Unsupervised Anomaly Detection in Energy Time Series Data using Variational Recurrent Autoencoders with Attention

João Pereira
Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal
joao.p.cardoso.pereira@tecnico.ulisboa.pt

Margarida Silveira
Institute for Systems and Robotics, Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal
msilveira@isr.tecnico.ulisboa.pt

***Abstract*—In the age of big data, time series are being generated in massive amounts. In the energy field, smart grids are enabling a unprecedented data acquisition with the integration of sensors and smart devices. In the context of renewable energies, there has been an increasing interest in solar photovoltaic energy generation. These installations are often integrated with smart sensors that measure the energy production. Such amount of data collected makes the quest for developing smart monitoring systems that can detect anomalous behaviour in these systems, trigger alerts and enable maintenance operations.**

In this paper, we propose a generic, unsupervised and scalable framework for anomaly detection in time series data, based on a variational recurrent autoencoder. Furthermore, we introduce attention in the model, by means of a variational self-attention mechanism (VSAM), to improve the performance of the encoding-decoding process. Afterwards, we perform anomaly detection based on the probabilistic reconstruction scores provided by our model.

Our results on solar energy generation time series show the ability of the proposed approach to detect anomalous behaviour in time series data, while providing structured and expressive representations. Since it does not need labels to be trained, our methodology enables new applications for anomaly detection in energy time series data and beyond.

***Index Terms*—Anomaly Detection, Variational Recurrent Autoencoder, Attention, Solar Photovoltaic Energy**

I. Introduction

One of the key assets of the smart grid is the data it collects. The data gathered from smart meters in the grid makes it possible to develop machine learning algorithms that can analyse and monitor the data collected and detect anomalous behaviour. With the integration of renewable energy sources such as solar photovoltaic, it is important to ensure reliability, security and correct operation of these systems in order to promote good performances and a long lifetime of the equipments.

The problem of finding patterns in data that do not conform to expected or normal behaviour is often referred to as Anomaly Detection (AD) [1]. Over time many approaches to anomaly detection have been proposed. In particular, with the progress made in deep learning, new frameworks to tackle the challenges of anomaly detection were developed. However, a significant amount of these approaches are based on supervised machine learning models that require (big) labelled datasets to be trained. In the context of applications such as energy, annotating large datasets is difficult, time-consuming or even too expensive, while it requires domain knowledge from experts in the field. The lack of labels is, indeed, one of the reasons why anomaly detection has been such a great challenge for researchers and practitioners.

Furthermore, some of the proposed methods do not consider the sequential nature of the data by assuming it is independent in time. Smart grid data is often sequential by nature and mostly time series and, hence, it is crucial to take into account the order and structure of the data.

The main contributions of this work can be summarized as follows:

- Unsupervised reconstruction-based model using a variational autoencoder with recurrent encoder and decoder;
- Variational self-attention mechanism to improve the encoding-decoding process;
- Generic framework for anomaly detection in time series data;
- Application to solar photovoltaic generation time series.

II. Background

In this section, we revise autoencoders, recurrent neural networks, attention mechanisms and autoencoder-based anomaly detection.

A. Autoencoder (AE)

Autoencoders [2, 3] are neural networks that aim to reconstruct their input. They consist of two parts: an *encoder* and a *decoder*. The encoder maps input data $\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}}$ to a latent space (or code) $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$ and the decoder maps back from latent space to input space.

The autoencoders training procedure is unsupervised and it consists of finding the parameters that make the reconstruction $\hat{\mathbf{x}}$ as close as possible to the original input $\mathbf{x}$, by minimizing a loss function that measures the quality of the reconstructions (e.g., mean squared error).

Typically the latent space $\mathbf{z}$ has a lower dimensionality than the input space $\mathbf{x}$ and, hence, AEs are forced to learn compressed representations of the input data. This characteristic makes them suitable for dimensionality reduction (DR) tasks, where they were proven to perform much better than other DR techniques, such as Principal Component Analysis [4].

B. Variational Autoencoder (VAE)

The variational autoencoder [5, 6] is a deep generative model that constrains the latent code $\mathbf{z}$ of the conventional AE to be a random variable distributed according to a prior distribution $p_\theta(\mathbf{z})$, usually a standard Normal distribution, $\mathrm{Normal}(\mathbf{0}, \mathbf{I})$. Since the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable for a continuous latent space $\mathbf{z}$, the variational inference technique is often used to find a deterministic approximation, $q_\phi(\mathbf{z}|\mathbf{x})$, of the intractable true posterior. The parameters of the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$, often called the *variational parameters*, are derived using neural networks (e.g., mean $\boldsymbol{\mu}_\mathbf{z}$ and variance $\boldsymbol{\sigma}^2_\mathbf{z}$, in the case of a Normal distribution).
Hence, the training objective of the VAE is to maximize the evidence lower bound (ELBO) on the training data log-likelihood. For a data point $\mathbf{x}$, the evidence lower bound is given by the following equation, where θ and ϕ are the encoder and decoder parameters, respectively.

$$\mathcal{L}_{\mathrm{ELBO}}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})\big)$$

The expectation in the equation above can be approximated by Monte Carlo integration. The second term represents the Kullback-Leibler divergence ($\mathcal{D}_{\mathrm{KL}}$) between the approximate posterior and the prior. The distribution for the likelihood is usually a multivariate Normal or $\mathrm{Bernoulli}$, depending on the type of data being continuous or binary, respectively.

C. RNNs, LSTMs and Bi-LSTMs

Conventional (feed-forward) neural networks make the assumption that data is independent in time. However, this assumption does not hold for sequential data, such as time series. Therefore, with time series data, recurrent neural networks (RNNs) are often used.
RNNs are powerful sequence learners designed to capture the temporal dependencies of the data by introducing *memory*. They read a sequence of input vectors $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ and, at each timestep t, they produce a hidden state $\mathbf{h}_t$. The main feature of RNNs is a feedback connection that establishes a recurrence mechanism that decides how the hidden states $\mathbf{h}_t$ are updated. In simple "vanilla" RNNs, the hidden states $\mathbf{h}_t$ are updated based on the current input, $\mathbf{x}_t$, and the hidden state at the previous timestep, $\mathbf{h}_{t-1}$, as $\mathbf{h}_t = f(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1})$. f is usually a $\tanh$ or $\mathrm{sigmoid}$ function and $\mathbf{U}$ and $\mathbf{W}$ are weight matrices to learn, shared across timesteps. The hidden state $\mathbf{h}_t$ can, thus, be interpreted as a summary of the sequence of input vectors up to timestep t. Given a sequence of hidden states $\mathbf{h}_t$, a RNN can generate an output, $\mathbf{o}_t$, at every timestep or produce a single output, $\mathbf{o}_T$, in the last timestep.

Despite the effectiveness of RNNs for modeling sequential data, they suffer from the vanishing gradient problem, that arises when the output at timestep t depends on inputs much earlier in time. Therefore, long short-term memory networks (LSTMs) [7, 8] were proposed to overcome this problem and they do so by means of a memory cell and three gates. The memory cell ($\mathbf{c}_t$) stores information about the input sequence across timesteps. The *gates* are functions that control the proportion of the current input to include in the memory cell ($\mathbf{i}_t$), the proportion of the previous memory cell to forget ($\mathbf{f}_t$) and the information to output from the current memory cell ($\mathbf{o}_t$). The memory updates, at each timestep t, are computed as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{x}_t) \tag{1}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{x}_t) \tag{2}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{x}_t) \tag{3}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{x}_t) \tag{4}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{5}$$

In the previous equations, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ and $\mathbf{h}_t$ denote the input gate, the forget gate, the output gate, the memory cell and the hidden state, respectively. $\odot$ denotes an element-wise product. The other parameters are weight matrices to be learned, shared between all timesteps.

LSTMs can still not integrate information from future instants of time and, therefore, bidirectional long short-term memory networks (Bi-LSTMs) [9] were proposed. Bi-LSTMs exploit the input sequence, $\mathbf{x}$, in both directions by means of two LSTMs: one executes a forward pass and the other a backward pass. Hence, two hidden states ($\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$) are produced for timestep t, one in each direction. These states act like a summary of the past and the future. The hidden states at similar timesteps are often aggregated into a unique vector $\mathbf{h}_t = \big[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t\big]$ that represents the whole context around timestep t, typically through concatenation.

D. Sequence to Sequence Models and Attention Mechanisms

The sequence to sequence (Seq2Seq) learning framework [10, 11] is often linked with a class of encoder-decoder models, in which the encoder and decoder are RNNs. The encoder reads a variable-length input sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{T_\mathbf{x}}) \in \mathbb{R}^{T_\mathbf{x} \times d_\mathbf{x}}$ and converts it into a fixed-length vector representation (or context vector), $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$, and the decoder takes this vector representation and converts it back into a variable-length sequence $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{T_\mathbf{y}}) \in \mathbb{R}^{T_\mathbf{y} \times d_\mathbf{y}}$. In general, the learned vector representation corresponds to the final hidden state of the encoder network, which acts like a summary of the whole sequence. A particular instance of a Seq2Seq model is the Seq2Seq autoencoder, in which the input and output sequences are aligned in time ($\mathbf{x} = \mathbf{y}$) and, thus, have equal lengths ($T_\mathbf{x} = T_\mathbf{y}$).

Seq2Seq models have their weakness in tackling long sequences (e.g., long time series), mainly because the intermediate vector representation $\mathbf{z}$ does not have enough capacity to capture information of the entire input sequence $\mathbf{x}$.
Therefore, attention mechanisms were proposed to allow the decoder to selectively *attend* to relevant encoded hidden states. Several attention models were proposed in the past few years [12, 13] and, in general, they operate as follows. At each timestep t, during decoding, the attention model computes a context vector $\mathbf{c}_t$ obtained by a weighted sum of all the encoder hidden states. The weights of the sum, a_{ij}, are computed by a score function that measures the similarity between the currently decoded hidden state, $\mathbf{h}^{\mathrm{d}}_t$, and all the

encoded hidden states $\mathbf{h}^{\mathrm{e}} = (\mathbf{h}_1^{\mathrm{e}}, \mathbf{h}_2^{\mathrm{e}}, ..., \mathbf{h}_{T_\mathbf{x}}^{\mathrm{e}})$. Afterwards, these scores are normalized using the $\mathrm{softmax}$ function, so that they sum to 1 along the second dimension. The computation of the weights and the context vectors can be described as follows:

$$a_{ti} = \frac{\exp\left(\mathrm{score}(\mathbf{h}_t^{\mathrm{d}}, \mathbf{h}_i^{\mathrm{e}})\right)}{\sum_{j=1}^{T_\mathbf{x}} \exp\left(\mathrm{score}(\mathbf{h}_t^{\mathrm{d}}, \mathbf{h}_j^{\mathrm{e}})\right)} \tag{6}$$

$$\mathbf{c}_t = \sum_{j=1}^{T_\mathbf{x}} a_{tj} \mathbf{h}_j \tag{7}$$

Attention mechanisms were developed mainly for natural language processing (NLP) tasks and improved significantly the performance of Seq2Seq models in applications such as machine translation [12]. Even though *attention* has been mostly applied to NLP problems involving text data, it is suitable for other tasks dealing with other types of data such as time series and videos. In fact, *attention* is a natural extension of Seq2Seq models for any kind of sequential data.

E. Autoencoder-based Anomaly Detection

The main idea behind autoencoder-based anomaly detection is to focus on what is normal, rather than modelling what is anomalous. The autoencoder is trained to reconstruct data with normal pattern (e.g., normal time series) by minimizing a loss function that measures the quality of the reconstructions. After training, the model is able to reconstruct well data with normal pattern, while it fails reconstruct anomalous data, since it never saw them during training. The detection is performed using the reconstruction metrics (e.g., reconstruction error) as anomaly score. In other words, the model learns a normal data manifold and the distance between a given observation and the normal data manifold is used to compute anomaly scores, either in the latent space of representations, $\mathbf{z}$, or in the reconstructions space, $\hat{\mathbf{x}}$.

III. Related Work

The work on anomaly detection in time series data has increased significantly over the past few years and has benefited from the progress made in deep learning. In particular, Seq2Seq and autoencoder models have been applied with success to time series AD tasks. Using this framework, Malhotra *et al.* [14] proposed a prediction-based model based on LSTMs and used the distribution of the prediction errors to compute anomaly scores. However, this approach is not suitable for time series affected by external factors not captured by sensors, making them unpredictable. Later on, reconstruction-based approaches were proposed to overcome this limitation, such as [15], which try to reconstruct the input time series and use the reconstruction errors as anomaly scores. After the introduction of the variational autoencoder, Bayer and Osendorfer [16] used variational inference and RNNs to model time series data and introduced stochastic recurrent networks (STORNs), which were subsequently applied to anomaly detection in robot time series data [17]. An and Cho [18] proposed a method based on a VAE and introduced a novel probabilistic anomaly score that takes into account the variability of the data (the *reconstruction probability*). Recently, Park *et al.* [19] applied a LSTM-based variational autoencoder to AD in robot assisted feeding data and introduced a progress-based prior over the latent variables. Finally, Xu *et al.* [20] applied a VAE to AD in seasonal key performance indicators (KPIs) time series and provided a theoretical explanation for VAE-based anomaly detection.

IV. Proposed Model

In this section we describe our proposed approach, that relies on two fundamental stages: the reconstruction model (autoencoder) and the detection strategy. Let $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ denote a dataset composed of N independent sequences of observations. Each sequence $\mathbf{x}^{(n)}$ has T timesteps, i.e. $\mathbf{x}^{(n)} = (\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ..., \mathbf{x}_T^{(n)})$, and each observation at timestep t, $\mathbf{x}_t^{(n)}$, is a $d_\mathbf{x}$-dimensional vector. Therefore, the dataset $\mathcal{X}$ has dimensions $(N, T, d_\mathbf{x})$.

A. Variational Bi-LSTM Autoencoder

The model takes as input a sequence of observations $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$. We then apply a denoising autoencoding criterion [21] that consists on adding noise $\mathbf{n} \sim \mathrm{Normal}(\mathbf{0}, \boldsymbol{\sigma}_\mathbf{n}^2 \mathbf{I})$ to the inputs and force the autoencoder to reconstruct the clean version of the input, $\mathbf{x}$, from the corrupted one, $\tilde{\mathbf{x}}$. Since it is a regularization technique, this phase is only active at training time. The encoder is parametrized using a Bi-LSTM with $\tanh$ activation that generates a sequence of hidden states in both directions, forward $\rightarrow$ and backward $\leftarrow$. The final encoder hidden states of both passes are concatenated with each other to produce the vector $\mathbf{h}_T^{\mathrm{e}} = \left[\overrightarrow{\mathbf{h}}_T^{\mathrm{e}}; \overleftarrow{\mathbf{h}}_T^{\mathrm{e}}\right]$.
The prior distribution $p_\theta(\mathbf{z})$ over the latent variables $\mathbf{z}$ is defined as an isotropic multivariate Normal, i.e. $p_\theta(\mathbf{z}) = \mathrm{Normal}(\mathbf{0}, \mathbf{I})$. The parameters of the approximate posterior - the mean $\boldsymbol{\mu}_\mathbf{z}$ and the co-variance $\boldsymbol{\Sigma}_\mathbf{z} = \boldsymbol{\sigma}_\mathbf{z}^2 \mathbf{I}$ - are derived from the final encoder hidden state $\mathbf{h}_T^{\mathrm{e}}$ using two fully connected layers with Linear and $\mathrm{SoftPlus}$ activations, respectively.
To simplify the implementation of the denoising criterion, we adopted the same approach as Park *et al.* [19] and define the approximate posterior given a corruption distribution around $\mathbf{x}$ with a single Gaussian, i.e. $\tilde{q}_\phi(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$, instead of a mixture of Gaussians as in [21]. The latent variables are obtained by sampling from the approximate posterior, $\mathbf{z} \sim \mathrm{Normal}(\boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\sigma}_\mathbf{z}\mathbf{I})$, using the re-parametrization trick $\mathbf{z} = \boldsymbol{\mu}_\mathbf{z} + \boldsymbol{\sigma}_\mathbf{z} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathrm{Normal}(\mathbf{0}, \mathbf{I})$ is an auxiliary noise variable and $\odot$ represents an element-wise product.

Furthermore, we integrate a special attention mechanism in the reconstruction model, that we call Variational Self-Attention Mechanism (VSAM). The self-attention model receives as input a sequence of encoded hidden states and outputs a sequence of context vectors $\mathbf{c}_t$, with the same length (T), each one of them computed as a weighted sum of all the encoded hidden states. In detail, the mechanism works as follows. First, the relevance of every pair of encoded hidden states $\mathbf{h}_i^{\mathrm{e}}$ and $\mathbf{h}_j^{\mathrm{e}}$ is scored (eq. 8) using the *scaled dot-product* similarity, employed in *Transformer* [22] (a neural

network model for NLP, based on a self-attention mechanism). The use of the dot-product as relevance measure makes the self-attention model more efficient than previous attention mechanisms that need to learn a similarity matrix.

$$s_{ij} = \text{score}(\mathbf{h}_i^{\text{e}}, \mathbf{h}_j^{\text{e}}) = \frac{(\mathbf{h}_i^{\text{e}})^T \mathbf{h}_j^{\text{e}}}{\sqrt{d_{\mathbf{h}^{\text{e}}}}} \tag{8}$$

In equation 8, $d_{\mathbf{h}^{\text{e}}}$ is the size of the encoder Bi-LSTM state. Second, the attention weights a_{ij} are computed by normalizing the scores over the second dimension, as in equation 9, where $\mathbf{a}_t = (a_{t1}, a_{t2}, ..., a_{tT})$. This normalisation ensures that, for each timestep t, $\sum_{j=1}^{T} a_{tj} = 1$.

$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t) \tag{9}$$

Finally, for deriving the new context-aware vector representations $\mathbf{c}_t$ we adopted a variational approach. This choice is motivated by the *bypassing phenomenon* pointed out by Bahuleyan *et al.* [23]. In fact, if the decoder has a direct and deterministic access to the encoder hidden states through attention, the latent code $\mathbf{z}$ may not be forced to learn expressive representations, since the self-attention mechanism could bypass most of the information to the decoder. This problem can be solved by applying to the context vectors $\mathbf{c}_t$ the same constraint applied to the latent variables of the VAE, by modelling them as random variables. To do so, we first compute deterministic context vectors, $\mathbf{c}_t^{\text{det}} = \sum_{j=1}^{T} a_{tj}\mathbf{h}_j$ and then transform them using another layer, similarly to [23]. The prior distribution over the context vectors is defined as a standard Normal, $p(\mathbf{c}_t) = \text{Normal}(\mathbf{0}, \mathbf{I})$, and the parameters of the approximate posterior $\tilde{q}_\phi^a(\mathbf{c}_t|\mathbf{x})$, $\boldsymbol{\mu}_{\mathbf{c}_t}$ and $\boldsymbol{\Sigma}_{\mathbf{c}_t}$, are derived in similar fashion to the latent variables $\mathbf{z}$, including the dimensionality ($d_{\mathbf{c}_t} = d_{\mathbf{z}}$). The final context vectors are sampled from the approximate posterior, $\mathbf{c}_t \sim \text{Normal}(\boldsymbol{\mu}_{\mathbf{c}_t}, \boldsymbol{\Sigma}_{\mathbf{c}_t})$.

The decoder is also a Bi-LSTM with $\tanh$ activation that receives, at each timestep t, a latent representation $\mathbf{z}$, shared across timesteps, and a context vector $\mathbf{c}_t$. Unlike other works that use a Normal distribution for $p_\theta(\mathbf{x}_t|\mathbf{z})$, we use a Laplace distribution with parameters $\boldsymbol{\mu}_{\mathbf{x}_t}$ and $\boldsymbol{b}_{\mathbf{x}_t}$. The practical implication of this choice is that the training objective aims to minimize an ℓ_1 reconstruction loss $\propto \|\mathbf{x}_t - \boldsymbol{\mu}_{\mathbf{x}_t}\|_1$ rather than an ℓ_2 reconstruction loss $\propto \|\mathbf{x}_t - \boldsymbol{\mu}_{\mathbf{x}_t}\|_2^2$. The ℓ_1-norm promotes sparse reconstruction errors. Such a choice is motivated by the assumption that anomalous observations are rare and sparse, which is, indeed, the case in several applications of interest. The outputs of the decoder are the parameters of the reconstructed distribution of the input sequence of observations, $\boldsymbol{\mu}_{\mathbf{x}_t}$ and $\boldsymbol{b}_{\mathbf{x}_t}$. These parameters are derived from the decoder Bi-LSTM hidden states using two fully connected layers with Linear and SoftPlus activations, respectively.
The loss function for a particular sequence $\mathbf{x}^{(n)}$ is given by:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}^{(n)}) = &-\mathbb{E}_{\mathbf{z}\sim\tilde{q}_\phi(\mathbf{z}|\mathbf{x}^{(n)}),\mathbf{c}_t\sim\tilde{q}_\phi^a(\mathbf{c}_t|\mathbf{x}^{(n)})}\left[\log p_\theta(\mathbf{x}^{(n)}|\mathbf{z}, \mathbf{c})\right] \\ &+ \lambda_{\text{KL}}\Big[\mathcal{D}_{\text{KL}}\Big(\tilde{q}_\phi(\mathbf{z}|\mathbf{x}^{(n)})\|p_\theta(\mathbf{z})\Big) \\ &+ \eta\sum_{t=1}^{T}\mathcal{D}_{\text{KL}}\Big(\tilde{q}_\phi^a(\mathbf{c}_t|\mathbf{x}^{(n)})\|p(\mathbf{c}_t)\Big)\Big]\end{aligned}$$

where λ_{KL} weights the reconstruction and KL losses and η balances the attention KL loss and the latent space KL loss. Figure 1 illustrates the proposed model.

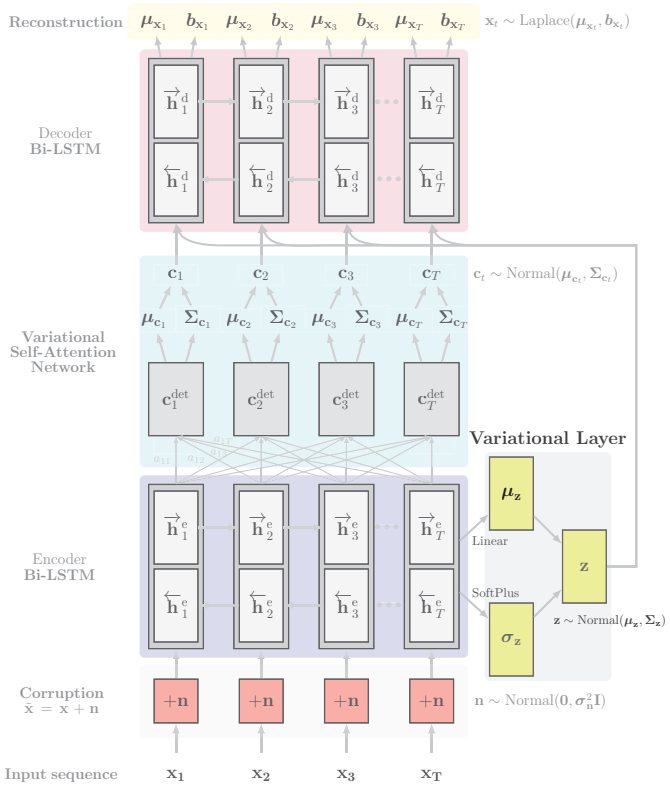

Fig. 1. Variational Bi-LSTM Autoencoder with Variational Self-Attention.

B. Anomaly Detection

The anomaly detection strategy is based on the following principle. The Variational Bi-LSTM Autoencoder with Attention is trained on normal sequences, so that it learns the normal pattern of data. At test time, normal sequences are expected to be well reconstructed whereas anomalous ones are not, since the model has not seen anomalous data during training.

Unlike deterministic autoencoders, the proposed model based on a VAE reconstructs the distribution parameters (mean $\boldsymbol{\mu}_{\mathbf{x}}$ and diversity $\boldsymbol{b}_{\mathbf{x}}$) of the input variable rather than the input variable itself. Therefore, it is possible to use probability measures as anomaly scores. One approach is to compute the *reconstruction probability*, introduced by An and Cho [18], that is an estimation of the reconstruction term of the VAE loss function by Monte Carlo integration.

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] \approx \frac{1}{L}\sum_{l=1}^{L}\log p(\mathbf{x}|\mathbf{z}_l)$$

The process can be described as follows. First, an input sequence $\mathbf{x}$ is propagated through the encoder and the posterior parameters $\boldsymbol{\mu}_{\mathbf{z}}$ and $\boldsymbol{\Sigma}_{\mathbf{z}}$ are obtained in a fully deterministic fashion. Then, L samples are drawn from an isotropic Gaussian distribution with these parameters. Each sample $\mathbf{z}_l$ is

propagated though the decoder that outputs the distribution parameters of the reconstruction. Afterwards, the log-likelihood of the input sample $\mathbf{x}$, given a latent code $\mathbf{z}_l$ drawn from the approximate posterior distribution is computed. Finally, the reconstruction probability is averaged over all $\mathbf{z}$ samples. Algorithm 1 summarizes the computation process.

Algorithm 1 Reconstruction Probability Score

Input: $\mathbf{x} \in \mathbb{R}^{T \times d_{\mathbf{x}}}$
Output: $ReconstructionProbability \in \mathbb{R}^{T}$
$(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{z}}) \leftarrow \text{Encoder}(\mathbf{x})$
for $l = 1$ to L **do**
  $\mathbf{z}_l \sim \text{Normal}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{z}})$
  $(\boldsymbol{\mu}_{\mathbf{x}}^{l}, \mathbf{b}_{\mathbf{x}}^{l}) \leftarrow \text{Decoder}(\mathbf{z}_l)$
  $\text{score}^{l} \leftarrow \log p(\mathbf{x}|\boldsymbol{\mu}_{\mathbf{x}}^{l}, \mathbf{b}_{\mathbf{x}}^{l})$
end for
$ReconstructionProbability \leftarrow \frac{1}{L}\sum_{l=1}^{L} \text{score}^{l}$
return $ReconstructionProbability$

The anomaly score itself is the negative reconstruction probability, so that the lower the reconstruction probability, the higher the anomaly score. There are several advantages of using the reconstruction probability instead of a deterministic reconstruction error which is commonly used in autoencoder-based anomaly detection approaches. The first one is that the reconstruction probability does not requires data-specific detection thresholds, since it is a probabilistic measure. Using such a metric provides a more intuitive way of analysing the results. The second one is that the reconstruction probability takes into account the variability of the data. Intuitively, anomalous data has higher variance than normal data and, hence, the reconstruction probability is likely to be lower for anomalous examples. The idea of using the variability of data for anomaly detection enriches the expressive power of the proposed model relatively to conventional autoencoders. Even in the case where normal and anomalous data can share the same expected value, the variability is different and, thus, provide an extra tool to distinguish anomalous examples from normal ones. For comparison purposes we also compute a (stochastic) reconstruction error (RE), given by equation 10.

$$RE_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}(\mathbf{x}) = \frac{1}{L}\sum_{l=1}^{L} \left\| \mathbf{x} - \underbrace{\mathbb{E}\left[p_{\theta}\left(\mathbf{x}_l|\mathbf{z}_l\right)\right]}_{\boldsymbol{\mu}_{\mathbf{x}_l}} \right\|_1 \tag{10}$$

V. Training Framework

A. Data

The energy data in this work is a dataset $\mathcal{X}$ of univariate time series ($d_{\mathbf{x}} = 1$) of solar photovoltaic (PV) energy generation from several residential installations. The training data was obtained by selecting a subset $\mathcal{X}^{\text{normal}}$ of 1430 daily sequences with normal pattern (days without clouds and any kind of anomaly, where the energy generated is as expected). Samples were recorded each 15 min and, therefore, each (daily) sequence as 96 observations. The solar PV curves have a strong seasonality, with a predominant seasonal period of a day. We divided our dataset of normal sequences into two subsets - a training set $\mathcal{X}_{\text{train}}^{\text{normal}}$ and a validation set $\mathcal{X}_{\text{val}}^{\text{normal}}$ - with a splitting ratio of 80/20, respectively. The data was also normalised to the installed capacity, so that the range of observed values lies in the interval $[0, 1]$.

B. Modes

The proposed approach for anomaly detection can work under the following two modes:

- *Off-line Mode*: Training is performed with non-overlapping sequences of length T and the observations within a sequence share a unique representation in the latent space $\mathbf{z}$. All the scores for an input window are considered for detection and the score at a particular timestep t in a window can depend on future observations within the same window.
- *On-line Mode*: Training is executed using overlapping sequences obtained with a sliding window with a width T and a step size of 1. At test time, detection is performed without considering observations of future time instants, by feeding to the model a window of observations in which the last point corresponds to the current timestep t. The anomaly score at timestep t corresponds to the score of the last observation within each sequence. In this mode, for a long sequence with length L, $L - T + 1$ windows are produced, each one of them having its own representation in the $\mathbf{z}$-space. Since these windows overlap, the latent space will exhibit trajectories over time.

C. Optimization and Regularization

The models were implemented using the Keras deep learning library for Python [24], running on top of TensorFlow [25]. Optimization was performed using *AMS-Grad* optimizer [26], a variant of *Adam* [27], in mini-batches of size 200 (off-line mode) and 10000 (on-line mode), during 1500 epochs. The learning rate was 0.001. The full model has 274.958 parameters to optimize. We set the latent space dimensionality ($d_{\mathbf{z}}$) and the context vectors dimensionality ($d_{\mathbf{c}_t}$) to 3. The encoder and decoder Bi-LSTM both have 256 units, 128 in each direction. The noise added at the input level for the denoising autoencoding criterion has variance $\sigma_{\mathbf{n}}^2 = 0.1\sigma_{\mathbf{x}}^2$. We set the number L of Monte Carlo samples to 1 during training, following the work of Kingma and Welling [5]. The gradients were clipped by value with a clip value of 1.0. To prevent the KL-divergence vanishing problem, we applied a KL-annealing scheme [28] that consists on varying the weight λ_{KL} during training. By doing so, λ_{KL} is initially close to zero in order to allow accurate reconstructions in the early stages of training and is gradually increased to promote smooth encodings and diversity. The parameter η is 0.01. We also apply a sparsity regularizer in the hidden layer of the encoder Bi-LSTM [29], that penalizes the ℓ_1-norm of the activations with a weight of 10^{-8}.

Training was done on a single NVIDIA GTX 1080 TI GPU

with 11GB of memory, in a machine with an 8th generation i7 processor and 16GB of DDR4 RAM.

VI. Experiments and Results

In this section, we present the results of the experiments obtained with our proposed model. To illustrate the effectiveness of our approach, a few examples of solar energy generation curves representative of different patterns and behaviours ($\mathcal{X}_{\text{test}}$) were annotated, such as a normal sequence used as ground truth, a brief shading, a fault, a spike anomaly, an example of a daily curve where snow covered the surface of the PV panel and a sequence corresponding to a cloudy day.

We evaluate the training results using the training and validation losses, presented in Table I.

TABLE I
Training and validation losses.

Set	**Training** $(\mathcal{X}_{\text{train}}^{\text{normal}})$	**Validation** $(\mathcal{X}_{\text{val}}^{\text{normal}})$
Loss	-3.1457	-3.1169

The training and validation losses are similar, meaning that the model is not over-fitting to the training data and is being able to generalize to unseen (normal) sequences.

A. Anomaly Scores

Figure 2 shows some examples of solar PV generation daily curves with different kinds of patterns and the corresponding anomaly scores: the reconstruction probability (top bar) and the reconstruction error (bottom bar), both obtained by Monte Carlo integration using $L = 512$ samples.

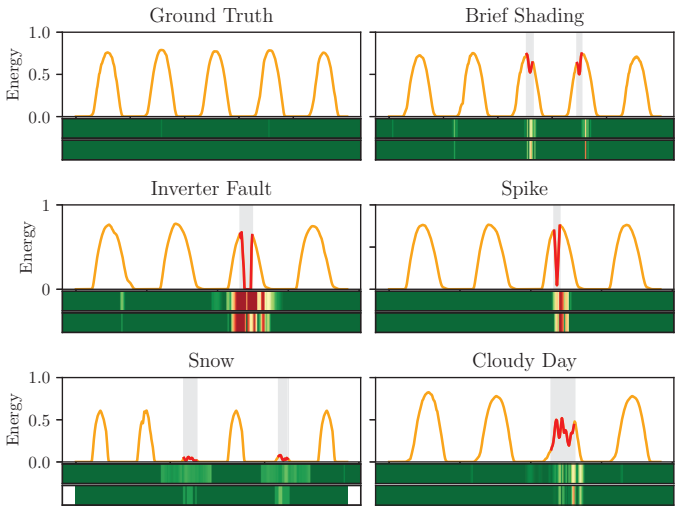

Fig. 2. Anomaly scores for some representative sequences (off-line mode, non-overlapping sequences with $T = 96$ timesteps).

B. Latent Space Analysis

The experiments were performed using a 3-dimensional latent space ($d_{\mathbf{z}} = 3$). For visualization purposes, we reduced the dimensionality of the latent space to 2D using Principal Component Analysis (PCA) and t-distributed Stochastic Neighbour Embedding (t-SNE) [30]. Figure 3 represents the latent space $\mathbf{z}$ of the training set containing only normal sequences ($\mathcal{X}_{\text{train}}^{\text{normal}}$). The label corresponds to the time instant of the last observation within each sequence.

Fig. 3. Latent space visualization of $\mathcal{X}_{\text{train}}^{\text{normal}}$ in 2D via t-SNE (left) and PCA (right). (on-line mode, training executed using overlapping sequences with $T = 32$ timesteps).

The latent space shows evidence that the model is mapping sequences aligned in time onto the same region of the $\mathbf{z}$-space and, more interestingly, it reveals a cyclic trajectory whose period matches exactly the seasonal period of the solar PV curves: one day. In other words, the model has learned the seasonal property of the data without being told of it and using training sequences with a length $32 < 96$, shuffled during training. Previous works have shown latent spaces with this behaviour, even though without analysing it, until the recent work of Xu *et al.* [20] that provided for the first time an explanation for this effect that they called *Time Gradient*.

In the context of time series anomaly detection, it is interesting to exploit the latent representations to find out how the representations of anomalous data compare with the ones of normal examples. Figure 4 shows the representations of the sequences that we annotated. Since the variational latent space is obtained by sampling from the approximate posterior, in this plot we represent the mean $\boldsymbol{\mu}_{\mathbf{z}} = \mathbb{E}\big[q_\phi(\mathbf{z}|\mathbf{x})\big]$ space, which is deterministically obtained from the encoder Bi-LSTM output.

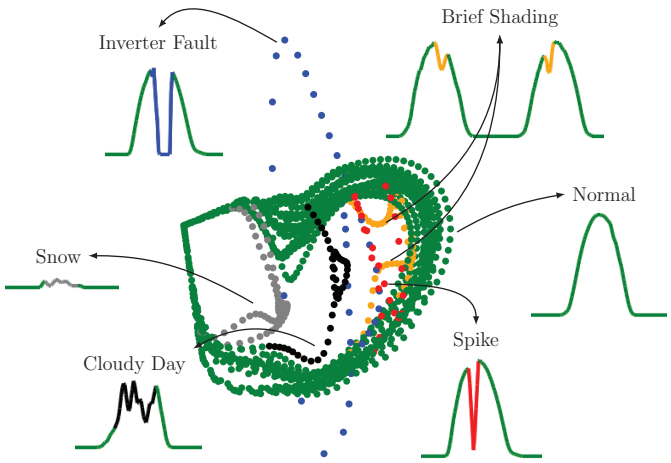

Fig. 4. Latent space visualization of $\mathcal{X}_{\text{test}}$ in 2D via PCA (on-line mode, training executed using overlapping sequences with $T = 32$ timesteps).

Figure 4 shows structured and expressive representations of sequences with various patterns. The *normal* examples (green) and the *anomalous* ones are represented differently in the space and there is clear a deviation of anomalous windows from the normal trajectory. The normal data have also slightly different trajectories in the space mainly because even though the curves have the same qualitative (normal) pattern, they are shifted in time due to different locations of the installations where the sun starts shining on the PV panel at different moments and also due to different inclinations.

C. Attention Visualization

The Variational Self-Attention Mechanism learns to pay more attention to particular encoded hidden states. Therefore, the attention model produces a 2D map for each sequence, with length T, that shows where the network is *putting* its attention. Figure 5 shows the attention maps for different test sequences with and without anomalies.

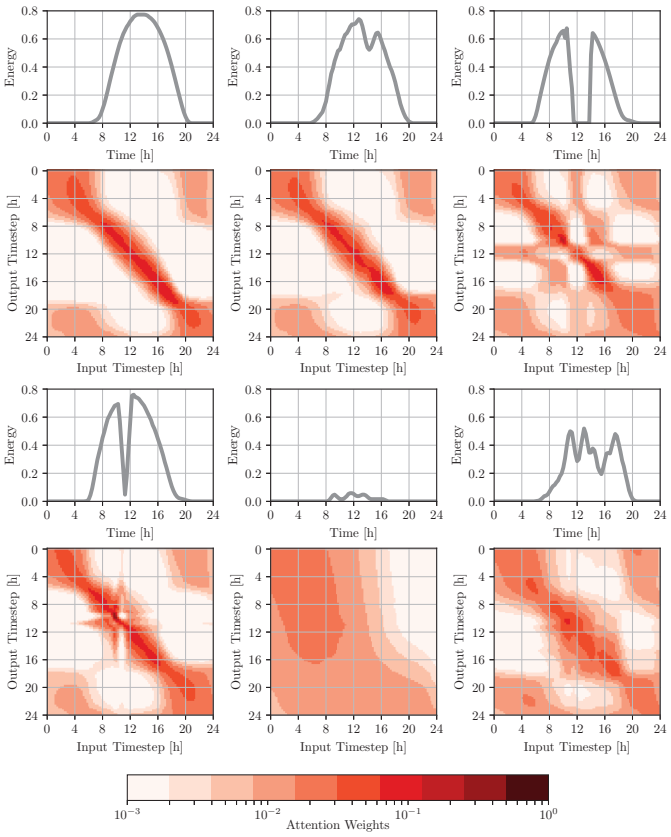

Fig. 5. Attention maps for sequences with different patterns. The attention weights are represented in a logarithmic scale.

The attention maps show evidence that the self-attention model is producing context-aware representations, which can be seen by the distribution of the attention weights in a small window around the first diagonal of the maps. This result supports the intuition that most of the temporal context of an observation in a time series lies in a narrow window around it. Furthermore, for different anomalies, the maps show different distributions of the attention weights. In some cases, the self-attention model is capturing dependencies between hidden states far in time. This conclusion validates the proposed reconstruction-based anomaly detection approach, since it tells that the network struggles to reconstruct well anomalous sequences while it tries to capture long-term dependencies in those.

It is also possible to visualize the context vectors $\mathbf{c}_t^{\text{det}}$ in the mean space $\boldsymbol{\mu}_{\mathbf{c}_t}$. The visualization, shown in Figure 6, was performed by reducing the dimensionality of $\boldsymbol{\mu}_{\mathbf{c}_t}$ to 2D using PCA. The labels represent the corresponding time instant t. Each context vector is computed as a weighted sum of all the encoder hidden states, so each one of them combines information from different time instants.

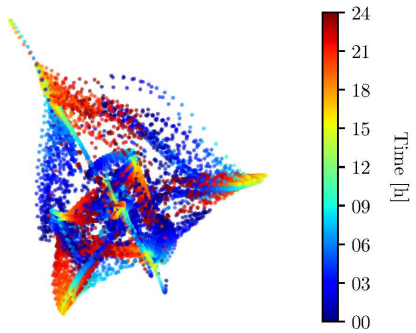

Fig. 6. Context vectors of the validation set $\mathcal{X}_{\text{val}}^{\text{normal}}$.

Figure 6 shows that context vectors aligned in time tend to be roughly represented in the same region of the space, while the mixed structure suggests that different sequences lead to context vectors that combine the encoder hidden states differently, using different attention weights.

VII. Conclusions and Future Work

In this paper, we presented a generic, unsupervised and scalable framework for anomaly detection in time series data that can operate both off-line and on-line. Our approach consists of a reconstruction model based on a variational autoencoder. We parametrized the encoder and decoder with recurrent neural networks to take into account the temporal dependencies of time series data. We also proposed a variational self-attention mechanism that aids the decoding process by allowing the model to pay more attention to particular encoded hidden states and, at the same time, provides a straightforward visualization scheme for the sequences.

Our results show that the model is able to detect anomalous patterns by using the probabilistic reconstruction metrics as anomaly scores. Moreover, the attention maps show evidence that the model changes its attention according to the kind of pattern of the input sequence. In particular, it *attends* differently depending on whether the sequence is normal or anomalous. A future line of work can exploit the usefulness of the attention maps for detection.

Furthermore, even though we applied the proposed model to solar PV generation univariate time series, it is suitable to

multivariate data as well, in which $\mathbf{x}$ can be a $d_{\mathbf{x}}$-dimensional vector of observations. Moreover, it can even be applied to other types of sequential data beyond time series, such as text and videos.

One of the major challenges of this work was the full absence of labels that is, actually, a common scenario in the context of real-world applications, such as energy. This motivated the unsupervised framework for anomaly detection that we proposed. The main advantage of following such an approach is that it can be applied to a wide range of time series data available. On the other hand, the main difficulty that we found due to the lack of labels was evaluation, since it is not possible to compute conventional classification metrics under this scenario. In fact, evaluation metrics for unsupervised anomaly detection algorithms, in the absence of labels and ground truth, remains a challenging problem where the literature is still scarce, even though some recent work has been done on the subject [31].

In this work, we focused on assigning an anomaly score to every observation in a sequence and not discriminating between different anomalies. However, the proposed approach can be extended to a multi-class framework, to allow distinguishing between anomalies. For this purpose, the detection phase might take into account the representations learned in the $\mathbf{z}$-space, which reveal to be expressive enough to allow for such a scenario.

Finally, in unsupervised anomaly detection, the concept of *normality* turns out to be hard to define in formal terms and might be prone to change/drift over time. Dealing with concept drift is a subject that we intend to address in future work.

ACKNOWLEDGEMENT

This work was funded by FCT project UID/EEA/50009/2013.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, pp. 533–536, 1986.

[3] H. Bourlard and Y. Kamp, "Auto-association by Multilayer Perceptrons and Singular Value Decomposition," *Biological Cybernetics*, vol. 59, no. 4, pp. 291–294, Sep 1988. [Online]. Available: https://doi.org/10.1007/BF00332918

[4] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[5] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *CoRR*, vol. abs/1312.6114, 2013. [Online]. Available: http://arxiv.org/abs/1312.6114

[6] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1278–1286. [Online]. Available: http://proceedings.mlr.press/v32/rezende14.html

[7] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[8] A. Graves, "Generating Sequences With Recurrent Neural Networks," *CoRR*, vol. abs/1308.0850, 2013.

[9] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition," in *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ser. ICANN'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 799–804. [Online]. Available: http://dl.acm.org/citation.cfm?id=1986079.1986220

[10] I. Sutskever, Q. V. Le, and O. Vinyals, "Sequence to Sequence Learning with Neural Networks," *CoRR*, vol. abs/1409.3215, 2014.

[11] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *CoRR*, vol. abs/1406.1078, 2014.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

[13] M. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," *CoRR*, vol. abs/1508.04025, 2015.

[14] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," in *Proceedings of the 23rd European Symposium on Artificial Neural Networks*, 2015.

[15] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection," *CoRR*, vol. abs/1607.00148, 2016.

[16] J. Bayer and C. Osendorfer, "Learning Stochastic Recurrent Networks," *ArXiv e-prints*, Nov. 2014.

[17] M. Sölch, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational Inference for On-line Anomaly Detection in High-Dimensional Time Series," *CoRR*, vol. abs/1602.07109, 2016.

[18] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," *CoRR*, vol. 2015-2, 2015.

[19] D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder," *CoRR*, vol. abs/1711.00614, 2017.

[20] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1802.03903

[21] Y. Bengio, D. J. Im, S. Ahn, and R. Memisevic, "Denoising Criterion for Variational Auto-Encoding Framework," *CoRR*, vol. abs/1511.06406, 2015.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017.

[23] H. Bahuleyan, L. Mou, O. Vechtomova, and P. Poupart, "Variational Attention for Sequence-to-Sequence Models," *CoRR*, vol. abs/1712.08207, 2017.

[24] F. Chollet, "Keras," https://keras.io, 2015.

[25] M. Abadi *et al.*, " TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[26] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[28] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," *CoRR*, vol. abs/1511.06349, 2015.

[29] D. Arpit, Y. Zhou, H. Ngo, and V. Govindaraju, "Why Regularized Auto-Encoders learn Sparse Representation?" in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 136–144. [Online]. Available: http://proceedings.mlr.press/v48/arpita16.html

[30] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[31] N. Goix, "How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?" *ArXiv e-prints*, 2016.