Recurrent Autoencoder with Sequence-Aware Encoding

Robert Susik[0000−0003−0653−433X]

Institute of Applied Computer Science,
Łódź University of Technology, Poland rsusik@kis.p.lodz.pl

Abstract. Recurrent Neural Networks (RNN) received a vast amount of attention last decade. Recently, the architectures of Recurrent AutoEncoders (RAE) found many applications in practice. RAE can extract the semantically valuable information, called context that represents a latent space useful for further processing. Nevertheless, recurrent autoencoders are hard to train, and the training process takes much time. This paper proposes a new recurrent autoencoder architecture with sequence-aware encoding (RAES), and its second variant which employs a 1D Convolutional layer (RAESC) to improve its performance and flexibility. We discuss the advantages and disadvantages of the solution and prove that the recurrent autoencoder with sequence-aware encoding outperforms a standard RAE in terms of model training time in most cases. The extensive experiments performed on a dataset of generated sequences of signals shows the advantages of RAES(C). The results show that the proposed solution dominates over the standard RAE, and the training process is the order of magnitude faster.

Keywords: Recurrent · AutoEncoder · RNN · Sequence.

1 Introduction

Recurrent Neural Networks (RNN) [22, 29] received a vast amount of attention last decade and found a wide range of applications such as language modelling [18, 24], signal processing [8, 23] and anomaly detection [19, 25].

The RNN is (in short) a neural network adapted to sequential data that have the ability to map sequences to sequences achieving excellent performance on time series. The RNN can process the data of variable length or fixed length (in this case, the computational graph can be unfolded and considered a feed-forward neural network). Multiple layers of RNN can be stacked to process efficiently long input sequences [12, 21]. The training process of deep recurrent neural network (DRNN) is difficult because the gradients (in backpropagation through time [29]) either vanish or explode [3, 9]. It means that despite the RNN can learn long dependencies the training process may take a very long time or even fail. The problem was resolved by the application of Long Short-Term Memory (LSTM) [13] or much newer and simpler Gated Recurrent Units (GRU) [6]. Nevertheless, it is not easy to parallelize calculations in recurrent neural networks which impacts the training time.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

A different and efficient approach was proposed by Aäron et al. [20] who proved that stacked 1D convolutional layers can process efficiently long sequences handling tens of thousands of time steps. The CNNs have also been widely applied to autoencoder architecture to solve problems such as outlier and anomaly detection [1, 14, 16], noise reduction [5], and more.

Autoencoders [4] are unsupervised algorithms capable of learning latent representations (called *context* or *code*) of the input data. The context is usually smaller than input data to extract only the semantically valuable information. Encoder-Decoder (Sequence-to-Sequence) [7, 26] architecture looks very much like autoencoder and consists of two blocks: encoder, and decoder, both containing a couple of RNN layers. The encoder takes the input data and generates the code (a semantic summary) used to represent the input. Later, the decoder processes the code and generates the final output. The encoder-decoder approach allows having variable-length input and output sequences in contrast to classic RNN solutions. Several related attempts, including an interesting approach introduced by Graves [12] have been later successfully applied in practice in [2,17]. The authors proposed a novel differentiable attention mechanism that allows the decoder to focus on appropriate words at each time step. This technique improved the state of the art in neural machine translation (NMT) and was later applied even without any recurrent or convolutional layers [28]. Besides the machine translation, there are multiple variants and applications of the Recurrent AutoEncoders (RAE). In [10], the authors proposed the variational recurrent autoencoder (VRAE), which is a generative model that learns the latent vector representation of the input data used later to generate new data. Another variational autoencoder was introduced in [11, 27] where authors apply convolutional layers and WaveNet for audio sequence. Interesting approach, the Feedback Recurrent AutoEncoder (FRAE) was presented in [30]. In short, the idea is to add a connection that provides feedback from decoder to encoder. This design allows efficiently compressing the sequences of speech spectrograms.

This paper presents an autoencoder architecture that applies a different context layout and employs a 1D convolutional layer to improve its flexibility and reduce the training time. We also propose a different interpretation of the context (the final hidden state of the encoder). We transform the context into the sequence that is passed to the decoder. This technical trick, even without changing other elements of architecture, improves the performance of recurrent autoencoder.

We demonstrate the power of the proposed architecture for time series reconstruction (the generated sequences of signal). We perform a wide range of experiments on a dataset of generated signals, and the results are promising.

Following contributions of this work can be enumerated: (i) We propose a recurrent autoencoder with sequence-aware encoding that trains much faster than standard RAE. (ii) We suggest an extension to proposed solution which employs the 1D convolutional layer to make the solution more flexible. (iii) We show that this architecture performs very well on univariate and multivariate time series reconstruction.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

2 The model

In this section, we describe our approach and its variants. We also discuss the advantages and disadvantages of the proposed architecture and suggest possible solutions to its limitation.

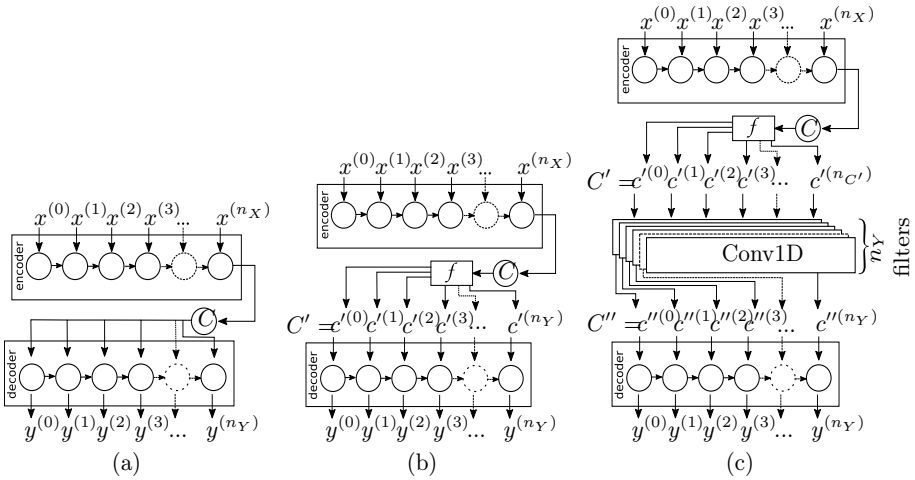


Fig. 1: Recurrent autoencoder architectures: (a) Recurrent AutoEncoder (RAE) [7, 26]; (b) Recurrent AutoEncoder with Sequence-aware encoding (RAES); (c) Recurrent AutoEncoder with Sequence-aware encoding and 1D Convolutional layer (RAESC).

2.1 Recurrent AutoEncoder (RAE)

The recurrent autoencoder generates an output sequence $Y = (y^{(0)}, y^{(1)}, \ldots, y^{(n_Y-1)})$ for a given an input sequence $X = (x^{(0)}, x^{(1)}, \ldots, x^{(n_X-1)})$, where n_Y and n_X are the sizes of output and input sequences respectively (both can be of the same or different size). Usually, $X = Y$ to force autoencoder learning the semantic meaning of data. First, the input sequence is encoded by the RNN encoder, and then the given fixed-size context variable C of size m_C (a single vector containing m_C features) is decoded by the decoder (usually also RNN), see Figure 1a. If $m_C < n_X$, then the autoencoder is called undercomplete. On the other hand if $m_C > n_X$, then the autoencoder is called overcomplete. The first variant is much more popular as it allows the autoencoder to learn the semantically valuable information.

2.2 Recurrent AutoEncoder with Sequential context (RAES)

We propose a recurrent autoencoder architecture (Figure 1b) where the the final hidden state of the encoder $C = (c_0, c_1, \ldots, c_{m_C-1})$ is interpreted as the sequence

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

of time steps $C' = (c'^{(0)}, c'^{(1)}, \ldots, c'^{(n_C-1)})$, thus $c'^{(i)} = (c_0'^{(i)}, c_1'^{(i)}, \ldots, c_{m_{C'}-1}'^{(i)})$, where n_C is the number of time steps (equal to n_Y) and $m_{C'}$ is the number of features (called λ later). It is an operation performed on the context and may be defined as $f : C \mapsto C'$. The code $C = (c_i)_{i=0}^{n_C-1}$ is transformed to

$$C' = ((c_{i\lambda+j})_{j=0}^{\lambda-1})_{i=0}^{n_C/\lambda-1} \tag{1}$$

where $\lambda = n_C/n_X$ ($\lambda \in \mathbb{N}$). Once the context is transformed ($C' = (c'^{(0)}, c'^{(1)}, \ldots, c'^{(n_X-1)})$), the decoder starts to decode the sequence C' of $m_{C'} = \lambda$ features producing the output sequence Y. This technical trick in the data structure speeds up the training process (Section 3).

Additionally, this way, we put some sequential meaning to the context. It means that this architecture model should attempt to learn the time dependencies in the data and store them in the context. Therefore, we can expect the improvement in the training performance, in particular for long sequences (hundreds of elements).

Finally, the one easily solvable disadvantage of this solution is that the size of context must be multiple of input sequence length $n_C = \lambda n_X$, where n_C is the size of context C, which limits the possible applications of such architecture.

2.3 RAES with 1D Convolutional layer (RAESC)

In order to solve the limitation mentioned in Section 2.2, we propose adding a 1D convolutional layer (and max-pooling layer) to the architecture right before the decoder (Figure 1c). This approach gives the ability to control the number of output channels (also denoted as *feature detectors* or *filters*), defined as follows:

$$C''(i) = \sum_k \sum_l C'(i+k, l) w(k, l) \tag{2}$$

In this case, n_C does not have to be multiple of n_X, thus to have the desired output sequence of n_Y length, the number of filters should be equal to n_Y. Moreover, the output of the 1D convolution layer $C'' = \text{conv1D}(C')$ should be transposed. Hence each channel becomes an element of the sequence as shown in Figure 1c. Finally, the desired number of features on output Y can be configured with hidden state size of the decoder.

A different and simpler approach to solve the mentioned limitation is stretching the context C to the size of decoder input sequence and filling the gaps in with averages.

The described variant is very simplified and is only an outline of proposed recurrent autoencoder architecture (the middle part of it, to be more precise) which can be extended by adding pooling and recurrent layers or using different convolution parameters (such as stride, or dilation values). Furthermore, in our view, this approach could be easily applied to other RAE architectures (such as [11, 30]).

The recurrent neural network gradually forgets some information at each time step, and may completely ignore the time dependencies between the beginning

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

and end of the sequence. It is even worse in the recurrent autoencoder where the context size is usually limited. We believe that the proposed layout of the context enforces the model to store the time dependencies in the context, thus it works well on long sequences. Additionally, this characteristic may be very interesting in signal classification tasks where such memory may be crucial to identify an object.

3 Experiments

In order to evaluate the proposed approach, we run a few experiments, using a generated dataset of signals. We tested the following algorithms:

- Standard Recurrent AutoEncoder (RAE) [7, 26].
- RAE with Sequence-aware encoding (RAES).
- RAES with Convolutional and max-pooling layer (RAESC).

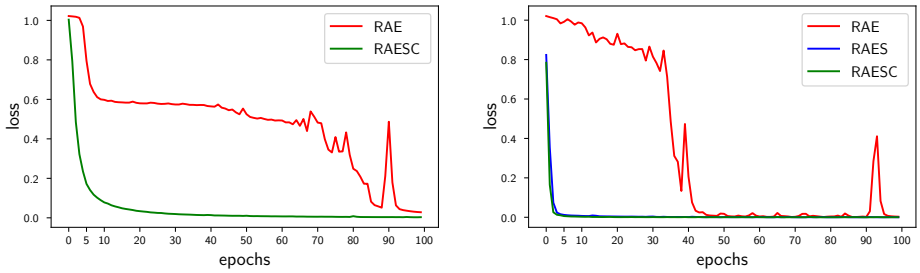
The structure of decoder and encoder is the same in all algorithms. Both decoder and encoder are single GRU [6] layer, with additional time distributed fully connected layer in the output of the decoder. The algorithms were implemented in Python 3.7.4 with TensorFlow 2.3.0. The experiments were run on a GPU server with an AMD Epyc 7702P CPU (64 cores, 128 threads) clocked at 2.0 GHz with 4 MiB L1, 32 MiB L2 and 256 MiB L3 cache and 6x Quadro RTX 6000 graphic cards with 24220MiB VRAM each (only one was used). The test machine was equipped with 504 GB of RAM and running Ubuntu 18.04.4 64-bit OS. We trained the models with Adam optimizer [15] in batches of size 100 and Mean Squared Error (MSE) loss function. All the presented algorithms were implemented and the source codes (including all the datasets mentioned above) can be found at the following URL: https://github.com/rsusik/raesc. The dataset contains generated time series (sum of sine waves) that consists of 5000 sequences of length 200 with {1, 2, 4, 8} features and is published in the same repository along with source codes. The dataset was shuffled and split to training and validation sets in proportions of 80:20, respectively.

In the first set of analyses, we investigated the impact of context size and the number of features on performance. We noticed a considerable difference in training speed (number of epochs needed to achieve plateau) between the classic approach and ours. To prove whether our approach has an advantage over the RAE, we performed tests with different size of the context n_C and a different number of input features m_X. We set context size (n_C) proportionally to the size of the input and we denote it as:

$$\sigma = \frac{n_C}{m_X n_X} \tag{3}$$

Figure 2 proves that the training process of the RAE needs much more epochs than RAESC to achieve a similar value of loss function. In chart a) the size of context is set to $\sigma = 25\%$ and in b) it is set to $\sigma = 100\%$ of the input size. For $\sigma = 25\%$ the RASEC achieves plateau after 20 epochs while the RAE does not

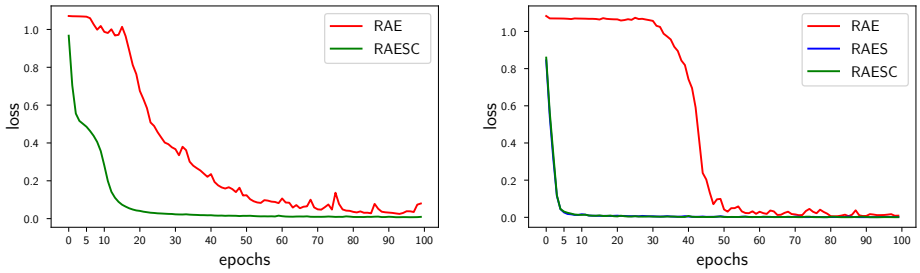
ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4



(a) $\sigma = 25\%$ (b) $\sigma = 100\%$

Fig. 2: Loss as function of epoch number for univariate data and $\sigma = \{25\%, 100\%\}$.

at all (it starts decreasing after nearly 80 epochs, but behaves unstable). There is no RAES result presented in this plot because of the limitation mentioned in Section 2.2 (size of the code was too small to fit the output sequence length). For $\sigma = 100\%$ both RASEC and RAES achieve the plateau in less than five epochs (order of magnitude faster) while the RAE after about 35 epochs.



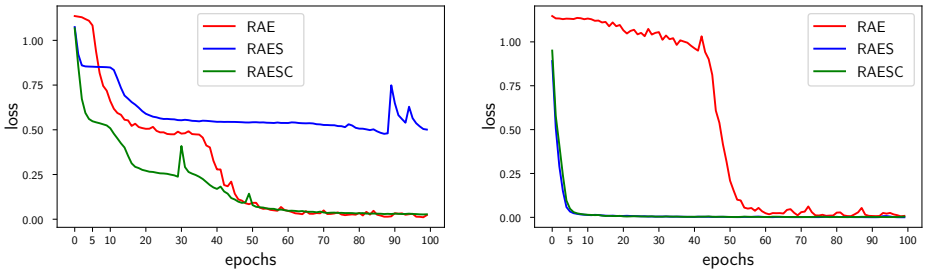
(a) $\sigma = 25\%$ (b) $\sigma = 100\%$

Fig. 3: Loss as function of epoch number for two features ($m_X = 2$) and $\sigma = \{25\%, 100\%\}$.

Figure 3 shows the loss in function of the number of epochs for two features in input data. This experiment confirms that both RAES and RAESC dominates in terms of training speed, but a slight difference can be noticed in comparison to univariate data (Figure 2). It shows that the RAE achieves plateau in about 50 epochs for both cases while RAES and RAESC after 20 epochs for $\sigma = 25\%$ and in about five epochs for $\sigma = 100\%$.

Figure 4 presents a loss in function of the number of epochs for four features. Comparing Figure 4a to previous ones (Figure 3a and Figure 2a) we can clearly

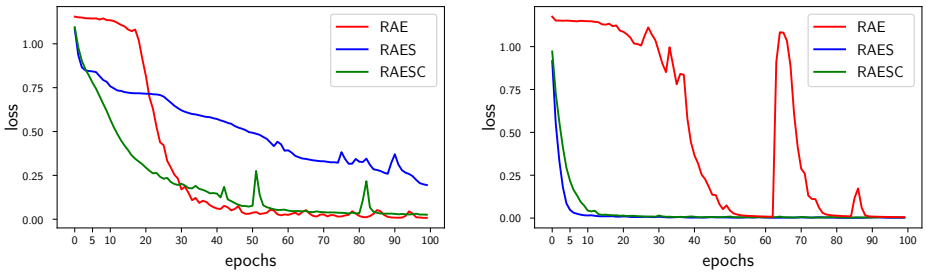
ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4



(a) $\sigma = 25\%$ (b) $\sigma = 100\%$

Fig. 4: Loss as function of epoch number for four features ($m_X = 4$) and $\sigma = \{25\%, 100\%\}$.

see a downward trend (for growing number of features) in all architectures' performance, but the slope for proposed ones look steeper than for RAE. On the other hand, it can not be observed for $\sigma = 100\%$ (Figure 4b) because in this case, the difference is marginal.



(a) $\sigma = 25\%$ (b) $\sigma = 100\%$

Fig. 5: Loss as function of epoch number for $m_X = 8$ and $\sigma = \{25\%, 100\%\}$.

Figure 5 shows a loss in function of the number of epochs for eight features. This figure is interesting in several ways comparing to the previous ones (Figures 2, 3, 4). The chart a) shows that, for a much larger number of features and relatively small size of the context, the training of RAES variant takes much more epochs. The similar observation may be noticed for RAESC, where the loss drops much faster than the RAE at the beginning of the training but achieves the plateau at almost the same step. On the other hand, chart b) shows that for a larger size of context, the proposed solution dominates.

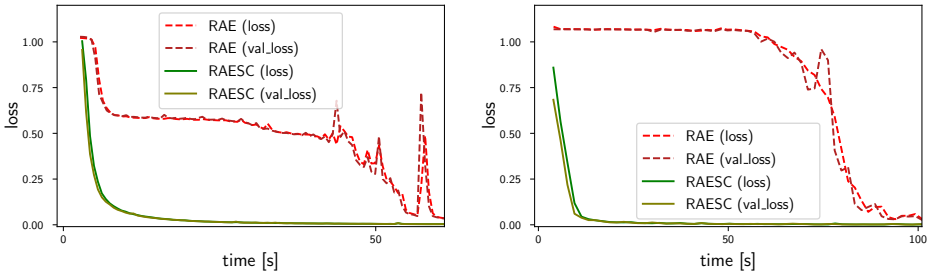
We measured each algorithm's training time to confirm that the proposed solution converges faster than RAE for the same size of context. Table 1 shows

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

features (m_X)	algorithm	σ		
		25%	50%	100%
1	RAE	0.61	0.61	0.93
	RAES	-	-	0.89
	RAESC	0.63	0.63	0.98
2	RAE	0.64	0.92	1.77
	RAES	-	0.88	1.57
	RAESC	0.65	0.96	1.85
4	RAE	0.91	1.74	4.65
	RAES	0.89	1.57	3.81
	RAESC	0.97	1.85	4.75
8	RAE	1.75	4.63	13.22
	RAES	1.56	3.80	10.07
	RAESC	1.85	4.74	13.47

Table 1: Epoch time [s] (median) for different number of features (m_X) and context size (σ).

the median of epoch time for a different number of features and context size. The table confirms that the RAES is faster than RAE by about 5% for univariate data and about 31% faster for $m_X = 8$. The training process (the epoch) of RAESC algorithm takes a slightly more time than RAE, which is marginal (less than 2%) for $m_X = 8$.



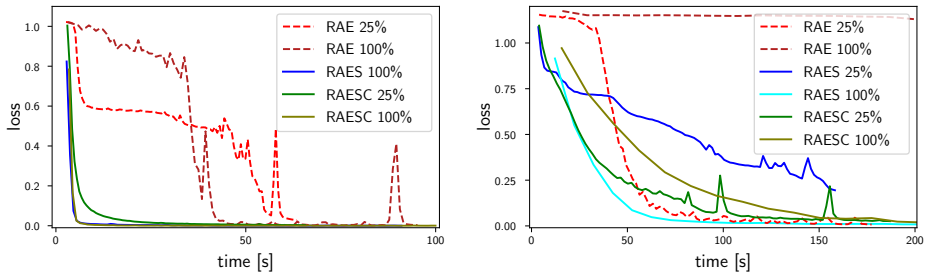
(a) $m_X = 1$, $\sigma = 25\%$ (b) $m_X = 2$, $\sigma = 100\%$

Fig. 6: Loss and validation loss as function of time [s] for $m_X = \{1, 2\}$ and $\sigma = \{25\%, 100\%\}$ respectively.

To confirm that the presented architectures do not tend to overfit, we compared a loss and validation loss functions. Figure 6 illustrates the loss and validation loss of RAE and RAESC (to make the chart more readable, RAES is excluded) in the function of time (in seconds). We can clearly see on both charts (Figure 6a and Figure 6b) that the validation loss goes approximately along with

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

loss function (except for a few bounces). We observed similar behaviour for all the experiments performed.



(a) $m_X = 1$ (b) $m_X = 8$

Fig. 7: Loss as function of time [s] for $m_X = \{1, 8\}$ and $\sigma = \{25\%, 100\%\}$.

To make it transparent and clear, we compared all the algorithms, including the training time for different σ. Figure 7 presents a loss in a function of time (in seconds) for $m_X = \{1, 8\}$ features. As expected, we can clearly see that proposed architecture dominates for univariate data disqualifying RAE (Figure 7a). Interestingly, the RAE with larger context size ($\sigma = 100\%$) converges faster than the one with smaller context size ($\sigma = 25\%$). The fastest are RAES and RAESC ($\sigma = 100\%$ both), achieving almost the same results (the loss functions of both overlap on the chart). For the multivariate data (in Figure 7b) on the contrary to univariate we can see that the RAE with smaller context size converges much faster than with a larger one. The most striking fact to emerge from these results is that the RAE 100% does not drop in the whole period. The training of RAE 25% is slower at the begining than proposed architecture but speeds up after 40 seconds achieving very similar result after 100 seconds.

In most of the charts presented it can be noticed that the training process of RAE fluctuates significantly on the contrary to the proposed solution where it is relatively stable. It is also worth mentioning that all the experiments were performed with a fixed filter size for both convolutional and max-pooling layers, and it is likely that we could achieve better results by tuning these hyperparameters.

4 Conclusions and future work

In this work, we proposed an autoencoder with sequence-aware encoding. We proved that this solution outperforms the RAE in terms of training speed in most cases.

The experiments confirmed that the training of proposed architecture takes less time than the standard RAE. It is a critical factor if the training time is limited, for example, in Automated Machine Learning (AutoML) tools or in

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

a hyperparameter optimization. The context size and the number of features in the input sequence have a high impact on training performance. Only for a relatively large number of features and small size of the context the RAE achieves comparable results to the proposed solution. In other cases our solution dominates and the training time is an order of magnitude shorter.

In our view, these results constitute a good initial step toward further research. The implementation of proposed architecture was simplified, and the use of different layers and hyperparameter tunning seems to offer great opportunities to tune it achieving even better results.

The latent space produced by proposed architecture is still underexplored yet but seems to be an interesting point to be addressed in future research. These findings suggest an application of this architecture to different recurrent models such as variational recurrent autoencoder, which could significantly improve the training performance of generative models.

We believe that the proposed solution has a wide range of practical applications and is worth confirming.

References

1. An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE **2**(1), 1–18 (2015)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5**(2), 157–166 (1994)
4. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics **59**(4-5), 291–294 (1988)
5. Chiang, H.T., Hsieh, Y.Y., Fu, S.W., Hung, K.H., Tsao, Y., Chien, S.Y.: Noise reduction in ecg signals using fully convolutional denoising autoencoders. IEEE Access **7**, 60806–60813 (2019)
6. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. CoRR **abs/1409.1259** (2014), http://arxiv.org/abs/1409.1259
7. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
8. Ding, J., Wang, Y.: Wifi csi-based human activity recognition using deep recurrent neural network. IEEE Access **7**, 174257–174269 (2019)
9. Doya, K.: Bifurcations of recurrent neural networks in gradient descent learning. IEEE Transactions on neural networks **1**(75), 218 (1993)
10. Fabius, O., van Amersfoort, J.R.: Variational recurrent auto-encoders. arXiv preprint arXiv:1412.6581 (2014)
11. Gârbacea, C., van den Oord, A., Li, Y., Lim, F.S., Luebs, A., Vinyals, O., Walters, T.C.: Low bit-rate speech coding with vq-vae and a wavenet decoder. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 735–739. IEEE (2019)
12. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4

13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
14. Kieu, T., Yang, B., Guo, C., Jensen, C.S.: Outlier detection for time series with recurrent autoencoder ensembles. In: IJCAI. pp. 2725–2732 (2019)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Liao, W., Guo, Y., Chen, X., Li, P.: A unified unsupervised gaussian mixture variational autoencoder for high dimensional outlier detection. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 1208–1217. IEEE (2018)
17. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
18. Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., Khudanpur, S.: Extensions of recurrent neural network language model. In: 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP). pp. 5528–5531. IEEE (2011)
19. Nanduri, A., Sherry, L.: Anomaly detection in aircraft data using recurrent neural networks (rnn). In: 2016 Integrated Communications Navigation and Surveillance (ICNS). pp. 5C2–1. Ieee (2016)
20. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. In: Arxiv (2016), https://arxiv.org/abs/1609.03499
21. Pascanu, R., Gulcehre, C., Cho, K., Bengio, Y.: How to construct deep recurrent neural networks. In: Proceedings of the Second International Conference on Learning Representations (ICLR 2014) (2014)
22. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)
23. Shahtalebi, S., Atashzar, S.F., Patel, R.V., Mohammadi, A.: Training of deep bidirectional rnns for hand motion filtering via multimodal data fusion. In: GlobalSIP. pp. 1–5 (2019)
24. Shi, Y., Hwang, M.Y., Lei, X., Sheng, H.: Knowledge distillation for recurrent neural network language modeling with trust regularization. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 7230–7234. IEEE (2019)
25. Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2828–2837 (2019)
26. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014)
27. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. In: Advances in Neural Information Processing Systems. pp. 6306–6315 (2017)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
29. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE **78**(10), 1550–1560 (1990)
30. Yang, Y., Sautière, G., Ryu, J.J., Cohen, T.S.: Feedback recurrent autoencoder. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3347–3351. IEEE (2020)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_4