HW4: Anomaly detection

Narayana Ghosh (G01348658)

Name: nghosh2
Rank: 2
Accuracy = 1

Introduction:

Anomaly detection is a crucial task in various domains, including industrial control systems, where identifying abnormal behavior can help prevent catastrophic failures and security breaches. In this context, we propose the application of the StrOUD algorithm for anomaly detection on centrifuge data. The StrOUD algorithm leverages the reconstruction error of an Autoencoder as the strangeness function, enabling it to capture deviations from normal patterns effectively. The dataset utilized in this study comprises electromagnetic signals recorded by a sensor attached to a control unit for a centrifuge.

The centrifuge exhibits four distinct normal modes of operation, denoted as Mode A, Mode B, Mode C, and Mode D, each containing sampled signals represented by floating-point numbers. Additionally, the dataset includes data captured during a controlled infection of the centrifuge with the Stuxnet virus, represented by Mode M. To evaluate the algorithm's performance, we also incorporate test data that combines normal signals from Modes A, B, C, and D with malicious data from Mode M. By implementing the StrOUD algorithm and utilizing the reconstruction error from an Autoencoder as the strangeness function, we aim to enhance the detection capabilities for identifying anomalous patterns in the centrifuge data, particularly when faced with previously unseen malicious behavior.

Method:

Create baseline/Training and Test data from the given datasets:

To create the baseline for training data in the anomaly detection process, we exclusively select all normal data signals from folders ModeA, ModeB, ModeC, and ModeD, omitting any empty files. Each file represents a signal with multiple features, where each feature corresponds to a measurement or sample of the signal. Sufficient signals are available, allowing the generation of one or more random samples for potential validation purposes.

We have taken 10% of the data from each modes for validation purpose and rest of the data is used for Training the auto encoder.

Each signal undergoes Fast Fourier Transform (FFT) in Python, producing a vector with the same number of features as the original signal. This transformed data becomes our baseline sample or training data, which will be utilized in the subsequent stages of the anomaly detection algorithm.

Training the Auto encoder:

The dataset generated for training is utilized to train an auto encoder, wherein each point in the baseline is reconstructed using the other points in the baseline. By computing the reconstruction error for each point with respect to the other points, a list of strangeness measures is obtained. This list is then sorted in ascending order, resulting in the strangeness training list, which will be used in the subsequent steps of the anomaly detection process.

Testing the trained Auto encoder:

To create a test set for anomaly detection, a balanced dataset is produced by randomly selecting signals (files) from ModeA, ModeB, ModeC, and ModeD folders, representing normal behavior, and combining them with randomly selected signals from ModeM, which represent anomalies. Care is taken to ensure that the test set remains balanced by taking 10% of the data from each modes. Additionally, data from ModeA, ModeB, ModeC, and ModeD that were not used in the training dataset are included in the test set. FFT is applied to each signal in the test set, resulting in transformed data points.

In the evaluation phase of the anomaly detection process, the test strangeness list is analyzed as follows: For each measure in the test strangeness list, its rank is determined by finding how many measures in the training strangeness list are higher or equal to it. This rank is denoted as "b." Then, the p-value of that test point is computed by dividing (b+1) by (N+1), where N is the size of the training strangeness list. The resulting ratio is the p-value for that specific test point. This procedure is performed for all test points in order, generating a list of test p-values that corresponds to the test points' order. These p-values provide valuable insights into the significance of each test point's strangeness, aiding in the identification of potential anomalies in the dataset.

To assess the performance of the anomaly detection algorithm, the list of test p-values obtained in the previous step is used to construct the corresponding Receiver Operating Characteristic (ROC) curve. The ROC curve is built by considering confidence levels ranging from 0 to 1.0 at chosen intervals. For each confidence level, the p-value of each test point is compared with 1 minus the confidence level. If the p-value is less than 1 minus the confidence level, the test point is classified as an anomaly; otherwise, it is considered normal. By comparing these predictions to the true test labels, the False Positive rate and True Positive rate are computed at each confidence level, representing individual points on the ROC curve.

We have run the experiment multiple times but since we are taking 10% of data each time we noticed AUC as 1. We tried taking 0.2% of the test set and could notice the ACU 0.99.

P-value for Test dataset for Miner2:

Since the AUC of the above test data created from the training dataset for validation is 1 the auto encoder is perfect using 10% of the test split and gives highest accuracy. An AUC (Area Under the Curve) value of 1 indicates that the auto encoder-based anomaly detection algorithm has achieved perfect performance in distinguishing between normal and anomalous instances in the test set. A value of 1 for AUC implies that the True Positive Rate (TPR) is always 1, and the False Positive Rate (FPR) is always 0 across all confidence levels, which means there are no misclassifications, and all anomalies are correctly identified while maintaining zero false alarms. This is an excellent result and signifies that the auto encoder has successfully learned to capture the unique patterns of normal data and detect anomalies with high accuracy. With a True Positive count of 10 and only 1 False Positive, it suggests that the algorithm has effectively identified 10 anomalies while making only a single false alarm, further confirming its robustness and reliability in anomaly detection. We have used two different AUE architecture but it yielded the same AUC hence choose to consider the **standard AUE with 0.1% of validation data**.

Hence we have used the same auto encoder to generate the p-value of the test dataset provided and submitted in Miner2.

Result:

Result of the Auto encoder with multiple experiment with different AUE architecture and different validation split

Experiment#	AUE Architecture	Validation split for each Mode	AUC	TP	FP
1	Standard	0.1	1	10	1
2	Sparse	0.1	1	10	1
3	Sparse	0.2	0.99	19	4
4	Standard	0.2	0.99	19	4

(Result with split 0.1)

```
AUC: 1.0
True Positives: 10
False Positives: 1
```

(Result with Split 0.2)

```
AUC: 0.9965625
True Positives: 19
False Positives: 4
```

Conclusion:

In conclusion, the implementation of the StrOUD algorithm using an auto encoder for anomaly detection on centrifuge data has yielded highly promising results. The use of the auto encoder to generate a baseline training dataset allowed for effective learning of normal data patterns and the computation of reconstruction errors, serving as the strangeness measures for subsequent analysis. The algorithm demonstrated exceptional performance with an AUC of 1, indicating perfect discrimination between normal and anomalous instances in the test set. Moreover, achieving a True Positive count of 10 while having only 1 False Positive highlights the algorithm's accuracy in identifying anomalies while maintaining a low false alarm rate. These outcomes showcase the StrOUD algorithm's effectiveness in capturing and differentiating subtle deviations indicative of anomalous behavior within the centrifuge data. The success of this approach suggests its potential applicability in real-world industrial control systems, providing valuable insights for early anomaly detection and preemptive maintenance to prevent potential critical failures or security breaches.