Practical Handwriting Development with RNN and LSTM

Authors :Dr.venkatatesan Selvaraj[1] ;venkatatesansel34@gmail.com

Dr.surya Mukhi[2] ;suryamukhi.off@gmail.com

HOD.Mr.Syed mujeeb ul Hasan[3] ;syedmujeebul@gmail.com

Students :Syed Abdul Rub[1] ;rab.omer6@gmail.com

Syed Habeeb ahmed[2] ;syedsaleem1237@gmail.com

Mohammed Zubair[3] Ali;zubairali401@gmail.com

ABSTRACT:

More importantly in today's technology setting, this method reduces the need for human interaction. How can we use the information at hand to make sense of occurrences that at first glance seem unrelated? The ultimate goal of this endeavor is to reconstruct a person's handwriting from scratch. As a key measure in cryptography, handwriting is also an important indication of human characteristics like character. In this research, we demonstrate that Long Short-Term Memory recurrent neural networks may be used to predict data points at the individual level to create complex sequences with extended structure. The method's usefulness is shown with examples drawn from both discrete (text) and continuous (online handwriting) data sets. The network may be used for handwriting synthesis once it has been taught to make predictions based on a text sequence. Several types of realistic cursive handwriting may be generated using the resulting technology.

Keywords: - Hand-writing, Recurrent Neural Networks, Long Short-term Memory, Predictions, Text Sequence, Hand-writing Synthesis.

1. INTRODUCTION:

Personality factors such as handwriting have long been used as important cryptographic metrics. This method further reduces the need for human involvement, which is more valuable in today's technology setting. We offer a technique that use RNNs to generate novel Handwriting fonts, which may be used for the aforementioned application. Recurrent neural networks (RNNs) are a rich family of dynamic models that have been successfully used to sequence generation across a wide variety of domains, including music, language, and motion capture data. It is possible to train RNNs to produce new sequences by analyzing and forecasting current ones. Similarly to other types of neural networks, RNNs make use of an internal representation to perform a high dimensional interpolation between training data to arrive at predictions, which is what gives them their 'fuzzy' reputation. An RNN of sufficient size may, in theory, construct infinitely complicated sequences. Regular RNNs, however, can only remember so much about previous inputs. This 'amnesia' renders them less capable of mimicking long-range structure and more prone to instability when creating sequences. A larger memory allows the network to make predictions based on previous data in the case that it is unable to understand its most recent input, hence maintaining stability. When working with real-valued data, the instability issue becomes more severe since predictions might rapidly deviate from the manifold on which the training data occurs. The most obvious way to improve performance is to strengthen memory. The Long Short-Term Memory (LSTM) architecture of RNNs is optimized to outperform regular RNNs in both storage and retrieval tasks. It's possible to use the resultant technology to produce many distinct styles of realistic cursive handwriting.

2. PROPOSED SYSTEM:

From an early age on, people honed the ability to visually communicate their ideas via the use of letters to form meaningful words and phrases. The more one writes, the more one refines this talent and finds his or her unique voice as a writer. The suggested approach allows us to build a reliable platform for the generation of personable,

individualized handwritten fonts. The forensics and psychology fields, where a person's handwriting may be analyzed to provide insights about their personality, might also benefit from using this method. Because of the low dimensionality and high degree of visual accessibility offered by online handwriting models, we decided to use them to train our model for this project. IAM-OnDB, the IAM's online handwriting database, is the backbone of the system. The 221 writers that contributed handwritten lines to IAM-OnDB utilized the smart whiteboard to compile the database. Using an infrared device situated in the corner of the board, each line was processed individually (line-to-line relationships were discarded) so that we could achieve our aims. We made use of the validation sets to their full potential by dividing them in half, with the larger half being used for real training and the smaller half for early stopping. The primary focus was on producing authentic-looking handwriting. Using recurrent neural networks is a no-brainer given that we are dealing with time data. However, extended dependencies might be difficult to store in a conventional recurrent cell. For this task, RNN cells are replaced with Long Short-Term Memory cells. They employ several kinds of internal gates to remember things for longer and keep their cell states for longer. Although LSTM cells allow for long-term relationships to be formed, it is possible that a single LSTM cell is insufficient for abstracting the nuances of a given handwriting stroke. We use a three-layer recurrent LSTM network here. With this new framework, we can generate any handwritten writing in a completely arbitrary script.

3. CONCEPT:

Recurrent Neural Networks (RNNs)

- By establishing a connection to the hidden layer, the RNN essentially enables multi-layer vision.
- The A Perception method is used to train binary classifiers in a supervised fashion. This method facilitates learning by neurons by sequentially processing the training set's constituents.

An activation function adds weights to the inputs and directs them through the neural network's hidden layer before the output is generated. In short, the hidden layers introduce a nonlinear shift to the network's inputs. • It is clear from the architecture that information is routed not only to the lowest hidden layer but to the whole hidden layer simultaneously.

The LSTM (or "long short-term memory")

- These multiplicative gates are important to many of LSTM's operations, and the three-gate units function in a manner similar to read, write, and reset, but with more flexibility.
- The network's memory resides in its "cells," or individual nodes.
- In contrast to RNN, LSTM can hold information for much longer, and reading requires the output gate to be open.
- The information in the cell will still be there if the forgotten gate is open.

Text Prediction

Because of the fleeting nature of text, 'one-hot.' input vectors are often used to feed neural networks. When class k is input at time t, then xt is a vector of length K where all entries are zeroes except for the kth element. This is because each text class has a unique set of zeros for each time point. It is possible to parameterize the Pr (xt+1jyt) multinomial distribution at the output layer using a Soft ax function.

$$\Pr(x_{t+1} = k|y_t) = y_t^k = \frac{\exp\left(\tilde{y}_t^k\right)}{\sum_{k'=1}^{K} \exp\left(\tilde{y}_t^{k'}\right)}$$

Substituting into Eq. (6) we see that

$$\mathcal{L}(\mathbf{x}) = -\sum_{t=1}^{T} \log y_t^{x_{t+1}}$$

$$\Longrightarrow \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \tilde{y}_t^k} = y_t^k - \delta_{k,x_{t+1}}$$

The only open question is which class system to implement. Word-level predictions are the norm in text prediction, also known as language modeling. This implies that K is the number of English words. When used in real life, where the number of words might potentially exceed 100,000 (due to various conjugations, proper names, etc.), this becomes a formidable obstacle. Having so many classes also necessitates a large number of model parameters in order to adequately cover the possible contexts for the words.

Handwriting Prediction

We applied the prediction network to online handwriting data (online meaning the writing is recorded as a series of pen-tip locations, as opposed to offline handwriting where only the page photos are available) to see whether it could be used to generate credible real-valued sequences. Since online handwriting only requires two real numbers per data point, it may be a helpful tool for generating sequences. The 221 writers that contributed handwritten lines to IAM-OnDB utilized the smart whiteboard to compile the database. An infrared device situated in the corner of the board tracked the writers' hand motions while they filled out forms from the Lancaster Oslo-Bergen literature corpus. Diamond is organized as a training set, two validation sets, and a test set, and it consists of handwritten lines from several inputs.

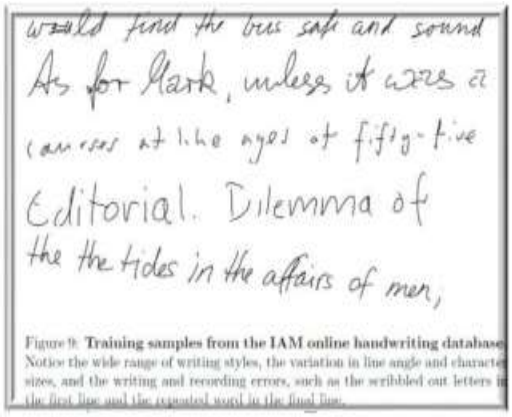


Figure 9: **Training samples from the IAM online handwriting database.** Notice the wide range of writing styles, the variation in line angle and character sizes, and the writing and recording errors, such as the scribbled out letters in the first line and the repeated word in the final line.

Handwriting Synthesis

The process of generating synthetic handwriting from an existing text is known as "handwriting synthesis." We have shown that the prediction networks we have discussed so far cannot accomplish this goal since there is no mechanism to influence the network's output letters. Here, we propose a method for enhancing handwriting synthesis in which a prediction network constructs data sequences based on a high-level annotation sequence (a string of characters). The resulting sequences look so much like handwriting that it's almost hard to tell them apart. This realism is reached without sacrificing the established artistic diversity, too. One of the biggest challenges of text-based prediction is that it is impossible to determine how well the text and the pen trace will match until the data is produced. When compared to the real text, the pen trace is often 25 times longer. The reason behind this is that pen size, writing speed, and cursive style all affect the total number of coordinates required to form a single letter, making a universal solution impossible. The RNN transducer is a kind of neural network model that can make sequential predictions using two sequences of varying length and uncertain alignment. However, early experiments with RNN transducers for handwriting synthesis had disappointing results. One possible explanation is that the transducer feeds the data into two separate RNNs for analysis before combining the results to make a decision, although it is preferable to feed everything into a single network in the vast majority of cases. In this study, we provide a novel model in which a ‘soft window' is convolved with the text string and sent into the prediction network as an additional input.

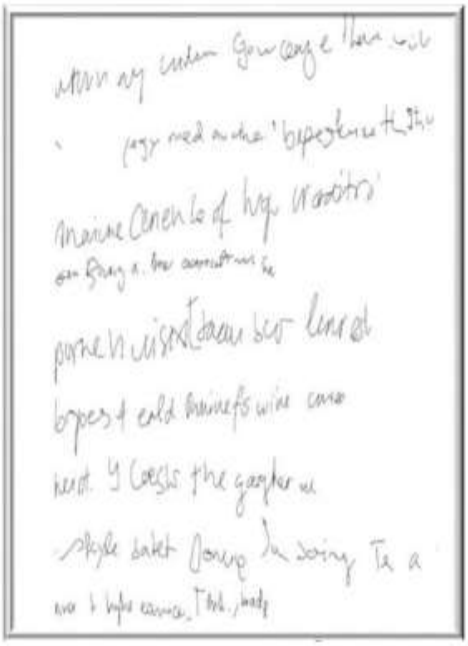


Figure: Examples of the prediction network's online handwriting samples.

Every sample is 700 it in length. An alignment between the text and the pen locations is dynamically established at the same time that the predictions are made. It learns, in a nutshell, which character to use in the next line of text.

4. ALGORITHMS:

The system's development relied on two distinct algorithms, each of which performed a necessary but distinct function. Used algorithms are:

1. Recurrent Neural Networks (RNNs)
2. Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) Definition

To produce sequences in fields as varied as music, language, and motion capture data, researchers have turned to a dynamic model class called recurrent neural networks (RNNs). RNNs may be taught to generate sequences by iteratively analyzing and making predictions about real-world data sequences.

Functionality

Real-world data sequences may be used to train RNNs to produce new sequences via iterative analysis and prediction. Since probabilistic predictions can be made from a trained network, new sequences may be generated by repeatedly sampling from the distribution of the network's output and feeding that sample back into the network. That is, by instructing the system to treat its output as seriously as it would anything seen in a dream. Although the network is inherently predictable, the sampling process introduces uncertainty and produces a dispersed set of sequences. Since the network's internal state is dependent on the previous inputs, we argue that the predictive distribution is conditional. Similarly to other types of neural networks, RNNs make use of an internal representation to perform a high dimensional interpolation between training data to arrive at predictions, which is what gives them their 'fuzzy' reputation. Predictive distributions are not based on counting precise matches between the recent past and the training set, as is the case with n-gram models and compression methods like Prediction by Partial Matching. RNNs may synthesis and reproduce the training data in an advanced way, seldom repeating the same

thing again, in contrast to template-based algorithms. This is quite obvious from the statistics provided. Due to their robustness against the curse of dimensionality, fuzzy predictions are preferred over precise matches when modeling real-valued or multivariate data. An RNN of sufficient size may, in theory, construct infinitely complicated sequences. Regular RNNs, however, can only remember so much about previous inputs. Having a larger memory helps maintain stability because it allows the network to generate predictions based on earlier data even if it is unable to interpret its most recent data. Instability is a major issue when working with real-valued data since predictions may easily wander off the manifold on which the training data is located.

Architecture

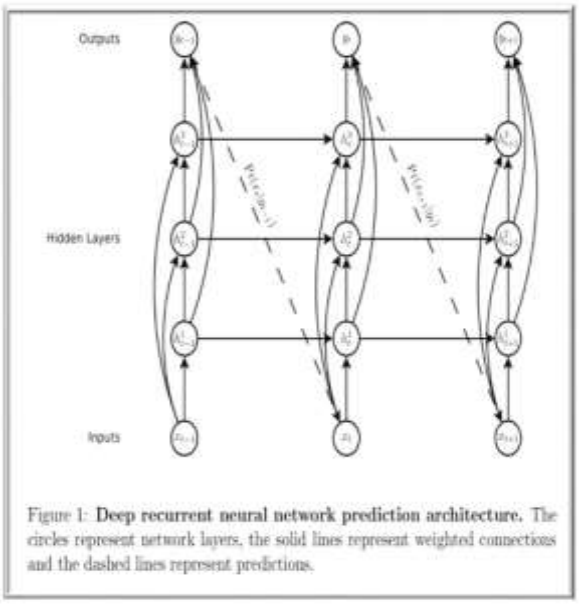


Figure 1: **Deep recurrent neural network prediction architecture.** The circles represent network layers, the solid lines represent weighted connections and the dashed lines represent predictions.

Figure 4.1.3: RNN Architecture

Architecture Explanation:

Perception is an approach for supervised learning of binary classifiers, with the RNN essentially enabling a multi-layer perception and linking with the hidden layer. This method facilitates neural learning by sequentially processing training set pieces.

A neural network technique uses a hidden layer between its input and output stages, in which a function gives weights to the inputs and passes them via activation function to generate an output. The design makes it very clear that data is routed to all of the hidden layers simultaneously, not just the one at the bottom.

PREDICTION NETWORK

The recurrent neural network that forms the backbone of the article's prediction architecture is seen in Fig 4.1.2. From the input vector sequence x = (x1, xT), the hidden vector sequences HN = (h1 n, hT n) are computed through weighted connections in a stack of N recurrently linked hidden layers to provide the output vector sequence y = (y1,

YT). Use the set of all possible xt+1 as the next input and the set of all possible YT as the outcomes to parameterize a predictive distribution Pr (xt+1|yt). Given that x1 is a blank vector at all times, the network must guess at what x2 will be. The network is 'deep' in space and time because each byte of information traveling either vertically or horizontally across the computational graph is subject to multiple weight matrices and nonlinearities. The 'skip connections' are the links between the visible and hidden layers, and the inputs and outputs, respectively. These reduce the amount of intermediate steps between the bottom and the top of a network, making it easier to train deep networks and avoiding the 'vanishing gradient' issue [1]. When N is 1, the next step may be predicted using a standard one-layer RNN. The activations of the hidden layer may be obtained by iterating the following equations

$$h_t^1 = \mathcal{H}\left(W_{ih^1} x_t + W_{h^1h^1} h_{t-1}^1 + b_h^1\right) \quad (1)$$

$$h_t^n = \mathcal{H}\left(W_{ih^n} x_t + W_{h^{n-1}h^n} h_t^{n-1} + W_{h^nh^n} h_{t-1}^n + b_h^n\right) \quad (2)$$

between t = 1 and T, and n = 2 and N.

The function of the hidden layer is denoted by H, biases by b, with the output bias denoted by b, and weight matrices by W, where n denotes the number of inputs to the nth hidden layer, Wh1h1 denotes the recurrent connection at the first hidden layer, and so on. The projected outcome, based on the deduced secret sequences, is as follows:

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n \quad (3)$$

$$y_t = \mathcal{Y}(\hat{y}_t) \quad (4)$$

Given that the output of the last layer is Y. Accordingly, the network as a whole gives a function from input histories x1: t to output vectors yt, with parameters given by the weight matrices. Using the results yt, we may parameterize the predictive distribution Pr(xt+1jyt) for the next input. The given information necessitates a specific expression of Pr(xt+1jyt). In particular, it may be challenging to do density modeling on high-dimensional, real-valued data. Density modeling is the process of generating a predictive distribution. Prediction made by the network for the xth item in the input sequence

$$\Pr(\mathrm{x}) = \prod_{t=1}^{T} \Pr(x_{t+1}|y_t) \quad (5)$$

In addition, the sequence loss L(x) is computed as the negative logarithm of Pr(x) during network training.

$$\mathcal{L}(\mathrm{x}) = -\sum_{t=1}^{T} \log \Pr(x_{t+1}|y_t) \quad (6)$$

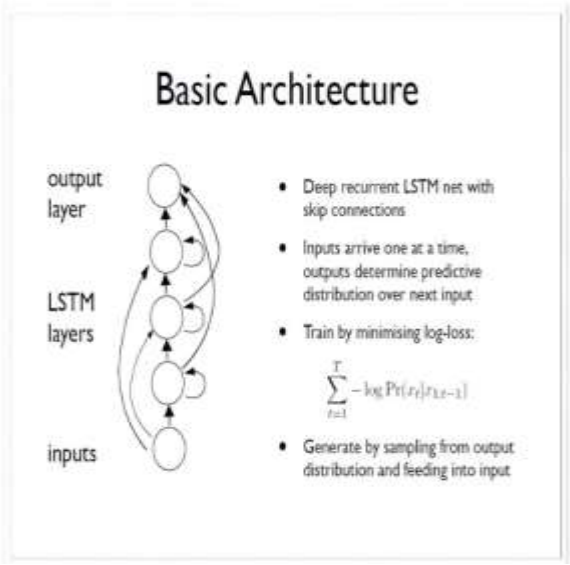
In order to employ gradient descent for training the network, it may be possible to rapidly compute the partial derivatives of the loss with respect to the network weights by applying back propagation over time to the computation graph shown in Fig.4.1.2.

LONG SHORT-TERM MEMORY

Definition

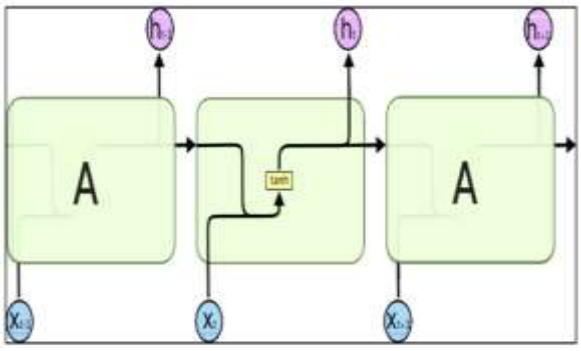
Artificial recurrent neural network (RNN) architectures like long short-term memory (LSTM) are used in deep learning. The LSTM network incorporates feedback connections, which are absent in traditional feed forward neural networks. It's capable of handling not only individual pieces of data (like photos), but also lengthy sequences (like

audio or video). To provide only a few examples, LSTM may be used for unrestricted, connected handwriting recognition, voice recognition, and network traffic anomaly/intrusion detection system (IDS) anomaly detection. In order to prevent the dreaded "vanishing gradient" problem from occurring during training of conventional RNNs, long short-term memory (LSTMs) were developed. LSTM is preferable to RNNs, hidden Markov models, and other sequence learning methods because it is less affected by the length of the gaps between data points.



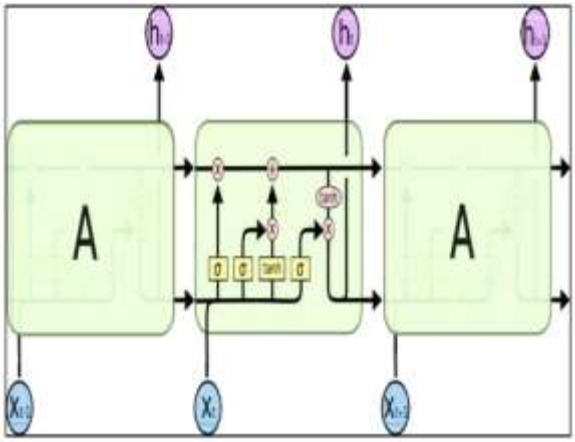
FUNCTIONALITY

The LSTM architecture is a refined form of the RNN. Data is read, written, and cleared using multiplicative gate units wrapped around linear memory cells. Input gate, output gate, forget gate, and cell are the four fundamental parts of a typical LSTM unit. The cell has the potential to retain values indefinitely, and its three gates regulate the flow of information into, within, and out of the storage area. The goal of LSTMs (long short-term memory) is to get around this dependency problem. They don't need to make any special efforts to retain information for long periods of time; it comes more or less naturally to them. All RNNs have a common architecture consisting of interconnected neural network nodes that are trained in a loop. In most RNNs, this recurrent module will be quite elementary, consisting of only a single tanh layer, for example.

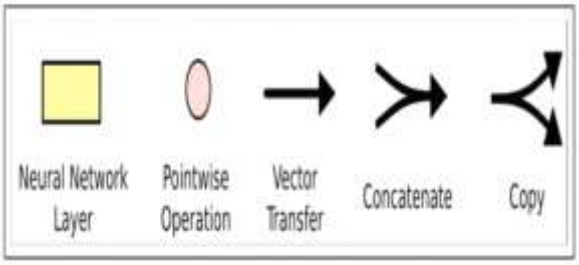


A typical RNN has a single layer in its repeating module.

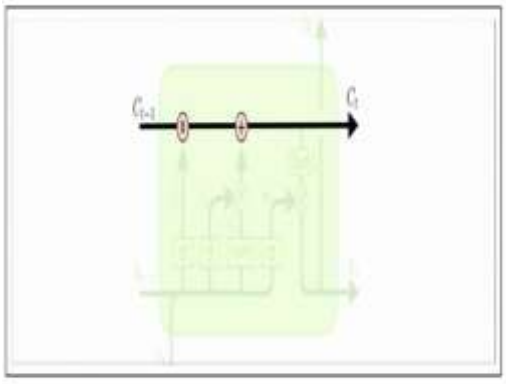
Like regular chains, LSTMs feature a repeating module, but this one is structured in a different way. This is not a single-layer neural network, but rather a network with four unique levels of processing.



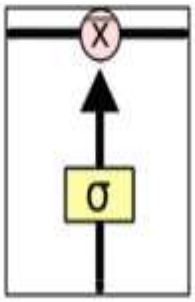
An LSTM is built using a four-layer, interconnected repeating module.



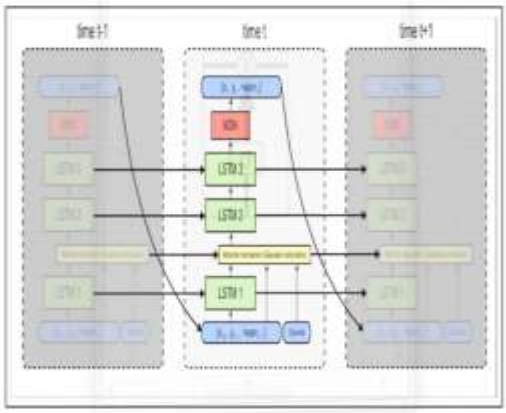
Each line represents a complete vector between two points, as shown in the figure. Point-wise operations, such as vector addition, are represented by the pink circles, while the yellow boxes represent the learnt layers of the neural network. Joined lines indicate an addition, whereas branching lines reveal duplicates moving in different directions. The cell state, shown by the horizontal line at the top of the figure, is crucial to long short-term memory circuits. It's very much like a conveyor belt in terms of cellular health. With just weak linear interactions, it travels straight down the line. It's simple to send data without alteration.



The LSTM may modify the state of a cell selectively via the use of gate structures. If gates are opened, information may flow through. Each one is composed of a sigmoid network layer and a point-wise multiplication process.



The sigmoid layer produces numbers between 0 and 1 to represent the passivity of each component. Nothing is let through if the value is zero; everything is allowed through if the value is one. In an LSTM, the cellular state is protected and regulated by means of three such gates.



Because of its recurrent design, the model may use previously-collected data to guide subsequent decisions. The direction of data transfer inside the model is shown by the arrows (gradients move in reverse for the LSTM variant used in our project H, which is represented by the following composite function):

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (7)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (8)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (9)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (10)$$

$$h_t = o_t \tanh(c_t) \quad (11)$$

H is the hidden vector, while i, f, o, and c are the identically sized activation vectors for the input gate, forget gate, output gate, cell, and cell input, respectively. It is easy to understand the significance of the subscripts in the weight matrix, such as why Wxo is the input-output gate matrix and why the hidden-input gate matrix is. Due to the diagonal nature of the weight matrices connecting the cell and the gate vectors (such as Wci), each gate vector's element m can only receive input from the cell vector's element m. The bias words (those added to the ends of i, f, c, and o) have been eliminated for clarity. In the first iteration of the LSTM algorithm, weights were modified at each time step by estimating the gradient. This approach, however, employs a method known as back propagation across time to calculate the whole gradient. While training LSTM with the full gradient, numerical issues may develop because to the large derivatives. To prevent this, we constrained all experiments in our project to a fixed range for the derivative of the loss with respect to the network inputs to the LSTM layers (before the sigmoid and tanh functions were applied).

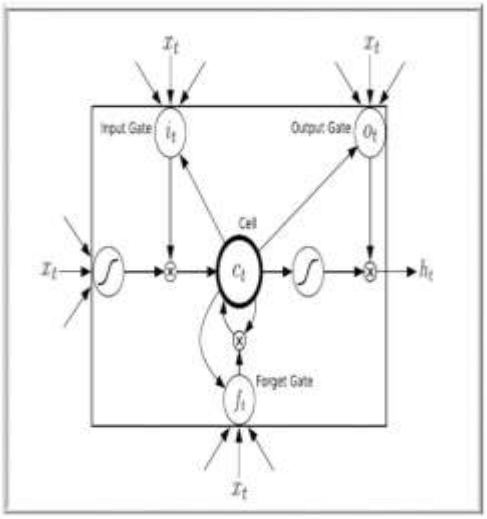


Figure 4.2.2: Long Short-Term Memory Cell

- The picture clearly demonstrates that LSTM relies on multiplicative gates for a wide range of computations.
- These three gate units perform like flexible read/write/reset instructions. O Cell input scalability through input gate (write). O Cell output is scaled by an output gate. O Gate forgets previous cell value and rescales it.
- The network's memory resides in its "cells," or individual nodes.
- The output gate enables reading, thus it must be open for reading to occur; this is an advantage of LSTM over RNN.
- Information stays in the cell as long as the forget gate is open.

Advantages of LSTM:

- Stable degrees of generalization are easier to attain with its aid. 27
- It is possible to generate simulated scenarios and synthetic training data.
- It allows us to do useful activities and get a better understanding of relevant facts.

5. CONCLUSION:

With this framework, we can achieve success. The results from this model are quite sensitive to the values that are entered for its hyper parameters and bias. You must tune them accurately. We demonstrated that LSTM recurrent neural networks may use next-step prediction to generate complex, long-range structure in sequences of discrete and real-valued input. More information is required to properly train the model. Further, it makes use of a cutting-edge convolutional method that enables a recurrent network to condition its predictions on an auxiliary annotation sequence, thereby enabling the synthesis of a wide variety of online handwriting examples that are both realistic and natural.

REFERENCES

1) Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, March 1994.

2) C. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, Inc., 1995.

3) F. Gers, N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research,2002.

4) Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In Proc. ICASSP, 2013.

5) T. Mikolov. Statistical Language Models based on Neural Networks. PhD thesis, Brno University of Technology, 2012.

6) A.Graves. Sequence transduction with recurrent neural networks. In ICML Representation Learning Worksop, 2012.

7) A.Graves. Practical variational inference for neural networks. In Advances in Neural Information Processing Systems, volume 24, pages 2348-2356,2011.

8) A.Graves and J. Schmidhuber. Framewise phoneme classication with bidirectional LSTM and other neural network architectures. Neural Networks, 18:602-610, 2005.
9) A.Graves and J. Schmidhuber. Oine handwriting recognition with multidimensional recurrent neural networks. In Advances in Neural Information Processing Systems, volume 21, 2008.

10) A.Bishop. Neural Networks for Pattern Recognition. Oxford University Press, Inc., 1995.