Recurrent Neural Networks

Sagar Prakash Barad[1]

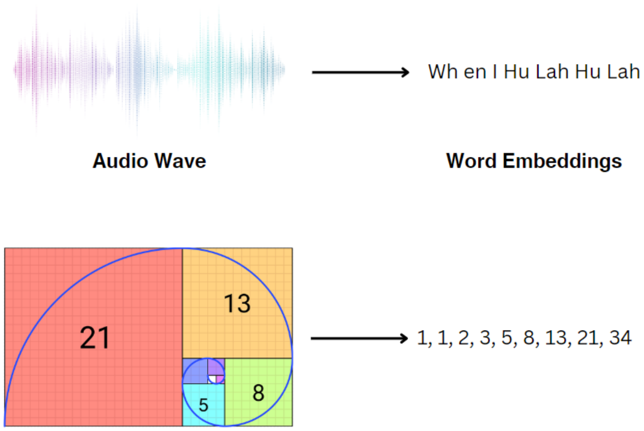
[1] *School of Physical Sciences, National Institute of Science Education and Research*

I. OUTLINE

- Motivation
- Simple RNN
- LSTM
- GRU

II. MOTIVATION

A. Modelling Sequences for getting Targets



$$f_n = f_{n-1} + f_{n-2}$$

- Convert input sequence to output sequence in another domain. For exp, *sound pressures to word embedding.*
- Given a sequence generating the next term in the sequence.

input sequence $\longrightarrow$ input sequence + next term

Memoryless models for sequences

- Auto-regressive Models
- Feed Forward Neural Network

Beyond memoryless models, we have

Recurrent Neural Network

- Distributed hidden state that allows them to store a lot of information about the past efficiently[1].
- Non-linear dynamics that allows them to update their hidden state in complicated ways.

III. SIMPLE RNN

A. Architecture

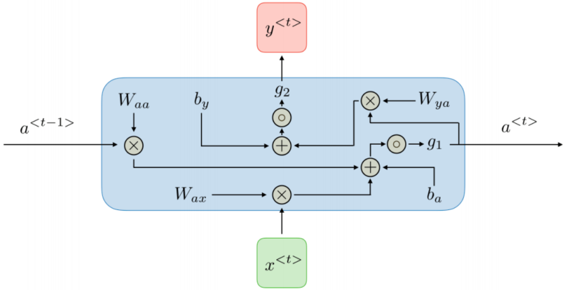


Fig 1. Architecture of a simple RNN cell[2].

Goal: Learn the mapping from the inputs $\mathbf{x}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_T)$ to outputs $\mathbf{y}_{1:T} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, ..., \mathbf{y}_T)$.

$$\boldsymbol{h}_t = \phi_h \left(W_h \boldsymbol{h}_{t-1} + W_x \boldsymbol{x}_t + \boldsymbol{b}_h \right),$$

$$\mathbf{y}_t = \phi_y \left(W_y \boldsymbol{h}_t + \boldsymbol{b}_y \right).$$

$\boldsymbol{\theta} = \{W_h, W_x, W_y, \boldsymbol{b}_h, \boldsymbol{b}_y\}$ are network parameters with ϕ_h and ϕ_y being non-linear activation functions and $\boldsymbol{h}_t$, y_t are our hidden state and output, respectively.

For $t = 1$, $\boldsymbol{h}_0 = \mathbf{0}$ and $\boldsymbol{h}_1 = \phi_h \left(W_x \boldsymbol{x}_1 + \boldsymbol{b}_h \right)$.

III. SIMPLE RNN

A. Architecture

Advantages

- Possibility of processing input of any length
- Model size not increasing with size of input
- Computation takes into account historical information
- Weights are shared across time

Disadvantages

- Computation being slow
- Difficulty of accessing information from a long time ago
- Cannot consider any future input for the current state

III. SIMPLE RNN

B. Applications of RNN

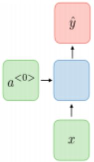
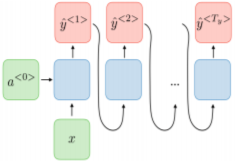
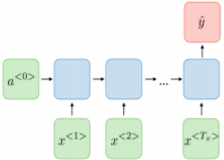
Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
One-to-many $T_x = 1, T_y > 1$		Music generation
Many-to-one $T_x > 1, T_y = 1$		Sentiment classification

Fig 2. Variation of RNNs based on task in hand[2].

III. SIMPLE RNN

B. Applications of RNN

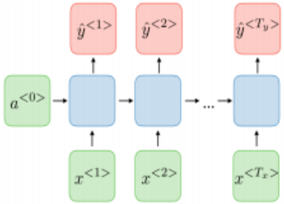
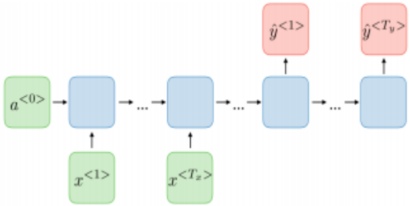
Many-to-many $T_x = T_y$		Name entity recognition
Many-to-many $T_x \neq T_y$		Machine translation

Fig 2. Variation of RNNs based on task in hand[2].

C. Loss

Loss function: The loss function $\mathcal{L}$ is defined for loss at every time step.

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{t=1}^{T} \mathcal{L}\left(\mathbf{y}_t\right)\}$$

We need to compute $\frac{d}{d\boldsymbol{\theta}}\mathcal{L}\left(\mathbf{y}_t\right)$ for $\boldsymbol{\theta} = \{W_h, W_x, W_y, \boldsymbol{b}_h, \boldsymbol{b}_y\}$.

- Derivative of $\mathcal{L}\left(\mathbf{y}_t\right)$ w.r.t. W_y and $\boldsymbol{b}_y$:

$$\frac{d\mathcal{L}\left(\mathbf{y}_t\right)}{dW_y} = \frac{d\mathcal{L}\left(\mathbf{y}_t\right)}{d\mathbf{y}_t}\frac{d\mathbf{y}_t}{dW_y}, \quad \frac{d\mathcal{L}\left(\mathbf{y}_t\right)}{d\boldsymbol{b}_y} = \frac{d\mathcal{L}\left(\mathbf{y}_t\right)}{d\mathbf{y}_t}\frac{d\mathbf{y}_t}{d\boldsymbol{b}_y}$$

C. Loss

- Derivative of $\mathcal{L}(\mathbf{y}_t)$ w.r.t. W_x and $\boldsymbol{b}_h$:

$$\frac{d\mathcal{L}(\mathbf{y}_t)}{dW_x} = \frac{d\mathcal{L}(\mathbf{y}_t)}{d\mathbf{y}_t}\frac{dy_t}{d\boldsymbol{h}_t}\frac{d\boldsymbol{h}_t}{dW_x}, \quad \frac{d\mathcal{L}(\mathbf{y}_t)}{d\boldsymbol{b}_h} = \frac{d\mathcal{L}(\mathbf{y}_t)}{d\mathbf{y}_t}\frac{dy_t}{d\boldsymbol{h}_t}\frac{d\boldsymbol{h}_t}{d\boldsymbol{b}_h}$$

  W_x and $\boldsymbol{b}_h$ also contributes to $\boldsymbol{h}_t$ through $\boldsymbol{h}_{t-1}$.

$$\frac{d\boldsymbol{h}_t}{dW_x} = \frac{\partial\boldsymbol{h}_t}{\partial W_x} + \frac{d\boldsymbol{h}_t}{d\boldsymbol{h}_{t-1}}\frac{d\boldsymbol{h}_{t-1}}{dW_x}, \quad \frac{d\boldsymbol{h}_t}{d\boldsymbol{b}_h} = \frac{\partial\boldsymbol{h}_t}{\partial\boldsymbol{b}_h} + \frac{d\boldsymbol{h}_t}{d\boldsymbol{h}_{t-1}}\frac{d\boldsymbol{h}_{t-1}}{d\boldsymbol{b}_h}$$

- Derivative of $\mathcal{L}(\mathbf{y}_t)$ w.r.t. W_h : by chain rule, we have

$$\frac{d\mathcal{L}(\mathbf{y}_t)}{dW_h} = \frac{d\mathcal{L}(\mathbf{y}_t)}{d\boldsymbol{h}_t}\frac{d\boldsymbol{h}_t}{dW_h}$$

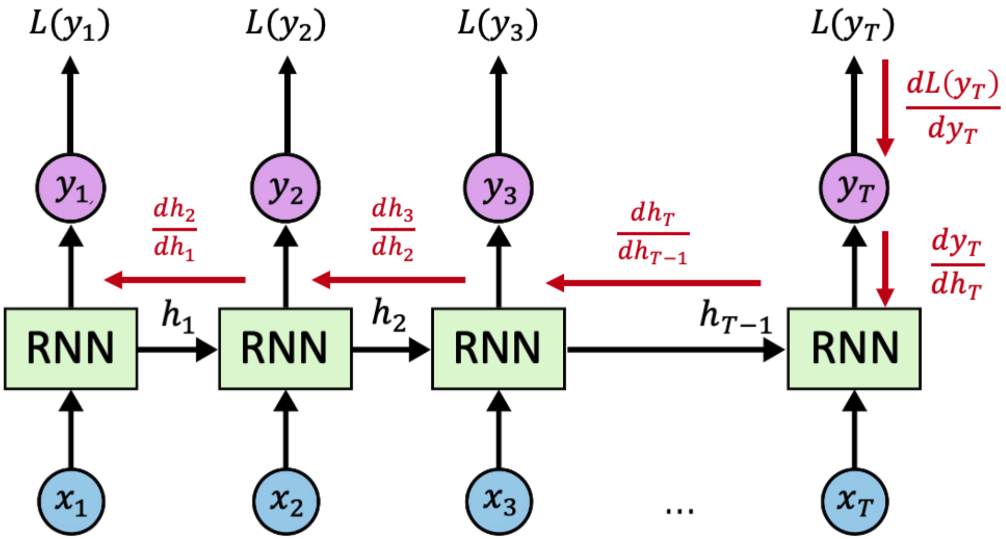


Fig 3: Visualizing Full BPTT for $\nabla_{W_h}\mathcal{L}(y_t)$ *[3].*

C. Loss

First, we find $\frac{d\boldsymbol{h}_t}{dW_h}$ as follows:

$$\boxed{\frac{d\boldsymbol{h}_t}{dW_h} = \frac{\partial \boldsymbol{h}_t}{\partial W_h} + \frac{d\boldsymbol{h}_t}{d\boldsymbol{h}_{t-1}} \frac{d\boldsymbol{h}_{t-1}}{dW_h}}$$

$$\frac{d\boldsymbol{h}_t}{dW_h} = \sum_{\tau=1}^{t} \left(\prod_{l=\tau}^{t-1} \frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l} \right) \frac{\partial \boldsymbol{h}_\tau}{\partial W_h}$$

When $\tau = t, \prod_{l=t}^{t-1} \frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l} = 1$.

truncated BPTT

$$\text{truncate}\left[\frac{d\boldsymbol{h}_t}{dW_h}\right] = \sum_{\tau=\max(1,t-L)}^{t} \left(\prod_{l=\tau}^{t-1} \frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l} \right) \frac{\partial \boldsymbol{h}_\tau}{\partial W_h}$$

III. SIMPLE RNN

D. Gradient vanishing/explosion issues

$$\frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l}^\top = \phi_h'\left(W_h\boldsymbol{h}_l + W_x\boldsymbol{x}_{l+1} + \boldsymbol{b}_h\right) \odot W_h$$

$\prod_{l=\tau}^{t-1} \frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l}$ contains products of $t-\tau$ copies of W_h and the derivative $\phi_h'(\cdot)$ at time steps $l = \tau, \ldots, t-1$.

For simplification consider, $\phi_h'(\cdot) = 1$.

Then $\prod_{l=\tau}^{t-1} \frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l} = \left(W_h^{t-\tau}\right)^\top$ will *vanish* or *explode* when $t-\tau$ is **large**, depending on whether $W_h < 1$ or not.

Since W_h is a matrix, if

$$\max\left(|\lambda_{\max}(W_h)|, |\lambda_{\min}(W_h)|\right) < 1$$

then $\prod_{l=\tau}^{t-1} \frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l} = \left(W_h^{t-\tau}\right)^\top$ will vanish or explode when $t-\tau$ increases.

Gradient clipping

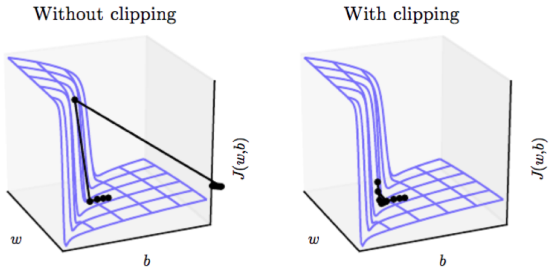


Fig 4: Visualising the gradient step with/out gradient clipping[4].

We fix a hyper-parameter γ, now a gradient $\boldsymbol{g}$ is clipped when $\|\boldsymbol{g}\| > \eta$:

$$g \leftarrow \frac{\gamma}{\|g\|} g$$

IRNN Initialization

We use **ReLU** activation for ϕ_h and initialize W_h as the *identity matrix* $(\mathbf{I})$ and $\boldsymbol{b}_h$ as *zero vectors* $(\mathbf{0})$[5].

Therefore, $\phi_h'(t) = \delta(t > 0)$ and $\frac{d\boldsymbol{h}_{l+1}}{d\boldsymbol{h}_l} = \delta(W_x \boldsymbol{x}_{l+1} > 0)$.

Orthogonal or Unitary Weight Matrix

Construct W_h as an *orthogonal* or *unitary* matrix[6].

IV. LSTM

A. Architecture

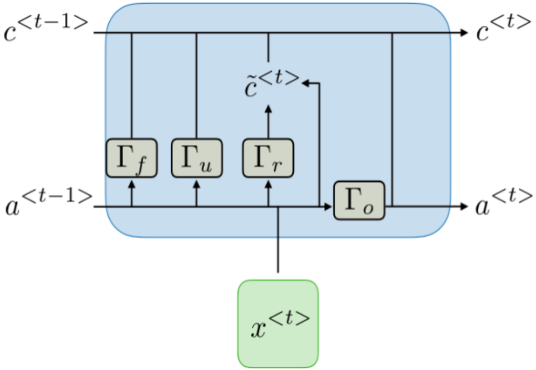


Fig 5. Architecture of a LSTM RNN cell[2].

- Proposed by [Hochreiter and Schmidhuber, 1997] to addressing the gradient vanishing/explosion problem.
- Introduced memory cell states and gates.

IV. LSTM

B. Introducing Memory cell and Gates

$$
\begin{aligned}
&\boldsymbol{f}_t : \text{ forget gate} && \boldsymbol{f}_t = \sigma\left(W_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f\right) \\
&\boldsymbol{i}_t : \text{ input gate} && \boldsymbol{i}_t = \sigma\left(W_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i\right) \\
&\boldsymbol{o}_t : \text{output gate} && \boldsymbol{o}_t = \sigma\left(W_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o\right) \\
&\boldsymbol{x}_t : \text{input} && \tilde{\boldsymbol{c}}_t = \tanh\left(W_c \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c\right) \\
&\boldsymbol{c}_t : \text{ memory cell state} && \boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
&\boldsymbol{h}_t : \text{hidden state} && \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right)
\end{aligned}
$$

The initial cell state $\boldsymbol{c}_0$ starts as zero, and both the elements in $\boldsymbol{c}t$ and $\boldsymbol{h}_t$ are constrained within the range of (-1, 1).

B. Introducing Memory cell and Gates

Table I: Summary of Gates in Gated RNNs

Gate Name	Function	Close to 1	Close to 0
Input Gate	Controls the influence of new input on cell state	Accepts and stores new input	Rejects new input
Forget Gate	Controls the retention of past information in cell state	Keeps and remembers past information	Forgets past information
Output Gate	Determines the influence of cell state on the output	Emphasizes cell state for output	Suppresses cell state for output

IV. LSTM

B. Gradient Computation

$$\frac{d\mathcal{L}\left(\mathbf{y}_t\right)}{dW_c} = \frac{d\mathcal{L}\left(\mathbf{y}_t\right)}{d\boldsymbol{h}_t}\frac{d\boldsymbol{h}_t}{dW_c}$$

$$\frac{d\boldsymbol{h}_t}{dW_c} = \boldsymbol{o}_t \odot \frac{d\tanh\left(\boldsymbol{c}_t\right)}{dW_c} + \tanh\left(\boldsymbol{c}_t\right) \odot \frac{d\boldsymbol{o}_t}{dW_c}$$

$$\frac{d\boldsymbol{o}_t}{dW_c} = \frac{d\boldsymbol{o}_t}{d\boldsymbol{h}_{t-1}}\frac{d\boldsymbol{h}_{t-1}}{dW_c}$$

$$\frac{d\boldsymbol{c}_t}{dW_c} = \boldsymbol{f}_t \odot \frac{d\boldsymbol{c}_{t-1}}{dW_c} + \boldsymbol{c}_{t-1} \odot \frac{d\boldsymbol{f}_t}{dW_c} + \boldsymbol{i}_t \odot \frac{d\tilde{\boldsymbol{c}}_t}{dW_c} + \tilde{\boldsymbol{c}}_t \odot \frac{d\boldsymbol{i}_t}{dW_c}$$

.

B. Gradient Computation

$$\frac{d\boldsymbol{o}_t}{dW_c} = \frac{d\boldsymbol{o}_t}{d\boldsymbol{h}_{t-1}} \left(\boldsymbol{o}_{t-1} \odot \frac{d \tanh\left(\boldsymbol{c}_{t-1}\right)}{dW_c} + \tanh\left(\boldsymbol{c}_{t-1}\right) \odot \frac{d\boldsymbol{o}_{t-1}}{dW_c} \right)$$

$$\frac{d\boldsymbol{c}_t}{dW_c} = \underbrace{\left(\boldsymbol{f}_t + \boldsymbol{o}_{t-1} \odot \frac{d \tanh\left(\boldsymbol{c}_{t-1}\right)}{d\boldsymbol{c}_{t-1}} \odot \frac{d\boldsymbol{c}_t}{d\boldsymbol{h}_{t-1}} \right)}_{= \frac{d\boldsymbol{c}_t}{dc_{t-1}}} \frac{d\boldsymbol{c}_{t-1}}{dW_c} + \tanh\left(\boldsymbol{c}_{t-1}\right) \odot \frac{d\boldsymbol{c}_t}{d\boldsymbol{h}_{t-1}} \frac{d\boldsymbol{o}_{t-1}}{dW_c} + \boldsymbol{i}_t \odot \frac{\partial \tilde{\boldsymbol{c}}_t}{\partial W_c}$$

$$\frac{d\boldsymbol{c}_t}{d\boldsymbol{h}_{t-1}} = \boldsymbol{c}_{t-1} \odot \frac{d\boldsymbol{f}_t}{d\boldsymbol{h}_{t-1}} + \tilde{\boldsymbol{c}}_t \odot \frac{d\boldsymbol{i}_t}{d\boldsymbol{h}_{t-1}} + \boldsymbol{i}_t \odot \frac{d\tilde{\boldsymbol{c}}_t}{d\boldsymbol{h}_{t-1}}.$$

$$\prod_{l=\tau}^{t-1} \frac{d\boldsymbol{c}_{l+1}}{d\boldsymbol{c}_l} = \prod_{l=\tau}^{t-1} \left[\boldsymbol{f}_{l+1} + \boldsymbol{o}_l \odot \frac{d\tanh(\boldsymbol{c}_l)}{d\boldsymbol{c}_l} \odot \frac{d\boldsymbol{c}_{l+1}}{d\boldsymbol{h}_l} \right]$$

The Use of Forget Gates

- On expanding the LSTM equations, we find such as $f_i \odot Q_i$ (where i ranges from $\tau + 1$ to $t - 1$).
- If the network "forgets" the previous cell state ($f_i \to 0$), it reduces the impact of these terms, helping prevent gradient explosion.

Maintaining Cell State

- $\frac{d_{c_t}}{dW_c}$ has terms related to Q_i and f_i for the range of i (from $\tau + 1$ to $t - 1$).
- When the network retains the cell state ($f_i \to 1$) for some time steps, significant terms in dW_{dctc} emerge, especially when $o_\tau \to 1$ (where o_τ is the output gate at time τ).
- This prevents gradient information at time τ from vanishing, enabling the learning of longer-term dependencies.

V. GRU

A. Architecture

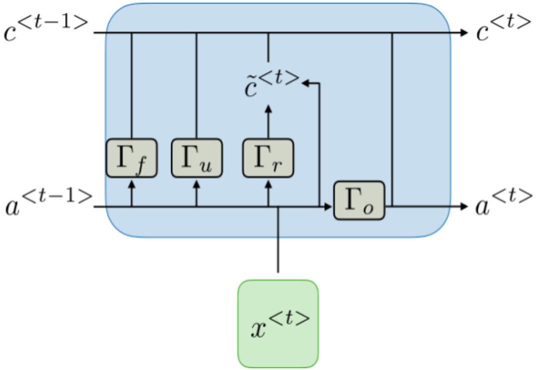


Fig 6. Architecture of a GRU RNN cell[2].

- The Gated Recurrent Unit (GRU) [Cho et al., 2014] enhances the simple RNN with gating mechanisms.
- In contrast to LSTM, GRU dispenses with input/output gates and the cell state but still preserves a form of the forgetting mechanism.

B. Gating Mechanism

$$\boldsymbol{z}_t = \sigma\left(W_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_z\right)$$

$$\boldsymbol{r}_t = \sigma\left(W_r \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r\right)$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(W_h \cdot [\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t$$

$\boldsymbol{z}_t$ serves as the update gate, controlling the inclusion of current information into the hidden states, while $\boldsymbol{r}_t$ functions as the reset gate, influencing the retention of historical information.

VI. REFERENCES

[1] G. Hinton, *Lecture 10: Recurrent neural networks*, Tech. Rep. (University of Toronto, 2013).

[2] A. Amidi and S. Amidi, "Recurrent neural networks cheatsheet," (2018).

[3] M. Team, "Recurrent neural network generation tutorial," (2016).

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).

[5] Q. V. Le, N. Jaitly, and G. E. Hinton, arXiv preprint arXiv:1504.00941 (2015).

[6] A. M. Saxe, J. L. McClelland, and S. Ganguli, In International Conference on Learning Representations (2014).