CSC730
Assignment 1
January 13th, 2024

Names: *Mangesh Sakordekar & Jackson Cates*

1 Introduction

For assignment 1, we were assigned to use an imbalanced dataset taken from MNIST. This dataset contains samples from the original MNIST dataset, but with some classes having far fewer samples than others. This can be seen in Fig. 1, where some classes like 3's and 4's are very common, while other classes like 7's and 1's are not very common.

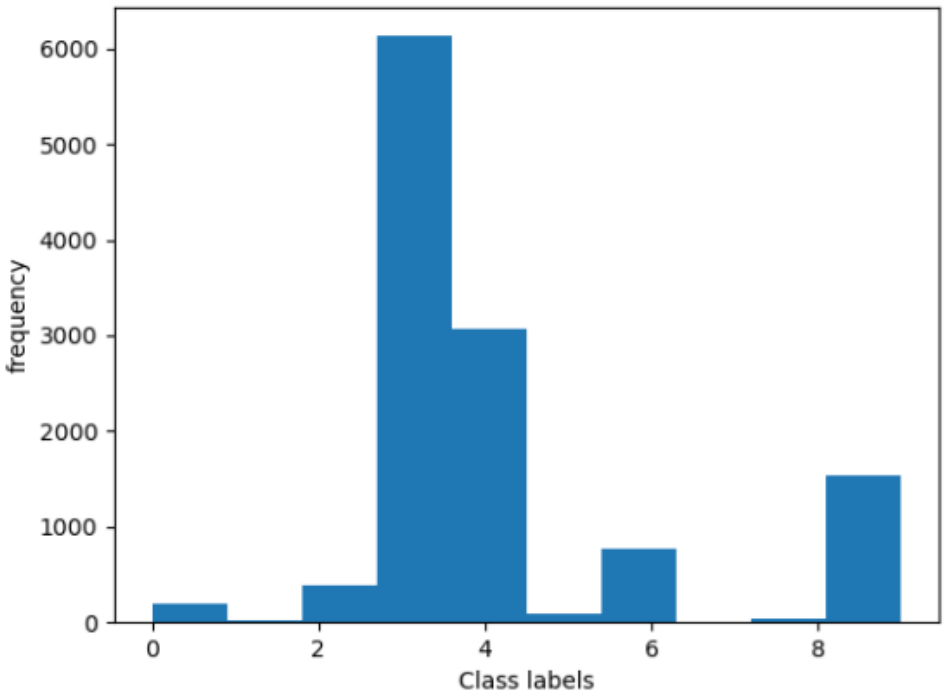

Figure 1: Histogram of MNIST class digits. Notice some classes are far less common than others, such as the 7's and 1's.

Our assignment is to come up with our own method of ranking each sample by "anomalousness." We will then rank each sample from most to least anomalous, and compare that to the anomalousness. We attempt to determine the anomalousness by utilizing dimensionality reduction. For this, we utilize PCA and a Convolutional Autoencoder. The goal in mind is that these methods will learn the more common observations well such that common observations will have a low reconstruction error, while uncommon or "anomalies" will have a high reconstruction error. We can use the reconstruction error directly to score our samples. In this report, we will compare our two approaches to solving this problem. We will also discuss how well we rank anomalousness and some learning we had throughout the project.

2 Principal Component Analysis

Principal Component Analysis (PCA) is a popular and widely used method of dimensionality reduction. PCA involves computing the covariance matrix of the data and then performing eigen decomposition on this covariance matrix to determine the principal components. For this project we utilize the Singular Value Decomposition. Then the principal components are used to project the original data down the PCA space, performing the dimensionality reduction. We used PCA for this problem by taking all of our images $X \in \mathbb{R}^{784\times12244}$ (with 12244 observations as our columns with 784 features for a 28×28 image), and scaling them between 0-1. We then compute the singular value decomposition of our images as,

$$USV^T = \text{svd}(X)$$

where U contains our left-singular vectors, S contains our singular values on the diagonal, and V contains our right-singular vectors. To perform dimensionality reduction, we need to pick the number of left-singular vectors to keep. We can do this by creating a scree plot, which shows the explained variance ratio of each singular value. We compute the explained variance ratio for the i^{th} principal component as

$$\frac{S_{ii}}{\sum_{n=1}^{784} S_{nn}}.$$

The result of this is shown in Fig. 2, and the number of principal components are picked at the elbow of this plot.

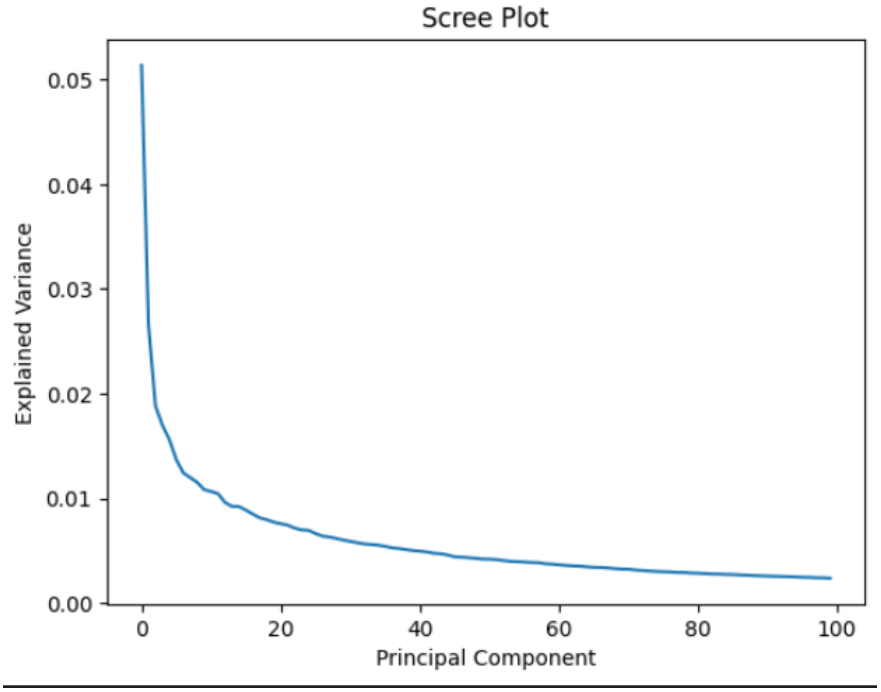

Figure 2: Scree plot of our PCA analysis. Typically the principal components are picked on the "elbow" of the plot.

From this plot, we decided to pick $k = 4$ since that is roughly the elbow of the plot. We then can keep the first k left-singular vectors in U_k and use those for dimensionality reduction. We then can project the images into PCA space by

$$\tilde{X} = U_k^T X$$

where $\tilde{X} \in \mathbb{R}^{k \times 12244}$ will be our observations that is projected to PCA space. and then reconstruct our observations by

$$\hat{X} = U_k X.$$

$\hat{X}$ will now contain our reconstructed images. Hopefully, anomalies will have a poor reconstruction while more common observations will have a good reconstruction. A figure of our images reconstructed is shown in Fig. 3.

Figure 3: Demostration of reconstructed images from PCA. Top row: original images. Bottom row: reconstructed images.

After we compute $\hat{X}$, we then can simply score anomalies based on our reconstruction error via

$$\text{score for } i^{\text{th}} \text{ observation } = ||\hat{X}_i - X_i||_2.$$

3 Convolutional Autoencoder

We tried to use a convolutional autoencoder provided in the Neural Nets textbook. The autoencoder compresses the 28 by 28 images into 2 values and tries to regenerate the the image from these values. The assumption behind this approach was to train the autoencoder on skewed data so, the decoder will reconstruct the more commonly occurring images easily and struggle with reconstructing the anomalies. Thus, looking at the reconstruction score, we could detect the anomalies.

The encoder has two hidden convolutional layers whose output is then flattened and fed into a dense layer. The output layer of the encoder is a dense layer with two neurons. The decoder mimics the encoder in the reverse order taking in two values at the input layer and outputting a

28 by 28 image. The auto encoder was trained using the skewed MNIST dataset provided for 10 epochs. For the training the total loss was around 158.7 and the reconstruction loss was around 155.3. Training the model for a limited number of epochs ensures that the model is not familiarized with the outliers but can reconstruct the common classes accurately.

Once the model was trained, the skewed MNIST dataset was fed into the encoder. The encoded images were then fed into the decoder for reconstruction. A sample of the reconstructed images can be seen in Fig. 4

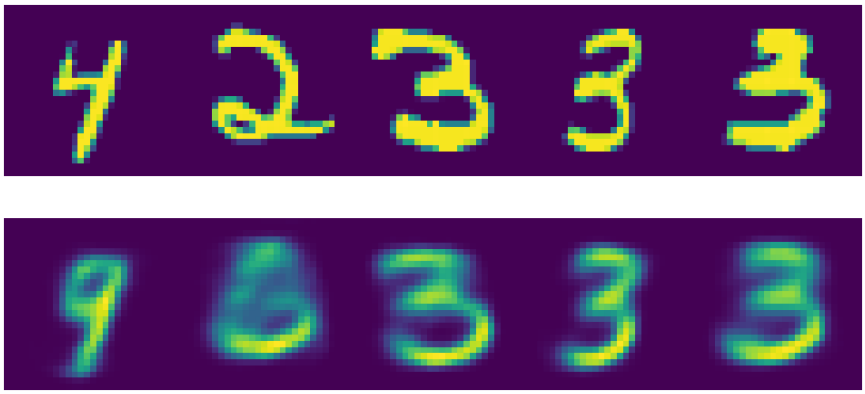

Figure 4: Demostration of reconstructed images from Autoencoder. Top row: original images. Bottom row: reconstructed images.

As we can observe, the commonly occurring class of 3's is reconstructed quite correctly whereas outliers 2 are reconstructed poorly. To get the reconstruction score, we subtracted the pixel values of the original image from the reconstructed image.So, lower the reconstruction score, the closer the image is to the original.

$$\text{reconstruction\_score} = \sum_{i=0}^{784} |original\_pixel_i - reconstructed\_pixel_i|$$

4 Results

Assuming the observations are scored randomly, our baseline is an accuracy of 33.3%. The accuracies we achieved with our two methods are 43.7% with PCA and 35.3% for the Convolutional Autoencoder. These accuracy's do seem low, and we believe it is mostly due to how reconstruction errors can look for "normal" observations. If we further investigate, the confusion matrix of our scores are shown in Table 1 and Table 2. Notice how the digits 3 and 4 are our most common observations, and about 2400 observations (1262 & 1183) are "mislabeled" for these two digits, which is about 20% of our data. The same may hold true to other pairs of digits, such as 3's and 9's being fairly common, but combined has about 1200 observations that are "mislabeled". We believe that this shows that our methods may fail to rank whats constitutes as "more normal" with this method.

Table 1: Confusion matrix for PCA

19	7	29	38	43	10	29	1	3	12
8	0	2	2	1	2	4	0	0	4
26	1	52	114	88	11	61	1	4	25
17	2	10	3945	1183	8	166	0	6	794
29	1	74	1262	1035	22	188	1	12	441
11	5	25	23	12	2	11	1	1	4
33	4	83	249	196	18	109	4	9	61
4	0	0	0	1	1	3	0	0	2
12	0	10	7	3	2	11	0	0	2
32	3	98	491	503	19	184	3	12	187

Table 2: Confusion matrix for Autoencoder

28	0	21	75	33	0	25	0	0	9
5	0	3	5	5	0	3	0	0	2
27	0	48	148	79	0	55	0	2	24
9	22	45	3269	1464	42	183	6	25	1066
30	0	85	1586	815	31	263	3	14	238
19	0	11	37	10	3	10	0	0	5
33	0	67	296	221	1	75	0	1	72
3	0	2	2	2	0	1	0	0	1
13	0	3	12	5	0	9	0	1	4
24	1	98	701	431	18	142	2	4	111

To investigate this a bit further, we also viewed this as a binary classification approach. We labeled ”abnormal” observations if their class appeared less than 5% of all observations (0, 1, 2, 5, 7, 8). We would like to see if these methods would be useful as an outlier detector in general. We then can compare the reconstruction errors on a ROC plot as shown in Fig. 5. This plot we want to compare the true positive rate and the false positive rate for our predictions, and with this we get a fairly decent classifier. This may indicate that if we cant rank anomalies well, this may be a good method in order to just detect them in a binary fashion.

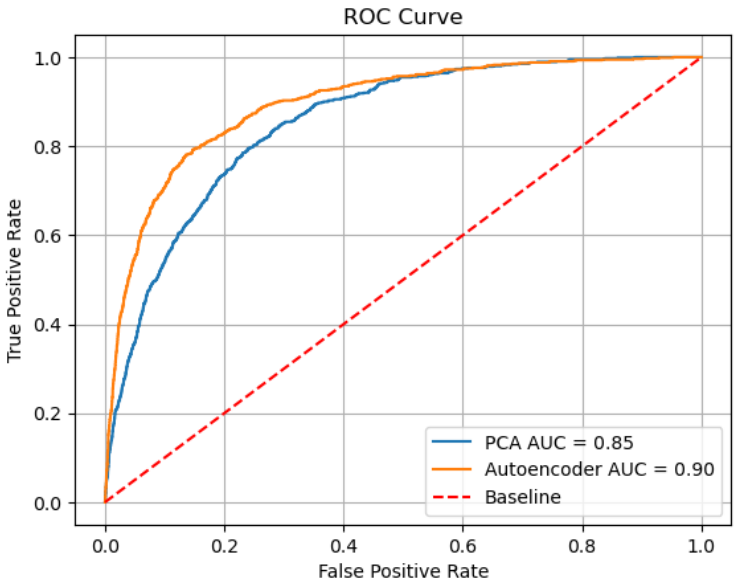

Figure 5: ROC Curve plot for our methods. The closer the curve gets to the upper left, the better. AUC of each method is shown on the legend.

5 Conclusion

Overall we compared our two approaches to rank anomalies within this dataset. We found that our reconstruction method did not perform well with ranking anomalies, likely due to how we rank ”normal” observations. However, we did learn that these methods may work for a binary classification approach, where the goal is just to detect if a point is an anomaly or not.