Unsupervised Time Series Anomaly Detection

Peter Chen
10/1/2019

United Technologies Digital

Project Overview

Proposal: R&D of unsupervised anomaly detection algorithms for time series data

- Benchmark different unsupervised methods on open-source datasets
- Develop methodology to use Autoencoders to perform anomaly detection
- Only results on open source datasets will be shared with Georgia Tech. Benchmarks on internal datasets will not be shared for compliance purposes

Benefits

- Successful ML model can save $ by preventing anomalies at UTC
- Common code base can be applied to various anomaly detection problems at UTC

Datasets

Primary Dataset - Paper Mill Failure Dataset

Secondary Datasets – Credit Card Default, IBM IoT Dataset, Machine Failure Datasets

Current Status

Completed

1. Self-learning for autoencoders and unsupervised anomaly detection via online resources
2. Self-learning for tensorflow/keras toolkit for deep learning
3. Dataset selection and literature review – focus on time series anomaly detection datasets
4. Metric selection – Area Under Precision Recall Curve
5. Develop preliminary methodology for Autoencoders using reconstruction error
6. Benchmark autoencoder model and compare performance of different performance intervals and topologies
7. Benchmark autoencoder model performance to XGBoost (in paper) and K means clustering technique

Next Steps

1. Assess K means clustering on encoded layer outputs
2. Visualize encoded layers using T-sne visualizations – attempt to visually separate failures and non-failures
3. Try different aggregation techniques for reconstruction error such as standard deviation and median
4. Assess unsupervised + supervised techniques such as autoencoder + logistic regression
5. Self learnings on generative models – Generative Adversarial Networks and Variational Autoencoders
6. Benchmark model on additional datasets – compare performance
7. Develop reusable code-base for autoencoder anomaly detection

Anomaly detection

Traditional Anomaly Detection for Machine Data

- Tiered alarm systems using expert set thresholds and statistical analysis of data

Challenges

- Lots of manpower needed to set thresholds
- Lack of failure data - class imbalance
- High Dimensional – difficult to visualize
- Non-stationary – changing contexts (ex. Different ambient conditions)
    - One size fits all thresholding technique is not ideal for non-stationary data

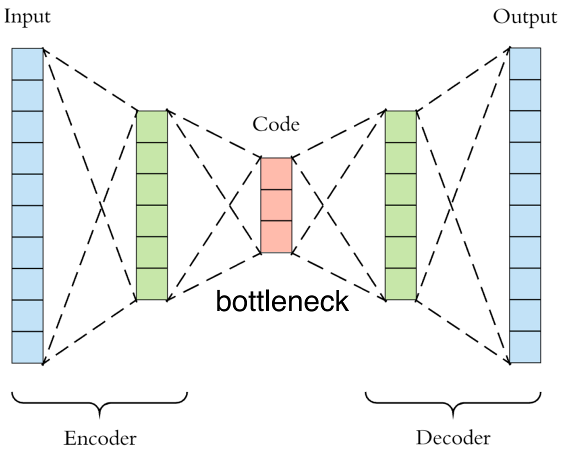
Potential Improvement

- Unsupervised learning – Clustering/Autoencoders
    1. Learn to model nominal behavior
    2. Assess deviation for new data points to determine likelihood of anomaly

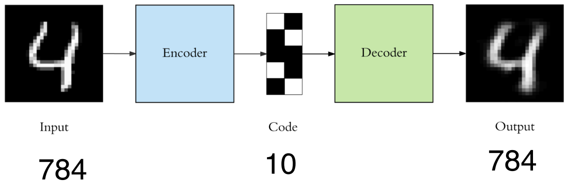
Autoencoders Overview

- Symmetric neural network where inputs = outputs
- Encoder – Decoder system with a middle bottleneck layer for dimension reduction
- Goal – Learn the intrinsic structure of the data by prioritizing the important information (code)
- Key Assumption: Structure is different for nominal data vs anomalous data
  - Therefore, reconstruction error for anomalous data will be higher
  - Use the difference in error to detect anomalies

Autoencoder structure



Example Use Case



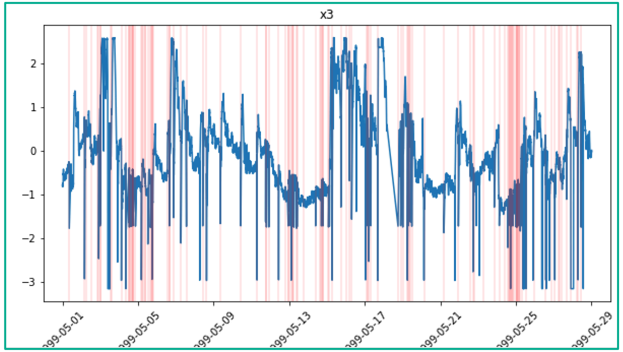
United Technologies Digital

Paper Mill Dataset

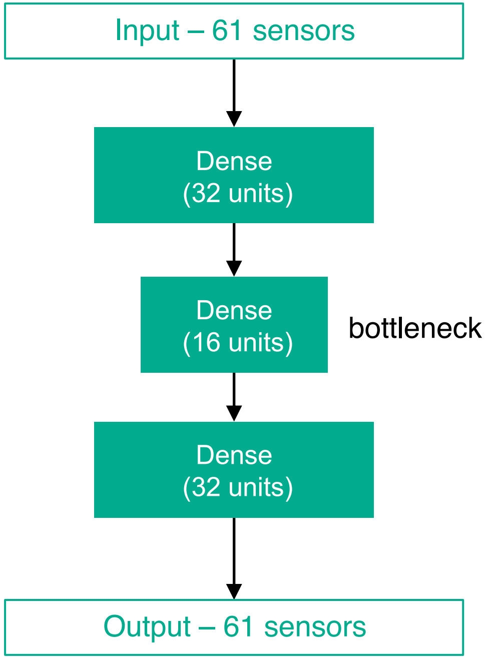
Open Source Dataset #1: Paper Mill Machine Sensors

- https://arxiv.org/pdf/1809.10717.pdf
- **Data**: Time series of paper mill machine with 60 sensor parameters and 1 binary target variable indicating failures
    - Failures represent when the machine breaks, breaks can take over 1 hour to resolve
- **Variables**: Large class imbalance – 124 failure vs 18,274 non-failure points (failures in red)
- **Benchmark:** In the paper, the best supervised model (XGBoost) had F1 score of .11

Sensor 3 out of 61



Simple Autoencoder for Paper Mill



Loss: mean squared error

Proposed Anomaly Detection Method

1. Train model on non-failure data

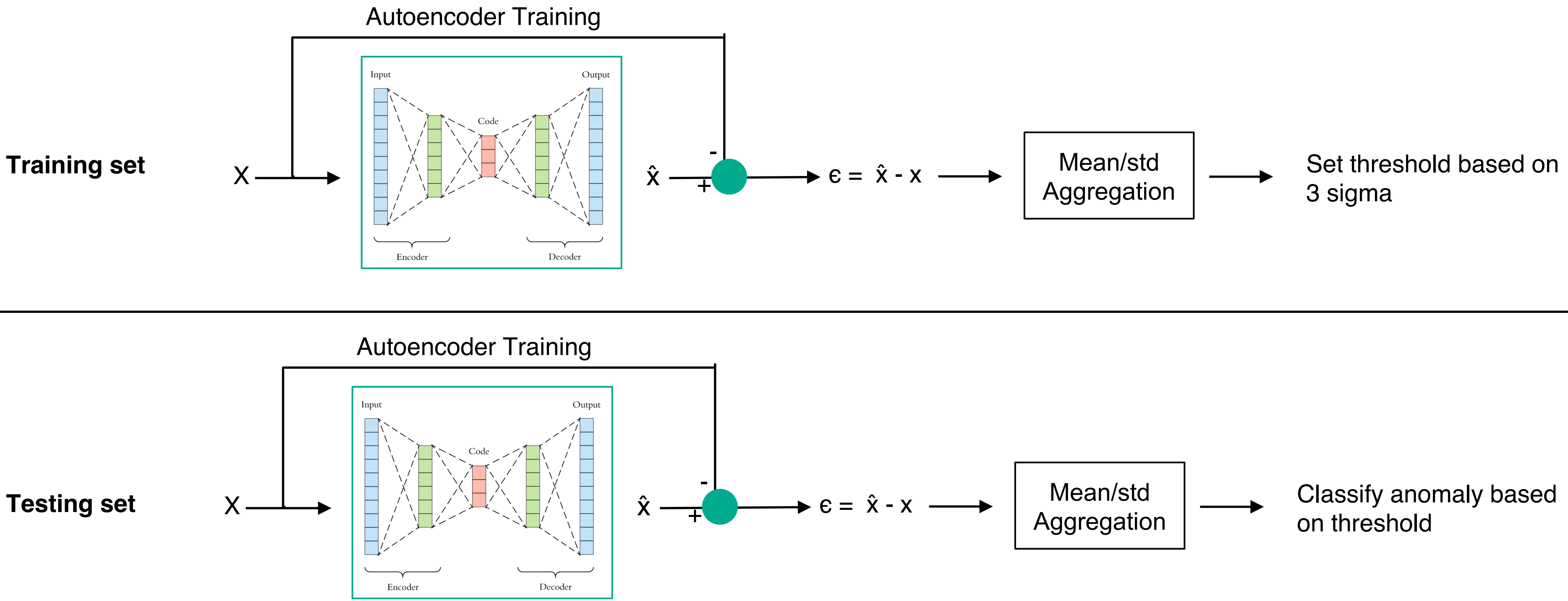
   X = sensor inputs, y = sensor inputs

2. Set statistical threshold for nominal behavior based on training data's reconstruction error

   try 3 sigma, 5 sigma, and max threshold

3. Predict on test data
4. Calculate reconstruction error for test data
5. Classify anomaly based on error threshold
6. Benchmark models based on area under Precision-Recall Curve. (ROC AUC is not suited for class imbalance)

Autoeoncoder Flow chart



United Technologies Digital

Data Preprocessing – Autoencoder

* Only use non-failures for training

CSV file

Drop non-sensor variables → Binary target variable padding → Standard scaler → Train/test split

Binary target variable padding:

- Allow for early prediction intervals
- Early interval = 1 means predicting failure 1 timestep beforehand

Train/test split:

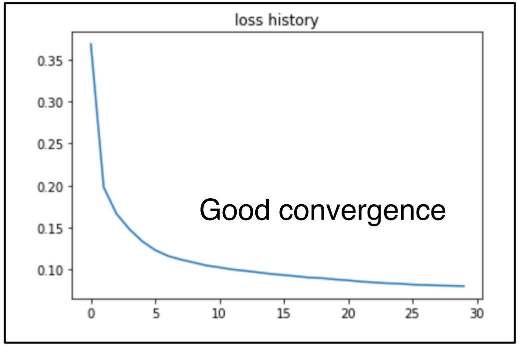
- Train = 90% of non-failures
- Test = 10% of non-failures and 100% of failures

Preliminary Results

Autoencoder weights – 5K params

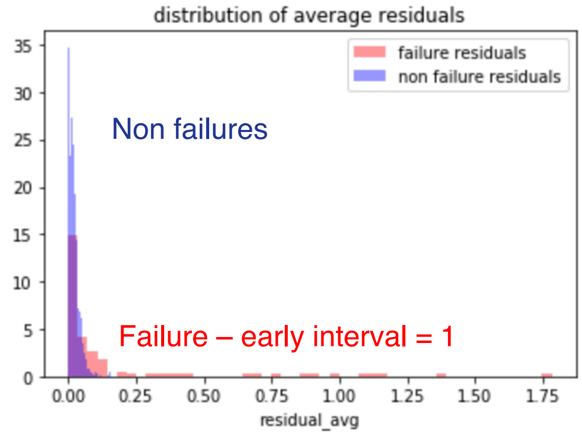
```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                1920
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dense_3 (Dense)              (None, 32)                544
_________________________________________________________________
dense_4 (Dense)              (None, 59)                1947
=================================================================
Total params: 4,939
Trainable params: 4,939
Non-trainable params: 0
```

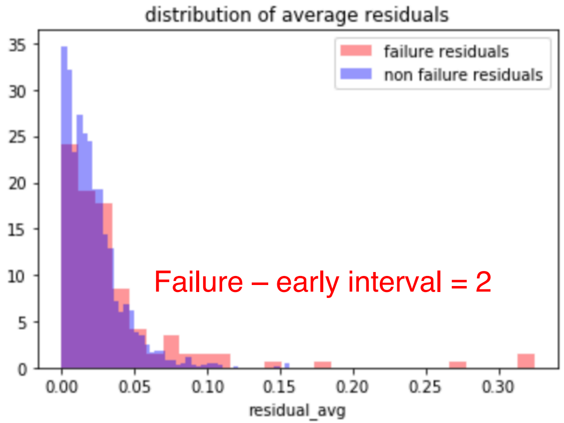
Loss Curve (MSE)



Training time = 1 second per epoch

Average Reconstruction Error





United Technologies Digital

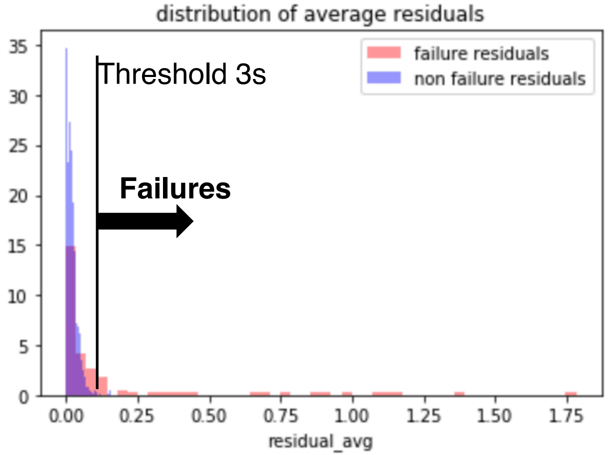
Early interval = 1 shows better separation of failures and non-failures

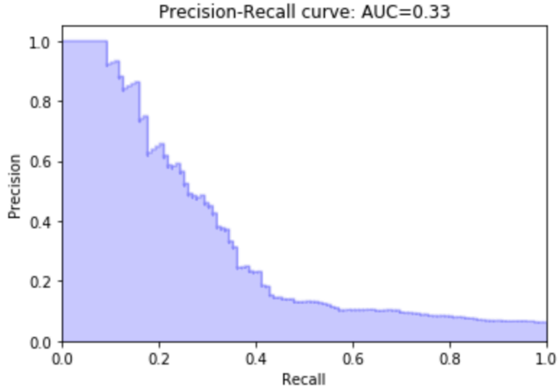
Comparison to XGBoost

- Thresholds calculated based on training data – 3 and 5 sigma
- Perform classification based on threshold
- Early prediction interval = 1

	Logistic Regression	3σ threshold	5σ threshold
Topology	NA	32-16-32	32-16-32
Precision	.51	.554	.84
Recall	.07	.303	.218
F1 Score	.11	.391	.347
ROC AUC	Not provided	.70	
PR AUC	Not provided	.33	

Test Residuals

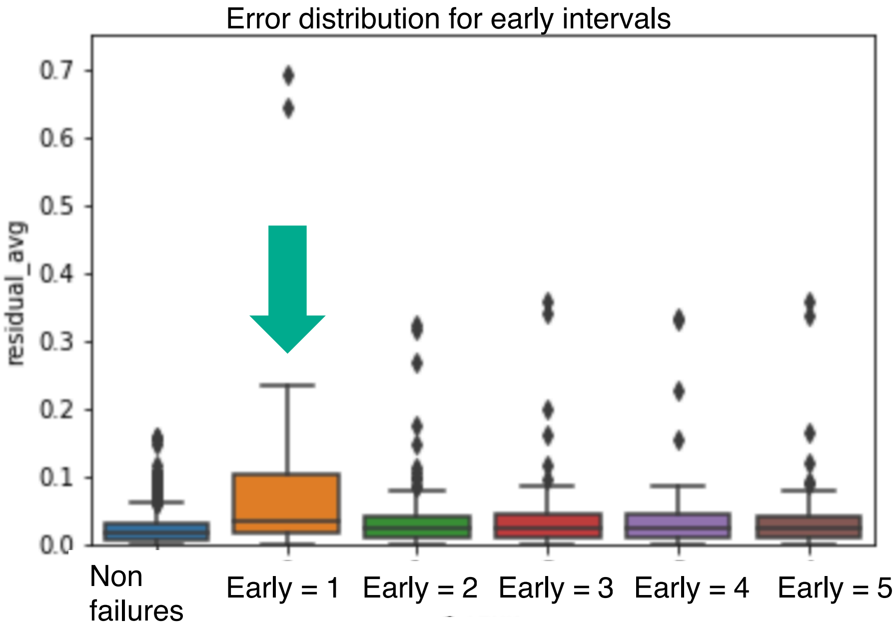




Autoencoders handles class imbalance better than XGB for this dataset

United Technologies Digital

Residual distribution vs early prediction interval



	Early 1	Early 2	Early 3
F1 Score	.391	.147	.1
ROC AUC	.70	.60	.60
PR AUC	.33	.12	.10

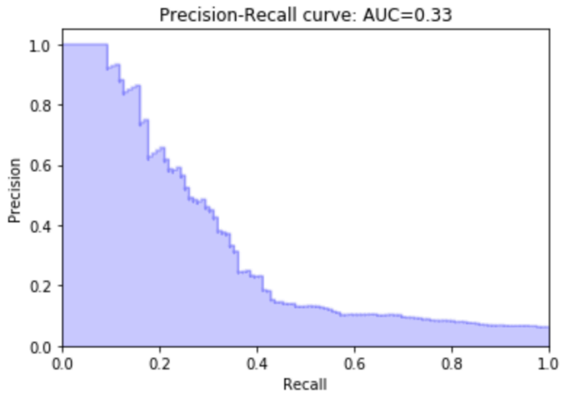
Autoencoder's capability deteriorates as early interval increases

Different Feature Comparison

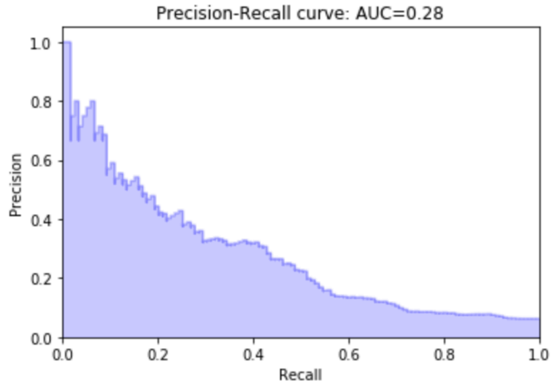
- Threshold = 3 sigma training error
- Early prediction interval = 1
- Batch size = 12
- Epochs = 50
- Topology = tune for each feature set

Hidden Layer Shape	All features - 59	Top 15 XGBoost Features
F1 Score	.391	.192
ROC AUC	.70	.66
PR AUC	.33	.28

All 61 features



Top 15 features



United Technologies Digital

Preliminary Feature selection did not improve model

Different Topology comparison

- Threshold = 3 sigma training error
- Early prediction interval = 1
- Batch size = 12
- Epochs = 50

	Complex	Base Case	Simple
Topology	32-16-8-16-32	32-16-32	16
F1 Score	.368	.391	.257
ROC AUC	.68	.70	.65
PR AUC	.30	.33	.23

United Technologies Digital

3 Layer auto encoder performed the best

Comparison to K means clustering anomaly detection

Model Fitting

Step 1: Chose 5 clusters based on elbow plot for non-failures

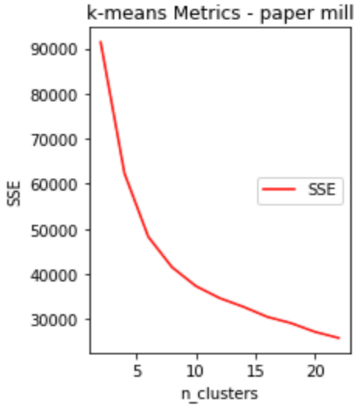
Step 2: Fit 5 clusters on train data
* Find 3 sigma threshold for mean distance to cluster centroid

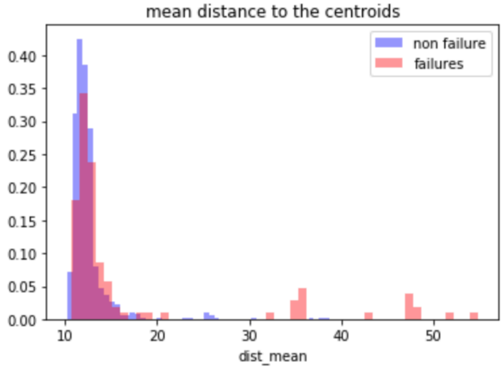
Model Testing

Step 3: Obtain average distance to centroids for test data (failures and non-failures)

Anomaly Detection

Step 4: Classify anomaly based on threshold for average distance to centroid





Early Interval = 1	Autoencoder – 61 variables	Km - sensor inputs - 61 variables	Km - encoded outputs
F1 Score	.391	.253	.071
ROC AUC	.70	.66	.67
PR AUC	.33	.24	.10

- K means on sensor-inputs performed slightly worse than autoencoder
- K means on encodings performed very poorly

United Technologies Digital

Additional Dataset Description

1. Machine Failure Dataset https://bigml.com/user/czuriaga/gallery/dataset/587d062d49c4a16936000810
   - 9000 rows – 81 failures
   - 17 sensor variables
2. Credit Card Fraud Dataset - https://www.kaggle.com/mlg-ulb/creditcardfraud
   - 280K rows – 492 frauds
   - 28 anonymous variables
3. IBM IOT dataset - https://developer.ibm.com/patterns/predict-equipment-failure-using-iot-sensor-data/
   - 900 rows, 393 failures
   - 9 sensors
4. Credit Card Default Dataset - https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
   - 30000 rows, 6600 defaults
   - 25 payment variables
5. Semiconductor Manufacturing - https://archive.ics.uci.edu/ml/machine-learning-databases/secom/
   - 1500 rows, 100 failures
   - 590 sensor variables

Appendix