Unsupervised representation learning for time series anomaly detection and beyond

Ferreira Correia; L.

Chapter 5

Online Multivariate Time Series Anomaly Detection

5.1 Introduction

In this chapter, a novel data-driven approach for anomaly detection in time series is proposed. Fundamentally, it consists of the temporal VAE (TeVAE), which builds on the variational autoencoder architecture introduced in Chapter 2.4 and relies on bidirectional long short-term memory (BiLSTM) layers, presented in Chapter 2.3, to model dynamic properties within time series data. TeVAE is further enhanced by multi-head attention, a mechanism which weighs correlated time steps more, in theory facilitating modelling of longer sequences. In the following section, Chapter 5.2, TeVAE's characteristics, design choices, as well as associated experiments are introduced. In literature, architectures featuring multi-head attention are observed to suffer under the so-called *bypass phenomenon* [99], and hence this problem is introduced, along with its relevance for TeVAE. Another component of the approach is a novel method to remap fixed-length windows to variable-length sequences, so-called reverse-windowing. It is especially useful for evaluation as, compared to evaluating the time series windows output by TeVAE, reverse-windowing allows for a higher degree of interpretability for domain experts. Furthermore, it is also outlined how the novel reverse-window method (RWM) is compared with other, more conventional methods. Additionally, it is of interest how well TeVAE can distinguish unseen, but nominal patterns from anomalous ones. The reasoning behind such an experiment and the setting itself are

also discussed. Following that, more generic experimental settings are outlined, like the issue of unsupervised threshold estimation, online detection performance quantification and testing environment. Moreover, Chapter 5.3 presents the results for the real-world-inspired dataset generated using simulation, as well as for the real-world dataset. Here, TeVAE's detection performance is held up against a range of state-of-the-art anomaly detection approaches, whose source code was either freely available, or whose respective publication allows for re-implementation. Finally, the findings obtained are summarised and conclusions are drawn in Chapter 5.4.

Mathematically, time series anomaly detection in discrete-sequence settings can be defined as follows. Recall dataset $\mathcal{D}$ containing data from an arbitrary discrete-sequence problem. It consists of the training subset $\mathcal{D}^{\text{train}} = \{\mathcal{S}_1, ..., \mathcal{S}_m, ..., \mathcal{S}_M\}$ and test subset $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$, where $\mathcal{S}_m \in \mathbb{R}^{T_m \times d_\mathcal{D}}$ and $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_\mathcal{D}}$. Here, T_m and T_n are the number of time steps in the training sequence $\mathcal{S}_m$ and test sequence $\mathcal{S}_n$, respectively, and $d_\mathcal{D}$ the dimensionality of the dataset, i.e. the number of features. For each test sequence $\mathcal{S}_n$ an anomaly score $\mathbf{s}_n$ is assigned which, in an online setting, is a function of time, such that $\mathbf{s}_n \in \mathbb{R}^{T_n}$. Generally, anomalous behaviour is predicted when time steps in $\mathbf{s}_n$ exceed a threshold τ. As explained in Chapter 4.1 the time series data is transformed from its natural variable-length form to fixed length sequences, called *windows*, to facilitate a more efficient and effective training procedure of model-based approaches. This process also needs to be applied to testing, where each sequence is evaluated individually. Hence, an arbitrary sequence $\mathcal{S}_n$ from $\mathcal{D}^{\text{test}}$ is windowed using the `window` function from Algorithm 1, yielding a rank 3 window tensor $\mathbf{X}_n \in \mathbb{R}^{\lambda_n \times w \times d_\mathcal{D}}$, where λ_n represents the number of windows resulting from $\mathcal{S}_n$, w the window size and $d_\mathcal{D}$ the feature count of the dataset $\mathcal{D}$. λ_n is a function of the number of time steps T_n in $\mathcal{S}_n$, the window size w and the window shift k, as shown in Equation 5.1.

$$\lambda_n = (T_n - w)/k + 1 \tag{5.1}$$

Window $\mathbf{W}$ is hence an arbitrary slice of the first axis in $\mathbf{X}_n$, such that $\mathbf{W} \in \mathbb{R}^{w \times d_\mathcal{D}}$.

5.2 Proposed Methodology

5.2.1 Temporal Variational Autoencoder

To detect anomalies in multivariate time series data, TeVAE is proposed, which is illustrated in Figure 5.1. During training, the encoder q_ϕ, consisting of BiLSTM layers and parameterised by set ϕ, maps input window $\mathbf{W}$ to a temporal distribution with

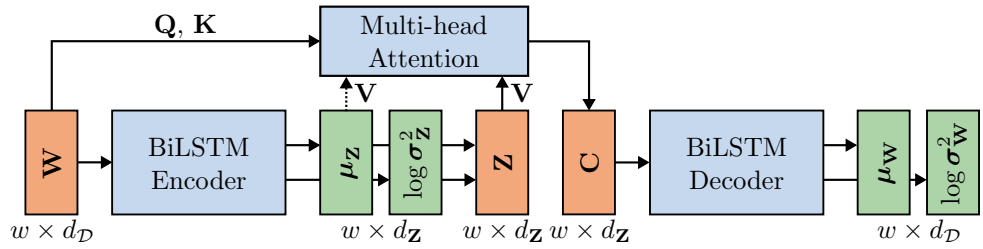

Figure 5.1: An illustration of the proposed TeVAE model. Blue shapes designate trainable models, orange deterministic tensors and green distribution parameters. The shape of each tensor is designated below it. During training $\mathbf{Z}$ is used as the value matrix, denoted by the solid arrow, whereas during inference $\boldsymbol{\mu}_{\mathbf{Z}}$ is used as the value matrix, denoted by the traced arrow.

mean and standard deviation parameters $\boldsymbol{\mu}_{\mathbf{Z}}$ and $\log \boldsymbol{\sigma}_{\mathbf{Z}}^2$, respectively, in the forward pass, Equation 5.2.

$$(\boldsymbol{\mu}_{\mathbf{Z}}, \log \boldsymbol{\sigma}_{\mathbf{Z}}^2) = q_\phi(\mathbf{W}) \tag{5.2}$$

Given the latent distribution with mean and standard deviation parameters $\boldsymbol{\mu}_{\mathbf{Z}}$ and $\log \boldsymbol{\sigma}_{\mathbf{Z}}^2$, respectively, the latent matrix $\mathbf{Z}$ is sampled from the resulting distribution, as shown in Equation 5.3.

$$\mathbf{Z} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{Z}}, \boldsymbol{\sigma}_{\mathbf{Z}}) \tag{5.3}$$

Note that the covariance is not modelled, hence $\boldsymbol{\sigma}_{\mathbf{Z}}$ represents the diagonal of the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{Z}}$, such that $\boldsymbol{\sigma}_{\mathbf{Z}} = \text{diag}(\boldsymbol{\Sigma}_{\mathbf{Z}})$. Then, the input window $\mathbf{W}$ is linearly transformed through weight matrices $\boldsymbol{\Omega}_i^Q$ and $\boldsymbol{\Omega}^{\mathbf{K}}{}_i$ to obtain the query matrices $\mathbf{Q}_i$ and key matrices $\mathbf{K}_i$ for each head i, respectively. Likewise, the sampled latent matrix $\mathbf{Z}$ is also transformed to the value matrix $\mathbf{V}_i$ using weight matrix $\boldsymbol{\Omega}_i^V$, all shown in Equation 5.4.

$$\mathbf{Q}_i = \mathbf{W}\boldsymbol{\Omega}_i^Q \qquad \mathbf{K}_i = \mathbf{W}\boldsymbol{\Omega}_i^K \qquad \mathbf{V}_i = \mathbf{Z}\boldsymbol{\Omega}_i^V \tag{5.4}$$

The weight matrices consist of learnable parameters which are adjusted during the training process. To output the context matrix $\mathbf{C}_i$ for each head i, the softmax of the through $\sqrt{d_{\mathbf{K}}}$ normalised query and key product is multiplied with the value matrix, Equation 5.5.

$$\mathbf{C}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d_{\mathbf{K}}}}\right)\mathbf{V}_i \tag{5.5}$$

The final context matrix $\mathbf{C}$ is the result of the by $\mathbf{\Omega}^O$ linearly transformed concatenation of each head-specific context matrix $\mathbf{C}_i$, as expressed in Equation 5.6.

$$\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_h]\mathbf{\Omega}^O \tag{5.6}$$

The decoder p_θ, also consisting of BiLSTM layers and parameterised by set θ, then maps the context matrix $\mathbf{C}$ to $\boldsymbol{\mu}_\mathbf{W}$ and $\log \boldsymbol{\sigma}^2_\mathbf{W}$, as shown in Equation 5.7. These can then be used to parametrise output distribution $\mathcal{N}(\boldsymbol{\mu}_\mathbf{W}, \boldsymbol{\sigma}_\mathbf{W})$ where, again, covariance is not modelled, i.e. $\boldsymbol{\sigma}_\mathbf{W} = \text{diag}(\mathbf{\Sigma}_\mathbf{W})$.

$$(\boldsymbol{\mu}_\mathbf{W}, \log \boldsymbol{\sigma}^2_\mathbf{W}) = p_\theta(\mathbf{C}) \tag{5.7}$$

Mapping the output to a distribution rather than a deterministic vector allows TeVAE to model uncertainty, which is assumed to be normally distributed. This forward pass is also summarised in Algorithm 2.

Algorithm 2 TeVAE Training Forward Pass

Require: Window $\mathbf{W} \in \mathbb{R}^{w \times d_\mathcal{D}}$
1: $(\boldsymbol{\mu}_\mathbf{Z}, \log \boldsymbol{\sigma}^2_\mathbf{Z}) \leftarrow q_\phi(\mathbf{W})$ ▷ Input window into encoder
2: $\mathbf{Z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{Z}, \boldsymbol{\sigma}_\mathbf{Z})$ ▷ Sample from latent distribution
3: **for** $i = 1 \rightarrow h$ **do**
4: $\quad \mathbf{Q}_i = \mathbf{W}\mathbf{\Omega}^Q_i$ ▷ Obtain query matrix for head i
5: $\quad \mathbf{K}_i = \mathbf{W}\mathbf{\Omega}^K_i$ ▷ Obtain key matrix for head i
6: $\quad \mathbf{V}_i = \mathbf{Z}\mathbf{\Omega}^V_i$ ▷ Obtain value matrix for head i
7: $\quad \mathbf{C}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_\mathbf{K}}}\right)\mathbf{V}_i$ ▷ Obtain context matrix for head i
8: **end for**
9: $\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_h]\mathbf{\Omega}^O$ ▷ Concatenate and transform context matrix
10: $(\boldsymbol{\mu}_\mathbf{W}, \log \boldsymbol{\sigma}^2_\mathbf{W}) \leftarrow p_\theta(\mathbf{C})$ ▷ Input context matrix into decoder
11: **return** Parameters of output distribution $(\boldsymbol{\mu}_\mathbf{W}, \log \boldsymbol{\sigma}^2_\mathbf{W})$

Despite the generative capabilities of variational autoencoders (VAEs), TeVAE does not leverage generation for anomaly detection. Rather than sampling a latent matrix $\mathbf{Z}$ as shown in Equation 5.3 during inference, sampling is disabled and only $\boldsymbol{\mu}_\mathbf{Z}$ is taken as the input for the multi-head attention mechanism, as done by von Schleinitz et al. [80]. Equation 5.3 (line 2 in Algorithm 2), therefore, is replaced by Equation 5.8 for the forward pass, as shown in line 2 in Algorithm 3.

$$\mathbf{Z} = \boldsymbol{\mu}_\mathbf{Z} \tag{5.8}$$

This not only accelerates inference by eliminating the sampling process but is also

Algorithm 3 TeVAE Inference Forward Pass (`tevae`)

Require: Window $\mathbf{W} \in \mathbb{R}^{w \times d_{\mathcal{D}}}$
$(\boldsymbol{\mu}_{\mathbf{Z}}, \log \boldsymbol{\sigma}^2_{\mathbf{Z}}) \leftarrow q_\phi(\mathbf{W})$ ▷ Input window into encoder
$\mathbf{Z} = \boldsymbol{\mu}_{\mathbf{Z}}$ ▷ Instead of sampling from $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{Z}}, \boldsymbol{\sigma}_{\mathbf{Z}})$
for $i = 1 \rightarrow h$ **do**
  $\mathbf{Q}_i = \mathbf{W}\boldsymbol{\Omega}_i^Q$ ▷ Obtain query matrix for head i
  $\mathbf{K}_i = \mathbf{W}\boldsymbol{\Omega}_i^K$ ▷ Obtain key matrix for head i
  $\mathbf{V}_i = \mathbf{Z}\boldsymbol{\Omega}_i^V$ ▷ Obtain value matrix for head i
  $\mathbf{C}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d_{\mathbf{K}}}}\right)\mathbf{V}_i$ ▷ Obtain context matrix for head i
end for
$\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_h]\boldsymbol{\Omega}^O$ ▷ Concatenate and transform context matrix
$(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}^2_{\mathbf{W}}) \leftarrow p_\theta(\mathbf{C})$ ▷ Input context matrix into decoder
return Parameters of output distribution $(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}^2_{\mathbf{W}})$

empirically found to be a good approximation of an averaged latent matrix if it were sampled several times as done by Pereira et al. [100]. The TeVAE layout during inference is also shown in Figure 5.1, where the traced arrow designates the information flow from the encoder to the multi-head attention (MA) mechanism during inference.

5.2.2 Bypass Phenomenon

VAEs, when combined with an attention mechanism, can exhibit a behaviour called the bypass phenomenon [99]. When the bypass phenomenon occurs, the latent path between encoder and decoder is bypassed, and information flow occurs mostly or exclusively through the attention mechanism, as it has deterministic access to the encoder hidden states and therefore avoids regularisation through the D_{KL} term. In an attempt to counteract this, Bahuleyan et al. [99] propose variational attention, which, like the VAE, maps the input to a distribution rather than a deterministic vector. Applied to natural language processing, Bahuleyan et al. [99] demonstrate that this leads to a diversified generated portfolio of sentences, indicating alleviation of the bypassing phenomenon. Only Pereira et al. [100] apply this insight in the anomaly detection domain; however, they do not present any evidence of alleviation of the bypass phenomenon in their work. TeVAE on the other hand, cannot suffer from the bypass phenomenon in the sense that information flow ignores the latent variational path between encoder and decoder, since the MA mechanism requires the value matrix $\mathbf{V}$ from the encoder to output the context matrix. Assuming the bypass phenomenon also applies to a case where information flow ignores the attention mechanism, one

could claim that TeVAE is not immune. To investigate this further, the attention mechanism is removed from the model in an ablation study to see if anomaly detection performance is affected. In this case, $\mathbf{Z}$ is instead directly input into the decoder during training, whereas $\mu_{\mathbf{Z}}$ is used during inference. If detection performance drops, it is considered evidence of the contribution of the attention mechanism to the model performance and hence is not bypassed. The results of this ablation study are shown and discussed in Chapter 5.3.

5.2.3 Mapping Fixed-length Windows to Variable-length Sequences

Since the TeVAE is trained to reconstruct fixed-length windows, the same applies during inference. However, to decide whether a given measurement sequence $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$ is anomalous, a continuous reconstruction of the measurement is required. A trivial way to do so would be to window the input measurement $\mathcal{S}_n$ using a shift $k = 1$, input the windows into the model and append the last time step from each output window to obtain a continuous sequence [101], which is referred to as last-type RWM from now on. Alternatively, it is also possible to invert the same procedure by taking the first time step rather than the last one, which from now on is referred to as first-type RWM. Considering the BiLSTM nature of the encoder and decoder, the first and last time steps of a window can only be computed given the states from one direction, making these values, in theory, less accurate. To overcome this, averaging matching time steps in overlapping windows is proposed, which is called *mean-type* RWM. In practice, overlapping time steps of different windows are added together, while a counter array keeps track of how many windows contribute to each time step. Finally, each time step is divided by the corresponding value inside the counter array, yielding the average, as detailed in Algorithm 4.

With the foundations laid out, the process of inference and detecting anomalies can be introduced. The input for this process is the output distribution parameters $(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}^2_{\mathbf{W}})$, so it essentially averages the distributions and hence can only be applied to the mean and *variance* parameters. Consider the distributions $\mathcal{N}(\mu_x, \sigma_x^2)$ and $\mathcal{N}(\mu_y, \sigma_y^2)$, both assumed to be independent and normally distributed. The sum of both distributions results in normal distribution $\mathcal{N}(\mu_{x+y}, \sigma^2_{x+y})$, obtained as shown in Equation 5.9 [102].

$$\mu_{x+y} = \mu_x + \mu_y \qquad \sigma^2_{x+y} = \sigma^2_x + \sigma^2_y \tag{5.9}$$

Therefore, the standard deviation of the resulting distribution σ_{x+y} is characterised

Algorithm 4 Mean-type RWM (`reverse_window`)

Require: Window Tensor $\mathbf{X} \in \mathbb{R}^{\lambda \times w \times d_{\mathcal{D}}}$, Window Shift $k \in \mathbb{N}$

$\hat{\mathcal{S}}_n \leftarrow \text{zeros}(T_n, d_{\mathcal{D}})$ ▷ Pre-allocate sequence
$\mathbf{c} \leftarrow \text{zeros}(T_n, d_{\mathcal{D}})$ ▷ Pre-allocate counter array
for $i = 1 \rightarrow \lambda$ **do**
  $\mathbf{W} \leftarrow \mathbf{X}[i]$ ▷ Get one window from the window tensor
  $\hat{\mathcal{S}}_n[i * k : i * k + w] \leftarrow \hat{\mathcal{S}}_n[i * k : i * k + w] + \mathbf{W}$ ▷ Add window
  $\mathbf{c}[i * k : i * k + w] \leftarrow \mathbf{c}[i * k : i * k + w] + 1$ ▷ Update counter
end for
$\hat{\mathcal{S}}_n \leftarrow \hat{\mathcal{S}}_n / \mathbf{c}$ ▷ Obtain average by dividing by counter
return Sequence $\hat{\mathcal{S}}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$

by Equation 5.10.

$$\sigma_{x+y} = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{5.10}$$

Hence, to take the mean of the distributions, the variance $\boldsymbol{\sigma}_{\mathbf{W}}^2$ is averaged, and then converted to the *standard deviation* $\boldsymbol{\sigma}_{\mathbf{W}}$.

With a continuous mean $\boldsymbol{\mu}_{\mathcal{S}}$ and standard deviation $\boldsymbol{\sigma}_{\mathcal{S}}$, the continuous negative log-likelihood, i.e. the anomaly score $\mathbf{s}$, is computed for the respective sequence. The anomaly detection process is also outlined in Algorithm 5. Note that, this algorithm makes use of the `window` function from Algorithm 1, the TeVAE inference forward pass function `tevae` function from Algorithm 3 and the `reverse_window` function from Algorithm 4.

Algorithm 5 Anomaly Detection Process

Require: Sequence $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$, Threshold $\tau \in \mathbb{R}$

$\mathbf{X} \leftarrow \texttt{window}(\mathcal{S}_n)$ ▷ Obtain window tensor from sequence
for $i = 1 \rightarrow \lambda$ **do**
  $\mathbf{W} \leftarrow \mathbf{X}[i]$ ▷ Get one window from the window tensor
  $(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}_{\mathbf{W}}^2) \leftarrow \texttt{tevae}(\mathbf{W})$ ▷ Inference forward pass by TeVAE
  $\boldsymbol{\mu}_{\mathbf{X}}[i] \leftarrow \boldsymbol{\mu}_{\mathbf{W}}$
  $\boldsymbol{\sigma}_{\mathbf{X}}^2[i] \leftarrow \boldsymbol{\sigma}_{\mathbf{W}}^2$
end for
$\boldsymbol{\mu}_{\mathcal{S}} \leftarrow \texttt{reverse\_window}(\boldsymbol{\mu}_{\mathbf{X}})$ ▷ Reverse-window output mean parameter
$\boldsymbol{\sigma}_{\mathcal{S}}^2 \leftarrow \texttt{reverse\_window}(\boldsymbol{\sigma}_{\mathbf{X}}^2)$ ▷ Reverse-window output variance parameter
$\mathbf{s}_n \leftarrow -\log p(\boldsymbol{\mu}_{\mathcal{S}}, \boldsymbol{\sigma}_{\mathcal{S}} | \mathcal{S}_n)$ ▷ Obtain continuous anomaly score
$l_n \leftarrow \max(\mathbf{s}_n) > \tau$ ▷ Compare maximum value in anomaly score with threshold
return Label l_n for sequence $\mathcal{S}_n$

A comparison between the detection performance obtained using the mean-type,

last-type, and first-type RWMs is provided in Chapter 5.3.

The intrinsic delay δ_{intr} associated with each of the RWMs can be discussed ahead of the results in Chapter 5.3, however. δ_{intr} is defined as the delay naturally introduced by each RWM. To aid analysis, the intrinsic delay δ_{intr} is plotted against time t to demonstrate the delay for each of the RWMs, shown in Figure 5.2.

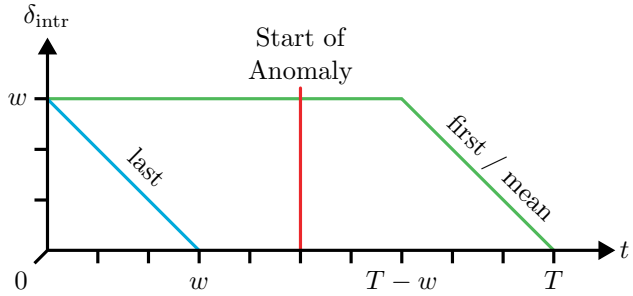

Figure 5.2: Intrinsic delay δ_{intr} plotted against time t. The last-type RWM is represented by a solid blue line, and the first-type and mean-type by a solid green line. The red line represents the start of a hypothetical sub-sequence anomaly.

Consider the last-type RWM during $0 < t < w$. Since the models considered in this thesis are trained on fixed-length windows, evaluation can only start once w time steps are recorded to constitute a window, meaning that an anomalous subsequence starting at $t = 0$ can only be detected $\delta_{\text{intr}} = w$ time steps later. Likewise, an anomaly starting at $t = 1$ can only be detected $\delta_{\text{intr}} = w - 1$ time steps later and another starting at $t = w$ can be detected immediately, which holds true for the rest of the sequence, as the last time step of each window corresponds to the current real-world time step, i.e. $\delta_{\text{intr}} = 0$ for $w < t < T$. A non-zero intrinsic delay until $t = w$ is natural to any window-based approach, and hence the first-type and mean-type RWMs show the same behaviour. In contrast to last-type RWM, however, these methods only output the first value of each window, i.e. evaluation is $\delta_{\text{intr}} = w$ time steps behind for $0 < t < T - w$. At $t = T$, though, the time steps $T - w < t < T$ are all output at the same time, meaning that $\delta_{\text{intr}} = 0$ and no extra time is needed for evaluation after streaming ends.

It becomes clear that, in *online* time series anomaly detection, the last-type RWM is in theory faster than the other two types. For instance, consider a sub-sequence anomaly starting at time step $w + 2$, designated by the red line in Figure 5.2. Apart from the time required for inference, the last-type RWM can detect the anomaly with a intrinsic delay $\delta_{\text{intr}} = 0$, whereas the other two methods can only detect the anomaly

$\delta_{\text{intr}} = w$ time steps later. For $0 < t < T - w$, a intrinsic delay of $\delta_{\text{intr}} = w$ time steps is, therefore, the best that first-type and mean-type RWMs can achieve, however, while the best delay the last-type can achieve is $\delta_{\text{intr}} = 0$, it will be higher in reality, perhaps higher than w time steps.

5.2.4 Generalisation Capabilities

In some real-world cases, the available training data may not cover all real-world driving scenarios, for example at the initial stages of a project with limited data availability, or outside a test bench scenario on a real road, where driving does not follow a provided pattern, but is much more random due to external influences, like traffic or weather. Translating this to an experimental setting, some driving cycles within the test subset are not present in the training subset, unlike the previous experiments. Modelling nominal behaviour is therefore especially challenging, as the main problem involves distinguishing between nominal but unseen driving patterns and anomalous driving patterns. To investigate how well TeVAE performs in such a setting, the powertrain anomaly time series benchmark (PATH) training subset is split into two sets of three folds. Driving cycles are characterised by the vehicle speed, but it is not available as a channel, however vehicle speed is directly proportional to axle angular speed, and hence it is used as a proxy instead. The first set of folds consists of sequences with a low, medium, and high average axle angular speed, essentially splitting by operational range. Splitting the dataset this way will lead to the folds representing mostly urban, country roads and motorway rides, respectively. The second set of folds, on the other hand, consists of sequences with a low, medium, and high axle angular speed standard deviation, therefore splitting by levels of dynamics. Splitting the dataset this way will lead to the folds representing mostly constant, mixed and highly dynamic rides, respectively. For each fold, TeVAE is trained using seeds 1, 2 and 3, and is evaluated using the same test subset each time.

5.2.5 Experimental Setup

Threshold Estimation Method

As mentioned in Chapter 3.2, the vast majority of literature chooses a detection threshold τ by using labelled data one way or another, despite claiming to propose an unsupervised approach to anomaly detection. Therefore, to provide a truly unsupervised baseline, the so-called *unsupervised threshold* τ_{us} is set as the maximum anomaly score

observed in the validation subset $\mathcal{D}^{\text{val}}$. In cases with exclusively nominal sequences in $\mathcal{D}^{\text{val}}$, this will naturally lead to a high precision, as the number of false positives $N_{\mathbf{fp}}$ will be low. In cases where $\mathcal{D}^{\text{val}}$ may be contaminated by anomalous data, however, this threshold choice can lead to poor detection performance. For instance, if, due to a high anomaly score from an anomaly in $\mathcal{D}^{\text{val}}$, threshold τ is set too large, all sequences within $\mathcal{D}^{\text{test}}$ will be labelled as nominal, leading to a perfect precision figure of $P = 1$ but a recall of $R = 0$ and therefore an $F_1 = 0$.

Clearly, the threshold choice has a large impact on the anomaly detection performance, though it is not the only factor, which can hold back the detection performance of an otherwise well-performing approach. To ablate the threshold choice from other factors that influence the detection performance, the so-called *hypothetical best threshold* τ_{best} is employed. τ_{best} is the threshold that leads to the best F_1 score on the respective test subset possible with the respective approach, and is obtained by grid-searching through the percentiles of all anomaly scores stemming from the test subset $\mathcal{D}^{\text{test}}$. Clearly, this would not be observable in the real world given the absence of a labelled test subset, hence why it is referred to as *hypothetical*.

Evaluation Metrics

As discussed in Chapter 3.1, no set of evaluation metrics proposed in the literature takes timeliness into account, is parameter-free and compatible with discrete-sequence problems. To address this, this section introduces a novel set of metrics. These metrics mostly rely on the conventional definition of true positive, false positive, true negative and false negative labels applied to each individual discrete sequence, except in the case of a sub-sequence anomaly.

For partially anomalous time series, it can be labelled as a true positive or a false negative, but also as a false positive, which occurs when an anomaly detector flags a time step prematurely. Formally, this is the case when the first flagged time step is far enough ahead of the first ground-truth time step that the model cannot have had access to it. As is evident, the proposed metrics can only be applied to discrete time series data sets where anomalous time series have at most one contiguous ground-truth anomalous sub-sequence of any length.

In addition to that, the time between detection and ground-truth anomaly start is also quantified for each anomalous $\mathcal{S}_n$, with false positives being assigned the absolute value of the "negative" delay and false negatives being assigned the length of the anomalous sub-sequence within $\mathcal{S}_n$. The detection delays are finally aggregated into the average detection delay $\bar{\delta}$, given in seconds. The issue with these metrics is that

they are not *recall consistent* as defined by Wagner et al. [33], meaning that the recall monotonically decreases with an increasing threshold. Consider a time series with a sub-sequence anomaly starting off as nominal and eventually becoming anomalous, as shown in Figure 5.3. For a very low threshold τ_1, the anomaly is considered a false

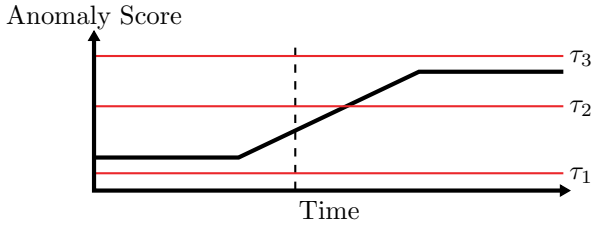

Figure 5.3: Arbitrary anomaly score as a function of time. The dotted vertical line denotes the start of the anomalous behaviour, and the red lines represent three different thresholds.

positive, since it is an early detection. As the threshold increases to τ_2, it leads to a true positive, but when the threshold reaches τ_3, it becomes a false negative. This leads to an increase and then decrease in recall, and hence the metrics do not qualify as recall consistent, which also prevents them from being used to calculate the area under the precision-recall curve to quantify the uncalibrated detection performance. Additionally, unlike the improved range-based recall R_{rb} and precision P_{rb} by Wagner et al. [33], this set of metrics does not account for predictive patterns. Despite its short-comings, it is the only set of metrics that are apt for online discrete-sequence problems. Nonetheless, there is still a clear need for a novel metric which combines the compatibility with discrete-sequence problems with recall consistency and consideration of predictive patterns.

Competing Approaches

In this work, several approaches from the literature are considered, which either published the source code provided enough information for re-implementation. The selection includes various autoencoder-based models, as well as a state-of-the-art nonparametric approach.

In the time series anomaly detection literature, the only other model that uses the combination of a VAE and an attention mechanism is by Pereira et al. [100]. For the purpose of this work, it is referred to as variational self-attention VAE (VS-VAE). Their approach consists of a BiLSTM encoder and decoder, where, for an input window of length w, the $t = w$ encoder hidden states of each direction are passed on to the variational self-attention (VS) mechanism [99]. The resulting context vector is

then concatenated with the sampled latent vector and then passed on to the decoder. Pereira et al. [100] claims that applying VS to the VAE solves the bypass phenomenon, however, no evidence for this claim is provided.

OmniAnomaly (OA) [2] attempts to create a temporal connection between latent distributions by applying a linear Gaussian state space model to them. Furthermore, it concatenates the last gated recurrent unit (GRU) hidden state with the latent vector sampled in the previous time step. In addition to that, it uses planar normalising flow [103] by applying K transformations to the latent vector to approximate a non-Gaussian posterior.

As part of the VAE-based selective prediction (VASP) framework [80], a variational autoencoder architecture is proposed to increase the robustness of time series prediction when faced with anomalies. While the main contribution is attributed to the framework itself, not the VAE, it should be noted that during inference only the mean parameter of the latent distribution is passed to the decoder.

Thill et al. [71] suggest a temporal convolutional autoencoder (TCN-AE) with skip connections to prevent an overreliance of the model on the high dilation rate representation. Furthermore, Thill et al. propose using the reduced representations between hidden layers in addition to map reduction layers after each hidden layer to reduce the dimensionality of the convolution and therefore the number of tuneable parameters.

Smoothness-inducing sequential VAE (SISVAE) [81] tries to improve the modelling robustness by the addition of a smoothing term in the loss function which contributes to the reduction of sudden changes in the reconstructed signal, making it less sensitive to noisy time steps.

The lightweight LSTM-VAE (LW-VAE) [94] is another variational autoencoder model which offers no significant contributions besides being relatively parameter-light.

Time series anomaly detection through incremental search (TSADIS) [104] is a non-parametric approach based on the matrix profile (MP) computation. As far as the author is aware, it is the only MP-based approach that can be applied to multivariate time series problems. One major benefit of TSADIS, is that, given its non-parametric nature, it does not rely on a computationally expensive training procedure, lowering the hurdle of implementation in the real world. It is simply called on the entire variable-length sequence, which while simple, also means that it is technically an *offline* approach, as all time steps need to be recorded before it can be evaluated.

The hyperparameters for each approach are set as specified in the respective publi-

cation, though early stopping is applied to all that require a training procedure. Early stopping is parameterised such that the respective reconstruction error is monitored and training is stopped once validation loss has stopped decreasing for 250 epochs. To this end, 20 % of the unlabelled training subset $\mathcal{D}^{\text{train}}$ is separated to form the validation subset $\mathcal{D}^{\text{val}}$.

Environment

The framework used for model training is TensorFlow 2.15.1 and TensorFlow Probability 0.23 on Python 3.10 on a workstation running Ubuntu 22.04.5 LTS, equipped with two Nvidia RTX A6000 GPUs. All work involving TSADIS is done in a separate environment with the latest compatible Python version of 3.9. Further information on library versions used can be found in the *requirements.txt* file in the GitHub repository under `github.com/lcs-crr/Thesis`.

5.3 Results

5.3.1 Online Anomaly Detection

PATH Dataset

First, the selection of approaches is tested on the unsupervised version of the PATH dataset. The PATH dataset is a synthetic but real-world-inspired dataset, representing the behaviour of an automotive powertrain. For more details, please refer to Chapter 4.1. This version of the dataset has a contaminated training subset, i.e. it contains unlabelled anomalies. The results obtained using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are shown in tabular form in Table 5.1.

All approaches perform poorly in terms of F_1 score when τ_{us} is employed, with OA performing slightly better. Over all the folds and seeds, the average precision P varies by a large margin. On average, VS-VAE manages to label 78 % of the ground-truth nominal sequences as nominal, with OA 60 % and TCN-AE 48 %. When it comes to recall R, however, all approaches achieve low figures, which explains the high average detection delay $\bar{\delta}$ and low F_1 scores. Additionally, TSADIS finds no anomalies, which is why F_1 score, precision P and recall R are 0 throughout. These observations can be explained by the threshold choice, which sets the boundary between nominal and anomalous samples too large, sometimes even so high, that all sequences in $\mathcal{D}^{\text{test}}$ lie below it, as is the case with TSADIS. After finding τ_{best}, however, a poor threshold

Table 5.1: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for a range of approaches applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
VS-VAE	0.03 ± 0.02	0.78 ± 0.42	0.01 ± 0.01	991.4 ± 71.1	
OA	$\mathbf{0.06 \pm 0.06}$	0.60 ± 0.44	0.03 ± 0.03	994.9 ± 69.7	
VASP	0.02 ± 0.02	0.21 ± 0.34	0.01 ± 0.01	991.9 ± 70.8	
TCN-AE	0.02 ± 0.02	0.45 ± 0.41	0.01 ± 0.01	989.8 ± 66.9	τ_{us}
SISVAE	0.03 ± 0.03	0.13 ± 0.10	0.02 ± 0.02	993.1 ± 69.6	
LW-VAE	0.01 ± 0.02	0.26 ± 0.41	0.01 ± 0.01	991.9 ± 71.0	
TSADIS	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	–	
TeVAE	0.02 ± 0.03	0.27 ± 0.36	0.01 ± 0.02	992.7 ± 70.7	
VS-VAE	0.30 ± 0.07	0.35 ± 0.26	0.34 ± 0.05	792.0 ± 75.4	
OA	0.50 ± 0.09	0.60 ± 0.11	0.44 ± 0.11	767.0 ± 114.7	
VASP	0.11 ± 0.01	0.07 ± 0.01	0.47 ± 0.24	673.5 ± 181.2	
TCN-AE	0.14 ± 0.01	0.09 ± 0.01	0.41 ± 0.16	750.2 ± 121.1	τ_{best}
SISVAE	0.22 ± 0.03	0.19 ± 0.05	0.28 ± 0.04	826.2 ± 80.0	
LW-VAE	0.13 ± 0.01	0.07 ± 0.01	0.51 ± 0.22	634.8 ± 172.6	
TSADIS	0.10 ± 0.01	0.05 ± 0.00	1.00 ± 0.00	–	
TeVAE	$\mathbf{0.58 \pm 0.08}$	0.69 ± 0.12	0.50 ± 0.09	676.4 ± 103.2	

choice can be isolated from the detection performance. Now, a larger gap between F_1 scores is observable, with TeVAE performing the best, with a score of $F_1 = 0.58$ and a much reduced average detection delay of $\bar{\delta} = 676.4\,\text{s}$. OA and VS-VAE perform second and third best, scoring $F_1 = 0.50$ and $F_1 = 0.30$, respectively.

Next, the selection of approaches is tested on the semi-supervised version of the PATH dataset. This version of the dataset consists of a nominal-only training subset, i.e. it contains no anomalies. The results obtained using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are shown in tabular form in Table 5.2.

For the semi-supervised results using τ_{us}, the detection performance of some approaches improve significantly, namely for VS-VAE, OA, SISVAE and TeVAE. All above-mentioned approaches now label over 90 % of the nominal sequences as such. At least for TeVAE, the average detection delay has now also decreased by 41 %. These gains can be attributed to a much better placement of the threshold, as well as more successful modelling of nominal behaviour without contamination within the training subset. Despite the relaxed conditions in this experiment compared to the conditions

Table 5.2: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for a range of approaches applied to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version without anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

	Approach	F_1	P	R	$\bar{\delta}$ $[s]$
τ_{us}	VS-VAE	0.43 ± 0.15	0.90 ± 0.09	0.30 ± 0.11	825.9 ± 130.8
	OA	$\mathbf{0.67 \pm 0.15}$	0.96 ± 0.04	0.54 ± 0.19	713.7 ± 203.8
	VASP	0.04 ± 0.03	0.34 ± 0.28	0.02 ± 0.01	986.7 ± 64.7
	TCN-AE	0.02 ± 0.02	0.37 ± 0.39	0.01 ± 0.01	988.6 ± 66.6
	SISVAE	0.61 ± 0.09	0.96 ± 0.05	0.46 ± 0.11	807.8 ± 151.8
	LW-VAE	0.03 ± 0.02	0.40 ± 0.35	0.02 ± 0.01	987.5 ± 66.0
	TSADIS	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	–
	TeVAE	0.63 ± 0.22	0.95 ± 0.06	0.52 ± 0.26	589.1 ± 212.0
τ_{best}	VS-VAE	0.53 ± 0.05	0.84 ± 0.14	0.40 ± 0.07	780.1 ± 100.0
	OA	$\mathbf{0.86 \pm 0.05}$	0.88 ± 0.05	0.85 ± 0.06	467.7 ± 91.6
	VASP	0.12 ± 0.01	0.07 ± 0.01	0.42 ± 0.19	712.6 ± 148.7
	TCN-AE	0.15 ± 0.01	0.09 ± 0.01	0.48 ± 0.12	703.0 ± 96.1
	SISVAE	0.71 ± 0.05	0.84 ± 0.10	0.62 ± 0.04	732.6 ± 124.2
	LW-VAE	0.13 ± 0.01	0.08 ± 0.01	0.48 ± 0.23	655.8 ± 162.6
	TSADIS	0.10 ± 0.01	0.05 ± 0.00	1.00 ± 0.00	–
	TeVAE	0.80 ± 0.15	0.88 ± 0.11	0.76 ± 0.18	412.6 ± 143.8

using the contaminated training subset, VASP, TCN-AE and LW-VAE still fail to yield satisfactory results, all scoring less than $F_1 = 0.05$. Looking at the semi-supervised results using τ_{best} reveals these approaches do not only struggle due to the threshold choice, but also with the challenge of detecting anomalies in this dataset in general. While their results have slightly improved, none score more than $F_1 = 0.15$. In contrast, VS-VAE, OA, SISVAE and TeVAE make slight gains, with OA again scoring the highest with $F_1 = 0.86$. Because TSADIS does not rely on a training procedure, its results naturally remain the same, regardless of the presence of contamination in $\mathcal{D}^{\text{train}}$.

Proprietary Dataset

Now, the selection of approaches is tested on a real-world dataset. This dataset also represents the behaviour of an automotive powertrain on a test bench in a semi-supervised setting, meaning that the training subset $\mathcal{D}^{\text{train}}$ consists of nominal sequences only. For more details, please refer to Chapter 4.2. The results obtained

using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are shown in tabular form in Table 5.3.

Table 5.3: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for a range of approaches applied to the real-world dataset. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
VS-VAE	0.41 ± 0.19	0.86 ± 0.07	0.30 ± 0.19	697.9 ± 120.4	
OA	0.26 ± 0.07	0.67 ± 0.16	0.17 ± 0.05	782.9 ± 19.2	
VASP	0.14 ± 0.04	0.66 ± 0.11	0.08 ± 0.02	792.1 ± 6.6	
TCN-AE	0.24 ± 0.03	0.79 ± 0.11	0.14 ± 0.02	773.6 ± 5.9	τ_{us}
SISVAE	0.35 ± 0.12	0.73 ± 0.16	0.25 ± 0.14	742.7 ± 78.9	
LW-VAE	0.16 ± 0.03	0.75 ± 0.07	0.09 ± 0.02	791.1 ± 6.2	
TSADIS	0.11 ± 0.00	0.37 ± 0.04	0.06 ± 0.00	–	
TeVAE	$\mathbf{0.42 \pm 0.16}$	0.83 ± 0.12	0.31 ± 0.14	720.5 ± 59.3	
VS-VAE	0.53 ± 0.15	0.65 ± 0.24	0.55 ± 0.21	569.1 ± 147.8	
OA	0.38 ± 0.03	0.42 ± 0.24	0.59 ± 0.29	650.1 ± 121.2	
VASP	0.43 ± 0.03	0.43 ± 0.04	0.44 ± 0.05	546.1 ± 46.6	
TCN-AE	0.52 ± 0.03	0.57 ± 0.10	0.49 ± 0.05	622.7 ± 46.0	τ_{best}
SISVAE	0.42 ± 0.09	0.61 ± 0.30	0.54 ± 0.30	609.0 ± 169.4	
LW-VAE	0.37 ± 0.05	0.39 ± 0.08	0.38 ± 0.07	636.4 ± 55.3	
TSADIS	0.12 ± 0.00	0.07 ± 0.00	1.00 ± 0.00	–	
TeVAE	$\mathbf{0.64 \pm 0.10}$	0.82 ± 0.15	0.55 ± 0.12	642.5 ± 67.9	

The results indicate that this dataset is more challenging compared to the semi-supervised version of the PATH dataset. When τ_{us} is used, TeVAE and VS-VAE perform best and identically relative to the other approaches. While OA, SISVAE and TeVAE perform considerably worse than with the semi-supervised PATH, VS-VAE remains remarkably consistent. VASP, TCN-AE, LW-VAE and TSADIS, on the other hand, actually improve their results on the real-world dataset. Using τ_{best} closes the field significantly. If TSADIS is ignored, the results for the semi-supervised PATH dataset using τ_{best} lie anywhere between $F_1 = 0.12$ and $F_1 = 0.86$, whereas in the case of the real-world dataset between $F_1 = 0.37$ and $F_1 = 0.64$. Furthermore, using τ_{best} leads to modest detection performance improvements across the board, except for TSADIS. TSADIS performs similarly in both the PATH and the real-world dataset with τ_{best}, achieving perfect recall R but poor precision P both times, suggesting that τ_{best} is set very low.

Like with the PATH dataset, these results also indicate that a more refined thresh-

old choice can improve the detection performance.

5.3.2 Ablation Study for Bypass Phenomenon

To find the impact of multi-head attention on TeVAE, its absence is tested on the unsupervised and semi-supervised versions of the PATH dataset. The results obtained are shown in tabular form in Tables 5.4 and 5.5.

Table 5.4: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for TeVAE with and without the multi-head attention mechanism (NoMA) applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
NoMA	$\mathbf{0.03 \pm 0.04}$	0.33 ± 0.38	0.02 ± 0.02	992.7 ± 68.4	τ_{us}
TeVAE	0.02 ± 0.03	0.27 ± 0.36	0.01 ± 0.02	992.7 ± 70.7	
NoMA	0.36 ± 0.09	0.57 ± 0.23	0.28 ± 0.06	817.1 ± 66.1	τ_{best}
TeVAE	$\mathbf{0.58 \pm 0.08}$	0.69 ± 0.12	0.50 ± 0.09	676.4 ± 103.2	

Considering the results previously shown in Table 5.1, it is not surprising that both are essentially useless, with no major discernable difference between TeVAE and NoMA when τ_{us} is used on the unsupervised PATH dataset. Using τ_{best}, however, shows that the presence of the multi-head attention mechanism in TeVAE makes a significant difference. Its absence in NoMA reduces the F_1 score by 38 % and increases the average detection delay by 47 %.

Table 5.5: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for TeVAE with and without the multi-head attention mechanism (NoMA) to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
NoMA	$\mathbf{0.66 \pm 0.20}$	0.96 ± 0.02	0.54 ± 0.24	572.9 ± 193.0	τ_{us}
TeVAE	0.63 ± 0.22	0.95 ± 0.06	0.52 ± 0.26	589.1 ± 212.0	
NoMA	0.77 ± 0.15	0.91 ± 0.07	0.69 ± 0.19	466.9 ± 175.6	τ_{best}
TeVAE	$\mathbf{0.80 \pm 0.15}$	0.88 ± 0.11	0.76 ± 0.18	412.6 ± 143.8	

Applying both variants on the semi-supervised version of the PATH dataset, once again shows that, in the absence of contaminated data, τ_{us} leads to much higher

F_1 scores. Furthermore, it becomes clear that NoMA performs much closer to TeVAE regardless of threshold used, indicating that here, the multi-head attention mechanism plays less of a role. In light of the results using τ_{best} in Table 5.4, however, the multi-head attention mechanism seems to contribute to more robust modelling of nominal behaviour when training subset $\mathcal{D}^{\text{train}}$ is contaminated.

5.3.3 Impact of Reverse-window Method

To measure the potential benefit of the mean-type RWM, it is compared to the first-type and last-type RWMs using both the unsupervised and semi-supervised versions of the PATH dataset. The respective results are shown in Tables 5.6 and 5.7.

Table 5.6: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for different types of RWM applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

RWM	F_1	P	R	$\bar{\delta}$ $[s]$	
first	0.00 ± 0.01	0.11 ± 0.21	0.00 ± 0.00	995.8 ± 69.1	τ_{us}
last	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	995.8 ± 69.1	
mean	$\mathbf{0.02 \pm 0.03}$	0.27 ± 0.36	0.01 ± 0.02	992.7 ± 70.7	
first	0.23 ± 0.05	0.18 ± 0.06	0.39 ± 0.12	833.5 ± 118.6	τ_{best}
last	0.17 ± 0.02	0.11 ± 0.02	0.54 ± 0.17	601.3 ± 128.6	
mean	$\mathbf{0.58 \pm 0.08}$	0.69 ± 0.12	0.50 ± 0.09	676.4 ± 103.2	

As has become a pattern, results obtained using the unsupervised version of the PATH dataset and τ_{us} turn out to be of limited use, as the threshold choice veils any differences between the RWMs. Once τ_{best} is used, a clearer picture emerges, however. Even in this unsupervised best-case, both first-type and last-type methods fall short of their mean-type counterpart, in terms of F_1 score, with the first-type method edging out the last-type method. Interestingly, though unexpected, the average detection delay associated with the last-type method is lower than the mean-type method, which, as explained in Chapter 5.2, can be attributed to the higher recall but also fact that any ground-truth anomalies starting after w time steps can be detected with 0 delay.

While all methods show improvements using the semi-supervised version PATH, it does not affect the pecking order observed in the unsupervised version, regardless of what threshold is used. Additionally, the last-type method no longer features the lowest average detection delay, as its theoretical advantage is offset by the higher recall

Table 5.7: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for different types of RWM applied to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

RWM	F_1	P	R	$\bar{\delta}$ $[s]$	
first	0.45 ± 0.15	0.90 ± 0.10	0.31 ± 0.15	934.9 ± 72.5	τ_{us}
last	0.24 ± 0.10	0.84 ± 0.12	0.15 ± 0.07	882.8 ± 77.9	
mean	$\mathbf{0.63 \pm 0.22}$	0.95 ± 0.06	0.52 ± 0.26	589.1 ± 212.0	
first	0.58 ± 0.12	0.66 ± 0.23	0.57 ± 0.14	854.3 ± 81.5	τ_{best}
last	0.40 ± 0.07	0.53 ± 0.29	0.41 ± 0.12	658.8 ± 104.8	
mean	$\mathbf{0.80 \pm 0.15}$	0.88 ± 0.11	0.76 ± 0.18	412.6 ± 143.8	

and therefore lower false negative count of the mean-type method. An RWM with a lower intrinsic delay is therefore of little use, if fewer anomalies can be detected with it.

5.3.4 Generalisation Study

As the last experiment in this section, TeVAE's ability to generalise is further investigated. As specified in Chapter 5.2, the PATH dataset is split into six different folds, with the former three representing cases missing one of three operational ranges, and the latter three representing cases missing one of three levels of dynamics. In this experiment, PATH is split differently to the previous experiments, and hence the results are only loosely comparable. Note that, since the generalisation capability of TeVAE is of interest in this experiment, only τ_{best} is used, as τ_{us} would mask the results obtained. The respective results for both the unsupervised and semi-supervised versions of the PATH dataset are shown in Tables 5.8 and 5.9.

First, consider the results for the first three folds. As is evident, detection performance is best in fold 1 and drops significantly with the other two folds. In other words, the absence of the low average speed drives affects detection performance the least, whereas the absence of the medium and high average speed drives the most, compared to the *All* fold. One possible explanation could be that, all driving cycles start from rest and end at rest, meaning that even those with high average speeds feature some time steps in the lower ranges, therefore covering most of the operational range. In contrast, low average speed cycles only cover a subspace of the operational range, and need to extrapolate at higher speeds, a task that data-driven models are

Table 5.8: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the hypothetical best threshold τ_{best} only for a range of folds applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in the training subset $\mathcal{D}^{\text{train}}$. Folds one to three represent cases missing sequences of lower, middle, and high average vehicle speeds, respectively, whereas folds four to six represent cases missing sequences of lower, middle, and high levels of dynamics. The so-called *All* fold represents the case where no data is withheld.

Fold	F_1	P	R	$\bar{\delta}$ $[s]$
1	0.51 ± 0.08	0.46 ± 0.08	0.57 ± 0.09	537.0 ± 76.6
2	0.34 ± 0.07	0.27 ± 0.08	0.53 ± 0.15	573.3 ± 110.5
3	0.23 ± 0.02	0.15 ± 0.01	0.52 ± 0.06	567.8 ± 41.0
4	0.49 ± 0.03	0.45 ± 0.00	0.54 ± 0.07	583.4 ± 45.4
5	0.47 ± 0.07	0.42 ± 0.07	0.54 ± 0.08	545.0 ± 71.5
6	0.26 ± 0.01	0.16 ± 0.00	0.64 ± 0.13	484.1 ± 84.2
All	0.55 ± 0.08	0.65 ± 0.11	0.48 ± 0.07	619.8 ± 44.5

particularly weak at. The same pattern can be observed when considering the latter three folds. Clearly, the absence of the most dynamic driving cycles also hurts the detection performance the most, likely for the same reason.

Table 5.9: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the hypothetical best threshold τ_{best} only for a range of folds applied to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. Folds one to three represent cases missing sequences of lower, middle, and high average vehicle speeds, respectively, whereas folds four to six represent cases missing sequences of lower, medium, and high levels of dynamics. The so-called *All* fold represents the case where no data is withheld.

Fold	F_1	P	R	$\bar{\delta}$ $[s]$
1	0.66 ± 0.13	0.59 ± 0.18	0.79 ± 0.02	344.5 ± 9.1
2	0.49 ± 0.06	0.60 ± 0.29	0.56 ± 0.18	519.0 ± 118.0
3	0.43 ± 0.04	0.99 ± 0.01	0.28 ± 0.03	708.7 ± 22.9
4	0.53 ± 0.14	0.48 ± 0.08	0.61 ± 0.22	502.1 ± 164.8
5	0.64 ± 0.02	0.65 ± 0.11	0.67 ± 0.12	437.3 ± 100.1
6	0.38 ± 0.03	0.63 ± 0.32	0.45 ± 0.26	612.4 ± 180.5
All	0.89 ± 0.01	0.95 ± 0.04	0.84 ± 0.05	318.0 ± 12.3

The semi-supervised results tell much of the same story as the unsupervised results. When considering the first three folds, it is once again clear that the model lacking the higher average speed cycles performs worst. Interestingly, for the latter three folds, it

is the one lacking medium levels of dynamics which performs best, whereas in the case of the unsupervised results, it was the one lacking the low levels of dynamics. Another interesting aspect is that the F_1 scores for folds with the low speeds and low levels of dynamics have a much higher standard deviation.

5.4 Conclusions

Despite Table 5.2 featuring the lowest average detection delay in this entire work, the figure still seems fairly high, as 412.6 s corresponds to almost 7 min delay on average. However, recall that through mean-type RWM a intrinsic delay of $w = 256$ time steps is introduced in almost all cases, corresponding to around 2 min by default. Additionally, recall the nature of the online metrics. They not only punish premature detections with an absolute value of negative delay, but also false negatives with the entire length of the anomalous subsequence within the time series, which further adds to the detection delay.

Overall, the unsupervised and semi-supervised results show that, at least for the best performing approaches, the detection performance relies on both the threshold choice, but also the contamination level in $\mathcal{D}^{\text{train}}$. Therefore, further work needs to be done on truly unsupervised threshold estimation methods or other methods that do not assume labelled data is available for free. Additionally, the robustness of model-based approaches when trained on contaminated training subsets leaves room for improvement and should also be addressed.