RECURRENT AUTO-ENCODER MODEL FOR MULTIDIMENSIONAL TIME SERIES REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent auto-encoder model can summarise sequential data through an encoder structure into a fixed-length vector and then reconstruct into its original sequential form through the decoder structure. The summarised information can be used to represent time series features. In this paper, we propose relaxing the dimensionality of the decoder output so that it performs partial reconstruction. The fixed-length vector can therefore represent features only in the selected dimensions. In addition, we propose using rolling fixed window approach to generate samples. The change of time series features over time can be summarised as a smooth trajectory path. The fixed-length vectors are further analysed through additional visualisation and unsupervised clustering techniques. This proposed method can be applied in large-scale industrial processes for sensors signal analysis purpose.

1 BACKGROUND

Modern industrial processes are often monitored by a large array of sensors. Machine learning techniques can be used to analyse streams of signal data in an on-line scenario. This paper illustrates this using propietary data collected from a large-scale industrial production system. Sensors are attached to various parts of the system to monitor the production process. Vital real-valued measurements like temperature, pressure, rotary speed, vibration... etc., are recorded at different locations.

The system can be treated as a multidimensional entity changing through time. Each stream of sensor measurement is basically a set of real values received in a time-ordered fashion. When this concept is extended to a process with P sensors, the dataset can therefore be expressed as a time-ordered multidimensional vector $\{\mathbb{R}_t^P : t \in [1, T]\}$.

2 METHODS

A pair of RNN encoder-decoder structure can provide end-to-end mapping between an ordered multidimensional input sequence and its matching output sequence (Sutskever et al., 2014; Cho et al., 2014). Recurrent auto-encoder can be depicted as a special case of the aforementioned model, where input and output sequences are aligned with each other. It can be extended to the area of signal analysis in order to leverage recurrent neurons power to understand complex and time-dependent relationship.

2.1 ENCODER-DECODER STRUCTURE

At high level, the RNN encoder reads an input sequence and summarises all information into a fixed-length vector. The decoder then reads the vector and reconstructs the original sequence. Figure 1 below illustrates the model.

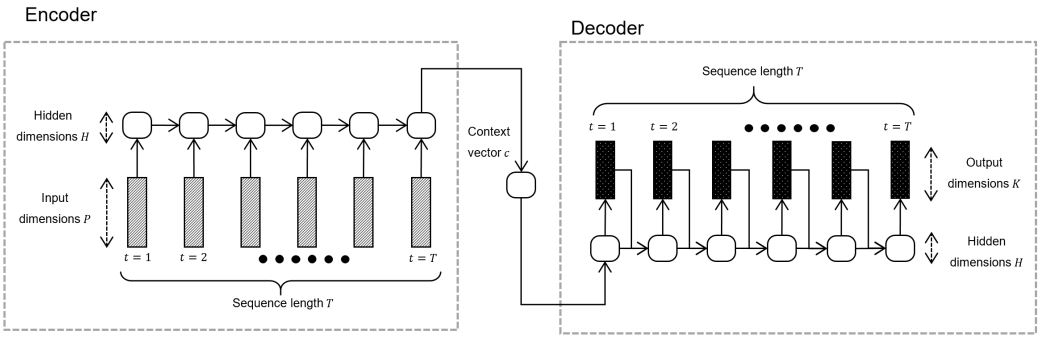


Figure 1: Recurrent auto-encoder model. Both the encoder and decoder are made up of multilayered RNN. Arrows indicate the direction of information flow.

2.1.1 ENCODING

The role of the recurrent encoder is to project the multidimensional input sequence into a fixed-length hidden context vector c. It reads the input vectors $\{\mathbb{R}_t^P : t \in [1, T]\}$ sequentially from $t = 1, 2, 3, ..., T$. The hidden state of the RNN has H dimensions which updates at every time step based on the current input and hidden state inherited from previous step.

Recurrent neurons arranged in multiple layers are capable of learning complex temporal behaviours. In most RNN models, LSTM neurons are usually used (Hochreiter & Schmidhuber, 1997). Nevertheless, Cho et al. (2014) suggested using gated recurrent unit (GRU) neuron which may improve training efficiency. Once the encoder reads all the input information, the sequence is summarised in a fixed-length vector c which has H hidden dimensions.

For regularisation purpose, dropout can be applied to avoid overfitting. It refers to randomly removing a fraction of neurons during training, which aims at making the network more generalisable (Srivastava et al., 2014). In an RNN setting, Zaremba et al. (2014) suggested that dropout should only be applied non-recurrent connections. This helps the recurrent neurons to retain memory through time while still allowing the non-recurrent connections to benefit from regularisation.

2.1.2 DECODING

The decoder is a recurrent network which reconstructs the context vector c back into the original sequence. To exemplify this, the decoder starts by reading the context vector c at $t = 1$. It then decodes the information through the RNN structure and outputs a sequence of vectors $\{\mathbb{R}_t^K : t \in [1, T]\}$ where K denotes the dimensionality of the output sequence.

Recalling one of the fundamental characteristics of an autoencoder is the ability to reconstruct the input data back into itself via a pair of encoder-decoder structure. This criterion can be slightly relaxed such that $K \leqslant P$, which means the output sequence is only a partial reconstruction of the input sequence.

Recurrent auto-encoder with partial reconstruction:

$$\begin{cases} f_{encoder} : \{\mathbb{R}_t^P : t \in [1, T]\} \rightarrow c \\ f_{decoder} : c \rightarrow \{\mathbb{R}_t^K : t \in [1, T]\} \end{cases} \quad K \leqslant P \tag{1}$$

In the large-scale industrial system use case, all streams of sensor measurements are included in the input dimensions while only a subset of sensors is included in the output dimensions. This means the entire system is visible to the encoder, but the decoder only needs to perform partial reconstruction of it. End-to-end training of the relaxed autoencoder implies that the context vector would summarise the input sequence while still being conditioned on the output sequence. Given that activation of the context vector is conditional on the decoder output, this approach allows the encoder to capture lead variables across the entire process as long as they are relevant to the selected output dimensions.

Having a subset of dimensions in the output sequence has practical significance for industrial application of the algorithm. In practice, multiple context vectors can be generated from multiple

recurrent auto-encoder models using identical sensors in the encoder input but different subset of sensors in the decoder output. Various subsets of sensors can reflect the underlying states of different parts of the system. As a result, these context vectors produced from the same time segment can be used as different diagnostic measurements in industrial context.

2.2 SAMPLING

For a training dataset of T' time steps, samples can be generated where $T < T'$. We can begin at $t = 1$ and draw a sample of length T. This process continues recursively by shifting one time step until it reaches the end of training dataset. For a subset sequence of length T, this method allows $T' - T$ samples to be generated. Besides, it can also generate samples from an unbounded time series in an on-line scenrio, which are essential for time-critical applications like sensor data analysis.

Algorithm 1: Drawing samples consecutively from the original dataset

Input: Dataset length T'
Input: Sample length T
$i \leftarrow 0$;
while $i \leqslant i + T$ **do**
  Generate sample sequence $(i, i + T]$ from the dataset;
  $i \leftarrow i + 1$;
end

Given that sample sequences are recursively generated by shifting the window by one time step, successively-generated sequences are highly correlated with each other. As we have discussed previously, the RNN encoder structure compresses sequential data into a fixed-length vector representation. This means that when consecutively-drawn sequences are fed through the encoder structure, the resulting activation at c would also be highly correlated. As a result, consecutive context vectors can join up to form a smooth trajectory in space.

Context vectors in the same neighbourhood have similar activation therefore they must have similar underlying states. Contrarily, context vectors located in distant neighbourhoods would have different underlying states. These context vectors can be visualised in lower dimensions via dimensionality reduction techniques such as principal component analysis (PCA).

Furthermore, additional unsupervised clustering algorithms can be applied to the context vectors. Each context vector can be assigned to a cluster C_j where J is the total number of clusters. Once all the context vectors are labelled with their corresponding clusters, supervised classification algorithms can be used to learn the relationship between them using the training set. For instance, support vector machine (SVM) classifier with J classes can be used. The trained classifier can then be applied to the context vectors in the held-out validation set for cluster assignment. It can also be applied to context vectors generated from unbounded time series in an on-line setting. Change in cluster assignment among successive context vectors indicate change in the underlying state.

3 RESULTS

Training samples were drawn from the dataset using windowing approach with fixed sequence length. In our example, the large-scale industrial system has 158 sensors which means the recurrent auto-encoder's input dimension has $P = 158$. The model's sequence has fixed length $T = 36$, while samples were drawn from the dataset with total size $T' = 2724$. The dataset was scaled into z-scores, thus ensuring zero-centred data which facilitates gradient-based training.

3.1 OUTPUT DIMENSIONITY

As we discussed earlier, the RNN decoder's output dimension can be relaxed for partial reconstruction. The output dimensionality was set at $K = 6$ which is comprised of sensors relating to key pressure measurements.

We have experimented three scenarios where the first two have complete dimensionality $P = 158; K = 158$ and $P = 6; K = 6$ while the remaining scenario has relaxed dimensionality

$P = 158; K = 6$. The training and validation MSEs of these models are visualised in figure 2 below.

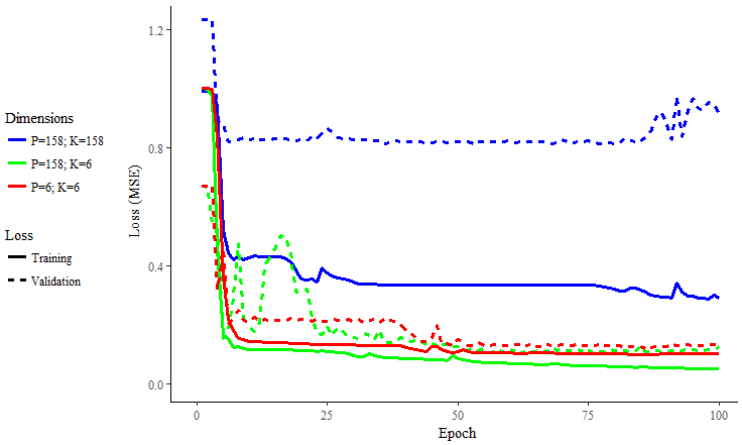


Figure 2: Effects of relaxing dimensionality of the output sequence on the training and validation MSE losses. Several sets of auto-encoder models were trained using Adam optimiser (Kingma & Ba, 2014). They contain same number of layers in the RNN encoder and decoder respectively. All hidden layers contain same number of LSTM neurons with hyperbolic tangent activation.

The first model with complete dimensionality ($P = 158; K = 158$) has visibility of all dimensions in both the encoder and decoder structures. Yet, both the training and validation MSEs are high as the model struggles to compress-decompress the high dimensional time series data.

For the complete dimensionality model with $P = 6; K = 6$, the model has limited visibility to the system as only the selected dimensions were included. Despite the context layer summarises information specific to the selected dimensionality in this case, lead variables in the original dimensions have been excluded. This prevents the model from learning any dependent behaviours among all available information.

On the other hand, the model with partial reconstruction ($P = 158; K = 6$) demonstrate substantially lower training and validation MSEs. As all information is available to the model via the RNN encoder, it captures all relevant information such as lead variables across the entire system.

Randomly selected samples in the held-out validation set were fed to this model and the predictions can be qualitatively examined in details. In figure 3 below, all the selected specimens demonstrate similarity between the original label and the reconstructed output. The recurrent auto-encoder model captures the shift mean level as well as temporal variations of all the output dimensions.

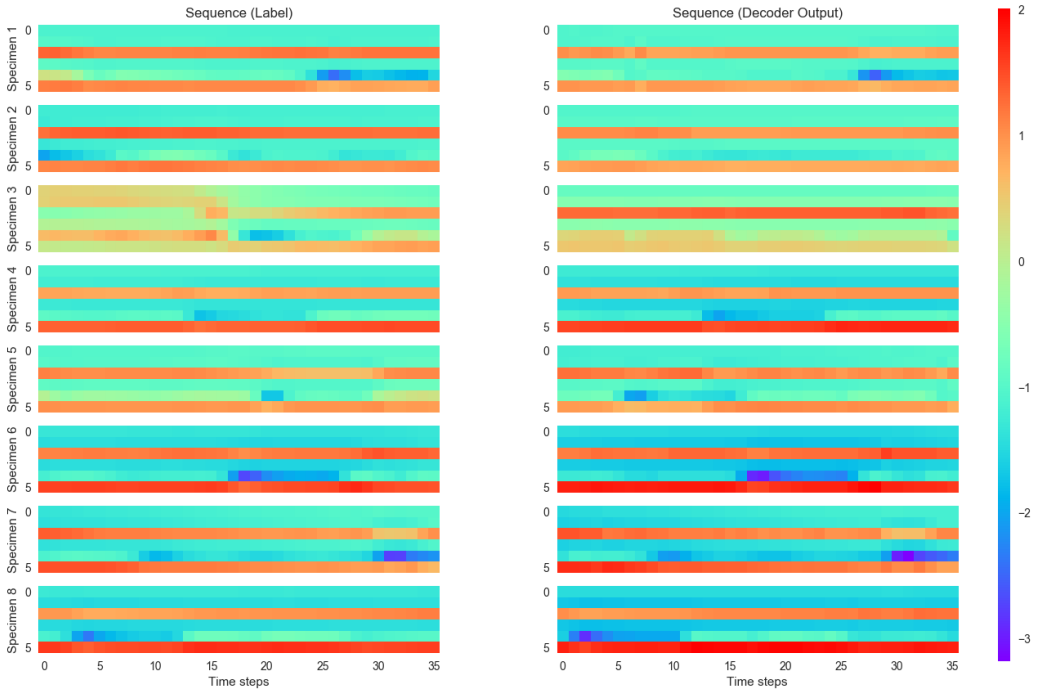


Figure 3: A heatmap showing eight randomly selected output sequences in the held-out validation set. Colour represents magnitude of sensor measurements in normalised scale.

3.2 Context Vector

Once the recurrent auto-encoder model is successfully trained, samples can be fed to the model and the corresponding context vectors can be extracted for detailed inspection. As we discussed earlier, successive context vectors have similar activation as they are only shifted by one time step. By calculating the correlation matrix of all context vectors and visualising them on a heatmap as in figure 4, it is found that the narrow band around the diagonal has consistently higher correlation. This indicates that successive context vectors are highly correlated.

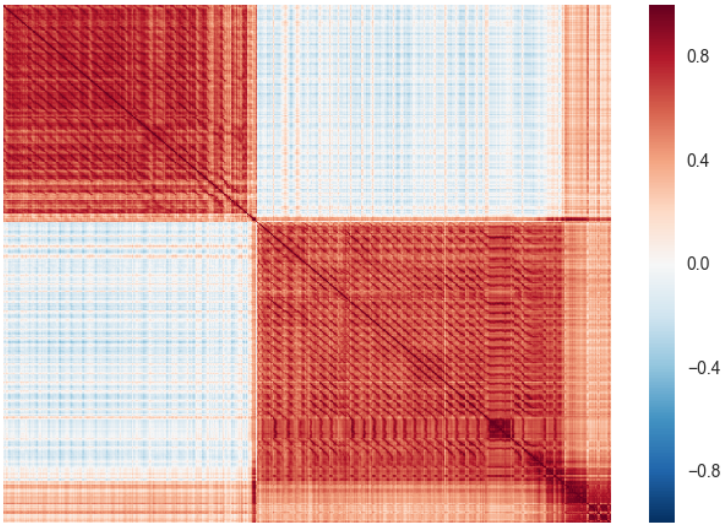
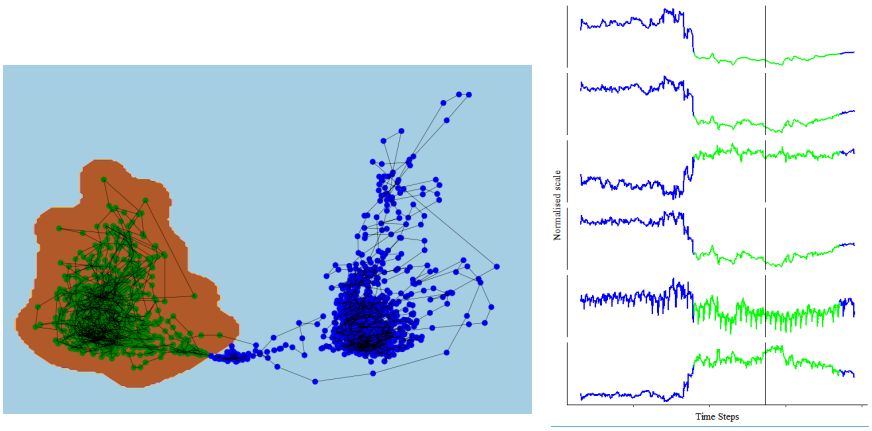


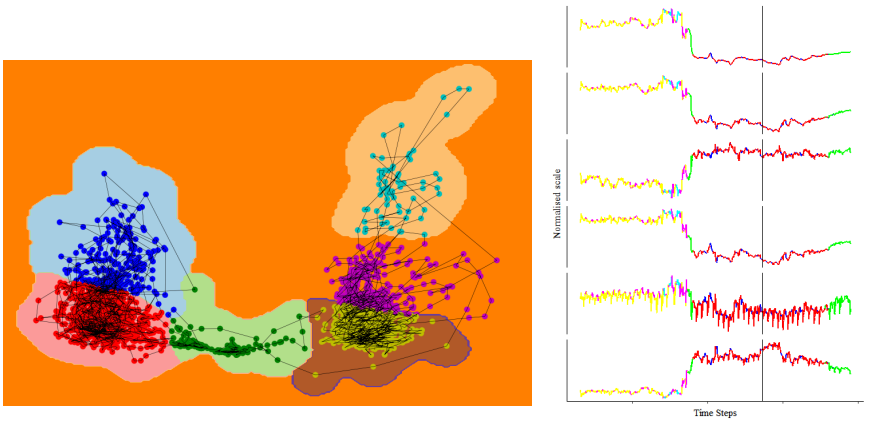
Figure 4: A correlation matrix showing the pairwise correlation of all context vectors. Notice the narrow band around the diagonal always has stronger correlation.

In the model we selected, the context vector c is a multi-dimensional real vector $\mathbb{R}^{400}$. Since the model has input dimensions $P = 158$ and sequence length $T = 36$, the model has achieved compression ratio $\frac{158 \times 36}{400} = 14.22$. Dimensionality reduction of the context vectors through principal component analysis (PCA) shows that context vectors can be efficiently embedded in lower dimensions (e.g. two-dimensional space).

At low-dimensional space, we can use supervised classification algorithms to learn the relationship between vectors representations and cluster assignment. The trained classification model can then be applied to the validation set to assign clusters for unseen data. In our experiment, a SVM classifier with radial basis function (RBF) kernel ($\gamma = 4$) was used. The results are shown in figure 5 below.



(a) 2 clusters



(b) 6 clusters

Figure 5: On the left, the context vectors were projected into two-dimensional space using PCA. The black solid line on the left joins all consecutive context vectors together as a trajectory. Different number of clusters were identified using simple K-means algorithm. Cluster assignment and the SVM decision boundaries are coloured in the charts. On the right, output dimensions are visualised on a shared time axis. The black solid line demarcates the training set (70%) and validation sets (30%). The line segments are colour-coded to match the corresponding clusters.

In the two-dimensional space, the context vectors separate into two clearly identifiable neighbourhoods. These two distinct neighbourhoods correspond to the shift in mean values across all output dimensions. When K-means clustering algorithm is applied, it captures these two neighbourhoods as two clusters in the first scenario.

When the number of clusters increases, they begin to capture more subtleties. In the six clusters scenario, successive context vectors oscillate back and forth between close clusters which correspond to interlacing troughs and crests in the output dimensions. Similar pattern can also be observed in the validation set, which indicates that the knowledge learned by the auto-encoder model is generalisable to unseen data.

4 DISCUSSION

Successive context vectors generated by windowing approach are always highly correlated, thus form a smooth trajectory in high-dimensional space. Additional dimensionality reduction techniques can be applied visualise the change of features as a smooth trajectory. One of the key contributions of this study is that, the context vectors form distinct neighbourhoods which can be identified through unsupervised clustering algorithms such as K-means. The clusters can be optionally labelled manually to identify process state (e.g. healthy vs. faulty). Alarm can be triggered when the context vector travels beyond the boundary of a predefined neighbourhood.

Another contribution of this study is that dimensionality of the output sequence can be relaxed, thus allowing the recurrent auto-encoder to perform partial reconstruction. Different subsets of sensors can be used to train different models. Since the context vectors generated are conditioned on each selected subset of sensors, the smooth trajectory and neighbourhood obtained from the algorithm can be used to reflect underlying state of different aspects of the system.

This proposed method performs multidimensional time series clustering, which can natively scale up to very high dimensionality as it is based on recurrent autoencoder model. It can be generalised to any multi-sensor multi-state processes for process state recognition. For example, it can be used to analyse vehicle telematics data in an on-line setting using pre-trained models. Alternatively, it can be used for human activity recognition using accelerometer and gyroscope data collected via smartphones.

Nevertheless, the proposed approach has not included any categorical sensor measurements (e.g. open/closed, tripped/healthy, start/stop... etc). Future research can focus on incorporating categorical measurements alongside real-valued measurements.

DISCLOSURE

The technical method described in this paper is the subject of British patent application GB1717651.2.

REFERENCES

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014. URL `http://arxiv.org/abs/1409.2329`.