Deep Fake Face Detection Using Deep Learning

Parth Gupta

Dept. of Computational Intelligence, SRM Institute of Science and Technology

Chennai, Tamil Nadu, India

DOI:10.37648/ijrst.v15i01.005

[1]*Received: 17 January 2025; Accepted: 22 February 2025; Published: 02 March 2025*

ABSTRACT

The phrase ”Seeing is believing” no longer holds true in today’s world, and this shift has profound consequences across numerous sectors. With the rapid advancement of technology, creating deepfakes has become increasingly accessible, even though mobile applications. Detecting deepfakes is a complex task, and it’s becoming harder for the human eye to identify them. However, some researchers are actively seeking solutions. Deepfakes are synthetic media generated using AI algorithms, where the machine learns features from both the target and source images. The result is the overlaying of the target image onto the source. This paper focuses on video deepfake detection using deep learning neural networks, particularly Long Short-Term Memory (LSTM) and InceptionResNextV2. Through transfer learning, where a pre-trained InceptionResNextV2 CNN extracts features, the LSTM network processes these features for classification.

Index Terms**:** *Deepfake detection; deep learning; Long Short- Term Memory (LSTM); InceptionResNetV2; convolutional neural networks (CNNs); video manipulation; AI-generated media; transfer learning; facial feature extraction; temporal sequence analysis.*

INTRODUCTION

In recent years, the capabilities of digital technology have expanded to a point where it can substantially modify content displayed online. A deepfake can be likened to an advanced form of Photoshop for videos, utilizing artificial intelligence (AI) to train systems to mimic human facial expressions and voices. By feeding the system with real-life photos, videos, and voice samples, it learns to generate fabricated videos of individuals. Deepfakes are created either by replacing the face of one person with another through machine learning, or by utilizing generative adversarial networks (GANs). This paper explores how deepfake creation can harm privacy, democracy, and security, as the manipulated media can easily deceive both people and computer systems.

The advancement of deep learning has enabled the creation of technologies like deepfakes, which pose significant risks to privacy, democracy, and national security. Deepfakes, in simple terms, refer to digitally manipulated media—such as images or videos—where one individual’s likeness is replaced by another’s. These manipulations often rely on sophisticated AI algorithms to generate or modify video and audio content, making it appear as though a person is saying or doing things they never did. Deepfake technology is commonly employed in contexts like humor, pornography, or politics, raising serious ethical concerns regarding privacy, consent, and the spread of misinformation. Tackling these challenges requires a holistic approach, encompassing regulatory frameworks, heightened public awareness, and the responsible use of AI. With the rise of advanced deep neural networks and vast data availability,

[1] How to cite the article: Gupta P.; March 2025; Deep Fake Face Detection Using Deep Learning; *International Journal of Research in Science and Technology*, Vol 15, Issue 1, 68-76, DOI: http://doi.org/10.37648/ijrst.v15i01.005

distinguishing between real and fake content has become increasingly difficult, deceiving both humans and automated systems. This paper contributes to ongoing research in deep learning and deepfake detection, with a particular focus on algorithms such as Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) for identifying deepfakes.

A. Convolution Neural Networks

Convolutional Neural Networks (CNNs) are a specific kind of neural network designed to process and interpret visual data. This specialization makes CNNs crucial in computer vision tasks, including image classification, object detection, and image enhancement. A CNN is structured with layers that perform convolution, pooling, and classification. The convolutional layers apply filters to the input image, detecting features such as edges and textures, while the pooling layers reduce the spatial dimensions of the data, simplifying it without losing key information. Finally, fully connected layers classify the processed data based on the extracted features.

1) Convolutional Layer: - The convolutional layer stands as a pivotal component within a CNN, playing a central role where the bulk of computations occur. It is the central building block that processes the input data. In this layer, convolutions are applied using small filter matrices (also called kernels or filters) on the input image to detect features like edges, textures, or shapes. The filter slides over the input image (or the output of the previous layer) to scan and detect patterns within the receptive field.

Wout = (W - F + 2P/S) + 1

2) Pooling Layer:- Following convolutional layers in a CNN, pooling layers play a crucial role in diminishing the spatial dimensions of the input while preserving essential features. Unlike convolutional layers that use filters, pooling layers employ kernels to aggregate and summarize information from small regions of the input. Typical pooling operations, such as max-pooling and average-pooling, are employed to down sample the input. This process reduces complexity and enhances the performance of CNNs. Wout = ((W - F) / S)+1

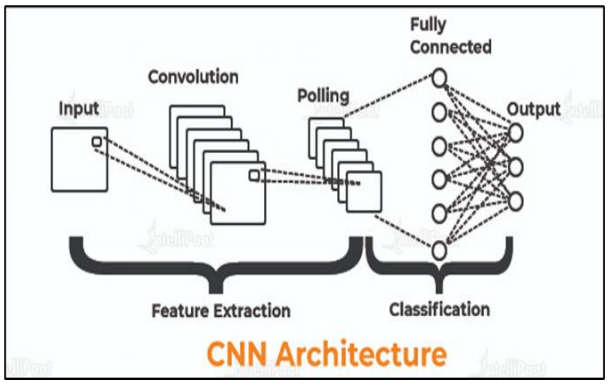

Fig. 1. CNN Architecture (1)

3) Fully Connected Layer:- The fully connected layer is the last part of the CNN and is responsible for classification based on the features extracted in the previous layers. Fully connected implies that every node or neuron in this layer is connected to every activation unit or neuron from the preceding layer. The connections between these nodes involve weighted connections, and the activation units apply non- linear activation functions to produce the final classification output.

B. *Recurrent Neural Network (RNN)*

Recurrent Neural Networks (RNNs) are a type of artificial neural network specifically designed to handle sequential or time-dependent data. These models excel in tasks such as language translation, natural language processing (NLP), speech recognition, and generating image captions, where the order of information is crucial. While RNNs share similarities with feedforward and convolutional neural networks in terms of learning from data, their defining characteristic is their ability to maintain a form of ”memory.” This memory allows RNNs to incorporate information from previous inputs to influence current decisions and outputs.

However, standard RNNs face challenges with maintaining long-term dependencies due to issues like vanishing or exploding gradients during training. This can limit their effectiveness in learning from distant past data points. To address this, specialized variants such as Long Short-Term Memory (LSTM) networks were developed. LSTMs use gating mechanisms that allow them to better capture and retain long-term dependencies, enabling more effective learning from both recent and distant inputs.

C. *Residual Next Convolutional Neural Network (ResNeXt)*

The Residual Next Convolutional Neural Network, known as ResNeXt, marks a notable advancement in deep learning architecture.[6] Building on the core principles of the Residual Network (ResNet), ResNeXt introduces the concept of ”cardinality,” which significantly enhances both the model’s ability to represent complex features and its computational efficiency.

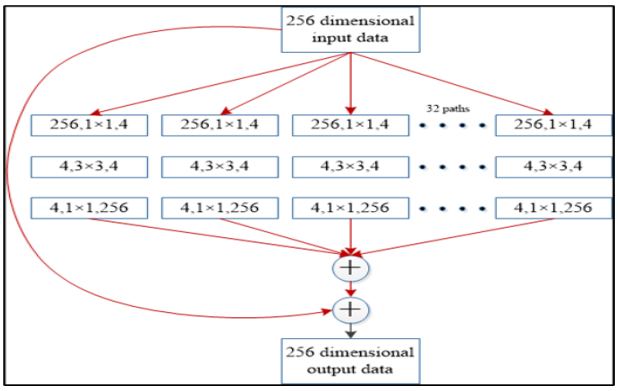

Fig. 2. ResNext Architecture (2)

This innovation recognizes the importance of aggregating multi-scale information and promoting collaborative learning within convolutional layers. By leveraging this collaborative framework through cardinality, ResNeXt achieves high accuracy in tasks like image classification while maintaining a simplified and efficient design.

ResNeXt’s architecture is based on ResNet, but with the added cardinality parameter that controls the number of parallel paths within each block of layers. These parallel paths, composed of small groups of convolutional layers, work together to im- prove feature learning. This cardinality increases the model’s capacity for expressive, diverse feature learning, allowing it to aggregate information efficiently at different scales. This design gives ResNeXt its remarkable performance in various computer vision applications, with the flexibility to scale based on available computational resources.

D. *Long Short-Term Memory (LSTM)*

Long Short-Term Memory Networks (LSTMs) are a particular type of RNN that are designed to process sequential data more effectively by addressing the vanishing gradient problem found in traditional RNNs.[2] LSTMs can model long-term dependencies, making them especially useful for tasks involving extended sequences of information. Their architecture includes three main components: the forget gate, input gate, and output gate. The forget gate determines whether information from the previous time step should be retained or discarded. The input gate processes new information from the current input, while the output gate helps pass updated information to the next step in the sequence. This structure enables LSTMs to overcome difficulties in maintaining important information over long periods.

In the context of deepfake detection, LSTMs are employed to analyze sequences of frames extracted from videos. These frames are preprocessed to identify key features, which are then fed into the LSTM for training. By learning patterns and temporal dependencies across the frames, the LSTM can distinguish between real and manipulated content. The model is trained on datasets containing both authentic and deepfake videos, allowing it to detect deepfakes based on learned deviations from genuine video behavior.

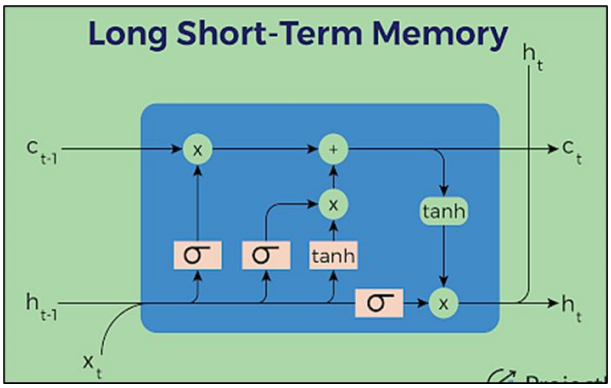

Fig. 3. LSTM Architecture (3)

LITERATURE SURVEY

In the recent times there is an explosive growth in the number of deepfakes. In the present times there are many software that facilitate the creation of these deepfakes. They are becoming a threat to privacy, democracy and trust. So, there is an increase in the demand for deepfake analysis. We are listing some of the approaches for deepfake detection.

- Y. Li, S. Lyu, have proposed a new method of comparison between generated face areas and their surrounding regions using Conventional Neural Networks. The method was based on observation whether images of limited resources can be generated by the DF algorithm.

- A. Ciftci, I. Demir and L. Yin, have aimed on feature extraction and then computing coherence and temporal consistence. The method extracted biological signals from facial regions from the fake and real video pair. An SVN and a CNN have been trained to find probabilities of authenticity.

- D. Guera and J. Delp, have used recognition pipeline to automatically detect deepfakes. They have proposed a two-step analysis. During first stage, features are extracted at frame level using CNN. The second stage consists of RNN which will capture erratic frames introduced due to face swapping process. The dataset contains 600 videos collected from various online sources was analysed.

- Y. Li, MC. Chang and S. Lyu, have introduced a new system of exposing deepfakes based on the eye blinking which are generated using neural networks. The paper focused on analyzing the eye blinking in the video as it is a natural signal and it cannot be presented well in the synthesized media. In the method the videos have been first preprocessed to locate face area in each frame, then a Long Term Recurrent Convolution Network(LRCN) find out temporal incongruity.

CHALLENGES AND LIMITATION IN EXISTING SYSTEM

- **Limited Temporal Understanding:**
CNN primarily focuses on spatial features in images and lacks built-in mechanisms for capturing temporal dependencies in video data. Deepfake detection often requires understanding the temporal context, which CNN may not handle optimally.

- **Large Computational Requirements:**
CNN architectures are deep and can be computationally expensive, especially when dealing with high-resolution video frames. This can pose challenges in real- time deepfake detection or applications with limited computational resources.

- **Vulnerability to Adversarial:**
CNN architectures, like many deep learning models, are susceptible to adversarial attacks. Adversarial examples specifically crafted to deceive the model could potentially lead to false negatives in deepfake detection.

- **Training Data Imbalances:**
If the training data is not balanced, with an over representation of certain types of deepfakes, CNN may be biased towards detecting those specific variations and may struggle with novel or less frequent types of manipulations.

PROPOSED WORK

The foundation of deepfake creation lies in an encoder- decoder architecture. Here, the encoder captures features from both the target and source faces, while the decoder reconstructs the target face by embedding these features into the source video. Although the final product can appear convincing, minute traces of manipulation remain, which can be detected through advanced methods. This paper proposes a model based on the InceptionResNeXtV2 architecture for feature extraction. The features obtained are passed to an LSTM network that analyzes the temporal patterns in video frames to detect whether a video has undergone manipulation.

METHODOLOGY

In this system, we have trained our PyTorch deepfake detection model on equal number of real and fake videos in order to avoid bias in the model. In the development stage, we curated and preprocessed a dataset, resulting in a newly processed dataset that contains only cropped face videos for more focused analysis.

A. Dataset

To enhance the model's effectiveness for real-time predictions, we compiled data from several available datasets,[9] including FaceForensic++, the Deepfake Detection Challenge (DFDC), and Celeb-DF. These datasets were combined to create a new dataset optimized for accurate and real-time detection of various types of videos. To avoid bias during training, the dataset was balanced with an equal proportion of 50The DFDC dataset also contains some audio-modified videos, which were beyond the scope of this study. We filtered out these audio-altered videos using a Python script. After preprocessing the DFDC dataset, we selected 1500 real and 1500 fake videos. Additionally, we included 1000 real and 1000 fake videos from FaceForensic++ (FF) and 500 real and 500 fake videos from the Celeb-DF dataset. In total, the dataset comprises 3000 real and 3000 fake videos, amounting to 6000 videos overall.

B. *Data Preprocessing*

Before training the model, several preprocessing steps were applied to the video data:

- **Frame Extraction:** Initially, videos were broken down into individual frames. This frame extraction is a key step in further analysis.
- **Face Detection and Cropping:** After extracting frames, a face detection algorithm was used to locate faces within each frame. The detected faces were then cropped to isolate only the relevant facial features, ensuring the model focuses on the most important aspects of the data.
- **Maintaining Uniform Frame Quantity:** To ensure consistency across the dataset, the mean number of frames for each video was computed. Based on this mean, a new dataset was created by retaining only the frames corresponding to this average, providing a uniform dataset for training and evaluation.
- **Frame Selection:** Any frames that lacked detectable faces were excluded from the preprocessing pipeline to ensure only the relevant frames with facial data were included.
- **Limiting Experimental Frames:** Given the computational cost of processing a 10-second video at 30 frames per second (resulting in 300 frames per video), we decided to limit the analysis to the first 100 frames for experimental purposes. This approach strikes a balance between managing computational resources and retaining sufficient data for training the model. In early experiments, selecting the first 140 frames was found to be an adequate starting point for model development.

C. *Dataset Split*

The dataset was divided into training and testing sets, with 70% of the videos (4,200) used for training and 30% (1,800) reserved for testing. Both the training and testing sets contain an equal distribution of real and fake videos, ensuring a balanced dataset in each split.

D. *Model*

The model is a combination of CNN and RNN architectures. We utilized a pre-trained ResNeXt CNN model to extract frame-level features. Based on these extracted features, an LSTM network was then trained to classify the videos as either deepfakes or real.

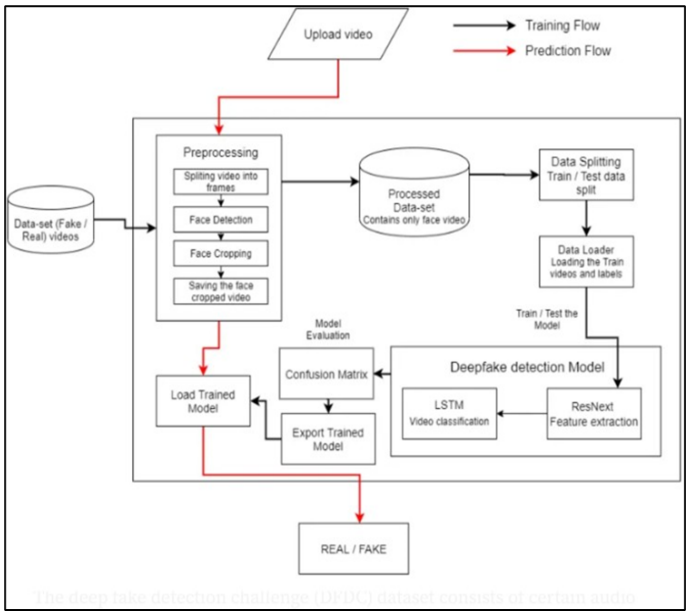

Fig. 4. System Architecture

During the training phase, video labels were loaded using a data loader and fed into the model.[7] Instead of developing the feature extraction from scratch, the pre-trained ResNeXt model was used, specifically the ResNeXt-50-32x4d architecture. This model, with 50 layers and dimensions of 32x4, is optimized for performance in deep neural networks. We fine- tuned the network by adding additional layers and selecting an appropriate learning rate to ensure the gradient descent converges properly. The 2048-dimensional feature vectors generated by the final pooling layers of ResNeXt were fed into the LSTM for sequence processing. LSTM for Sequential Processing the 2048-dimensional feature vectors from ResNeXt serve as input to the LSTM. The LSTM network, consisting of one layer with 2048 latent dimensions and 2048 hidden units, with a dropout rate of 0.4, was designed to meet the objective. The LSTM processes the frames in sequence, enabling the model to analyze temporal dynamics by comparing frames at different time points (e.g., comparing frame 't' with frame 't-n').

The model also includes a Leaky ReLU activation function. A linear layer with 2048 input features and 2 output features is utilized, allowing the model to learn the average correlation between inputs and outputs. An adaptive average pooling layer is used to target a specific output size (H x W) for image representation. The sequential layer enables processing frames in sequence, and a batch size of 4 is used for training. A SoftMax layer is employed to calculate the model's confidence during predictions.

*E. **Predict and Result***

During the prediction phase, when a new video is provided to the pre-trained model for classification, several preprocessing steps are applied to ensure the video format aligns with the trained model's requirements. This process starts by breaking the video down into individual frames. After that, facial regions are extracted from each frame through cropping. The model's output will be a binary classification, indicating whether the input video is genuine or fake, along- side a confidence score reflecting the model's certainty in its prediction. By analyzing the facial regions in the extracted frames, the model makes a well-informed decision regarding the authenticity of the video. It provides a summary of the prediction results on a set of test data for which the true values are known. This helps visualize the accuracy of a classification model by comparing the actual target values with the model's predicted values. The models accuracy is up to 85%

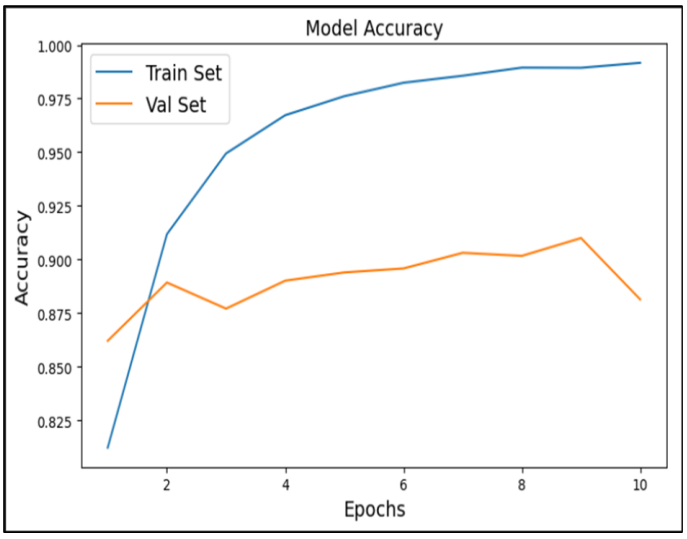

Fig. 5. Model Performance

*F. **Output***

The system's final output provides a classification that determines whether the uploaded video is authentic or a deepfake. This decision is based on the model's analysis of facial features and temporal inconsistencies.

G. Conclusion

The rise of deepfakes has eroded public trust, as video content no longer appears reliably authentic. In this research, we proposed an automatic deepfake detection model that leverages deep learning techniques. The model processes video frames through InceptionResNextV2 for feature extraction and LSTM for temporal analysis. This method demonstrated considerable accuracy and reliability in detecting manipulated media. Future research can build upon this foundation by exploring different architectures and detection techniques to further enhance deepfake detection accuracy.

In summary, the utilization of RNNs in deepfake detection represents a crucial step towards enhancing the reliability and efficacy of detection models. As the arms race between deepfake creators and detectors continues, the insights gained from temporal analysis through RNNs provide a valuable contribution to the ongoing efforts to mitigate the risks associated with synthetic media in today's digital landscape.

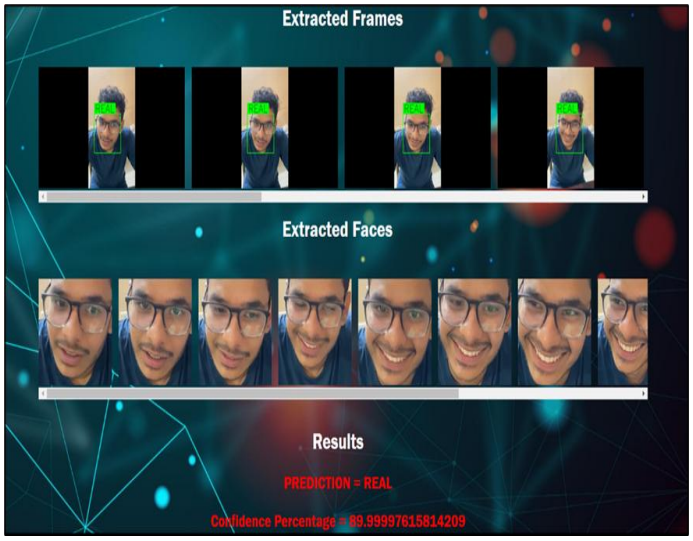

Fig. 6. Output

REFERENCES

1. Jadhav, A., Patange, A., Patil, H., Patel, J., & Mahajan, M. (2020). Deep residual learning for image recognition. *International Journal for Scientific Research and Development*, *8*, 1016–1019.
2. Li, Y., & Lyu, S. (2019). Exposing deepfake videos by detecting face warping artifacts. *arXiv preprint*. https://arxiv.org/abs/1811.00656
3. Liu, M.-Y., Huang, X., Yu, J., Wang, T.-C., & Mallya, A. (2021). Generative adversarial networks for image and video synthesis: Algorithms and applications. *Proceedings of the IEEE*, *109*(5), 839–862. https://doi.org/10.1109/JPROC.2021.3068755
4. Ciftci, U. A., Demir, I., & Yin, L. (2020). FakeCatcher: Detection of synthetic portrait videos using biological signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. https://doi.org/10.1109/TPAMI.2020.2995238
5. Güera, D., & Delp, E. J. (2018). Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (pp. 1–6). IEEE. https://doi.org/10.1109/AVSS.2018.8639163
6. Li, Y., Chang, M.-C., & Lyu, S. (2018). In ictu oculi: Exposing AI-created fake videos by detecting eye blinking. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)* (pp. 1–7). IEEE. https://doi.org/10.1109/WIFS.2018.8630787
7. Raghavendra, R., Raja, K. B., Venkatesh, S., & Busch, C. (2017). Transferable deep-CNN features for detecting

digital and print-scanned morphed face images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 1822–1830). IEEE. https://doi.org/10.1109/CVPRW.2017.232

8. Singh, R. K., Sarda, P. V., Aggarwal, S., & Vishwakarma, D. K. (2021). Demystifying deepfakes using deep learning. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 1290–1298). IEEE. https://doi.org/10.1109/ICCMC51019.2021.9418351
9. Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., & Ferrer, C. (2020). The Deepfake Detection Challenge (DFDC) dataset. *arXiv preprint*. https://arxiv.org/abs/2006.07397
10. General architecture of CNN [Figure]. (n.d.). *ResearchGate*. https://www.researchgate.net/figure/General-Architecture-of-CNN_fig2_351172173
11. ResNeXt building block [Figure]. (n.d.). *ResearchGate*. https://www.researchgate.net/figure/ResNext-building-block_fig6_330511306
12. Mohitv. (n.d.). LSTM networks. *Medium*. https://mohitv.medium.com/lstm-networks-75d44ac8280f