2023/AM01
Detection of Anomalous Behaviour in Industrial Robot

The Great Detector

Objective of Project

Objective of Project

1. Implement an autoencoder model that has capacity to be able to generate same output with input.

2. Demonstrate the importance of evaluation method selection in TAD.

3. Compare traditional training techniques by adversarial training with adversarial autoencoder.

Data

Data

Data Normal: The data that collected during production with normal velocity.

Data Slow: The data that collected during production with slow velocity.

Features: 'action', 'machine_nameKuka Robot_apparent_power', 'machine_nameKuka Robot_current' 'machine_nameKuka Robot_frequency', 'machine_nameKuka Robot_phase_angle', 'machine_nameKuka Robot_power', 'machine_nameKuka Robot_power_factor', 'machine_nameKuka Robot_reactive_power', 'machine_nameKuka Robot_voltage', 'sensor_id1_AccX' 'sensor_id1_AccY' 'sensor_id1_AccZ', 'sensor_id1_GyroX', 'sensor_id1_GyroY' 'sensor_id1_GyroZ' 'sensor_id1_q1' 'sensor_id1_q2', 'sensor_id1_q3' 'sensor_id1_q4' 'sensor_id1_temp', 'sensor_id2_AccX' ...

86 features

From 86 features;

- **10 feat** **33)**
- **76 feat**

Should we use all the features?

Data

Data Normal: T… …ected
during prod… …elocity.

Features: 'ac… …eKuka
Robot_frequency', … …bot_power_factor',
'machine_nameKuk… …nsor_id1_AccZ',
'sensor_id1_GyroX', … 'sensor_id1_temp',
'sensor_id2_AccX' …

From 86 fea…

- **10 feat… 33)**
- **76 feat…**

Should we use all the features?

Since the task is reconstruction, base time-series anomaly detection. The features which has less than 31 different attributes is extracted.

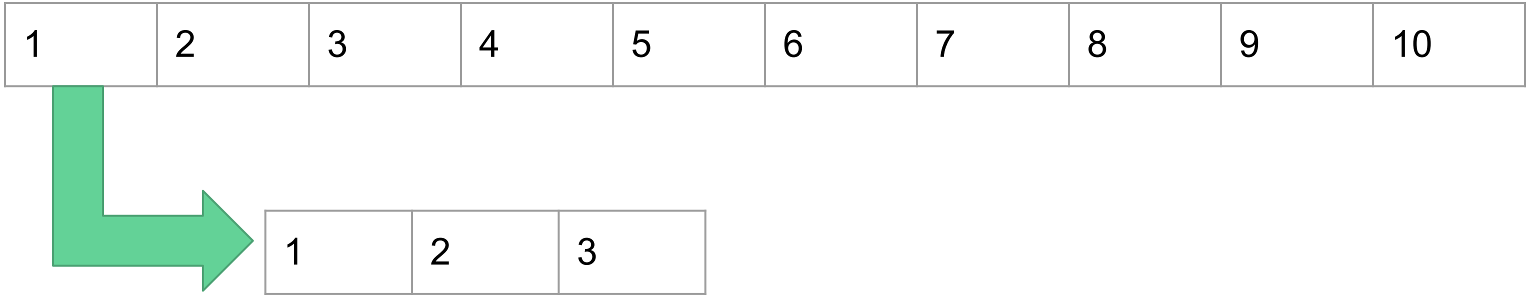
Therefore, there are 79 features in the prepared training and test datasets.

Data Preparation

Data Preparation

For each sample of data, a time window is prepared with given length of window.

```
data_train_all = np.array([data_train_scaled[i:i+width_window,:] for i in range(0, len(data_train_scaled)-width_window)]
data_test_all = np.array([data_test_scaled[i:i+width_window,:] for i in range(0, len(data_test_scaled)-width_window)])
```

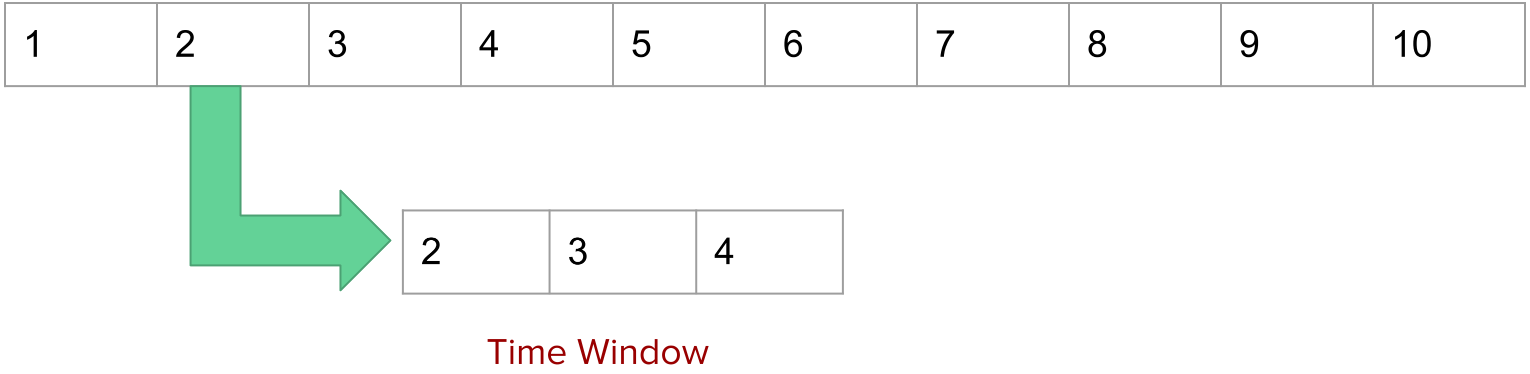


Time Window with length 3.

Data Preparation

For each sample of data, a time window is prepared with given length of window.

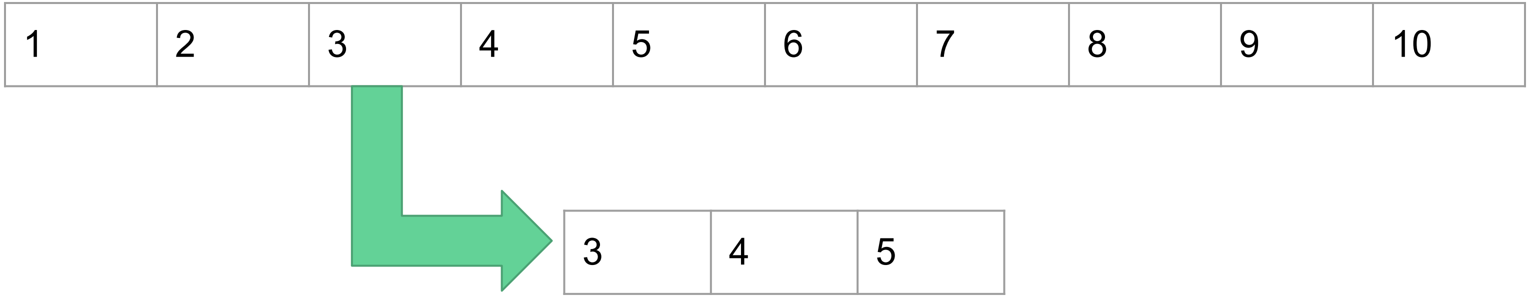
```
data_train_all = np.array([data_train_scaled[i:i+width_window,:] for i in range(0, len(data_train_scaled)-width_window)]
data_test_all = np.array([data_test_scaled[i:i+width_window,:] for i in range(0, len(data_test_scaled)-width_window)])
```



Data Preparation

For each sample of data, a time window is prepared with given length of window.

```
data_train_all = np.array([data_train_scaled[i:i+width_window,:] for i in range(0, len(data_train_scaled)-width_window)]
data_test_all = np.array([data_test_scaled[i:i+width_window,:] for i in range(0, len(data_test_scaled)-width_window)])
```



Time Window

Data Preparation

For each sa window.

```
data_train_all = r d)-width_window)]
data_test_all = r width_window)])
```

1

Then the prepared time windows are shuffled.

- 60% data for training.
- 20% data for validation.
- 20% data to mix with test data. (This will be mentioned with details in next slides.)

Time Window

New Test Set

Since slow data includes only anomalies, new test set prepared. And, part of normal data cropped and concatenated to the end of data slow. And new labels are created according to it.

```
train data shape: (23374, 50, 86)
test data shape: (4148, 50, 86)
label shape: (4148, 1)
test data prepared: (8296, 50, 86)
label prepared: (8296, 1)
```

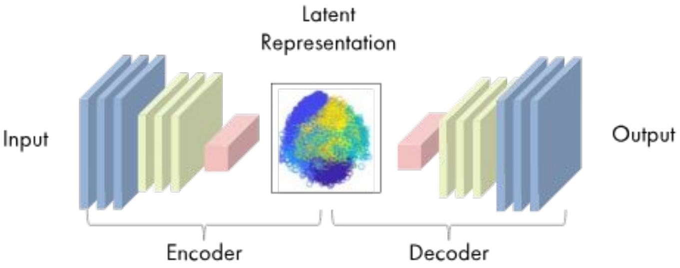
10% of data has taken to increase flexibility in early steps of project.

Samples from normal data concatenated to test data

New test data is created by 50% slow and 50% normal data

Implementation of Autoencoder

Autoencoder

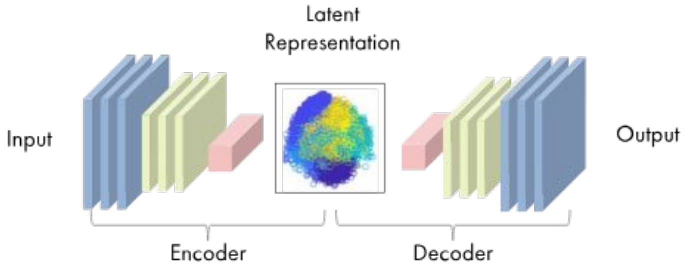


An autoencoder is a type of artificial neural network used to learn efficient codings of unlabelled data.

Requirements;

- Estimated 300,000 trainable parameters at least, to be able to reconstruct same output with input
- Efficient reconstruction ability

Autoencoder



An autoencoder is a type of artificial neural network used to learn efficient codings of unlabelled data.

Suggested by reference paper:

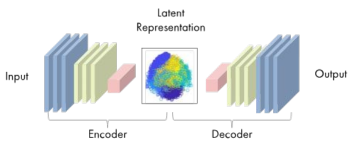
Single-layer LSTM autoencoder

Selected:

5-layer LSTM autoencoder

Complexity of Data

LSTM proofed its performance



Autoencoder - Baseline

- 476,199 trainable parameters
- Latent size is 10
- 5 LSTM layers for encoder, 5 for decoder
- Loss is Mean Squared Error (MSE)
- Optimizer is Adam with learning rate equals 10^{-4}

Faced by infinite loss issue

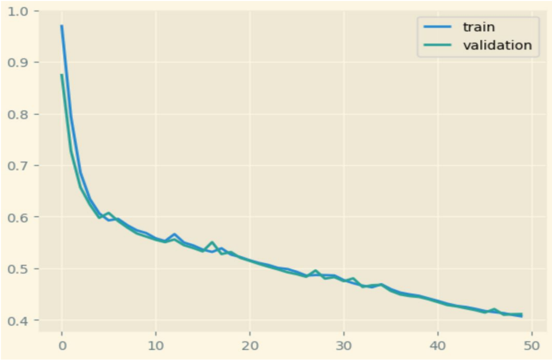
1. Pre-processing; Data has been normalized by standard scaler

Issue has been got better, but not solved totally

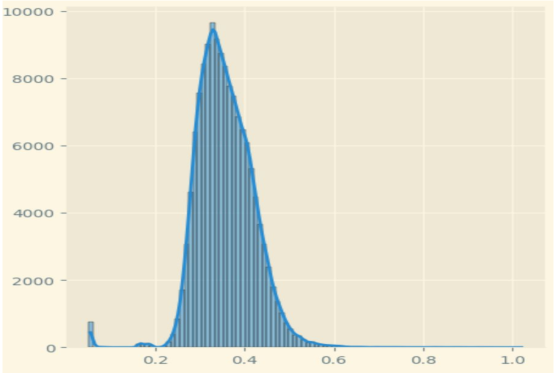
2. Normalization Layers were added after each LSTM layer.

Solved

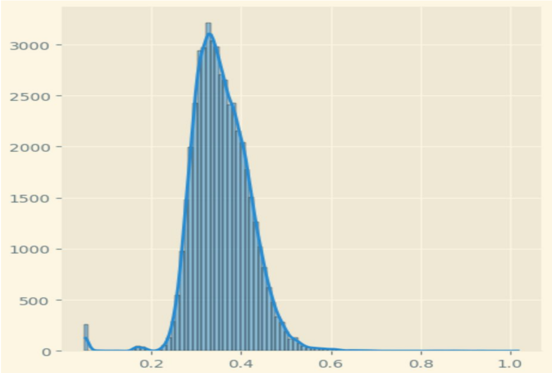
Baseline Training Details



Train & validation sets' losses during training.

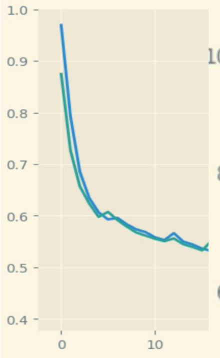


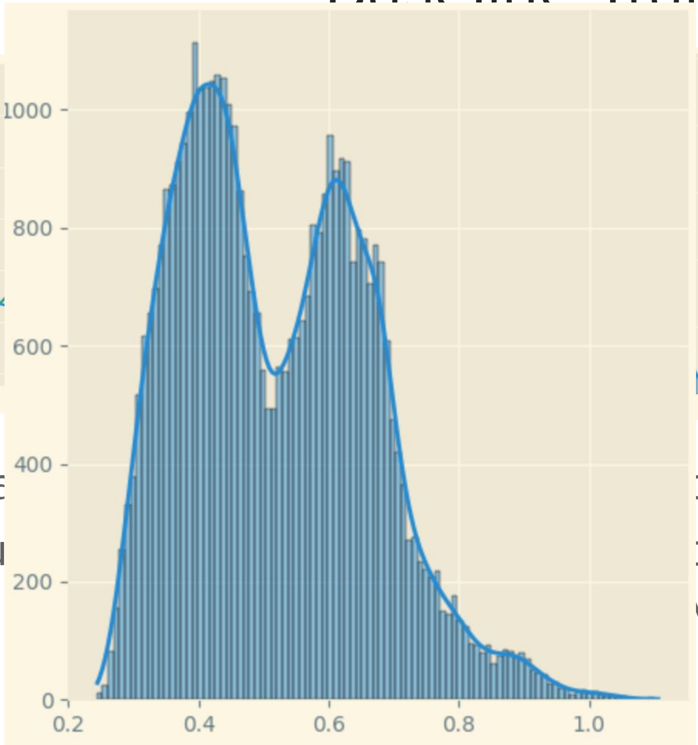
Reconstruction costs in training set for average of features. (Mean: 0.3561)

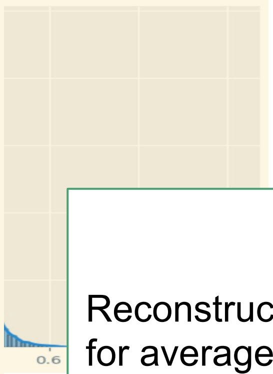


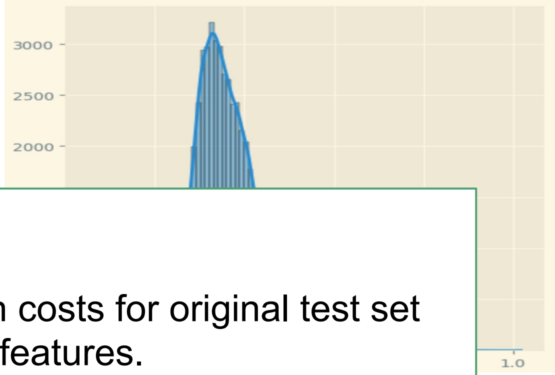
Reconstruction costs in validation set for average of features. (Mean: 0.3562)

Baseline Training Details









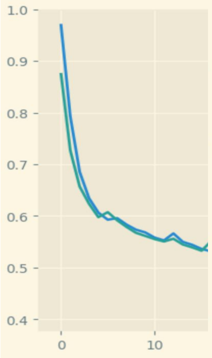
Reconstruction costs for original test set for average of features.
(Mean: 0.52)

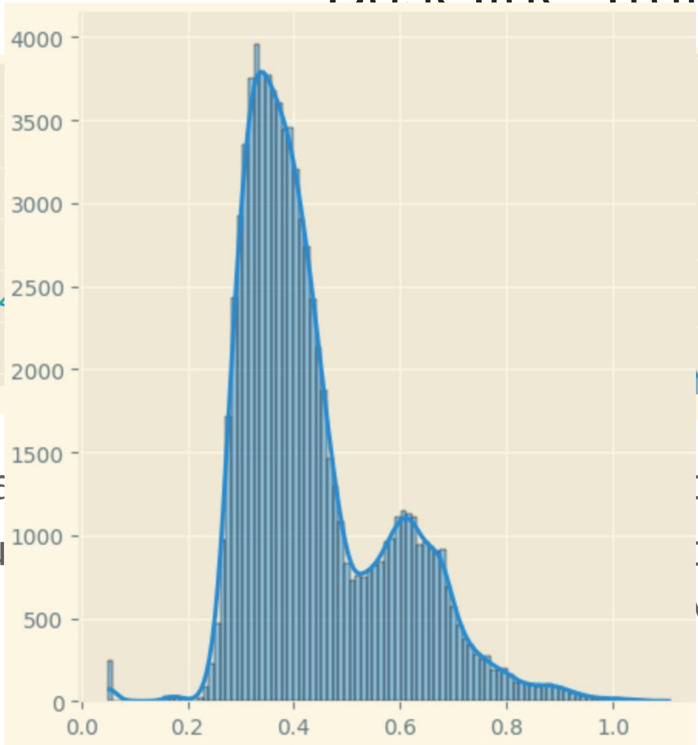
Train & va
losses du

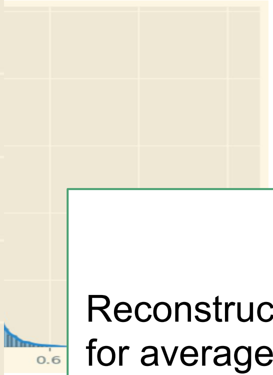
on c
or average
Mean:

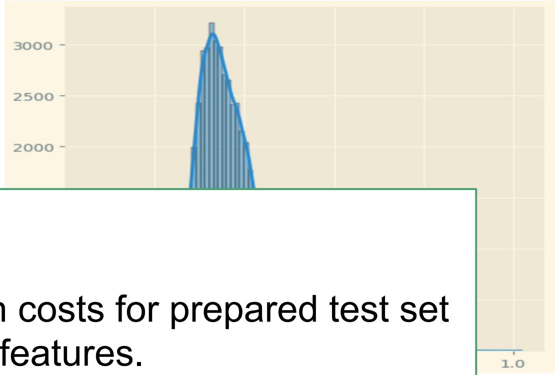
in
validation set for
average of features.
(Mean: 0.3562)

Baseline Training Details









Reconstruction costs for prepared test set for average of features.
(Mean: 0.437)

Train & va
losses du

on c
or average
Mean:

in
average of features.
(Mean: 0.3562)

Evaluation Method Selection

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

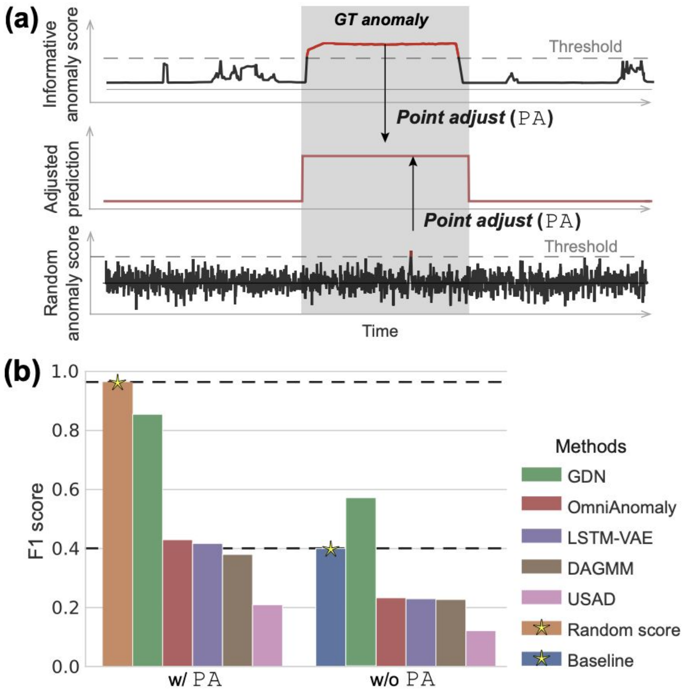
Evaluation Method Selection

Point Adjustment (PA):

If at least one moment in a contiguous anomaly segment is detected as an anomaly, the entire segment is then considered to be correctly predicted as an anomaly.

Most of the Time-Series Anomaly Detection (TAD) methods measure the F1 score after applying this peculiar evaluation protocol.

Greatly overestimates the detection performance.



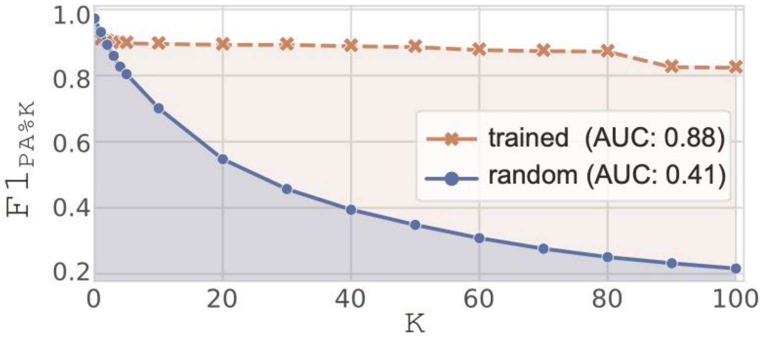
Evaluation Method Selection

PA%K:

Apply PA to the set only if the ratio of the number of correctly detected anomalies in the set to its length exceeds the PA%K threshold, K.

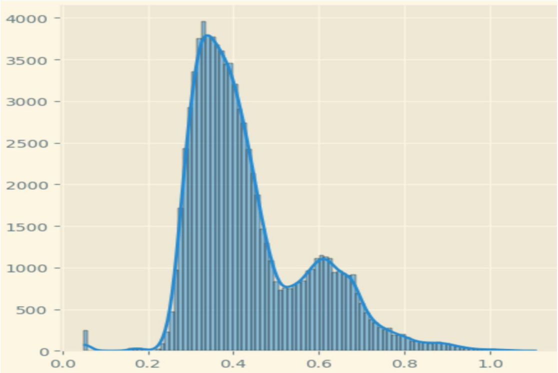
Mitigate the overestimation effect of $F1_{PA}$ & the possibility of underestimation of F1.

K can be selected manually between 0 and 100 based on prior information. (If test labels are reliable, higher K. And vice versa.)

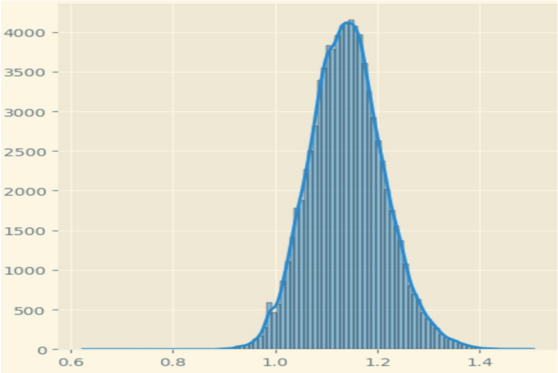


Results of baseline

Loss Distributions



Reconstruction costs in prepared test data for mean of features.

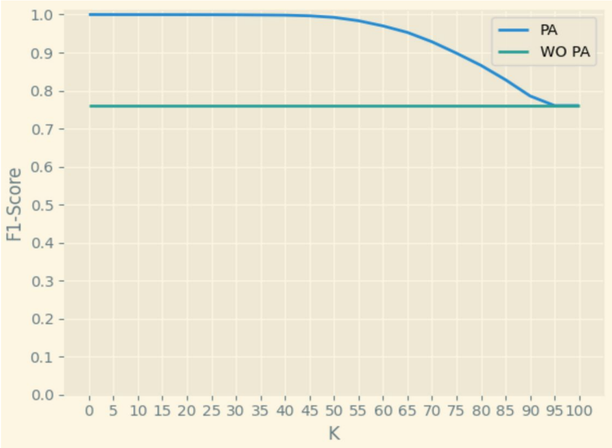


Reconstruction costs in prepared test data for mean of features.

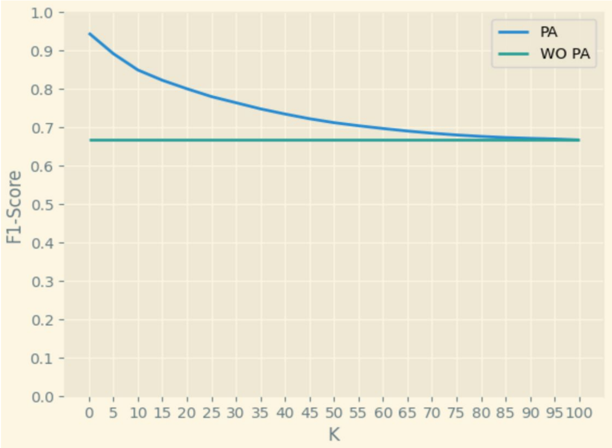
Untrained Model

Best F1-Score

Trained Model



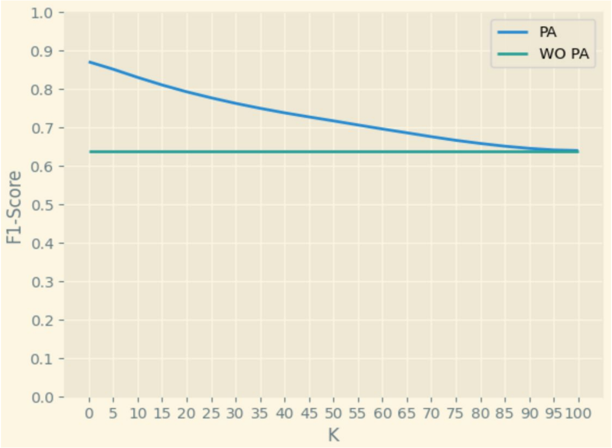
Untrained Model



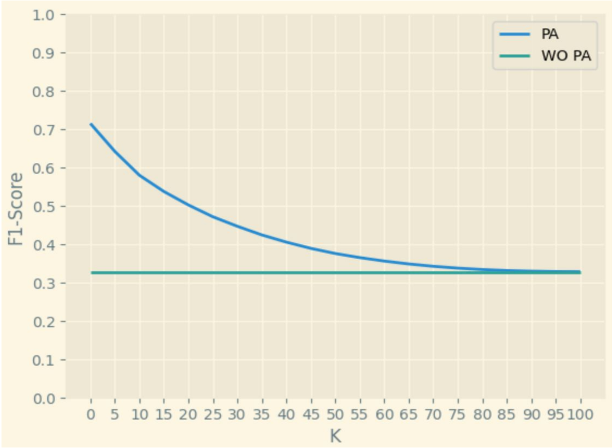
The x-axis represents the K values. The y-axis stands for best F1 scores.

Mean F1-Score

Trained Model



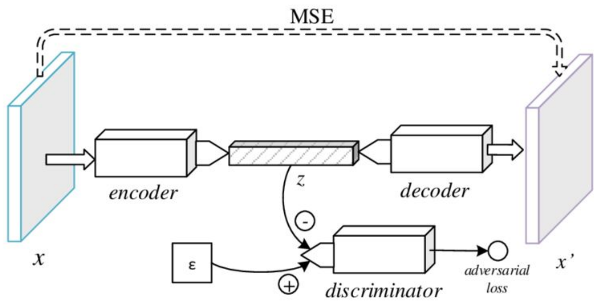
Untrained Model



The x-axis represents the K values. The y-axis stands for mean F1 scores.

Implementation of Adversarial Autoencoder

Adversarial Autoencoder



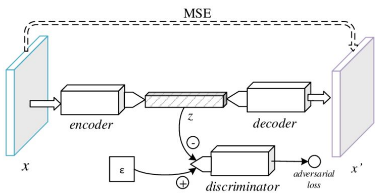
The Adversarial Autoencoder (AAE) is a brilliant concept that combines the autoencoder architecture with GAN's adversarial loss notion. It works similarly to the Variational Autoencoder (VAE), except instead of KL-divergence, it utilizes adversarial loss to regularize the latent code.

Objective:

- To be able to perform same or better comparison to baseline model

Challenges:

- To be able to train and properly optimize the model parameter



Adversarial Autoencoder

Autoencoder: Same autoencoder with baseline will be used to be able to present differences that are made by adversarial training.

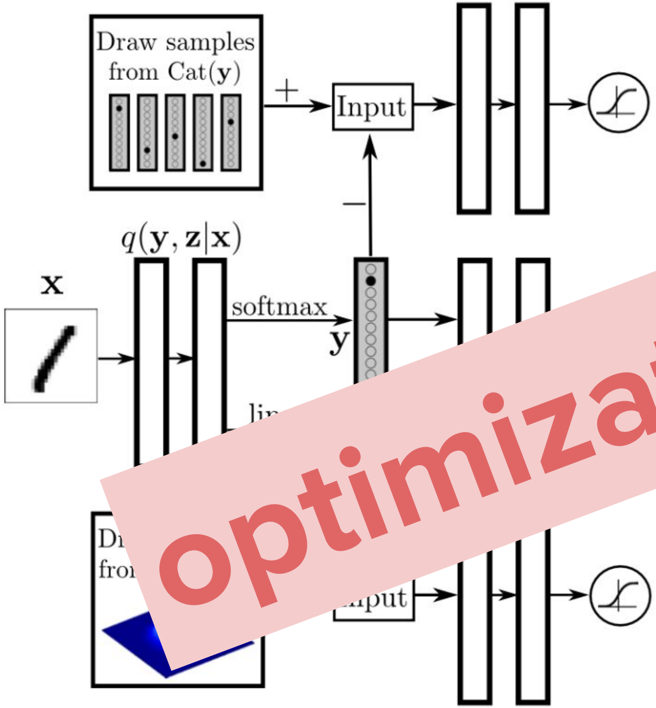
Discriminator: A model with 4 fully connected layers with Adam optimizer.

Discriminator loss function:

```
def discriminator_loss(real_output, fake_output, loss_weight):
    loss_real = cross_entropy(tf.ones_like(real_output), real_output)
    loss_fake = cross_entropy(tf.zeros_like(fake_output), fake_output)
    return loss_weight * (loss_fake + loss_real)
```

Generator: Encoder of the autoencoder.

Semi-Supervised Adversarial Autoencoder



2 separate a

First

gorical distribution on

representation

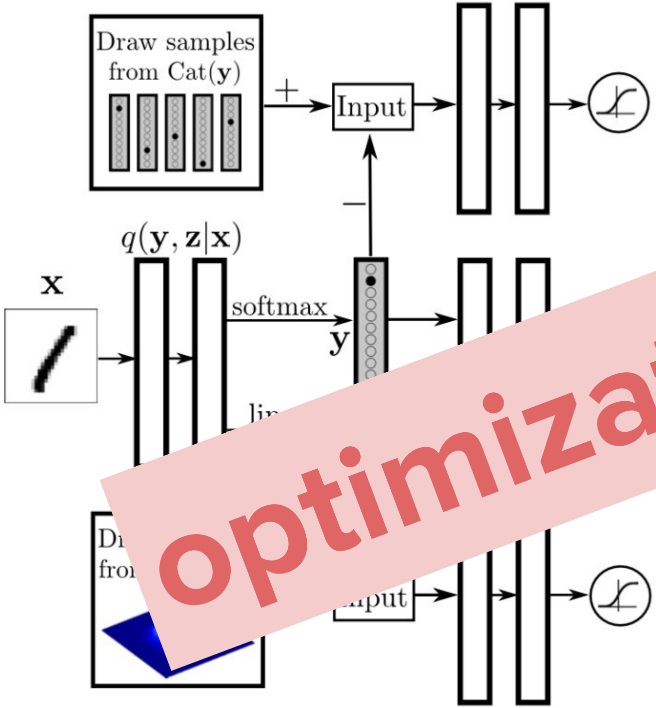
latent class variable y does not carry any style information

- aggregated posterior distribution of y matches the Categorical distribution

optimization on decoder

UTILIZED

Semi-Supervised Adversarial Autoencoder



2 separate a

Seco

us distribution on the

presentation

ensures the latent variable z is a continuous Gaussian variable

optimization on encoder

NON-USED

AAE Training Details

Since the tasks that should be executed are not the same difficulty:

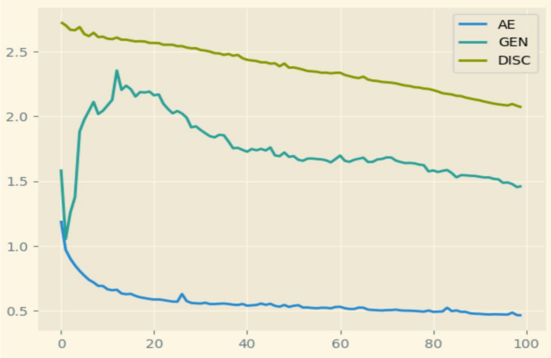
- Discriminator: to classify real and fake time-series
- Generator: to fool discriminator
- Autoencoder: to produce the same outputs with inputs

The balance between the models should be kept with weighted learning and different learning rate selection.

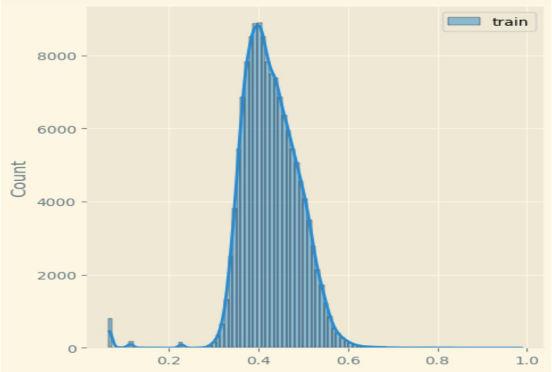
Weights: 0.2 for generator and discriminator, 1 for autoencoder

Learning rates: 10^{-4} for generator and autoencoder, 10^{-5} for discriminator

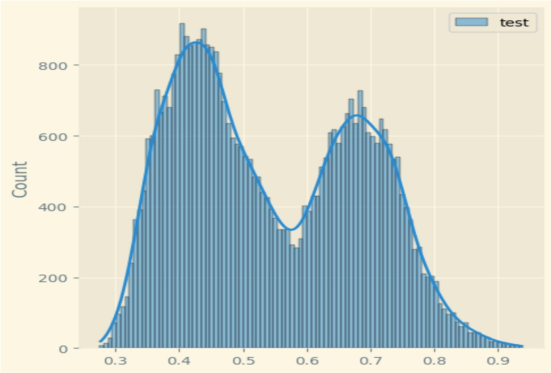
AAE Training Details



Losses of autoencoder, generator and discriminator during training.

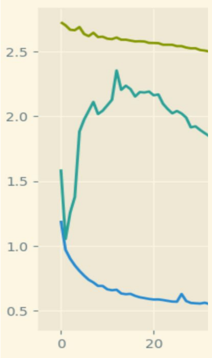


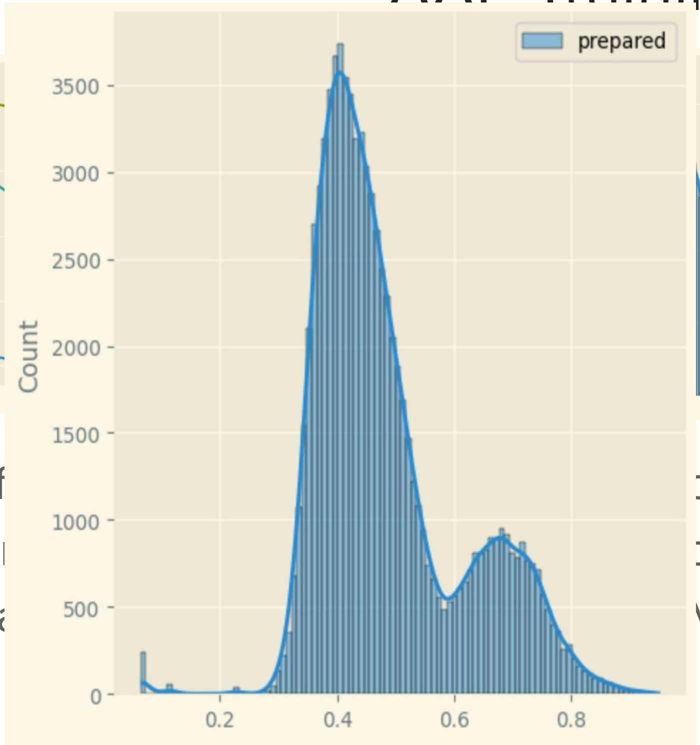
Reconstruction costs in training set for average of features. (Mean: 0.424)

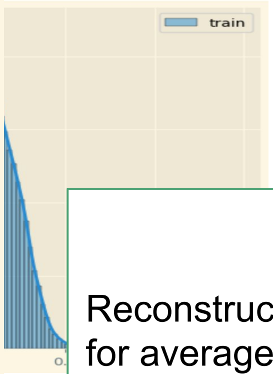


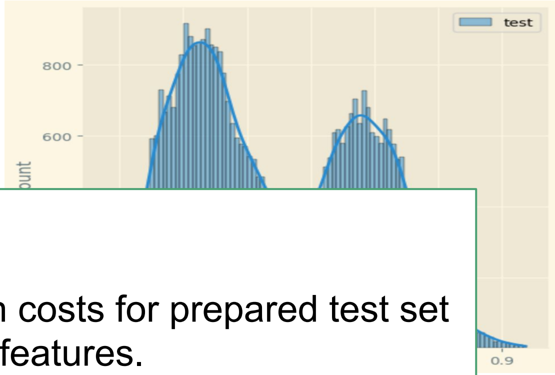
Reconstruction costs in test set for average of features. (Mean: 0.544)

AAE Training Details









Reconstruction costs for prepared test set for average of features.
(Mean: 0.484)

Losses of generator discrimina training.

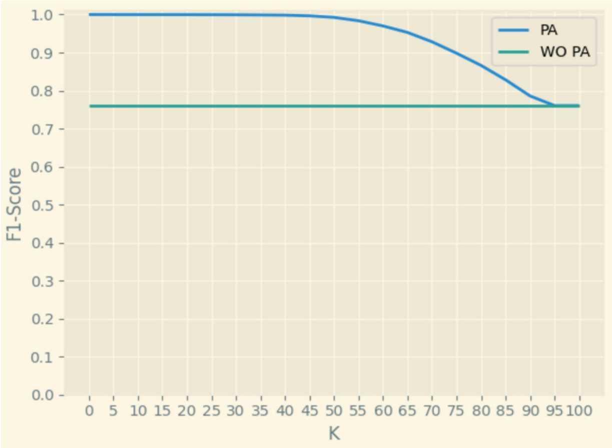
on or a Mean:

in features. (Mean: 0.544)

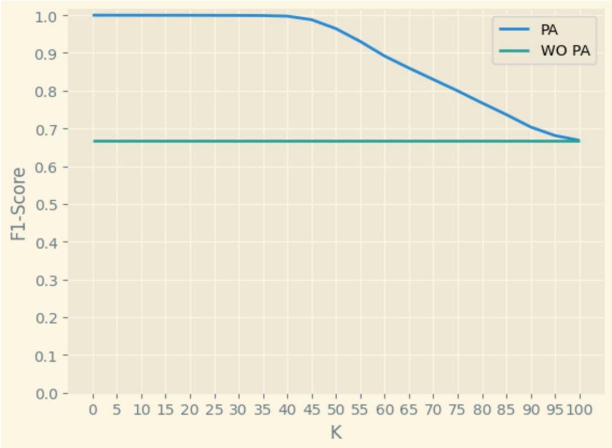
Results of AAE

Best F1-Score

AE Model



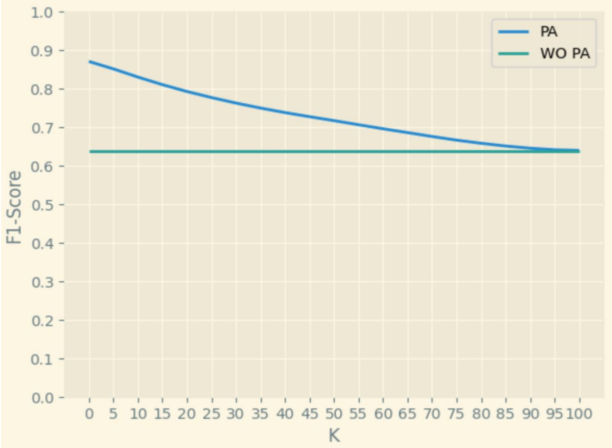
AAE Model



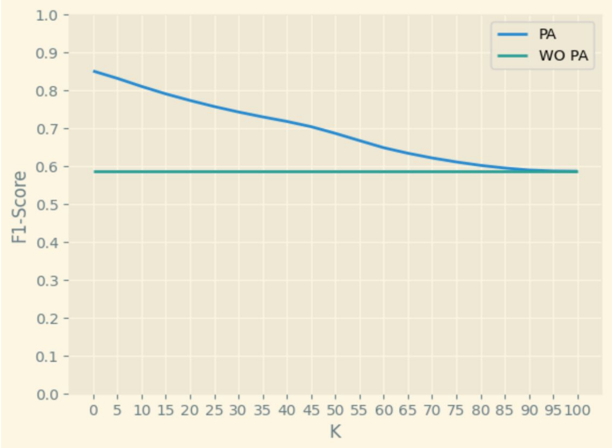
The x-axis represents the K values. The y-axis stands for best F1 scores.

Mean F1-Score

AE Model



AAE Model



The x-axis represents the K values. The y-axis stands for mean F1 scores.

Further Improvements

An Issue: Temporal Anomalies

Autoencoder models often become able to well reconstruct also the anomalies in the data.

This phenomenon is more evident when there are anomalies in the training set.

Solution:

Train autoencoders is to ignore anomalies and minimize the reconstruction error on normal data.

Autoencoder-SAD

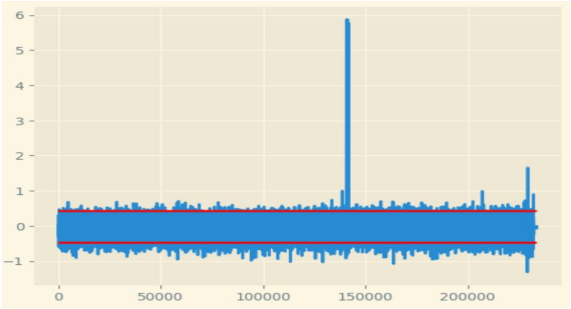
To achieve this, first, the loss function should be changed accordingly.

$$\mathcal{L}(\mathbf{x}) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 \quad \Rightarrow \quad \mathcal{L}_F(\mathbf{x}) = (1-y)\cdot||\mathbf{x}-\hat{\mathbf{x}}||^2 + \lambda\cdot y\cdot||F(\mathbf{x})-\hat{\mathbf{x}}||^2$$

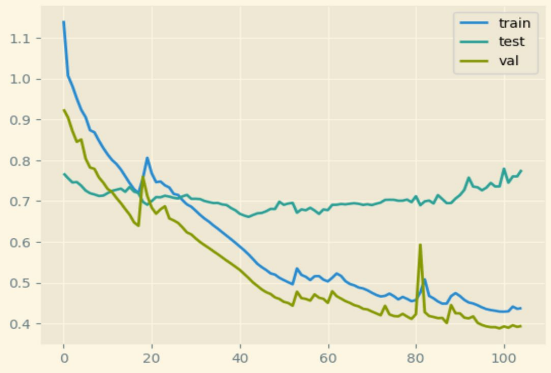
Second, the temporal anomalies should be labelled.

The most radical 1% of readings will be labelled as temporal anomaly for each feature.

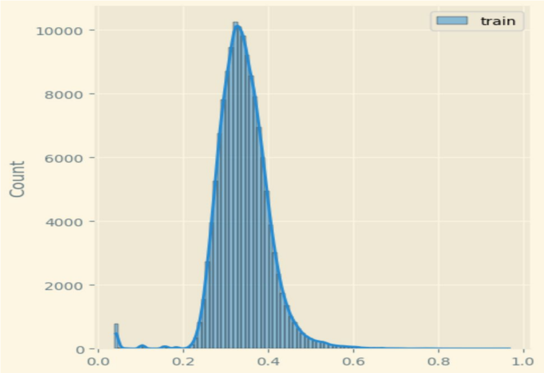
In total, 5% samples will be labelled as temporal anomaly.



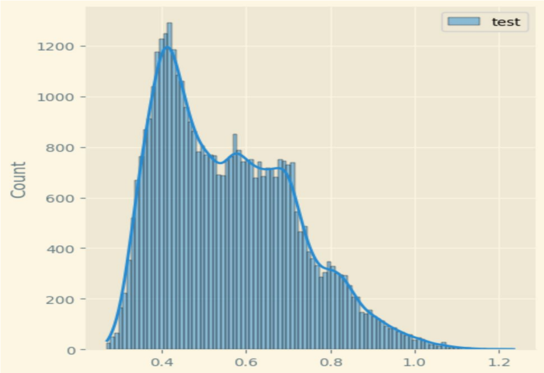
AE-SAD Training Details



Train, validation and test sets' losses during training.

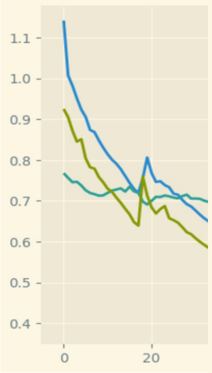


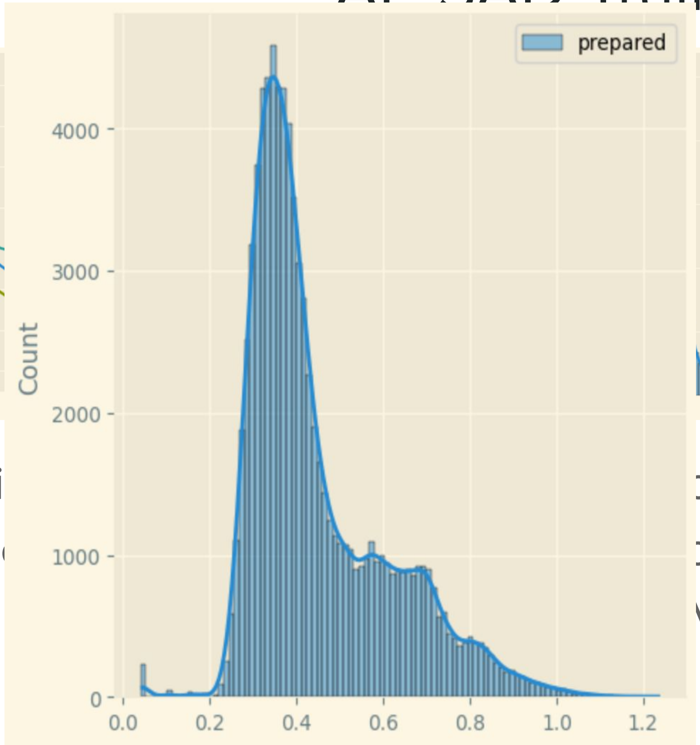
Reconstruction costs in training set for average of features. (Mean: 0.342)

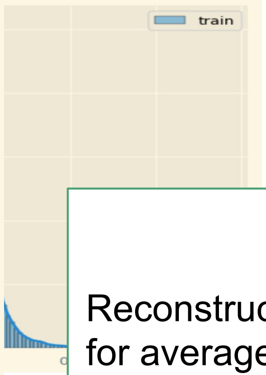


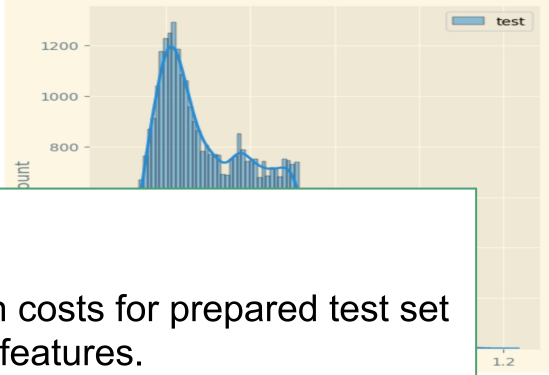
Reconstruction costs in test set for average of features. (Mean: 0.556)

AE-SAD Training Details









Reconstruction costs for prepared test set for average of features.
(Mean: 0.450)

Train, vali... sets' loss... training.

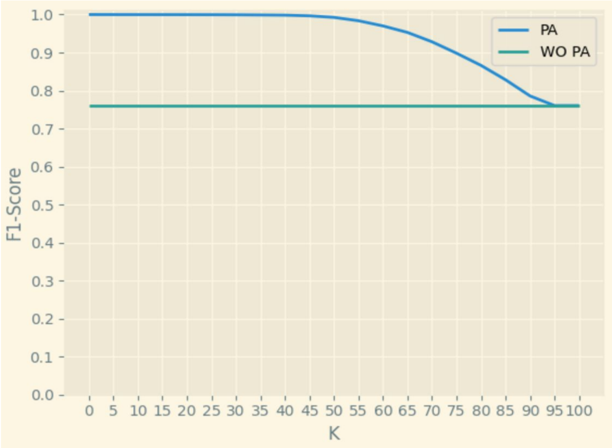
...on c... ...or average ... Mean:

...in ... test set for average of features. (Mean: 0.556)

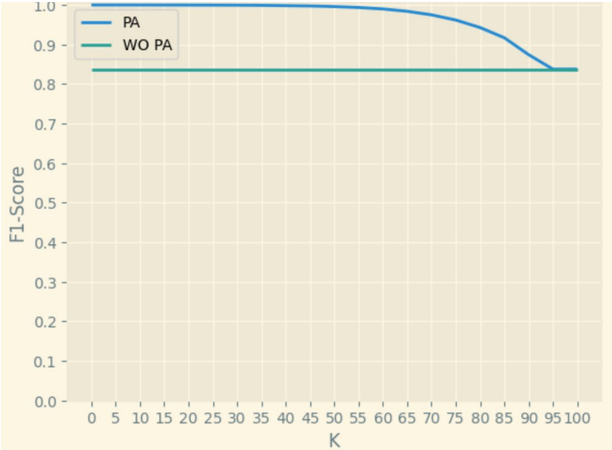
Results of AE-SAD

Best F1-Score

Trained Model



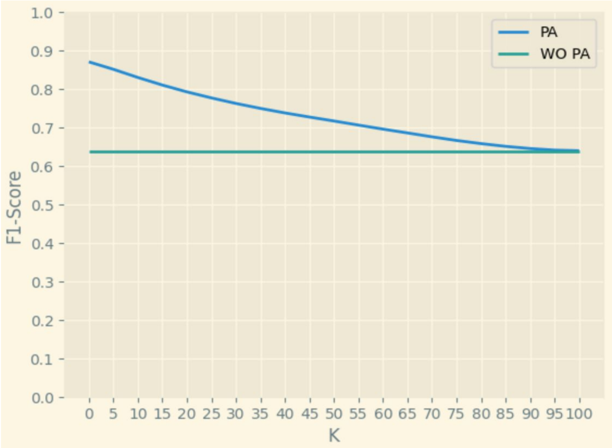
AE-SAD Model



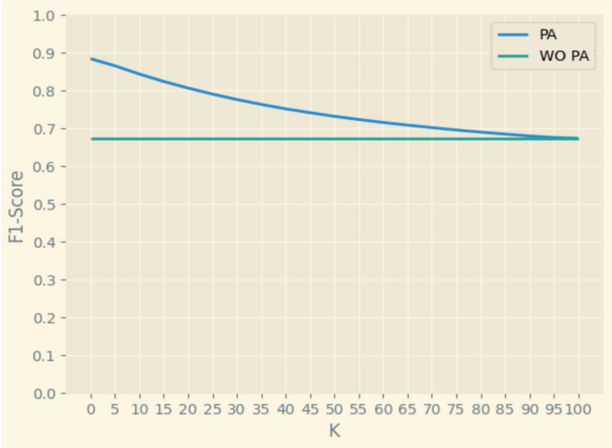
The x-axis represents the K values. The y-axis stands for best F1 scores.

Mean F1-Score

AE Model



AE-SAD Model



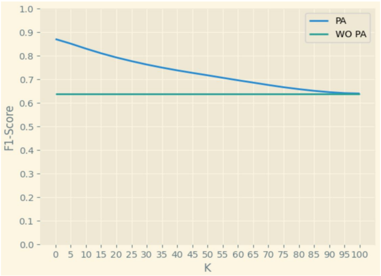
The x-axis represents the K values. The y-axis stands for mean F1 scores.

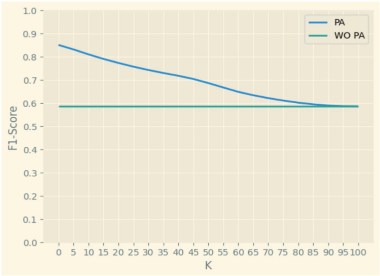
Discussion

Mean vs Best F1-Score for Evaluation

An evaluation method should show the difference of performances clearly. However, best F1-Score with point adjustment (PA) would be bad choice for clarity of difference in many K.

Mean F1-Score





Best F1-Score