CSC 730 Assignment 06
Measuring Performance – ROC and PR Curves

Mangesh Sakordekar

Overview

The aim of this assignment is to:

- Get the MNIST and MNIST-C dataset.
- Select 3 different types of image corruption from MNIST-C.
- Create mixed datasets for training and test using the original MNIST and corrupted images at a 100/1 ratio for each corruption type.
- Create a probability density based anomaly detector
- Using a range of probability thresholds, create the corresponding ROC and precision-recall curves.

Reading The Data

Upon downloading the MNIST-C dataset from the TensorFlow website the data was read in using numpy load function. The MNIST dataset was read in using the Keras datasets library. Among the corrupted images, Impulse Noise, Stripe and Spatter corruptions were used for this assignment.

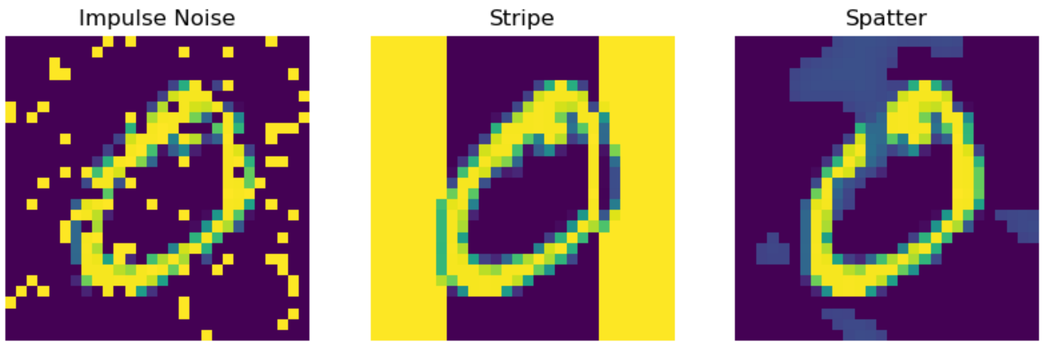
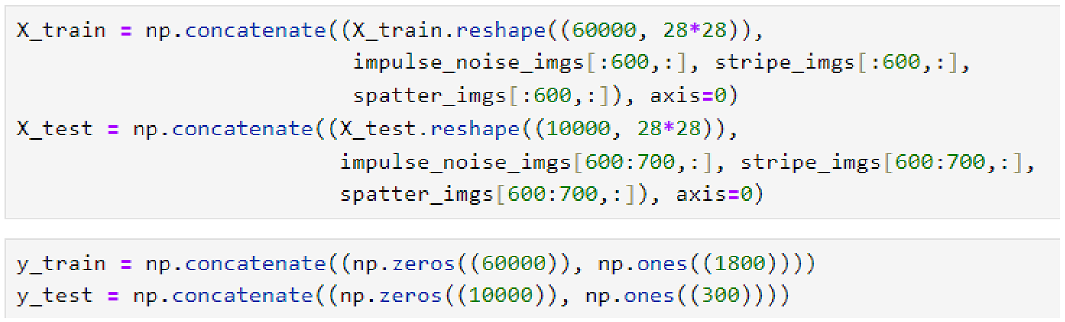


Figure 1: Corruptions examples

Creating the Datasets

To create the datasets, corrupted images were added to the MNIST dataset in 100 to 1 ratio. A total of 1800 corrupted images were added to the training dataset and 300 to the test dataset. Labels were created for these datasets with normal data labelled as 0 and corrupted images labelled as 1. The code for generating the datasets can be seen below.



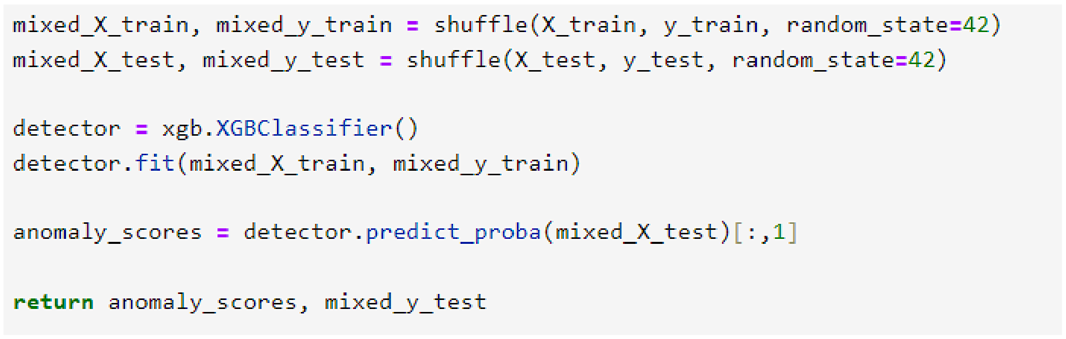
```
X_train = np.concatenate((X_train.reshape((60000, 28*28)),
                          impulse_noise_imgs[:600,:], stripe_imgs[:600,:],
                          spatter_imgs[:600,:]), axis=0)
X_test = np.concatenate((X_test.reshape((10000, 28*28)),
                         impulse_noise_imgs[600:700,:], stripe_imgs[600:700,:],
                         spatter_imgs[600:700,:]), axis=0)
```

```
y_train = np.concatenate((np.zeros((60000)), np.ones((1800))))
y_test = np.concatenate((np.zeros((10000)), np.ones((300))))
```

Figure 2: Creating Datasets

Anomaly Detector

For this task, I used XGBoost as my anomaly detector. The model was trained using the train dataset and the probabilities of the images in the test dataset being an corrupted image were calculated. These probabilities were used to measure the performance of the model and plot the graphs.

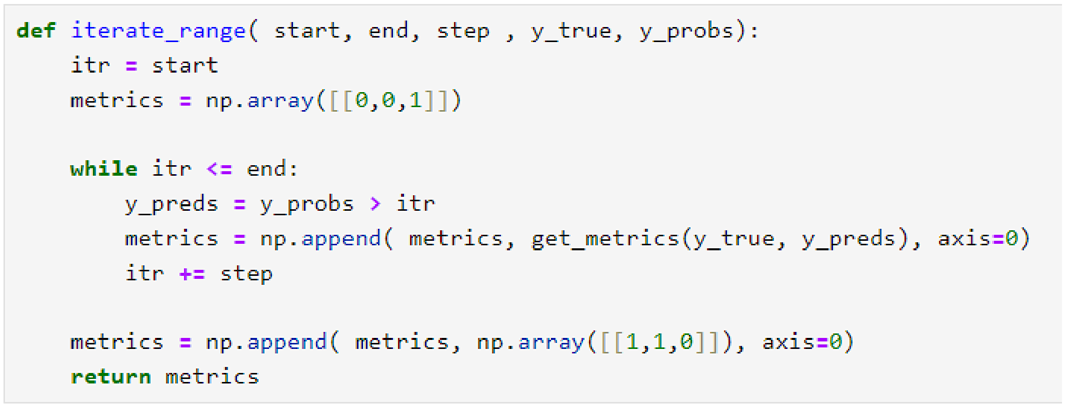


```
mixed_X_train, mixed_y_train = shuffle(X_train, y_train, random_state=42)
mixed_X_test, mixed_y_test = shuffle(X_test, y_test, random_state=42)

detector = xgb.XGBClassifier()
detector.fit(mixed_X_train, mixed_y_train)

anomaly_scores = detector.predict_proba(mixed_X_test)[:,1]

return anomaly_scores, mixed_y_test
```

Figure 3: XGBoost

Calculating Metrics

The range of thresholds tested for plotting of the ROC and Precision-Recall curves was from 0.01 to 1 with a step of 0.01 on each iteration.

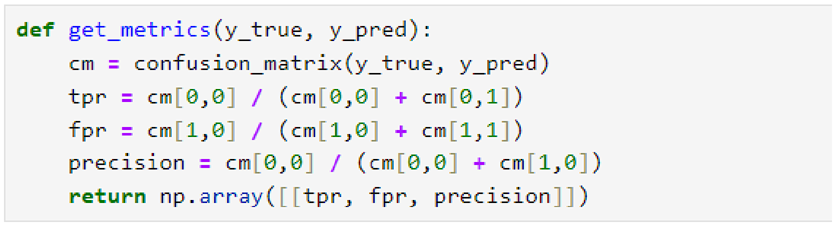


```
def iterate_range( start, end, step , y_true, y_probs):
    itr = start
    metrics = np.array([[0,0,1]])

    while itr <= end:
        y_preds = y_probs > itr
        metrics = np.append( metrics, get_metrics(y_true, y_preds), axis=0)
        itr += step

    metrics = np.append( metrics, np.array([[1,1,0]]), axis=0)
    return metrics
```

Figure 4: Varying the threshold

Predicted labels were generated for each threshold value and were compared to the true labels using a confusion matrix. The True Positive Rate (Recall), False Positive Rate and the

Precision were calculated using the confusion matrix. These values were stored in a numpy array and used for plotting.



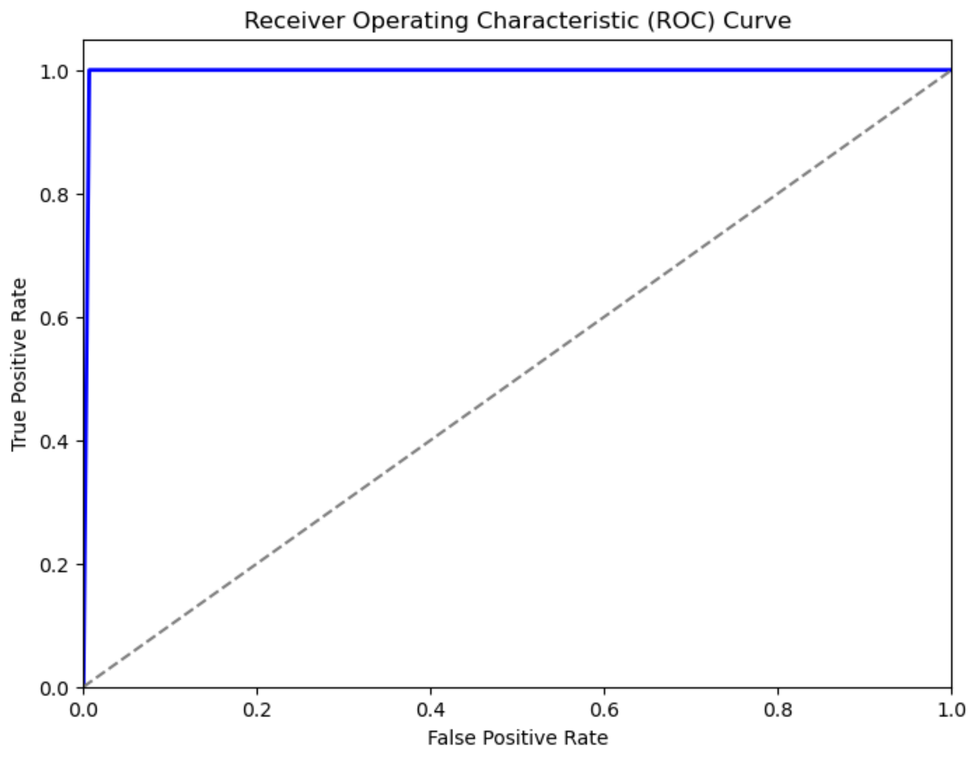
```python
def get_metrics(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    tpr = cm[0,0] / (cm[0,0] + cm[0,1])
    fpr = cm[1,0] / (cm[1,0] + cm[1,1])
    precision = cm[0,0] / (cm[0,0] + cm[1,0])
    return np.array([[tpr, fpr, precision]])
```

Figure 5: Calculating Metrics

Plots

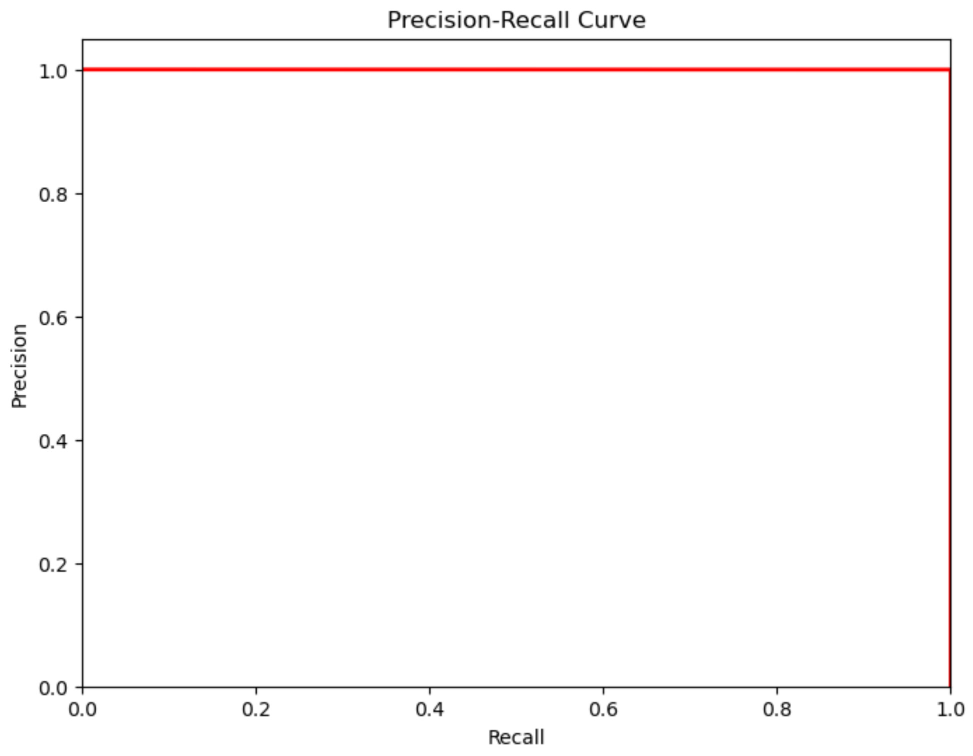
Receiver Operating Characteristic (ROC) Curve

ROC curve plots the True Positive Rate against the False Positive Rate calculated in the previous step.



Precision-Recall Curve

Precision Recall curve plots the Precision against the True Positive Rate also known as Recall.



Conclusion

Looking at the plots, we can say that this model works extremely well at detecting anomalies in the given MNIST dataset. Though we have to keep in mind that ROC curve is overly optimistic with imbalanced datasets as the one used in this assignment.