3	90.9	90.5	82.9	88.8 ± 3.3
Digit 5	81.7	83.3	82.7	69.1	80.4 ± 6.4
Digit 6	95.0	95.3	95.2	89.5	93.5 ± 2.5
Digit 7	90.4	90.4	90.0	83.4	88.8 ± 3.0
Digit 8	87.2	89.7	87.4	78.0	86.0 ± 4.6
Digit 9	91.0	92.0	91.1	80.7	89.3 ± 4.8
<hr/>					
T-shirt	86.7	86.3	86.8	81.9	84.9 ± 2.4
Trouser	92.7	92.2	91.9	91.1	92.2 ± 0.8
Pullover	84.0	84.0	83.9	80.3	83.1 ± 1.6
Dress	86.3	86.0	85.4	83.3	85.3 ± 1.2
Coat	87.5	87.9	87.2	83.7	86.4 ± 1.7
Sandal	88.1	88.4	88.2	86.3	87.6 ± 0.9
Shirt	77.0	77.3	76.7	76.6	76.8 ± 0.3
Sneaker	94.0	94.6	94.4	94.4	94.3 ± 0.2
Bag	86.3	86.1	87.8	78.4	84.2 ± 3.8
Ankle boot	96.0	96.4	96.2	95.5	96.0 ± 0.4
<hr/>					
Airplane	62.0	63.6	64.0	65.8	63.5 ± 1.5
Automobile	52.7	55.1	53.1	59.6	53.9 ± 3.9
Bird	59.9	62.0	61.0	57.8	60.1 ± 1.6
Cat	55.5	55.2	54.4	55.5	54.5 ± 1.6
Deer	67.7	69.2	67.8	68.2	67.6 ± 1.6
Dog	56.2	56.1	55.6	53.0	54.5 ± 2.0
Frog	65.5	65.0	65.7	63.0	65.2 ± 1.3
Horse	54.4	54.4	54.4	54.1	54.3 ± 0.2
Ship	62.1	62.7	60.8	62.3	61.9 ± 0.7
Truck	57.2	55.4	57.2	57.2	56.8 ± 0.8

Table 6: Balanced Accuracy in %.

Dataset	$\rho = 5$	$\rho = 10$	$\rho = 15$	$\rho = 20$	$\rho = 25$	$\rho = 30$	Mean \pm Std.
Digit 0	96.2	96.8	95.6	95.2	95.3	95.2	95.7 \pm 0.7
Digit 1	98.0	97.9	97.4	97.4	97.4	96.5	97.4 \pm 0.5
Digit 2	86.8	88.4	86.8	85.8	85.7	86.5	86.7 \pm 1.0
Digit 3	90.8	90.9	89.9	88.6	88.8	88.8	89.6 \pm 1.1
Digit 4	89.6	90.9	88.8	88.8	90.3	89.0	89.6 \pm 0.9
Digit 5	81.2	83.3	81.0	79.7	80.8	79.8	81.0 \pm 1.3
Digit 6	94.4	95.3	93.0	93.6	93.6	92.5	93.7 \pm 1.0
Digit 7	90.6	90.4	89.2	88.6	89.4	88.5	89.5 \pm 0.9
Digit 8	87.6	89.7	87.1	85.1	86.5	85.5	86.9 \pm 1.7
Digit 9	90.4	92.0	89.2	89.3	88.6	88.3	89.6 \pm 1.4
T-shirt	86.0	86.3	80.1	84.5	84.9	85.3	84.5 \pm 2.3
Trouser	93.7	92.2	92.6	91.5	89.7	91.4	91.9 \pm 1.4
Pullover	83.8	84.0	83.1	83.6	83.0	83.4	83.5 \pm 0.4
Dress	84.8	86.0	84.1	85.2	84.6	84.2	84.8 \pm 0.7
Coat	87.1	87.9	87.2	87.0	87.4	87.4	87.3 \pm 0.3
Sandal	88.2	88.4	88.1	87.5	87.8	87.4	87.9 \pm 0.4
Shirt	77.0	77.3	75.4	76.2	76.3	77.2	76.6 \pm 0.7
Sneaker	94.6	94.6	93.5	94.5	94.8	94.7	94.5 \pm 0.5
Bag	84.2	86.1	85.0	85.6	83.7	85.3	85.0 \pm 0.9
Ankle boot	95.8	96.4	96.1	96.4	96.1	97.0	96.3 \pm 0.4
Airplane	65.1	63.6	63.4	63.9	62.5	61.2	63.3 \pm 1.3
Automobile	50.9	55.1	52.6	50.5	51.0	54.3	52.4 \pm 1.9
Bird	60.4	62.0	59.5	59.0	59.4	59.8	60.0 \pm 1.1
Cat	54.7	55.2	54.4	55.9	54.1	53.4	54.6 \pm 0.9
Deer	67.9	69.2	68.8	67.9	66.1	68.2	68.0 \pm 1.1
Dog	54.9	56.1	54.1	56.2	55.8	55.1	55.4 \pm 0.8
Frog	65.7	65.0	65.8	63.9	65.4	67.0	65.5 \pm 1.0
Horse	54.7	54.4	55.7	54.3	55.0	55.0	54.9 \pm 0.5
Ship	62.1	62.7	62.6	60.5	62.0	62.1	62.0 \pm 0.8
Truck	56.6	55.4	54.7	54.6	55.7	57.2	55.7 \pm 1.0