

Order Statistics Based Training and Scoring Algorithms for Deep Outlier Detection

Ahmet Zahid Balcioglu, Erhan Çene

May 16, 2022

1. Outlier Detection Using Autoencoder Networks
2. Short Comings of Autoencoders
3. Role of Order Statistics
4. Selecting Threshold Kappa Value
5. Order Statistic Augmented Loss
6. Kappa Threshold Early Stopping
7. Experiments

§1 Outlier Detection Using Autoencoder Networks

What is an autoencoder?

- A kind of neural network, in which the dependent and independent variables are same.
- Therefore, important to discourage to optimize for the trivial solution using means such as regularization, randomness, and dimension reduction.
- Among the most popular methods for outlier detection.

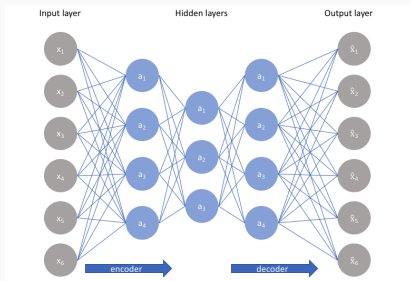


Figure 1: Structure of an autoencoder with three hidden layers.[4]

What is an autoencoder?

Definition 1.1 (Autoencoder Network [7])

An autoencoder network consisting of an encoder network ϕ_e and a decoder network ϕ_d , with the corresponding parameters Θ_e and Θ_d .

$$\begin{aligned} & \text{for} && z = \phi_e(x; \Theta_e), \hat{x} = \phi_d(z; \Theta_d) \\ & \text{minimize} && \sum_{x \in X} \mathcal{L}(x - \phi_d(\phi_e(x; \Theta_e); \Theta_d)) \end{aligned}$$

The optimum parameters for the network are

$$\{\Theta_e^*, \Theta_d^*\} = \arg \min_{\Theta_e, \Theta_d} \sum_{x \in X} \mathcal{L}(x - \phi_d(\phi_e(x; \Theta_e); \Theta_d)).$$

If we choose the most commonly used L_2 distance as our loss function we have

$$\|x - \phi_d(\phi_e(x; \Theta_e^*); \Theta_d^*)\|^2.$$

Outlier Detection with PCA

- PCA and autoencoder share a lot of similarities, in fact when the encoder is linear they learn the same space as PCA if trained under L_2 loss. [5]
- PCA is one of the classical methods used for outlier detection in multivariate data. Most methods using PCA rely on outliers having very little correlation with the principal components.

Outlier Detection Using Autoencoder Networks

Much like Principal Component Analysis, outlier detection using Autoencoder networks is done by learning a lower dimensional representation of the data. Then quantify outlier scores using recovery error. Data points which have a higher recovery error than a certain threshold are considered to be outliers.

$$\mathcal{O} = \{x \in \mathcal{X} : \|x - \hat{x}\| \geq \mathcal{T}\}$$

Algorithm 1: Outlier Detection with Autoencoders

Data: \mathcal{X} , training data

Train the autoencoder using definition 1.1 for optimum parameters Θ_e^* and Θ_d^*

Calculate reconstruction loss for each data point $x \in \mathcal{X}$:

$$\mathcal{L}(x) = \|x - \phi_d(\phi_e(x; \Theta_e^*); \Theta_d^*)\|^2.$$

Select a threshold T based on the calculated losses

Observations above T are labelled as outliers

$$\mathcal{O} = \{x \in \mathcal{X} : \|x - \hat{x}\| \geq T\}.$$

2-Short Comings of Autoencoders

- Complexity of autoencoders allow them to generalize to outliers just as well as the rest of the data, effectively memorizing the data set.[6]
- This is also exacerbated by non-convexity, which makes it difficult to determine when to stop the training process.

Points of Leverage

- Similar to simple linear regression where outliers have a potential to be points of leverage.
- Even at the beginning of the training process data points with most errors tend to be outliers, [6] which makes them behave as potential points of leverage due to their disproportionate contribution to the model early on.

Choice of Loss Function

- Loss function has multiple purposes during training, and inference. As a result, scoring based on loss function introduces some bias to the data.
- It also makes the model dependent upon the choice of loss function.

3- Role of Order Statistics

About Using Order Statistics

- Order statistics allow to treat data independent of its distribution.
- Used in related fields such as extreme value theory.
- **Our objective:** In applications of autoencoders, often some a priori is needed about the dataset to decide on a cut-off point for outliers. Here we will go one step further and decide upon a scoring function using order statistics and eliminate the need for such assumptions.

Motivation: A single outlier

Let X_1, X_2, \dots, X_n be i.i.d. non-negative random variables, and let

$$X_{(1)}, X_{(2)}, \dots, X_{(n-1)}, X_{(n)}$$

be the corresponding order statistics. We can investigate whether $X_{(n)}$ is an outlier by looking at $\frac{X_{(n)}}{X_{(n-1)}}$ and comparing it to $\frac{X_{(i)}}{X_{(i-1)}}$, for $i < n$.

Definition 3.1

Let X_1, X_2, \dots, X_n be i.i.d. non-negative random variables with cumulative distribution function F . We may further assume that these are absolutely continuous, and call the common p.d.f. f . Let $X_{(1)}, \dots, X_{(n)}$ be the corresponding (increasing) order statistics. We are to investigate the problem of outliers in a way that the number of κ -outliers is defined via moving blocks as

$$\zeta_n := n - \min \left\{ i : \sum_{j=1}^i X_{(j)} < \kappa \sum_{j=i+1}^n X_{(j)} \right\} \quad (1)$$

From our definition to quantities are important, the so-called right-hand and left-hand sums:

- $T_{k,n} \equiv T_k = \sum_{i=n-k+1}^n X_{(i)}, \quad 1 \leq k \leq n$
- $S_{m,n} \equiv S_m = \sum_{i=1}^m X_{(i)}, \quad 1 \leq m \leq n$

Subsequently, putting the two together, we investigate probabilities of the form $\mathbb{P}(S_m < \kappa T_{n-m})$ where κ is fixed. We call the random variable

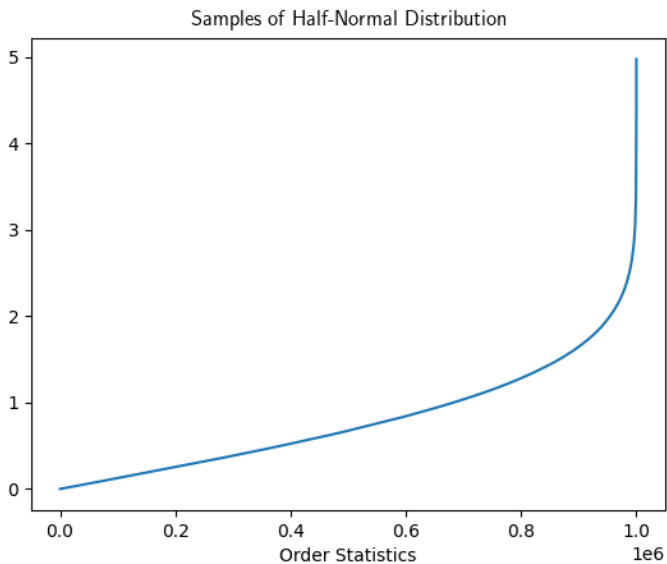
$$R := \mathbb{P}\left(\frac{S_m}{T_{n-m}} \leq \kappa\right)$$

as our outlier statistic.

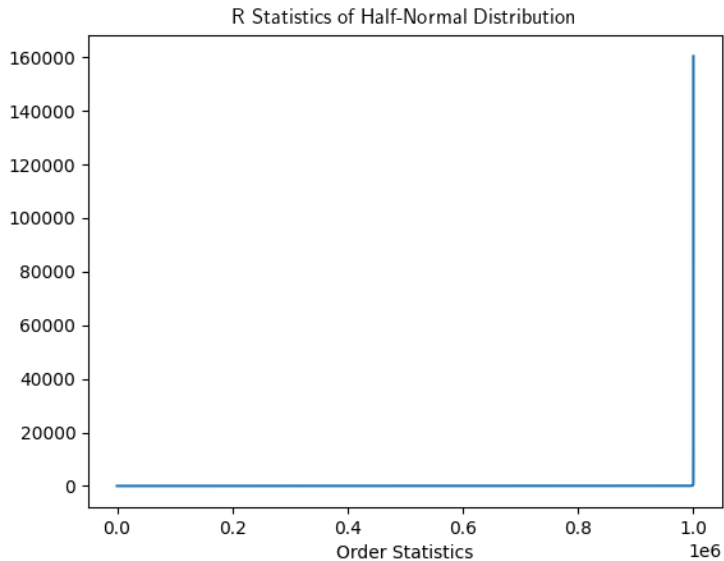
How does this statistic look?

We will look at Half-Normal, Half-Cauchy, and χ^2 distributions.
We will take a sample of 1000000 from each distribution and plot the results for each order statistic.

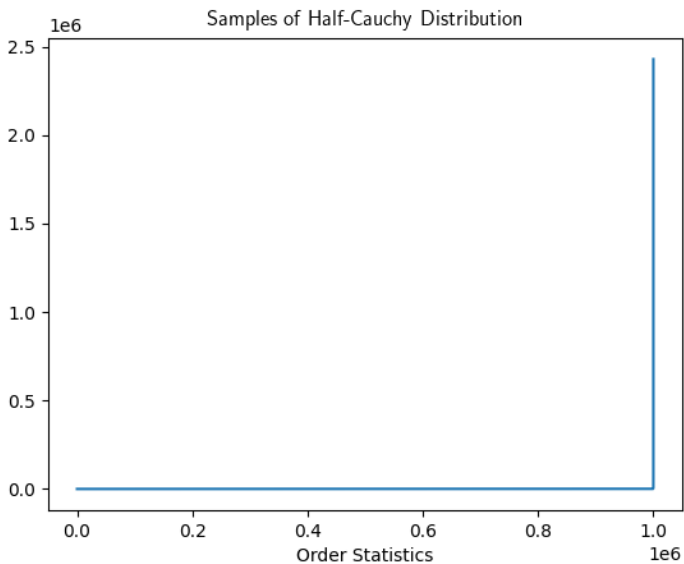
Half-Normal Distribution



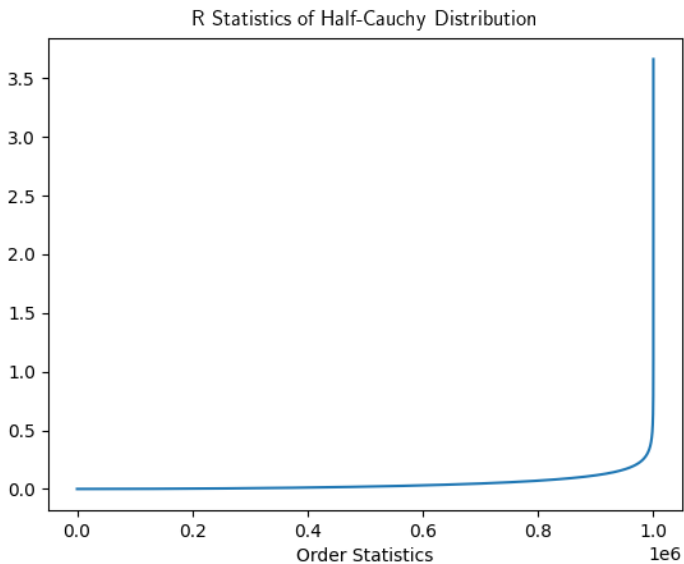
Half-Normal Distribution



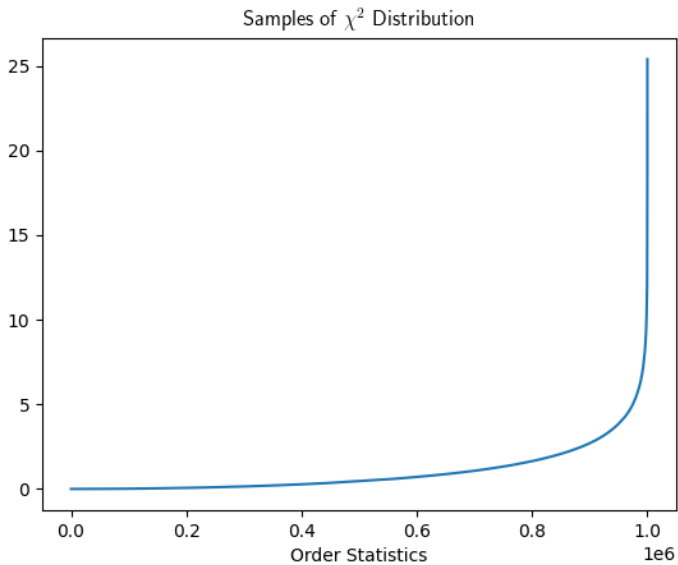
Half-Cauchy Distribution



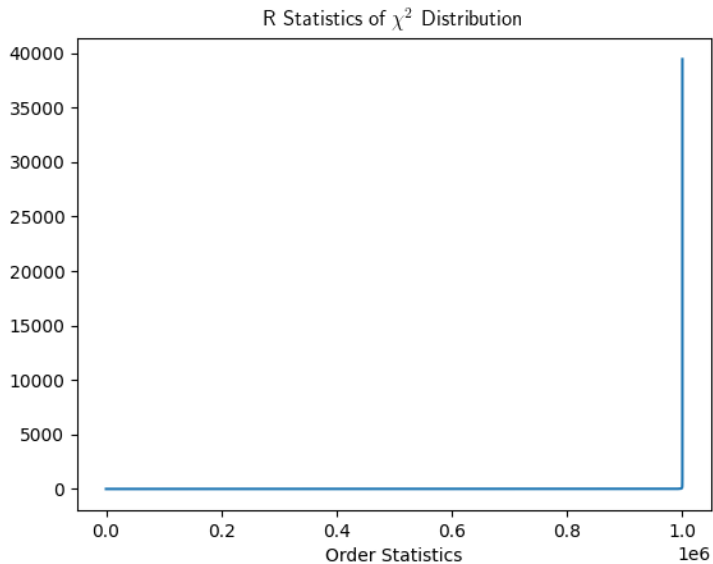
Half-Cauchy Distribution



χ^2 Distribution



χ^2 distribution



4- Selecting Threshold Kappa Value

Elbow Detection

- Ideally, we would like to define the cut-off point with respect to the derivatives of our distribution function, which then can be estimated from a given sample. However, there is no established well-defined notion for the elbow region, which our statistic produces.[1]
- We can leverage the algorithmic work done in elbow detection to estimate the best elbow point.

Many works in the kneecap or elbow detection literature is based on the following pointwise definition of the curvature of a function.

Definition 4.1 (Curvature of a function [8])

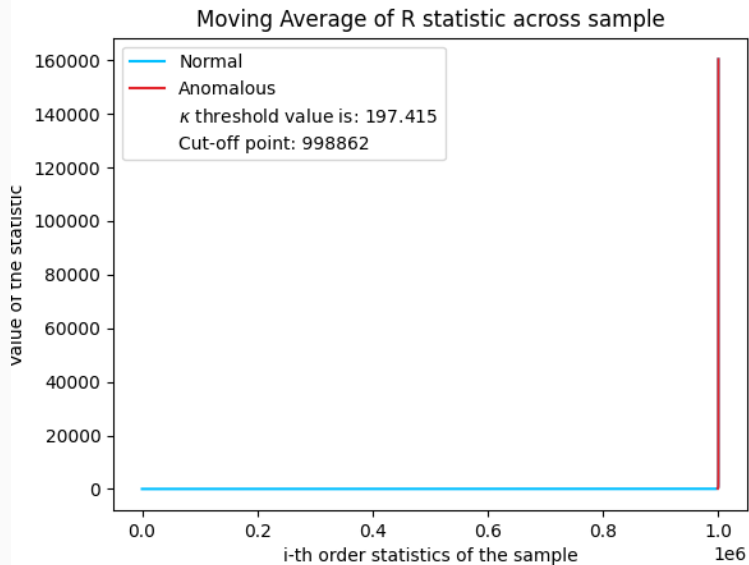
For any continuous function f , there exists a standard closed-form $K_f(x)$ that defines the curvature of f at any point as a function of its first and second derivative:

$$K_f(x) = \frac{f''(x)}{(1 + f'(x)^2)^{\frac{3}{2}}}$$

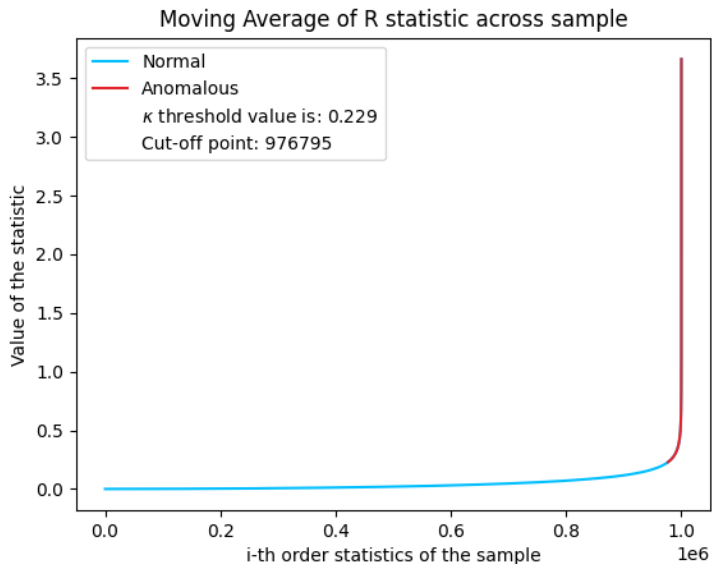
Kneedle Algorithm

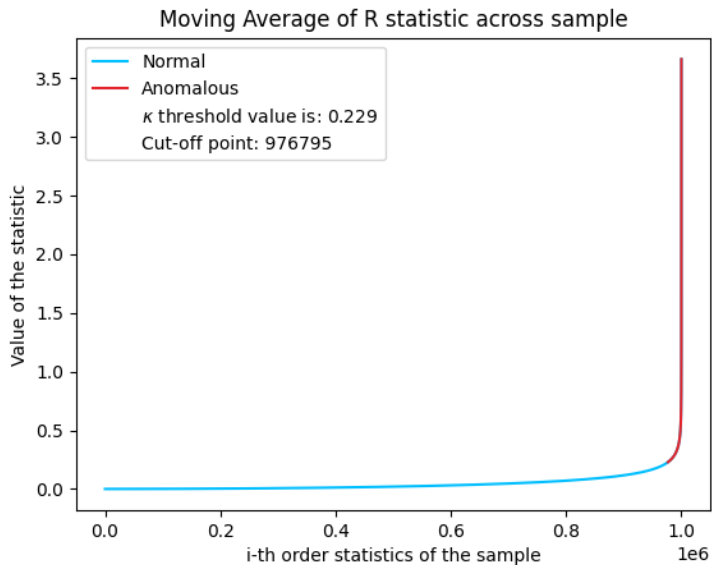
- We will use the kneedle detection algorithm for estimating the "elbow point" of our statistic [8].
- Kneedle algorithm uses the curvature definition 4.1 to find the point of maximum curvature.
- It can work on discrete datasets and uses a sensitivity parameter to single out potential elbows. We choose the sensitivity parameter as 5.0 in our experiments.

Half-Normal Distribution



Half-Cauchy Distribution





5- Order Statistic Augmented Loss

Motivation for using R-statistic in Autoencoders

- We have already mentioned that one particular weaknesses of autoencoders is that due to the high complexity of the model, it can generalize to outliers just as well as normal observations.
- It has also been reported that early in the optimization process the gap between outliers and normal observations is the most noticeable.[6]
- We can use our statistic throughout the optimization procedure and use the statistic to create a weighted some during the calculation of the loss function. In this way we can regularize the contribution of each observation to the loss by its outlier score, and retain the initial distinction between outliers and the rest of the data while optimizing for a model with greater generalizability.

Order statistic augmented loss function

- In order to make use of our statistic, we sample some proportion of the data to use as a scoring system for the entire data.
- We recalculate R statistic throughout the sample and also calculate the κ threshold value.
- For each observation with **greater loss than the κ threshold**, contribution to the loss is adjusted by the **inverse of its outlier score**.

Algorithm 2: Order Statistic Adjusted Loss function

Data: `outlier_sample` := Sample a portion of the data

Initialize uniform loss weights;

for *epoch in Total Epochs* **do**

for *batch in Data* **do**

 Calculate loss; Estimate loss weights using
 `outlier_sample`;

end

 Calculate R -statistic for the `outlier_sample`;

end

6- Kappa Threshold Early Stopping

Early Stopping

- Autoencoders are not convex statistical models, as a result their training takes longer and do not necessarily converge to a global minimum. Early stopping is a method which can determine the time to finish the training when the model is successful enough. It works by monitoring a metric pertinent to model success.
- Autoencoders are unsupervised methods, which limits the options for choosing a reliable metric for early stopping. One likely candidate is the loss function itself, [6] but the loss function is also used for other algorithms that dynamically change optimizer parameters such as learning rate.

κ Threshold Value

- κ threshold value can be good metric for early stopping.
- Stability of κ shows that the model has stopped distinguishing between outliers and the normal data.
- During training, we would expect κ to decrease as the loss is decreasing and stop when the changes in the loss no longer contribute to distinguishing outliers.

Problems with using κ

- Not a summary statistic, but a pointwise estimate.
- Unlike most metrics used in early stopping κ is not consistent.

What can we do?

- Block aggregation would help with inconsistencies and would trim the sudden movements in the metric.
- Rather than testing for increases or decreases in the metric like most early stopping algorithms, we can test for a stability condition.

Early Stopping algorithm using κ threshold

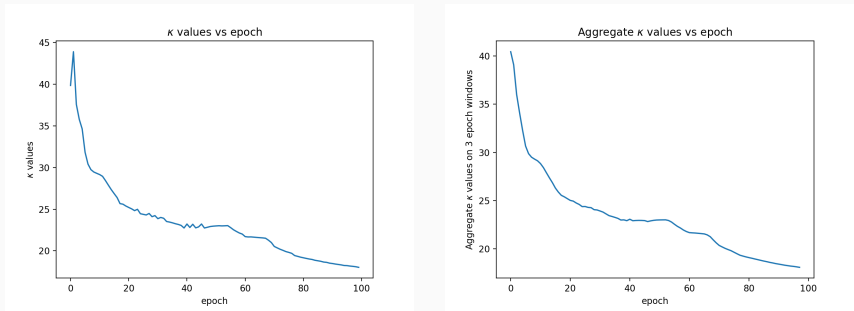


Figure 2: Unaltered and 3 epoch aggregated κ -threshold values during the first 100 epochs of a training session.

Algorithm 3: Stopping condition for Early Stopping

Function `check_improvement(κ_epochs , $block_size$, $early_iteration_limit$):`

```
  if  $size(\kappa\_epochs) \leq early\_iteration\_limit$  then  
    | return continue
```

```
  end
```

```
   $block\_k = block\_sum(\kappa\_epochs, block\_size)$ 
```

```
  if  $block\_sum$  not increasing then  
    | return stop
```

```
  else
```

```
    | return continue
```

```
  end
```

Algorithm 4: Early Stopping algorithm using κ threshold

Data: Data := Training data

Data: outlier_sample := Sample a portion of the Data

int early_iteration_limit

int block_size

κ _epochs = []

for *epoch* in *Total Epochs* **do**

for *batch* in *Data* **do**

 | Calculate loss

end

 Calculate R -statistic for the outlier_sample

 Calculate κ -threshold for the epoch

κ _epochs.append(κ -threshold)

 check_implevoment(κ _epochs, block_size,
 early_iteration_limit)

end

7- Experiments

Comparative studies

- As our algorithm is built on top of already established methods in autoencoders, we can test in a comparative way.
- We will take a well-known benchmark dataset.
- Compare the results of autoencoders, autoencoders with order statistic scoring method, finally we will add augmented loss and early stopping.

Dataset: MNIST

- MNIST is an image data set consisting of handwritten digits.
- Use sampling to create a outlier data set.
- We select 0 to represent normal instances
- Undersample from 4 for outliers. Final data has a 2% outlier ratio.



Figure 3: Examples from MNIST data set.[3]

Experiment Details

- We use the following architecture used by [2]. Autoencoder network with linear layers, the weights are respectively of shapes (784, 256), (256, 32), (32, 256), (256, 784).
- Loss: MSE
- Optimizer: Adam with learning rate 10^{-3}
- Batch size: 2991
- Epochs: 100

Results

Algorithm	$F1$	Confusion Matrix	
Simple Autoencoder	0.639	5879 43	44 77
only with order statistic scoring	0.846	5923 32	0 88
with augmented loss and early stopping	0.857	5923 30	0 90

References






Mário Antunes, Diogo Gomes, and Rui L. Aguiar. “Knee/Elbow Estimation Based on First Derivative Threshold”. In: *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*. 2018, pp. 237–240. DOI: 10.1109/BigDataService.2018.00042.



Gabriel B Cavallari, Leonardo SF Ribeiro, and Moacir A Ponti. “Unsupervised representation learning using convolutional and stacked auto-encoders: a domain and cross-domain feature space analysis”. In: *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE. 2018, pp. 440–446.



Juan Cazala. *mnist*. [Online; accessed May 12, 2022]. 2017. URL: <https://camo.githubusercontent.com/01c057a753e92a9bc70b8c45d62b295431851c09cffadf53106fc0a687474703a2f2f692e7974696d672e636f6d2f76692f30514933786>

-  Jordan Jeremy. *Introduction to autoencoders*. [Online; accessed April 3, 2022]. 2018. URL: <https://www.jeremyjordan.me/content/images/2018/03/Screen-Shot-2018-03-07-at-8.24.37-AM.png>.
-  Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.
-  Nicholas Merrill and Azim Eskandarian. “Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning”. In: *IEEE Access* 8 (2020), pp. 101824–101833. DOI: 10.1109/ACCESS.2020.2997327.



Guansong Pang et al. “Deep Learning for Anomaly Detection”. In: *ACM Computing Surveys* 54.2 (2021), 1–38. ISSN: 1557-7341. DOI: 10.1145/3439950. URL: <http://dx.doi.org/10.1145/3439950>.



Ville Satopaa et al. “Finding a “Kneedle” in a Haystack: Detecting Knee Points in System Behavior”. In: *2011 31st International Conference on Distributed Computing Systems Workshops*. 2011, pp. 166–171. DOI: 10.1109/ICDCSW.2011.20.