NIET Journal of Engineering & Technology

Volume 15, Issue 1, 2024 **ISSN: 2229-5828 (Print)**

Analysis of Social Media Trends Using Integrated Bi-LSTM and CNN

Abhyudaya Dwivedi[1] , Ajit Kumar[2] ,Dr. Apoorva Joshi[3] ,Ashish Jha[4]

[1](MCA, Department of Computer Applications, Noida Institute of Engineering & Technology

Greater Noida – 201306 Email – abhi76599@gmail.com)

[2](Assistant Professor, Department of Computer Applications, Noida Institute of Engineering & Technology, Greater Noida – 201306 Email – ajitjnvassam@gmail.com)

[3](Associate Professor, Department of Computer Applications, Noida Institute of Engineering & Technology, Greater Noida - 201306 E-mail - apoorvajoshi16@gmail.com)

[4](MCA, Department of Computer Applications, Noida Institute of Engineering & Technology

Greater Noida – 201306 Email – ashishjha120720@gmail.com)

I. ABSTRACT

In the digital age, social media platforms generate vast amounts of data that present both challenges and opportunities for trend analysis. Traditional approaches, such as Bag-of-Words, N-grams, sentiment lexicons, and rule-based systems, often struggle with ambiguity, sarcasm, domain specificity, and language variations. To overcome these limitations, this study proposes an integrated deep learning approach combining Bidirectional Long Short-Term Memory (Bi-LSTM) and Convolutional Neural Networks (CNN). Bi-LSTM captures contextual understanding and sequential dependencies, while CNN extracts key patterns, enhancing the robustness and accuracy of social media trend analysis.

Using a Twitter sentiment dataset, our methodology includes preprocessing steps such as noise removal, tokenization, and vectorization to optimize model performance. The proposed model achieved a validation accuracy of 99.40% and a recall value of 90%, demonstrating its effectiveness in analysing and predicting social media trends. This research provides a data-driven framework that can assist businesses, policymakers, and researchers in understanding digital communication patterns and forecasting emerging trends.

II. KEYWORDS

Deep learning, Bi-LSTM, CNN, Machine learning, Data preprocessing, Natural language processing (NLP)

**Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.**

III. INTRODUCTION

In the modern digital era, social media platforms have emerged as powerful channels of information, communication, and influence. The unequal volume of data generated on these platforms raises both challenges and opportunities for researchers seeking to decode the trends that shape our society. Understanding these trends is not only for academic purposes but also for businesses, policymakers, and other fields.

In this study, we investigate the field of social media trend analysis using a sophisticated combination of deep learning methods, namely Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM).

Traditional methods of trend analysis often fail to capture the context of social media data, which is characterized by its unstructured nature, rapid evolution, and diverse content types. These traditional methods of social media trend analysis are

- 1.1 BoW (Bag-of-words)
- 1.2 N-grams Method
- 1.3 Sentiment Lexicons
- 1.4 Machine learning Algorithms
- 1.5 Rule-based systems for an analysis of trends

While we use traditional methods, we have to face several challenges like

- 1.1 Ambiguity and Context
- 1.2 Sarcasm and Irony
- 1.3 Domain specificity
- 1.4 Language variations
- 1.5 Subjectivity and opinionated texts

We want to get over these restrictions and offer a more reliable framework for trend research by utilizing the strengths of Bi-LSTM and CNN. The combination of Bi-LSTM and CNN represents a match between time and pattern learning; both architectures work well together in that BiLSTM can capture contextual understanding and sequential dependencies over time. While CNN is good at capturing patterns, which helps us find important themes and subjects in the data, The combination of these two strengths allows the model to analyse social media trends more robustly and comprehensively. It can manage the data's sequential structure as well as the patterns that emerge from the relationships between various parts of the vast sea of social media content.

Our goal is to better understand social media trends and how they affect other fields, such as crisis management and public opinion analysis. By using a data-driven methodology based on deep learning, which seeks to forecast future trends based on patterns found in the data, in addition to describing the current trends seen in social media data, For corporations and governments, this predictive feature is essential since it helps them foresee changes and plan ahead with initiatives.

In the following sections of this paper, we will talk about the framework of Bi-LSTM and CNN and our approach to combining these methods in the context of social media trend analysis. We will discuss the

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

experimental findings and a demonstration of our method's effectiveness versus using any of the approaches alone.

We will also talk about the applications of our results and suggest directions for further study in this expanding subject.

Fundamentally, our study provides an explanation of social media trends and provides a strong framework for analysis that surpasses conventional approaches to uncover new avenues for comprehending and managing the power of digital communication.

IV. METHODOLOGY

1.1 Data Acquisition

In order to train our deep learning model, we have obtained the labelled dataset from **kaggledataset.com**, which is **twitter sentiment analysis.csv**. The dataset was noisy, so we pre-processed it to remove the noise. This resulted in obtaining only highly visible social media posts, from which it is simple to figure out the ideology of the user who posted them.

A.

1.2 Data Pre-processing

Using a variety of Python packages, we have pre-processed the dataset to make our work easier. Using Python's **Regular expression (re)** module, we have eliminated **@username** from the social media statements because usernames in the statements have no influence on our research.

Following the process of removing usernames from the statement, we have moved on to the removal of stopwords. For this, we have used the Python **Natural Language Toolkit (NLTK)** library, which has predefined lists of word types that are stopwords. Then we used the **Regular Expression (re)** module to determine which words were stopwords and remove them. The definitions of the words in this **NLTK** library are only for the purpose of preserving grammar beauty; they have no influence on our study. We are limited to using only highly attention-grabbing terms to train our model.

Next, we have removed the emoticons. Emojis don't have any significance in our text analysis; however, they are present in statements nearly everywhere. Emojis were eliminated by utilizing the **Regular expression (re)** module, which matches each emoji to its Unicode and eliminates it.

We also remove links and hyperlinks that aren't relevant to text analysis or trends. Here, we locate and eliminate hyperlinks and links in our dataset by using the more powerful and user-friendly **Regular expression(re)** module.

The hashtags are now extracted from the dataset and added to a different dataframe so that it is simpler to analyse which hashtag is trending more and which dataframe it belongs to: positive, negative, or neutral. We have established a method for hashtag discoveries and applied it to various datasets based on the labels of the statements. To complete this work, we have to use our **Regular expression(re)** module. Once we've completed this process successfully, we are prepared to determine which hashtag in our dataset is trending more.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

1.3 *Featuring and vectorization of dataset*

This section of the research makes use of the Keras preprocessing module, which offers a number of tools for preparing text data before it is used as input for our deep learning neural network model.

The **Keras.preprocessing.text** module offers a number of features that we have to utilize to achieve our objectives.

- Tokenization: This is the process of breaking sentences into phrases, which are then further broken down into words and sub words.
- Sequencing: Text can be transformed into an integer sequence by replacing each word or character that has been tokenized through the tokenization process with a suitable integer index from the vocabulary.
- Padding: Because neural networks require stable input sizes and because our dataframe input sizes may become unbalanced after sequencing in integers and splitting, we have to pad up all sequencing integers with zeros in this process to give them a fixed, uniform length.

These Keras preprocessing features allow us to meet the requirements of our dataframe for deep learning model training and testing.

1.4 *Splitting The dataset into A Training set and A Testing Set*

We have a dataset which we have to divide into two parts: 20% of the data will be used to test our trained deep learning model, and the remaining 80% of the data will be utilized for training.

We have to use a random seed to split our data so that the reproducibility of the data remains constant. This is because each time the code is executed, the dataset is randomly split into a training set and a testing set. Even with the same code and model parameters, this randomness causes variations in our model's performance between different executions of code.
If you run the code on the same machine or on a different machine with the same random state value, the random state parameter of the random seed has guarantees that your results are reproducible and that you will obtain the same split.

1.5 *Initialization of Deep Learning Model*

The first step in initializing a Deep learning sequential model is to import Keras layers and arrange them in a stack. We have imported certain layers to carry out deep learning model training from the Keras Layers Library.

1.6 *Embedding layer*

Inside the embedding layer maintains an embedding matrix where each row corresponds to the vector representation of the word and the columns reflect the dimensions of the embedding space. This layer is mostly utilized for the embedding of words in natural language.

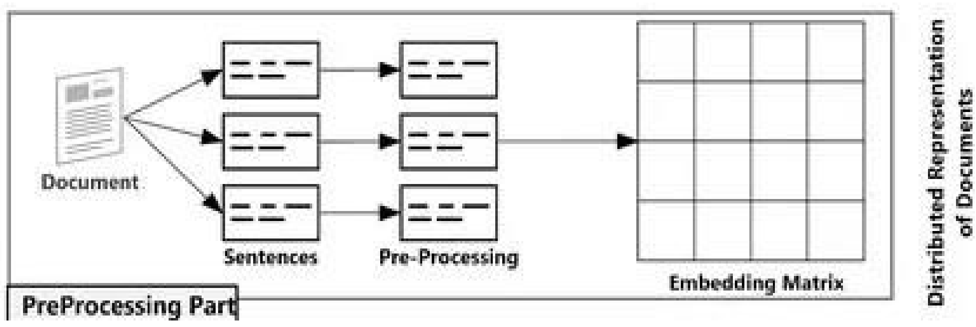
Number of Rows in Embedding matrix = Number of Unique words in the dataset

An embedding layer have receives a 2D tensor as input and produces a 3D tensor as output.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

Batch size and input length have been provided to the embedding layer as input. where batch size is a representation of how many training samples are used in a single gradient descent run. The length of the input sequences that have been fed into the embedding layer is represented by the input length.

We have batch size, input length, and output dimension when the embedding layer produces a 3D tensor. The dimensionality of the embedding space is represented by the output dimension. It determines the vector representation size allotted to every word in the vocabulary.



1.7 *one dimensional convolutional Layer*

The embedding layer produces a dense embedding vector for each word after receiving an integer sequence of words as input. The sequence formed from these embedding vectors has sequence length and is considered a one-dimensional input tensor.

The embedding layer returns 3D tensor output, which is converted into a 2D matrix with the shape (input_length, output_dim) where rows of the matrix correspond to a word in the input sequence and each column corresponds to a dimension of the word embedding vector.

An essential building element for processing 1D data is the convo1D layer, a kind of convolutional neural network (CNN) layer that executes a convolution operation over a 1D input signal to extract local patterns or features.

The Conv1D layer operates a 2D matrix by sliding a one-dimensional convolutional filter along the sequence of word vectors.

This operation involves sliding a small window called a filter or kernel along the sequence and computing the dot product between the filter and the sequence within the window. This process generates a new sequence (feature map) by capturing local patterns or features from the input.

1.8 *Purpose of Convo1D*

- Feature extraction: Convo1D layers use filters or kernels to search through data and find patterns and features in time series or sequences.
- Local Receptive filter size: The layer looks for local patterns in the incoming data by using the local receptive filter size.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

- Parameter sharing: By using the same set of parameters for every place in the sequence, the Convo1D layer enables the model to learn from various input segments and enhance generalization.

1.9 *Parameters of Convo1D*

- Number of Filters: It indicates how many feature maps are produced as an output once vector processing is completed.
- Filter size: It specifies each filter's length. Smaller filters catch small details, while larger filters capture larger patterns.
- Stride: It sets the step size at which the filter examines the sequence of input.
- Padding: It determines if the input sequence needs to be padded, usually in order to keep the output sequence's length constant. There are two options: "Same," which indicates padding to retain the input size, and "Valid," which indicates no padding at all.
- Activation function: The computations of neural networks become non-linear as a result. Applying an activation function to a neural network's output neurons allows it to decide whether or not to activate a neuron's output in response to the input it receives.
- Weights and Biases: Weights and biases related to each filter in the layer are gathered during training.

1.10 *Max Pooling 1D layer*

Convolutional neural networks apply the Max Pooling 1D type of pooling layer to analyse 1D sequential data. It shortens the input feature maps while keeping all of the relevant information by downsampling them along the temporal dimensions.

The MaxPooling1D layer's primary goal is to downsample the input representation by reducing the spatial dimensions of the input feature maps.

The MaxPooling1D keeps the most significant features while removing unnecessary or unimportant data by choosing the maximum value inside each pooling window.

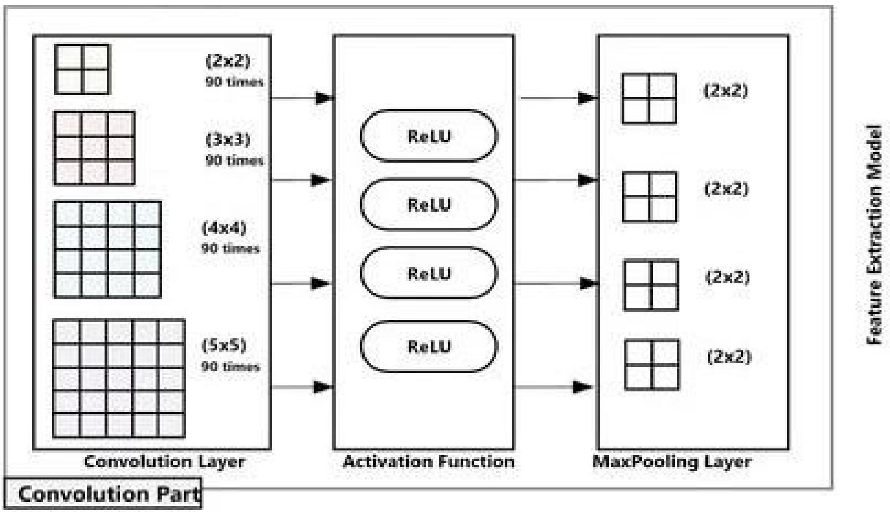
1.11 *Purpose of Maxpooling 1D*

- Dimension reduction: The input feature map's length is decreased through max pooling. As a result, the network has fewer parameters and computations, which helps to avoid overfitting.
- Feature extraction: The layer helps in maintaining the important aspects of the input by choosing the highest value within a pool.

1.12 *Parameter of Maxpooling 1D*

- Pool Size: The size of pooling window means the layer will examine two adjacent values at a time if the pooling window's size is 2.
- Stride: Stride controls how the window moves across the input, which means a stride of 1 indicates that the window shifts one element at a time.
- Padding: To keep the input feature map's size unchanged or avoid edge values from getting lost, padding can be added to the input array.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.



1.13 BiLSTM (Bi- Directional Long short-term Memory)
A kind of recurrent neural network architecture known as BiLSTM (Bi-Directional Long Short-Term Memory) comprises two LSTM layers: one handles input in the forward direction (Left to Right) and the other handles input in the backward direction (Right to Left). As a result, the model is able to incorporate data from both historical and future contexts.

- Memory Cell: A memory cell with long-term data storage capacity is present in every LSTM layer.
- Gates: Gates such as input, forget, and output gates are used by each layer of the LSTM to regulate the information flow into and out of the memory cell.
- Learning Dependencies: LSTM layers are useful for sequence modelling jobs because each layer may identify long-range dependencies in the data.

1.14 Advantages of BiLSTM:
- Contextual Understanding: It can understand sequence context better because of BiLSTM bidirectional processing, which enhances performance.
- Long Range Dependencies: BiLSTM learns long-range dependencies in the data more efficiently by processing the data in both directions.

1.15 Disadvantages of BiLSTM:
- Complexity: Because bidirectional LSTM uses two LSTM layers, it is more complex and computationally difficult than unidirectional LSTM.
- Training Time: Training periods may be longer due to the extra computation, especially with larger datasets.

1.16 Dense Layer Or Fully Connected layer:
A dense layer in neural networks, also known as a fully connected layer, is a layer in which every neuron is connected to every other neuron in the layer above it.

Dense layers are a core component of many neural network structures and are essential for extracting complicated patterns from data.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

The layer uses biases and learnable weights to perform a linear transformation on the input data.

An activation function is typically applied to the model after the linear transformation to provide non-linearity and enable it to learn complicated representations.

1.17 Parameters of the Dense Layer:

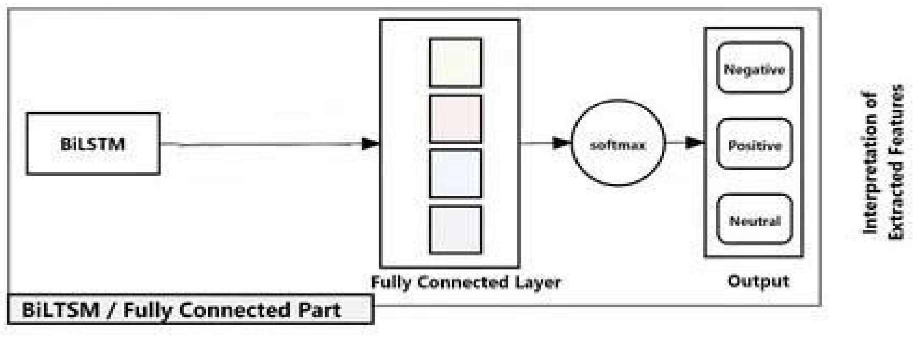
- Weights: The links between the input and output layers are determined by a matrix of learnable parameters called weights.
- Biases: A vector of learnable parameters, or biases, are those that shift the activation function and make the model more data-fitting.
- Number of Neurons: One important factor that affects both the layer's density and the model's complexity is the number of neurons in the dense layer.
- Activation function: The learning performance of the model can be strongly affected by the activation function that is applied to the linear transformation.

1.18 Advantages of Dense Layer:

- Flexibility: Dense layers are adaptable and can be used in many different kinds of neural networks and for a wide range of activities.
- Learning capacity: Dense layers are effective for a variety of tasks because they can figure out complex connections between input features.
- Simplicity: Dense layers are easy to use and understand, especially for beginners, due to their simplicity.

1.19 Disadvantages of Dense layers:

- Overfitting: Dense layers have the potential to overfit, especially if they have a large number of parameters compared to the volume of training data.
- Computational cost: The dense layer's high number of connections could make it computationally costly when working with high-dimensional data.


BiLSTM
softmax
Negative
Positive
Neutral
Fully Connected Layer
Output
BiLTSM / Fully Connected Part
Interpretation of Extracted Features

1.20 *Compilation Of Sequential Deep learning Model*

Compilation is the process of configuring the model for training by specifying the loss function, optimizer, and evaluation metrics.

a) Loss function:

The function measures the difference between the predicted probabilities from the model's output and the true labels of one hot encoded target label. It calculates the cross-entropy loss for each sample and averages it across the batch.

Categorical cross-entropy: In multi-class classification problems, the loss function commonly used is one that represents each class as a binary vector with a single 1 representing the true class. The target labels in this case are one hot encoded.

During the data gathering process, true class labels are frequently obtained and added to the dataset. which, in the case of multi-class classification issues, can be encoded in a standard format. One-hot vectors are created by further encoding the integer-encoded labels.

The model's predictions are compared with the one hot encoded true class labels during training. The difference between true class labels and the model's prediction can be measured by the loss function.

1.21 *Optimizer:*

During training, the optimizer modifies the model's weights in an effort to reduce the loss function.

1.22 **Nadam Optimizer (Nesterov-accelerated adaptive moment estimation):**

It uses moment estimation and exponential moving averages of gradients much like Adam and preserves adaptive learning rates for every parameter. It is a hybrid of the Adam optimizer with Nesterov momentum.

Through the adjustment of gradient updates, Nesterov momentum accelerates the optimization process. As compared to conventional optimizers like SGD *(Stochastic Gradient Descent)*, Nadam provides smoother updates and faster convergence.

1.23 *Evaluation Metrics:*

Metrics help you keep an eye on the model's performance both during training and during evaluation. They provide additional insight on the model's learning efficiency.

1.24 *Training of sequential model:*

Before we start training our model, we must initialize a few variables that would be important to its performance and monitor the model's training accuracy.

1.25 *Batch size and Epoch:*

The number of training samples used in a single gradient descent iteration is represented by the batch size. Our model batch size is set to 64 for further training, which allows the use of 64 training samples in a single cycle. The epoch represents the concept that we learn in a cycle. during every loop iteration. Using a given batch size, the model is trained on the training set for one epoch.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

1.26 *Training Loop:*

After that, the code enters a loop that will continue for the given number of epochs. The model is trained on the training data for one epoch in each loop iteration, with a specified batch size. The verbose option is used to adjust the amount of information printed to the console or logs throughout the model training process.

1.27 *Model predictions and accuracy on the training set:*

Model.predict() is used to make predictions for the training set, and **np.argmax()** is used to extract the predicted labels from the model's output.
The **accuracy_score** function of the **sklearn.metrics** module is used to calculate accuracy on the training set by comparing the predicted labels to the actual training labels. To the desired list, where the accuracy of every epoch is stored, is appended the estimated training accuracy.

1.28 *Model predictions and accuracy on the test set:*

After the model has been trained for one epoch, predictions are produced on the test set using **model.predict().** The predicted labels are then retrieved from the model's output using **np.argmax()** along the last axis, most likely converting probability to class labels.

By comparing the predicted labels with the actual test labels, the **accuracy_score** function from the **sklearn.metrics** module calculates the accuracy of the test set. An attachment to the list contains the calculated test accuracy.

V. CONCLUSION

Our suggested technique for analysing social media trends is based on an integrated CNN-BiLSTM approach for statements. Due of the model's high sensitivity to overfitting, training was limited to the 7th epoch. We wanted to record the number of accurate predictions for positive and negative outcomes as well as inaccurate forecasts using in order to test our model against data and analyse it.

Accuracy= True Positive + True Negative / Total predictions

We have computed basic accuracy using this formula. After training and testing with an accuracy of 89%, the model produced validation accuracy of up to 99.40% when tested against our dataset. Moreover, the model's 90% recall value shows how effective it is at analysing social media trends. A higher recall number indicates greater accuracy or completeness across the board for the category of assertions.
The precision value and F1 score of 91% suggest that our analysis is very reliable.

REFERENCES

- "Analysis of Social Media Trends Using Integrated Bi-LSTM and CNN," unpublished manuscript, Feb. 2025. Accessed: Feb. 17, 2025. [Online]. Available: https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/53567526/93ac2c2a-e2a7-4fa1-99f9-a956c7f1deec/Paper-Social-media.docx

- S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735-1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

- Y. Kim, "Convolutional neural networks for sentence classification," in Proc. 2014 Conf. Empirical Methods Natural Lang. Process. (EMNLP), Oct. 2014, pp. 1746-1751, doi: 10.3115/v1/D14-1181.
- R. Bharal and O. V. V. Krishna, "Social Media Sentiment Analysis Using CNN-BiLSTM," Int. J. Sci. Res. (IJSR), vol. 10, no. 9, pp. 659-664, Sep. 2021. [Online]. Available: https://www.ijsr.net/archive/v10i9/SR21913110537.pdf
- L. Jixian et al., "Social Media Multimodal Information Analysis based on BiLSTM-Attention-CNN-XGBoost," Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 12, 2022, doi: 10.14569/IJACSA.2022.0131215.
- Kaggle, "Twitter Sentiment Analysis Dataset," 2023. [Dataset]. Accessed: Feb. 17, 2025. [Online]. Available: https://www.kaggle.com/datasets/kazanova/sentiment140
- C. N. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in Proc. COLING 2014, Aug. 2014, pp. 69-78. [Online]. Available: https://aclanthology.org/C14-1008
- F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," Neural Comput., vol. 12, no. 10, pp. 2451-2471, Oct. 2000, doi: 10.1162/089976600300015015.
- Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
- M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Software]. Accessed: Feb. 17, 2025. [Online]. Available: https://www.tensorflow.org/

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.