An Input Residual Connection for Simplifying Gated Recurrent Neural Networks

Nicholas I.H. Kuo
Research School of Computer Science (RSCS), the Australian National University (ANU),
Canberra, ACT, Australia
u6424547@anu.edu.au

Mehrtash Harandi
Department of ECSE, Monash University,
Melbourne, Victoria, Australia
mehrtash.harandi@monash.edu

Nicolas Fourrier
Research Center of Léonard de Vinci Pôle Universitaire,
Paris La Défense, France
nfourrier@gmail.com

Christian Walder, Gabriela Ferraro
Data61, CSIRO & RSCS, ANU,
Canberra, ACT, Australia
{christian.walder, gabriela.ferraro}@data61.csiro.au

Hanna Suominen
Data61, CSIRO[1] *& RSCS, ANU*[1]
Department of Future Technologies, University of Turku[2]
[1]Canberra, ACT, Australia & [2]Turku, Finland
hanna.suominen@anu.edu.au

***Abstract*—Gated Recurrent Neural Networks (GRNNs) are important models that continue to push the state-of-the-art solutions across different machine learning problems. However, they are composed of intricate components that are generally not well understood. We increase GRNN interpretability by linking the canonical Gated Recurrent Unit (GRU) design to the well-studied Hopfield network. This connection allowed us to identify network redundancies, which we simplified with an Input Residual Connection (IRC). We tested GRNNs against their IRC counterparts on language modelling. In addition, we proposed an Input Highway Connection (IHC) as an advance application of the IRC and then evaluated the most widely applied GRNN of the Long Short-Term Memory (LSTM) and IHC-LSTM on tasks of i) image generation and ii) learning to learn to update another learner-network. Despite parameter reductions, all IRC-GRNNs showed either comparative or superior generalisation than their baseline models. Furthermore, compared to LSTM, the IHC-LSTM removed 85.4% parameters on image generation. In conclusion, the IRC is applicable, but not limited, to the GRNN designs of GRUs and LSTMs but also to FastGRNNs, Simple Recurrent Units (SRUs), and Strongly-Typed Recurrent Neural Networks (T-RNNs).**

We release our codes at https:\\github.com\Nic5472K\IJCNN2020_IRC .

***Index Terms*—GRU, LSTM, Hopfield network, interpretability**

I. Introduction

Recurrent neural networks (RNNs) are the machine learning tools of choice for modelling data with temporal dependencies. Their applications include language modelling [1], speech recognition [2], and image generation [3]. Earlier RNN designs suffer from exploding and vanishing gradients [4] and are difficult to train. To this end, *gated* RNNs (GRNNs) were developed to stabilise gradients. GRNNs possess gated units to control how information flows in and out of the recurrent network feedback. Though extensively applied, GRNN architectures are not immediate clear, and the individual mathematical meanings of gated units are not well understood.

Previously, GRNNs have been analysed through empirical evaluations and through search studies. In *natural language processing* (NLP) of text [5], GRNN outputs responded distinctly to different types of characters; and on spoken NLP [6], GRNN gated units could be used to suggest phoneme boundaries. However, these evaluations were conducted on trained networks, and that the GRNN components possess unknown properties prior to training. The work of [7] conducted an extensive search on GRNN architectures, but was unable to yield a design with significant improvements against the canonical *gated recurrent unit* (GRU) design [8].

In this paper, we increase interpretability by returning to first principles and establish mathematical meanings for GRNN variables by linking GRUs to the well-studied Hopfield network [9]. We model GRU memories as systems of differential equations and show that they share desirable qualities possessed by Hopfield networks. Under our mathematical framework, we identify GRU redundancies and dispense them with a novel *input residual connection* (IRC). Furthermore, we present *input highway connection* (IHC) as an advance application of the IRC.

Our main contribution is showing that the IRC is generally applicable to most existing GRNN designs. We tested the GRU, the *long short-term memory* (LSTM) [10], the *simple recurrent unit* (SRU) [11], the *strongly-typed RNN* (T-RNN) [12], and the FastGRNN [13] against their IRC counterparts on language modelling [1]. In addition to NLP, we took the most widely applied GRNN of the LSTM and tested it against the IHC-LSTM on a *computer vision* (CV) task on image generation [3] and on a *meta-learning / learning to learn* (L2L) task to update another learner network [14]. Despite having less parameters, all IRC-GRNN exhibited higher generalisation than their baseline counterparts. Furthermore, all IHC-LSTMs exhibited comparative performances to LSTMs; and notably, the IHC-LSTM removed 85.4% of the parameters on the image generation task.

II. Connecting GRUs to Hopfield Networks

This section links GRUs [8] to Hopfield Networks [9]. We model GRU memories as systems of differential equations and describe their desirable properties.

A. The GRU Design

For each time step t, GRUs receive input data $\mathbf{x}_t$ to compute

$$\mathbf{i}_t, \mathbf{r}_t = \sigma(\mathbf{W}_{\{I,R\}}\mathbf{x}_t + \mathbf{U}_{\{I,R\}}\mathbf{h}_{t-1} + \mathbf{b}_{\{I,R\}}), \quad (1)$$

$$\mathbf{a}_t = \tanh(\mathbf{W}_A\mathbf{x}_t + \mathbf{U}_A(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_A), \text{ and} \quad (2)$$

$$\mathbf{h}_t = (1 - \mathbf{i}_t) \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \mathbf{a}_t. \quad (3)$$

The input gate $\mathbf{i}_t$ is employed to control the update of the network memory $\mathbf{h}_t$, also known as the recursive feedback and the hidden state. Parts of the old memory is replaced by an equal portion of the newly generated input of $\mathbf{a}_t$. The reset gate $\mathbf{r}_t$ fine-tunes recurrent connections within $\mathbf{a}_t$, whereas the parameter $\mathbf{b}$ is the bias and $\mathbf{W}$ and $\mathbf{U}$ are the respective forward and recursive connections. With the input and hidden dimensions of N and M, GRU components have dimensionalities $\mathbf{x}_t \in \mathbb{R}^N$, $\mathbf{i}_t, \mathbf{r}_t, \mathbf{a}_t, \mathbf{h}_t, \mathbf{b} \in \mathbb{R}^M$, $\mathbf{W} \in \mathbb{R}^{M\times N}$, and $\mathbf{U} \in \mathbb{R}^{M\times M}$. The symbols σ, $\tanh$, and $\odot$ represent the sigmoid, the hyperbolic tangent, and the element-wise product functions, respectively.

B. Recursive Feedbacks as Systems of Differential Equations

The GRU $\mathbf{h}_t$ is updated like variables of a system of differential equations. Eq. (3) updates $\mathbf{h}_t$ from $\mathbf{h}_{t-1}$ with the additive update $\mathbf{a}_t$. For this reason, we model $\mathbf{h}_t$ as a differential equation with the autonomous function

$$\mathbf{h}'_t = H(\mathbf{h}_{t-1}, \mathbf{a}_t). \quad (4)$$

Ideally, the function H should provide the GRU memory with robustness and plasticity. Robustness allows the network memory to generalise data under the presence of unknown noises; whereas plasticity allows the network to shape its memory as according to the input data. Both desirable qualities can be found in the theoretical physics model of the dynamic spin-glass Hopfield network [9], [15], [20].

Dynamic Hopfield networks possess attractive equilibria [20] and phase transitions [21]. Attractive equilibria are convergent singularities in the neural solution space which allow for the creation of robust network memory. Phase transitions are parametric transitional boundaries for internal degrees of freedom to successively fall out of equilibrium, and provide the network memory with the plasticity to transmute among various configurations of robust memories.

For the reasons above, we formulate H as a dynamic Hopfield model and write (4) as

$$\mathbf{h}'_t = H(\mathbf{h}_{t-1}, \mathbf{a}_t) = -\mathbf{h}_{t-1} + \mathbf{a}_t. \quad (5)$$

Equivalently, the discrete time version of Eq. (5) is

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \xi(-\mathbf{h}_{t-1} + \mathbf{a}_t), \quad (6)$$

where the positive constant ξ serves as the discretised time step. Alternatively, ξ can be a positive vector to enable different timescales. This is similar to the practice of scaling in Echo State Networks [22]. Thus, we write Eq. (6) as

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \xi \odot (-\mathbf{h}_{t-1} + \mathbf{a}_t), \quad (7)$$

and rearranging Eq. (7) yields

$$\mathbf{h}_t = (1 - \xi) \odot \mathbf{h}_{t-1} + \xi \odot \mathbf{a}_t. \quad (8)$$

By comparing Eq. (3) to Eq. (8), we see that the GRU $\mathbf{h}_t$ is updated as a discretised dynamic Hopfield network. Under this mathematical framework, the input gate $\mathbf{i}_t$ plays a similar role to ξ and serves as the dynamic time step. This connection thus increases the individual interpretability of GRU components.

However, readers should note that Eq. (8) is not a specific update for any RNN. Instead, it represents a class of RNN memory updates with the desirable properties of Hopfield network-like robustness and plasticity. That is, the GRU formulation can only be recovered if we were to choose to define ξ as $\mathbf{i}_t$ in Eq. (1), and $\mathbf{a}_t$ as in Eq. (2) with $\mathbf{r}_t$ as in Eq. (1).

III. An Input Residual Connection

As we discussed in **Section** II, the GRU is one design that updates its memory as according to Eq. (8). That is, there are infinitely many ways to formulate the update of an RNN memory to achieve the same, or potentially better, level of Hopfield network-like robustness and plasticity. In this section, we look for the lightest possible simplification for the GRU.

Eq. (8) highlights two types of GRU redundancies. One relates to the network non-linearities, and the second relates to the recursive network feedback.

A. An Analysis on the Non-linearity of $\mathbf{a}_t$

Eq. (8) required neither the step size ξ nor the update $\mathbf{a}_t$ to undergo non-linear transformation. The hyperbolic tangent function of $\mathbf{a}_t$ can be limiting for the network because it bounds $\mathbf{a}_t \in [-1, 1]$. The upper and lower bound restrict $\mathbf{a}_t$ from distinguishing pre-activated neurons with large magnitudes. Thus, the hyperbolic tangent function in $\mathbf{a}_t$ restricts a piece of diverse information to be added towards the hidden state $\mathbf{h}_t$. For this reason, our first simplification on the GRU $\mathbf{a}_t$ of Eq. (2) is

$$\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t + \mathbf{U}_A(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_A. \quad (9)$$

However, for a very different reason, we choose to keep the sigmoid activation function in $\mathbf{i}_t$.

B. An Analysis on the Non-linearity of $\mathbf{i}_t$

We keep the sigmoid activation function in $\mathbf{i}_t$ on the same grounds as having small learning rates for gradient descents. The update of $\mathbf{h}_t$ in Eq. (6) is similar to gradient descent methods, which in its simplest form, updates a neural network with parameters θ_t via

$$\theta_t = \theta_{t-1} + \xi(-\nabla_{\theta_{t-1}}\mathcal{L}_t), \quad (10)$$

where $\nabla_{\theta_{t-1}}\mathcal{L}_t$ is the gradient of the optimised loss at time step t with respect to the existing parameters θ_{t-1}.

Both Eq. (6) and the gradient descent of Eq. (10) comply with the properties of a vector field. The vector field in gradient descent can be described with the loss landscape. Eq. (10) first specifies the direction to update the network parameters via $-\nabla_{\theta_{t-1}}\mathcal{L}_t$, and then takes a step ξ towards this specific direction on the loss landscape. In Eq. (10), ξ is known as the learning rate. Large learning rates can cover more ground in each step but risks overshooting minima. Similarly, Eq. (6) possess entities with equivalent importance to minima in Eq. (10); such entities are known as the attractive equilibria [19].

A point $\mathbf{z}$ in the network solution space is an equilibrium if, for all solutions which start or enter $\mathbf{z}$, remain on $\mathbf{z}$ for all time. An equilibrium is said to be attractive if nearby solutions were to be updated towards it. That is, they are singularities that behave as the relative minima on different regions of the vector field; they describe stable behaviours within the solution space. For this reason, a small step size of ξ in Eq. (6) is required in order for the network memory to traverse within the vicinity of an attractive $\mathbf{z}$. This will help to stabilise network memories for achieving robustness. Since a comparison between Eq. (8) and Eq. (3) showed that the input gate $\mathbf{i}_t$ acts as the dynamic time step of the GRU, we decide to keep the sigmoid function in Eq. (3) to restrict $\mathbf{i}_t \in [0, 1]$.

C. An Analysis on The Recurrent Feedback

Judging by Eq. (8), it is not immediately clear how the recursive pre-activated neural values of $\mathbf{U}\mathbf{h}_{t-1}$ should be designed in Eq. (1) and Eq. (2). Thus, it is necessary to rethink whether $\mathbf{a}_t$ and $\mathbf{i}_t$ should take the forms of

$$\begin{cases} \mathbf{i}_t &= \mathbf{i}_t(\mathbf{x}_t, \mathbf{h}_{t-1}) \\ \mathbf{a}_t &= \mathbf{a}_t(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{r}_t) \end{cases} \tag{11}$$

or as

$$\begin{cases} \mathbf{i}_t &= \mathbf{i}_t(\mathbf{x}_t) \\ \mathbf{a}_t &= \mathbf{a}_t(\mathbf{x}_t) \end{cases}. \tag{12}$$

Since Eq. (11) and Eq. (12) both suffice for Eq. (8), we hypothesise that the recursive $\mathbf{U}\mathbf{h}_{t-1}$ is not directly required for formulating $\mathbf{i}_t$, $\mathbf{r}_t$, and $\mathbf{a}_t$. Hence, we simplify Eq. (1) as

$$\mathbf{i}_t, \mathbf{r}_t = \sigma(\mathbf{W}_{\{I,R\}}\mathbf{x}_t), \tag{13}$$

and further simplify Eq. (2) from Eq. (9) as

$$\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t. \tag{14}$$

Note, the biases were also removed from Eq. (13) and Eq. (14). We made this change because the mathematical framework of Eq. (8) does not require them. Interestingly, an identical form for $\mathbf{a}_t$ in Eq. (14) can be found in the SRU [11]. However, the sole concern for [11] was the proposal of a highly parallelisable RNN architecture, and their modifications towards $\mathbf{a}_t$ were provided without explanations.

After removing the recursive $\mathbf{U}\mathbf{h}_{t-1}$, this simplified version of the GRU is no longer a real RNN. We describe where we restore the relinquished recurrent feedback in the next section.

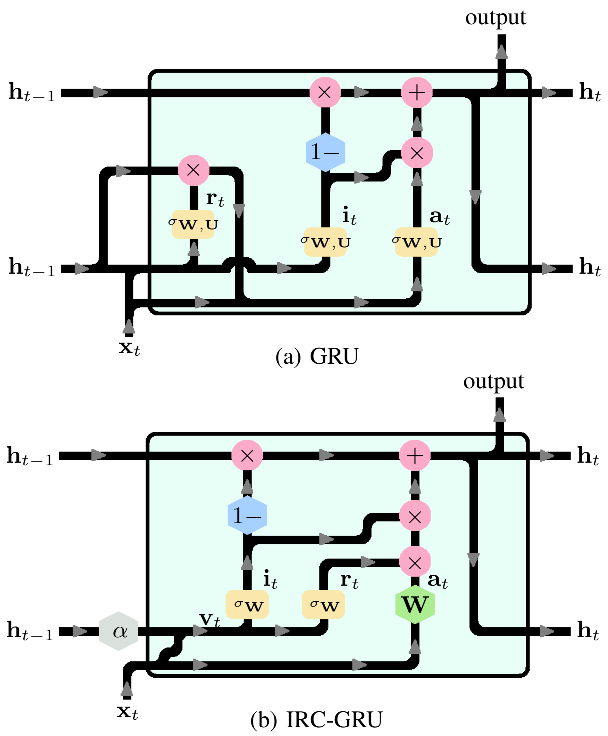

Fig. 1. The GRU architecture and the IRC-GRU architecture

D. An Input Residual Connection

Thus far, we have established the connection between the dynamic Hopfield network and the GRU in **Section** II, and used this connection to motivate the simplified $\mathbf{i}_t$, $\mathbf{r}_t$, and $\mathbf{a}_t$ provided in Eq. (13) and Eq. (14). Removing $\mathbf{U}\mathbf{h}_{t-1}$ and biases $\mathbf{b}$ have resulted in a large parametric reduction to the GRU. However, this variant of the GRU does not utilise the recurrent feedback $\mathbf{h}_t$ and hence is not a RNN. Here, we introduce our novel input residual connection (IRC) design to restore $\mathbf{h}_t$ while maintaining a light architecture.

Since Eq. (12) suffices for Eq. (8), we hypothesise that GRUs function properly with $\mathbf{W}\mathbf{x}_t$ as the sole pre-activated neural values. For this reason, we see the recursive $\mathbf{U}\mathbf{h}_{t-1}$ as regularisations to $\mathbf{W}\mathbf{x}_t$ for easing the training of the feedforward input in the activation functions. Under this interpretation, we propose to regularise $\mathbf{W}\mathbf{x}_t$ before they enter the activation functions. We replace $\mathbf{x}_t$ in the gated units of Eq. (13) with the regulated inputs of $\mathbf{v}_t$, $\mathbf{x}_t \leftarrow \mathbf{v}_t$, such that

$$\mathbf{v}_t = \mathbf{x}_t + \alpha \odot (\mathbf{U}_V\mathbf{h}_{t-1}). \tag{15}$$

Variable $\mathbf{v}_t$ connects the recursive $\mathbf{U}_V\mathbf{h}_{t-1}$ to the input $\mathbf{x}_t$ with a residual connection [23] to provide $\mathbf{x}_t$ with a learnt discretised update based on the network memory. Since the residual connection is applied to the input, we call this technique an IRC. The new variables have dimensionality $\alpha \in \mathbb{R}^N$ and $\mathbf{U}_V \in \mathbb{R}^{N\times M}$. Note, the recursive connections in the original GRU design of Eq. (1) and Eq. (2) have dimensionality $\mathbf{U} \in \mathbb{R}^{M\times M}$ and are typically much larger than the size of that of $\mathbf{U}_V$ in Eq. (15). The vector α is an idea which we borrowed from the FastGRNN [13]. This

TABLE I
A comparison between baseline GRNNs against IRC-GRNNs

Traditional design	Simplified design
GRU [8]	IRC-GRU (Ours)
$\mathbf{i}_t, \mathbf{r}_t = \sigma(\mathbf{W}_{\{I,R\}}\mathbf{x}_t + \mathbf{U}_{\{I,R\}}\mathbf{h}_{t-1} + \mathbf{b}_{\{I,R\}}),$ (1) $\mathbf{a}_t = \tanh(\mathbf{W}_A\mathbf{x}_t + \mathbf{U}_A(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_A),$ and (2) $\mathbf{h}_t = (1 - \mathbf{i}_t) \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \mathbf{a}_t.$ (3)	$\mathbf{v}_t = \mathbf{x}_t + \alpha \odot (\mathbf{U}_V\mathbf{h}_{t-1}),$ (15) $\mathbf{i}_t, \mathbf{r}_t = \sigma(\mathbf{W}_{\{I,R\}}\mathbf{v}_t),$ (16) $\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t,$ and (14) $\mathbf{h}_t = (1 - \mathbf{i}_t) \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot (\mathbf{r}_t \odot \mathbf{a}_t).$ (17)
LSTM [10]	IRC-LSTM (Ours)
$\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t = \sigma(\mathbf{W}_{\{F,I,O\}}\mathbf{x}_t + \mathbf{U}_{\{F,I,O\}}\mathbf{h}_{t-1} + \mathbf{b}_{\{F,I,O\}}),$ (18) $\mathbf{a}_t = \tanh(\mathbf{W}_A\mathbf{x}_t + \mathbf{U}_A\mathbf{h}_{t-1} + \mathbf{b}_A),$ (19) $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{a}_t,$ and (20) $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).$ (21)	$\mathbf{v}_t = \mathbf{x}_t + \alpha \odot (\mathbf{U}_V\mathbf{h}_{t-1}),$ (22) $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t = \sigma(\mathbf{W}_{\{F,I,O\}}\mathbf{v}_t),$ (23) $\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t,$ (24) $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{a}_t,$ and (25) $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).$ (26)
SRU [11]	IRC-SRU (Ours)
$\mathbf{f}_t, \mathbf{p}_t = \sigma(\mathbf{W}_{\{F,P\}}\mathbf{x}_t + \omega_{\{F,P\}} \odot \mathbf{c}_{t-1} + \mathbf{b}_{\{F,P\}}),$ (27) $\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t$ (28) $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{a}_t,$ and (29) $\mathbf{h}_t = \mathbf{p}_t \odot \mathbf{c}_t + (1 - \mathbf{p}_t) \odot \mathbf{x}_t.$ (30)	$\mathbf{v}_t = \mathbf{x}_t + \alpha \odot (\omega_V \odot \mathbf{h}_{t-1}),$ (31) $\mathbf{f}_t, \mathbf{p}_t = \sigma(\mathbf{W}_{\{F,P\}}\mathbf{v}_t),$ (32) $\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t,$ (33) $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{a}_t,$ and (34) $\mathbf{h}_t = \mathbf{p}_t \odot \mathbf{c}_t + (1 - \mathbf{p}_t) \odot \mathbf{x}_t.$ (35)
T-LSTM [12]	IRC-T-LSTM (Ours)
$\mathbf{f}_t, \mathbf{o}_t = \sigma(\mathbf{W}_{\{F,O\}}\mathbf{x}_t + \mathbf{U}_{\{F,O\}}\mathbf{x}_{t-1} + \mathbf{b}_{\{F,O\}}),$ (36) $\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t + \mathbf{U}_A\mathbf{x}_{t-1} + \mathbf{b}_A,$ (37) $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{a}_t,$ and (38) $\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t.$ (39)	$\mathbf{v}_t = \mathbf{x}_t + \alpha \odot (\omega_V \odot \mathbf{h}_{t-1}),$ (40) $\mathbf{f}_t, \mathbf{o}_t = \sigma(\mathbf{W}_{\{F,O\}}\mathbf{v}_t),$ (41) $\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t,$ (42) $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{a}_t,$ and (43) $\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t.$ (44)
FastGRNN [13]	IRC-FastGRNN (Ours)
$\mathbf{f}_t = \sigma(\mathbf{W}_F\mathbf{x}_t + \mathbf{U}_F\mathbf{h}_{t-1} + \mathbf{b}_F),$ (45) $\mathbf{a}_t = \tanh(\mathbf{W}_A\mathbf{x}_t + \mathbf{U}_A\mathbf{h}_{t-1} + \mathbf{b}_A),$ (46) $\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (\beta(1 - \mathbf{f}_t) + \kappa) \odot \mathbf{a}_t.$ (47)	$\mathbf{v}_t = \mathbf{x}_t + \alpha \odot (\mathbf{U}_V\mathbf{h}_{t-1}),$ (48) $\mathbf{f}_t = \sigma(\mathbf{W}_F\mathbf{v}_t),$ (49) $\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t,$ and (50) $\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + \beta(1 - \mathbf{f}_t) \odot \mathbf{a}_t.$ (51)

Note: With IRC highlighted in colour red, and network redundancies highlighted in colour brown; best viewed in colour.

vector contains trainable weights where $0 \leq \alpha_j \leq 1$ for $j = 1, \ldots, N$ and are parameterised by the sigmoid function. The vector α is employed to limit the extent that the recurrent feedback modifies the features of $\mathbf{x}_t$.

To summarise, the IRC-GRU consists
i) the IRC regulated input of Eq. (15)

$$\mathbf{v}_t = \mathbf{x}_t + \alpha \odot (\mathbf{U}_V\mathbf{h}_{t-1}),$$

ii) which substitutes $\mathbf{x}_t$ in the simplified gates of Eq. (13)

$$\mathbf{i}_t, \mathbf{r}_t = \sigma(\mathbf{W}_{\{I,R\}}\mathbf{v}_t), \tag{16}$$

iii) the simplified internal input of Eq. (14)

$$\mathbf{a}_t = \mathbf{W}_A\mathbf{x}_t, \text{ and}$$

iv) the hidden state update

$$\mathbf{h}_t = (1 - \mathbf{i}_t) \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot (\mathbf{r}_t \odot \mathbf{a}_t). \tag{17}$$

Readers should note that, unlike the gated units of Eq. (16), the IRC-GRU $\mathbf{a}_t$ does not substitute $\mathbf{x}_t$ with $\mathbf{v}_t$. The reason is because that, similar to the dynamic Hopfield network of Eq. (5), the memory $\mathbf{h}_t$ is supposed to receive a new source of information $\mathbf{a}_t$ that is dissimilar to the existing memory. See **Fig. 1** for a figurative comparison of the architectural differences between the GRU and the IRC-GRU.

IV. IRC-/IHC-GRNNs

In the previous section, we addressed IRC as a technique for simplifying the GRU. In this section, we introduce other GRNN baselines which we will use in our experimental section. The alternative GRNN designs in this section are the LSTM [10], the SRU [11], the T-LSTM [12], and the FastGRNN [13]. We show that the IRC is generally applicable to all GRNNs which we consider, and we contrast the baseline designs against their IRC simplified counterparts in **Table I**. This section mainly focuses on the differences between the baseline models and their IRC counterparts. Refer to **Section** VI for an in-depth discussion.

A. LSTMs and IRC-LSTMs

The LSTM architecture is given in Eq. (18) to Eq. (21). It possesses 3 gates and 2 network memories. The cell $\mathbf{c}_t$

is similar to the GRU $\mathbf{h}_t$ in Eq. (3). The LSTM $\mathbf{h}_t$ is $\mathbf{c}_t$ transformed with controlled exposure from the output gate $\mathbf{o}_t$. Unlike GRUs of Eq. (3), the LSTM $\mathbf{c}_t$ is updated with a forget gate $\mathbf{f}_t$ and an input gate $\mathbf{i}_t$. Therefore, in the LSTM terminologies, GRUs couple their forget and input gate to update their network memories. The IRC adapts LSTMs in a similar way to GRUs, and is given in Eq. (22) to Eq. (26).

B. SRUs and IRC-SRUs

The SRU architecture is given in Eq. (27) to Eq. (30). SRUs regulate their pre-activated recursive neural values linearly through ω. Thus, the IRC-SRU, as given in Eq. (31) to Eq. (35), linearly regulates the memory through ω_V with trainable weights $-1 \le \omega_{V_j} \le 1$ for $j = 1, \ldots, N$. The weights are parameterised by the hyperbolic tangent function.

C. T-LSTMs and IRC-T-LSTMs

The T-LSTM architecture is given in Eq. (36) to Eq. (39); and the IRC-T-LSTM architecture is given in Eq. (40) to Eq. (44). Eq. (40) is identical to Eq. (31) because matrices $\mathbf{W}$ and $\mathbf{U}$ in Eq. (36) and in Eq. (37) have the same dimensionality.

D. FastGRNNs and IRC-FastGRNNs

The FastGRNN architecture is given in Eq. (45) to Eq. (47); and the IRC-FastGRNN architecture is given in Eq. (48) to Eq. (51). FastGRNNs possess one gated unit, but regulate their memory with the additional trainable constants $0 \le \beta, \kappa \le 1$ parameterised by the sigmoid function.

E. A Highway Connection

Here we introduce the more powerful IRC variant of the IHC. As the name implies, IHC prepares a regulated input with a highway connection [24]. The variable $\mathbf{v}_t$ of Eq. (15) is alternatively computed as

$$\mathbf{v}_t = (1 - \mathbf{g}_t) \odot \mathbf{x}_t + \mathbf{g}_t \odot (\mathbf{U}_V \mathbf{h}_{t-1}) \text{ where} \tag{52}$$

$$\mathbf{g}_t = \sigma(\mathbf{W}_G \mathbf{x}_t + \Gamma \mathbf{h}_{t-1} + \mathbf{b}_G). \tag{53}$$

The components have dimensionality $\mathbf{g}_t, \mathbf{b}_G \in \mathbb{R}^N$, and $\Gamma \in \mathbb{R}^{M \times N}$. Similar to $\mathbf{U}_V$ in Eq. (15), the size of matrix Γ is typically much smaller than $\mathbf{U} \in \mathbb{R}^{M \times M}$ of Eq. (1) and Eq. (2). The parametric vector of α in Eq. (15) is replaced with the gate $\mathbf{g}_t$. The gate $\mathbf{g}_t$ provides a two-fold advantage over α. First, unlike the static control exerted by α, $\mathbf{g}_t$ limits the extent of modification from $\mathbf{U}_V \mathbf{h}_{t-1}$ to $\mathbf{x}_t$ in a dynamic manner based on the memory of the network. Second, $\mathbf{x}_t$ is modified with $1 - \mathbf{g}_t$; and in principle, this allows the network to learn when to extrapolate information based on the recurrent feedback $\mathbf{U}_V \mathbf{h}_{t-1}$ rather than the input $\mathbf{x}_t$. We reserve IHC for LSTMs in our experiments, and replace Eq. (22) with Eq. (52) and Eq. (53) if the IHC is used instead.

V. Experiments

The GRNNs included in this section are the GRU and all alternative designs mentioned in **Section** IV. The focus of our study is to show that the IRC and the IHC enable GRNNs to achieve comparative performances to their baseline counterparts while removing network redundancies. It is noteworthy however, that our results actually demonstrate a superior generalisation performance for our IRC-GRNNs in comparison with their baseline counterparts. We conjecture that this improvement may be due to the theoretical advantages of the low complexity of simpler models [32].

This section includes 3 tasks. First, we tested the GRNNs against their IRC counterparts on language modelling [1]. Then, we tested LSTM against IHC-LSTM on image generation [3], and L2L to update another learner network [14].

Image generation and L2L were reserved for LSTMs for fairness. GRNNs have been applied on a wide range of tasks, and to our knowledge, most of the baseline GRNNs included in this paper have previously been tested on language modelling; but such is not the case for image generation and for L2L. For this reason, we reserved image generation and L2L for the most extensively applied GRNN of LSTM.

The 3 tasks were carefully chosen to analyse different aspects of GRNNs. The GRNNs were employed as generative language models to test for their abilities to represent and to manipulate high dimensional probability distributions. The GRNNs were employed as decoders in image generation to test for their reconstruction powers. Last, the GRNNs operated for variable lengths on L2L to test their robustness against their regularisation of compounded prediction errors conditioned on unseen sequential context at test time.

A. GRNNs vs IRC-GRNNs on Language Modelling

Background. *Language models* employ RNNs to construct probabilistic predictions of the next word given preceding ones. For this task, we used the Penn Treebank (PTB) dataset [25], which consists of 929K training words, 73K validation words, and 82K test words, with vocabulary size of 10K words.

Setup. The codes of our GRNNs were based on the popular AWD-LSTM [1] model, and all GRNNs and their IRC counterparts were trained for 50 epochs. The hyperparameters were not fixed across all models; instead, the dimensionality and the amount of layers were taken from their respective original paper (if provided). Below, we use the tuple of (ED, HD) to denote the combination of word embedding dimension and hidden dimension.

Following [12], we tested 2-layer IRC-GRU and 3-layer IRC-T-LSTM with (650, 650). The FastGRNN paper [13] tested language models with training settings significantly different to [1][1]. Thus we implemented 1-layer and 2-layer FastGRNN and IRC-FastGRNN with (650, 650). The SRU paper [11] did not include word-level language modelling as a task. Thus we tested 3-layer SRU and IRC-SRU with (1150, 1150). Last, we trained 3-layer LSTM and IHC-LSTM with (400, 1150) for 800 epochs.

Results. All of our IRC-GRNNs achieved lower perplexities than their baseline counterparts (**Table II**). Thus, all IRC-GRNNs had better generalisation ability than their GRNN

[1]Refer to page 6 of Section 3.2.1 of [13] for their training scheme, and Refer to page 22 of [13] for additional hyperparametric changes.

TABLE II
PERPLEXITY ON THE PTB DATASET.

Model	Test PPL	#RNN-Params	Reduction
	Lower is better		
3-layers, (ED, HD) = (400, 1150)			
LSTM [1]	57.3	24.9M	–
LSTM (Ours)	58.32	24.9M	–
IHC-LSTM (Ours)	57.44	**15.1M**	39.2%
1-layers, (ED, HD) = (650, 650)			
FastGRNN$^{\pi}$ [13]	116.11	–	–
FastGRNN (Ours)	88.48	1.7M	–
IRC-FastGRNN (Ours)	*85.60*	**1.3M**	25.0%
2-layers, (ED, HD) = (650, 650)			
GRU [12]	93.44	5.1M	–
T-LSTM [12]	81.52	5.1M	–
FastGRNN$^{\pi}$ [13]	106.23	–	–
FastGRNN (Ours)	89.30	3.4M	–
IRC-GRU (Ours)	*76.51*	**3.4M**	33.4%
IRC-T-LSTM (Ours)	*79.56*	**2.5M**	50.0%
IRC-FastGRNN (Ours)	*85.60*	**2.5M**	25.0%
3-layers, (ED, HD) = (1150, 1150)			
SRU (Ours)	92.42	12.0M	–
IRC-SRU (Ours)	*86.70*	12.0M	0.06%

Note:
PPL = Perplexity; M = Million;
(ED, HD) = (Word embedding dimension, Hidden dimension);
π = With different (ED, HD) to the original paper, see text.
We emphasised the reduced parameters of our IRC-/IHC-GRNNs in bold font, and highlighted their better performances in italic font.

counterparts. In addition, this was achieved while IRC-GRNNs had lower amount of learnable parameters. Redundancy removal is most remarkable when the hidden dimension is much larger than the word embedding dimension. The IRC-T-LSTM removed up to 50.0% of the T-LSTM redundancies; while the IRC-SRU only removed up to 0.06% SRU redundancies. As a final note, we showed that our IHC-LSTM was capable of achieving a competitive sub-60 perplexity. This matched the result reported in [1] despite a near-40% parametric reduction.

B. LSTMs vs IHC-LSTMs on Image Generation

Background. [3] proposed the *deep recurrent attention writer* (DRAW) architecture to generate highly realistic natural images. DRAW employed LSTM-decoders to sequentially adjust the generated pixels of an image in order to provide the canvas with more details. This experiment used the multi-digit Street View House Numbers (SVHN) dataset [27] which contains 231,053 training images, and 4,701 validation images.

Setup. Following [3], we employed single-layer LSTM and IHC-LSTM with 800-hidden dimension to decode encoded pixel information. The encoded inputs were latent vectors of $\mathbb{R}^{100}$ and the canvas was reassessed by the LSTM for 32 iterations as according to the paper. We used the pre-cropped Format 2 dataset provided by [27], and thus the canvas of our input images were of boxes of pixels of size 32×32.

Results. No metric was given in [3] for assessing the quality of the generated SVHN images. However, as we show in **Fig. 2** overleaf, the IHC-LSTMs were capable of generating images of similar qualities to those of the LSTM. The LSTM-decoders had 2.88 million parameters, while the IHC-LSTM-decoders had only 0.42 million, thereby removing 85.4% parameters.

C. LSTMs vs IHC-LSTMs on L2L

Background. Learning to learn (L2L) was selected to test IHC-LSTM's ability to run online for a variable length of instances. The work of [14] trained optimisee learner networks with LSTM-based optimisers, and found that the optimisees were trained faster than traditional optimisation methods such as SGD [28] and ADAM [29], and yielded lower losses.

Setup. We followed [14] and classified the CIFAR-10 dataset [30] under the L2L framework. The optimisee network in [14] was a relatively small network with 3 convolutional layers with max pooling followed by a fully-connected layer. Their LSTM-based optimiser was trained to update their optimisee network for 100 steps; and during test time, was set to run freely to optimise the optimisee for 1000 steps. The iteration in test time is significantly longer than that in train time to test for the LSTM's robustness against unseen error during training. We selected to train the much larger ResNet-34 [23] optimisee network. Our ResNet-34 contained one entry convolutional layer, 16 ResNet building blocks, followed by a fully-connected layer. Each ResNet building block contained a pair of filters, and a shortcut connection connected the input to the output of every block. We updated weights of the ResNets according to [31] (a follow-up paper of [14]).

Results. The average result over training 50 ResNet-34s are shown in **Fig. 3** overleaf. Both GRNN-optimisers significantly outperformed the hand-crafted SGD and ADAM. The IHC-LSTM-based optimiser yielded ResNets with lower losses than the LSTM-based optimiser. With 20-input dimension and 20-hidden dimension, the LSTM-based optimiser had 3.28 thousands parameters, while the IHC-LSTM-based optimiser had 2.82 thousands, thereby removing 14.0% of parameters.

VI. RELATED WORKS AND DISCUSSION

Our related works section mainly covers two different types of studies. We review existing interpretations on GRNNs, and review GRNN architectures simpler than the canonical GRU design. Then, we provide an in-depth comparison on the findings in those papers to our own mathematically driven approach, in which we connected GRUs to dynamic Hopfield networks in order to propose the generally applicably GRNN simplification of IRC and IHC.

Search study and Empirical study

Here we provide more details to the literature mentioned in **Section** I. Paper [7] employed a large scale mutation scheme starting with LSTMs and GRUs as baseline models. Random edits were applied on activation functions, on element-wise operations, and on node values. The paper was unable to yield a design significantly better than the LSTM and the GRU. LSTMs have shown to possess the capability of capturing data attributes. In speech [6], the average forget gate activation could be used to suggest phoneme boundaries; and in text [5], hidden states responded differently to normal English characters and to special C-programming characters.

Ablation study

Paper [16] presented an in-depth ablation study on LSTMs.

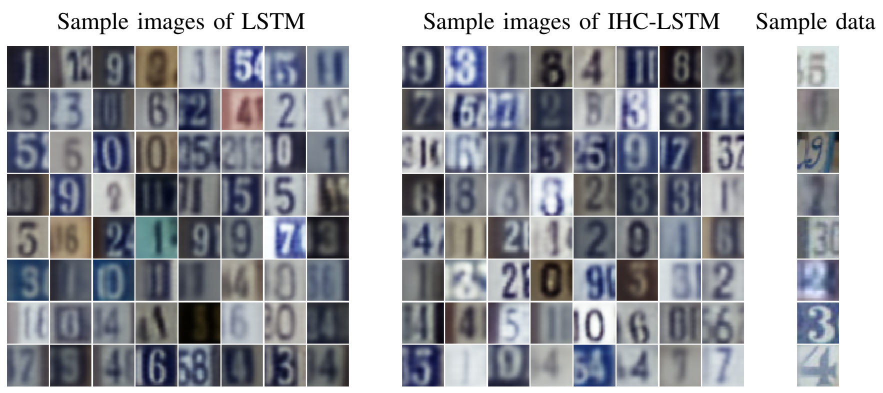

Fig. 2. DRAW for generating street-view-house-number images. With style of presentation adopted from [3] (p. 7, Fig. 9).

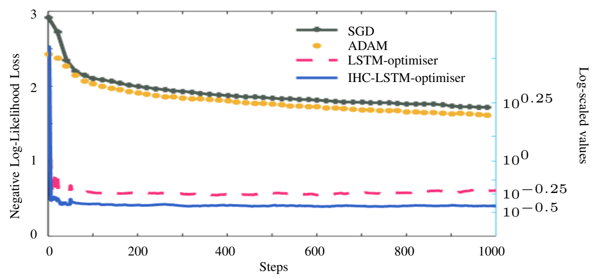

Fig. 3. Optimising ResNet-34 on Cifar10

The paper found that significant performance reductions occurred by removing the output activation function (no OAF) or by removing the forget gate (NFG). The authors considered the two lesions to be caused by similar reasons. They hypothesised that OAF prevented unbounded cell states from destabilising learning; and similarly, NFGs could triggered unbounded cell growths for not relinquishing old cell values. They conjectured that GRUs do not need OAF because the cell is bounded by the coupling of the input and the forget gate.

Simplified GRNNs

GRNN simplification is an active field of study. SRU [11] achieved convolutional neural network-like parallelism with high reliance on element-wise computations. As shown in Eq. (27), they replaced $\mathbf{U}\mathbf{h}_{t-1}$ with $\omega \odot \mathbf{h}_{t-1}$. T-RNNs [12] reconfigured the recurrent network feedback to ensure dimensional homogeneity. As shown in Eq. (36) and in Eq. (37), they replaced $\mathbf{U}\mathbf{h}_{t-1}$ with $\mathbf{U}\mathbf{x}_{t-1}$; and T-LSTMs outperformed LSTMs on language modelling. In another paper, the kilobyte sized FastGRNN [13] achieved LSTM-like performance with a single gate by regularising the input gate to address ill-conditioned gradients. As shown in Eq. (47), they replaced $1 - \mathbf{f}_t$ with $\beta(1 - \mathbf{f}_t) + \kappa$.

Gated units are not the only approach to improve (G)RNNs. Clockwork RNNs (CWRNNs) [17] are simple recurrent units [18] without gated units and in favour for a periodic weight nullification masking mechanic. CWRNNs were shown to achieve better performances than LSTMs. By vacating the gated units, CWRNNs possessed significantly less parameters than LSTMs. However, the CWRNN masking mechanic required users to make assumptions on the data structure.

Discussion

The mutation scheme in paper [7] restricted candidate RNNs from both the creation of new gated units and that of new hidden states. As a result, their top 3 discovered architectures all share very close resemblances to the GRU design. As shown in Eq. (15), our IRC idea introduced a new regulated variable $\mathbf{v}_t$, and the experiments in **Section** V showed that IRC-GRNNs enjoyed fewer parameters and had better generalisation ability than their baseline GRNN counterparts. This suggests that optimal RNN designs might be networks with multiple small units, instead of networks with few large units.

Both the empirical studies of [5] and [6] were conducted on trained networks, and the mathematical properties of the LSTM components were not addressed. This was supported by [5][2], as they admitted having difficulties in explaining the behaviours of gated units in different LSTM layers. In **Section** II, we showed that GRUs update their network memories $\mathbf{h}_t$ as discretised dynamic Hopfield networks. In addition, we showed that the GRU input gate $\mathbf{i}_t$ behaved as adaptive time steps for the GRU $\mathbf{h}_t$ by comparing Eq. (3) to Eq. (8). The same interpretation is also applicable to the LSTM gated units. This will be separately discussed later in this section.

There are several reasons to simplify GRNNs. The main concern of the SRU paper was to optimise RNN parallelism for GPU/CUDA programming, but their modifications were not based on mathematical grounds. In contrast, the papers that introduced T-RNNs and FastGRNNs presented mathematical justifications for their network alterations. However, the analysis in FastGRNN mainly addressed the stability in learning, and that explanations for the individual importance of FastGRNN components were not in their scope. Furthermore, while T-LSTMs performed better than LSTMs on language modelling, T-GRUs performed considerably worse than GRUs on language modelling. In **Section** III, we showed that our IRC simplification was derived from a dynamic mathematical framework, and is generally applicable to the GRU, the LSTM, and all other GRNNs mentioned in this paper.

[2]Refer to page 5 of Section 4.2 of [5].

Note, the CWRNN is not a GRNN. The IRC is thus inapplicable on the CWRNN; and for this reason, it was not included in our experimental section.

Last, our findings complemented [16], and explains the architectural differences between GRUs and LSTMs. As stated in Eq. (3), GRUs update their hidden states via

$$\mathbf{h}_t = (1 - \mathbf{i}_t) \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \mathbf{a}_t;$$

in contrast, LSTMs update their hidden state along with their cell state as according to Eq. (20) and Eq. (21) where

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{a}_t,$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).$$

Under our mathematical framework from Eq. (5) to Eq. (8), we found that the GRU $\mathbf{i}_t$ served as the dynamic time step for updating the GRU $\mathbf{h}_t$. Hence, given the presence of $\mathbf{i}_t$, GRUs do not require an additional unit $\mathbf{f}_t$. By contrast, LSTMs require the extra unit $\mathbf{o}_t$ to restrict the exposure of the network memory from potential step size mismatches. That is, when $\mathbf{f}_t^j > 1 - \mathbf{i}_t^j$ occurs for any neuron $j = 1, \dots, N$ in the forget and input gate, the relinquishment rate of $\mathbf{c}_{t-1}^j$ is lower than the replenishment rate of $\mathbf{a}_t^j$, and will cause $\mathbf{c}_t^j$ to increase uncontrollably and ultimately destabilising learning. The same conjecture is also applicable on the performance reduction in [16] exhibited by LSTM with no OAF.

VII. Conclusion

This paper modelled GRU memories as discretised dynamic Hopfield networks. Our mathematical framework interpreted gated units as adaptive time steps, and highlighted redundant network components. Based on our insights, we introduced IRC and IHC as generally applicable techniques to simplify GRUs, LSTMs, SRUs, T-RNNs, and FastGRNNs.

For our experiments, we evaluated IRC-GRNNs against their GRNN counterparts in language modelling, and tested IHC-LSTMs against vanilla LSTM in image generation and L2L. All IRC-GRNNs showed better generalisation abilities than their respective baselines, and all IHC-LSTMs exhibited comparative performances to LSTMs. Most notably, the IHC-LSTM removed up to 85.4% LSTM parametric redundancies on the task of image generation.

Our work connected GRNNs to well-studied mathematical models. Hence it is likely that more mathematical theories, such as bifurcation analysis [19], can be applied to forward our understanding for recurrent neural networks.

Acknowledgement

This research was supported by the Australian Government Research Training Program (AGRTP) Scholarship.

References

[1] S. Merity, N.S. Keskar, and R. Socher, "**Regularizing and Optimizing LSTM Language Models**". In ICLR 2018.

[2] M. Sundermeyer, R. Schlüter, and H. Ney, "**LSTM Neural Networks for Language Modeling**". In INTERSPEECH 2012.

[3] K. Gregor, I. Danihelka, A. Graves, D.J. Rezende, and D. Wierstra, "**DRAW: A recurrent neural network for image generation**". In ICML 2015.

[4] Y. Bengio, P. Simard, and P. Frasconi, "**Learning Long-Term Dependencies with Gradient Descent is Difficult**". In IEEE Trans. on Neural Networks and Learning Systems 1994.

[5] A. Karpathy, J. Johnson, and F. Li, "**Visualizing and Understanding Recurrent Networks**". CoRR, 2015.

[6] Z. Wu and S. King, "**Investigating Gated Recurrent Networks for Speech Synthesis**". In ICASSP 2016.

[7] R. Jozefowicz, W. Zaremba, and I. Sutskever, "**An Empirical Exploration of Recurrent Network Architectures**". In ICML 2015.

[8] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "**Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation**". In EMNLP 2014.

[9] J.J. Hopfield, "**Neural Networks and Physical Systems with Emergent Collective Computational Abilities**". In James Anderson and Edward Rosenfeld (eds.), Neurocomputing: Foundations of Research 1988.

[10] S. Hochreiter and J. Schmidhuber, "**Long Short-Term Memory**". In Neural Comput. 1997.

[11] T. Lei, Y. Zhang, S.I. Wang, H. Dai, and Y. Artzi, "**Simple Recurrent Units for Highly Parallelizable Recurrence**". In EMNLP 2018.

[12] D. Balduzzi and M. Ghifary, "**Strongly-Typed Recurrent Neural Networks**". In ICML 2016.

[13] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "**FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network**". In NeurIPS 2018.

[14] M. Andrychowicz, M. Denil, S.G. Colmenarejo, M.W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, "**Learning to Learn by Gradient Descent by Gradient Descent**". In NeurIPS 2016.

[15] D. Sussillo and L.F. Abbott, "**Generating Coherent Patterns of Activity from Chaotic Neural Networks**". In Neuron 2009.

[16] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, and J. Schmidhuber, "**LSTM: A Search Space Odyssey**". In IEEE Trans. on Neural Networks and Learning Systems 2017.

[17] J. Koutník, K. Greff, F. Gomez, and J. Schmidhuber, "**A Clockwork RNN**". In proceedings of ICML 2014.

[18] J.L. Elman, "**Finding Structure in Time**". In Cognitive Science 1990.

[19] P. Glendinning, "**Stability, Instability and Chaos: An Introduction to the Theory of Nonlinear Differential Equations**". Vol. 11. Cambridge university press, 1994.

[20] H. Sompolinsky, A. Crisanti, and H. Sommers, "**Chaos in Random Neural Networks**". In Phys. Rev. Lett. 1988.

[21] J.C. Dyre, "**Colloquium: The Glass Transition and Elastic Models of Glass-Forming Liquids**". In Rev. Mod. Phys. 2006.

[22] M. Lukoševicius, "**A Practical Guide to Applying Echo State Networks**". In 2012, Jacobs University Bremen, Germany.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "**Deep Residual Learning for Image Recognition**". In IEEE on CVPR 2016.

[24] R.K. Srivastava, K. Greff, and J. Schmidhuber, "**Training Very Deep Networks**". In NeurIPS 2015.

[25] M. Marcus, B. Santorini, and M.A. Marcinkiewicz, "**Building a Large Annotated Corpus of English: The Penn Treebank**" In Computational Linguistics 1993.

[26] W. Zaremba, I. Sutskever, and O. Vinyals, "**Recurrent Neural Network Regularization**". In CoRR 2014.

[27] I.J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V.D. Shet, "**Multi-Digit Number Recognition from Street View Imagery Using Deep Convolutional Neural Networks**". In ICLR 2014.

[28] H. Robbins and S. Monro, "**A Stochastic Approximation Method**". In the Annals of Mathematical 1951.

[29] D.P. Kingma and J. Ba, "**Adam: A Method for Stochastic Optimization**". In ICLR 2015.

[30] A. Krizhevsky, "**Learning Multiple Layers of Features from Tiny Images**". In University of Toronto Technical Report 2009.

[31] S. Ravi and H. Larochelle, "**Optimization as a Model for Few-Shot Learning**". In ICLR 2017.

[32] B. Neyshabur, and S. Bhojanapalli and D. Mcallester, and N. Srebro, "**Exploring Generalization in Deep Learning**". In NeurIPS 2017