RNNS: Gated Recurrent Units, Long Short-Term Memory and Echo State Networks

MAST 680-A - Project Presentation

Timur Zhanabaev

Concordia University

zhanab.timur@gmail.com

April 14, 2023

Overview

The Elman and Jordan Networks

Jordan and Elman networks are two types of simple recurrent neural networks (RNNs) that store memory from previous step in the additional **hidden states** (also called *context units*) h_t.

The Elman Network Layer

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

The Jordan Network Layer

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

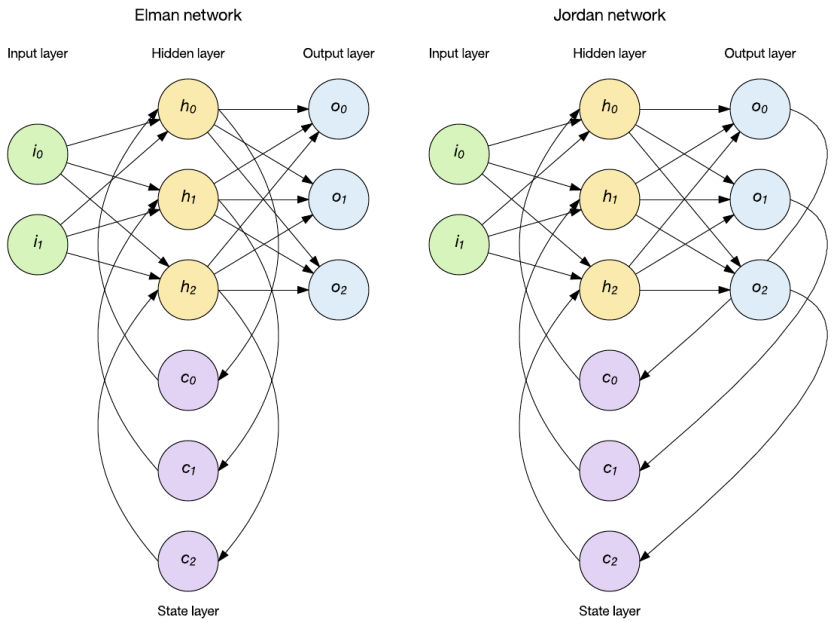

Figure 1: The Elman and Jordan network layer architectures [1]. (Here the hidden state layer with hidden state units is denoted as c_i)

The GRU Network

The RNNs tend to have the vanishing or exploding gradient problems. The GRU networks (*a drastic simplification of the LTSM layer*) were designed to alleviate the issue.

The GRU Layer: Gating Units

reset gate *(previous hidden state modulation)*:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

update gate *(new input modulation)*:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

GRU Layer: Updating States

candidate state:

$$\hat{h}_{t-1} = tanh(Wx_t + U(r_t \odot h_{t-1})$$

hidden state update:

$$h_t = (1 - z_t)h_{t-1} + z_t \hat{h}_{t-1}$$

A GRU Network Graphical Representation

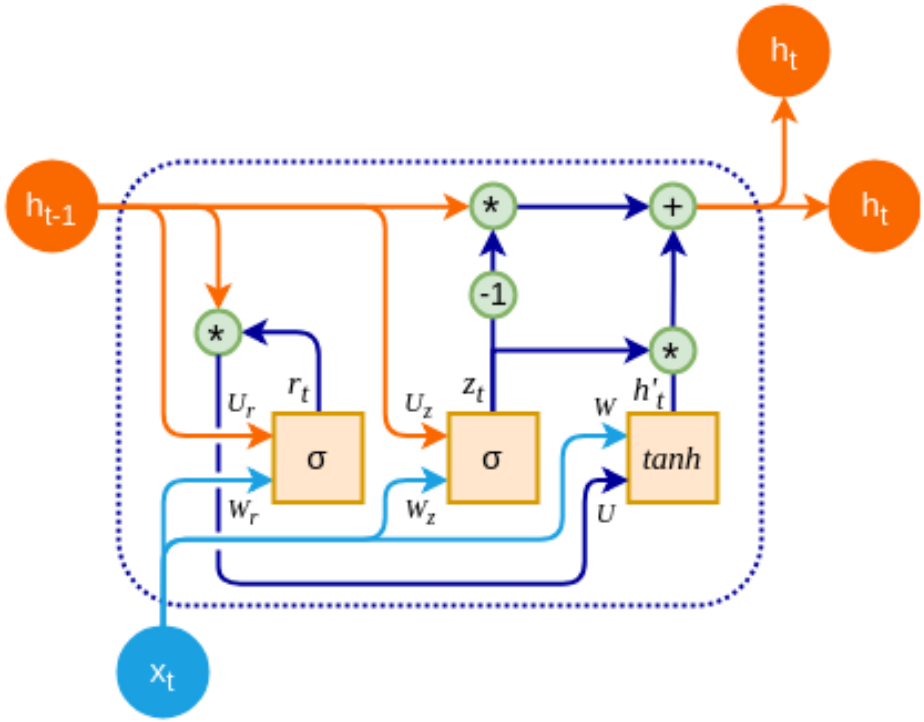

Figure 2: The Gated Recurrent Unit Layer Architecture[2]. With $[r_t]_i \approx 0$ the previous hidden states are forgotten. Hence input dominates in $\hat{h}_{t-1}$. As well with $[z_t]_i \approx 0$ we use the past hidden state to generate new hidden state, allowing the network to adaptively what to remember and forget.

The LSTM Network

Invented before the GRU Networks, the LTSM Networks were designed to mitigate the gradient problems, by controlling the error flow through the model [4]. The LTSM has a more convoluted architecture with multiple gates, input, forget and output gates. Additionally, memory states; cell states and hidden states.

The LTSM Layer Gates

With the input being again the N hidden layer units, then there is N gate units for each gate to match the dimensions.

forget gate: $f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$
input gate: $i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$
output gate: $o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$
cell input: $\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$
cell state: $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$
hidden state: $h_t = o_t \odot \sigma_h(c_t)$

Similarly to reset gate, the forget gates modulate the hidden state updates. While the input/output gates protect the memory contents. [5].

A LSTM Network Graphical Representation

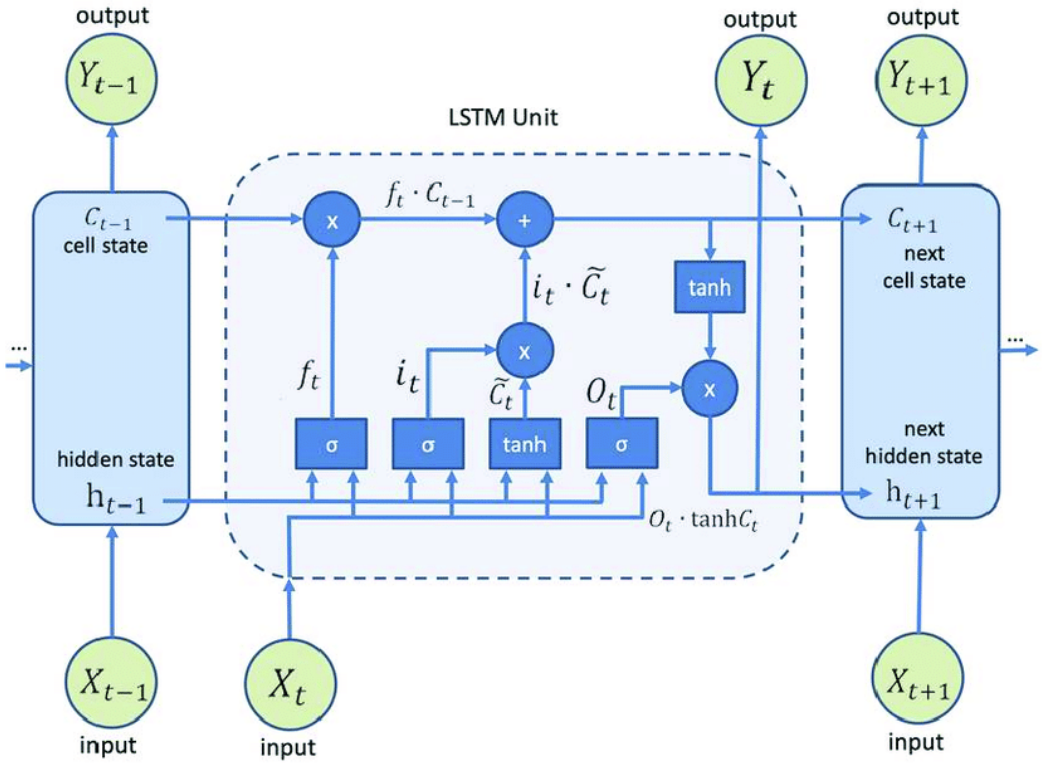

Figure 2: The Long-Short Term Memory Layer Architecture [3].

The ESN Network

The Echo State Networks follow the same idea of feeding back the output as input (RNN). Also part of Reservoir Computing paradigm, where the hidden layer is a very large non-trainable sparsely connected recurrent layer, that satisfying the echo-state property (input forgetting, state forgetting).

The ESN Layer

reservoir updates, where $W^{\text{in}} \in \mathbb{R}^{(N\times K)}$, $W \in \mathbb{R}^{(N\times N)}$, $W^{\text{back}} \in \mathbb{R}^{(N\times L)}$,

$$h_t = f(W^{\text{in}} x_t + W^{\text{res}} h_{t-1} + W^{\text{back}} y_{t-1})$$

output, where $W^{\text{out}} \in \mathbb{R}^{(L\times(K+N+L))}$,

$$y_t = f^{\text{out}}(W^{\text{out}} \langle h_t, x_t, y_{t-1} \rangle)$$

The most practical way to achieve the echo-state property is to scale the weight matrices s.t. $\sigma^{\max}(W) < 1$.

Hence, in practice we generate a sparse matrix where weights are {0.4,-0.4,0} with probabilities {0.025,0.025, 0.95} (connectivity of 5%).

An ESN Graphical Representation

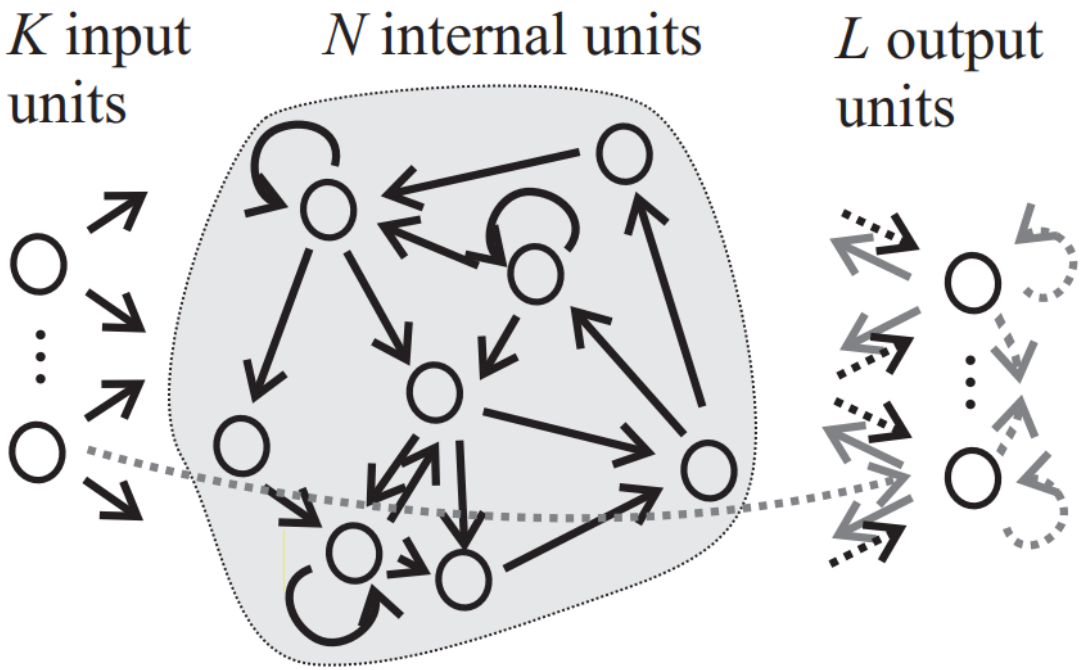

Figure 2: The Echo State Network Layer Architecture [4]. The grey arrows are optional forms of connecting input/output and output/hidden states.

The ESN Network - Leaky Integrator Neurons

For continuous time data it is recommended to use the **Leaky Integrator Neurons**.

Leaky Integrator Neurons Update

The hidden state unit then evolves according to the dynamics, as a difference equation with step size δ [5],

$$h_{t+1} = (1 - \delta C\gamma)h_t + \delta C \left\{ f(W^{\text{in}} x_{t+1} + W^{\text{res}} h_t + W^{\text{back}} y_t) \right\}$$

With more hyper-parameters to tune, C is a *time constant* and γ is the *leaking decay rate*. The condition for existence of echo-states requires the value to be $|1 - \delta C(\gamma - \sigma^{\max}(W))| < 1$.

The problem is solved as a linear regression problem, effectively circumventing the vanishing or exploding gradients problems, while still maintaining good performance for non-linear dynamic systems (if the conditions for echo-state property are maintained).

The Rössler Attractor System of ODEs

$$\dot{x} = -y - z$$
$$\dot{y} = -x + ay$$
$$\dot{z} = b + z(x - c)$$

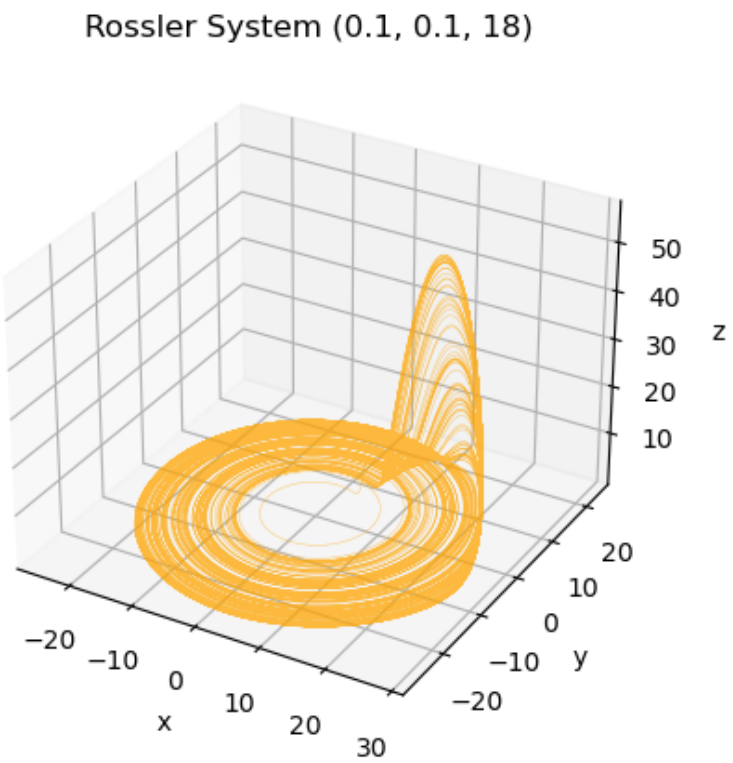

Figure 2: Chaotic Attractor Rössler System; parameters a = 0.1, b=0.1, c=18.

TensorFlow 2 Implementation

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
import tensorflow_addons as tfa

model = keras.Sequential()
```

The ESN Layer

```
model.add(tfa.layers.ESN(args))
```

The GRU Layer

```
model.add(layers.GRU(args))
```

The LSTM Layer

```
model.add(layers.LSTM(args))
```

Some arguments for ESN:
`units`, `connectivity` = [0,1], `leaky` = [0,1], `spectral_radius` = [0,1], `activation`='tanh'
Some arguments for GRU and LTSM:
`units`, `activation`='tanh', `recurrent_activation`='sigmoid'

Citations

Citations:

1. https://developer.ibm.com/articles/cc-cognitive-recurrent-neural-networks/
2. https://www.oreilly.com/library/view/python-deep-learning/9781789348460/88bbd930-3e25-47ab-9a50-b7d8c324994f.xhtml
3. Zhou, Dingyi Zuo, Xiaoqing Zhao, Zhifang. (2022). Constructing a Large-Scale Urban Land Subsidence Prediction Method Based on Neural Network Algorithm from the Perspective of Multiple Factors. Remote Sensing. 14. 1803. 10.3390/rs14081803.
4. Herbert Jaeger, The “echo state” approach to analysing and training recurrent neural networks, Fraunhofer Institute for Autonomous Intelligent Systems, January 26 2010.
5. Hochreiter, Sepp Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.