**Contents**

1. Introduction 3

1. Supporting Literature 3
   1. Literature Review 3
   2. Findings and Proposals 11
2. Working Principle 11
   1. Exploding and vanishing gradient 11
   2. LSTM Architecture 12
   3. LSTM Applications 17
   4. LSTM Drawbacks 17
3. Conclusions 18
4. References 19

**List of Figures**

2.1.1. LSTM Model 4

2.1.2. CNN-LSTM 7

2.1.3. RNN-LSTM 9

2.1.4. Activity Recognition 9

3.2.1. LSTM 12

3.2.2. LSTM Cell 13

3.2.3. Forget gate 13

3.2.4 Input gate 14

3.2.5 Cell state 15

3.2.6 Output gate 16

**Chapter 1**

1. **Introduction**

The term "LSTM" refers to a type of artificial neural network used in deep learning and artificial intelligence. LSTM features feedback connections as opposed to typical feedforward neural networks. Such a recurrent neural network may analyze complete data sequences in addition to single data points (such as photos) (such as speech or video). For instance, LSTM can be used to perform tasks like networked, unsegmented handwriting identification, speech recognition, machine translation, robot control, video games, and healthcare. The most frequently used neural network of the 20th century is LSTM.

A cell, an input gate, an output gate, and a forget gate make up a typical LSTM unit. The three gates control the flow of information into and out of the cell, and the cell remembers values across arbitrary time intervals.

Since there may be lags of uncertain length between significant occurrences in a time series, LSTM networks are well-suited to categorizing, processing, and making predictions based on time series data. To solve the vanishing gradient problem that might occur when training conventional RNNs, LSTMs were created. The advantage of LSTM over RNNs, hidden Markov models, and other sequence learning techniques in many applications is their relative insensitivity to gap length.

**Chapter 2**

1. **Supporting Literature**

**2.1. Literature Review**

1. Das, S., Partha, S. B., & Imtiaz Hasan, K. N. (2020). Sentence Generation using LSTM Based Deep Learning. 2020 IEEE Region 10 Symposium (TENSYMP).

Abstract: The process of predicting pertinent words in a particular order is served by sentence generation. This study aims to develop a process for producing sentences while upholding correct grammatical structure.

Using the Long Short-Term Memory (LSTM) architecture creates a phrase creation system here. The fundamentals of word embedding are generally followed by the system, where words from the dataset are tokenized and transformed into vector shapes.

Following processing, a layer of long short-term memory is used to store these vectors. After each repetition, the system generates a new set of words. As a result of this process, a sentence or passage will eventually be formed using pertinent words. In comparison to other existing approaches, the system's results are fairly compelling.

Methodology: A suitable dataset with a large number of words is used to train the system. A vocabulary is created using special terms. To create new words that are appropriate for the situation, the labels and features are retrieved, and long short-term memory architecture is then utilized. The general steps are broken down into a few key modules, which are discussed in more detail below.

* Data collection and description

Collect data and prepare model

* Processing data for model

Process collected data and train the model

* + Creating vocabulary

Here the words are split and a vocabulary of unique words is created

* + Tokenization

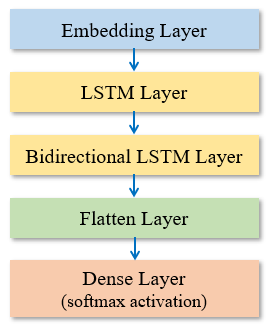
Each word gets tokenized meaning each of the words can be uniquely identified using a number

* + N-grams generation
  + Padding

N-grams generated may be of different lengths, here we use padding to acquire uniformity in length

* + Retrieving labels and features
  + One-hot encoding

Proposed Model: In this paper, the proposed model has several sequential layers

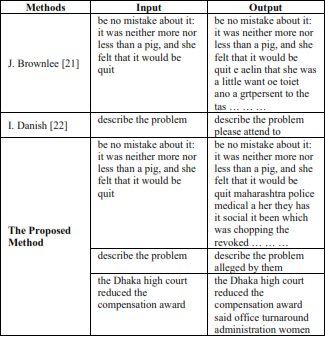


2.1.1 LSTM Model

* Embedding layer: In this layer, a set of words is mapped into vector forms to improve the ability of neural networks because working on numerical data is much easier
* LSTM
* Bidirectional LSTM: Bidirectional LSTM has two hidden states working in opposite directions and hence can work on both past and future states at a time
* Flatten layer: This layer takes the output of the previous layer and puts the value in a single vector
* Dense layer: The dense layer in the model uses a softmax activation function which is nonlinear and uses adam optimizer to fit features and labels
* Generating new word: After training the model, input is given to it in tokenized form. The model predicts a new word after each iteration maintaining the context. After generating every new word, the word also is added to the previous input and the new combination is considered the next input.

Result: 80 percent of the dataset is part of the system's training set. The corresponding loss and accuracy rate is noticed after the training data has been fitted into the model. From Fig, it is clear that the rate of loss is steadily declining while the rate of accuracy per epoch is steadily rising.

The table below shows that the first compared model generates words that aren't even real. Additionally, the meaning of the model's output is not particularly clear. The suggested model produces more insightful results for the same input sequence. No non-existent words are generated by the model either. They compared the model in the second comparison and generates an absurd sequence. To make the resilience of the system more understandable, it also shows a third example produced by the suggested system.



Conclusion: This study presents the idea of developing a deep learning-based model that can produce novel sentences. Word embedding and the Long Short-Term Memory (LSTM) architecture, a modified version of recurrent neural networks, serve as the foundation for all of the methodologies (RNN). The text data are transformed into vector form using word embedding since the neural network finds it easier to operate with numerical data. LSTM and Bidirectional LSTM networks are used to preserve the correct context. Because it can work on both the present and the past at once, the bidirectional LSTM is a crucial layer for the model. The model was effectively trained, and the outcomes were carefully examined and contrasted with those from other models.

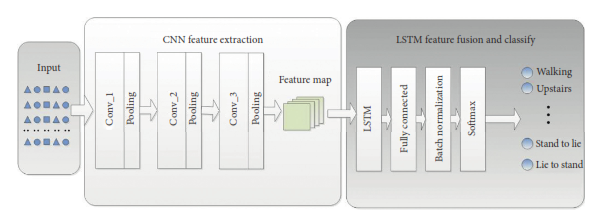
Future Work: In subsequent work, the model can be improved even more so that it can be applied to creating dynamic chatbot suggestions or summarising material.

2. Wang, H., Zhao, J., Li, J., Tian, L., Tu, P., Cao, T., … Li, S. (2020). Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques. Security and Communication Networks, 2020, 1–12.

Introduction: Human behavior recognition (HAR) is the detection, interpretation, and recognition of human behaviors, which can use smart health care to actively assist users according to their needs. Human behavior recognition has wide application prospects, such as monitoring in smart homes, sports, game controls, health care, elderly patients care, bad habits detection, and identification. It plays a significant role in in-depth study and can make our daily life smarter, safer, and more convenient. This work proposes a deep learning-based scheme that can recognize both specific activities and the transitions between two different activities of short duration and low frequency for health care applications.

Dataset: This paper adopts the international standard Data Set, Smartphone-Based Recognition of Human Activities, and Postural Transitions Data Set to conduct an experiment, which is abbreviated as HAPT Data Set. The data set is an updated version of the UCI Human Activity Recognition Using popularity Data set. It provides raw data from smartphone sensors rather than preprocessed data and collects data from accelerometer and gyroscope sensors. In addition, the action category has been expanded to include transition actions. The HAPT data set contains twelve types of actions. A total of 815,614 valid pieces of data are used here.

Proposed method:



2.1.2. CNN-LSTM

The above figure displays the three-part architecture design for the strategy suggested in this paper. The initial step is the preprocessing and transformation of the raw data, which combines the raw acceleration and gyroscope data into a two-dimensional array that resembles an image. The composite image must then be entered into a CNN network with three layers so that it can automatically detect from the activity image, extract the motion features, abstract the features, and then map them into the feature map. The third step involves feeding the feature vector into the LSTM model, establishing a connection between time and action sequence, and then introducing the entire connection layer to fuse the numerous features. Additionally, Batch Normalization (BN) is presented. BN can normalize the data in every layer before sending it to the Softmax layer for action classification.

This paper also examines transition actions in addition to typical fundamental acts. Transition actions are present in a few accessible data sets. For this reason, the HAPT Data Set, also known as the Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set, is used in this paper to experiment. A newer version of the UCI Human Activity Recognition Using Popularity Data collection is used in this dataset. Instead of preprocessed data, it offers raw data from smartphone sensors. Additionally, transition actions have been added to the category of activities. All of the data without labels were removed after the initial processing of the original data. In the end, 815,614 valid data points were gathered. There is a sizable disparity in data volume between transition action and basic action as a result of the low frequency and brief duration of transition action as well as the high frequency and lengthy duration of fundamental action. The six transition actions account for just about 8% of the overall data, which is significantly less than the data amount of the other basic acts. The initial collection of data is divided into three components: a training set, a verification set, and a test set. The training set is used to train the model, the verification set to alter its parameters, and the test set to assess the quality of the resultant model.

3. Agarwal, P., & Alam, M. (2020). A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices. Procedia Computer Science, 167, 2364–2373.

Introduction: Here the architecture for the proposed Lightweight model is developed using Shallow Recurrent Neural Network (RNN) combined with Long Short Term Memory (LSTM) deep learning algorithm. then the model is trained and tested for six HAR activities on resource-constrained edge devices like RaspberryPi3, using optimized parameters. The experiment is conducted to evaluate the efficiency of the proposed model on the WISDM dataset containing sensor data of 29 participants performing six daily activities: Jogging, Walking, Standing, Sitting, Upstairs, and Downstairs. And lastly, the performance of the model is measured in terms of accuracy, precision, recall, f-measure, and confusion matrix and is compared with certain previously developed models.

Dataset: Here Android smartphone having an inbuilt accelerometer is used to capture tri-axial data. The dataset consists of six activities performed by 29 subjects. These activities include walking, upstairs, downstairs, jogging, standing, and sitting. Each subject performed different activities by carrying a cell phone in the front leg pocket. A constant Sampling rate of 20 Hz was set for the accelerometer sensor. A detailed description of the dataset is given in table 1 below.

Total no of samples: 1,098,207

Total no of subjects: 29

Activity Samples: Percentage

Walking 4,24,400 38.6%

Jogging 3,42,177 31.2%

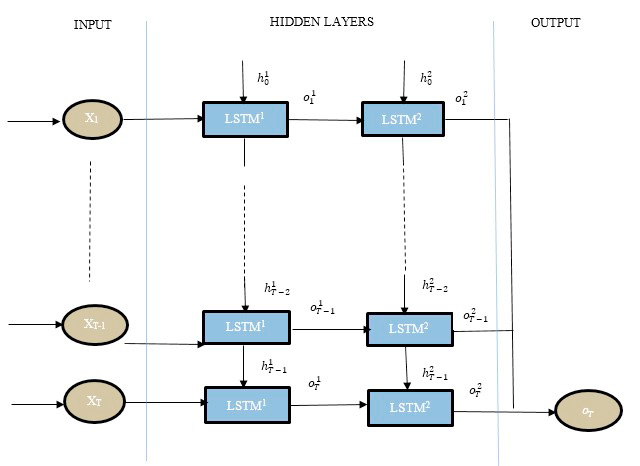
Upstairs 1,22,869 11.2%

Downstairs 1,00,427 9.1%

Sitting 59,939 5.5%

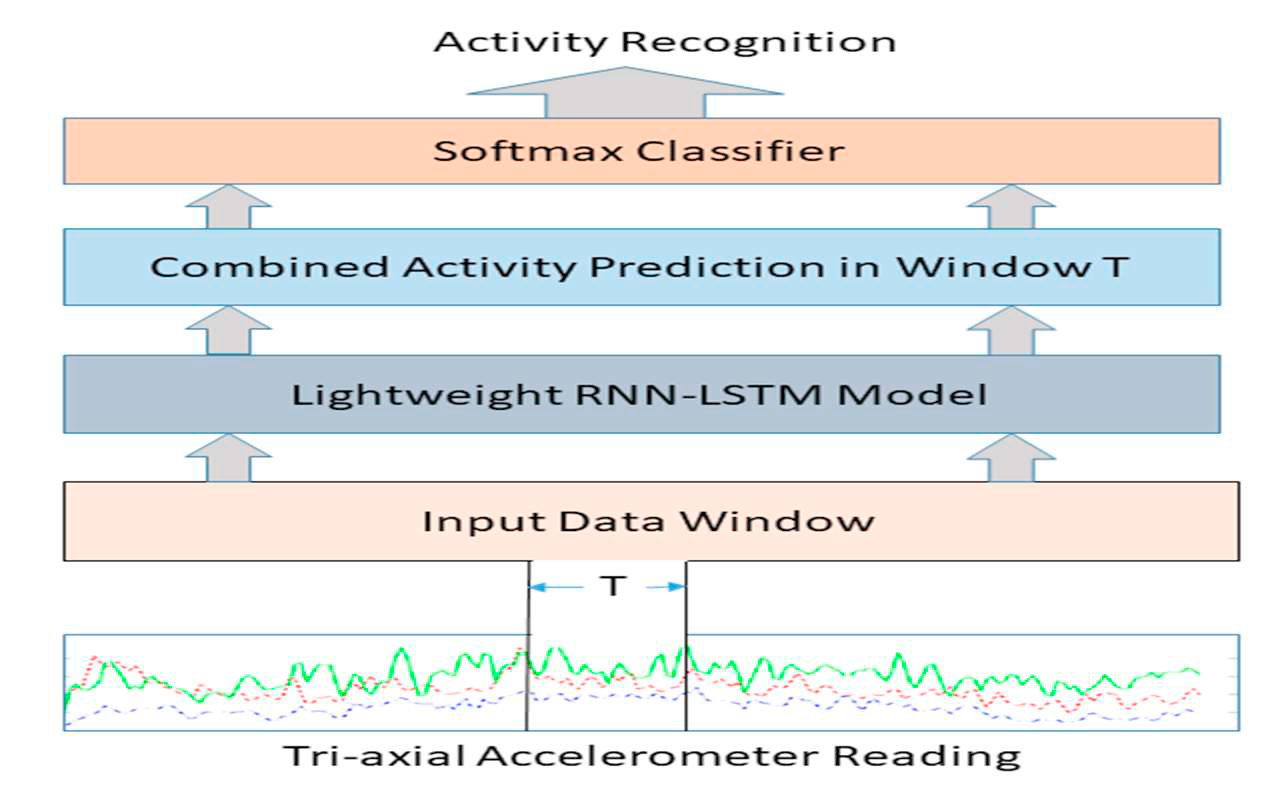
Standing 48,397 4.4%

Proposed method: RNN and LSTM are used to create the proposed model. With only two hidden layers and 30 neurons, it has a shallow structure that makes it practical to install on edge computing devices such as IoT boards (Raspberry Pi, Audrino, etc.), Android, and iOS-based resource-constrained devices.



2.1.3. RNN-LSTM

The below figure depicts the operation of a lightweight RNN-LSTM-based HAR system for edge devices. The reading from the accelerometer is divided into fixed windows of size T. A collection of readings (x1, x2, x3,.., xT-1, xT) recorded in time T, where xt is the reading recorded at any time instance t, serves as the input to the model. Readings from this segmented window are then fed into a lightweight RNN-LSTM model. The model combines the output from many states using a softmax classifier to create a single final output for that specific window as oT.

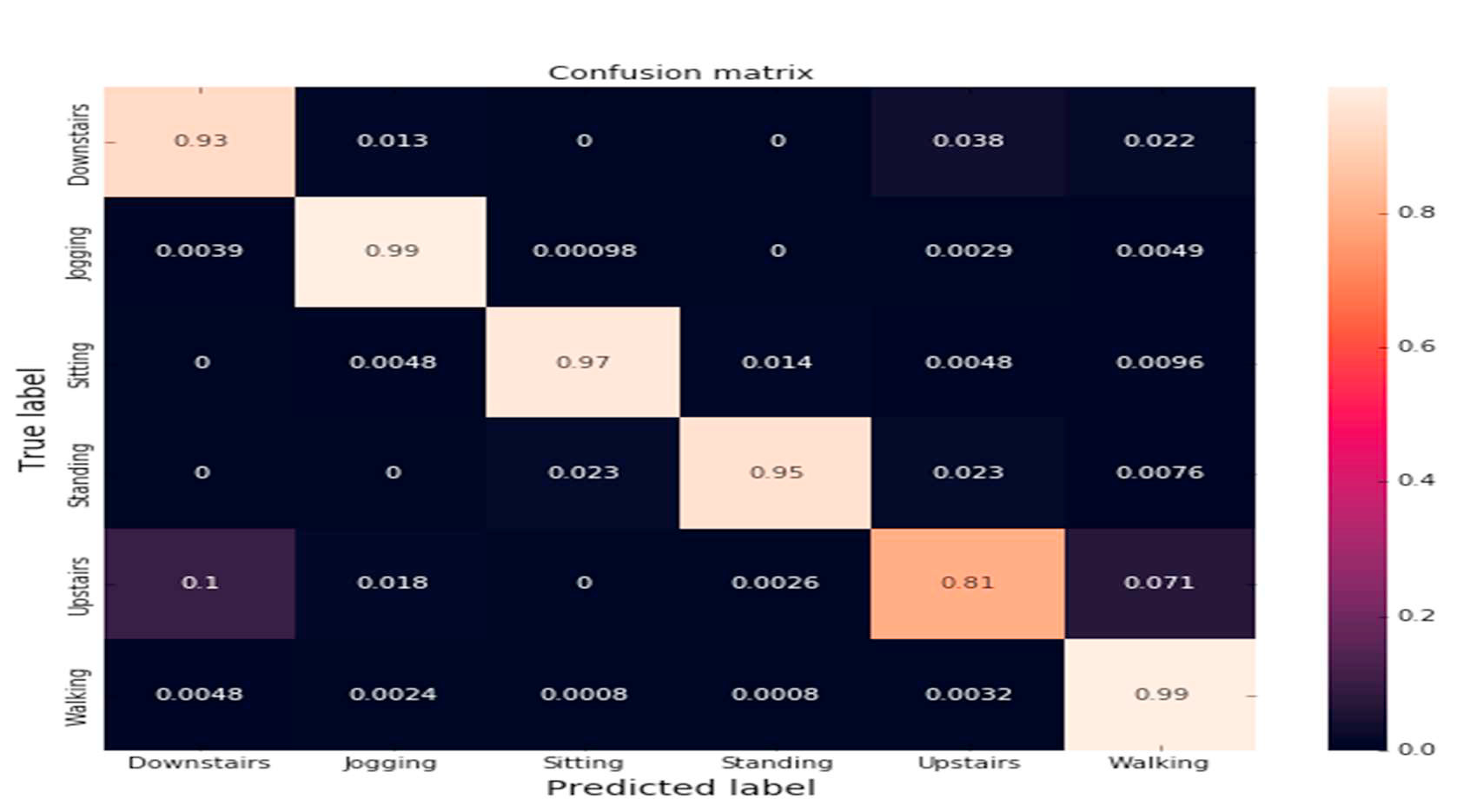


2.1.4. Activity Recognition

Wisdm dataset, split into 70:30 for training and testing, is used to train the lightweight RNN-LSTM. According to the activation function, the model's weights are updated. The cost function between the predicted labels and the ground truth is the mean cross-entropy. The Adam optimizer is employed to update model parameters and minimize the cost function. This model was developed on a Raspberry Pi 3 to test its edge device compatibility. Using the hit-and-trial method, different combinations of parameters, including the number of epochs, batch size, window size, and learning rate, were examined.

• Performance metrics include – Accuracy, Precision, Recall & F1 Score

Result: The evaluation findings for the Lightweight RNN-LSTM model are presented in this part, along with comparisons to some of the earlier efforts. The below figure depicts the model's confusion matrix. For walking and jogging exercises, the Lightweight RNN-LSTM obtained a 99 percent accuracy. For upstairs action, a minimum accuracy of 81 percent is attained.



2.1.5. Confusion Matrix

Equations are used to determine accuracy, precision, recall, and f1-score. Accuracy, precision, recall, and F1-score for Lightweight RNN-LSTM were all 95.78 percent, 95.81 percent, and 95.73 percent, respectively. Recall and precision are calculated to validate performance since accuracy could generate false results if the data in each class of dataset is unbalanced.

Conclusion: In this paper, a lightweight HAR model is built. This model is implemented on a Raspberry Pi3 edge device. Communication delay, costs, and network traffic are all decreased when human activities are recorded on edge devices. In comparison to numerous other machine learning and deep learning models, the proposed model yields better outcomes

Future Work: This research can be expanded in the future to distinguish more intricate behaviors. It may be installed on different edge devices running iOS or Android. Static sliding window architecture was used in the development of this model. Future testing of this design can also include a dynamic windowing mechanism. A single tri-axial accelerometer was also used in the development of this system. Multi-sensor data can be supported by expanding it.

**2.2. Findings and Proposals**

The study has shown that this recurrent system is capable of handling a wide range of issues, including sentiment analysis, computer vision, time series forecasting, text recognition, natural language processing, picture and video captioning, and text recognition. It was discovered that combining CNNs with LSTM to achieve the best performance is a typical strategy when modeling the majority of these issues.

Convolution and pooling layers were utilized in such hybrid models to drastically eliminate representational redundancy while reducing the problem's dimensionality. Additional architecture customization might always be used to increase precision.

Based on the study, the learning rate is the most significant hyperparameter in the backpropagation algorithm, while the forget gate and output transfer function are the most crucial parts of the LSTM block. Therefore, additional research into these elements may result in LSTM variants with enhanced prediction skills. Another equally important study area discusses less computationally intensive learning techniques to modify the parameters that can be learned.

**Chapter 3**

1. **Working Principle**

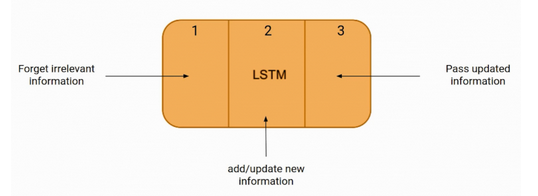
**3.1. Exploding and vanishing gradient**

The main goal of network training is to lower the losses (in terms of cost or error) visible in the network's output when training data is fed through it. We first determine the gradient, or loss, concerning a certain weight set, alter the weights in light of this and then repeat the procedure until we find the weights that will ensure the loss is as minimal as possible. Reverse-tracking is designed with this in mind. The gradient can occasionally become quite minimal. It is significant to remember that certain characteristics of the layers below affect how much gradient is present in a given layer. The gradient will appear smaller if any component is tiny (less than one). The scaling effect is another name for this. A lower value is produced when this effect is multiplied by the rate of learning, which is a negligible number that lies between 0.1 and 0.001. As a result, the findings are almost unchanged and the weights haven't changed much known as the vanishing gradient.

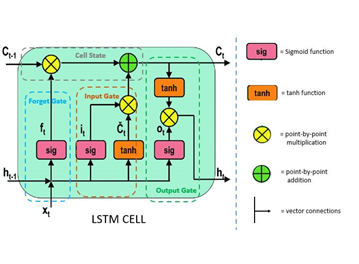
If the weights are modified to be greater than the ideal value and the gradients are severe due to the large components. The problem is also known as the explosive gradients problem. The neural network unit was constructed with the scale factor set to one to stop this scaling effect. The cell was later improved with several gating units, and it was given the name LSTM.

**3.2. LSTM Architecture**

The LSTM network's internal operation is seen below. As seen in the image below, the LSTM is composed of three sections, each of which has a distinct function. The first section determines whether the information from the preceding timestamp needs to be remembered or can be ignored. The cell attempts to learn new information from the input to this cell in the second section. The cell finally transmits the revised data from the current timestamp to the next timestamp in the third section.



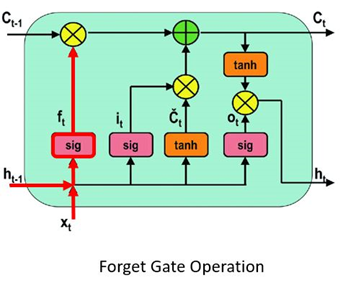
3.2.1 LSTM



3.2.2 LSTM Cell

Gates refers to these three LSTM cell components. The Forget gate, Input gate, and Output gate are the names of the three components, respectively.

**Forget gate**:



3.2.3. Forget gate

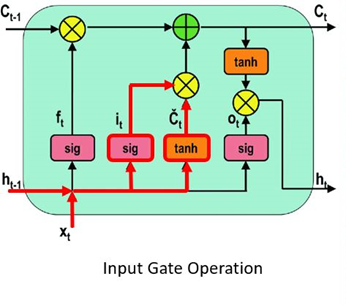
The forget gate is the initial stage of the procedure. In this step, we will determine which pieces of the cell state - the network's long-term memory - are relevant in light of both the prior hidden state and the fresh incoming data.

A neural network is fed with the prior hidden state as well as the fresh input data to do this. This network produces a vector with each member falling within the range [0, 1]. (ensured by using the sigmoid activation). This network (inside the forget gate) is trained to output a value near 0 when an input component is regarded irrelevant and a value closer to 1 when the input component is deemed important. It is helpful to think of each component of this vector as a sort of filter or sieve that lets through more data as the value approaches 1.

The preceding cell state is pointwise multiplied with these output values before being transferred upward. The components of the cell state that the forget gate network has determined to be irrelevant will be multiplied by a value near 0 as a result of this pointwise multiplication, which means they will have less of an impact on the subsequent steps.

In summary, based on the previous concealed state and the new data point in the sequence, the forget gate determines which parts of the long-term memory should now be forgotten (have less weight).

**Input gate**:



3.2.4. Input gate

The new memory network and the input gate are involved in the following step. This step's objective is to decide what new information, in light of the prior concealed state and the incoming input data, has to be added to the network's long-term memory (cell state).

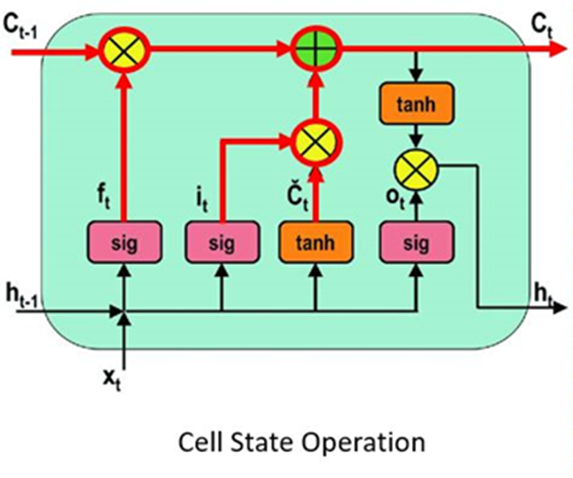
The new memory network is a tanh-activated neural network that has mastered the art of fusing the prior hidden state with fresh input data to produce a "new memory update vector". Given the context from the previous hidden state, this vector essentially contains information from the new input data. Given the new information, this vector indicates how much to update each part of the network's long-term memory (cell state).

As you can see, we are using a tanh because its values fall between and might be negative. If we want to lessen the influence of a component on the cell state, the possibility of negative values is required here.

The second sigmoid function is initially provided the current state X(t) and the previously hidden state h(t-1). Transformed values range from 0 to 1.

The tanh function will then receive identical data from the hidden state and current state. The tanh operator will build a vector (C(t)) containing every possible value between -1 and 1 to control the network. The output values produced by the activation functions are prepared for multiplication on a point-by-point basis.

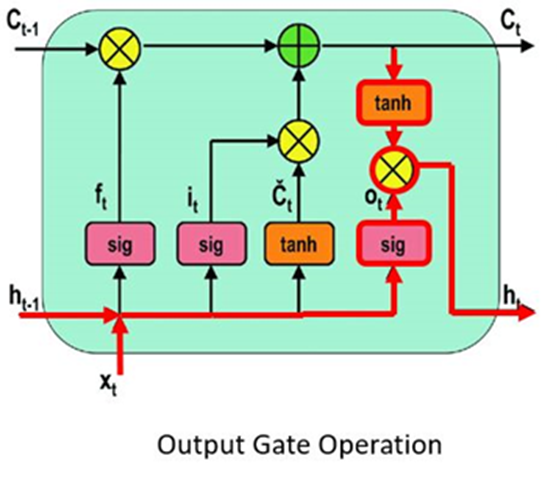
**Cell state**:



3.2.5 Cell State

The input gate and forget gate have provided the network with sufficient information. Making a decision and storing the data from the new state in the cell state comes next. The forget vector f multiplies the previous cell state C(t-1) (t). Values will be removed from the cell state if the result is 0. The network then executes point-by-point addition on the output value of the input vector i(t), updating the cell state and creating a new cell state C. (t).

**Output gate:**



3.2.6 Output gate

The value of the following hidden state is decided by the output gate. Information about prior inputs is contained in this state. The third sigmoid function is first called with the values of the previous concealed state and the current state. The tanh function is then applied to the new cell state that was created from the original cell state. These two results are multiplied one by one. The network determines which information the hidden state should carry based on the final value. Predictions are made using this hidden state.

The new concealed state and the new cell state are then carried over to the following time step.

The forget gate selects which pertinent information from the earlier processes is required to conclude. The output gates complete the next concealed state, while the input gate determines what pertinent information can be supplied from the current stage.

**3.3. LSTM Applications**

Before being used in real-world applications, LSTM models must be trained using a training dataset. The following is a discussion of some of the most demanding applications:

1. Language Modeling: When a word sequence is provided as input for language modeling or text production, words are computed. Language models can be used at several levels, including characters, n-grams, sentences, and even paragraphs.
2. Image processing is the process of analyzing a picture and turning the results into sentences. For this, a dataset with a sizable number of images and correspondingly detailed captions is needed. To forecast the characteristics of the photos in the dataset, a trained model is employed. Data for a photo. The dataset is then analyzed such that only the most intriguing terms are included in it. Data in text format. We attempt to fit the model using these two sources of data. By using input words that were previously predicted by the model and the image, the model's task is to produce a descriptive phrase for the picture, one word at a time.
3. Speech and Handwriting Recognition
4. LSTMs forecast musical notes instead of text by studying a combination of supplied notes fed as input in music production, which is relatively similar to that of text generation.
5. Language translation entails translating a sequence from one language to another. Similar to image processing, only a portion of a dataset comprising phrases and their translations are used to train the model once it has been cleaned. The input sequence is converted to a vector representation (encoding) before being output to a translated version using an encoder-decoder LSTM model.

**3.4. LSTM Drawbacks**

As is common knowledge, everything in this world has advantages as well as downsides. The following are some of the disadvantages of LSTMs:

1. Due to their ability to address the issue of vanishing gradients, LSTMs gained popularity. As it turns out, they are unable to entirely remove it. The fact that the data still needs to be transferred from cell to cell for analysis is the difficulty. Additionally, the cell has grown rather sophisticated as a result of the introduction of new features, such as forget gates.
2. To be taught and prepared for use in the real world, they need a lot of resources and time. Technically speaking, they require a high memory bandwidth because each cell contains linear layers, which the system typically is unable to provide. Thus, LSTMs become relatively inefficient in terms of hardware.
3. Developers are searching for a model that can retain historical data longer than LSTMs in light of the growth of data mining. The human propensity to break down a certain amount of information into manageable chunks for simple memory serves as the motivation for this type of approach.
4. Because of how different random weight initializations affect them, LSTMs exhibit behavior that is very similar to that of a feed-forward neural network. Instead, they choose minimal weight initialization.
5. It is challenging to use the dropout technique to address the overfitting problem with LSTMs. A regularisation technique called dropout excludes input and recurrent connections to LSTM units probabilistically from weight and activation updates during network training.

**Chapter 4**

1. **Conclusion**

Long short-term memory networks are unquestionably an improvement over recurrent neural networks since they are far more creative in their execution of tasks that recurrent neural networks might be able to complete. Long short-term memory is a key advancement in Deep Learning and enhances performance.

You can anticipate more precise predictions and a deeper knowledge of LSTM as these advancements continue to materialize. It dealt with the problem of long-haul conditions of the recurrent neural network, where the recurrent neural network can produce more accurate forecasts from the incoming input but cannot foresee the word stored in the long-term memory. A recurrent neural network cannot provide efficient execution as the total length increases.

**Chapter 5**

1. **References**
2. Abobakr, A., Hossny, M., Abdelkader, H., & Nahavandi, S. (2018). RGB-D Fall Detection via Deep Residual Convolutional LSTM Networks. 2018 Digital Image Computing: Techniques and Applications (DICTA).
3. Bulbul, E., Cetin, A., & Dogru, I. A. (2018). Human Activity Recognition Using Smartphones. 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).
4. Wang, H., Zhao, J., Li, J., Tian, L., Tu, P., Cao, T., … Li, S. (2020). Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques. Security and Communication Networks, 2020, 1–12.
5. Agarwal, P., & Alam, M. (2020). A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices. Procedia Computer Science, 167, 2364–2373.
6. Das, S., Partha, S. B., & Imtiaz Hasan, K. N. (2020). Sentence Generation using LSTM Based Deep Learning. 2020 IEEE Region 10 Symposium (TENSYMP).
7. Shojaei-Hashemi, A., Nasiopoulos, P., Little, J. J., & Pourazad, M. T. (2018). Video-based Human Fall Detection in Smart Homes Using Deep Learning. 2018 IEEE International Symposium on Circuits and Systems (ISCAS).
8. Alemayoh, T. T., Hoon Lee, J., & Okamoto, S. (2019). Deep Learning-Based Real-time Daily Human Activity Recognition and Its Implementation in a Smartphone. 2019 16th International Conference on Ubiquitous Robots (UR).
9. Ullah, M., Ullah, H., Khan, S. D., & Cheikh, F. A. (2019). Stacked Lstm Network for Human Activity Recognition Using Smartphone Data. 2019 8th European Workshop on Visual Information Processing (EUVIP)
10. Sun, B., Liu, M., Zheng, R., & Zhang, S. (2019). Attention-based LSTM Network for Wearable Human Activity Recognition. 2019 Chinese Control Conference (CCC).
11. Deep, S., & Zheng, X. (2019). Hybrid Model Featuring CNN and LSTM Architecture for Human Activity Recognition on Smartphone Sensor Data. 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)