Time Series Processing, Anomaly Detection and Explanation

Neena Sharon Betti Mol [NS]*
Technical University Dortmund, Germany

Rahul Poovassery [RP]†
Technical University Dortmund, Germany

Akash Chandra Baidya [AB]‡
Technical University Dortmund, Germany

Mridul Varghese Koshy [MK]§
Technical University Dortmund, Germany

Nidhi Patel [NP]¶
Technical University Dortmund, Germany

Ashish Saini [AS]‖
Technical University Dortmund, Germany

Kübra Turum [KT]**
Technical University Dortmund, Germany

Supritha Palguna [SP]††
Technical University Dortmund, Germany

Akanksha Tanwar [AT]‡‡
Technical University Dortmund, Germany

ABSTRACT

[MK]Energy consumption data has become increasingly important in recent years as governments and businesses look for ways to reduce the carbon footprint of their operations. Energy consumption data recorded from different sensors over time is an important indicator of resource efficiency and can provide insights into how energy use varies over time. Analyzing time series data with different machine learning models can help identify anomalous patterns in energy use and can be used to develop strategies for reducing energy consumption. It is difficult to find anomalous patterns in time series data, so we propose three different machine learning models to help. The first two models employ the encoder-decoder anomaly detection method using LSTM (long short-term memory) and TCN (temporal convolutional network). Both anomaly detection models use reconstruction error to identify anomalies after learning to reconstruct "normal" time-series behavior. The third model is based on a generative model called DGHL (Deep Generative Model with Hierarchical Latent Factors). This is a new approach that uses MCMC (Markov chain Monte Carlo), posterior sampling, and alternating back-propagation for detecting anomalies in time series data, as compared to previous approaches focused on simple variational autoencoders and generative adversarial networks. All three models are discussed, analyzed, and compared. Auto-encoder models are trained with different parameters, and the results are analyzed using different scoring functions. A comparison of the latent space representation of both auto-encoder models is also included. Data from ten different buildings at the Technical University Dortmund is used in these experiments. Point anomalies and contextual anomalies were injected into some building data to check the performance of all three models.

Additional Key Words and Phrases: anomaly score, autoencoder, convolutional network, DGHL, LSTM, LSTM AE, neural networks, synthetic data, TCN, time series data

1 INTRODUCTION

[RP] In recent years, many industries have started collecting time-series data to monitor and analyse trends and derive relevant statistics from the data over time. For instance, stock prices are collected and analyzed over time in finance to identify trends and make predictions about future market behaviour. In order to recognize patterns and trends and improve energy efficiency, statistics on energy production and consumption are gathered over time in the energy sector. Furthermore, in healthcare, patient data is collected to monitor health outcomes and track the effectiveness of several treatments. When it comes to the manufacturing sector, information is gathered over time about the performance of production methods and equipment to pinpoint areas for improvement and maximize operational effectiveness. Thus, it is a well-known fact that time-series data collection is a critical component for businesses and industries to monitor and analyse trends, identify opportunities for improvement, and make data-driven decisions.

Time series is prone to outliers and these are data points that deviate significantly from the normal pattern or trend of the data. These may be due to various factors, such as measurement errors, data entry errors, equipment malfunctions, or unusual events. Analysis of these outlier points is of extreme importance in time-series data as they can skew the statistical analysis which is interpreted and thus result in wrong conclusions being made. Statistical techniques like normalization can be used to deal with outliers, but the popular and much more reliable option is to use anomaly detection algorithms to identify outliers. The existence of an outlier may not necessarily mean that the data is unclean but rather these outliers in most cases can shed some light on some changes made in the processes used to measure the data and the evaluation of the outliers can help to understand if the outlier is a valid data point or not.

[NS] In this report, we have used 3 anomaly detection models to process time-series data collected from 10 different buildings which encompass the campus of the Technical University of Dortmund. The first model discussed is a Long Short-Term Memory Autoencoder network, which is based on an Encoder-Decoder architecture.

*All authors contributed equally to this research.
†All authors contributed equally to this research.
‡All authors contributed equally to this research.
§All authors contributed equally to this research.
¶All authors contributed equally to this research.
‖All authors contributed equally to this research.
**All authors contributed equally to this research.
††All authors contributed equally to this research.
‡‡All authors contributed equally to this research.

Autoencoders are neural networks that can be trained to reconstruct their input data and are commonly used for dimensionality reduction and anomaly detection. The architecture of this model is based on the one proposed in Malhotra et al. (2016). Here, an encoder uses a vector representation of the input which is then passed to the decoder to achieve the output as a reconstructed time series. The model is trained to minimize this reconstruction error between the input time-series data and the reconstructed data using a loss function, and based on this error, an anomaly score can be computed, which is compared against a predefined limit to detect an anomaly in the data. In the second model, a Temporal Convolutional Autoencoder is used to detect the anomalies (Thill et al. (2021)). This model again follows the same idea as the LSTM AE, and an encoder and decoder layer exist in the network to reconstruct the input time series given. The TCN-AE comprises of dilated convolutional layers, which create a wide receptive field to reconstruct and process the time series input data. As in the LSTM AE, the anomaly is detected using a reconstruction error and the anomaly score. The final model proposed is the Deep Generative Model with Hierarchical Latent Factors (DGHL) (Challu et al. (2022)), a top-down Convolution Network (ConvNet). The model works by learning a hierarchical representation of the time-series data, where each level of the hierarchy represents a different level of abstraction. Along with this, the model also includes a generative component that can be used to generate new samples of the time-series data. Anomalies are again identified as data points that have a high reconstruction error, thereby signifying that they deviate from the expected pattern.

For the LSTM AE and the TCN AE, several experiments were carried out by manipulating the architecture and the anomaly scoring function, and their results were compared to find if there was a significant increase in the performance of the model. The models were run on the building datasets as well as a two-dimensional sinusoidal curve and some synthetic data. A latent space visualization was also done using these models to discover clusters. For the DGHL model, we have examined the model's ability to correctly identify the anomalies for a few variations of the input data : the original time-series data, hourly recorded data, synthesized data, and finally, a dataset that combines similar buildings.

[NP] Overall, we noticed that the performances of all the models under a controlled environment (on dummy data with known true labels) are much better than those on the real-time sensor data for various buildings. We assume the reason behind it is two-fold. First, poor data quality as given datasets contains too many missing values which in terms adversely affects the training of the model and consequently also affects the performance of the model on testing data. And the second reason could be a lack of proper labeling technique since the true labels for the building data are unknown, we assign the labels based on the assumption that public holidays are anomalies which might not be that accurate. Some solutions to overcome these shortcomings are also suggested in further sections.

[NP] The report has been divided into 5 main sections. In Section 2, the related work applicable to the models proposed in the paper has been discussed. In addition, the objective of the project is explained in detail. Section 3 explains the architecture of the models used for the experiments in further detail and also talks about the evaluation matrices and statistical methods used in the report. In Section 4, we talk about the dataset used, and its preparation to be fed into the models. This is followed by the experiments that are carried out along with the results observed. The concluding Section 5 summarizes the results and interpretations and discusses briefly any further possible analysis.

2 RELATED WORK

[AB] By using the prediction error or a function of the prediction error as a measure of the severity of the anomaly, time-series prediction models have been demonstrated to be useful for detection of anomaly. Several time series prediction models have been proposed in recent years for anomaly detection. However, these models have limited capacity to handle complex and non-linear patterns in the data.

Wulsin et al. (2010) proposed the use of Deep Belief Nets (DBNs) for anomaly detection. DBNs are generative models that can learn a compact and distributed representation of the data. This approach can also be used for time-series data, but it requires pre-processing the data into a fixed-length feature vector. Moreover, the training of DBNs is computationally expensive.

Sakurada and Yairi (2014) proposed a denoising autoencoder approach for anomaly detection in non-temporal data. This approach can also be used for time-series data by treating each time-step as a separate feature. However, it has limitations in handling complex and non-linear patterns in the data.

Malhotra et al. (2015) proposed the use of stacked LSTM-based architecture for time series anomaly detection. In this paper, the authors propose an anomaly detection method that uses an LSTM-based architecture to process time-series data.

One key advantage of using an LSTM-based Encoder-Decoder architecture for anomaly detection is its ability to model long-term dependencies in the input sequence. The LSTM units contain gating mechanisms that control the flow of information through the network, allowing it to selectively remember or forget information from previous time steps. This makes the architecture well-suited for processing sequences that exhibit complex temporal dynamics, which is often the case in multi-sensor time-series data. The authors also propose a training strategy that incorporates outlier exposure, which involves adding anomalous instances to the training set in order to improve the robustness of the model to unseen anomalies. This is achieved by augmenting the training set with anomalous instances generated by corrupting the original data with random noise or outliers.

The effectiveness of the suggested method is assessed against a number of cutting-edge anomaly detection techniques using a variety of real-world datasets, such as the ECG, space shuttle, power demand, and multi-sensor engine dataset. The findings demonstrate that, in terms of both detection accuracy and robustness to hidden abnormalities, the LSTM-based Encoder-Decoder architecture with outlier exposure surpasses the other techniques, such as RNN.

Choi et al. (2018) provided a comprehensive review and analysis of the application of deep learning methods for anomaly detection in time-series data. The authors first discussed the characteristics of time-series data and the challenges of anomaly detection in such

data. They then provided an overview of deep learning methods, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders, and their applications in time-series anomaly detection.

The authors also presented a detailed analysis of various factors that can affect the performance of deep learning methods for anomaly detection in time-series data, such as the choice of network architecture, training data, and hyperparameters. The authors provide guidelines for selecting appropriate deep learning methods and optimizing their performance for time-series anomaly detection. Experiments on various datasets to demonstrate the effectiveness of deep learning methods for anomaly detection in time-series data were included in that study. The results showed that deep learning methods, especially RNN-based models such as LSTMs, outperform traditional methods for time-series anomaly detection.

The LSTM-based Encoder-Decoder architecture can be a powerful tool for multi-sensor anomaly detection that is capable of modeling long-term dependencies in the input sequence. The ability of the architecture to selectively remember or forget information from previous time steps makes it well-suited for processing complex temporal dynamics, and its use of outlier exposure can improve the robustness of the model to unseen anomalies.

[MK]Thill et al. (2021) proposed an unsupervised anomaly detection algorithm based on dilated convolutional autoencoder for learning temporal patterns in time series data. The algorithm outperforms several state-of-the-art algorithms on a real-world electrocardiogram (ECG) dataset of patients with cardiac arrhythmia. The TCN-AE model used in this study is suitable for capturing complex quasiperiodic temporal patterns over a long period of time.

Oord et al. (2016) proposed a deep neural network, WaveNet, that generates raw audio waveforms in a fully probabilistic and autoregressive manner. It can efficiently train on audio data with tens of thousands of samples per second. It outperforms parametric and concatenative systems in text-to-speech tasks for English and Mandarin. It can also model multiple speakers equally well and generate realistic musical fragments.

Bai et al.(2018) proposed a Temporal Convolutional Network (TCN) for sequence modeling. The TCN ouputs were compared with the recurrent networks such as LSTMs. The TCN result outperforms the recurrent networks while having faster computational time. He and Zhao (2019) presented a novel approach for anomaly detection in time series using TCN. The TCN is trained on normal sequences and used to predict trends in a number of time steps, and prediction errors are used to calculate anomaly scores.

Reconstruction-based models. [AT] We have different models for anomaly detection. Reconstruction-based models are used for the same purpose where we calculate the reconstruction error and use them as anomaly scores. The LSTM-VAE proposed in the (Park et al. (2018)) model is implemented on the same concept. The local information of a short window is condensed into the low-dimensional embedding using a VAE model. To manage the sequential patterns over a longer period of time, we use an LSTM model that operates on the low dimensional embeddings generated by the VAE model. We can identify anomalies occurring over both short and long time periods because of the hierarchical structure. In order to detect anomalies, Generative Adversarial Networks (GANs) are also used. The Multimodal Anomaly Detection with GAN (MAD-GAN) methodology presented in (Li et al. (2019)) takes into account the complete variable set concurrently to capture the latent interactions among the variables. The generator and discriminator generated by the GAN find anomalies using discrimination and reconstruction. MTAD-GAT (Multivariate Time-series Anomaly Detection via Graph Attention Network) is another model presented in (Zhao et al. (2020)), which is for better representations of time-series data, MTAD-GAT concurrently trains a forecasting-based model and a reconstruction-based model. A joint objective function can simultaneously optimize the two models. By adding a joint optimization objective, we combine the benefits of forecasting-based and reconstruction-based models. The reconstruction-based model develops a latent representation of the full-time series, whereas the forecasting-based model concentrates on single-timestamp prediction. We also have models that find anomalies based on embeddings and latent representation. A framework for anomaly identification on time series called Neural Contextual Anomaly Detection presented in (Carmona et al. (2021)), is based on splitting each time series into overlapping and fixed-size windows. Each window is further separated into two sections: a context window and a suspect window, and both portions are mapped into neural representations(embeddings) by Temporal Convolutional Networks (TCNs). The goal of the model is to find anomalies in the suspect window.

Generative models with Alternating Back-Propagation. The DGHL model uses the Alternating back-propagation algorithm presented in (Han et al. (2017)) and the short-MCMC algorithm proposed in (Nijkamp et al. (2020)) to train the DGHL model. The detail of the Alternating Back-Propagation algorithm is given in section 3.4.4.

3 METHODOLOGY

3.1 LSTM AutoEncoder

3.1.1 ***LSTM Overview.*** [AB]Long Short-Term Memory (LSTM), a variety of Recurrent Neural Network (RNN), are frequently employed in sequential data analysis for tasks including speech recognition, natural language processing, and time-series prediction. The main advantage of LSTM is that it can learn long-term dependencies in sequential data, whereas regular RNNs cannot do so because of the issue of vanishing gradients. In next sections, we will discuss the general structure of RNN, an LSTM model and its components in detail.

Recurrent Neural Networks. Before discussing LSTM, it is important to understand recurrent neural networks (RNNs), which are the foundation on which LSTMs are built. In an RNN, the output of a previous time step is fed back into the model as input for the current time step. This creates a feedback loop, allowing the model to consider the previous inputs and the current input when generating an output. An input series of x values is mapped by a

recurrent neural network to an output sequence of o values. A loss 'L' calculates the discrepancy between the actual and expected outputs. The RNN additionally has connections from hidden input to hidden output, hidden to hidden connections, and hidden to hidden connections, all of which are parametrized by weight matrices U, W, and V, respectively. Then, we apply the following equation from time step t = 1 through time step t = n (Goodfellow et al.(2016)):

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)},$$
$$h^{(t)} = \tanh\left(a^{(t)}\right),$$
$$o^{(t)} = c + Vh^{(t)},$$
$$\hat{y}^{(t)} = \text{softmax}\left(o^{(t)}\right),$$

However, RNNs suffer from the vanishing gradient problem, which makes it difficult for them to retain information from previous time steps for long periods of time. This is because the gradients used in backpropagation are multiplied in each time step, causing them to decrease exponentially as they propagate backward through the network. This can make it difficult for the model to learn long-term dependencies in the data. Recurrent neural networks are designed

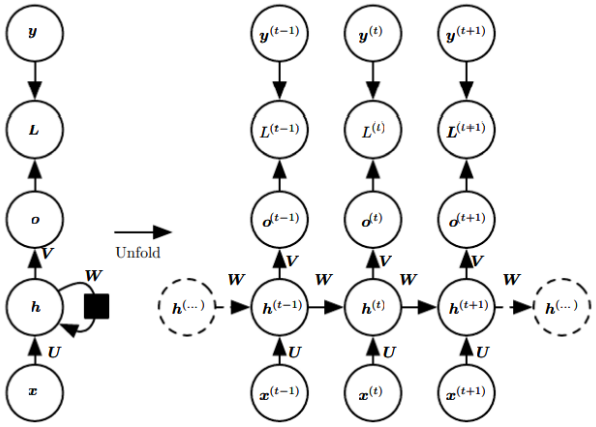


Figure 1: The architecture of a recurrent neural network is depicted here. The network consists of several components: the input (x), the hidden state (h), the output (o), the loss (L), and the target value (y). The input is processed by the hidden state, which generates an output. The output is compared to the target value using the loss function. The goal of training the network is to minimize the loss function by adjusting the weights and biases of the network. (Goodfellow et al.(2016))

to process sequences of inputs by processing each input in the sequence and maintaining a state that is updated with each new input.

LSTM. An LSTM model has four main components: the input layer, the output layer, the forget layer, and the memory cell. Each component has its own set of weights, which are learned during the training process.

Input Layer. The input layer is responsible for receiving the input sequence and transforming it into a format that the LSTM cells can process. It contains one node for each input in the sequence, and each node is connected to the input gate of each LSTM cell. The input layer is typically followed by a sequence of LSTM cells, which process the input and update the hidden state of the network.

Output Layer. The output layer is responsible for generating the output sequence based on the hidden state of the LSTM cells. It contains one node for each output in the sequence, and each node is connected to the output gate of each LSTM cell. The output layer is typically followed by a softmax activation function, which converts the output into a probability distribution.

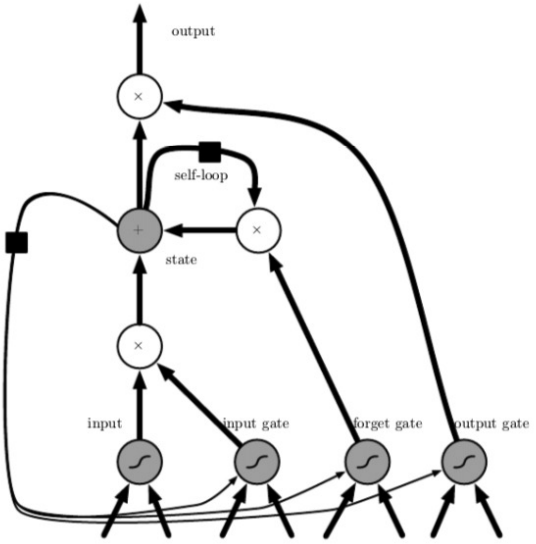


Figure 2: A feed-forward neural network uses an LSTM cell in place of a typical hidden unit (i.e., neuron). Via a sigmoidal activation function, the input, forget, and output gating units allow the cell to accumulate or shut off, respectively, the current input, long-term dependence, and output. Here, the square denotes a one-time-step delay, while the operation symbols in the circles denote the operation using the outputs of the gates. (Goodfellow et al.(2016))

Forget Layer. The forget layer is responsible for controlling the flow of information through the LSTM cells by determining which information to keep and which to discard. It contains one node for each LSTM cell, and each node is connected to the forget gate of the corresponding cell. The forget layer is typically followed by a sigmoid activation function, which determines the strength of the flow of information.

Memory Cell. The memory cell is responsible for storing the long-term memory of the LSTM model. It contains one node for each LSTM cell, and each node is connected to the input and forget gates of the corresponding cell. The memory cell is typically followed by a hyperbolic tangent activation function, which controls the strength

of the stored memory. A specific formulation of the process of LSTM is given below (Goodfellow et al.(2016))-

$$c^{(0)} = 0$$
$$h^{(0)} = 0$$
$$z^{(t)} = g\left(W_z x^{(t)} + U_z h^{(t-1)} + b_z\right) \text{ transformed input}$$
$$i^{(t)} = \sigma\left(W_i x^{(t)} + U_i h^{(t-1)} + b_i\right) \text{ input gate}$$
$$f^{(t)} = \sigma\left(W_f x^{(t)} + U_f h^{(t-1)} + b_f\right) \text{ forget gate}$$
$$c^{(t)} = z^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)} \text{ cell}$$
$$q^{(t)} = \sigma\left(W_q x^{(t)} + U_q h^{(t-1)} + b_q\right) \text{ output gate}$$
$$h^{(t)} = g\left(c^{(t)}\right) \odot q^{(t)} \text{ output}$$

where 'c' denotes the cell that stores the LSTM unit state, 'z' the transformed value of input, 'i 'the value of input gate, 'f' the value of forget gate, 'q' the value of output gate, and 'h' the value of LSTM unit output. Each of the gates has its own set of parameters, marked with the appropriate index.

The network can only learn long-term dependencies in sequential input with the help of the LSTM's gates. The LSTM may learn to identify patterns and make predictions in sequential data by selectively letting new input data into the cell state, retaining or forgetting the previous cell state, and exposing the right amount of cell state to the next layer of the network.

3.1.2 ***LSTM AutoEncoder Architecture.*** [RP] The LSTM encoder-decoder model aims to take as input a time series,

$X = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(L)}\}$ of length L and using a layer of encoders and decoders, the model is trained to reconstruct instances of the time-series. For a given time-series X, the encoder, with c number of units in its hidden layer, reads a fixed length vector representation of the input time-series, X. $\mathbf{h}_E^{(i)}$ is the hidden state of the encoder at time t_i for each $i \in \{1, 2, \ldots, L\}$. To produce the hidden state $\mathbf{h}_E^{(i)}$ of the encoder at t_i, the value $\mathbf{x}^{(i)}$ and the hidden state $\mathbf{h}_E^{(i-1)}$ are used. (Malhotra et al. (2016))

The output from the encoder is then used by the decoder along with the current hidden state and value from the previous time-step to reconstruct the time-series. The last state of the encoder $\mathbf{h}_E^{(L)}$ is fed to the first state of the decoder and the time-series is reconstructed in reverse order $\left\{\mathbf{x}^{(L)}, \mathbf{x}^{(L-1)}, \ldots, \mathbf{x}^{(1)}\right\}$ by jointly training the encoder and the decoder. The decoder takes $\mathbf{x}^{(i)}$ to produce the hidden state $\mathbf{h}_D^{(i-1)}$ and relating to target $\mathbf{x}^{(i-1)}$ predicts $\mathbf{x}'^{(i-1)}$. At each t_i, the predicted $\mathbf{x}'^{(i)}$ is given to the decoder to receive the hidden state $\mathbf{h}_D^{(i-1)}$ and predict $\mathbf{x}'^{(i-1)}$. The goal is then to train the model to minimize $\sum_{X \in s_N} \sum_{i=1}^{L} \left\|\mathbf{x}^{(i)} - \mathbf{x}'^{(i)}\right\|^2$, where s_N is the set of normal training sequences. A linear layer is also applied over the decoder layer to predict the target. To calculate $x'^{(i)}$, a linear layer with bias vector $b \in R^m$ and weight matrix w $(c \times m)$ can be used, $x'^{(i)} = \mathbf{w}^{\mathbf{T}} \mathbf{h}_D^{(3)} + \mathbf{b}$. (Malhotra et al. (2016))

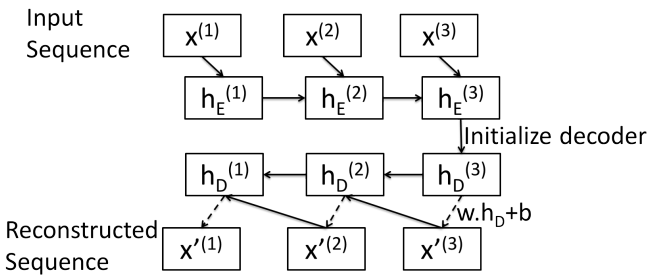


Figure 3: (Malhotra et al. (2016)) LSTM Autoencoder architecture for an input sequence $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}\right\}$ to predict $\left\{\mathbf{x}'^{(1)}, \mathbf{x}'^{2)}, \mathbf{x}'^{(3)}\right\}$

[NS] Based on the input sequence and the reconstructed output sequence, for any time stamp t_i, a reconstruction error vector $\mathbf{e}^{(i)}$ can be given by $\mathbf{e}^{(i)} = \left|\mathbf{x}^{(i)} - \mathbf{x}'^{((i)}\right|$. This error vector is fitted to a Normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ to estimate the parameters $\boldsymbol{\mu}$ and Σ by using Maximum Likelihood estimation, where $\boldsymbol{\mu}$ is the mean and Σ is the covariance matrix. Using these parameters, the anomaly score is found for a point $\mathbf{x}^{(i)}$ as $a^{(i)} = \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)^T \Sigma^{-1} \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)$. (Malhotra et al. (2016))

An anomaly can be seen as a point or any data which is deviating from the majority of the data. To identify an anomalous point, a threshold τ needs to be set and any point $a^{(i)}$ which is above the threshold value can be considered to be anomalous. This means that if $a^{(i)} >= \tau$, then it is "anomalous" and if $a^{(i)} < \tau$, then it is considered to be a normal point. If any window of the time series data contains a point which is considered to be anomalous, then the entire window is regarded as anomalous. A threshold τ is learned across the likelihood values to maximize $F_\beta = \left(1 + \beta^2\right) \times P \times R / \left(\beta^2 P + R\right)$, where $\boldsymbol{P}$ denotes precision and $\boldsymbol{R}$ means recall, "anomalous" denotes the positive class, and "normal" is the negative class, when there are sufficient anomalous sequences. (Malhotra et al. (2016))

The architecture of the previously discussed LSTM AE model, as well as the previously mentioned anomaly score, can be expanded upon and modified to obtain new perspectives on the model's performance and accuracy. This is further explained in the following subsections.

3.1.3 ***Manipulation of LSTM-AE architecture.***

Bidirectional LSTM. [RP] A bidirectional LSTM model might be suggested here as an alternative to the unidirectional LSTM model used in the design that was previously proposed. The principle of a bidirectional LSTM is to split the neurons of a regular LSTM into two directions - one towards the positive time direction, i.e., the forward state, and another towards the negative time direction, i.e., the backward state. The output of any one of these states' are not coupled with the other but rather they traverse parallel to each

other. In short, it includes doubling the number of LSTM layers in the network, feeding the first layer the input sequence exactly as it is, and feeding the second layer a copy of the input sequence that has been reversed, which changes the direction of information flow. The combined output of the two layers is then taken into consideration. Using two time directions allows for the usage of input data from the present time frame's past and future. (Gopali and Siami Namin (2022))

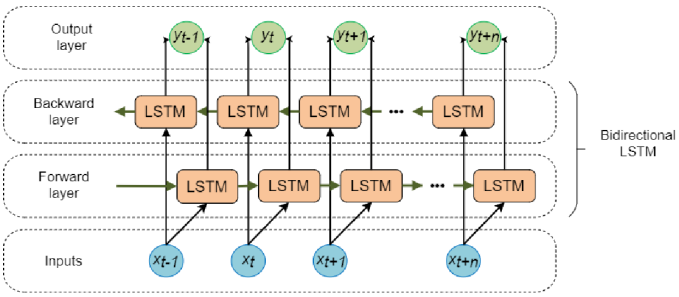


Figure 4: (Kulevome et al. (2021)) Illustration of Bidirectional LSTM Architecture

Multi-layered Output layer. The decoder layer represents a compressed representation of the input sequence, and a linear layer is applied over the decoder layer to turn this representation into the autoencoder's output. The number of features in the hidden state and the size of the output, which is the actual number of features present, are both inputs for this linear layer. The output of this layer is the autoencoder's final output, which is a reconstruction of the input sequence. The output of the autoencoder can also be achieved in the model using a multi-layered linear output layer instead of a single linear layer which was proposed earlier. A sequential list of three layers is applied in this multi-layered linear layer. The number of units in the hidden state is the input for the first layer, which reduces its size to two-thirds of its input. The second layer further reduces this by a factor of 4/9, and the final layer uses the output of the second layer to produce the expected output of the autoencoder, whose size is the number of features. The use of a multi-layer linear network broadens the network's application to more intricate non-linear modifications to the hidden state of the LSTM, which further improves performance for applications like sequence reconstruction in the autoencoder.

Number of LSTM layers. There is a single LSTM layer in both the encoder and decoder in the LSTM AE's basic architecture. It is possible to increase the number of LSTM layers used by the encoder and decoder to 2, in which case the encoder will use two LSTM layers to encode the input data into a higher-dimensional representation, and the decoder will use two LSTM layers to decode the representation back into the original data shape. More LSTM layers in the encoder and decoder can increase modeling capacity. Higher-quality reconstructions are made possible by the autoencoder's ability to learn complicated representations of the input data as it learns additional layers. Additionally, as more layers are added, the model's capacity to generalize is enhanced since the autoencoder can now effectively capture abstract and higher-level features in the input. The extra layers also assist the autoencoder in learning more insightful and organized representations of the input data, which is helpful for later tasks like classification or clustering.

3.1.4 *Anomaly Score computation.*

[NS] An anomaly score in a time series is a numerical value given to a data point or collection of data points that identifies how much the data deviates from the time series' expected behavior. The anomaly score, which can be used to find trends that don't follow the time series' typical pattern, is generated using machine learning algorithms or statistical techniques. If the data point considerably deviates from the expected pattern, it receives a high anomaly score; otherwise, it receives a low anomaly value, which means that the data point is consistent with the time series' typical behavior. The anomaly score $a^{(i)}$ for a point $\mathbf{x}^{(i)}$ at each time-stamp t_i, $a^{(i)} = \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)^T \Sigma^{-1} \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)$, was determined using the reconstructed error vector,$\mathbf{e}^{(i)}$, computed using the L1 loss function in the previous section. In this section, additional anomaly scores for the LSTM AE encoder have been proposed below.

Log-pdf of multivariate normal distribution. For the purpose of calculating the anomaly score, the multivariate Gaussian distribution can be used to determine the negative log-likelihood of the reconstruction error. A lower score denotes a lesser likelihood that the reconstruction error is an anomaly, presuming the error has a Gaussian distribution (Anderson (2009)). For a random variable $\mathbf{x} \in \mathbb{R}^d$ the multivariate log-pdf is given by

$$\log f(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}),$$

where $\boldsymbol{\mu}$ is the mean vector, and $\Sigma \in \mathbb{R}^{d\times d}$ is the covariance matrix, log is the natural logarithm and $|\cdot|$ gives the determinant of a matrix (Anderson (2009)). This function can be used to compute the anomaly score $a^{(i)}$ for a point $\mathbf{x}^{(i)}$ at each time-stamp $\mathbf{t_i}$,

$$a^{(i)} = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{e}^{(i)}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{e}^{(i)}-\boldsymbol{\mu}).$$

Mean function. The mean of the reconstruction error vectors over all the features of the input at every given time-stamp $\mathbf{t_i}$ can be computed as an alternative technique to calculate the anomaly score. The anomaly score $a^{(i)}$ for a point $\mathbf{x}^{(i)}$ can then be understood as, $a^{(i)} = \frac{1}{N}\sum_{i=1}^{n} \mathbf{e}^{(i)}$, where n denotes the number of features in the input series and N is the total number of time-stamps present.

Max function. Analogous to the mean function, a max function can also be used to compute the anomaly score $a^{(i)}$. Here rather than finding the mean of the reconstruction error vectors over all the features, the maximum value in the error vector at a time-stamp $\mathbf{t_i}$ is selected as the anomaly score for a point $\mathbf{x}^{(i)}$. The function can be represented as $a^{(i)} = \max(e^{(i)})$. Here, the assumption is that the error vectors are likely to have at least one feature with a significantly higher error vector value than the other features.

Mahalanobis distance. The Mahalanobis distance relates to the correlation between variables or the variance–covariance matrix. The Mahalanobis distance has an advantage over the Euclidean distance in that it merely needs the covariance matrix to be invertible rather than assuming the data distribution to be normal. For any data point $\mathbf{x}^{(i)}$, the anomaly score can be calculated as

$$a^{(i)} = \sqrt{\left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)^{\mathrm{T}} \Sigma^{-1} \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)},$$

where μ is the mean vector and Σ is the covariance matrix at time-stamp t_i using the reconstruction error vector $\mathbf{e}^{(i)}$ (Dodge (2008)).

The anomaly scoring function chosen can have a big impact on how well an LSTM encoder decoder time series model performs and is able to detect anomalies in the data. These scoring functions have the benefit of being able to capture several facets of the reconstruction mistake, allowing for a more thorough examination of the anomaly score. The connection between the features can also be taken into consideration by using a multivariate Gaussian distribution in the scoring functions, which makes the anomalous score more resistant to changes in the data distribution. The specific properties of the time series data and the application needs should guide the choice of function. Finding the optimal choice of anomaly score function for a certain use case can be aided by experimentation and review of several score functions.

3.2 TCN AutoEncoder

[MK] The baseline TCN-AE is used as a building block for our temporal autoencoder model. It comprises an encoder network, enc(·), and a decoder network, dec(·). The encoder aims to create a condensed representation of the input sequence that captures its main characteristics and can be used for good reconstruction. It uses a TCN network to identify short and long-term patterns, applies a 1x1 convolutional layer with filter size c to reduce the dimension of the feature map, and downsamples the sequence along the time axis using average-pooling. The resulting compressed representation is denoted as $g[n] = enc(x[n])$, where $g : \{0, 1, ..., T/s - 1\} \rightarrow \mathbb{R}^c$, where T is the time series and s is the sample rate that determines the factors by which T is reduced. The decoder reconstructs the original input sequence by upsampling the sequence, applying a second TCN block, and using a 1x1 convolutional layer to restore the original dimension. The TCN architecture of the encoder and decoder can be different, although both encoder and decoder use the same TCN block in this experiment(Thill et al. (2021)). Figure 4 shows the architecture of the baseline TCN-AE.

The architecture of the TCN-AE we propose is composed of multiple components, which will be explained in detail below.

3.2.1 *Temporal Convolutional Networks.*

The TCN is a convolutional network that draws inspiration from various other convolutional architectures but is unique in its combination of autoregressive prediction, residual blocks, simplicity, and very long memory. The TCN consists of a stack of n residual

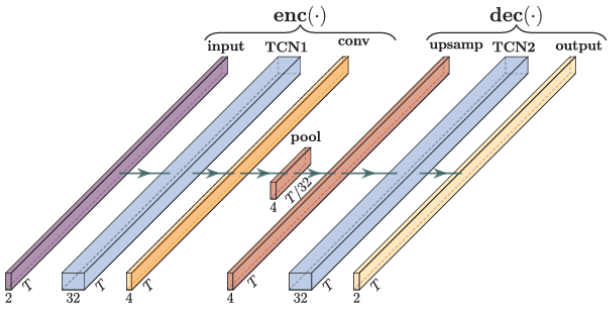


Figure 5: TCN Autoencoder architecture

blocks, each comprising two sub-blocks that contain a dilated convolutional layer, a weight normalization layer, a ReLu activation function, and a spatial dropout layer. The residual block output is bypassed by a skip connection and added to the block output. The TCN's parameters are defined by a list of dilation rates, the number of filters, and the kernel size. The output of each residual block and the final output is a sequence $\mathbf{y} : \mathbb{T} \rightarrow \mathbb{R}^{n_{\text{filters}}}$ where n_{filters} are the number of filters.

3.2.2 *Dilated Convolutional Layers.*

The convolution operation is used to describe the filtering process of a one-dimensional signal $x[n]$, where $x[n]$ is the nth element in the signal. The convolution involves a filter $h[n]$ with weights $h[i]$ and is usually defined as

$$y[n] = (x * h)[n] = \sum_{i=0}^{k-1} h[i] \cdot x[n - i],$$

where $*$ denotes the convolution operator and $y[n]$ is the output of the filter, $h[i] \in R$ is the ith filter weight and k specifies the length of the filter. The input sequence $x[n]$ is slid over by a window of length k containing the filter weights $h[i]$, and at each time step, a weighted sum of $x[n]$ is calculated using the weights $h[i]$. The resulting output signal is one-dimensional and shorter in length than the input signal. To obtain an output signal of the same length as the input, the input signal is typically padded with zeros before applying the filter. The operation is referred to as one-dimensional convolution, as the filter only slides along the time axis. The main concept behind convolutional neural networks is to learn appropriate filter weights instead of presetting them.

For a multivariate input signal $\mathbf{x}[n]$, of dimension d, where each dimension, $\mathbf{x}_j[n]$, is convolved independently with a corresponding sub-filter $\mathbf{h}_j[n]$, resulting in a one-dimensional dot product, $y[n]$. This dot product is calculated by summing the products of each sub-filter and its corresponding input dimension over a range of values as shown in the equation

$$y[n] = (\mathbf{x} * \mathbf{h})[n] = \sum_{i=0}^{k-1} \mathbf{h}[i]^{\top} \mathbf{x}[n - i].$$

Dilation rate, denoted as q, is an additional parameter in dilated convolution, which determines the number of skipped elements between filter taps in the input signal $\mathbf{x}[n]$. Dilated convolution is expressed as a sum of products between sub-filters and their

corresponding input dimensions, where the filter taps are spaced by q elements in the input signal. The dilated convolution equation is as:

$y[n] = (\mathbf{x} *_q \mathbf{h})[n] = \sum_{i=0}^{k-1} \mathbf{h}[i]^\top \mathbf{x}[n - qi]$.

The primary strategy is to construct a series of dilated convolutional layers, where the dilation rate progressively increases with each additional layer. Typically, the dilation rate for the initial layer is set to 1, and it doubles for every subsequent layer. By adopting this technique, we can significantly increase the model's receptive field, without decreasing the resolution, unlike what occurs with pooling or strided convolutions.

A convolutional layer generally consists of numerous separate filters, and the resulting individual outputs, $y[n]$, are organized into a feature map. When a signal, $x[n]$, with a length of T_{train} is processed through a convolutional layer that has $n_{filters}$ filters, the resulting feature map will have dimensions of $T_{train} \times n_{filters}$ (in the case of a padded signal) (Thill et al. (2021)).

3.2.3 *TCN:Anomaly score computation*.

[NP]This section defines various distance measures as scoring functions in the context of anomaly detection. To start with, scoring functions are some function of reconstruction error which is obtained by calculating the difference between the actual time series and the reconstructed series from the model. Basically, the anomaly score quantifies to what extent the reconstructed value at a given timestamp deviates from the actual value. The greater the deviation, the larger the value of an anomaly score at a given timestamp. In this section, we will be considering Mahlanobis distance, mean reconstruction error, L1 norm as well as L2 norm as scoring functions.(Thill et al. (2021))

Distance measures as scoring functions: Reconstruction error $e^{(i)}$:

$$e^{(i)} = (test^{(i)} - Rec^{(i)})$$

test: original input data , *Rec*: reconstructed output, *n*: number of features

Mahlanobis distance: Mahlanobis distance is a distance measure that determines the distance between a particular point and the distribution in multidimensional space. For that reason, it is a widely used distance measure for the purpose of anomaly detection as it detects the points which significantly differ from the distribution of the rest of the dataset. It considers the correlation between different features since it is calculated with a covariance matrix, which in terms is helpful for datasets containing highly correlated variables. Anomaly detection in higher dimension data can easily be supported with Mahlanobis as it does not suffer from the curse of dimensionality, unlike other scores. (Thill et al. (2021))

$$M^{(i)} = \sqrt{\left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)^{\mathrm{T}} \Sigma^{-1} \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)}$$

,

where $M^{(i)}$ is the mahalnobis distance of observation $x^{(i)}$ at any given timestamp, μ is the mean vector and Σ is the covariance matrix of the observations of a given time-stamp with respect to reconstruction error vector $\mathbf{e}^{(i)}$.

Mean distance: For a given timestamp, the mean reconstruction error is the average of the differences between the input data and its reconstruction for all the features. A significantly greater reconstruction error suggests that the data sample is more likely to be an anomaly since it differs more from the typical data that the autoencoder was trained on. (Patel (2019))

$$Mean^{(i)} = \frac{1}{N} \sum_{i=1}^{n} \mathbf{e}^{(i)}$$

L2 norm: For a given timestamp, L2 norm (also known as Euclidian distance) is calculated by taking the sum of the squared difference of the original input and its reconstruction across all the features. L2 norm magnifies the degree of difference between the original input and its reconstruction as a consequence, it gets easier to detect anomalies with even minor deviations from the original input. One disadvantage of L2 norm is that it assumes normal distribution. (Patel (2019))

$$L2^{(i)} = \sqrt{\sum_{i=1}^{n} (test^{(i)} - Rec^{(i)})^2}$$

L1 norm: For a given timestamp, the L1 norm (also known as Manhatten distance) is calculated by taking the sum of the absolute difference of the original input and its reconstruction across all the features. As compared to the L2 norm, the L1 norm is more resilient to outliers as L1 norm provides equal weight on all errors. (Patel (2019))

$$L1^{(i)} = \sum_{i=1}^{n} |test^{(i)} - Rec^{(i)}|$$

3.3 Latent Space Visualization

[AB] A compressed representation of the input data, known as the latent space of an autoencoder, captures the most prominent aspects of the original data in a smaller-dimensional space. It is believed that by projecting the input data onto a lower-dimensional latent space, we can learn a compressed representation of the data that is more illuminating and better reflects its key features.

Since the autoencoder is a reconstruction-based model that aims to capture the most prominent elements of the input data rather than cluster them, using the latent space for clustering presents some difficulties. Because of this, it could be difficult to distinguish between distinct clusters in the latent space and the latent space may not be suited for clustering. The possibility that the latent space is not continuous or smooth presents a related difficulty since it could prevent small changes in the latent space from reflecting tiny changes in the input data. This can make it challenging to

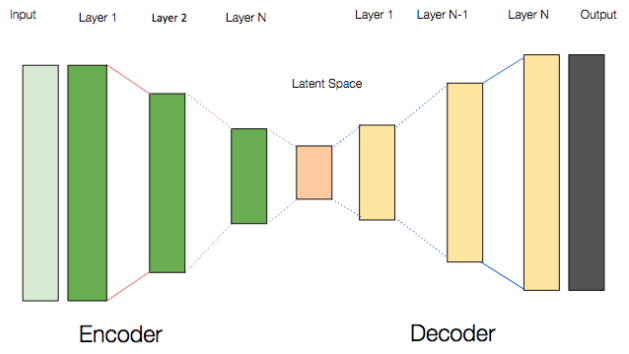


Figure 6: Encoder-Decoder system with latent space.

carry out operations on the latent space that correlate to significant changes in the input data, such as interpolation or modification. Researchers have created methods that promote the latent space to be smooth and continuous, like adversarial training or regularization, to overcome this problem.

The possibility that the latent space may be sensitive to the particular training data used to learn it presents another difficulty. The latent space might not be able to produce realistic faces of various races or genders, for instance, if the training data only comprises faces of a particular race or gender. More varied training data can be used to solve this, as can explicitly encouraging the latent space to be gender- or race-invariant.
Latent space offers a lot of opportunities despite these limitations. Data augmentation, which involves creating new examples of the incoming data by modifying the latent space, is one possibility. The latent spaces of two existing photos can be interpolated to create new facial images, for instance, or the latent space of a language model can be sampled to create new text.

Additionally, the choice of the dimensionality of the latent space and the design of the autoencoder architecture can also influence the clustering performance of the model. If the latent space dimension is too low, it may not be able to capture the complexity of the data, while if it is too high, the model may overfit to the training data and fail to generalize to new data. Figure 6 shows the general architechture of latent space in encoder decoder model.

The performance of an autoencoder for finding latent space for clustering depends on a variety of factors, including the dimension of latent space, the design of autoencoder, and the clustering method. Further model parameter adjustment may be necessary if the model has not yet developed a compressed representation of the data that is sufficiently informative. This might occur if the latent space cannot distinguish between two distinct clusters in the original data (Bengio et al. (2013).

[MK]The encoder output of TCN (Temporal Convolutional Networks) and LSTM models can be taken as a latent space representation of the input data. This latent space can be used for downstream tasks such as classification or clustering. TSNE (t-Distributed Stochastic Neighbor Embedding) is a powerful technique for visualizing high-dimensional data in a lower-dimensional space. The encoder output of each model is given to the TSNE function to reduce the dimensionality to 2. The goal is to visualize the output of the encoder of both model and see if any clusters or patterns can be observed in the lower-dimensional space.

After reducing the dimension of the latent space to a 2D space using TSNE, each data point is labeled according to the label of the corresponding data point in the original dataset. A scatter plot can be created to visualize the data. In this plot, each data point represents the output of the model's encoder for a particular input, and the position of the point in the plot corresponds to the reduced dimensionality of the data.

By visually inspecting the plot, it is possible to see if any clusters or patterns exist in the data. This can help to identify if the model is accurately capturing the underlying structure of the data. For example, if the plot shows clear clusters of similar data points, this may indicate that the model is effectively capturing the patterns in the data. On the other hand, if the data points are spread out evenly across the plot, this may indicate that the model is not accurately capturing the underlying patterns in the data.

3.4 DGHL

[SP] DGHL is a new group of generative models developed to detect anomalies in time series data. Here, maximization of the observed likelihood through alternate backpropagation and short-run MCMC are used to train the model. Hence, it does not depend on encoders or discriminators like VAEs and GANs. The training time of DHGL is 10x times less compared to other generative models making it computationally more efficient. DHGL also reduces the risk of overfitting and is robust to missing data and variable features. The following sections give an overview of the algorithms involved, the architecture and the training of DGHL. (Challu et al. (2022))

3.4.1 ***Short-run MCMC:.*** [AT] Monte Carlo approximation is a non-parametric approximation method for estimating a value by sampling from a distribution of possible outcomes. For higher dimensional problems, the main issue is to find the posterior samples efficiently, in this case, it is more common to use Markov chain Monte Carlo or MCMC. There are several variations of MCMC, including Metropolis-Hastings, Gibbs sampling, and the Bootstrap Filter. The DGHL paper that we referred to, used a generative model based on short-run MCMC. Short-run MCMC is shown as an inferential model in which we can optimize the hyper-parameters depending on different criteria. This makes the algorithm fall between the MCMC and variational inference. (Nijkamp et al. (2020))

3.4.2 ***Hamiltonian Monte Carlo:*** The Hamilton mechanic relates the kinetic energy to the potential energy using differential equations. Hamiltonian equation determines the relation between the position and momentum. The hamiltonian has a vector of d-dimension for the position given as q and a vector of d-dimension for momentum given as p. Hamiltonian is defined by these two

functions p and q such as H(p,q).

$$H(q,p) = U(q) + K(p) \tag{1}$$

where U(q) is the potential energy with respect to position q and K(p) is the kinetic energy. (Neal et al. (2011))

3.4.3 ***Hierarchical Latent Factors.*** [SP] Consider m features and $\mathbf{Y} \in \mathbb{R}^{m \times s_w}$ be the multivariate time-series window of size s_w. This window is equally divided into sub-windows $\mathbf{Y_j}$ which is given by,

$$\mathbf{Y_j} = G_\beta(s_j) + e_j \tag{2}$$

where,

$s_j = F_\alpha(z^1_{\left[\frac{j}{a_1}\right]}, \ldots, z^L_{\left[\frac{j}{a_1}\right]})$,
F_α is the *State model*,
G_β is the *Generator model*,
$\alpha, \beta \in \theta$ are the parameters,
$\mathbf{Y_j} \in \mathbb{R}^{m \times \frac{s_w}{a_L}}$ with $j = 0, \ldots, a_L$, $a = [a_1, \ldots, a_L]$
L is the number of levels,
l is the level and a_l refers to the number of consecutive sub-windows,
s_j is the state vector,
$e_j \sim N(0, \mathbf{I}_D)$,

Hierarchical latent factor space which can be inferred using Langevin dynamics is given by,

$$Z = \{z^l_{\left[\frac{j}{A_l}\right]} \in \mathbb{R}^{d_l}\}_{l,j} \tag{3}$$

Figure 7 shows the architecture of *Generator*. Here the *Generator model* uses top-down Convolution Network (ConvNet). The input state vector is mapped to the multivariate time-series window using it. The *State model* uses concatenation layer, in which L latent vector inputs are present for each sub-window. These latent vectors are tied on each level l.

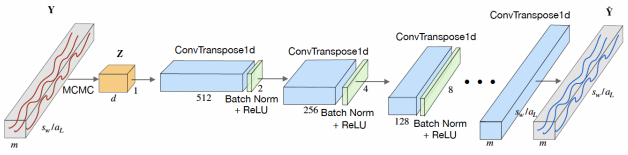


Figure 7: Architecture of the *Generator* (Challu et al. (2022))

Larger lower hierarchy level reduces the number of false positives by reducing the reconstruction error and making the model more flexible. Also, the contextual anomalies are detected better when there are larger tied vectors, which makes the model more strict. Hence, the flexibility of the model is controlled by the relative size of lowest state vector level and the upper levels. (Challu et al. (2022))

3.4.4 ***Training with Alternating Back-Propagation.*** [SP] The Alternating Back-Propagation algorithm maximizes the observed log-likelihood to learn parameters θ.

$$L(\theta) = \sum_{i=1}^{n} log p_\theta(\mathbf{Y}^{(i)}) = \sum_{i=1}^{n} log \int p_\theta(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) d\mathbf{Z}^{(i)} \tag{4}$$

where,

$\mathbf{Y}^{(i)}, i = 1, ..., n$ is the training set,
$\mathbf{Z}^{(i)}$ is the latent vector for ith window

Analytically, the observed likelihood $L(\theta)$ is elusive. Hence, the gradients $L(\theta)$ for a specific observation, $L'(\theta)$ can be reduced to the below equation as,

$$\frac{\partial}{\partial\theta} log p_\theta(\mathbf{Y}^{(i)}) = \frac{1}{p_\theta(\mathbf{Y}^{(i)})} \frac{\partial}{\partial\theta} \int p_\theta(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) dz \tag{5}$$

$$= \mathbb{E}_{p_\theta(Z^{(i)}|\mathbf{Y}^{(i)})} \left[\frac{\partial}{\partial\theta} log p_\theta(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) \right] \tag{6}$$

where, $p_\theta(\mathbf{Z}^{(i)}|\mathbf{Y}^{(i)}) = p_\theta(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) / p_\theta(\mathbf{Y}^{(i)})$ refers to the posterior. Using the MCMC algorithm, described in the Section 3.4.1, the expectation in the equation 5 can be approximated as,

$$\mathbf{Z}^{(i)}_t + 1 = \mathbf{Z}^{(i)}_t + \frac{s}{\sigma_z} \frac{\partial}{\partial \mathbf{Z}^(i)} log p_\theta(\mathbf{Z}^{(i)}_t | \mathbf{Y}^{(i)}) + \sqrt{2s}\epsilon_t \tag{7}$$

$$= \mathbf{Z}^{(i)}_t + \sqrt{2s}\epsilon_t + \tag{8}$$

$$\frac{s}{\sigma_z} \left[(\mathbf{Y}^{(i)} - f(\mathbf{Z}^{(i)}_t, \theta)) \frac{\partial}{\partial \mathbf{Z}^(i)} f(\mathbf{Z}^{(i)}_t, \theta) - \mathbf{Y}^{(i)}_t \right] \tag{9}$$

where, t is the time step, σ_z is the one which controls the relative size of the injected noise and s is the step size. A single sample of posterior is taken in iteration by using Langevin Dynamics (Neal et al. (2011)), Hamiltonian Monte Carlo algorithm by the Alternating Back-Propagation. In this iteration, the latent factors are chosen in such a way that it minimizes the residual on reconstruction $\mathbf{Y}^{(i)} - f(\mathbf{Z}^{(i)}_t, \theta)$. When σ_z is high, the posterior will resemble the prior and with low σ_z, the posterior differs more from the prior. The likelihood is approximated using Monte Carlo as,

$$L'(\theta) \approx \frac{\partial}{\partial\theta} log p_\theta(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) = \frac{1}{\sigma^2} (\mathbf{Y}^{(i)} - f(\mathbf{Z}^{(i)}_t, \theta)) \frac{\partial}{\partial\theta} f(\mathbf{Z}^{(i)}_t, \theta) \tag{10}$$

Mini-batch Alternating backpropagation iterates in two steps. In step one, the latent vectors are determined and in step two, Generator model gets the latent vector as input and SGD is computed to update the parameters. When Bayesian posterior sampling is used, the overfitting is reduced (Welling and Teh. (2011)). Hence, in DGHL, the risk of overfitting is reduced with the help of MCMC sampling.

The anomaly score for each time-stamp t, is calculated as Mean Square Error(MSE),

$$s_t = \frac{1}{m} \sum_{i=1}^{m} (y_{i,t} - \hat{y}_{i,t})^2 \tag{11}$$

Algorithm 1: Mini-batch Alternating backpropagation (Challu et al. (2022))

Input : training examples $\mathbf{Y}^{(i)}, i = 1, ..., n$, Langevin steps l, learning iterations $\mathbf{T}$
Output : learned parameters θ, inferred latent vectors $\{\mathbf{Z}^{(i)}, i = 1, ..., n\}$
Let $t \leftarrow 0$, initialize θ ;
Initialize $Z^{(i)}$, for $i = 1, ..., n$;
while $t < T$ **do**
  Take a random multi-batch $\mathbf{Y}^{(j)}, j = 1, ..., n$;
  Inferential back-propagation: For each j, run l steps of Langevin dynamics to sample $\mathbf{Z}^{(j)} \sim p(\mathbf{Z}|\mathbf{Y}^{(j)}, \theta)$ following equation 7 ;
  Learning back-propagation: Compute gradients following equation 10 and update θ ;
  $t \leftarrow t + 1$;
end

with m time-series, $y_{i,t}$ is the data of interest, $\hat{y}_{i,t}$ is the reconstructed target. Anomalies can be detected sooner with the help of step size s and window size s_w. (Challu et al. (2022))

3.4.5 ***Anomaly Detection with missing data.*** [SP] As discussed in the previous section, the latent factors are chosen in such a way that the residual of the reconstruction is minimized. This makes the model to overcome the problem of missing data by calculating the residuals using only the observed values $\mathbf{Y}_{obs}$ and inferring latent vector $\mathbf{Z}$. Then, the posterior distribution, $p_\theta(\mathbf{Z}|\mathbf{Y}_{obs})$, corresponding to inferred vector conditioned on the observed value is computed. There is no mapping between the learnable parameters and latent space. At first, the DGHL reconstructs observed data with the missing values and later, it tries to recover the missing data with good precision. Hence, it can be said that DGHL is more robust to missing data. (Challu et al. (2022))

3.4.6 ***Synthetic Anomaly Injection:*** [AT] Synthetic data is artificial information that is produced by a code rather than being gathered from real-time data. Synthetic data allows testing various algorithms which is not possible with real-time data. When compared to real data anomalies, which are difficult to identify, synthetic anomaly generation is quite helpful. In this experiment, we are injecting synthetic anomalies in the given dataset and use this data for training on different models. We use different data augmentation techniques to inject synthetic anomalies. For injecting contextual anomalies, we use the method of Contextual Outlier Exposure and for injecting point anomalies, the method of Anomaly Injection presented in (Carmona et al. (2021))

Contextual Anomalies Injection: The contextual anomalies are injected into the dataset using the method Contextual Outlier Exposure presented in (Carmona et al. (2021)). In this method, we take randomly a window in which we inject anomaly by replacing the values with the values in another dimension time window. The injected values will be different from the surrounding values which causes it to be an anomaly. We inject the anomalies in only some set of dimensions. The dimensions in which we inject anomalies are the ones where we have few or no true anomalies. Figure 8 is an example of a contextual anomaly generated in the Chemie building on the dimension 6_TmpDiff.

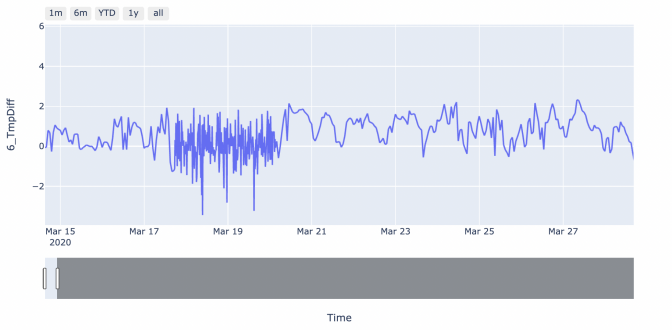


Figure 8: Contextual Anomaly in dimension 6_TmpDiff of Chemie Building

Point Anomalies Injection: In this method, a spike at a point is injected. For a dimension that has few or no true anomalies, we randomly select a sequence of timestamps known as a location where the point anomaly is injected. We inject a spike in that location. The value of the injected spike is equal to 0.5 and 3 times the interquartile range of the location points. Figure 9 is an example of point anomalies in the Chemie building for the 6_WTarif1 dimensions. (Carmona et al. (2021))

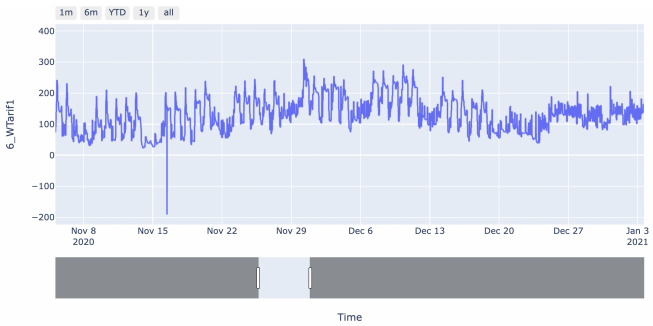


Figure 9: Point Anomaly in dimension 6_WTarif1 of Chemie Building

3.5 General methods

[NP] This subsection will define some general methods used for experiment purposes on all the models:

3.5.1 Min-Max normalization: When a dataset's values need to be scaled to a certain range, usually between 0 and 1, min-max normalization is employed as a data pre-processing technique. The process includes differencing the minimum value of the feature from each data point followed by dividing the difference by the range of the feature in order to scale the difference within the range

of 0 and 1 rescaling the maximum value of features to 1 and the minimum value of features to 0. (Patel (2019))

$$x_{norm}^{(i)} = (x^{(i)} - x_{min})/(x_{max} - x_{min})$$

3.5.2 *Evaluation matrix and measures:*

Confusion matrix: The effectiveness of a classifier model is usually assessed using a confusion matrix. It presents a comparison of the expected and actual classifications in table format same as table 1. All classifications(labels) are further grouped into two categories named actual class (represents the true label of observation) and predicted(represents the label predicted by the model for observation) based on the values of observations under above mention classes, the observations are further divided into 4 categories as follows:
True positive (TP): Number of observations with predicted label=1 and actual label=1.
False positive (FP): Number of observations predicted label=1 and actual label=0.
True negative (TN): Number of observations with predicted label=0 and actual label=0.
False negative (FN): Number of observations with predicted label=0 and actual label=1. (Alpaydin (2010))

Table 1: Confusion matrix for binary classification

Actual/Predicted	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

F1 score: Before deriving the F1 score, it is important to understand two crucial factors which are defined as follows:

Precision: It represents the ratio of observations that are truly predicted as positive (TP) out of all positive predicted (TP+FP) observations.(Alpaydin (2010))

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: It represents the ratio of observations that are truly predicted as positive (TP) out of all actual positive (TP+FN) observations. (Alpaydin (2010))

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The F1 score can be derived by calculating the harmonic mean of precision and recall with equal weightage assigned to both factors. (Alpaydin (2010))

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

AUC score: The AUC score evaluates a model's capacity to differentiate between positive and negative classes. In order to derive the AUC score, first a curve known as the Receiver Operating Characteristic (ROC) curve is generated. The false positive rate (FPR) is plotted against the true positive rate (TPR) at various classification thresholds to create the ROC curve. Where, the TPR also known as recall (or sensitivity), is calculated by taking the ratio of correctly predicted positive observations to the actual total positive instances, whereas the FPR can be derived from the ratio of incorrectly predicted positive observations to the actual total negative instances.(Alpaydin (2010))

$$\text{TPR(Sensitivity/Recall)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Finally, the AUC score is determined as the area under the ROC curve. The AUC score always ranges from 0 to 1 with 0.5 indicating that the model is underperforming and is probably just making random guesses. AUC score of 1 for a model that perfectly distinguishes between positive and negative classes and hence indicates the most accurate model. (Alpaydin (2010))

3.6 Overview of the datasets

[KT]This section will provide an overview of the dataset by describing each dataset used in the experiment as well as commonly used hyperparameters during modeling. After that, we will give the model-wise inter findings while stating the evaluating parameters employed. We have used the dataset of six buildings for all the experiments by all three models. Table 2 provides a broad summary of the information for the six datasets that are most frequently utilized. We will go into more depth about the actual experiment setting in the next subsections.

Commonly used Building Dataset: In this subsection we will include datasets for the buildings that are utilized by each of the three models: OH12, OH14, HG2, Großtagespflege, Kita Hokida, Chemie, EF40, EF40a, EF42, and Erich Brost. While hourly data is gathered for EF40a and Chemie, we see a frequency interval of 15-minutes for all other datasets of buildings. Except for HG2 and Großtagespflege, the splitting date for the building dataset was the 1st of October 2021. For those two datasets, we used the 1st of April 2022. The table details the number of features and size of the training and test data sets.

Combined Building Dataset: For DGHL, we integrated additional Office Building and Chemietechnik Building Datasets. Hence, we employed two distinct strategies described in detail in section 6.3. For the first mentioned one, we integrated the dataset of the Erich Brost Institute, Building EF40, and EF40a. We aggregated the three datasets and treated them as one single building. Similar approach was used for the Chemietechnik Building Dataset. With these datasets, we sought abnormalities and put our interest in finding one building with fewer or no anomalies. As a general strategy, we used aspects that are present in all buildings and added

one or two features that are unique to one or two buildings. The common features, such as Wärmetarif, VolumenKanal, WV+tariflos, WV+Tarif1, and features exclusive to EF40a (BV+tariflos and BV-tariflos), were selected for the office buildings. For Chemietechnik, we used the features associated with OBIS Kennzahl 6 (Wärmeleistung, Wärmetarif, Rücklauftemperatur, and Vorlauftemperatur) and OBIS Kennzahl 8 (Volumen Kanal) as common ones. Furthermore, we included the features "1 BV+Tarif1" and "1 WV Arbeit Tarif1" which are exclusive to one of the buildings. A 15-minute frequency is shared by the first and third Chemietechnik Buildings, and hourly records are kept in the second building (features starting with Obis Kennzahl 6). Regarding the office building, EF40a has hourly, while the remaining two have 15-minute frequencies.

Synthetic Building Dataset: To demonstrate the use and advantages of the synthetic dataset approach described in section xxx, two buildings were chosen to compare the standard dataset with the synthetic one, in which contextual and point anomalies were injected. Herefore, we used Chemietechnik and EF42 Building Dataset as they are the most similar to one another among all other buildings, with EF42 having the fewest abnormalities.

After combining Chemietechnik and Office Buildings respectively into a single Building using the shared Data Management and Data Preprocessing, we ran the model on each building separately.

Dummy Dataset: [RP] A two-dimensional synthetic dataset which contains 10,000 samples, where each sample is a set of feature values was generated using an AnomalySineGenerator function. This dataset was injected with 250 anomalies and these contain both contextual and point anomalies. Overall, this synthetic dataset provides a way to evaluate the performance of the LSTM and TCN anomaly detection algorithms to identify anomalous data points.

4 DATA UNDERSTANDING

In this section, we perform various small tasks to understand the raw data, find hidden patterns, and measure the data quality.

4.1 Data Description

[SP] The dataset used in this project is an energy consumption data of TU Dortmund University. It contains data from 13 different buildings contained in 20 files. Table 2 describes the shape of dataframe, number of missing values and time period of each building file. Each building contains different number of features. Some of the important features present in the buildings are explained below,

OBIS Bezeichnung (Object Identification System) codes are used for the unique identification of measured values from a particular device, *Betriebsstunden* is the operating hours measured in hours, *Fehlerstunden* is the amount of time that there was error in operating hours and is measured in hours, *Wärmeenergie Tarif 1* refers to heat energy consumption which is measured in kWh, *Volumen* (m^3) is the amount of water flowing through the pipes of the observed part of the heating system, *Durchfluss* (m^3/h) is the speed of water flowing through the pipes, *Vorlauftemperatur*(°C) is the temperature of the heated water before entering the heating pipes and radiators, *Rücklauftemperatur*(°C) is the temperature of the water after running through the heating pipes (at the end of the circulation). This water will then be reheated, *Temperaturdifferenz*(K) refers to the temperature difference between *Vorlauftemperatur* and *Rücklauftemperatur*, *Wärmeleistung* (kW) is the usable heat output.

Volumen Kanal 1, *Volumen Kanal 2* and *Volumen Kanal 3* are the amount of water flow in different volume channels (m^3). These features represent the amount of water consumption.

Electric energy consumption is represented by the following features, *WV+ Arbeit tariflos* is the positive active energy total, *WV+ Arbeit Tarif 2* is positive active energy total tariff 2, *WV- Arbeit tariflos* is the negative active energy total, *BV+ Arbeit Tarif 1* is the positive reactive energy tariff 1, *BV- Arbeit Tarif 1* refers to the negative reactive energy total, *BV+ Arbeit tariflos* is the positive reactive energy total, *BV- Arbeit tariflos* indicates negative reactive energy total, *WV+ Momentanwert Tariflos* is the positive active instantaneous power.

Großtagespflege building has the most number of features when compared to others. Chemietechnik G2-F2 contains more data than the other buildings followed by EF 40. EF40a_01_22-07_19 building has the least amount of data followed by Chemietechnik G3-F3, Chemie_01_26-07_19 and OH12_01_26-07_19 buildings. Chemietechnik G2-F2 has the most number of missing values followed by Chemie. OH14_01_26-07_19 has no missing values at all and Chemietechnik G3-F3 has only seven missing values.

The time period for the buildings (Chemie, Chemie_01_26-07_19), (Kita Hokido, Kita Hokido_05_22_20-07_19_22), (OH12, OH12_01_26-07_19), EF 42, EF 40, (EF40a, EF40a_01_22-07_19), Erich-Brost-Institut, Chemietechnik G2-F2 ranges from the year 2020 to the year 2022. The buildings (Großtagespflege, Großtagespflege_04_05-07_19), (HGII, HGII_01_26-07_19), (OH14, OH14_01_26-07_19), Chemietechnik G3-F3, Chemietechnik G1-F1 have time period range from the year 2021 to the year 2022.

4.2 Outside Temperature:

[AS] What is the temperature outside? This is the one question we always ask when discussing energy usage since the weather mainly determines how much power, heating, and water are consumed. We have data on energy consumption for various TU Dortmund campus buildings that were gathered for this research using numerous sensors.

We obtain information on outside temperatures from the Deutsche Wetterdienst (*DWD*) Wuppertal because it is closest to Dortmund. They keep track of the outdoor temperature. We use energy consumption data hourly to calculate the effect [25].

The effect of outside temperature on energy consumption can vary depending on several factors, including the type of heating or cooling system used, the amount of solar gain, and the insulation of the building. In this part of the research, we capture the direct effect of outside temperature on energy usage, excluding all other external features. We perform the Pearson correlation test to calculate the

Table 2: Description of data

Building	Shape of dataframe	No.of missing values	Time period(Begin)	Time period(End)
Chemie	(65531, 22)	1088547	2020-03-14 15:15:00	2022-01-26 05:30:00
Chemie_01_26-07_19	(16706, 14)	96929	2022-01-26 00:45:00	2022-07-19 01:45:00
Großtagespflege	(15626, 46)	201514	2021-10-24 08:30:00	2022-04-05 02:30:00
Großtagespflege_04_05-07_19	(10092, 25)	407	2022-04-05 00:00:00	2022-07-19 02:30:00
HG II	(12072, 23)	1425	2021-09-22 12:45:00	2022-01-26 05:15:00
HGII_01_26-07_19	(16712, 21)	87	2022-01-26 00:00:00	2022-07-19 02:30:00
Kita Hokido	(65531, 8)	294900	2020-03-14 15:15:00	2022-01-26 05:15:00
Kita Hokido_05_22_20-07_19_22	(65531, 8)	295815	2020-05-22 12:00:00	2022-04-05 02:15:00
OH12	(43652, 27)	275837	2020-10-28 12:45:00	2022-01-26 05:15:00
OH12_01_26-07_19	(11498, 20)	11848	2022-01-26 00:00:00	2022-05-25 19:00:00
OH14	(19564, 18)	15361	2021-07-06 11:45:00	2022-01-26 05:15:00
OH14_01_26-07_19	(16712, 17)	0	2022-01-26 00:00:00	2022-07-19 02:30:00
EF 42	(65531, 20)	132204	2020-03-14 15:00:00	2022-01-26 05:15:00
EF 40	(65531, 35)	27397	2020-03-14 15:00:00	2022-01-26 05:15:00
EF40a	(65531, 11)	148067	2020-03-10 15:15:00	2022-01-22 05:30:00
EF40a_01_22-07_19	(17097, 6)	40531	2022-01-21 22:45:00	2022-07-19 01:30:00
Erich-Brost-Institut	(65531, 16)	295218	2020-03-10 15:30:00	2022-01-22 05:45:00
Chemietechnik G3-F3	(9209, 24)	7	2021-10-22 08:45:00	2022-01-26 05:30:00
Chemietechnik G2-F2	(65531, 38)	1316390	2020-03-14 15:15:00	2022-01-26 05:30:00
Chemietechnik G1-F1	(9202, 12)	3694	2021-10-22 10:30:00	2022-01-26 05:30:00

impact of outdoor temperature on heating, electricity, and water consumption.

Pearson correlation coefficient is widely used in statistics and data analysis. It provides a simple and intuitive measure of the strength and direction of the linear relationship between two variables. It takes values between −1 and +1.

Results: From the table below, we can infer that the outside temperature significantly affects energy consumption. The results are shown for $OH14$ buildings. Features like Wärmeenergie (representing heating consumption) negatively correlate with outside temperature, meaning if the temperature increases, the heating consumption decreases. And on the other hand, we have the $OH12$ building, where Wärmeenergie positively correlates with the outside temperature. We know that $OH12$ has server rooms, which we can count as one reason for a positive correlation between energy consumption and outside temperature. We also divided our data into four parts, lecture period, break period, and summer and winter sessions, and then tried to capture the effect of outside temperature on energy consumption. In conclusion, we observed that the outside temperature affects other features, and we added outside temperature as an exogenous variable while training our model. Further results are mentioned in Table 10 in Appendices.

Table 3: Result of Pearson Correlation test For OH14 & OH12

Features	OH14	OH12
Wärmeenergie Tarif 1	-0.72	0.17
Volumen Kanal 1	-0.43	0.19
Wärmeenergie Tarif 2	-0.77	-
WV+Arbeit Tarif 1	-0.79	0.04

4.3 Repeating Patterns

[KT] In this subsection, we will examine our ten-building dataset to check if any repeating patterns can be detected. Initially, we examined the general recurring pattern, both annually and monthly. The repeating pattern was not unexpected, given that we are considering data on energy usage. During winter months, from September to March, we can observe high peaks, whilst, over summer, lower peaks are visible. Note that during the winter break, we notice a decrease in energy consumption, which is shown in the appendix 29 .

Plotting the data weekly, we can observe a pattern for predefined features (Vorlauftemperatur, Rücklauftemperatur, Wärmeleistung, Durchfluss, Volumen Kanal1) for nearly most of the buildings. It fluctuates from its average value during the weekdays and is at its lowest on Saturdays. For Volumen Kanal, the numbers are almost zero on Saturdays and Sundays and start increasing again on Mondays (Figure 30 in Appendix)

From August until the start of the semester, we can see that the values for Wärmetarif, Volumenkanal, and Wärmeleistung increases from 7 or 8 am and then decline slowly, starting from approximately 3 or 4 pm. Every two hours, Durchfluss, Vorlauftemperatur, and Rücklauftemperatur follow a pattern that often peaks at eight in the morning and then either decreases or increases. As Vorlauf- and Rücklauftemperatur reach their lowest peaks at 8 am and revert to their average values, Durchfluss reaches its greatest peak at that time.

During the winter period (starting from 1st November till 31st December), Wärmetarif, Volumenkanal, and Wärmeleistung start to increase around 6 o'clock in the morning and decrease after 8 or 9 pm. Furthermore, we see that during the winter break, the

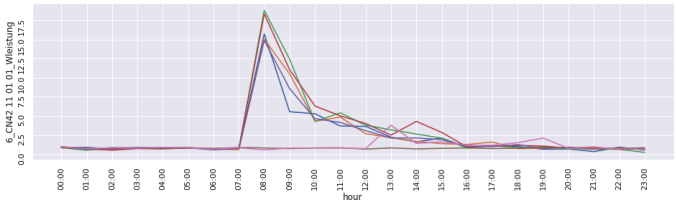


Figure 10: Repeating pattern for the feature Wärmeleistung during summer

repeating pattern of these features increases after 10 am, reaching its peak at about 11 am, and then begins to fall after 9 pm. During the winter vacation, the alternating pattern with Durchfluss and Vorlauf- and Rücklauftemperatur is also apparent. Finally, during exam periods(from 1st January till 28th of February) we noticed that Wärmetarif, Volumen Kanal and Wärmeleistung starts to increase from 6 am till 10 pm. (figure 31, figure 32 in Appendix).

Thus we conclude that there is a repeating pattern, with low energy consumption in the mornings, having a peak in the mid-afternoon, and almost zero again in the evening.

4.4 Seasonality

[MK]Seasonality is a recurring pattern in data that occurs at regular intervals, such as daily, weekly, monthly, or yearly. It refers to the regular fluctuations in data that are caused by a repeating event, such as the changing of seasons, holidays, or other predictable events. In time-series data, seasonality can have a significant impact on the overall trend and forecast accuracy.

Time series data can be separated into trend, seasonality and residual component. Mathematically, for an additive model, the observed time series is modeled as the sum of its trend, seasonal, and residual components. The equation for the additive seasonal decomposition model can be written as follows:

$$Y(t) = T(t) + S(t) + e(t),$$

where $Y(t)$ is the observed time series at time t, $T(t)$ is the trend component at time t, $S(t)$ is the seasonal component at time t, $e(t)$ is the irregular or error or residual component at time t.

Figure 11 shows the seasonality for the consumption of electricity, water and heat energy for the building Chemie. A general pattern can be observed for all the 3 features with a high usage from start to middle of the month, followed by a gradual decrease towards the end. This indicates that the energy consumption tends to be higher during the first half of a month. This pattern can be observed for all the buildings.

4.5 Trends

[RP]A trend in the energy consumption data for a university can be identified as a long-term rise or fall in the data . A trend is shift in the time-series data, which can follow an upward or downward trend. It can also move in a linear or nonlinear fashion. The trend

Figure 11: Results of seasonality for the building Chemie for electricity, heat energy and water

in energy consumption data for the university is characterized by external factors such as fluctuations in temperature or durations of lecture and lecture free periods.

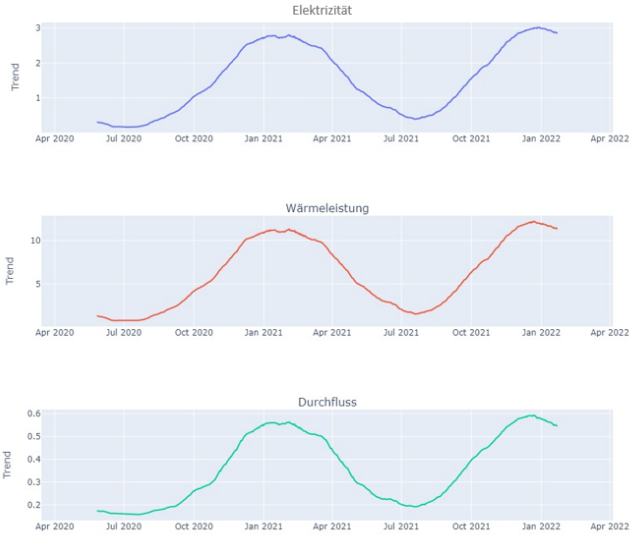


Figure 12: Results of trend for the building Kita Hokida for electricity, heat energy and water

Figure 12 shows the trend for the consumption of electricity, water and heat energy for the building Kita Hokida. A general pattern can be observed for all the 3 features with increased usage from the months of October to February and decreased usage from the month of March to September. This indicates that the energy consumption tends to be higher in the winter and lower during the months of summer. An exception from this pattern can be observed for the building OH12, where the energy usage does not alter significantly with the changes in seasons. The presumptive

cause may be related to the existence of server rooms, which must be maintained at a specific temperature.

4.6 Covid Effects

[NS]In this section, we try to understand the impact of the Corona pandemic on the energy consumption of various buildings in the university. Due to the stringent safety regulations that were imposed in the entire state, the university was also running at its lowest occupancy levels for several months in the year 2020. To determine whether there is a noticeable difference in the consumption levels between the months when a lockdown was imposed and the months when the university was operating at normal occupancy, the data collected during these different time periods can be compared.

For the data collected from the 10 buildings in the university, 7 buildings - Chemie, Kita Hokido, OH12, EF42, EF40, EF40a and Erich Brost Institut - provide data that covers data consumption for the year 2020. The remaining bukidings - OH14, Großtagespflege and HG II - only have data going back to 2021, which renders it hard to observe the pandemic's effects on them.

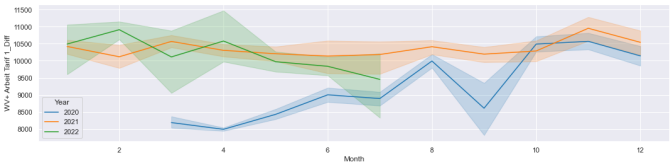


Figure 13: [NS] Electricity consumption for Chemie for the years 2020-2022

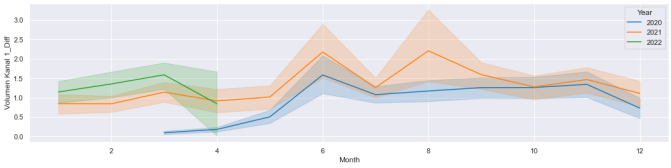


Figure 14: [NS] Water consumption for Kita Hokido for the years 2020-2022

The figure 13 shows the consumption of electricity for the Chemie building for the years 2020, 2021, and 2022. Upon inspection, the consumption for the year 2020 is fairly low during March to October and this can be contributed to the lower occupancy of the building during the lockdown. In October, the consumption slowly increases and for the succeeding years the consumption levels are within the same range. The consumption of water shown in the figure 14 for the building Kita Hokido shows a similar trend. Although the three years' patterns for the consumption are similar, the levels of consumption in 2020 are significantly lower.

For HG II, where the data for the year 2020 is not available, the consumption of heat energy seems to decrease during the summer and increase in the winter. But this data is not sufficient, to draw

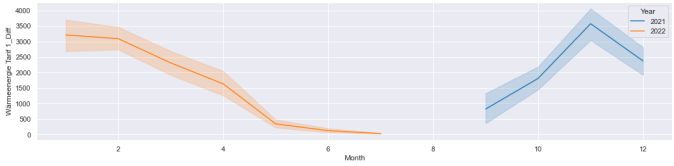


Figure 15: [NS] Heat energy consumption for HG II for the years 2021-2022

a contrast for the heat used during the lockdown period. A graph depicting the heat energy consumption for HG II is provided in figure 15.

4.7 Public Holidays

[NP]In this section, we will be comparing the effects of holidays on the consumption of resources with any other regular days. Technically in the context of resource consumption, holidays are referred to as days of the year for which the consumption pattern is most likely to be altered. This section is intended to analyze to what extent holidays differ from normal regular days. In addition, we will also compare all the holidays among themselves in order to find out how the consumption pattern on the particular holiday differs from the other holidays.

To start with, all the public holidays are categorized into two groups: 1). Holidays fall on weekdays, and 2). Holidays fall on weekends and are analyzed separately. This is due to the expectation of different behavior between the two above-mentioned groups. Also, this categorization will be helpful to answer the most obvious question: " Can public holidays be considered equivalent to weekends?" since the nature of both (public holidays and weekends) is intuitively the same in terms of behavior.

While analyzing the resource consumption of different buildings, the most significant result has been noticed for building OH14 which is discussed in this section.

Comparison of holidays among themselves:

As per figures 16 and 17, for public holidays on weekends, Wärmeleistung(heat output) and water flow rate (Durchfluss) for Sensor 1 have been observed around the level of less than 10 kW and between 0 to 0.5 m^3/h respectively for most of the holidays on weekends. However, there is a sudden spike(up to 140 kW and 2.5 m^3/h respectively) in the heat output and water consumption level starting from the early morning Christmas(2021-12-25 midnight around 3 A.M.) which remains stable at 60 kW and 1.0 m^3/h throughout the rest of the day, and the same level was constantly maintained for the entire duration of the next day (2021-12-26) as well.

Similarly, as per figures 33 and figure 34 in the appendix, for public holidays on weekdays, Wärmeleistung(heat output) and water flow rate (Durchfluss) for Sensor 1 has been observed around the level of less than 10 kW and between 0 to 0.5 m^3/h respectively for most of the holidays accept some noticeable fluctuations on All

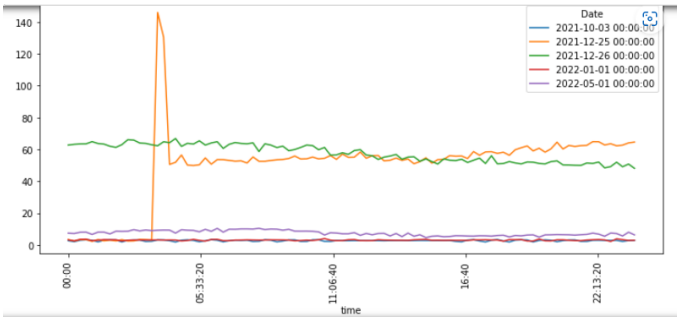


Figure 16: Heat energy consumption throughout the day on public holidays(weekends) for OH14

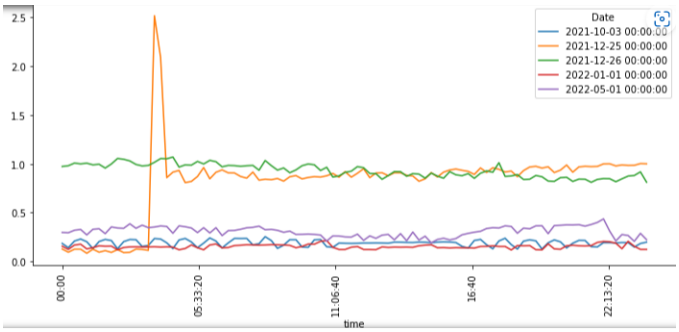


Figure 17: Water consumption throughout the day on public holidays(weekends) for OH14

Saints Day (2021-11-01) after 8 P.M. up to 120 kW for Wärmeleistung and 2.5 m^3/h for Durchfluss.

Overall, we can notice by comparing figures from both groups for OH14 that on public holidays, Wärmeleistung (heat output) tends to be stable at the level of less than 10 kW and the water flow level (durchfluss) is fluctuating between 0 to 0.5 m^3/h irrespective of day of the week accept some exceptions in both groups. Which ultimately implies that public holidays on weekends are the same as holidays on weekdays. Therefore, the grouping based on assumption that consumption on holidays falls on weekends might differ from that of the public holidays on weekdays doesn't make any sense in the end.

Comparison of public holidays with regular days:

Looking at the figures 35 and figure 36 in the appendix, it is evident that heat output and water flow rate for 25^th and 26^th December are almost similar to that on any regular day which is between 50 to 60 kW Wärmeleistung for and 0.8 to 1.0 m^3/h Durchfluss. Which is also reflected in figure 16 and 17.

Now, this leads to our final question, are public holidays equivalent to regular weekends in the context of resource consumption? To answer this question let's compare summary statistics for Wärmeleistung and Durchfluss of OH14 for sensor 1 and sensor 2 in tables8(for 103 weekend days) and 9 (for 11 holidays) from the appendix. As we can see from the comparison, the mean and standard deviation of both quantities are somewhat nearby for weekends and holidays. However, we refrain from commenting on anything regarding the equivalence without performing a statistical test of some sort.

4.8 Denoising

[AT] We use a denoising technique to get time series data with uncorrupted signals instead of model data with noise. De-noising is a data preprocessing step for the time series data. In our data understanding task, we are using one of the denoising algorithms known as the Kalman Filter. The Kalman filter is an iterative algorithm or precisely we can say it's a type of optimal estimator that allows us to estimate the important signals in the dataset. The Kalman filter belongs to the family of state-space models. Kalman filtering is a technique that produces estimates of some unknown variables using the measurements that have been taken over time. The Kalman filter is based on two assumptions, the first assumption is that the actual values are not perfect, they might have some errors and the second assumption is that we get the actual value from the measurement based on the prior information and the error rates.

We are using the Kalman filter for efficient smoothening of the time series data. Kalman filter applies smoothing and denoising after receiving the time series input. Figure 18 shows the plot after applying the Kalman filter to the column Vorlauftemperatur for the Großtagespflege building. The blue line is the observed input of the dataset and the orange line is the estimated values that are achieved through the Kalman filter. The time series after smoothing has the same pattern as the original data with constant noise reduction.

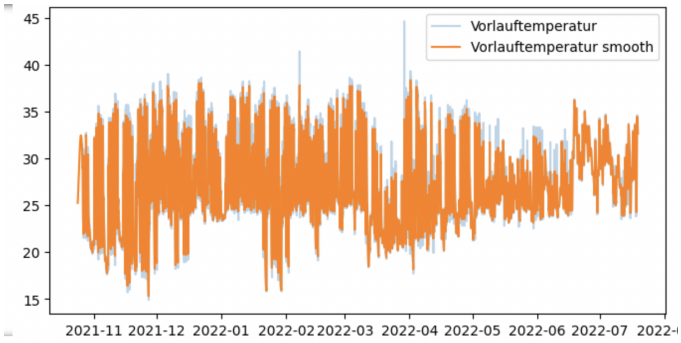


Figure 18: Denoising of feature Vorlauftemperatur for the Großtagespflege building

The process of denoising is essential only when we have data generated from different sources. In this experiment, the given dataset does not have many columns which have noise.

4.9 Summary Statistics per sensor

[AB] Summary statistics per sensor are a common way to gain insights into large datasets collected by sensors of several buildings.

The data collected by these sensors can be complex and difficult to analyze without appropriate statistical techniques. Summary statistics per sensor involve calculating a set of statistical measures for each sensor in the network. These measures can include the mean, standard deviation, maximum, minimum, and range of the data collected by each sensor. By calculating these measures for each sensor, it is possible to gain insights into the behavior of individual sensors and to identify potential issues or anomalies in the data.
Summary statistics per sensor can be used to identify patterns and trends in the data, as well as to detect outliers and anomalies. They can also be used to compare the performance of different sensors in the network and to identify areas where improvements can be made. In addition, summary statistics per sensor can be used as input to more advanced statistical techniques, such as machine learning algorithms, to make predictions or to identify correlations between different sensors.

Wärmeenergie Tarif 1(Heat): Erich-Brost-Institut building has the highest average ($1.91 \cdot 10^8$) Wärmeenergie Tarif 1, suggesting that it is using more heat energy than the other buildings. Chemie building has the second-highest average, followed by Chemietechnik and Grosstagespflege building has the lowest average. This suggests that Kita2 is using less heat energy(Wärmeenergie Tarif 1) than the other buildings. In Fig 19, these average values for each building are depicted.

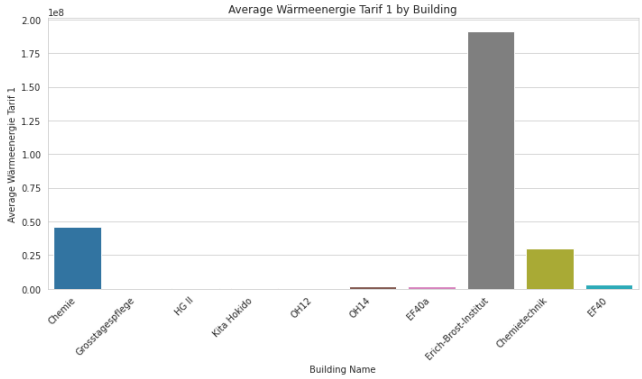


Figure 19: Average Warmenergie Tarif for all buildings

Wärmeleistung(Heat): Erich brost instiut building has a maximum Wärmeleistung value (2.86×10^5) that is much higher than the mean, indicating that it has periods of high heat output that are significantly higher than the average and it has a high variability (1.85×10^4) in heat output, suggesting that it has periods of both high and low heat output(Wärmeleistung). In Fig 20, box plot visualization of Wärmeleistung for all buildings is illustrated.

Fehler Flags (Electricity): All the buildings have no values for Fehler Flags except the building OH12 where there are few observations with value 0. This requires investigating these observations further to understand the root cause of the errors and to ensure that the data is accurate and reliable.

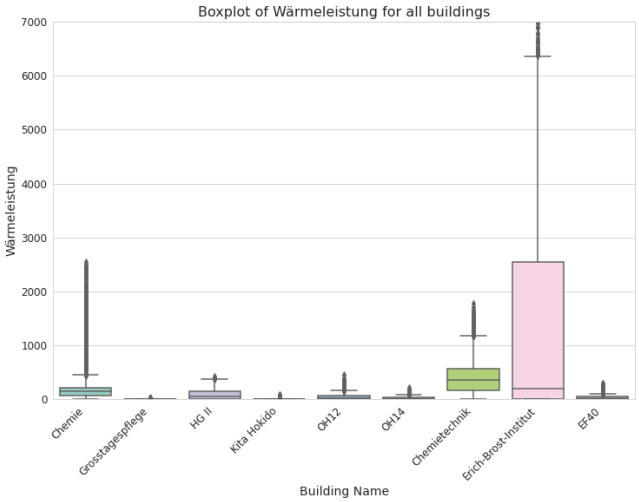


Figure 20: Box plot visualization of Wärmeleistung for all buildings

Vorlauftemperatur (Heat): Chemietechnick building have a maximum Vorlauftemperatur value (100.62) that is higher average than the other buildings but Chemie building has highest value (140.8) observed. It means both buildings have high heat flow values, but "Chemie" building has the highest value among all the buildings. The fact that the maximum heat flow in "Chemietechnick" building is higher than the average heat flow temperature (Vorlauftemperatur) in other buildings could suggest that this building has a higher demand for heating or cooling compared to other buildings.

Volumen Kanal 1 (Water): Both HGII (135.6) and Grosstagespflege building (5.25) have the lowest mean in Volumen Kanal 1 (Water), suggesting that the usage of water is less than the other buildings.

WV+ Arbeit Tarif 1 (Electricity): Grosstagespflege building has a mean WV+ Arbeit Tarif 1 with a value of 200.18 (Electricity) that is much lower than the other buildings, suggesting that it may consume less electricity.

BV+ Arbeit tariflos (Electricity): Except building HG II, Kita Hokido, OH12 and OH14, BV+ Arbeit tariflos (Electricity) is present in all building. In Chemie building, it is present in only one observation (with value 612.58) , while in EF40a building, higher average value is observed with value 2077.064. This suggests that these two buildings have a different tariff structure for electricity than the other buildings.

In summary, these overviews of the sensor data might offer insightful analysis before delving into the model deployment. A better model can be created to forecast anomalies by studying the data first.

5 DATA PREPARATION

[AS] This section describes the common preprocessing steps and column naming scheme.

Our data is stored in tabular format files; for certain buildings, the data is contain in two files. The files are first concatenated to create one single file for each building. In light of the fact that the provided column names are in lengthy format, we then develop distinctive, approachable column names. Thus, we specify a constant structure for naming as the first digit of the OBIS Kennzahl (which indicates that the feature pertains to heating or electricity usage), followed by the building code and the feature name's abbreviation. For example, Wärmeenergie of the $OH12$ building is written as $6\_CN45\ 11\ 01\ 01\_Wtotal$. Throughout this case study, we use this nomenclature. Duplicate timestamps are removed, and the modified files are saved in parquet format.

Furthermore, we perform the following steps to clean the data:

- We remove those features that are duplicates, comparable to other features, have only null values, or have 0 in the OBIS Kennzahl since these characteristics are not essential for our task.
- With our knowledge of the data and how some feature values change over time, we take the row-wise difference of numerical variables.
- Afterward, we create time and date-related exogenous variables like: day, normalized time stamp, day of the week, and hour of the day. We also introduce outside temperature as a new variable.

After cleaning the data, we define our target variable for this case study, marking the North Rhine-Westphalia region's public holidays as 1 (anomaly), while the remaining public holidays are marked as 0. (No anomaly). To use the data later, we finally divided it into train and test sets and put it in a parquet file. Except for the HG II and Grösstagesflege buildings, we chose 2021-10-01 as a split date for the training and testing sets, and for these two buildings, we used 2022-04-01.

Missing values & data normalization: We interpolated the missing values and ormalized the data by using min-max normalization for TCN and LSTM. For DGHL, the pre-preprocessing is explained in the next section.

5.1 DGHL specific data preprocessing:

[SP] *Feature-dimension wise normalization*: Standard normalization is performed to make the values of each feature have a unit variance and zero mean. In order to compute a new data point x_{new}, the following formula is used after determining the mean and standard deviation for each feature,

$$x_{new} = \frac{x - \mu}{\sigma}$$

where x is the original data point, μ is the mean of each feature and σ is the standard deviation of each feature.

Masking: [SP] An array called train mask, which is a combination of 0 and 1, is created. The shape of this mask is same as the training data. 0s in this mask represent the corresponding values that are missing in the training data whereas 1s represent the values that are present in the training data. Missing values or the null values in the training data are replaced with zeros, and both training data and train mask are given to the DGHL model for training.

5.2 Data Preparation of Synthetic Data:

[AT] We have injected the point and contextual anomalies in the building dataset. We have injected two-point anomalies and three contextual anomalies on dimensions that have few or no true anomalies. Before adding the anomalies, we normalised the dataset and implemented masking. Here, we have shown an experiment of anomaly injection in the Chemie building. We have selected three dimensions in a Chemie building with few or no anomalies and we have injected the anomalies in those dimensions randomly. Figure 21 shows the three dimensions with the injected anomalies. The orange ones are the point anomaly with the location points and the red ones are the contextual anomalies. The procedure to insert contextual anomalies is that we first randomly take a window and replace the values of dimension under that window with the values of another dimension time window. To inject a point anomaly, we randomly take a sequence of 200 timestamps known as location points and inject a spike in this set of location points.

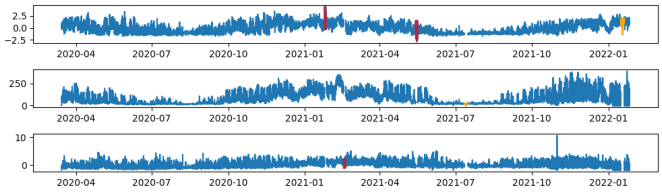


Figure 21: Synthetically generated data for Chemie building

The method for contextual and point anomaly injection also set the labels to 1 for the timestamps in which the anomaly is injected. We have created a plot that outputs the corresponding labels of the features affected during anomaly injection. The plot for feature-wise labelling of the dimensions is shown in figure 22. The timestamps that have true anomalies are included in this labelling in advance.

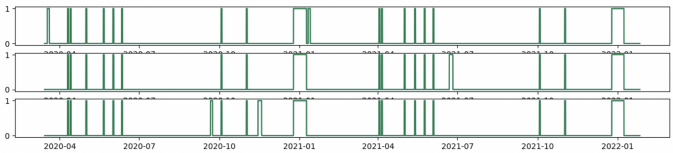


Figure 22: Feature-wise labelling for Chemie building

6 EXPERIMENTS

[KT] By outlining each dataset used in the experiment, we want to provide an overview of the data. In addition, we will summarize the evaluation parameters used when presenting the model-specific results.

6.1 Experiment and results for LSTM

[NS] The goal of this experiment is to forecast the anomalous points of the time series data collected from 10 different buildings of the university and to assess the accuracy of the LSTM autoencoder model. This is further expanded to incorporate various model structures and scoring schemes and a contrast is drawn for the results obtained.

Experiments: To conduct these experiments, we have considered a total of 13 datasets - a dummy dataset, data from 10 different buildings in the university and two synthetic datasets which are based on the data from the buildings Chemietechnik and EF42. In order to execute the experiments on the datasets, about 70% of each dataset is used for training the model, and the remaining 30% is taken for testing the model. For each building, the time-series data used as input for the model is recorded at an interval of 15 minutes, thus yielding a total of 96 values per day. We presumptively label public holidays, for the time period March 2020 through July 2022, as anomalies, i.e., all 96 values of a public holiday are viewed as anomalous data.

For the hyperparameters, we choose to consider overlapping sequences of length 96 as the window size, a batch size of 24 and for the hidden layer, we consider 128 units for buildings with features greater than 10 and 64 units for those with less than 10 features. All the hyperparameters were selected manually by a random trial and error process. The experiment is first done on an autoencoder with a single LSTM layer, each for the encoder and decoder, and a linear output layer with these chosen hyperparameters. For the input time series, the model generates a reconstructed output and an error vector for all features is calculated using L1 loss for each time stamp in the data. This error vector is then used to find the anomaly score at each time stamp and here the anomaly score is set as the multivariate log pdf function. The model has been trained by observing the epoch-loss graph and the process is stopped when a convergence of this graph is observed, which means it is stopped at a point when the increase in epochs does not affect the training loss further. The results of this experiment for all the datasets under consideration are given under the column 'Base Setting' in Table 6.

For the succeeding set of experiments, the hyperparameters are not changed and take on the same values as given above. The model structure of the LSTM autoencoder is manipulated further and applied on the datasets. Firstly, the number of LSTM layers in the encoder and the decoder of the model are increased to 2 and the results are observed and recorded under the '#2 Hidden Layers' column of the Table 6. For the next two sets of experiments, the LSTM model is made bidirectional in the first and the number of linear output layers is increased to 3 in the second and the results are respectively recorded under the columns 'Bidirec. LSTM' and '#3 Output Layers' of the Table 6. Further 3 experiments are conducted by changing the anomaly scoring functions and using the L1 loss to obtain the reconstruction error vectors. The scoring functions employed here are the Mean function, Max function and the Mahalanobis distance and the results of these are recorded under the columns 'Mean', 'Max' and 'Mahalanobis Dist.' of Table 7.

Results: [RP] Upon completion of the experiments, we generated various graphs like the epoch vs loss graph, plots for the reconstructed output sequence and the error that occurred in this reconstruction. Additionally, the confusion matrix, the ROC curve and an anomaly score plot for the true and predicted label are plotted for each test data and the evaluation measures like precision, recall and F1 score are calculated. The latent space plot was carried out in order to identify the compressed representation of the data by utilizing the t-SNE method.

To classify a day as an anomalous day, we selected the maximum value from the 96 anomaly scores available each day, which means that we ultimately have one anomaly score per day. With these anomaly scores, a threshold is set at the 90th percentile any day with an anomaly score greater than this threshold is considered an anomalous day. This eventually means that if at least one timestamp is anomalous then the associated day can be considered to be an anomaly.

When examining the reconstructions produced by our model, it is apparent that most of the features are poorly reconstructed, with the exception of the buildings containing the features like Temperatur differenz, Durchfluss, Volumen Kanal, Wärmeleistung, WV+Arbeit Tarif 1 and Wärmeenergie Tarif, where the model is able to reconstruct better as these buildings have lesser missing values. One of the possible factors could be the high number of missing values, which is causing the interpolation of the missing values to perform less effectively than expected. Figure 27a shows an example of correctly reconstructed output and error of the feature WV+ Arbeit Tarif 1 for the building Kita Hokida, which has clean data and very few missing values. Tables 6 and 7 give an overview of the evaluation matrices for the experiments done. We can observe that for the buildings Chemie and Erich Brost Institute, we get higher values of AUC scores. Using the mean as the scoring function an AUC score of 82% is obtained for the Erich Brost Institute building and for the Chemie building, the highest AUC score is produced when using when the model structure with 2 LSTM layers in the encoder and decoder. Tables 6 and 7 also show that the model does not classify anomalous days accurately and generates a low F1 score for all the experiments done on the remaining buildings.

[AB] The tsne plot shown in Figure 23 depicts the attempt to obtain a lower-dimensional representation of the EF40 building dataset. The plot shows that no significant pattern was learned, which could be attributed to several factors, such as the network structure, data quality, choice of activation functions, and optimization algorithm used. One possible reason for the lack of pattern could be that the labels were assigned manually.

[RP] However, upon an investigation of the results generated by the model for the dummy dataset, we can see that the true

anomalies have been captured by the model and predicted as an anomalous data point. The tests involving the scoring functions consistently show that the model gives an F1 score of over 60% and in the experiments which involve the manipulation of the model architecture, increasing the number of hidden layers and using a bidirectional LSTM model gives the highest F1 score of around 73%. For all the sets of experiments carried out, the AUC score steadily has a value of over 95%. In the case of the datasets Chemietechnik Synthetic and EF42 Synthetic, the model again fails to predict the anomalies and from Tables 1 and 2 and it is clear that the F1 score and AUC score are also considerably low and thus do not give much insight about the performance or learning capability of the model.

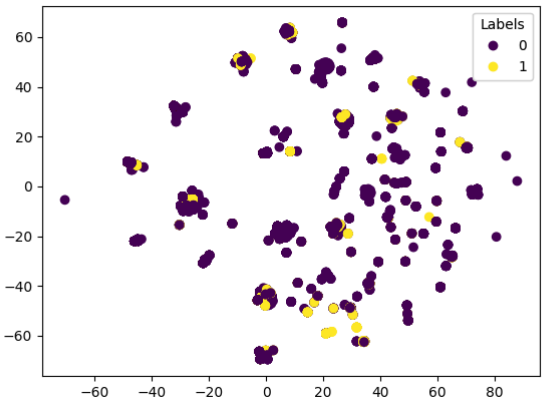


Figure 23: Latent space for building EF40 using LSTM.

Taking into consideration the fairly poor evaluation matrices we received from the model, we still tried to find which model structure or anomaly scoring function has a better performance over the rest and came to the conclusion that increasing the number of hidden layers to 2 has a slightly improved chance of predicting anomalies correctly. However, it should also be noted that the results generated by this model structure still fail to predict the anomalies in several datasets.

6.2 Experiment and results for TCN

Experiments:

[MK]This section illustrates the exact anomaly detection method in the context of the given time series. To start with, all buildings have long-term data (mostly between the years 2020 to 2022) of multiple days as 96 timestamps (records) per day containing the different number of features. As an output, the model is generating the reconstruction of the series (Rec) for each feature which is the same length as an input. Now, for the purpose of the evaluation, we are considering original input data(test) along with the reconstruction output(Rec) from the model and applying the scoring functions mentioned in section 3.2.3 which in terms generates a single score per timestamp for each scoring function. After obtaining multiple scores(as per different scoring functions) for each timestamp for all the records in the time series, we proceed toward deciding the threshold to classify the anomalous timestamps. The hyperparameters for the model include the number of epochs, sequence length, number of filters, filter size, and latent sample rate. The number of epochs was set to 100, sequence length of 96 was taken without overlapping. The number of filters, "nb_filters" (i.e., the number of features the model will learn to detect) used in the convolutional layers of the model and "filters_conv1d", the size of the filters used in the convolutional layers of the model are set to 96. "Latent_sample_rate", the rate at which the model will sample the latent representation of the input data is set to 4.

However, before deriving the threshold, it is important to make sure that all the scores derived from the same scoring function are on the same scale so that the process of deriving the threshold is efficient and more accurate. For that reason, the min-max method is performed for the normalization of the scores which ultimately maps the scores between the range of 0 and 1 as mentioned in section 3.5.1. After normalization of the scores for each timestamp, the scores are aggregated on a higher level which is on a daily basis. As in the given data, $96 timestapms = 1 day$. Therefore, we aggregate the batch of consecutive 96 labels by taking the maximum value of the batch as the anomaly score for the day. Finally, the threshold is derived based on the assumption that 10% of timestamps from the total timestamps are anomalous. Therefore, we sort the scores in ascending order(low to high) and pick the score equivalent to the 90th percentage from the sorted scores which is ultimately the value of the 90th percentile as a threshold. After sorting the data in ascending order, we can derive the P^{th} percentile for N number of total observations by the following equation:

$$\text{Index} = \frac{p}{100} \times (N + 1)$$

[NP]Here *index* is the position(or index) in the sorted score such that p% (90% in our case) records falls below the *index* and $(100 - p)$% (10%) records lie above the same. After deriving thresholds for each different type of score calculated from different scoring functions, all the timestamps are classified in labels as an anomaly (label=1) or non-anomaly(label=0) by comparing the score of the given timestamp with the threshold. The function which maps anomaly score to label is as follows:

$$\text{f(x)=} \begin{cases} 0, & \text{score}\_i[x] < \text{score}\_i[index] \\ 1, & \text{score}\_i[x] \geq \text{score}\_i[index] \end{cases}$$

Here the value of *f(x)* indicates the value of the label corresponding to timestamp x, score_i[x] indicates the score corresponding to the timestamp x calculated using the scoring function i where possible values of i = Mahlanobis distance, Mean score, L2 and L1 score as mentioned in section 3.2.3 and score_i[index] is the threshold value for the scoring function i on position $x = index$ for a sorted list of the scores.

In the final step, after assigning the labels, we compute some evaluation matrix and measures in order to determine which score yields the final labels nearest to the actual labels for all the buildings. As an evaluation matrix, the confusion matrix is taken into account and as evaluation scores, F1 scores and AUC (Area Under the Curve)

scores are considered as mentioned in the section 3.5.2. Latent space visualization was also done as explained as referred in section 4.4.

Results:
[NP]In this subsection, we will be discussing the results obtained by applying the methodology mentioned in subsection 1 6.2 to the test data of different buildings. We have also considered the dummy data which consists of the sin wave intentionally injected with some random anomalies in between in order to test the efficiency of the model in a controlled environment. The table 5 illustrates different scores derived for different buildings using the methodology in section 6.2

As we can see from the comparison, the overall F1 scores for anomaly labeled by mean scoring functions perform slightly better than the rest of the scoring methods. However, for dummy data, the F1 scores obtained by the labeling score of the L2 norm (0.86) are better than the mean score (0.81). From the table 5, it is evident that for all the buildings, anomaly labeling using L1 and L2 scoring functions yields almost the same F1 scores. F1 scores derived by the labels from L1 or L2 norm are slightly better than mahlanobis distance scoring for most of the buildings except Chemie, Kita Hokida, Tagespflege, and OH14. For synthetic data, all scoring functions perform more or less the same. It is important to note that, the F1 scores of all functions for synthetic data for building EF42 are noticeably less than the original data however, the case is completely opposite for the Chemie building.

Comparing all F1 scores(from different functions) obtained for the dummy data with that for the rest of the buildings, we can notice a significant difference. The F1 scores of the model for the dummy data are much higher than the F1 scores for buildings. One interpretation of the same could be that, while our model is good at reconstruction, it underperforms for buildings, possibly due to the poor quality of the building data (some of the building's data contains too many missing values). Another reason behind the lower scores for building data could be, the lack of proper labeling technique as, we are labeling the records solely based on the assumption that all holidays are anomalies, however, as discussed in subsection 4.7. weekends tend to exhibit behavior that is comparable to those of holidays, which the model can mistakenly classify as an anomaly (consequently increasing false positives).

[MK]As far as AUC scores are concerned, the values of AUC scores for all buildings range from slightly greater than 0.5 to around 0.3 except for a few exceptions which in terms indicate that the model is performing the same as random guessing or even worse than that which is rather obvious. However, we are getting the perfect AUC score (= 1) for the L2 scoring function for the dummy data which also exhibits the highest F1 score (0.86) out of all. This indicates that for the labels assigned using L2 score, the model is best able to separate actual anomalies from non-anomalies for dummy data.

Figure 24 visualizes latent space using t-SNE for building EF40. The plot appears to lack any significant patterns or useful information that can be derived from it. One possible explanation for this could be the manual assignment of labels, which may have contributed to the lack of discernible patterns in the visualization.

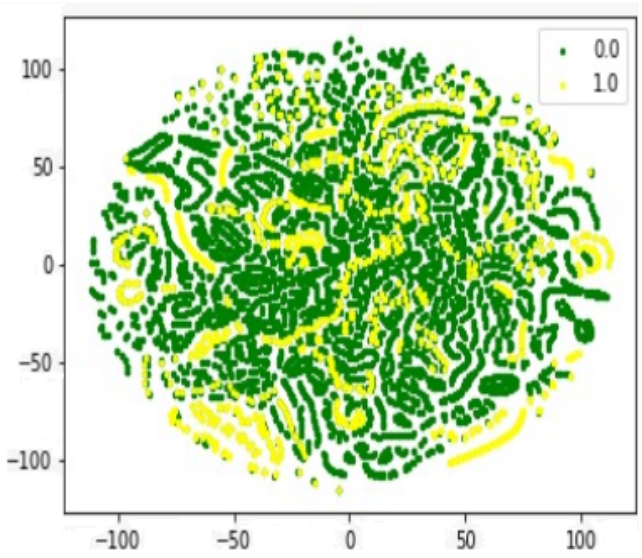


Figure 24: Latent space for building EF40 using TCN.

6.3 Experiment and results for DGHL

[KT] The general settings for the datasets of each building, namely, hourly dataset, combined dataset, and synthetic dataset will be shown in the following subsections, along with a list of the applied hyperparameters. We mask each missing observation using the masking approach described in section 5. After that, we trained our model using the provided dataset. Consequently, we can make all features the same size without downsampling or interpolating. Furthermore, a summary of the outcomes from training our model on the various datasets is provided. In order to evaluate our model for each building, we will display and contrast the Recall, Precision, F1- and AUC-Scores.

Occlusion experiment: [SP] Figure 28 in Appendix, represents the occlusion experiment on *Kita Hokido* building. The blue lines represent the actual values and the orange lines represent reconstructed values. The shaded gray region in the background corresponds to missing values. DGHL is able to reconstruct the observed values, even though some of the values are missing. The anomaly score is calculated using the observed features, and the missing data is recovered.

Experiment results with Building Data [KT] In the following Table 4, we look at the buildings where 15-minute frequency is recorded. We will explain and evaluate the results after training the model. Beginning with EF40, there are 11 158 observations in the test set and 54 365 in the train set. The test size for OH12 is 22633 and the training size is 32 490. The table below lists further details regarding the quantity of tests and training size.

The default Hyperparameters were modified during modeling our training set, which is listed in the appendix in Table 11. Due to our 15-minute frequency, we have a window size of 96, which covers one day. Similarly, the filter multiplier for Convolutional Net was equal to 16. In order to get non-overlapping data, the Step

Table 4: Training and Test Size of building data

	Training Size	Test Size
Chemie	71832	10376
Großtagespflege	15322	10379
HG II	18377	10379
Kita Hokido	54364	17766
OH12	32490	22633
OH14	8401	27847
EF 42	54365	11158
EF 40	54365	11158
EF40a	54748	27843
Erich-Brost-Institut	54747	10776

size was modified to 96. Regarding hyperparameter optimization, we found that boosting our model's learning rate to 5e-3 and batch size to 25 resulted in better learning. We continued to perform the standard training phase of 1000 iterations.

Volumen Kanal, Durchfluss, Wärmeenergie, and Wärmetarif are feature dimensions of particular interest in relation to our anomalies because, unlike other features, our model can better reconstruct these features. When comparing our findings to the other buildings, we can see that Building EF42 has less abnormalities. Further, we included certain exogenous features, such as Date, Day, Day of the week, Hour of the day, Normalize Timestamp, and outside temperature to see if our model had any impact on identifying anomalies. After doing multiple experiments for the various structures, we discovered that we could disregard the exogenous properties because they do not affect the model's learning ability. Looking at the reconstructions generated from our model, it is noticeable that the reconstruction is generally poor for most of the buildings, which is evident from the example below for Building HG2 (see Figure 38). One of the possible causes might be the masking of the missing values, which is not performing well as the bench dataset due to the number of missing values.

Since the Office buildings are quite similar to each other, we anticipated having the best reconstructions in the Office Building dataset, which is also demonstrated by looking at the reconstruction plots for building EF42 (see Figure 37). When comparing the F1 scores, we notice that EF40 and EF42 have the highest F1 scores, respectively, 0.20 and 0.12. The building HGII has the lowest F1-score of 0.02 and also an AUC score of 0.20. Table 12 in the Appendix lists the evaluation parameter values.

Experiment results with hourly buildings data: [AS] In the following section, we evaluate the performance of the DGHL model on buildings with observations recorded hourly for some variables.

In the above section, we used all the features recorded every 15 minutes. But for some buildings like EF40a, Erich Brost, and Chemie, features like wärmeenergie, Durchfluss, Vorlauf- and Rücklauftemperatur are recorded hourly. With 17 features for the Chemical building, we have 71832 observations for training and 10376 for testing for this experiment. Since the DGHL model was used for training, there is no need to downsample or extrapolate the features. For capturing this difference of features recorded at a different frequency, we trained our model on different hyperparameter settings. Here, we used learning_rate = 0.0001, and batch_size = 15, with the same window size and steps which are equal to 96. We used early stopping during model training, which ended when the epochs stopped impacting the validation and training losses. As a result of this, we achieved an F1 and AUC score of 0.11 and 0.50, respectively, for the Chemie building. For EF40a and Erich Brost, we achieved an f1 score equal to 0.001 and 0.22, respectively.

Experiment results with combined building data: [KT] In the following section, we evaluate the performance of the DGHL model on the combined building data.

With this strategy, we generated our data using two techniques. First, we considered all the observations from related buildings into a single data building, resulting in more observations but duplicated timestamps per day. The idea for this approach was to test if our model is impacted by the number of observations and whether learning is enhanced. Our second approach is to avoid duplicated timestamps by alternating over the three building datasets so that the model does not know from which building the data is coming. Our strategy is to see how our model would respond if it were unaware of the source of the data. To increase our model's learning, we trained it with more iterations and applied early stopping.

Chemietechnik building

- Combined building
  This dataset has a recall value of 0.46 and an accuracy value of 0.29. In addition, the F1 Score is around 0.36, which is higher than the values from the common building dataset but still below the F1 Score of the Chemietechnik Building 2, which has the highest score among all the Datasets. The AUC-Score is 0.55. In general, our model was able to reconstruct most of the features, although it struggled for few features (eg. Rücklauftemperatur 39).
- Single Building
  Initially we would like to describe how we created the single Building Dataset. The training dataset of Building two starts from 2020-03-14, while building one and three is recorded from October 22, 2021. Because of this reason, we used the training set of building two until the 22nd of October and alternated Building two and three afterward. This was only the case for the training dataset. Throughout testing, we used alternating from all three buildings. The results for the evaluation parameters show that our model struggled to reconstruct and learn from this dataset (see Figure 40, 41). The F1 score and AUC fell by 6% and 3%, respectively. The recall is 0.38, and the precision value is 0.24. After comparing the results, we found that the first strategy performs better in terms of the evaluating parameters, indicating that our model is more effective and able to learn from more data.

Office building

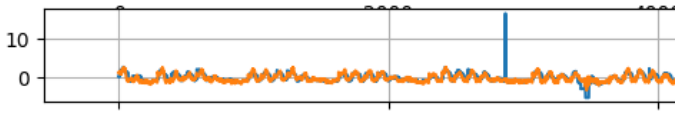


Figure 25: Reconstruction on Building 42 on feature WV+T1

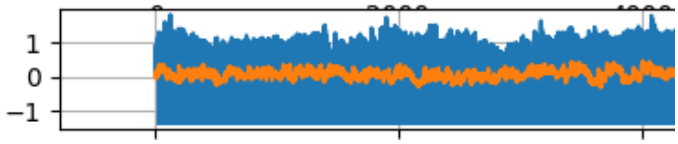


Figure 26: Reconstruction on combined Chemie on feature Rücklauftemperatur

- Combined building and Single Building: We followed similar approach for Office Building Datasets as well. We selected the most similar buildings, which are EF40, EF40a, and Erich Brost Institute. For the combined Office Buildings we have an F1-Score of 0.15 and AUC-Score of 0.58. While for the single Building approach we got F1 and AUC Score as 0.15 and 0.56 repectively.

Experiment results with synthetic data: [AS] In the following section, we evaluate the performance of the DGHL model on the synthetic label data.

For this case study, we consider public holidays as a target variable, but in this section, we decided to create synthetic points and contextual anomalies and train our model on this new data to check our model performance. Here we trained our DGHL model using standard hyperparameters, except the learning rate equal to 0.003 and batch size equal to 20. From the results of other models, we identified one or two buildings with few anomalies and for those buildings, we imputed synthetic anomalies in some features. We considered Ef42 and Chemie technique buildings in this case study and created synthetic data. For the Ef42 building, we considered features like Wärmeneergie, Wärmeleistung, and WV+Arbeit Tarif1. We observed that using synthetic data, our F1 and AUC scores improved by 0.24 and 0.52, respectively, compared to previous results. We considered features like Temperature Difference, Wärmeneergie, and Volumen for the Chemie technique building. For the training we have same number of observation in training and testing set as shown in Table 4. As a result, we observed that our F1 and AUC scores improved by 0.24 and 0.52, respectively, compared to previous results for Ef42.

6.4 Experiment conclusion

[AS] As a result of all experiments, the performance of all three models on the given datasets is not quite good. For the approach of masking null values, we can conclude that if the data have more null observations, then the reconstruction of masked data is not quite well. With both approaches of using public holidays as a label and synthetic labels, the model was not able to capture the true anomalies accurately as excepted. One of the main causes of a low F1 score is that there are not many labeled anomalies in the test data; this is because, when splitting the data, we see that we have only two or three days that are tagged as anomalies.

7 CONCLUSION AND FUTURE WORK

[AB] In this study, three separate models i.e. LSTM, TCN, and DGHL—were used to create a system for detecting anomalies in time series data. Data on energy consumption from TU Dortmund University buildings as well as artificial datasets were used in the experiments. To determine the most appropriate model, various scoring functions, distance measurements, and architectures were used. Also, the latent space was attempted to be identified using two distinct models, LSTM and TCN. AUC score, precision, recall, accuracy, and other evaluation metrics were utilized to evaluate the effectiveness of the models in spotting anomalies.None of the models, however, demonstrated promising outcomes in terms of identifying anomalies in the sensor data from all buildings. Nonetheless, both from a model and data viewpoint, there is still potential for development.

We propose training the models for longer iterations from a model perspective. By enabling the models to pick up on more intricate patterns and relationships in the data, this can enhance their accuracy. Window overlapping implementation can also help reduce information loss and provide a more thorough view of the data, improving results. To increase the performance of the models and to tailor them to the particular dataset, hyperparameter tuning can be utilized.

From a data perspective, we recommend introducing some true anomalies as labels. True anomalies are real-world examples of anomalous behavior that have been confirmed by domain experts or other sources. These examples can be used to train machine learning models to recognize and identify similar patterns in new data. By including true anomalies as labels in the dataset, the models can learn from diverse and realistic examples, leading to more accurate and reliable anomaly detection. True anomalies can be identified through various methods, such as manual inspection by domain experts, historical data analysis, or data-driven approaches. Adding true anomalies to the dataset can also help to reduce the number

a Reconstruction of WV+ Arbeit Tarif 1 for Kita Hokida

b Anomaly Score for Kita Hokida

Figure 27: Predictions from LSTM model with the base settings for Kita Hokida for Electricity

of false positives generated by the models, which can lead to more effective and efficient model.

[AS] For instance, as described above, the hypothesis of taking public holidays as a label does not hold. Therefore, instead of concentrating only on F1 and AUC scores, we took into consideration the false positives (observations that we marked as no anomalies, and our model predicted them as anomalies) and found that in Kita Hokida for the Wärmenergie feature, there is a contextual anomaly for the time period from 7th March to 9th March 2022. In the figure 27a, we can see that there is a clear peek at this particular time period. So, we marked these observations as 0 (no anomalies), and all three models predicted these days as an anomaly. Figures 42a, 42b, 42c provide another illustration of the same.

[AB] This can help improve the accuracy of the models by providing more diverse examples of anomalous behaviour. Reducing the number of null values in the dataset is also crucial for the models to learn from the complete and accurate data. It is also important to track missing timestamps to ensure the data is complete and accurate, as missing timestamps can lead to incorrect interpretations of the data.

We recommend adding the labels lecture-free and exam periods to future work in order to better understand how human activity affects the building's energy usage. This can assist facility managers in organizing and optimizing their energy use during certain times to lower expenses and consumption. Moreover, adding synthetic data for additional structures can broaden the dataset's diversity and enhance the models' generalizability. Finally, by utilizing the similarities between the attributes of similar structures, it is possible to augment the training data and enhance the performance of the models. This can aid in the development of an anomaly detection system for energy-efficient systems that is more precise and trustworthy.

REFERENCES

[1] Ethem Alpaydin. 2010. *Introduction to Machine Learning*. MIT Press, Cambridge, MA.

[2] T.W. Anderson. 2009. *AN INTRODUCTION TO MULTIVARIATE STATISTICAL ANALYSIS, 3RD ED.* Wiley India Pvt. Limited. https://books.google.de/books?id=1iF0CgAAQBAJ

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. https://doi.org/10.48550/ARXIV.1803.01271

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 35. IEEE, 1798–1828.

[5] Chris U Carmona, François-Xavier Aubet, Valentin Flunkert, and Jan Gasthaus. 2021. Neural contextual anomaly detection for time series. *arXiv preprint arXiv:2107.07702* (2021). https://arxiv.org/pdf/2107.07702.pdf

[6] Cristian I. Challu, Peihong Jiang, Ying Nian Wu, and Laurent Callot. 2022. Deep Generative model with Hierarchical Latent Factors for Time Series Anomaly Detection. 151 (28–30 Mar 2022), 1643–1654. https://proceedings.mlr.press/v151/challu22a.html

[7] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. 2018. Deep learning for anomaly detection: A review. *arXiv preprint arXiv:1812.01596* (2018).

[8] Y. Dodge. 2008. *The Concise Encyclopedia of Statistics*. Springer New York. https://books.google.de/books?id=k2zklGOBRDwC

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA.

[10] Saroj Gopali and Akbar Siami Namin. 2022. Deep Learning-Based Time-Series Analysis for Detecting Anomalies in Internet of Things. *Electronics* 11, 19 (2022). https://doi.org/10.3390/electronics11193205

[11] Han, Tian, Lu, Yang, Zhu, Song-Chun, Wu, and Ying Nian. 2017. Alternating back-propagation for generator network. 31, 1 (2017). https://arxiv.org/pdf/1606.08571.pdf

[12] Yangdong He and Jiabao Zhao. 2019. Temporal Convolutional Networks for Anomaly Detection in Time Series. *Journal of Physics: Conference Series* 1213, 4 (jun 2019), 042050. https://doi.org/10.1088/1742-6596/1213/4/042050

[13] Delanyo Kwame Bensah Kulevome, Hong Wang, , and Xuegang Wang. 2021. A BIDIRECTIONAL LSTM-BASED PROGNOSTICATION OF ELECTROLYTIC CAPACITOR. *Progress in Electromagnetics Research C* 109 (2021), 139–152.

[14] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. 2019. Multivariate anomaly detection for time series data with generative adversarial networks. *International Conference on Artificial Neural Networks* 42 (2019), 1–3. https://arxiv.org/pdf/1901.04997.pdf

[15] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. https://doi.org/10.48550/ARXIV.1607.00148

[16] Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2015. Long short term memory networks for anomaly detection in time series. *Neural Networks* 71 (2015), 204–212. https://doi.org/10.1016/j.neunet.2015.06.028

[17] Radford M Neal et al. 2011. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo* 2, 11 (2011), 2. https://arxiv.org/pdf/1206.1901.pdf

[18] Nijkamp, Erik, Pang, Bo, Han, Tian, Zhou, Linqi, Zhu, Song-Chun, Wu, and Ying Nian. 2020. Learning multi-layer latent variable model via variational optimization of short run mcmc for approximate inference. (2020), 361–378. https://arxiv.org/pdf/1912.01909.pdf

[19] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. https://doi.org/10.48550/ARXIV.1609.03499

[20] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. 2018. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters* 42 (2018), 1–3. https://arxiv.org/pdf/1711.00614.pdf

[21] Ankur A. Patel. 2019. *Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data.* O'Reilly Media, Sebastopol, CA.

[22] Makoto Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. *IEICE Transactions on Information and Systems* E97-D, 10 (2014), 2593–2600. https://doi.org/10.1587/transinf.2014EDL8201

[23] Markus Thill, Wolfgang Konen, Hao Wang, and Thomas Bäck. 2021. Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing* 112 (2021), 107751. https://doi.org/10.1016/j.asoc.2021.107751

[24] Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. *In Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), 681–688.

[25] Deutscher Wetterdienst and Monatlicher Witterungsbericht. 1998. Deutscher Wetterdienst. *Offenbach/Hamburg* (1998).

[26] David F. Wulsin, John R. Moffat, and James A. Dasch. 2010. Ensemble anomaly detection via feature subspacing. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 955–964. https://doi.org/10.1145/1835804.1835920

[27] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jie Tong Jing Bai, and Qi Zhang. 2020. Multivariate time series anomaly detection via graph attention network. *IEEE International Conference on Data Mining (ICDM)* 42 (2020), 3–4. https://arxiv.org/pdf/2009.02040.pdf

APPENDIX

Table 5: TCN Experimental results with Different scoring function

Datasets	Scoring Functions							
	Mahalanobis Dist.		Mean		L1		L2	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Dummy Data	0.79	0.93	0.81	0.79	0.79	0.90	0.86	1.0
EF40	0.16	0.44	0.19	0.39	0.24	0.5	0.24	0.5
EF40a	0.12	0.73	0.01	0.41	0.00	0.44	0.00	0.44
EF42	0.19	0.48	0.23	0.47	0.24	0.5	0.24	0.55
Erich Brost Institut	0.06	0.47	0.27	0.56	0.21	0.42	0.22	0.44
Chemie	0.12	0.54	0.00	0.5	0.00	0.51	0.00	0.51
Kita Hokida	0.18	0.49	0.13	0.54	0.12	0.53	0.12	0.53
Tagespflege	0.13	0.46	0.08	0.52	0.07	0.52	0.05	0.43
HG2	0.01	0.42	0.11	0.52	0.07	0.5	0.07	0.5
OH12	0.01	0.44	0.05	0.53	0.05	0.51	0.05	0.51
OH14	0.20	0.31	0.11	0.53	0.14	0.50	0.11	0.53
EF42 synthetic	0.08	0.55	0.05	0.4	0.07	0.53	0.06	0.53
Chemietechnik Synthetic	0.18	0.33	0.14	0.42	0.18	0.34	0.18	0.34

Table 6: [RP]LSTM Experimental results with Different Model Structures

Datasets	**Model Structures**							
	Base Setting		**#2 Hidden Layers**		**Bidirec. LSTM**		**#3 Output Layers**	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Dummy Data	0.67	0.97	0.73	0.98	0.73	0.96	0.65	0.97
EF40	0.09	0.41	0.09	0.61	0.11	0.52	0.22	0.66
EF40a	0.0	0.15	0.0	0.16	0.0	0.18	0.0	0.16
EF42	0.07	0.42	0.07	0.44	0.07	0.40	0.07	0.38
Erich Brost Institute	0.18	0.77	0.07	0.53	0.07	0.44	0.08	0.51
Chemie	0.20	0.59	0.0	0.61	0.24	0.57	0.20	0.58
Kita Hokida	0.0	0.34	0.0	0.42	0.0	0.39	0.0	0.36
Tagespflege	0.0	0.31	0.0	0.41	0.0	0.38	0.0	0.34
HG2	0.0	0.31	0.0	0.38	0.0	0.33	0.0	0.35
OH12	0.0	0.27	0.0	0.30	0.0	0.29	0.0	0.25
OH14	0.0	0.28	0.0	0.39	0.0	0.37	0.0	0.30
EF42 Synthetic	0.0	0.31	0.0	0.29	0.0	0.32	0.0	0.31
Chemietechnik Synthetic	0.1	0.63	0.1	0.64	0.1	0.62	0.0	0.59

Table 7: [NS] LSTM Experimental results with Different Scoring Functions

Datasets	Scoring Functions					
	Mean		Max		Mahalanobis Dist.	
	F1	AUC	F1	AUC	F1	AUC
Dummy Data	0.67	0.99	0.67	0.99	0.63	0.98
EF40	0.0	0.51	0.14	0.53	0.0	0.45
EF40a	0.0	0.23	0.0	0.18	0.0	0.21
EF42	0.07	0.32	0.07	0.31	0.07	0.28
Erich Brost Institute	0.2	0.82	0.0	0.75	0.0	0.71
Chemie	0.11	0.44	0.0	0.42	0.11	0.50
Kita Hokida	0.12	0.44	0.12	0.42	0.12	0.42
Tagespflege	0.0	0.21	0.0	0.15	0.0	0.46
HG2	0.0	0.42	0.0	0.43	0.0	0.36
OH12	0.0	0.27	0.0	0.11	0.0	0.30
OH14	0.0	0.52	0.0	0.55	0.0	0.38
EF42 Synthetic	0.0	0.31	0.0	0.35	0.0	0.33
Chemietechnik Synthetic	0.1	0.40	0.1	0.48	0.1	0.63

Table 8: Weekend summary statistics

	Durchfluss 1	Wärmeleistung 1	Durchfluss 2	Wärmeleistung 2
count	103	103	103	103
mean	0.33	11.71	4.39	25.05
std	0.22	14.9	0.97	3.01
median	0.23	4.09	4.00	24.5
min	0.069	1.57	3.05	20.56
max	0.99	62.85	6.99	30.37

Table 9: Public holiday summary statistics

	Durchfluss 1	Wärmeleistung 1	Durchfluss 2	Wärmeleistung 2
count	11	11	11	11
mean	0.35	13.73	3.94.39	24.24
std	0.26	19.8	0.79	2.37
median	0.27	6.09	3.85	24.5
min	0.15	2.68	3.04	20.88
max	0.92	60.85	5.89	27.74

Table 10: Outside temperature for OH12

Outside Temperature	Wärmeenergie Total	Durchfluss	WV+ Arbeit Tarif 1	WV+ Arbeit Tarif 2
lecture period winter	0.008	0.012	0.006	0.004
lecture period summer	0.44	0.05	-0.22	-0.29
winter break	0.25	-0.017	0.26	0.19
summer break	0.27	0.03	-0.25	-0.32

Table 11: Hyperparameter settings for DGHL

	default	Building Data	Hourly Data	Combined	Synthetic
Step size	256	96	96	96	96
ConvNet filters multiplier	32	16	16	16	16
Training step	1000	1000	1000	1000	1500
Batch size	4	25	25	25	25
Window size	64	96	96	96	96
Window_hierarchy	[1,4]	1	1	1	1

Table 12: Experiments results for DGHL

Features	F1-Score	AUC-Score
OH12	0.11	0.45
OH14	0.08	0.43
Kita Hokida	0.06	0.41
Großtagespflege	0.08	0.41
HG2	0.02	0.20
EF40	0.20	0.34
EF40a	0.08	0.51
Erich Brost	0.08	0.27
EF42	0.12	0.41
Chemietechnik Building 1	0.48	0.62
Chemietechnik Building 2	0.48	0.67
Chemietechnik Building 3	0.38	0.58
Chemietechnik combined	0.36	0.55
Chemietechnik singleBuilding	0.30	0.52
Office combined	0.18	0.60
Office singleBuilding	0.15	0.56
Chemietechnik synthetic	0.17	0.80
Ef42 synthetic	0.24	0.52

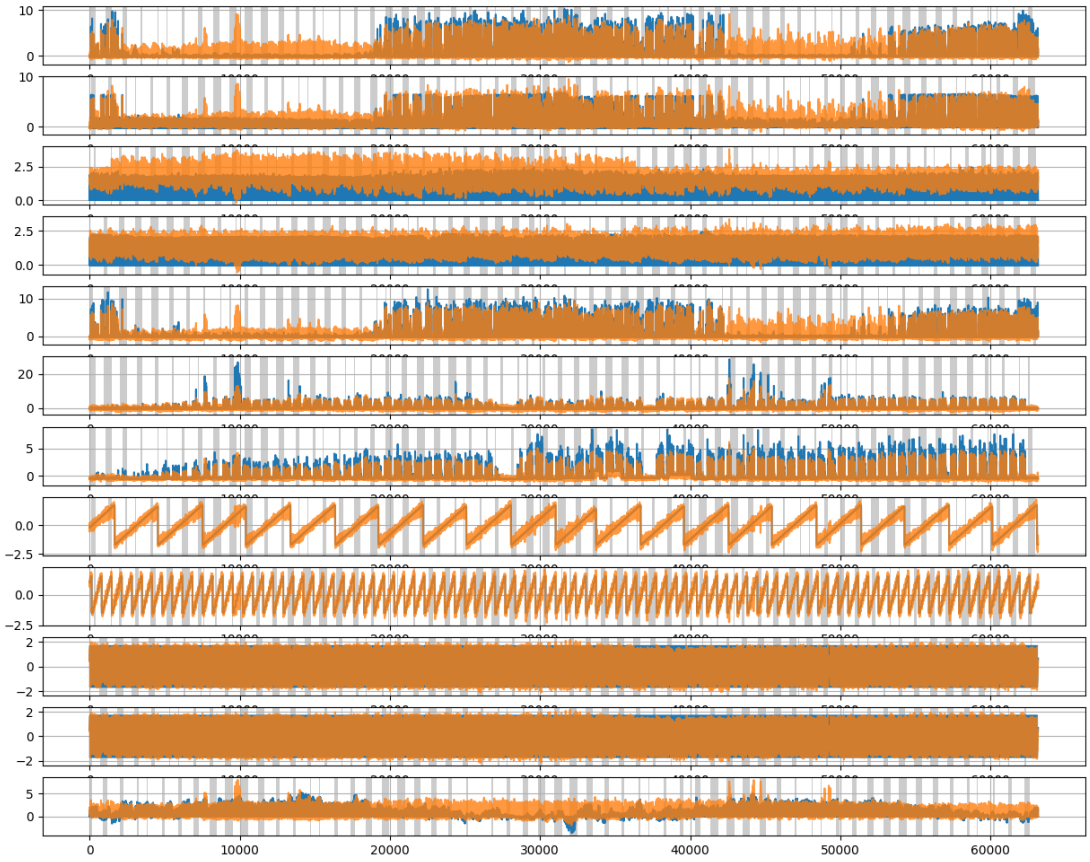


Figure 28: Occlusion experiment on Kita Hokida building. Blue lines denotes the actual values. Orange lines denote the reconstructed values. Gray shaded region denotes missing data.

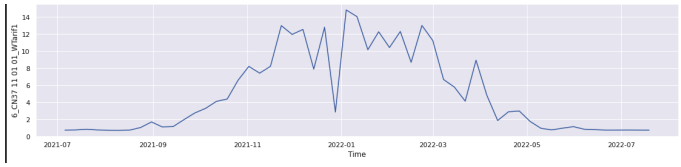


Figure 29: Repeating pattern on feature Wärmetarif yearly

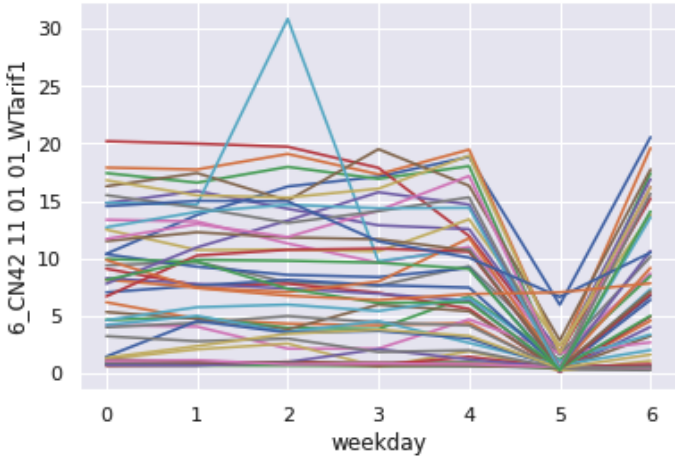


Figure 30: Repeating pattern on feature Wärmetarif on weekly basis

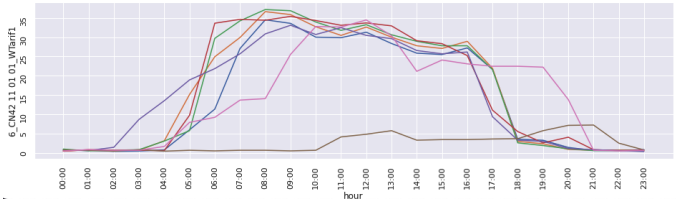


Figure 31: Repeating pattern on feature Wärmetarif during winter period

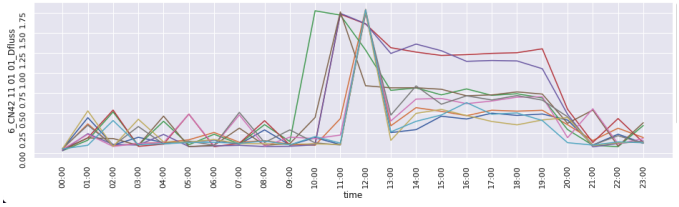


Figure 32: Repeating pattern on feature Durchfluss during exam period

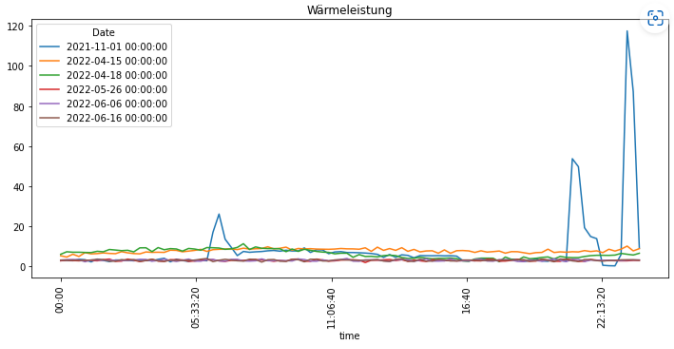


Figure 33: Heat output throughout the day on public holidays(weekdays) for OH14

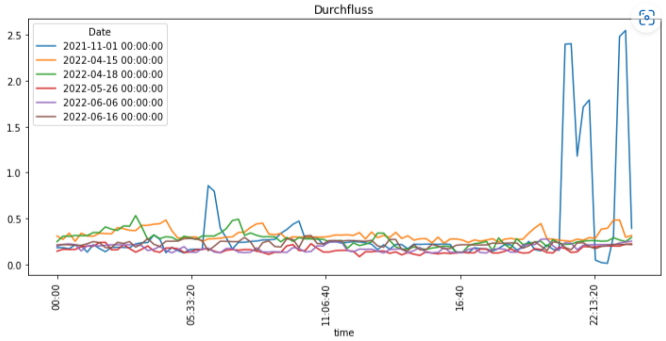


Figure 34: Water consumption throughout the day on public holidays(weekdays) for OH14

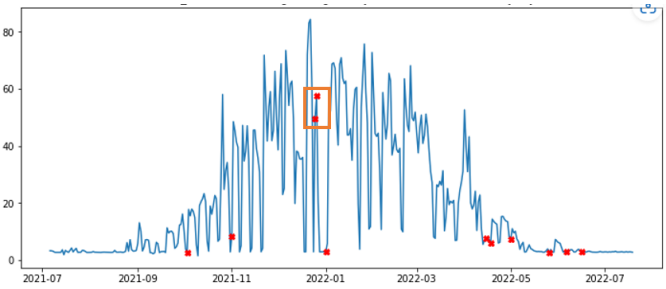


Figure 35: Wärmeleistung(heat output) of OH14 throughout the year (from July,21 to July,22) with markers on holidays

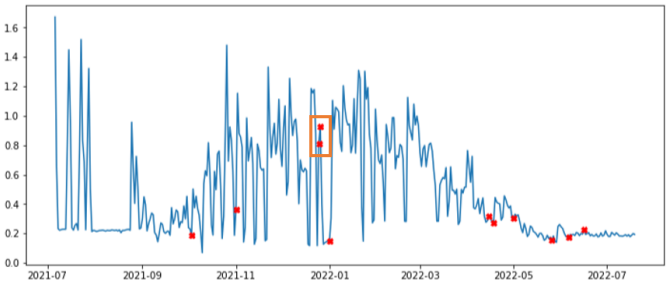


Figure 36: Water consumption of OH14 throughout the year (from July,21 to July,22) with markers on holidays

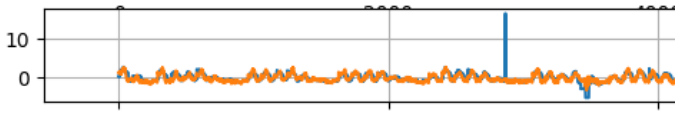


Figure 37: Reconstruction on Building 42 on feature WV+T1

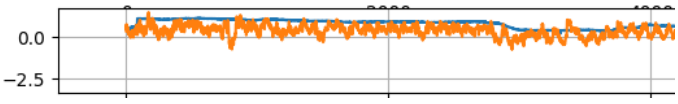


Figure 38: Reconstruction on Building HGII on feature Vorlauftemperatur

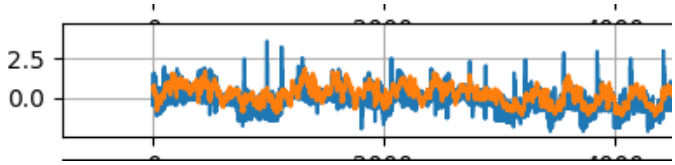


Figure 39: Reconstruction on combined Chemie on feature Volumen

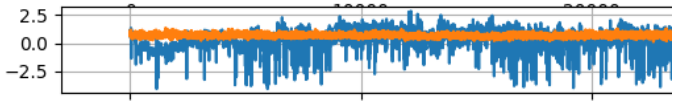


Figure 40: Reconstruction on single chemie building on feature Temperaturdifferenz

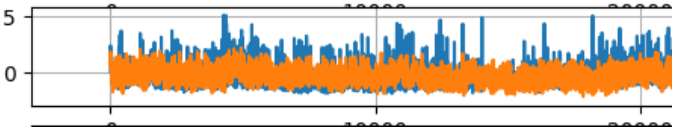
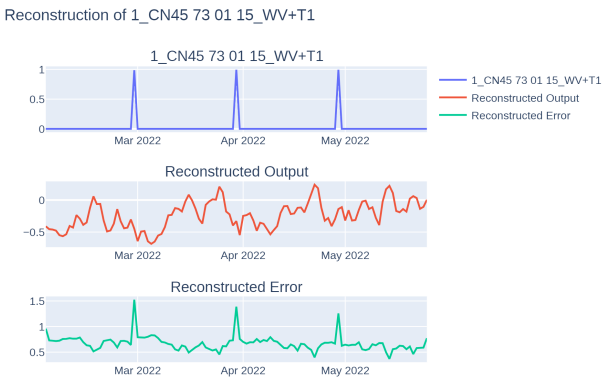
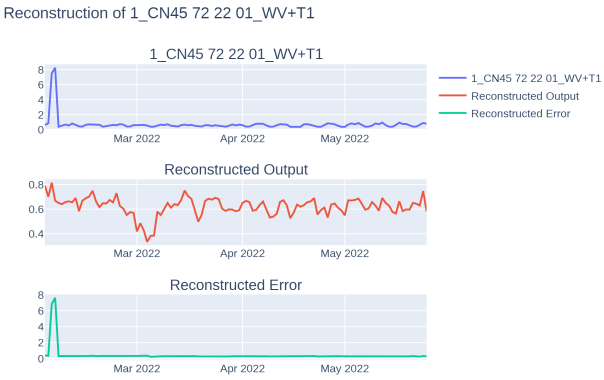


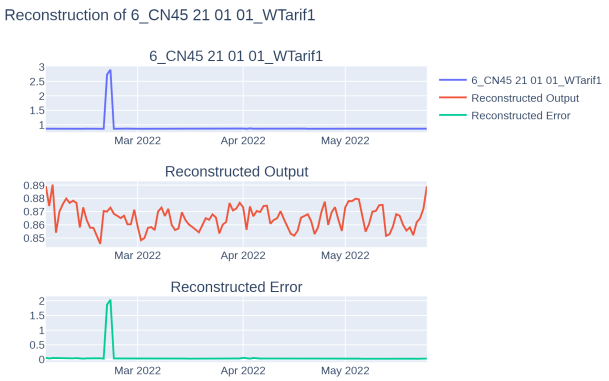
Figure 41: Reconstruction on single chemie building on feature Wärmeleistung



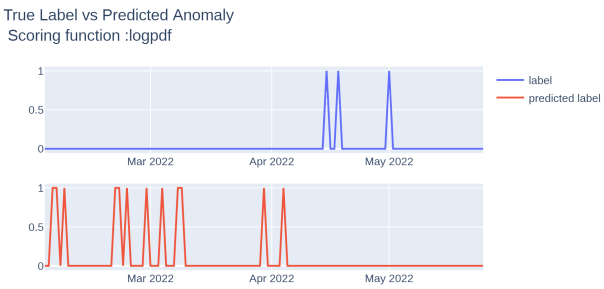
a Reconstruction of WV+ Arbeit Tarif 1(backup power system) for OH12



b Reconstruction of WV+ Arbeit Tarif 1(low voltage) for OH12



c Reconstruction of Wärmeenergie Tarif for OH12



d True Anomaly vs Predicted Anomaly for OH12

Figure 42: Predictions from LSTM model with the base settings for OH12