entropy

MDPI

Communication

Latent Network Construction for Univariate Time Series Based on Variational Auto-Encode

Jiancheng Sun [1,*], Zhinan Wu [2,3], Si Chen [1], Huimin Niu [1] and Zongqing Tu [1]

[1] School of Software and Internet of Things Engineering, Jiangxi University of Finance and Economics, Nanchang 330013, China; 2201921459@stu.jxufe.edu.cn (S.C.); 2201921462@stu.jxufe.edu.cn (H.N.); 2202020658@stu.jxufe.edu.cn (Z.T.)
[2] School of Information Management, Jiangxi University of Finance and Economics, Nanchang 330013, China; 2201910057@stu.jxufe.edu.cn
[3] School of Mathematics and Computer Science, Yichun University, Yichun 336000, China
* Correspondence: sunjc@jxufe.edu.cn; Tel.: +86-0791-8384-5702

Abstract: Time series analysis has been an important branch of information processing, and the conversion of time series into complex networks provides a new means to understand and analyze time series. In this work, using Variational Auto-Encode (VAE), we explored the construction of latent networks for univariate time series. We first trained the VAE to obtain the space of latent probability distributions of the time series and then decomposed the multivariate Gaussian distribution into multiple univariate Gaussian distributions. By measuring the distance between univariate Gaussian distributions on a statistical manifold, the latent network construction was finally achieved. The experimental results show that the latent network can effectively retain the original information of the time series and provide a new data structure for the downstream tasks.

Keywords: time series; complex network; statistical manifold; latent space

Citation: Sun, J.; Wu, Z.; Chen, S.; Niu, H.; Tu, Z. Latent Network Construction for Univariate Time Series Based on Variational Auto-Encode. *Entropy* **2021**, *23*, 1071. https://doi.org/10.3390/e23081071

Academic Editor: Claudia Szabo

Received: 29 July 2021
Accepted: 16 August 2021
Published: 18 August 2021

1. Introduction

The analysis of observed time series has always been a fundamental problem, both in the natural and social sciences. Depending on the application area and the problem to be solved, there are various methods for time series analysis, such as time-frequency analysis [1], classification [2], recursive graphs [3], forecasting of road traffic accidents [4], and chaotic time series analysis [5]. In recent years, as the volume of data and the complexity of systems have increased, new methods have been proposed to cope with these situations. For instance, deep learning has been successfully applied to classification [6] and forecasting of time series due to its excellent ability to automatically extract features [7,8].

In the last decade, transforming time series into complex networks and implementing time series analysis in the context of complex systems has become a critical approach. The transformation of time series into complex networks represents time series characteristics, i.e., it provides a new form of data for time series analysis. Classical approaches focus mainly on the evolutionary properties of time series. As the complexity of the system increases, it is not enough to address the evolutionary properties alone. In fact, the interaction between different components of a system or time series usually leads to an increase in complexity [9]. Therefore, we need to understand such interactions, which is the primary motivation for transforming time series into complex networks. Moreover, the transformation makes it possible to analyze time series using methods from the field of complex systems, which also provides new ways to characterize time series.

Converting a time series into a complex network usually involves three steps: extracting the components of the time series to form the nodes of the network, defining the metric of the edges, and forming the network according to certain criteria. For the extraction of network nodes, there are different approaches for different types of time series. In the case

of univariate time series, the time series is usually intercepted into subsegments, which correspond to network nodes [10,11]. Lacasa et al. directly used each point in the time series as a node, forming the so-called visibility graph [12]. In transition networks, Kulp et al. proposed to map the time series to a latent Markov chain, where each state acts as a node [13]. In the case of multivariate time series, it is common practice to treat each variable as a node [14]. Based on information geometry and differential geometry theory [15,16], complex networks of time series can be constructed on Riemannian manifolds [17]. Once the nodes are determined, the next thing to be addressed is the relationship between the nodes, that is, how to determine the edges between nodes. Generally, the appropriate edge metric is chosen according to the characteristics of the nodes. For example, in the case of subsegments as nodes, the correlation coefficient or Euclidean distance can be considered to determine the edges [10]; in [13], the transition probability was used as the edge metric; and in the visibility graph, the strategy of the landscape was used to determine the edges [12]. In the final step of network construction, it is usually necessary to determine a threshold of edges, i.e., edges larger than the threshold are retained, while those smaller than the threshold are discarded.

Previous studies have focused on the definition of nodes and edges manually, which is a challenging task, especially the identification of nodes. For example, when segmenting a time series, numerous parameters such as sub-segment length, sliding window length, and sliding steps need to be determined in advance, making the optimization of the model a challenging task. In addition, time series are high dimensional and are commonly contaminated with noise. Consequently, there is an urgent need to use compressed and efficient latent features to represent time series. To address these problems, based on Variational Auto-Encode (VAE) [18], we propose in this paper to first map the univariate time series into a space of latent probability distributions and then perform the construction of latent networks. The main advantage of our method is that the nodes of the latent network can be extracted automatically, and the method is also theoretically insensitive to noise.

2. Materials and Methods

In this section, we first introduce the model architecture and then elaborate the training of the VAE and the construction method of the latent network, respectively.

2.1. Problem Statement and Framework of Proposed Model

Given a labeled dataset, $\mathcal{D} = \{(\boldsymbol{x}_k, c_k)\}_{k=1}^{K}$ consists of K univariate time series $\boldsymbol{x}_k$ together with their class labels c_k, where $\boldsymbol{x}_k$ is a sequence of real numeric values and its length is N_k, that is, $\boldsymbol{x}_k = \left(x_1, x_2 \ldots x_{N_k}\right)$. To validate the performance of the network reconstruction, univariate time series with class labels are used here. In fact, for the network reconstruction itself, the class labels are not necessary. In addition, it is not required that the time series be of equal length.

The framework of the proposed model is shown in Figure 1, which consists of two main parts: the VAE training process and the latent network construction process. It is common practice to use Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) to cope with time series [19]. As shown in the upper part of Figure 1, here, we chose LSTM to implement the encoding and decoding in VAE. The motivation for using VAE here was its ability to automatically extract features of the time series and generate a latent space. It is important to note that this latent space is defined by the probability distributions rather than a vector space. Therefore, VAE belongs to generative models, which are usually used to generate new data rather than using latent variables to implement downstream tasks, such as classification or regression. Some studies using latent variables to implement downstream tasks usually use either a sampled latent vector or a latent distribution as the representation of the input. When the latent distribution is used, it is also common to use only the mean vector and ignore the standard deviation vector.

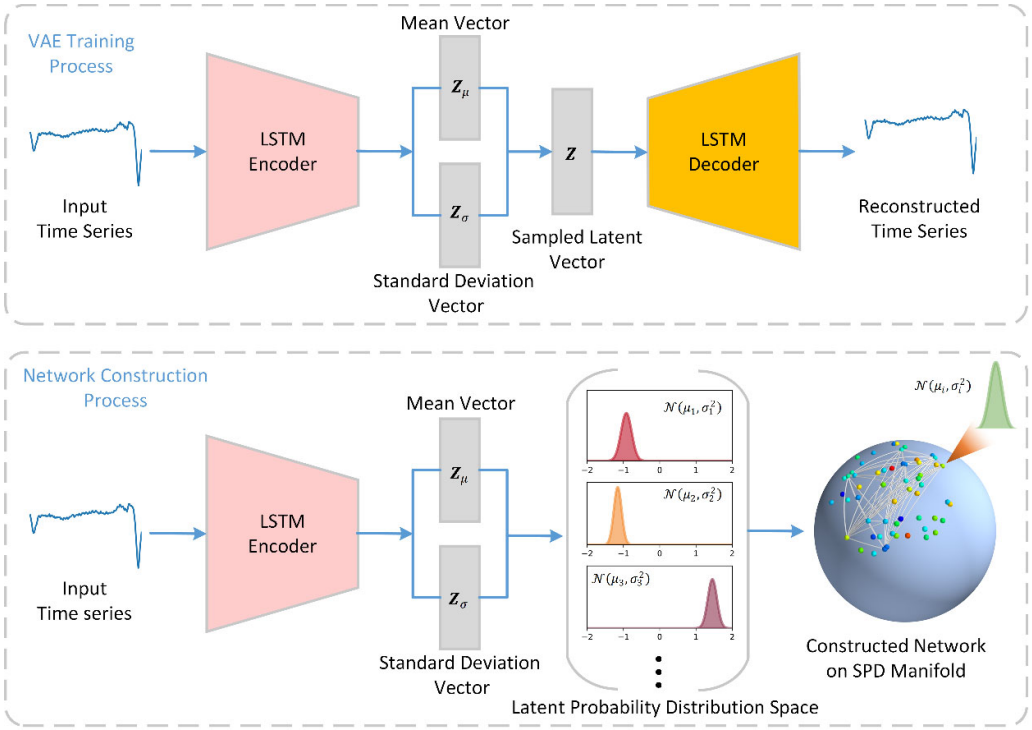


Figure 1. Framework of the model. The framework consists of the VAE training process (**upper part**) and the network construction process (**bottom part**).

Unlike previous strategies, we utilized all probabilistic information to create the latent network, i.e., including the mean and standard deviation. As shown in the bottom part of Figure 1, the input time series is first transformed into a mean vector $\boldsymbol{Z}_\mu \in \mathbb{R}^M$ and a standard deviation vector $\boldsymbol{Z}_\sigma \in \mathbb{R}^M$ using the trained encoder. The ith element in $\boldsymbol{Z}_\mu$ and $\boldsymbol{Z}_\sigma$ together form a univariate Gaussian probability distribution $\mathcal{N}(\mu_i, \sigma_i^2)$. According to the concept of information geometry [15], the probability distributions lie on a statistical manifold defined by symmetric positive definite (SPD) matrices. Finally, we derive the network by treating $\mathcal{N}(\mu_i, \sigma_i^2)$ $(i = 1, 2, \ldots, M)$ as nodes and the distance between them as edges.

2.2. VAE Triang Process

The structure of VAE is partially referenced in [19]. As seen in Figure 1, the VAE consists of four components: encoder, encoder-to-latent, latent-to-decoder, and decoder.

- The encoder feeds a series of input vectors $\boldsymbol{x}_k$ into the LSTM to obtain the output vector $\boldsymbol{h}_{out}^{enc}$, which is passed to the next layer;
- The role of encoder-to-latent is to map $\boldsymbol{h}_{out}^{enc}$ to the latent space, i.e., a mean vector $\boldsymbol{Z}_\mu$ and a standard deviation vector $\boldsymbol{Z}_\sigma$, using a linear layer;
- In the latent-to-decoder phase, the sampled latent vectors are passed through a linear layer to obtain the initial state $\boldsymbol{h}_0^{dec}$ of the decoder;
- In the final step, given the initial state $\boldsymbol{h}_0^{dec}$, the decoder input is initialized to zero and updated using the backpropagation method; then, the output of the decoder is passed to the output layer that is a linear layer for reconstructing time series $\hat{\boldsymbol{x}}_k$.

The loss function of VAE consists of two components, namely, the reconstruction error and the regularization term. The reconstruction error uses the MSE loss, while the regularization term (KL-divergence) makes the latent space more similar to the Gaussian distribution. The loss function can be expressed as:

$$\mathcal{L}(\theta; \boldsymbol{x}_k) = -D_{KL}(q(\boldsymbol{Z}|\boldsymbol{x}_k)||p(\boldsymbol{Z})) + \mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{x}_k)}[\log p_\theta(\boldsymbol{x}_k|\boldsymbol{Z})] \quad (1)$$

where D_{KL} is the KL-divergence and θ is the parameters of the model; p is the probability distribution of decoder and q is its approximation. Using the "reparametrization trick", the resampled vector is $\boldsymbol{Z} = \mu + \sigma\epsilon$ with $\epsilon \sim \mathcal{N}(0,1)$. Eventually, the loss function is reformulated as:

$$\mathcal{L}(\theta;\boldsymbol{x}_k) \approx \sum_{m=1}^{M}\left(1+\log\left((\sigma_k^m)^2\right)-(\mu_k^m)^2-(\sigma_k^m)^2\right)+\frac{1}{L}\sum_{l=1}^{L}\log p\left(\boldsymbol{x}_k\middle|\boldsymbol{Z}_k^l\right) \tag{2}$$

where M is the dimension of $\boldsymbol{Z}_\mu$ and L is the number of sampled vectors $\boldsymbol{Z}$. Here we focus on the model's architecture, while the details of LSTM and VAE can be found in [18,20].

2.3. Network Construction on Manifold

Once the VAE is trained, we can use it to encode a given time series to obtain the space of the latent probability distributions. In the latent space, we use the univariate Gaussian probability distribution $\mathcal{N}(\mu_i,\sigma_i^2)$ as a node and then measure the distances between different distributions as edges to form the network. Some classical methods can be used directly to measure the distance between Gaussian distributions, such as Kullback–Leibler Divergence or Bhattacharyya Distance [21]. Here, we used the geodesic distance to measure the similarity between probability distributions on a statistical manifold formed by the SPD matrices. Using geodesic distances, not only can network construction be achieved, but the manifold can also provide a convenient framework and properties for downstream tasks, such as classification, regression, and dimensionality reduction [22].

According to the theory of information geometry [23], the space of D-dimensional Gaussian distributions can be embedded into the space of $(D+1)\times(D+1)$-dimensional SPD matrices, which constitutes a manifold. For a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, this embedding is defined as:

$$\boldsymbol{P} = (\det\boldsymbol{\Sigma})^{-2/(D+1)}\begin{bmatrix}\boldsymbol{\Sigma}+\boldsymbol{\mu}\boldsymbol{\mu}^T & \boldsymbol{\mu}\\ \boldsymbol{\mu}^T & 1\end{bmatrix} \tag{3}$$

where $\boldsymbol{P}$ is the SPD matrix, which is localized on the manifold and corresponds to a multivariate Gaussian distribution. To measure the distance matrices $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$, we introduce the geodesic distance on the manifold as follows [22]:

$$d(\boldsymbol{P}_i,\boldsymbol{P}_j) = \mathrm{tr}\left(\log^2\left(\boldsymbol{P}_i^{-\frac{1}{2}}\boldsymbol{P}_j\boldsymbol{P}_i^{-\frac{1}{2}}\right)\right) \tag{4}$$

Considering a univariate Gaussian distribution $\mathcal{N}(\mu_i,\sigma_i^2)$, $\boldsymbol{P}$ is a 2×2-dimensional SPD matrix:

$$\boldsymbol{P}_i = \sigma_i^{-1}\begin{bmatrix}\sigma_i+\mu_i^2 & \mu_i\\ \mu_i & 1\end{bmatrix} \tag{5}$$

Assume that the time series $\boldsymbol{x}_k$ is compressed by the encoder to an M-dimensional Gaussian distribution space, i.e., $\boldsymbol{Z}_\mu \in \mathbb{R}^M$ and $\boldsymbol{Z}_\sigma \in \mathbb{R}^M$. Using (4), we can calculate the distances between pairs of univariate distributions that form the weighted adjacency matrix $\boldsymbol{A}_k = \left(a_{i,j}^k\right)_{M\times M}$, where $a_{i,j}^k = d(\boldsymbol{P}_i,\boldsymbol{P}_j)$. In some cases, to bring out the topology of the network, some edges can be removed by setting a threshold τ so that the adjacency matrix $\boldsymbol{A}_k$ is transformed into:

$$\overline{\boldsymbol{A}}_k = \begin{cases} a_{i,j}^k \; if \; a_{i,j}^k > \tau \\ 0 \quad otherwise \end{cases} \tag{6}$$

Thus far, we have obtained the latent network $\boldsymbol{A}_k$ of the time series $\boldsymbol{x}_k$. That is, we have transformed the time series into a network in the latent space, which is a representation of the time series. The network can be used to study the interactions within the latent variables or as new features for downstream tasks, such as clustering and classification.

To verify the reasonableness of the network $\boldsymbol{A}_k$, we additionally construct a network $\boldsymbol{B}$, where $\boldsymbol{A}_k$ is used as a node. That is, $\boldsymbol{B}$ is a network of networks. Using the class label c_k of $\boldsymbol{x}_k$, we can verify the reasonableness of the network $\boldsymbol{A}_k$ from the topology of network $\boldsymbol{B}$. To construct $\boldsymbol{B}$, the distance between metrics $\boldsymbol{A}_k$ is necessary. However, $\boldsymbol{A}_k$ is not an SPD matrix, so we cannot directly apply (4) to calculate the distance. Consequently, we introduce the graph Laplacian to transform $\boldsymbol{A}_k$ into an SPD matrix as:

$$\boldsymbol{L}_k = (\boldsymbol{D}_k - \boldsymbol{A}_k) + \lambda \boldsymbol{I} \tag{7}$$

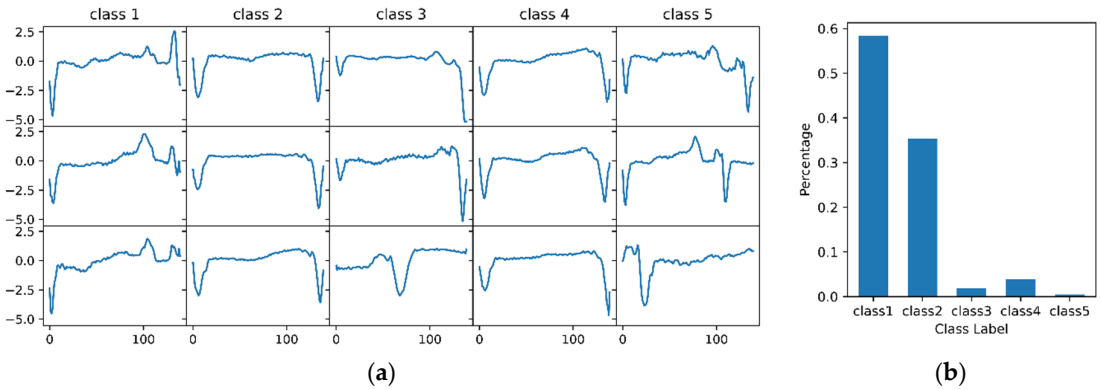
where $\boldsymbol{D}_k = \mathrm{diag}(\sum_j a_{i,j}^k)$ is the degree matrix of $\boldsymbol{A}_k$, $\lambda > 0$ is a regularization parameter, and $\boldsymbol{I}$ is the identity matrix. Now, we have two strategies to construct the network $\boldsymbol{B}$. One is to construct the network directly on the manifold. In this strategy, the distance between $\boldsymbol{L}_k$ is calculated using (4) to form the adjacency matrix, i.e., $\boldsymbol{B} = \left(b_{i,j}\right)_{K\times K}$, where $b_{i,j} = d\left(\boldsymbol{L}_i, \boldsymbol{L}_j\right)$ and K is the quantity of time series $\boldsymbol{x}_k$. The other strategy is to project $\boldsymbol{L}_k$ onto the tangent space of the manifold and then construct the network in the tangent space. Let $\boldsymbol{M} \in \mathcal{M}$ be the reference point defined as the mean of SPD matrices, and then the correspondence from manifold $\mathcal{M}$ to the tangent space $T_{\boldsymbol{M}}$ at $\boldsymbol{M}$ can be defined by a logarithmic mapping $\log_{\boldsymbol{M}} : \mathcal{M} \mapsto T_{\boldsymbol{M}}$ [22]:

$$\boldsymbol{y}_k = \log_{\boldsymbol{M}}(\boldsymbol{L}_k) = \boldsymbol{M}^{\frac{1}{2}} \log\left(\boldsymbol{M}^{-\frac{1}{2}} \boldsymbol{L}_k \boldsymbol{M}^{-\frac{1}{2}}\right) \boldsymbol{M}^{\frac{1}{2}} \tag{8}$$

Since $T_{\boldsymbol{M}}$ is a vector space, we can calculate the distance between $\boldsymbol{y}_k$ using an appropriate metric $D(,)$ in the Euclidean space to form the adjacency matrix, i.e., $\boldsymbol{B} = \left(b_{i,j}\right)_{K\times K'}$ where $b_{i,j} = D\left(\boldsymbol{y}_i, \boldsymbol{y}_j\right)$. In this study, we chose the second strategy to construct the network $\boldsymbol{B}$. The reason is that $T_{\boldsymbol{M}}$ is a Euclidean space, thus facilitating the application of some classical algorithms.

3. Experiment and Results

In this experiment, we used the electrocardiogram (ECG) dataset named ECG5000 from the UCR archive [24], where the training and test sets have a total of 5000 samples. We first merged the training and test datasets, and then 4500 randomly selected ECG data were used to train the VAE, and another 500 were used to build the latent network. This dataset has five classes with a sample length of 140, and the class labels were used to evaluate the performance of the network construction. Some of the samples are shown in Figure 2a, where each sequence corresponds to one heartbeat. It is important to note that the samples in each class are highly imbalanced, which is demonstrated in Figure 2b. It can be seen that the first and second classes of samples dominated, with few samples from the other three categories.



(**a**) (**b**)

Figure 2. ECG5000 dataset. (**a**) Three samples in each class were randomly selected for presentation. The horizontal and vertical axes indicate the class and sample, respectively. (**b**) Percentage of sample size in each class to total sample.

Due to the symmetry of VAE, the LSTM in both the encoder and decoder was one layer; each layer had 96 hidden units, and the dimension of the latent vectors $\boldsymbol{Z}_\mu$ and $\boldsymbol{Z}_\sigma$ was 20. The learning rate was set to 0.0005, and Adam was used to update the weights. Once the VAE was trained, we then used its encoder to obtain a representation of the test data, i.e., the space of the latent probability distributions. The result of the latent space corresponding to a sequence from class 1 is shown in Figure 3.

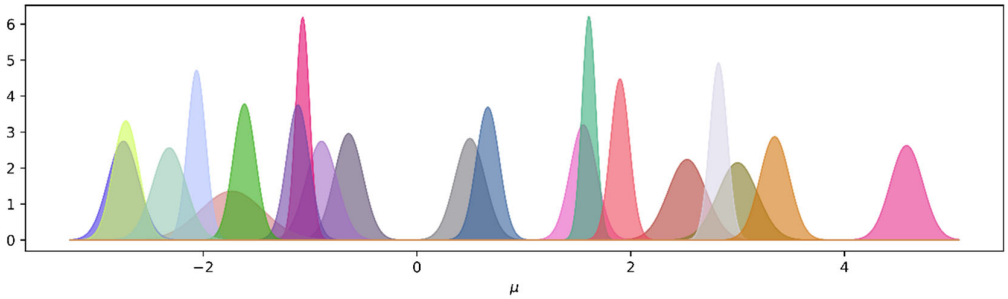


Figure 3. The latent probability distribution of a sequence. Different colors denote different univariate Gaussian distributions: $\mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i and σ_i^2 come from the ith element of the latent vectors $\boldsymbol{Z}_\mu$ and $\boldsymbol{Z}_\sigma$, respectively.

As can be seen in Figure 3, different $\mathcal{N}$ have different means and standard deviations. For the time series $\boldsymbol{x}_k$, we used (4) and (5) to calculate the distances between $\mathcal{N}$, thus forming the adjacency matrix $\boldsymbol{A}_k$. To highlight the topology of the network, we used (6) to remove most of the edges. In fact, how to determine an appropriate threshold τ in (6) has been an open problem in the construction of complex networks. Here, we retained 20% of the edges based on empirical values and visualization effects. Examples of latent networks corresponding to different time series are shown in Figure 4.

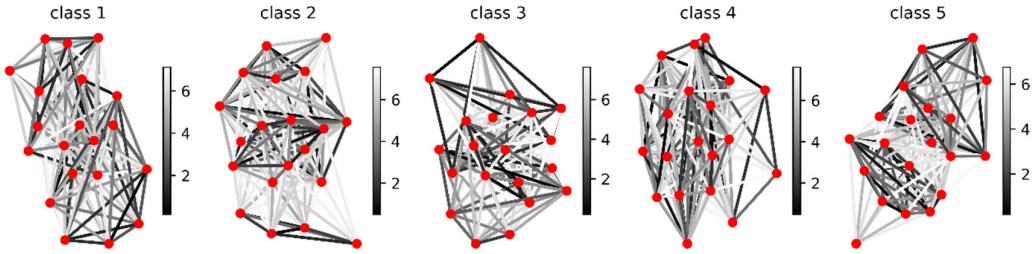


Figure 4. Latent networks of univariate ECG time series of different classes. Each node represents a univariate Gaussian distribution, and the edges indicate the distance between the nodes. The color bar denotes the weight of the edge.

Each network in Figure 4 corresponds to a time series that belongs to a different class. By transforming time series into networks, a new data structure is provided for time series analysis. Latent networks can capture the interaction of implicit variables in time series, which is the motivation for network construction. Consequently, latent networks can provide new data structures for time series mining and analysis. Once the latent network is in place, we can use numerous techniques from complex network science to characterize the time series. For example, the topological statistics of the network are extracted to form new features to analyze time series, such as classification of sleep stages [25] and analysis of seismic sequences [26]. In addition, graph neural networks (GNNs) have recently been rapidly developed and successfully used in many fields [27]. A prerequisite to apply this powerful tool is that the input data must be graph structured. In most cases, time series do not have an explicit graph structure. Therefore, the proposed latent network provides the conditions for using GNNs to analyze time series.

Latent networks are representations of time series; thus, different classes of networks usually have different topologies. In Figure 4, we cannot distinguish this difference with great certainty visually due to the complexity of the network connections. To verify the

plausibility of our method, using (7) and (8), we constructed a new network $\boldsymbol{B}$ based on the latent network $\boldsymbol{A}$, and then evaluated the construction of $\boldsymbol{A}$ in conjunction with the class labels. The results for network $\boldsymbol{B}$ are shown in Figure 5.

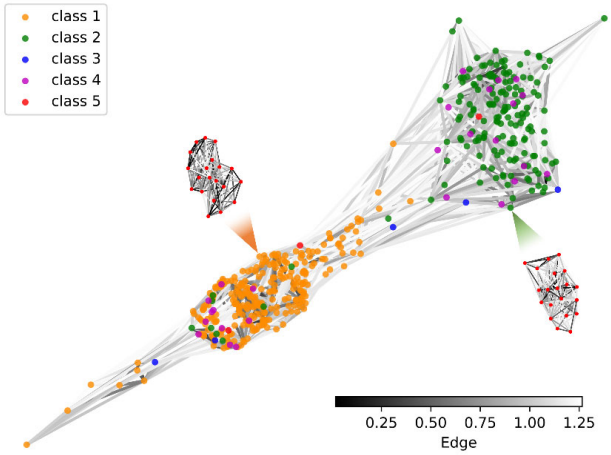


Figure 5. Network $\boldsymbol{B}$ of network $\boldsymbol{A}$. Class label identifies the color of the node, and the color bar denotes the weight (Euclidean distance) of the edges.

Figure 5 can be seen as a nested network, i.e., each node in the network also represents a network. As can be seen in the figure, the nodes of class 1 and class 2 clearly form two clusters. This topology indicates that network $\boldsymbol{A}$ retains the information of the original time series, allowing samples with similar attributes to cluster together, which also validates the effectiveness of network $\boldsymbol{A}$. However, it should also be noted that the other three classes of nodes are not effectively distinguished from each other. This phenomenon is mainly due to two factors. The first reason is the low distinguishability of these three classes of samples from the other two classes, especially the sequences of class 2 and class 4 (refer to Figure 2a). The second reason is that the percentage of samples in these three classes is too low (refer to Figure 2b), which influences the training of VAE.

4. Conclusions

Based on the trained VAE, our work implemented the construction of latent networks for univariate time series. We found that VAE can effectively extract the features of univariate time series and obtain intermediate representations. In addition, the constructed latent networks can effectively preserve the information of the original data and provide new data structures and analysis tools for time series analysis. The primary advantage of our work over traditional methods is that there is no need to manually identify the nodes of the network due to the ability of VAE to automatically extract the features of the time series and output the latent representation. In addition, the proposed approach is an unsupervised model, so the constructed latent network is not limited to a specific downstream task, but can provide new data structures for different tasks.

Author Contributions: Conceptualization, J.S.; software, Z.W. and S.C.; data curation, H.N.; writing—original draft preparation, J.S.; writing—review and editing, Z.W.; visualization, Z.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 62066017.

Institutional Review Board Statement: Ethical review and approval were waived for this study, due to the fact that the dataset used was publicly released and the dataset had been reviewed by the associated ethics committee.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data presented in this study are available on www.cs.ucr.edu/~eamonn/time_series_data/ (accessed on 17 August 2021).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Pham, T.D. Time–frequency time–space LSTM for robust classification of physiological signals. *Sci. Rep.* **2021**, *11*, 6936. [CrossRef] [PubMed]
2. Sun, J.; Yang, Y.; Liu, Y.; Chen, C.; Rao, W.; Bai, Y. Univariate time series classification using information geometry. *Pattern Recognit.* **2019**, *95*, 24–35. [CrossRef]
3. Marwan, N.; Romano, M.C.; Thiel, M.; Kurths, J. Recurrence plots for the analysis of complex systems. *Phys. Rep.* **2007**, *438*, 237–329. [CrossRef]
4. Popescu, T.D. Time series analysis for assessing and forecasting of road traffic accidents—case studies. *WSEAS Trans. Math.* **2020**, *19*, 177–185. [CrossRef]
5. Tsay, R.S.; Chen, R. *Nonlinear Time Series Analysis*; Wiley: Hoboken, NJ, USA, 2018; ISBN 978-1-119-26407-1.
6. Zhang, X.; Gao, Y.; Lin, J.; Lu, C.-T. TapNet: Multivariate time series classification with attentional prototypical network. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 6845–6852. [CrossRef]
7. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A. IoT type-of-traffic forecasting method based on gradient boosting neural networks. *Futur. Gener. Comput. Syst.* **2020**, *105*, 331–345. [CrossRef]
8. Lopez-Martin, M.; Sanchez-Esguevillas, A.; Hernandez-Callejo, L.; Arribas, J.I.; Carro, B. Additive ensemble neural network with constrained weighted quantile loss for probabilistic electric-load forecasting. *Sensors* **2021**, *21*, 2979. [CrossRef] [PubMed]
9. Mitchell, M. *Complexity: A Guided Tour*; Oxford University Press: Oxford, UK, 2011.
10. Zhang, J.; Small, M. Complex network from pseudoperiodic time series: Topology versus dynamics. *Phys. Rev. Lett.* **2006**, *96*, 238701. [CrossRef] [PubMed]
11. Donner, R.V.; Zou, Y.; Donges, J.F.; Marwan, N.; Kurths, J. Recurrence networks—A novel paradigm for nonlinear time series analysis. *New J. Phys.* **2010**, *12*, 033025. [CrossRef]
12. Lacasa, L.; Luque, B.; Ballesteros, F.; Luque, J.; Nuño, J.C. From time series to complex networks: The visibility graph. *Proc. Natl. Acad. Sci. USA* **2008**, *105*, 4972–4975. [CrossRef] [PubMed]
13. Kulp, C.W.; Chobot, J.M.; Freitas, H.R.; Sprechini, G.D. Using ordinal partition transition networks to analyze ECG data. *Chaos An Interdiscip. J. Nonlinear Sci.* **2016**, *26*, 073114. [CrossRef] [PubMed]
14. Jiang, M.; Gao, X.; An, H.; Li, H.; Sun, B. Reconstructing complex network for characterizing the time-varying causality evolution behavior of multivariate time series. *Sci. Rep.* **2017**, *7*, 10486. [CrossRef] [PubMed]
15. Amari, S.-I.; Nagaoka, H. *Methods of Information Geometry*; American Mathematical Society: Providence, RI, USA, 2000; Volume 191, ISBN 0821843028.
16. Manale, J. Integrating the Gaussian through differentiable topological manifolds. *WSEAS Trans. Math.* **2019**, *18*, 55–61.
17. Sun, J.; Yang, Y.; Xiong, N.N.; Dai, L.; Peng, X.; Luo, J. Complex network construction of multivariate time series using information geometry. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 107–122. [CrossRef]
18. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. In Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014—Conference Track Proceedings, Banff, AB, Canada, 14–16 April 2014.
19. Fabius, O.; van Amersfoort, J.R. Variational recurrent auto-encoders. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Workshop Track Proceedings, San Diego, CA, USA, 7–9 May 2015.
20. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
21. Nagino, G.; Shozakai, M. Distance measure between Gaussian distributions for discriminating speaking styles. In Proceedings of the INTERSPEECH 2006—ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, 17–21 September 2006; Volume 2, pp. 657–660.
22. Tuzel, O.; Porikli, F.; Meer, P. Pedestrian detection via classification on Riemannian manifolds. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 1713–1727. [CrossRef] [PubMed]
23. Lovrić, M.; Min-Oo, M.; Ruh, E.A. Multivariate normal distributions parametrized as a Riemannian symmetric space. *J. Multivar. Anal.* **2000**, *74*, 36–48. [CrossRef]
24. Chen, Y.; Keogh, E.; Hu, B.; Begum, N.; Bagnall, A.; Abdullah Mueen, G.B. The UCR Time Series Classification Archive. Available online: www.cs.ucr.edu/~{}eamonn/time_series_data/ (accessed on 17 August 2021).
25. Zhu, G.; Li, Y.; Wen, P. Analysis and classification of sleep stages based on difference visibility graphs from a single-channel EEG signal. *IEEE J. Biomed. Health Inform.* **2014**, *18*, 1813–1821. [CrossRef] [PubMed]
26. Telesca, L.; Lovallo, M. Analysis of seismic sequences by using the method of visibility graph. *Europhys. Lett.* **2012**, *97*, 50002. [CrossRef]
27. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]