# BEUD : Bifold-Encoder Uni-Decoder based Network for Anomaly Detection

Mohith Rajesh[0000−0002−3621−4946], Chinmay Kulkarni[0000−0003−2935−9861], and Shylaja S S

PES University, Bengaluru, India
{mohithraj9301,camkoolkarni}@gmail.com, shylaja.sharath@pes.edu

**Abstract.** Anomaly Detection is a very critical and significant data analysis mission given the raft of cyber-attacks these days. Used to identify thought-provoking and emerging patterns, predispositions, and irregularities in the data, it is an important tool to perceive abnormalities in many different domains, including security, finance, power automation, health, computer network intrusion detection, etc. Deep learning-based AutoEncoders have shown great potential in identifying anomalies. However, state-of-the-art anomaly scores are still based on reconstruction errors, which do not take advantage of the available anomalous data samples during the training phase. Towards this direction, we demonstrate a novel extension to the AutoEncoder that not only maintains the AutoEncoder's ability to discover non-linear features of non-anomalies but also uses the existing anomalous samples to assist in learning the features of the data better. Since the model architecture is designed to have two encoders and one decoder network, we name our model as Bifold-Encoder Uni-Decoder (BEUD) network. In this paper, we discuss two different ways of using the BEUD model to predict the anomalies in the data. BEUD is conceptually analogous to AutoEncoder but empirically more powerful. The experimental results of this architecture demonstrated a fairly good performance compared to the standard AutoEncoder architecture evaluated for the Anomaly Detection task.

**Keywords:** Anomaly Detection · Neural Network · AutoEncoder · BEUD · Contrastive Learning · Imbalanced dataset· Representation learning.

## 1 Introduction

Bearing in mind the swift spike in E-business and digital payment systems, there is an uptick in frauds pertaining to financial and banking transactions allied to credit cards. Analysts and researchers have been vigorously trying to discover solutions to overpower them and one of the solutions is to use data mining approaches. However, the collected credit card data can be a reasonable challenge for researchers because of the data characteristics that contain an imbalanced class distribution.

We need to recognise that the magnitude of the significant classes in the data, which are mostly deceitful transactions, is usually marginal. This is the main challenge in the problem of anomaly detection. Generally, it is easy for learning algorithms to unearth their regularities if they have adequate records. But when their number is insignificant, as is typically the case for anomalies, unearthing their regularities and generalising their definite conclusive regions using learning algorithms becomes difficult. Many learning algorithms are envisioned to maximise accuracy, which leans more towards majority classes and against minorities. Generally, for classification, models like Deep Neural Networks demand that all the classes have an adequate number of data samples. But it is strenuous to find data samples for anomalies.

In this paper, we discuss the AutoEncoder methodology for the Anomaly Detection problem. An AutoEncoder is used to perceive fraudulent transactions (anomalies) based on reconstruction error; only data with normal instances is used to train the AutoEncoder, which is why this method stands apart from other deep learning methods. We present an analysis showing that exposing the existing anomalous samples to the network is crucial. We also present the **BEUD** architecture, a model architecture that instead of relying entirely on an encoder representation of non-anomalies to draw global dependencies between input and output flags (Anomaly or Non-Anomaly), allows for significantly better encoder representations by making use of existing anomaly examples.

## 2 Related Work and Background

Anomaly Detection is a significant endeavour that has been extensively studied within various research areas and application domains. A long line of literature has thus been proposed, including reconstruction-based [9–14], density-based [1–8], one-class classifier [15, 16], and self-supervised [17–19] approaches. Overall, a majority of recent literature is concerned with (a) modeling the representation to better encode normality [20, 18], and (b) defining a new detection score [16, 19]. Recently, significant progress has been made in the self-supervised domain, with the use of contrastive learning [23].

Contrastive learning extracts a strong inductive bias from multiple (similar) views of a sample by letting them attract each other, yet repelling them to other samples. It is a machine learning technique used to learn the general features of a dataset by teaching the model which data points are analogous or different.

We first found that the existing AutoEncoder method is already reasonably effective for detecting anomalous samples with a proper detection score. We further observe that one can improve its performance by utilising available anomalous samples while training. In particular, the existing AutoEncoder method predicts anomalies by learning the reconstructions of the non-anomalous samples and using the appropriate detection score; whereas the BEUD model additionally pushes the encoding of anomalies and normal data apart while pulling the encoding of normal data closer. The model now learns a new task of discriminating between anomalies and non-anomalies, in addition to the original task of learning the reconstructions of non-anomalous data.

## 3 Dataset

The AutoEncoder demands non-anomalous data for training and predicts whether the new data passed is anomalous or not. Whereas the proposed BEUD model requires both anomalies and non-anomalies for training. So we require data which represents both the anomalous and non-anomalous proceedings.

This (https://www.kaggle.com/mlg-ulb/creditcardfraud) dataset which was prepared by Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) and provided by Kaggle perfectly matched our requirements.

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset contains transactions of two days, where there are 492 frauds out of 284,807 transactions. It is highly imbalanced, the positive class (frauds) account for 0.172% of all transactions.

The attributes are amount of money involved in transactions and time, other 28 columns are labelled as “V1”, “V2”, “V3”, till “V28”. The results of the transactions (whether Anomaly or Non-Anomaly) are given in the “Class” column, where 1 represents Anomalous transactions and 0 represents normal transactions. This dataset contains only numeric input variables which are the result of a PCA (Principal Component Analysis) transformation.

Additionally, we perform a Min/Max scaling before training, which is a normalization method that consists of re-scaling the range of features and the data is divided into 80% Train set and 20% Test set.

## 4 BEUD : Bifold-Encoder Uni-Decoder network

AutoEncoders are trained by unsupervised learning to learn reconstructions that are close to their original input; we leverage neural networks for the task of **representation learning**. Explicitly, we design a neural network architecture such that we enforce a bottleneck in the network which forces a **compressed knowledge representation** of the original input.

The encoder and decoder with a single hidden layer can be shown as equation 1 and equation 2, respectively. W and b are the weight and bias of the network respectively. T is the nonlinear transformation function, and the subscript xe represents that x is the input with which W is multiplied, and e is the output of the equation.

$$e = T(W_{xe}x + b_{xe}) \tag{1}$$

The encoder in equation 1 maps an input vector x to a hidden representation performing an affine mapping followed by a non-linear transformation.

$$d = T(W_{ex}e + b_{ex}) \tag{2}$$

The decoder in equation 2 maps the hidden representations back to the original input space as a Reconstruction.

$$e_t = |x - d| \tag{3}$$

The deviation of Reconstruction from the original input vector is called the Reconstruction error represented as $e_t$ in equation 3.

An AutoEncoder learns to minimize the mean of Reconstruction error given by equation 4.

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |e_t| \tag{4}$$

The data representing normal instances is used to train the AutoEncoder; When testing, the AutoEncoder will reconstruct normal data precisely, while failing to do so with anomalous data, as it has never run into the latter before. Reconstruction error is used as an anomaly score. If it is above the threshold, the data is flagged as anomalous.

The AutoEncoder method illustrated above can be effectively applied for anomaly detection in a dataset with few or no anomalous data samples accessible for training, as it requires only normal data for training. The anomalous samples existing in the training dataset are not taken advantage of while training the AutoEncoder.We present the BEUD model architecture, which is designed analogous to the AutoEncoder with an additional encoder to utilise anomalous data samples accessible for training.

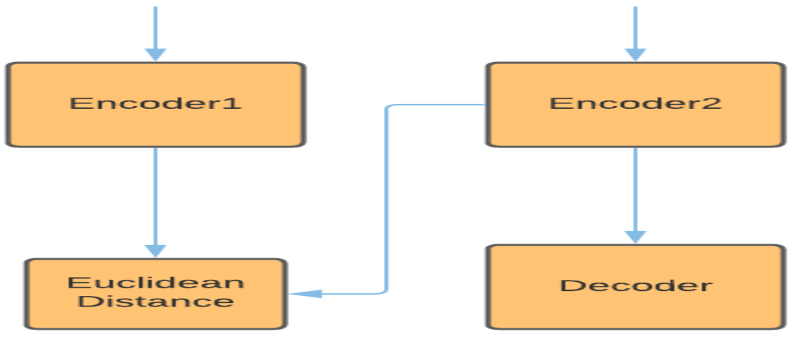

**Fig. 1.** BEUD Model

BEUD model architecture is depicted in figure 1. The encoder and decoder on the right represent the AutoEncoder. We will be referring to this encoder as Encoder2, whereas the encoder on the left is an additional component to the AutoEncoder in the BEUD architecture. We will be referring to this encoder as Encoder1. Encoders are named distinctly just to ease the explanation of different

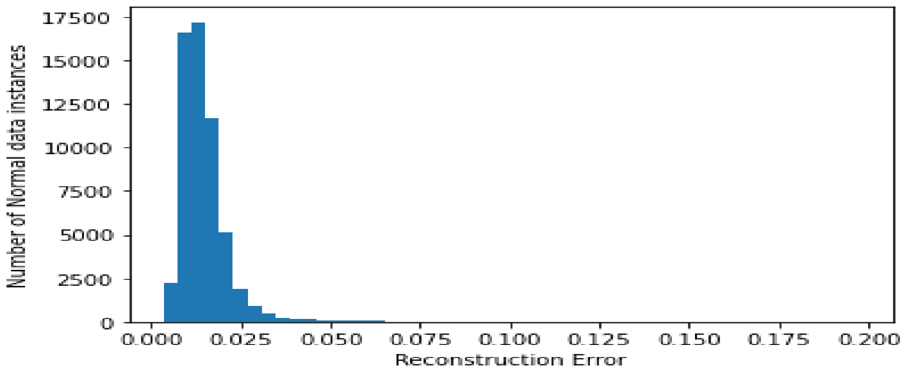

**Fig. 2.** Number of Non-Anomalous test examples v/s Reconstruction Error

phases of training the BEUD model; otherwise, they are homogeneous, i.e., they share weights.

The BEUD model goes through two stages of training. The first phase is similar to the AutoEncoder approach. The Encoder2 and Decoder have been trained to minimise the Mean Average Error (MAE) of the normal data reconstruction. Phase 1 has a single objective, so it may be completed with greater accuracy. The second phase of BEUD model training involves simultaneously reducing MAE and the Contrastive loss equation as indicated in equation 8 . In essence, weights learned in phase-1 are readjusted in phase-2 to maintain and reduce the MAE to some extent while also achieving the additional goal of effectively reducing the Contrastive loss 8 . The training phase-2 is made easier because the weights were taught to minimise MAE in phase-1.

The encoder2 receives and encodes only non-anomalous data during the training phase-1, thus the decoder learns to decode only non-anomalous data encoding. Even in the second phase of training, encoder2 only receives and encodes non-anomalous data, while encoder1 can receive both anomalous and non-anomalous input samples. If encoder1 gets a non-anomaly, encoder1's output encoding and encoder2's output encoding must be analogous. If encoder1 gets an anomaly, encoding of both encoder1 and encoder2 should be divergent. This is accomplished by lowering the Contrastive Loss, as shown in equation 8.

The Euclidean Distance Node calculates the Euclidean distance between the encoding of encoder1 and encoder2. Encoder1 is used to push the encoding of anomalies and non-anomalies distinctively and bring closer the encoding of two non-anomalies, consequently utilising the existing data of anomalies. So, a pair of data instances is passed to BEUD: if encoder1 (the first element of the pair) receives an anomaly, the data pair is labelled as 0, and if encoder1 receives a non-anomaly, the data pair is labelled as 1. The second element of the pair is always a non-anomaly, which is passed to encoder2. Thus, this pair of data along with labels is used in training phase-2. Equation 6 calculates the square of Euclidean distance between the encoding of encoder1 and encoder2, margin in equation 7 is a hyper-parameter usually set to 1, if encoder1 (the first element of the pair) receives an anomaly since the label for such a pair is 0, Contrastive Loss in equation 8 reduces to margin_square(equation 7). Thus, in case the encoder 1 receives an anomaly minimising the Contrastive Loss is nothing but increasing the distance between the encoding of encoder1 and encoder2. If encoder1 receives a non-anomaly, the data pair is labelled as 1. So, Contrastive Loss in equation 8 reduces to square_pred(equation 6). Thus, in this case minimising the Contrastive Loss is nothing but decreasing the distance between the encoding of encoder1 and encoder2.

In this approach, we are trying to learn a better encoding that represents the input feature, such that encoding of non-anomalous data are analogous and encoding of anomalous and non-anomalous data are divergent, aiding the model to perform better. **The algorithm for BEUD** is discussed elaborately below.

**Algorithm for BEUD**:

```
e1, e2 and d represent Encoder1, Encoder2 and Decoder respectively
Start
    $\phi$ →parameters of e1 and e2
    $\theta$ →parameters of d
    $\alpha$ →threshold reconstruction error
    $x\_train, x\_test$ → input features of train data and test data.
    $y\_train$ → labels(0 for non-anomaly 1 for anomaly) of train data
    $N$ →size of test data
    $\phi, \theta$ ←train the network using the second element in the pair of input features minimising MAE for $\phi$(Encoder2) and $\theta$ for n1 number of epochs.
    $\phi, \theta$ ←train the network using the pair of input features minimising Contrastive loss for $\phi$(Encoder1 and Encoder2) and MAE for $\phi$(Encoder2) and $\theta$ for n2 number of epochs.
if method1: //Reconstruction error as anomaly score
    Testing:
        for i = 1 to N:
            //error in reconstructing the original input by encoder2
            and decoder sub-networks of BEUD
            Reconstruction error(i)=
                $\|x\_test(i) - d_\theta(e2_\phi(x\_test(i)))\|$    (5)
            if Reconstruction error(i) $> \alpha$ :
                x_test(i) is an Anomaly
            else:
                x_test(i) is not an Anomaly
if method2: Training classifier for the output encoding of the Encoder of BEUD
    $newTrainX = e2_\phi(x\_train)$
    new Train Y = y_train
    classifier.fit(new Train X, new Train Y)
    Testing:
```

```
    for i = 1 to N:
        if classifier.predict(x_test(i)) > 0.5:
            x_test(i) is an Anomaly
        else:
            x_test(i) is not an Anomaly
Stop
```

**Contrastive Loss**:

$\hat{y}$ is the predicted distance between the encoding of the encoders and y is the label for the data pair.

$$square\_pred = (\hat{y})^2 \tag{6}$$

$$margin\_square = (max(margin - \hat{y}, 0))^2 \tag{7}$$

$$Loss = mean(y * square\_pred + (1 - y) * margin\_square) \tag{8}$$

In this paper we discuss 2 different ways of using BEUD model to predict the anomalies in the data.

### 4.1 Reconstruction error as the Anomaly score

The encoder2 and decoder of the BEUD model were initially trained to minimise the Mean Average Error (MAE) of the reconstructions of normal data, and later, the BEUD model was additionally trained to minimise the Contrastive loss along with MAE, reconstructions of normal data would be precise in comparison to the reconstructions of anomalous data. Thus, how well the data is constructed by the encoder and decoder could be used as the score to classify the data as an anomaly.

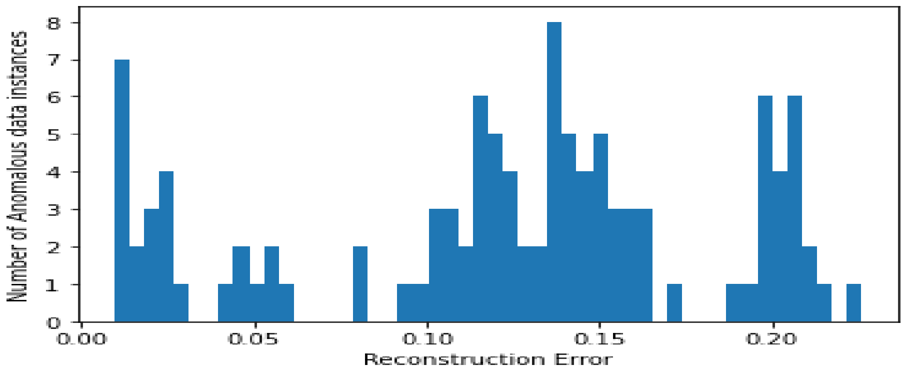

**Fig. 3.** Number of Anomalous test examples v/s Reconstruction Error

The BEUD model has three layers in both the encoder and decoder networks and was trained on 76,600 data pairs, with the first element being either an anomaly or non-anomaly, but the second element being always non-anomaly.

The data pairs are generated from the training dataset, and each pair is labelled appropriately, as stated previously. The labels 0 and 1 have an equal class distribution, which implies that there are 38,300 data pairs with the labels 0 and 1. However, since the availability of anomalies throughout the training data is minimal, 383 anomalies are duplicated to generate 38,300 data pairs for label 1, but the data pairs are distinct since the second element of each pair (normal data) is not duplicated. In phase 1, BEUD is exclusively trained on the second element in the pair to minimize MAE. It is done for 100 epochs, and then 76,600 data pairs with labels are utilised to train(phase-2) the BEUD for 500 epochs to minimise both MAE and Contrastive loss.

The threshold for data reconstruction error is determined; if the error in data reconstruction exceeds the threshold, the data sample is flagged as anomalous. figure 2 shows that reconstruction errors for most normal examples are less than 0.05, while figure 3 shows that reconstruction errors for many anomalous occurrences are greater than 0.1. A recall value of 75.23% and a precision of 75.93% were observed when the threshold reconstruction error was set to 0.1.

### 4.2 Training classifier for the output encoding of the Encoder of BEUD

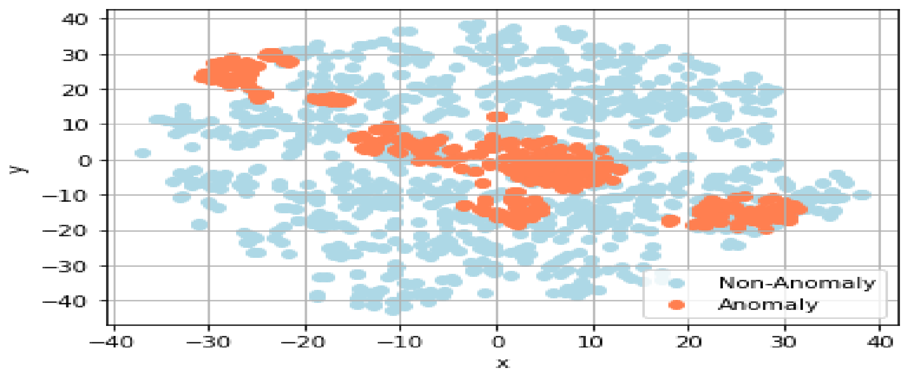

**Fig. 4.** Anomaly vs. Non-Anomaly (reduced dimension of original data)

Using T-SNE(t-Distributed Stochastic Neighbor Embedding), we can investigate the nature of our various transaction classes in greater depth. The visualisation generated after transforming the original data (lowering the dimension to 2) is displayed in figure 4. There is no satisfiable decision boundary that separates an anomaly from a non-anomaly. In this section, we'll look at how the BEUD model may be used to develop improved representations of the input so that anomalous and non-anomalous data can be distinguished.

The BEUD model is built using four layers in the encoder and three layers in the decoder network, and it is trained on 2298 data pairs (the first element

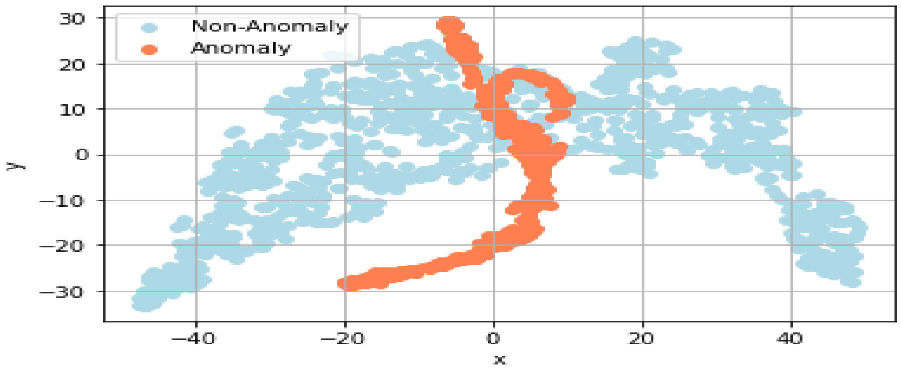

**Fig. 5.** Anomaly vs. Non-Anomaly (reduced dimension of the encoding)

being an anomaly or non-anomaly, but the second being non-anomaly always) to get improved representations of the data . As mentioned in the preceding section, data pairs are constructed from the train dataset, with each pair being suitably labelled and each label having an equal class distribution (4.1). To minimise MAE, the BEUD is first trained just on the second element in the pair. This initial phase (phase-1) lasts 10 epochs, after which 2298 data pairs with labels are utilised to train the BEUD for 30 epochs (phase-2) in order to reduce MAE and Contrastive loss. We can tell the difference between fraudulent and non-fraudulent cases by their inner nature. BEUD's encoder receives the original data, which is then translated into a lower dimension via the T-SNE transformation. We can see in figure 5 that both classes are now visible, and the anomalies exist in a specific region, i.e., they are not distributed across the normal data. It's worth noting that the T-SNE transformation is only utilised to obtain data visualisation insights; it's not employed in either the BEUD training or the prediction stage.

After the BEUD has been fully trained, we want to find the latent space representation of the input that the network has learned (encoder). The hidden representations of both non-anomalous and anomalous classes are then obtained. With the encoding of 50,000 non-anomalies and the encoding of 383 anomalies returned by the encoder, a new training dataset with labels is produced where label 1 represents the encoding of anomalous data and label 0 represents encoding of non-anoamlous data. This information is utilised to train Classifiers like SVM.

## 5 Experimental Results

In this section, we cover the outcomes of the BEUD and AutoEncoder anomaly detection methods, as well as the importance of leveraging existing anomalous data with additional encoder. We examine the performance of AutoEncoder and BEUD for anomaly detection on the credit card dataset.

### 5.1 Using Reconstruction error as anomaly score

We discussed in section 4.1 how BEUD could predict whether the data was an anomaly or not using the reconstruction error as the anomaly score. The Adam optimiser is used to train the BEUD model using 76600 data pairs (lr = 0.001). More information on the training phases have been discussed in section 4.1. In this section, we compare the BEUD model with AutoEncoder.In both the encoder and the decoder, the AutoEncoder has the same number of layers as BEUD. With the Adam optimiser (lr = 0.001), the autoencoder is trained on 76600 non-anomalous instances over 600 epochs. In the case of the AutoEncoder, the training goal is to reduce MAE for non-anomalous training instances. Because BEUD was trained on 600 epochs altogether, encompassing both phases, the AutoEncoder was also trained on 600 epochs to ensure a fair comparison. On the CreditCard test dataset, figure 6 shows the plot of precision vs recall for BEUD and AutoEncoder for various threshold reconstruction errors ranging from 0.01 to 0.2 with an increment of 0.0001. The BEUD model performed substantially better than the AutoEncoder, as shown in the figure 6. The precision of the BEUD model is much higher than the AutoEncoder for the same recall.

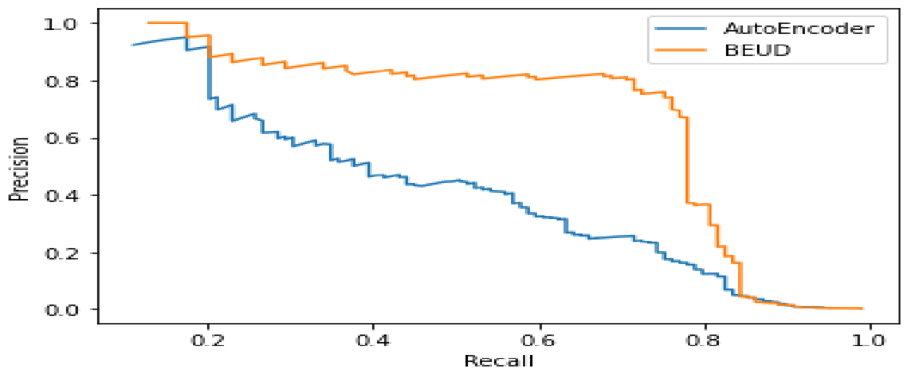

**Fig. 6.** Precision v/s Recall for BEUD and AutoEncoder on the CreditCard test dataset

### 5.2 Training classifier for the output encoding of the Encoder of BEUD

The BEUD model is trained on 2298 data pairs using the Adam optimiser (lr = 0.001) in this technique. More information on the training phases can be found in section 4.2. In this section, we compare the BEUD model against the AutoEncoder (same number of layers and units/layer in encoder and decoder as in BEUD), which was trained with Adam optimiser on 2298 non-anomalous examples over 40 epochs (lr = 0.001). The new training data is made up of the

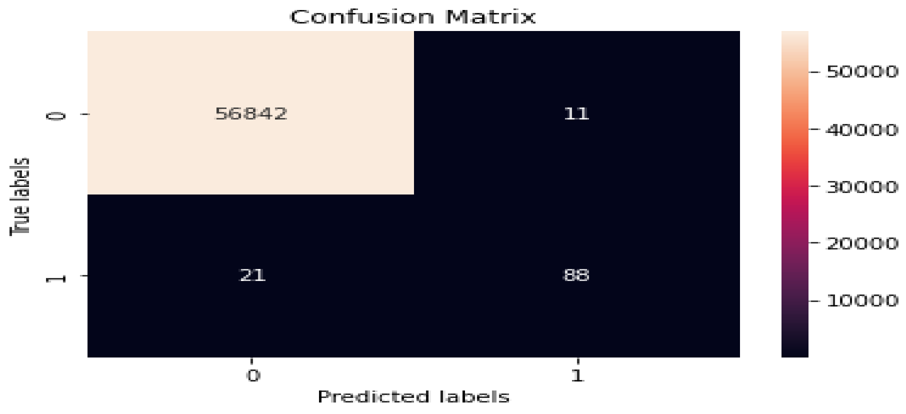

**Fig. 7.** Confusion matrix for the predictions of BEUD on test data

encoding returned by BEUD and AutoEncoder, as well as labels. This data is made up of the encoding of 50,000 non-anomalies and 383 anomalies, and it is used to train the SVM classifier. The results (Table 1) of SVM on test data using the encoding supplied by BEUD are substantially better than SVM using the encoding returned by the AutoEncoder. The confusion matrix for BEUD predictions on test data is shown in figure 7.

## 6 Discussions and Conclusion

In terms of anomaly detection on a credit card dataset, the BEUD model outperformed the AutoEncoder model, as shown in the preceding section; the BEUD model's results are fairly better compared to the AutoEncoder-based method discussed in "Applications of Machine Learning in Fin-tech Credit Card Fraud Detection" [22].

**Table 1.** Performance comparision of AutoEncoder and BEUD.

| | **Recall** | **Precision** |
|---|---|---|
| **AutoEncoder** | 70.64% | 85.55% |
| **BEUD** | 80.73% | 88.88% |

By altering the threshold for reconstruction error of BEUD, method 1 (section 4.1) can be used to control false-positives and false-negatives. As a result, precision and recall can be managed. BEUD is adaptable to a variety of applications. When it comes to bank transactions, recall is crucial. Declaring non-spam material as spam in spam mail detection could have disastrous consequences, as critical mail would be missed. Precision is of the utmost significance in this application.

Despite the fact that the BEUD model performed better than the AutoEncoder, the performance of the BEUD model necessitates being compared with other anomaly detection methods. The performance of BEUD needs to be evaluated on other anomaly scores and on image datasets. This model poses fresh prospects for research since the results of BEUD are quite good and enhancing the performance marginally will assist many anomaly detection applications.

## References

1. S. Zhai, Y. Cheng, W. Lu, and Z. Zhang. Deep structured energy based models for anomaly detection. In International Conference on Machine Learning, 2016.
2. E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? In International Conference on Learning Representations, 2019
3. H. Choi, E. Jang, and A. A. Alemi. Waic, but why? generative ensembles for robust anomaly detection. arXiv preprint arXiv:1810.01392, 2018.
4. E. Nalisnick, A. Matsukawa, Y. W. Teh, and B. Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using a test for typicality. arXiv preprint arXiv:1906.02994, 2019.
5. Y. Du and I. Mordatch. Implicit generation and modeling with energy based models. In Advances in Neural Information Processing Systems, 2019.
6. J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In Advances in Neural Information Processing Systems, 2019.
7. J. Serrà, D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In International Conference on Learning Representations, 2020.
8. W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. Your classifier is secretly an energy based model and you should treat it like one. In International Conference on Learning Representations, 2020.
9. T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In International conference on information processing in medical imaging, 2017.
10. B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In International Conference on Learning Representations, 2018.
11. L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. Image anomaly detection with generative adversarial networks. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2018.
12. S. Pidhorskyi, R. Almohsen, and G. Doretto. Generative probabilistic novelty detection with adversarial autoencoders. In Advances in Neural Information Processing Systems, 2018.
13. P. Perera, R. Nallapati, and B. Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In IEEE Conference on Computer Vision and Pattern Recognition, 2019.

14. S. Choi and S.-Y. Chung. Novelty detection via blurring. In International Conference on Learning Representations, 2020.
15. B. Schölkopf, R. C. Williamson, A. J. Smola, et al. Support vector method for novelty detection. In Advances in Neural Information Processing Systems, 2000.
16. L. Ruff, R. Vandermeulen, N. Goernitz, et al. Deep one-class classification. In International Conference on Machine Learning, 2018.
17. I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. In Advances in Neural Information Processing Systems, 2018.
18. D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song. Using self-supervised learning can improve model robustness and uncertainty. In Advances in Neural Information Processing Systems, 2019.
19. L. Bergman and Y. Hoshen. Classification-based anomaly detection for general data. In International Conference on Learning Representations, 2020.
20. D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. In International Conference on Machine Learning, 2019.
21. L. Ruff, R. Vandermeulen, N. Goernitz, et al. Deep one-class classification. In International Conference on Machine Learning, 2018.
22. Lacruz, Francisco, and Jafar Saniie. "Applications of Machine Learning in Fintech Credit Card Fraud Detection." 2021 IEEE International Conference on Electro Information Technology (EIT). IEEE, 2021.
23. Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9729–9738, 2020.