Recurrent Neural Network and Convolutional Network for Diabetes Blood Glucose Prediction

Minkai Sheng

Abstract—**The convolutional neural network (CNN) is one of the most widely used neural networks. Although the main battlefield is image recognition, natural language processing, and speech recognition, CNNs in other fields also have a good performance. In image recognition, they outperformed humans in some tests. CNN's have much fewer connections and parameters and so they are easier to train, while their theoretically best performance is likely to be only slightly worse. A Recurrent Neural Network (RNN) is a type of Neural Network used to process sequence data. Compared with ordinary neural networks, it can handle the data of sequence changes. Long short-term memory (LSTM) is a special RNN, mainly to solve the problem of gradient disappearance and gradient explosion in the training process of Long sequence. Simply put, LSTM performs better in longer sequences than a normal RNN. CNN and RNN models are presented to forecast the future glucose levels of patients with type 1 diabetes. Finally, we obtain the predictions of the testing dataset and evaluate the results by the root mean squared error (RMSE). The mean value of the best RMSE of 7.5545.**

Index Terms—**Convolutional neural network, recurrent neural network, diabetes.**

I. Introduction

It is estimated that 415 million people in the world suffer from diabetes. The predicted number increases are estimated to be approximately 642 million by 2040. Due to diabetes, one person (five million a year) dies every six seconds—more than HIV, tuberculosis, and malaria combined [1]. Diabetes is a chronic metabolic disease, many patients do not have obvious symptoms, but the complications can be quite severe if left undiagnosed. Most of the complications of diabetes are caused by poor control of blood sugar due to a damaged pancreas, so the most important thing for diabetic patients is to control blood sugar in a target range (70-180 mg/dL). Predicting blood sugar levels in advance can be used to regulate blood sugar through insulin.

In this paper, we introduce Long short-term Memory (LSTM), which is a Recurrent neural network and Gate Recurrent Unit(GRU) to glucose prediction. Due to its unique design structure, LSTM is suitable for processing and predicting important events with very Long intervals and delays in time series. For general-purpose sequence modeling, LSTM as a special RNN structure has proven stable and powerful for modeling long-range dependencies in various previous studies [2]. We also introduce the Convolutional Neural Networks which are a deep learning model or multi-layer perceptron similar to artificial Neural Networks, which are commonly used to analyze visual images. Specifically, we used three layers of CNN to add one layer of RNN, each using pooling, and dropout. The essence of pooling is sampling. We used two pooling layers of size 3 and one pooling layer of size 1. Dropout can effectively mitigate the occurrence of overfitting and it is greatly reduced by randomly omitting half of the feature detectors on each training case.

Both models achieved good results, and their loss was within 0.03.

Manuscript received November 11, 2021; revised May 16, 2022.

Minkai Sheng is with the Coventry University, UK (e-mail: markshengcn@gmail.com).

II. Dataset and Preprocessing

UVA/PADOVA Type 1 Diabetes Simulator was submitted to FDA in 2013, which is research for nonlinearities of insulin action in the hypoglycemic range and into glucagon kinetics. The dataset we used consisted of 103680 rows and 6 columns, including glucose level, Meal, Rheumatoid arthritis, Proinsulin level, and time. Through Sklearn.Preprocessing. MinMaxScaler each features to transform scaling to between 0 and 1.

III. Methods

This section mainly introduces network models and training parameters and activation functions. For the first model, the input parameters of Gate Recurrent Unit were glucose level, Meal, Rheumatoid arthritis, Proinsulin level, and time. GRU was chosen because its experimental effect is similar to LSTM, but easier to calculate [3]. The input/output structure of GRU is the same as that of ordinary RNN. In Fig. 1, there is a current input x^t and the hidden state h^{t-1} passed from the previous node, which contains information about the previous node. Combining x^t and h^{t-1}, the GRU gets the output of the current hidden node h^{t-1} and the hidden state passed to the next node h^t.

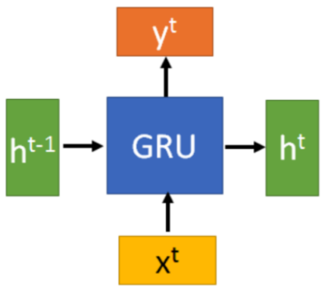

Fig. 1. Input and output structure of GRU.

Unit state is removed from the GRU structure and hidden state is used to transmit information. It has only two gate structures, the update gate and the reset gate.

In Fig. 2, the update gate functions similarly to an LSTM's forget and input gates. It determines what information should be discarded and what should be included. Another gate used to select how much old knowledge to forget is the reset gate [4].

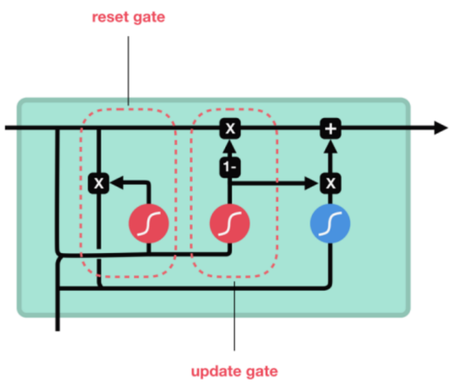

Fig. 2. Cell and gate of GRU.

That's a GRU, by the way. Because GRUs have fewer tensor operations, they are a bit faster to train than LSTMs. There is no obvious winner when it comes to which is best. Researchers and engineers typically test both to see which one is best for their application [5].

LSTM and GRU are proposed solutions to the short-term memory problem by introducing internal mechanisms called gates that regulate the flow of information. These gate structures can learn which data in a sequence is important information to keep and which to delete. By doing so, it can carry out predictions by passing relevant information along a long chain of sequences. Almost all advanced RESULTS based on RNN are implemented through these two networks. LSTM and GRU are often used in areas such as speech recognition, speech synthesis and text generation. They can also be used to generate captions for video.

For the second model, the Convolutional neural network and cyclic neural network are mainly used, and the pooling layer is used. Convolutional neural networks have been used to apply convolutional neural networks to various fields, such as object segmentation and style transformation. What CNN can really do is play the role of a feature extractor, so CNN is also suitable for this task [6]. The Pooling layer is still a process of feature selection and information filtering, in essence, that is to say, part of the information is intentionally lost to the network, which is a compromise of computing performance [7]. After extracting features using CNN, LSTM was used for prediction. Because we have access to both past and future input characteristics for a particular time in a regression job, we may use a bidirectional LSTM network [8]. The input of LSTM is the input of the three-layer CNN. The purpose of using CNN is to extract features, and the purpose of using the LSTM is to achieve regression [9]-[11].

The control flow of LSTM is similar to that of RNN. Both of them process the data transmitted by information in the forward propagation process. The difference lies in the change of the structure and operation of LSTM units, which can enable LSTM to selectively retain or forget some information. The core concepts of LSTM are its cell states and various gate structures. The unit state is equivalent to the pathway that can transmit related information, which allows information to be passed down in the sequence chain. This part can be regarded as the "memory" of the network. In theory, unit states can carry relevant information all the time during sequence processing. Therefore, information acquired in earlier time steps can also be transmitted to cells in later time steps, thus reducing the effect of short-term memory. In the process of network training, information can be added or removed by gate structure, and different neural networks can decide which relevant information to remember or forget by gate structure on unit state [12].

The gate structure contains the Sigmoid function, an activation function similar to the Tanh function. But its output interval is not (-1, 1), but (0, 1), which helps update or forget the data, because any number multiplied by 0 is 0, and that part of the information is forgotten. Again, any number multiplied by 1 is the same value, and this information is completely retained. By doing so, the network knows which data is unimportant to forget and which numbers are important to keep [13].

The Sigmoid function is defined by the following formula:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Sigmoid function graph such as S curve in the below Fig. 3:

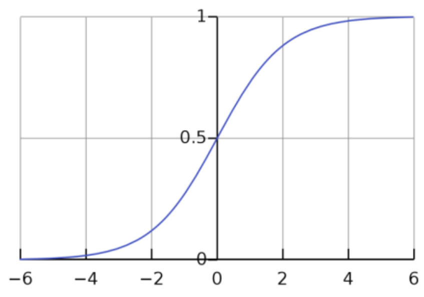

Fig. 3. Graph of Sigmoid function.

There are three types of gate structures in an LSTM unit that regulate the flow of information: forgetting gate, input gate and output gate.

In Fig. 4, the forget gate determines which information should be discarded or retained. Both information from the previously hidden state and information from the current input are entered into the Sigmoid function with output values between 0 and 1, the closer to 0 means the more you should forget, and the closer to 1 means the more you should keep[14].

In Fig. 5, the input gate is used to update the cell status. Input the previously hidden state information and the currently entered information into the Sigmoid function, and adjust the output value between 0 and 1 to determine which information to update, with 0 indicating unimportant and 1 indicating important[15]. You can also pass the hidden state and current input to the Tanh function, compress the value between -1 and 1 to adjust the network, and then multiply the Tanh output by the Sigmoid output, which determines which information in the Tanh output is important and needs to be preserved.

In Fig. 6, first multiply the previous cell state and the forgotten vector point by point. If it is multiplied by a value close to zero, it means that these values may be discarded in the new cell state. It is then added point by point to the output value of the input gate, and the new information discovered by the neural network is updated to the cell state, thus obtaining the new cell state [15].

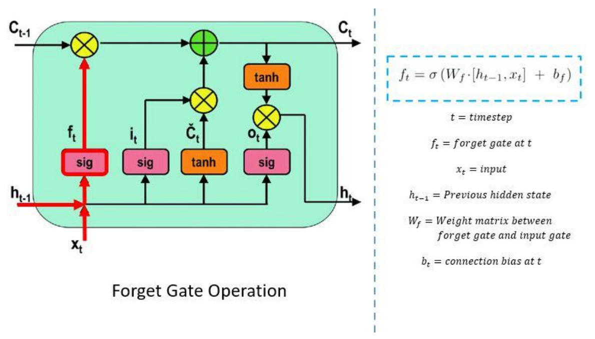
C_{t-1}
C_t
tanh
f_t
i_t
$\check{C}_t$
O_t
sig
sig
tanh
sig
h_{t-1}
h_t
x_t
Forget Gate Operation
$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$
t = timestep
f_t = forget gate at t
x_t = input
h_{t-1} = Previous hidden state
W_f = Weight matrix between forget gate and input gate
b_t = connection bias at t

Fig. 4. Graph of the forget gate of the LSTM.

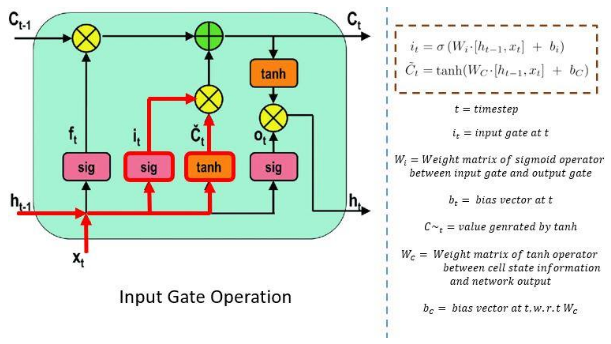
C_{t-1}
C_t
tanh
f_t
i_t
$\check{C}_t$
O_t
sig
sig
tanh
sig
h_{t-1}
h_t
x_t
Input Gate Operation
$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$
$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$
t = timestep
i_t = input gate at t
W_i = Weight matrix of sigmoid operator between input gate and output gate
b_t = bias vector at t
$C\sim_t$ = value genrated by tanh
W_c = Weight matrix of tanh operator between cell state information and network output
b_c = bias vector at t.w.r.t W_c

Fig. 5. Graph of the input gate of the LSTM.

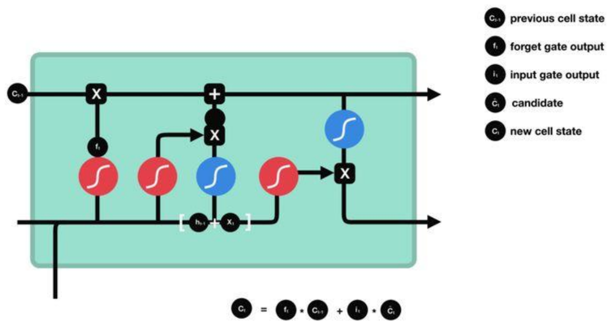

Fig. 6. Graph of the cell state of the LSTM.

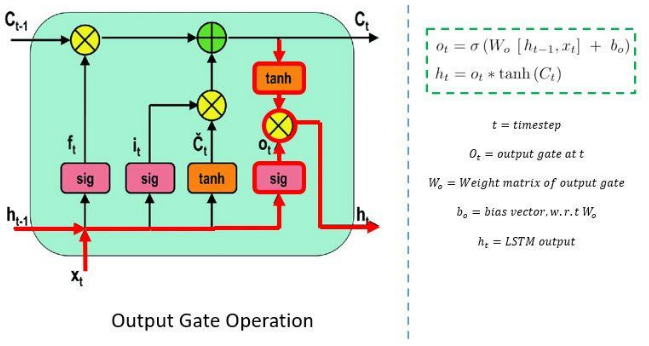
C_{t-1}
C_t
tanh
f_t
i_t
$\check{C}_t$
O_t
sig
sig
tanh
sig
h_{t-1}
h_t
x_t
Output Gate Operation
$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$
$h_t = o_t * \tanh\left(C_t\right)$
t = timestep
O_t = output gate at t
W_o = Weight matrix of output gate
b_o = bias vector.w.r.t W_o
h_t = LSTM output

Fig. 7. Graph of the output gate of the LSTM.

In Fig. 7, the output gate determines the value of the next hidden state, which contains information related to the previous input. Of course, hidden states can also be used for prediction. First, the previously hidden state and current input are passed to the Sigmoid function; And then I pass the new unit state to the Tanh function; The Tanh output is then multiplied by the Sigmoid output to determine what information the hidden state should carry; Finally, the hidden state is output as the current unit, and the new unit state and new hidden state are transmitted to the next time step [16].

RNN's are good for processing sequence data for predictions but suffers from short-term memory. LSTM's and GRU's were created as a method to mitigate short-term memory using mechanisms called gates. Gates are just neural networks that regulate the flow of information flowing through the sequence chain. LSTM's and GRU's are used in state of the art deep learning applications like speech recognition, speech synthesis, natural language understanding, etc.

Overfitting is a major issue. Large networks are extremely sluggish to utilize, making it difficult to avoid overfitting by integrating predictions from several large neural networks at test time. In Fig. 8, the dropout is a method of dealing with this issue. The essential concept is to remove units from the neural network at random during training [17]. Thus Dropout is used after each fully connected layer. Since this task is a regression problem, linear activation functions are used. In most situations, the affine transformation provides a linear translation of an input function to an output, as executed in the hidden layers before the final prediction of the class score for each label. The input vectors x transformation is given by $f\,(x) = w^T + b - (1.1)$, where x = input, w = weights, b = biases[18].

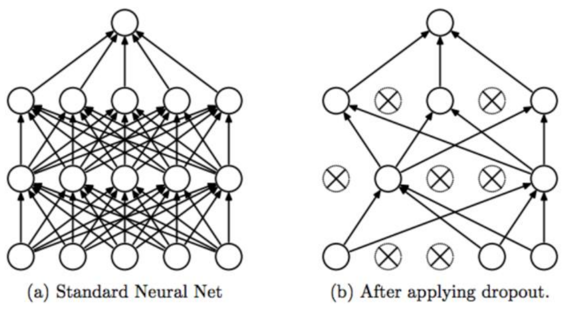

Fig. 8. Graph of the dropout function.

In this paper, all of the experiments are mainly use keras to achieve the neural network functions. Keras is a Python-based open source artificial neural network framework that serves as a high-level application interface for Tensorflow, Microsoft-CNTK, and Theano deep learning models construction, debugging, evaluation, application, and visualization. Keras is designed in an object-oriented, modular, and extendable style, with operating methods and documentation that consider user experience and difficulties while attempting to simplify the implementation of complex algorithms. Keras may participate in the development of statistical learning models through encapsulation and supports major techniques in contemporary ARTIFICIAL intelligence, such as feedforward and recursive neural networks. Keras supports multi-GPU parallel computation under multi-operating systems and may be transformed into components under Tensorflow, Microsoft-CNTK, and other systems according to backdrop Settings in terms of hardware and development environment [19].

Keras encapsulates the RNN model and is very convenient to call. Keras implements support for the RNN related layer model in the Recurrent module of the Layers package, as well as a bidirectional RNN wrapper on the Wrapper model. RNN models in Recurrent module include RNN, LSTM,GRU and other models [20]-[22].

IV. Result

The model1 part uses one layer of GRU and three layers of the fully connected neural network, and the activation function uses Linear. After 80 epoch sessions, the loss dropped to a minimum of 0.0224, as shown in Fig. 9. After 80 epochs, the loss did not continue to decrease, indicating that the network had been fully trained.

The model2 part uses 3 CNN layers and one LSTM layer, the activation function uses Linear. After 80 epoch sessions, the loss dropped to a minimum of 0.0299, as shown in Fig. 10.

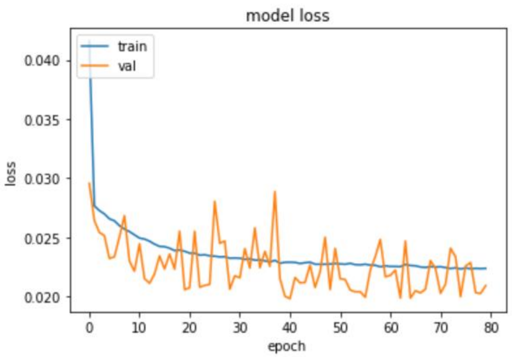

Fig. 9. The Loss of the Model1.

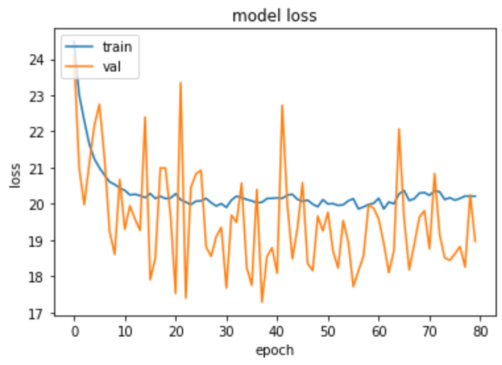

Fig. 10. The Loss of the Model2.

Using one of the essential evaluation metrics to evaluate the prediction performance: root mean squared error (RMSE) and the mean absolute relative difference (MARD), which can be expressed as:

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}$$

$$MARD = \frac{1}{N}\sum_{N}^{k=1}\frac{|yk - \hat{y}k|}{yk}$$

We mainly focus on RMSE for each patient and record the results for each patient after several runs. The GRU model's RMSE is 7.2692, and the MARD is 7.3090. The RCNN model's RMSE is 14.1687, and the MARD is 11.0979(See Table 1).

TABLE I: The Evaluation of Models

Model name	RMSE	MARD
GRU	7.5545	7.9259
RCNN	14.1687	11.0979

According to the results, it can be found that the GRU model obviously achieves better results compared with RCNN. GRU controls the transmission state through the gated state, remembers the information that needs to be remembered for a long time, and forgets the unimportant information. Unlike ordinary CRNN, which can only have one memory overlay mode.

V. Conclusion

In this paper, a regression problem is used to predict the blood glucose level of diabetic patients, and the GRU model has achieved a relatively ideal effect. Meanwhile, the effect of the RCNN model under this problem is compared, and it is found that although the effect of RCNN is not as ideal as that of GRU, it has extremely high performance, which greatly reduces the training network time. However, due to the possibility that too many features may be filtered out, the good effect cannot be reflected.

Author Contributions

Minkai Sheng is the independent author of this paper. He completed the program experiment, data analysis, and the writing and typesetting of the paper.

Acknowledgment

First of all, I would like to give my heartfelt thanks to all the people who have ever helped me with this paper. My sincere and hearty thanks and appreciations go firstly to my supervisor, Kezhi Li who is Assistant Professor from University College London, Taiyu Zhu who is a Ph.D student from Imperial College London, whose suggestions and encouragement have given me much insight into these studies.

I am also extremely grateful to all my friends and classmates Yuxi Zhang who is a student from the University of Liverpool, Yiyang Wang who is a student from Zhejiang Wanli University, and Yueyi Chen who is a student from China University of Petroleum(East China). They have kindly provided me with assistance and companionship in the course of preparing this paper.

References

[1] A. Krizhevsky, I Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, vol. 25, pp. 1097-1105, 2012.

[2] S. H. I. Xingjian, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems*, pp. 802-810, 2015.

[3] Natural Language Computing Group. (October 17, 2018). R-NET: Machine Reading Comprehension with Self-matching Networks. Microsoft Research. [Online]. Available: https://www.microsoft.com/en-us/research/publication/mcr/

[4] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[5] *Using LSTM and GRU Neural Network Methods for Traffic Flow Prediction*, IEEE Xplore, 2016.

[6] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," *ArXiv.Org*, 2014.

[7] H. Wu, "Max-pooling dropout for regularization of convolutional neural networks," *ArXiv.Org*, 2015.

[8] A. Graves, "Speech Recognition with deep recurrent neural networks," *ArXiv.Org*, 2013.

[9] M. Lin, "Network in network," *ArXiv.Org*, 2013.

[10] J. Chen, K. Li, P. Herrero, T. Zhu, and P. Georgiou, "Dilated recurrent neural network for short-time prediction of glucose concentration," *KHD@ IJCAI*, pp. 69-73, 2018.
[11] T. Zhu, K. Li, P. Herrero, J. Chen, and G. P. Eorgiou, "A deep learning algorithm for personalized blood glucose prediction," *KHD@ IJCAI*, pp. 64-78, 2018.
[12] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.
[13] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," arXiv preprint arXiv:1811.03378, 2018.
[14] R. C. Staudemeyer, "Understanding LSTM -- a tutorial into long short-term memory," *ArXiv.Org*, 2019.
[15] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: lstm cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.
[16] T. Zia and U. Zahid, "Long short-term memory recurrent neural network architectures for Urdu acoustic modeling," *International Journal of Speech Technology*, vol. 22, no. 1, pp. 21–30, 2018.
[17] N. Srivastava. (2014). Dropout: A simple way to prevent neural networks from Overfitting. Jmlr.Org. [Online]. Available: https://jmlr.org/papers/v15/srivastava14a.html
[18] F. Agostinelli, "Learning activation functions to improve deep neural networks," *ArXiv.Org*, 2014.
[19] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, 2019.
[20] Team, K. (n.d.). (2021). Keras documentation: Recurrent layers. Keras.Io. [Online]. Available: https://keras.io/api/layers/recurrent_layers/
[21] Team, K. (n.d.-a). (2021). Keras documentation: LSTM layer. Keras.Io. [Online]. Available: https://keras.io/api/layers/recurrent_layers/lstm/
[22] Team, K. (n.d.-a). (2021). Keras documentation: GRU layer. Keras.Io. [Online]. Available: https://keras.io/api/layers/recurrent_layers/gru/

Minkai Sheng was born in Shanghai, China, in 1999. He is a student studying Computing Science at Coventry University since 2020. He obtained an Infocomm technology Diploma at Singapore's Productivity and Standards Board Academy. And he is also a member of the Singapore Computer Society. His research interest includes medical intelligence and computer vision.