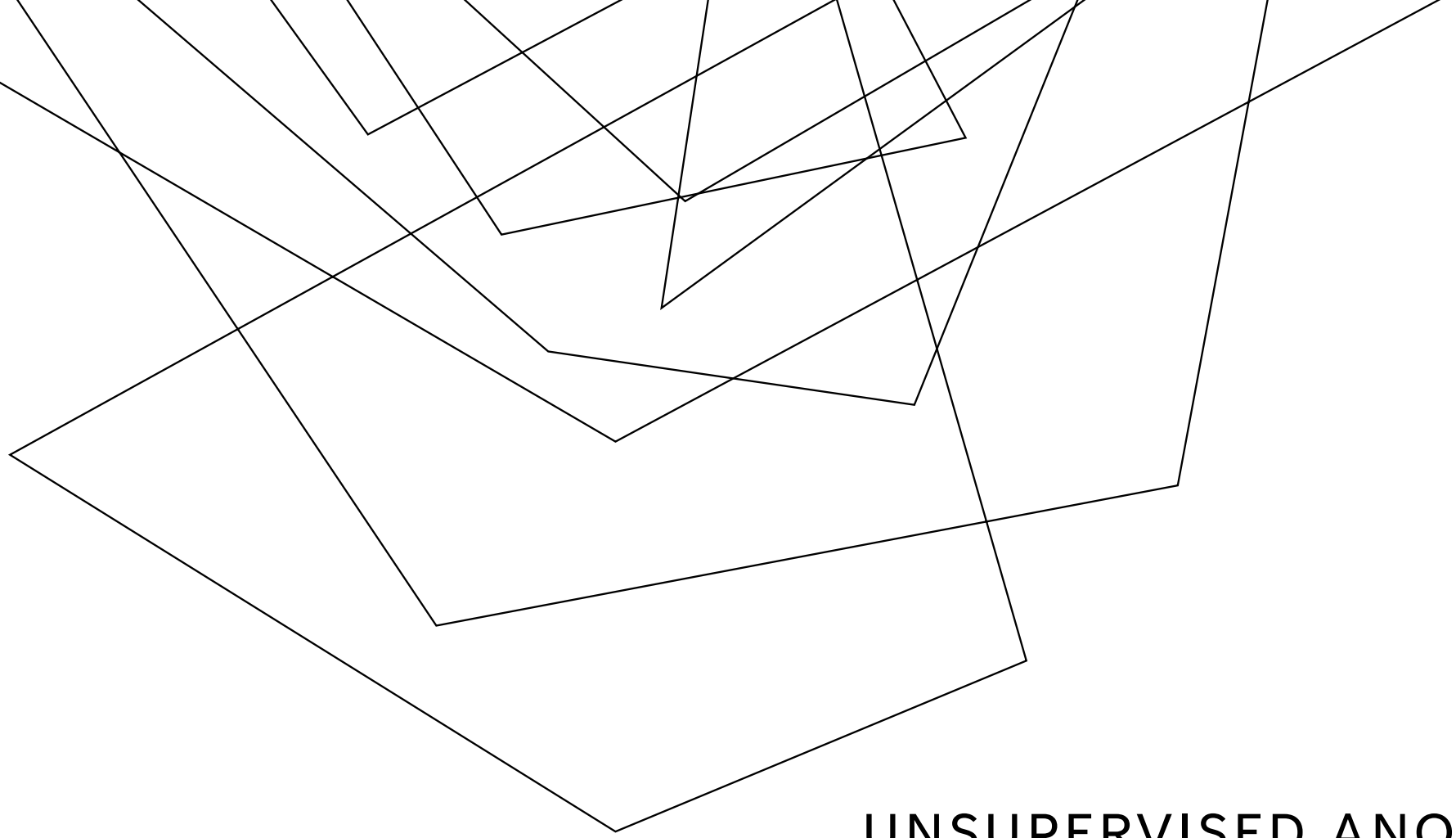
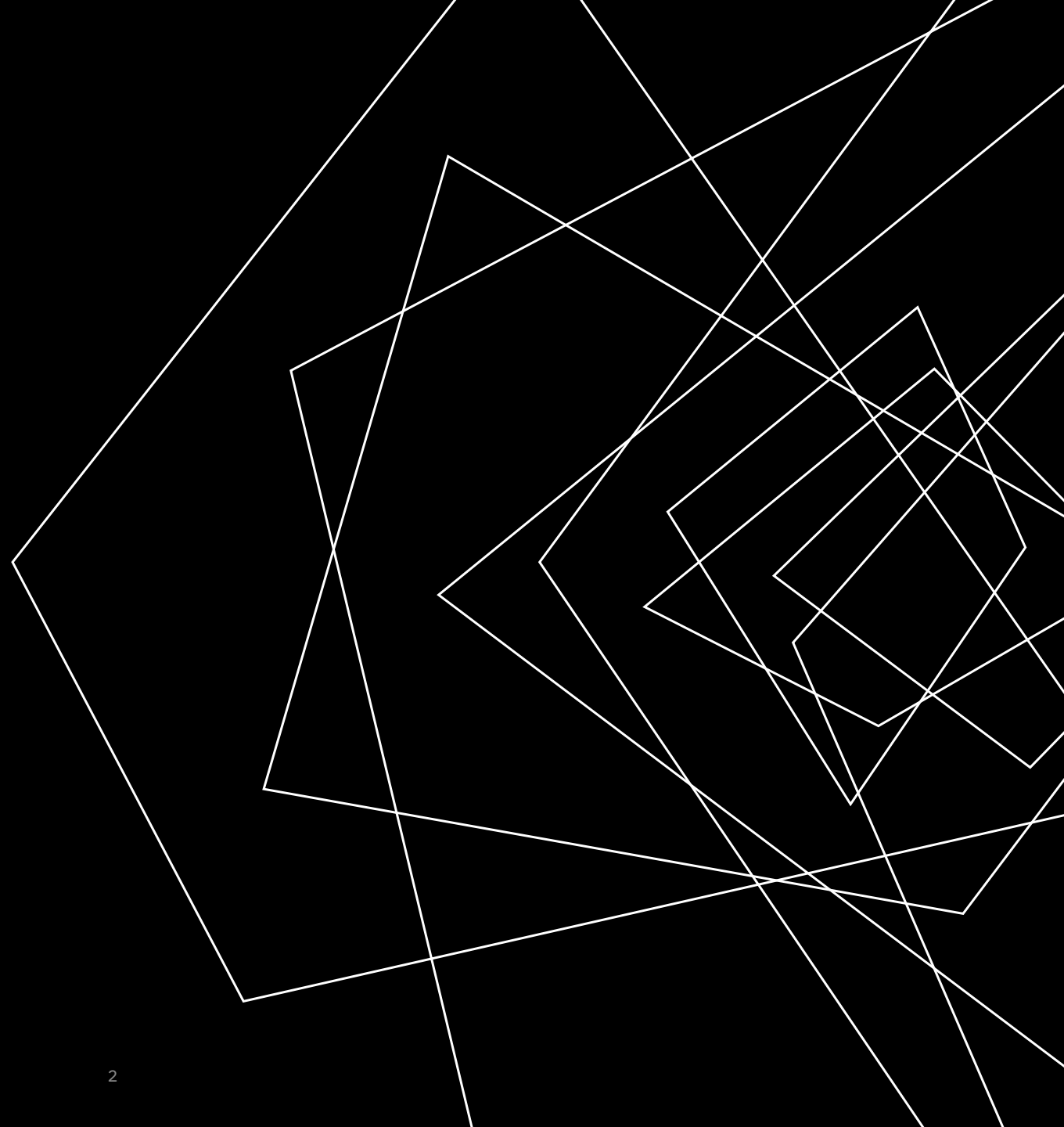


UNSUPERVISED ANOMALY DETECTION IN TIME SERIES DATA

Sadiqa Jafari

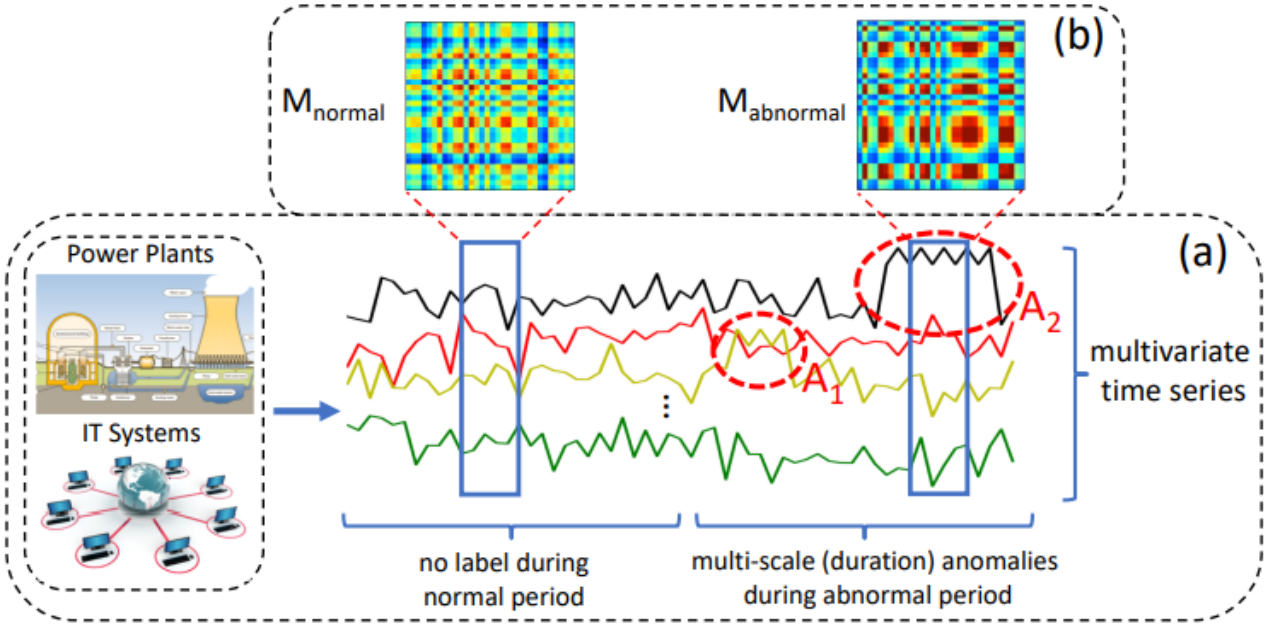
AD20216006

TABLE OF CONTENT



INTRODUCTION

- Complex systems are ubiquitous in modern manufacturing industry and information services. Monitoring the behaviors of these systems generates a substantial amount of multivariate time series data.
- A critical task in managing these systems is to detect anomalies in certain time steps such that the operators can take further actions to resolve underlying issues. For instance, an anomaly score can be produced based on the sensor data and it can be used as an indicator of power plant failure.



ANOMALY DETECTION

- Anomaly detection is a step in data mining that identifies data points, events, and/or observations that deviate from a dataset's normal behavior. Anomalous data can indicate critical incidents, such as a technical glitch, or potential opportunities, for instance a change in consumer behavior.

RELATED WORKS

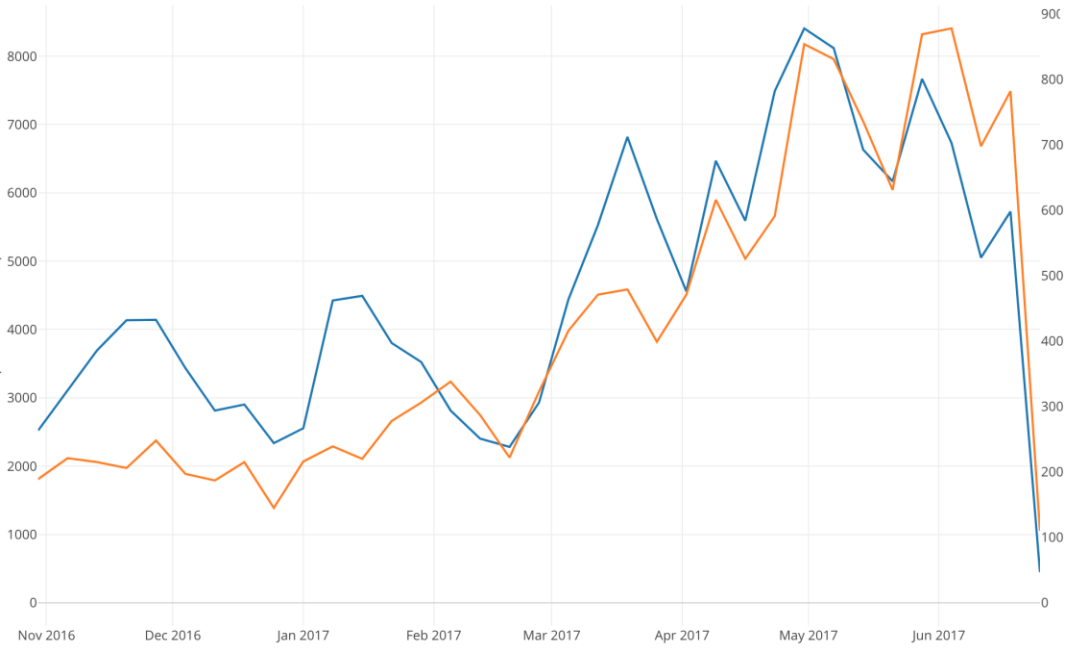
- Unsupervised anomaly detection on multivariate time series data is a challenging task and various types of approaches have been developed in the past few years.

KNN

- One traditional type is the distance methods (Hautamaki, ¨ Ka¨rkka¨ınen, and Franti 2004; Id ¨ e, Papadimitriou, and Vla- ´ chos 2007). For instance, the k-Nearest Neighbor (kNN) algorithm (Hautamaki, Ka ¨ ¨rkka¨ınen, and Franti 2004) com- ¨ putes the anomaly score of each data sample based on the average distance to its k nearest neighbors

deep learning

- Besides traditional methods, deep learning based unsupervised anomaly detection algorithms (Malhotra et al. 2016; Zhai et al. 2016; Zhou and Paffenroth 2017; Zong et al. 2018) have gained a lot attention recently. For instance, Deep Autoencoding Gaussian Mixture Model (DAGMM) (Zong et al. 2018) jointly considers deep auto-encoder and Gaussian mixture model to model density distribution of multi-dimensional data. LSTM encoder-decoder (Malhotra et al. 2016; Qin et al. 2017) models time series temporal dependency by LSTM networks and achieves better generalization capability than traditional methods. Despite their effectiveness, they cannot jointly consider the temporal dependency, noise resistance, and the interpretation of severity of anomalies



TIME SERIES DATA

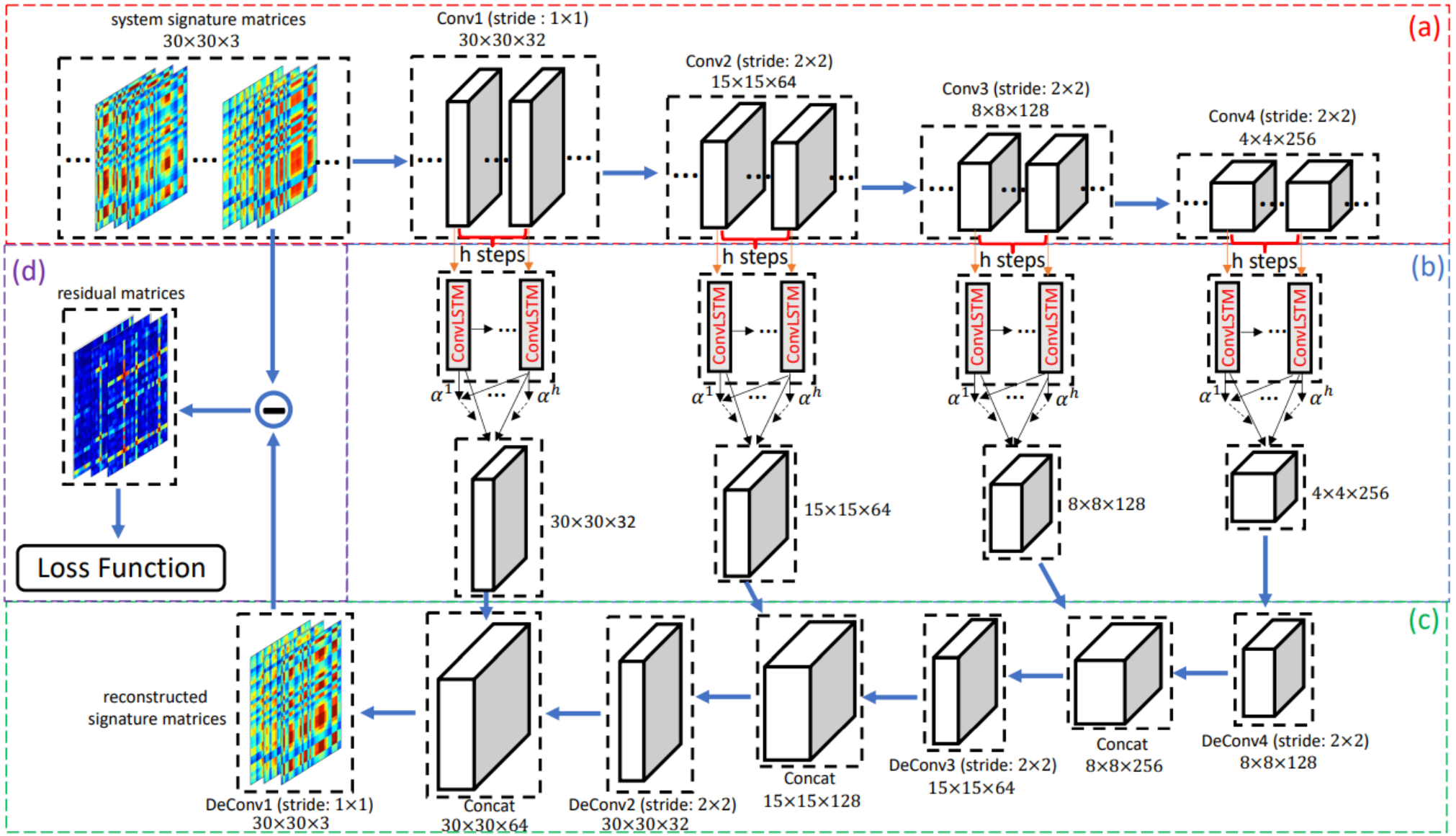
- Time series data, also referred to as time-stamped data, is a sequence of data points indexed in time order. Time-stamped is data collected at different points in time.
- These data points typically consist of successive measurements made from the same source over a time interval and are used to track change over time.

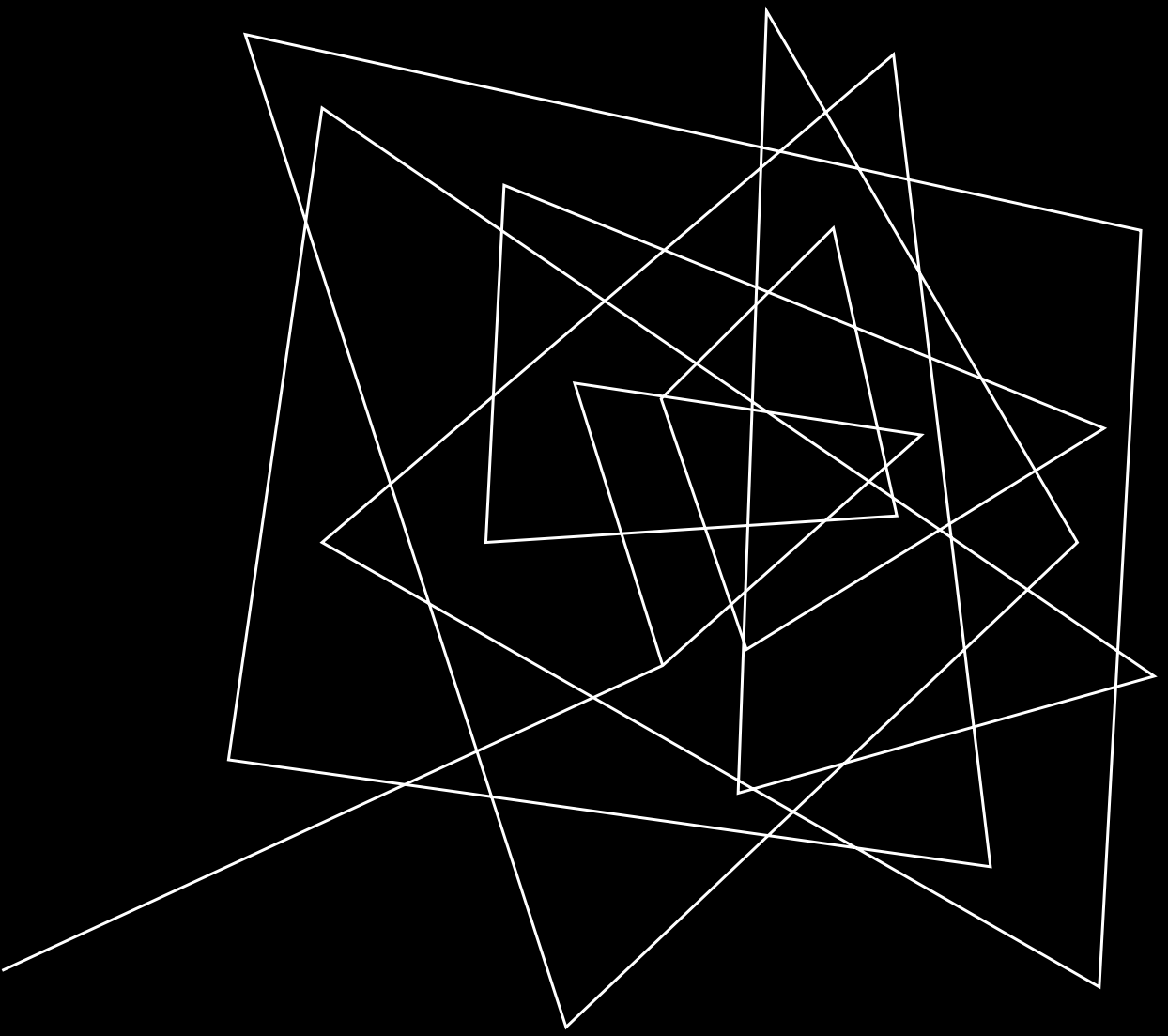
FRAMEWORK

- A Multi-Scale Convolutional Recurrent Encoder-Decoder.

specifically, MSCRED first constructs multi-scale (resolution) signature matrices to characterize multiple levels of the system statuses across different time steps. In particular, different levels of the system statuses are used to indicate the severity of different abnormal incidents. Subsequently, given the signature matrices, a convolutional encoder is employed to encode the inter-sensor (time series) correlations patterns and an attention based Convolutional Long-Short Term Memory (ConvLSTM) network is developed to capture the temporal patterns. Finally, with the feature maps which encode the inter-sensor correlations and temporal information, a convolutional decoder is used to reconstruct the signature matrices and the residual signature matrices are further utilized to detect and diagnose anomalies.

MULTI-SCALE CONVOLUTIONAL RECURRENT ENCODER-DECODER(MSCRED)





EXPERIMENTS

DATASET

- We use a synthetic dataset and a real world power plant dataset for empirical studies. The detailed statistics and settings of these two datasets are shown in this Table.

Statistics	Synthetic	Power Plant
# time series	30	36
# points	20,000	23,040
# anomalies	5	5
# root causes	3	3
train period	$0 \sim 8{,}000$	$0 \sim 10{,}080$
valid period	$8{,}001 \sim 10{,}000$	$10{,}081 \sim 18{,}720$
test period	$10{,}001 \sim 20{,}000$	$18{,}721 \sim 23{,}040$

CNN ENCODER

- cnn encoder: we made a function named tensorflow_variable which use variable function of tensorflow framework and xavier initializer for Tensor variable declaration initialization with our custom parameters, the input data size is 5*30*30*3 then we use cnn_encoder_layer function the output size for first layer is 30*30*32.

```
23  def cnn_encoder(data):
24
25      filter1 = tensor_variable([3, 3, 3, 32], "filter1")
26      strides1 = (1, 1, 1, 1)
27      cnn1_out = cnn_encoder_layer(data, filter1, strides1)
28
29      filter2 = tensor_variable([3, 3, 32, 64], "filter2")
30      strides2 = (1, 2, 2, 1)
31      cnn2_out = cnn_encoder_layer(cnn1_out, filter2, strides2)
32
33      filter3 = tensor_variable([2, 2, 64, 128], "filter3")
34      strides3 = (1, 2, 2, 1)
35      cnn3_out = cnn_encoder_layer(cnn2_out, filter3, strides3)
36
37      filter4 = tensor_variable([2, 2, 128, 256], "filter4")
38      strides4 = (1, 2, 2, 1)
39      cnn4_out = cnn_encoder_layer(cnn3_out, filter4, strides4)
40
41      return cnn1_out, cnn2_out, cnn3_out, cnn4_out
```

CNN DECODER

- cnn decoder: here we used tensorflow_variable function again then we used cnn_decoder_layer function which uses conv2d transpose function then we concatenate lstm output and decoded cnn.

```
80  def cnn_decoder(lstm1_out, lstm2_out, lstm3_out, lstm4_out):
81      d_filter4 = tensor_variable([2, 2, 128, 256], "d_filter4")
82      dec4 = cnn_decoder_layer(lstm4_out, d_filter4, [1, 8, 8, 128], (1, 2, 2, 1))
83      dec4_concat = tf.concat([dec4, lstm3_out], axis=3)
84
85      d_filter3 = tensor_variable([2, 2, 64, 256], "d_filter3")
86      dec3 = cnn_decoder_layer(dec4_concat, d_filter3, [1, 15, 15, 64], (1, 2, 2, 1))
87      dec3_concat = tf.concat([dec3, lstm2_out], axis=3)
88
89      d_filter2 = tensor_variable([3, 3, 32, 128], "d_filter2")
90      dec2 = cnn_decoder_layer(dec3_concat, d_filter2, [1, 30, 30, 32], (1, 2, 2, 1))
91      dec2_concat = tf.concat([dec2, lstm1_out], axis=3)
92
93      d_filter1 = tensor_variable([3, 3, 3, 64], "d_filter1")
94      dec1 = cnn_decoder_layer(dec2_concat, d_filter1, [1, 30, 30, 3], (1, 1, 1, 1))
95
96      return dec1
```

CNN LSTM

- cnn lstm: here we made convlstm celland calculate attention based on inner-product between feature representation of last step and other steps.

```python
def cnn_lstm_attention_layer(input_data, layer_number):

    convlstm_layer = tf.contrib.rnn.ConvLSTMCell(
        conv_ndims=2,
        input_shape=[input_data.shape[2], input_data.shape[3], input_data.shape[4]],
        output_channels=input_data.shape[-1],
        kernel_shape=[2, 2],
        use_bias=True,
        skip_connection=False,
        forget_bias=1.0,
        initializers=None,
        name="conv_lstm_cell" + str(layer_number))
    outputs, state = tf.nn.dynamic_rnn(convlstm_layer, input_data, dtype=input_data.dtype)
    attention_w = []
    for k in range(util.step_max):
        attention_w.append(tf.reduce_sum(tf.multiply(outputs[0][k], outputs[0][-1])) / util.step_max)
    attention_w = tf.reshape(tf.nn.softmax(tf.stack(attention_w)), [1, util.step_max])

    outputs = tf.reshape(outputs[0], [util.step_max, -1])
    outputs = tf.matmul(attention_w, outputs)
    outputs = tf.reshape(outputs, [1, input_data.shape[2], input_data.shape[3], input_data.shape[4]])

    return outputs, attention_w
```

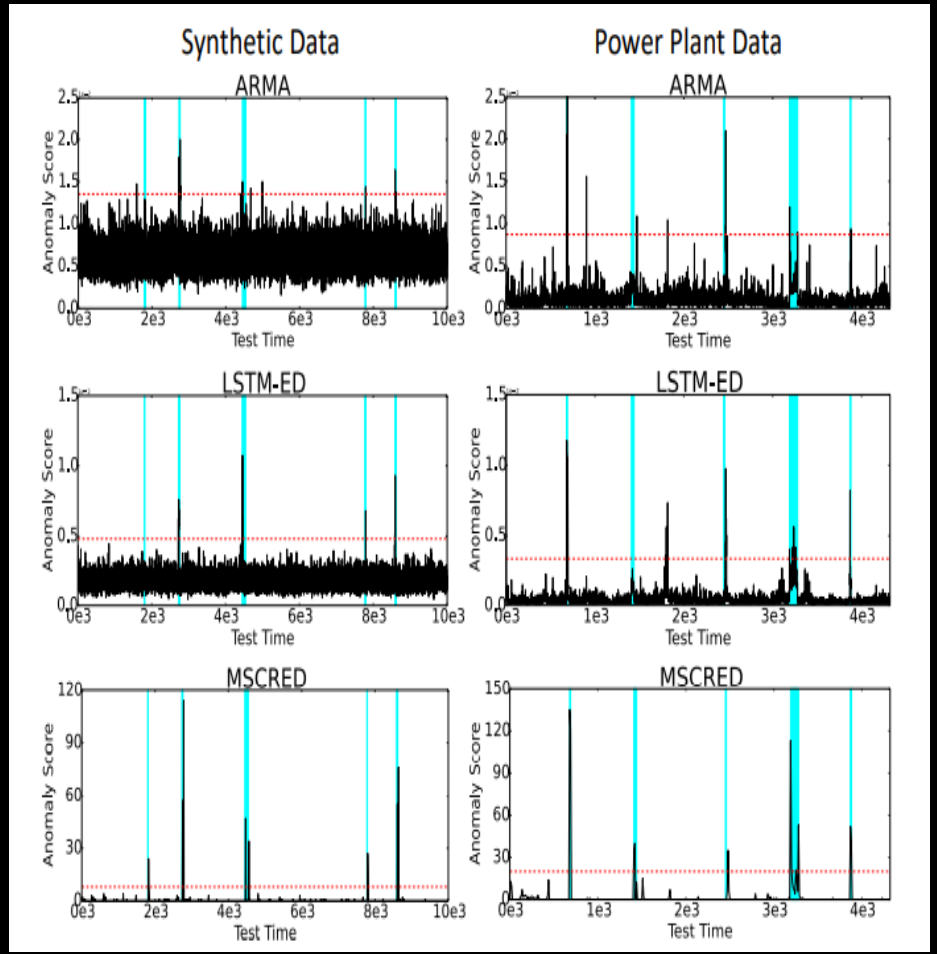
RESULTS

- The performance of different methods for anomaly detection are reported in Table , where the best scores are highlighted in bold-face and the best baseline scores are indicated by underline. The last row reports the improvement (%) of Multi-Scale convolutional Recurrent Encoder-Decoder (MSCRED) over the best baseline method.

Method	Synthetic Data			Power Plant Data		
	Pre	Rec	F_1	Pre	Rec	F_1
OC-SVM	0.14	0.44	0.22	0.11	0.28	0.16
DAGMM	0.33	0.20	0.25	0.26	0.20	0.23
HA	0.71	0.52	0.60	0.48	0.52	0.50
ARMA	0.91	0.52	0.66	0.58	0.60	0.59
LSTM-ED	<u>1.00</u>	<u>0.56</u>	<u>0.72</u>	<u>0.75</u>	<u>0.68</u>	<u>0.71</u>
$\text{CNN}^{ED(4)}_{ConvLSTM}$	0.37	0.24	0.29	0.67	0.56	0.61
$\text{CNN}^{ED(3,4)}_{ConvLSTM}$	0.63	0.56	0.59	0.80	0.72	0.76
$\text{CNN}^{ED}_{ConvLSTM}$	0.80	0.76	0.78	0.85	0.72	0.78
MSCRED	**1.00**	**0.80**	**0.89**	**0.85**	**0.80**	**0.82**
Gain (%)	–	30.0	23.8	13.3	19.4	15.5

CONCLUSION

- Case study of anomaly detection. The shaded regions represent anomaly periods. The red dash line is the cutting threshold of anomaly.
- Here the figure shows provides case study of MSCRED and two best baseline methods ARMA and LSTM-ED, for both datasets.
- We can observe that the anomaly score of ARMA is not stable and the results contain many false positives and false negatives.
- The anomaly score of LSTM-ED is smoother than ARMA while still contains several false positives and false negatives.
- MSCRED can detect all anomalies without any false positive and false negative.



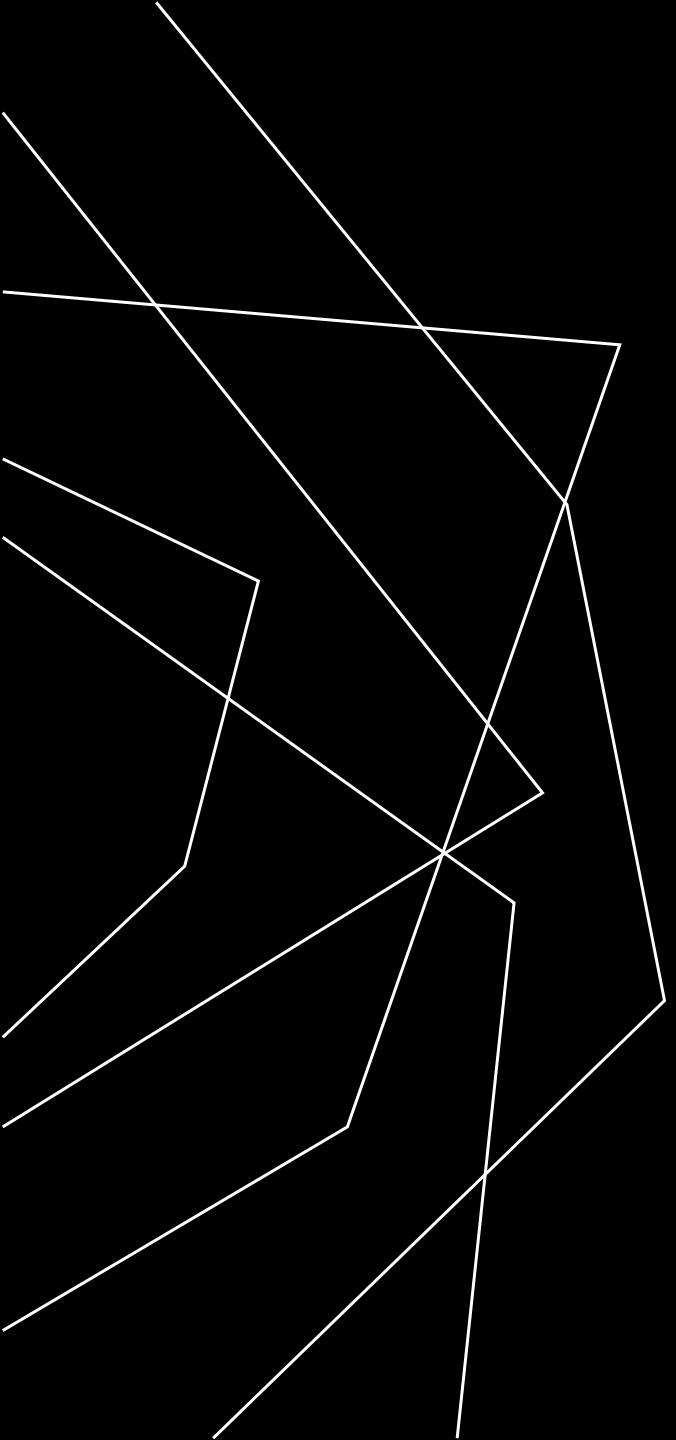
REFERENCES

V. Hautamaki, I. Karkkainen, P. Franti Published 2004

Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.

[Zhai et al. 2016] Zhai, S.; Cheng, Y.; Lu, W.; and Zhang, Z. 2016. Deep structured energy based models for anomaly detection. In ICML, 1100–1109.

[Zong et al. 2018] Zong, B.; Song, Q.; Min, M. R.; Cheng, W.; Lumezanu, C.; Cho, D.; and Chen, H. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In ICLR



THANK YOU

Sadiqa Jafari