DEEP LEARNING LAB(EEE4423-01) Week 10 - Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network

Hyeon Si Eun[1], 2019142214
[1] Department of Electric and Electronic Engineering, Yonsei University

1 Introduction

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have revolutionized the field of artificial intelligence, particularly in tasks involving sequential data. This report delves into the inner workings and applications of RNNs and LSTMs, highlighting their distinct advantages over traditional feedforward networks. Furthermore, we explore the challenges and limitations of these networks, while emphasizing current research efforts to enhance their performance and adaptability. Recurrent Neural Networks (RNNs) are a type of neural network architecture that are capable of modeling sequences and time series data. Long Short-Term Memory (LSTM) is a popular variant of RNNs that address the vanishing gradient problem, which can occur when training traditional RNNs on long sequences. In this report, we will discuss the architecture of RNNs and LSTMs, their applications in various fields.

2 RNN

A Recurrent Neural Network (RNN) is a type of artificial neural network designed to process sequences of data by maintaining an internal state that can remember information from previous time steps. RNNs are particularly well-suited for tasks that involve sequential inputs or outputs, such as natural language processing, time series prediction, and speech recognition.

2.1 Structure:

An RNN is composed of interconnected neurons, organized in layers. It has three main layers: input, hidden, and output. The input layer receives the input data, the hidden layer processes the data, and the output layer generates the final output. The hidden layer is where the recurrence happens, as it connects not only to the next layer but also to itself, enabling the network to maintain information from previous time steps.

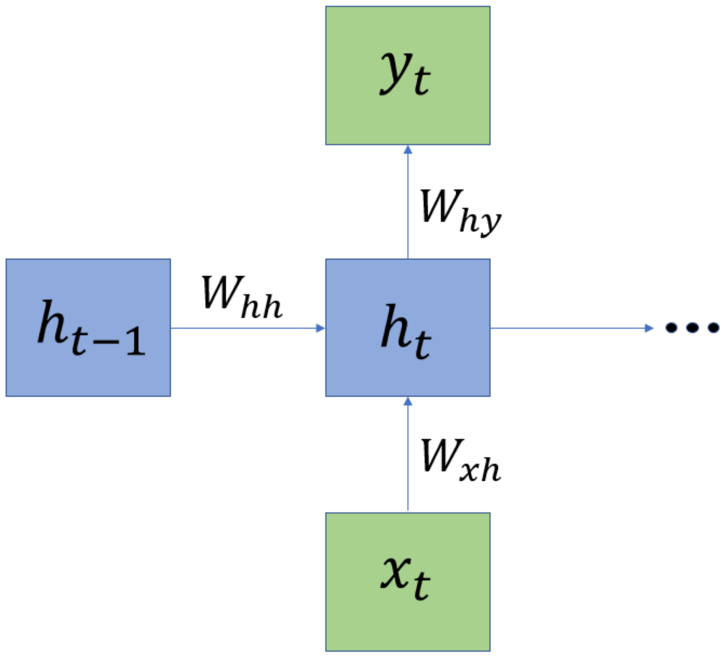

Figure 1: RNN cell Structure

This is an extended abstract. The full paper is available at the this paper

2.2 Basic components:

- Input vector (x_t): Represents the input data at a given time step.
- Hidden state vector (h_t): Represents the internal state of the network at a given time step. It is calculated using the input vector and the previous hidden state.
- Output vector (y_t): Represents the output of the network at a given time step, which is generated using the current hidden state.
- Weights (W_{xh}, W_{hh}, W_{hy}): These are learnable parameters that determine the connection strengths between neurons. W_{xh} connects input to the hidden layer, W_{hh} connects the hidden layer to itself, and W_{hy} connects the hidden layer to the output layer.
- Bias (b_h, b_y) : These are learnable biases of each layer.
Each t means the time step of the RNN.

2.3 Forward pass:

During the forward pass, the network processes input data sequentially. For each time step t, the input vector is fed into the network, and the following calculations are performed:
- Hidden state (h_t) = $f_h(W_{xh} * x_t + W_{hh} * h_{t-1} + b_h)$
- Output (y_t) = $f_y(W_{hy} * h_t + b_y)$
Here, f_h is usually a non-linear function like tanh, sigmoid, or ReLU, and f_y depends on the task (e.g., softmax for classification). And b_h and b_y are biases.
When the dimension of the input is d and the size of the hidden state is D_h, the size of each vector and matrix is as follows.

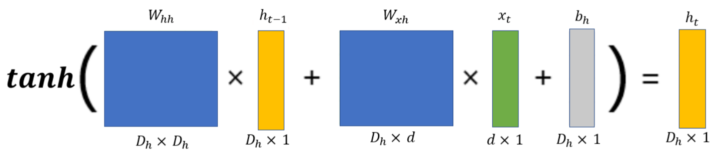

Figure 2: A pictorial representation of hidden layer operations $x_t : (d \times 1)$, $W_{xh} : (D_h \times d)$, $W_{hh} : (D_h \times D_h)$, $h_{t-1} : (D_h \times 1)$, $b_h : (D_h \times 1)$

2.4 Backpropagation through time (BPTT):

To train an RNN, the error between the predicted output and the actual output is minimized using a technique called backpropagation through time (BPTT). BPTT is an extension of the standard backpropagation algorithm, which accounts for the sequential nature of RNNs. It involves unfolding the network through time, computing gradients for each time step, and then updating the weights.

2.5 Challenges and solutions:

- Vanishing/exploding gradient problem: RNNs can suffer from vanishing or exploding gradients during training, making it difficult to learn long-range dependencies. Solutions like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks have been developed to address this issue by introducing gating mechanisms that allow for more effective gradient propagation.
- Computational complexity: Training RNNs can be computationally expensive, especially for long sequences. Techniques like truncated backpropagation through time and parallelization can help alleviate this issue.
In summary, RNNs are powerful models for dealing with sequential data, but they have some limitations. Modern architectures like LSTMs, GRUs,

and the more recent Transformer models have been developed to address these challenges and improve performance on various tasks.

3 LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to overcome the vanishing and exploding gradient problems encountered when training traditional RNNs. These issues make it difficult for RNNs to learn long-range dependencies in sequences. LSTMs, introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997, have been widely used in various sequence-to-sequence tasks such as machine translation, speech recognition, and natural language processing.

3.1 Structure:

LSTMs maintain an internal state (called cell state) that allows them to store and access information over long sequences. They consist of a set of interconnected memory cells, where each memory cell has three main components: input gate, forget gate, and output gate. These gates control the flow of information in and out of the cell, enabling the LSTM to learn what information to store, update, or output at each time step.

3.2 Gates and memory cell:

- Input gate (i): Determines the extent to which new input should be added to the cell state. It is controlled by a sigmoid activation function, which outputs values between 0 and 1.
- Forget gate (f): Determines how much of the previous cell state should be retained. It also uses a sigmoid activation function to output values between 0 and 1.
- Output gate (o): Controls how much of the current cell state should be exposed as the output (hidden state) of the LSTM cell. Like the other gates, it uses a sigmoid activation function.
- Cell state (c): Represents the internal memory of the LSTM cell. It is updated by combining the input gate's output with the previous cell state's output, as modulated by the forget gate.

3.3 Forward pass:

At each time step t, the LSTM cell performs the following operations:
- Compute input, forget, and output gate activations:

$$i_t = \text{sigmoid}(W_i * [h_{t-1}, x_t] + b_i)$$
$$f_t = \text{sigmoid}(W_f * [h_{t-1}, x_t] + b_f)$$
$$o_t = \text{sigmoid}(W_o * [h_{t-1}, x_t] + b_o)$$

- Compute the candidate cell state update:

$$g_t = \tanh(W_g * [h_{t-1}, x_t] + b_g).$$

- Update the cell state:

$$c_t = f_t * c_{t-1} + i_t * g_t.$$

- Compute the hidden state (output):

$$h_t = o_t * \tanh(c_t).$$

Here, W and b are the learnable weight matrices and bias vectors, respectively, for each gate.

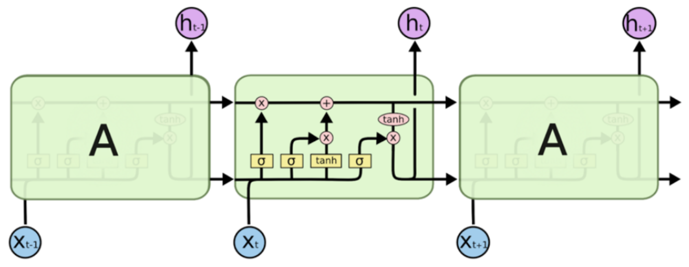

Figure 3: LSTM cell Structure[1]

3.4 Training:

LSTMs are trained using backpropagation through time (BPTT), similar to traditional RNNs. The error gradients with respect to the weights and biases are computed by unfolding the network over time and applying the chain rule. The weights and biases are then updated using an optimization algorithm like gradient descent or a variant thereof (e.g., RMSprop, Adam).

3.5 Variants and improvements:

Several LSTM variants and related architectures have been proposed to improve their performance or reduce their complexity. Some of these include:
- Gated Recurrent Unit (GRU): A simpler architecture that combines the input and forget gates into a single update gate and merges the cell state with the hidden state.
- Peephole connections: Allow the gates to access the cell state directly, providing additional information for the gating mechanism.
- Bidirectional LSTM (BiLSTM): Processes sequences in both forward and backward directions, capturing information from the past and future simultaneously.

In summary, LSTMs are a powerful RNN architecture that can learn and represent long-range dependencies in sequences. They have been successfully applied to a wide range of tasks, such as natural language processing, speech recognition, and time series prediction. While LSTMs have addressed some of the limitations of traditional RNNs, they still have their own challenges, such as computational complexity and the need for a large amount of training data. More recent architectures, like the Transformer, have been developed to further improve performance and address some of the shortcomings of LSTMs. Nonetheless, LSTMs remain an essential building block in the field of deep learning for sequential data.

4 the fields where RNN and LSTM are used

RNNs and LSTMs have been widely adopted across various domains due to their ability to handle sequential data effectively. In natural language processing, they are employed for tasks such as sentiment analysis, machine translation, and text summarization. In the realm of speech recognition, these networks are utilized to transcribe spoken language into written text. RNNs and LSTMs have also found applications in finance, where they are used for time series prediction, such as forecasting stock prices. Additionally, they play a pivotal role in areas like video analysis and music generation, where they analyze and generate sequences of frames or notes, respectively.

5 More Study

The goal of this paper[2] is to explain the essential RNN and LSTM fundamentals in a single document. If you want to know more detailed mathematical theory of RNN and LSTM, see this paper.

[1] David Liu and An Wei. Regulated lstm artificial neural networks for option risks. *FinTech*, 1(2):180–190, 2022.

[2] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.