Forecasting Method based upon GRU-based Deep Learning Model

Ali Jaber Almalki
University of Central Florida, USA
Department of Computer Science
Ali.almalki@knights.ucf.edu

Pawel Wocjan
University of Central Florida, USA
Department of Computer Science
pawel.wocjan@ucf.edu

***Abstract*—In this research, the world model has a modified RNN model carried out by a bi-directional gated recurrent unit (BGRU) as opposed to a traditional long short-term memory (LSTM) model. BGRU tends to use less memory while executing and training faster than an LSTM, as it uses fewer training parameters. However, the LSTM model provides greater accuracy with datasets using longer sequences. Based upon practical implementation, the BGRU model produced better performance results. In BGRU, the memory is combined with the network. There is no update gate and forget in the GRU. The forget and update gate are treated as one unit thus it is the primary reason of parameter reduction.**

Key words: BGRU, Deep learning, Reinforcement learning, RNN, and VAE.

I. Introduction

While bi-directional gated recurrent unit (BGRU) and long short-term memory (LSTM) deep learning techniques both help to predict data, the latter requires more memory, decreasing world model efficiency. A BGRU technique trains systems faster while using less memory. In this research, world model efficiency is increased by modifying core model components. The main goal of constructing this model is to inform the reinforcement learning algorithm (artificial agent) that it can use environmental information to inform its next action [1]. Learning models behave according to the technique applied to produce results.

The aim is to build an artificial agent that gets proficient at driving round a 2-D racetrack. The machine-learning algorithm is fed an spatial observation (64 x 64 pixel color image of a car and its surroundings) at each step, which it then uses to respond and return the next set of actions, such as acceleration, steering direction, and braking. The actions are then passed to the environment, which returns the next observation and caused the cycle to begin anew.

This work explores use of generative neural network models in popular reinforcement learning environments known as "world models". Recurrent neural networks (RNNs) are used to predict independent outputs and future input information [2]. These predictions are useful for defining correlation relations based on time and environment sequences.

Additionally, this paper covers a modified world model that uses three models (RNN, autoencoder, and controller). In this novel approach, the world model has a modified RNN model that employs a BGRU instead of LSTM. Experimental results are based upon agent training in the modified model [3]. A closer, detailed analysis is discussed along with generated plots and tests generated after model training.

II. Related Work

Substantial research has been done to predict agent behavior using heuristic-based and deep learning models. Various techniques and methods have been used to determine result accuracy. A proposed system that labelled extracted problems as T-Recs became complicated when the model produced large numbers of outputs [4]. Its algorithm also failed to identify environmental elements, leaving output dependent on a large number of parameters.

A data-driven approach was proposed that used a probability optimization technique and a statistical algorithm with a mixed column layout to solve problems. Given large amounts of data training, the results became difficult to control.

A distributed deep neural network was proposed within a multilingual acoustic model used for identifying and extracting features from hidden layers. It offers an environment where multiple tasks can be learned and uses the deep neural network process for optimization. For model training, a feedforward neural network (a limited network that performs calculations from the lower layer) was used to determine computational results, but it restricted the model's training by forbidding many input values including some from the lower layer itself. While every layer of the system was calculated to generate output, the model lost all data [5].

A method proposed neural network training via back propagation, using the chain rule method to calculate and produce output values. Regarding each parameter, it calculated a derivation of loss function based upon the chain rule method. However, the method produced a trained agent unable to reach the global minimum point. For practical network training, it used a stochastic gradient descent to accelerate the agent's learning of different environmental values [6].

III. Methodology

The world model consists of three trainable models and will be implemented on a car racing game. The model will be trained to generate an agent that can drive around a track.

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00096

The primary components of the world model are a variational autoencoder (VAE) model (V), a mixture density network (MDN)-RNN model (M), and a controller model (C).

A. The Model

The car racing game uses a normal reinforcement-learning environment in which an artificial agent tries to learn a policy, a trial and error method, to get the maximum possible reward.

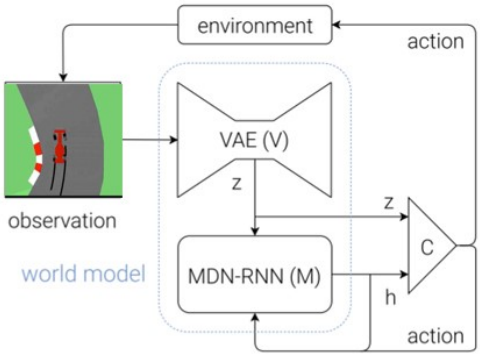


Fig. 1. World model.

B. Steps

A 2-D observation image is obtained to begin, a 64 x 64 x 3-D image (width, length, RGN depth). The image is then put into a convolutional VAE that encodes into a latent vector *z* of size 32. A memory module (M), an MDN-RNN in this case, subsequently takes *z* from the VAE via a previous action (chosen by C and a previous hidden state *h* of M). Its goal is to capture a latent observation of the environment by predicting the next *z*, forwarding the *h* of the RNN of length 256. Finally, a single-layer linear model (C) provides direction by mapping *z* and M's *h* to an action step at each time step. The output results in an action-vector of size 3, containing a quantitative representation of steering direction (-1 to 1), acceleration (0 to 1), and braking (0 to 1).

C. VAE Model

Drivers do not analyze every single pixel in view; instead, the human brain analyzes useful aspects, such as upcoming curves, road straightness, and land topography, to support making the right steps. This is what the VAE is trained to accomplish by condensing a 64 x 64 x 3 input image into a 32-dimensional latent vector that follows Gaussian distribution. This helps the artificial agent work with a smaller representation of its environment, allowing for more efficient learning.

The V in our case is trained to compress the domain registration of input images in a car race environment. The model consists of an encoder and a decoder that convert high dimensional input images into lower dimensional vectors, then reconstruct the image at the decoder output. This simple architecture is trained to minimize reconstruction errors between initial, encoded, and decoded data.

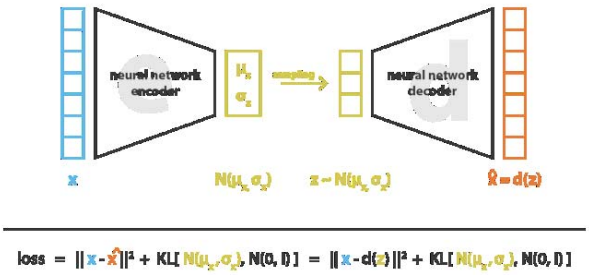


Fig. 2. Variational autoencoder model.

In VAEs, the loss function is composed of a reconstruction term (making encoding and decoding more efficient) and a regularization term (making the latent space regular).

D. MDN-RNN Model

An RNN is trained to predict the latent encoding of future frames using previous latent recordings and actions. Similar to the VAE, the RNN's goal is to capture the latent understanding of the car in its environment. By contrast, the M helps to predict future agent movements based on previous agent environmental performance [7]. Gaussian distribution is used to sample a prediction dependent on V. M provides prediction results based on both past and current information received from V. An MDN is combined with an RNN to gather better predictions [8].

A latent vector can then make effective predictions and handle a complex environment more effectively. Moreover, during sampling, parameter temperature can be adjusted. The combinational model of MDN-RNN allows the sequence generation problem to include both pen detection and handwriting and can handle both effectively.

The advantages of using an RNN include the ability to remember all stored data over time, which is described as long-short memory. They are also used in convolutional layers to extend the effective pixel neighborhood [9]. Common challenges include difficulty training an RNN, gradient vanishing, and exploding problems. An RNN also cannot process very long sequences when using tanh or relu as an activation function.

The role of V is to convert high dimensional input to low dimensional space represented by vector *z*. With V, each frame is converted into *z*. M learns to predict the next time frame vector *zt* + 1 based on the previous frame vector *zt*, basing it on an MDN-RNN [10]. The Gaussian mixture model is used in the MDN network to predict $P(zt + 1|at, zt, ht)$, where *at* is an action taken at time *t*, *zt* is the previous frame vector, and *ht* is the hidden state of the RNN at time *t*.

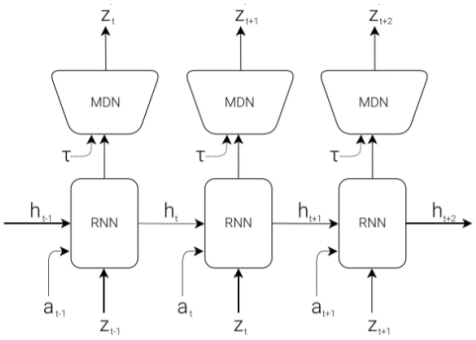


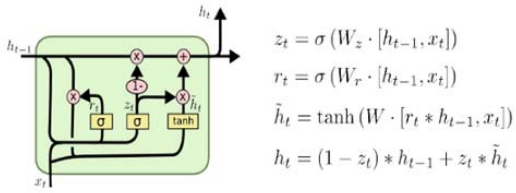
Fig. 3. MDN-RNN for future prediction.

Our modification to the trainable models is in the modification of the RNN model, replacing the LSTM with a bidirectional BGRU. The key difference between a BGRU and an LSTM is that the former has two gates (reset and update) while the latter has three gates (input, output, and forget). The BGRU's architecture combines forget and input gates into the update gate [11] and merges the cell and hidden states.

E. Bidirectional Gated Recurrent Units

The BGRU was discovered in 2014 and efficiently reduces parameters with performance that exceeds the LSTM by combining memory with the network. There is no update gate and forget in the BGRU. The forget and update gate are treated as one unit thus it is the primary reason of parameter reduction [12].

When making predictions about current states, a BGRU takes results from later and previous time steps, efficiently extracting information from backward and forward features. Using the different testing models, it offers better information semantics. Semantic analysis has verified its experimental effectiveness and good environmental performance due to the reduction in parameters [13].



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Fig. 4. GRU architecture.

F. GRU Architecture

This architecture does not maintain the internal cell system based upon three main gates. Instead, the information is stored in an LSTM recurrent unit that is incorporated with the hidden layer of the BGRU [15]. The three primary gates of a BGRU are the *update* gate *z* (which contains past knowledge required for future decisions. In an LSTM recurrent unit, the output gate operates the data in analog), the reset gate *r* (which estimates how much past knowledge is required to forget. In an LSTM recurrent unit, this is the combined functions of the forget and input gates), and the current memory state *ht* (which incorporates with the reset gate to introduce non-linearity in the input, making the input zero-mean). That last is also treated as the sub-part of the reset gate, helping to reduce the effects of past information.

The LSTM model has more control on the network since it has three gates, but we prefer using GRU due to some key advantages:

1. GRUs train faster and perform better than LSTM with less training data, if language modeling is used;
2. GRUs are simpler and easier to modify (e.g., adding new gates in case of additional inputs to the network) with less code; and
3. GRUs control information flow like an LSTM unit without using a memory unit, exposing the full hidden content without control.

G. Controller Model

The controller is a densely connected neural network that is used to determine the actions. Controller is single layer linear model which maps the output of MDN-RNN to the actions. It takes zt and ht as an input and generated the action at at each time step. at is defined as follows:

$$a_t = W_c[z_t h_t] + b_c \quad (1)$$

where Wc is the weight matrix and bc is the bias.

RESULTS

I. TRAINING VAE

VAE is dependent upon the convolutional neural network model where KL loss and a combination of the reconstruction used in model training. The rollout was generated, and the graph shows the loss.

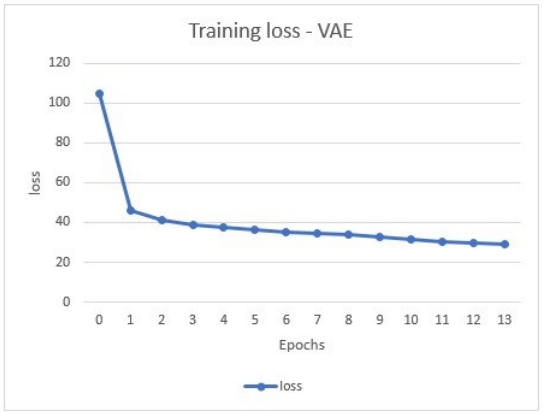


Fig 5: Training Loss - VAE

Training Loss for VAE			
Epochs	**Total loss**	**RL**	**KL loss**
1	58.0506	49.3829	8.6677
2	36.5457	28.9101	7.6356
3	33.8290	25.3999	8.4291
4	32.1465	23.0558	9.0906
5	31.2347	21.9740	9.2607
6	30.7034	21.3865	9.3168
7	30.0193	20.9659	9.3545
8	29.5871	19.1863	9.4680
9	29.1042	19.0748	9.4590
10	28.6543	19.6758	9.4283

IV. Training RNN (Bidirectional GRU)

The BGRU model produced more effective results by using less memory and training the agent faster. The BGRU's update gate performs efficiently, taking information from a previous layer. It handles information flow by deciding how much information should be taken as an input [14]. Its reset gate functions more like an LSTM

forget gate, increasing overall model efficiency and running data faster. BGRU train agent faster and perform better than LSTM with less training data.

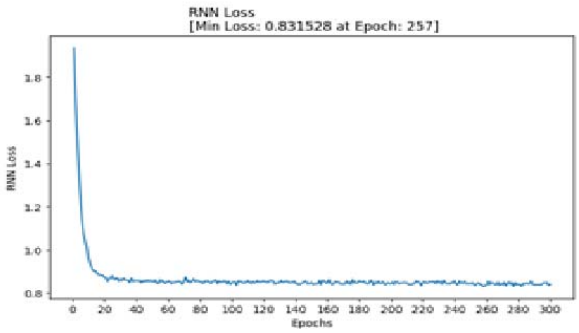


Fig 6. Training RNN (BIDIRECTION GRU)

V. Training Controller

C was based on a linear model with the weight matrix and bias presented in (1). It was trained with the evolutionary algorithm known as the covariance matrix adaptation-evolution strategy. This algorithm created copies of the model parameters with multiple random initializations (population), testing each population inside an environment, and recording the average score. After 10,000 evaluations, the best score produced by the model was 912.

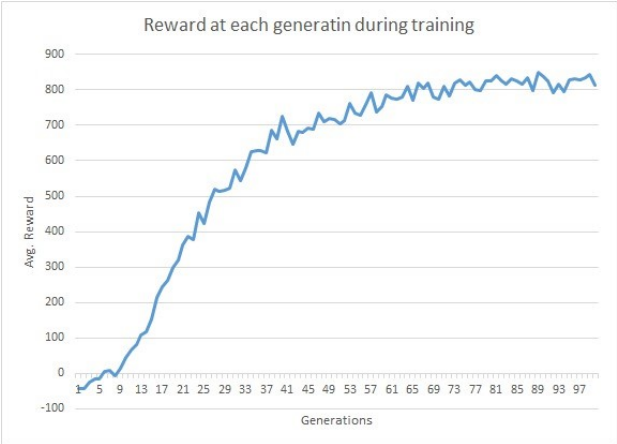


Fig 7. Training Controller Result

VI. Conclusion

A modification was made that replaced an LSTM with a GRU in an RNN world. A modified model produced more effective results by using less memory and training the agent faster. GRU contains updated gate instead of input, output and forget gate. The update gate performs efficiently and take the information from the previous layer. GRU control the information flow like an LSTM unit without using a memory unit. It exposes the full hidden content without control. GRU better results as compared to the existing model.

REFERENCES

[1] K. Chandrasekaran. (2019). Get up!: Assessing postural activity & transitions using bi-directional gated recurrent units (Bi-GRUs) on smartphone motion data. Presented at 2019 IEEE HI-POCT.

[2] S. Ali, "Table structure extraction with bi-directional gated recurrent unit networks," *2019 ICDAR,* vol. 4, no. 2, pp. 78–88, 2019.

[3] M.-H. Ahn, "An optimal control method of clamp switch for ZVS bi-directional DC-DC converter," *2016 IEEE 8th IPEMC-ECCE Asia,* pp. 88–89, 2016.

[4] T. Fan, "A new direct heart sound segmentation approach using bi-directional GRU," *2018 24th ICAC,* pp. 67–78, 2018.

[5] H. Jahangir, "Deep learning-based forecasting approach in smart grids with micro-clustering and bi-directional LSTM network," *IEEE Transactions on Industrial Electronics,* pp. 88-90, 2020.

[6] Ali. (2018). Variants of combinations of additive and multiplicative updates for GRU neural networks. Presented at 2018 26th SIU.

[7] R. Dey. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. Presented at IEEE 60th MWSCAS.

[8] L. Li, "A rumor events detection method based on deep bidirectional GRU neural network," *2018 IEEE 3rd ICIVC,* pp. 80-96, 2018.

[9] D. Ha, "World models," 2018.

[10] H. M. Lynn, "A deep bidirectional GRU network model for biometric electrocardiogram classification based on recurrent neural networks," *IEEE Access,* vol. 7, no. 3, pp. 67–90, 2019.

[11] S. Yang, "LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example," *2020 IWECAI,* pp. 78–89, 2020.

[12] G. Xiuyun. (2018). Short-term load forecasting model of GRU network based on deep learning framework. Presented at 2nd IEEE EI2.

[13] R. Fu, "Using LSTM and GRU neural network methods for traffic flow prediction," *31st YAC,* vol. 4, no. 2, pp. 78–89, 2016.

[14] A. Atassi, "The new deep learning architecture based on GRU and word2vec," *ICECOCS,* vol. 6, no. 2, pp. 89–88, 2018.

[15] Z. Qu, "A unsupervised learning method of anomaly detection using GRU," *IEEE BigComp,* 2018.