International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net

Vol-20 Issue-01 April 2025

Deep Learning-based social media Trend Analyzer to Predict the Trends Over Time

Sarasani Veda Reddy[1], Peechara Sai Mani Teja [2], Vallabhaneni Dhrushya Sree[3],Mr. Nagaraju[4]

[1,2,3] UG Scholar, Dept. of CSE(AI&ML),
St. Martin’s Engineering College, Secunderabad, Telangana, India, 500100
[4]Assistant Professor, Dept. of CSE(AI&ML), St. Martin’s Engineering College, Secunderabad, Telangana, India, 500100
sarasanivedareddy@gmail.com

Abstract:

Social media platforms like Twitter generate over 500 million tweets per day, with a 72% increase in global user engagement since 2019. Analyzing and predicting trends over time remains a significant challenge due to the dynamic nature of user interactions and content generation. Predicting what content will become a trend remains a complex task. With millions of tweets posted every day, identifying potential trends manually is nearly impossible. The dynamic and fast-paced nature of Twitter interactions makes it difficult to rely on traditional methods like keyword analysis or manual monitoring. As more companies and influencers seek to capitalize on trending topics for engagement, a reliable automated system for predicting trends over time is essential. This research proposes a novel Deep Learning (DL) classification model that predicts trends based on real-time Twitter data. The preprocessing pipeline includes text cleaning, tokenization, and feature extraction, followed by trend detection and classification. The proposed model aims to classify whether a specific topic will trend or not, providing valuable insights for marketers and content creators.

Keywords: Social media , Twitter, tweets, global user, dynamic, keyword analysis, influencers, Capitalize, automated system, Deep learning classification model, text cleaning, tokenization.

1.INTRODUCTION

Before the adoption of Machine Learning (ML) techniques, predicting Twitter trends relied on manual monitoring, rule-based systems, and keyword analysis, which posed multiple challenges. The sheer volume and velocity of tweets made manual trend identification impractical and time-consuming. Traditional statistical methods struggled with context understanding, making them ineffective in detecting emerging trends that rely on sentiment, slang, and evolving language patterns. Additionally, keyword-based methods often resulted in false positives, as a high frequency of words does not always indicate a trend. Furthermore, these approaches lacked adaptability to real-time interactions, missing the nuances of hashtags, retweets, and influencer-driven discussions. Businesses and digital marketers found it difficult to anticipate viral content, leading to missed engagement opportunities. The absence of an automated predictive model also affected crisis management, where early detection of viral damage. Given these limitations, there is a pressing need for a data-

2. LITERATURE SURVEY

Saqib Hakak et. al [1] proposed a machine-learning based fake news detection model using a supervised approach. They used the ensemble approach for training and testing purposes consisting of decision tree, random forest, and extra tree classifiers. The aggregation of outputs was done using the bagging approach and compared to the state-of-the-art, our model achieved better results.

Kaliyar et. al [2] propose a BERT-based (Bidirectional Encoder Representations from Transformers) deep learning approach (FakeBERT) by combining different parallel blocks of the single-layer deep Convolutional Neural Network (CNN) having different kernel sizes and filters with the BERT. Such a combination is useful to handle ambiguity, which is the greatest challenge to natural language understanding. Classification results demonstrate that our proposed model (FakeBERT) outperforms the existing models with an accuracy of 98.90%. Somya Ranjan Sahoo et. al [3] proposed a fake news detection approach for Facebook users using machine learning and deep learning classifiers in chrome environment. Our approach analyses both user profile and news content features. In this proposed work, they have developed a chrome extension that uses crawled data extracted by our crawler. Also, to boost up the performance of chrome extension, they have used deep learning algorithm called Long Short-Term Memory.

Anshika Choudhary et. al [4] proposed a solution to fake news detection and classification. In the case of fake news, content is the prime entity that captures the human mind towards trust for specific news. Therefore, a linguistic model is proposed to find out the properties of content that will generate language-driven features. This linguistic model extracts syntactic, grammatical, sentimental, and readability features of particular news. Language driven model requires an approach to handle timeconsuming and handcrafted features problems in order to deal with the curse of dimensionality problem. Therefore, the neural-based sequential learning model is used to achieve superior results for fake news detection. The results are drawn to validate the importance of the linguistic model extracted features and finally combined linguistic feature-driven model is able to achieve the average accuracy of 86% for fake news detection and classification. The sequential neural model

International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net

Vol-20 Issue-01 April 2025

results are compared with machine learning based models and LSTM based word embedding based fake news detection model as well. Comparative results show that features based sequential model is able to achieve comparable evaluation performance in discernable less time.

Aphiwongsophon et. al [5] proposes the use of machine learning techniques to detect Fake news. Three popular methods are used in the experiments: Naive Bayes, Neural Network and Support Vector Machine. The normalization method is important step for cleaning data before using the machine learning method to classify data. The result show that Naive Bayes to detect Fake news has accuracy 96.08%. Two other more advance methods which are Neural Network and Support Vector Machine achieve the accuracy of 99.90%.

Georgios Gravanis et. al [6] proposed a model for fake news detection using content-based features and Machine Learning (ML) algorithms. To conclude in most accurate model, they evaluate several feature sets proposed for deception detection and word embeddings as well. Moreover, they test the most popular ML classifiers and investigate the possible improvement reached under ensemble ML methods such as AdaBoost and Bagging. An extensive set of earlier data sources has been used for experimentation and evaluation of both feature sets and ML classifiers. Moreover, they introduce a new text corpus, the "UNBiased" (UNB) dataset, which integrates various news sources and fulfills 142 several standards and rules to avoid biased results in classification task. Our experimental results show that the use of an enhanced linguistic feature set with word embeddings along with ensemble algorithms and Support Vector Machines (SVMs) is capable to classify fake news with high accuracy.

Umer et. al [7] proposed to employ the dimensionality reduction techniques to reduce the dimensionality of the feature vectors before passing them to the classifier. To develop the reasoning, this work acquired a dataset from the Fake News Challenges (FNC) website which has four types of stances: agree, disagree, discuss, and unrelated. The nonlinear features are fed to PCA and chi-square which provides more contextual features for fake news detection. The motivation of this research is to determine the relative stance of a news article towards its headline. The proposed model improves results by ~4% and ~20% in terms of Accuracy and F1−score. The experimental results show that PCA outperforms than Chi-square and state-of-the-art methods with 97.8% accuracy. Vasu Agarwal et. al [8] discusses the approach of natural language processing and machine learning in order to solve this problem. Use of bag-of-words, n-grams, count vectorizer has been made, TFIDF, and trained the data on five classifiers to investigate which of them works well for this specific dataset of labelled news statements. The precision, recall and f1 scores help us determine which model works best.

Junaed Younus Khan et. al [9] presented an overall performance analysis of 19 different machine learning approaches on three different datasets. Eight out of the 19 models are traditional learning models, six models are traditional deep learning models, and five models are advanced pre-trained language models like BERT. They find that BERT-based models have achieved better performance than all other models on all datasets. More importantly, we find that pre-trained BERT-based models are robust to the size of the dataset and can perform significantly better on very small sample size. They also find that Naive Bayes with n-gram can attain similar results to neural network-based models on a dataset when the dataset size is sufficient. The performance of LSTM-based models greatly depends on the length of the dataset as well as the information given in a news article. With adequate information provided in a news article, LSTM-based models have a higher probability of overcoming overfitting.

Reis et. al [10] presented a new set of features and measure the prediction performance of current approaches and features for automatic detection of fake news. Our results reveal interesting findings on the usefulness and importance of features for detecting false news. Finally, they discuss how fake news detection approaches can be used in the practice, highlighting challenges and opportunities.

Agarwal et. al [11] proposed a deep learning model which predicts the nature of an article when given as an input. It solely uses text processing and is insensitive to history and credibility of the author or the source. In this paper, authors have discussed and experimented using word embedding (GloVe) for text pre-processing in order to construct a vector space of words and establish a lingual relationship. The proposed model which is the blend of convolutional neural network and recurrent neural networks architecture has achieved benchmark results in fake news prediction, with the utility of word embeddings complementing the model altogether. Further, to ensure the quality of prediction, various model parameters have been tuned and recorded for the best results possible. Among other variations, addition of dropout layer reduces overfitting in the model, hence generating significantly higher accuracy values. It can be a better solution than already existing ones, viz: gated recurrent units, recurrent neural networks or feed-forward networks for the given problem, which generates better precision values of 97.21% while considering more input features.

3. PROPOSED METHODOLOGY

This project implements a Graphical User Interface (GUI) application for detecting fake news using machine learning techniques, specifically utilizing a combination of LSTM (Long Short-Term Memory) neural networks and Random Forest classifiers. Here's an overview of the project:

- GUI Development: The project utilizes Tkinter, a Python library for creating GUI applications, to build a user-friendly interface for interacting with the fake news detection system.
- Functionality:
- Dataset Handling: Users can upload a dataset containing news articles labeled as genuine or fake.

- Preprocessing: The uploaded dataset undergoes preprocessing, including text cleaning, tokenization, and splitting into training and testing sets.
- Model Training and Evaluation: The LSTM model is trained on the preprocessed data, and its performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
- Prediction: Users can test the trained models by uploading a test dataset, preprocessing it, and making predictions on the news articles using the LSTM and Random Forest models.
- Preprocessing Steps: Text cleaning involves removing punctuation, stop words, and lemmatization to prepare the text data for modeling. Tokenization converts the cleaned text into numerical sequences, which can be fed into the LSTM model. The dataset is split into training and testing sets for model training and evaluation.
- Model Architecture: The LSTM model is constructed using Keras Sequential API, with bidirectional LSTM layers followed by dense layers for classification. Features extracted from the LSTM model are used to train a Random Forest classifier for improved prediction performance.
- Model Persistence: Trained LSTM and Random Forest models are saved to disk using serialization techniques like HDF5 and joblib to avoid retraining the models on subsequent runs.
- Results Display: The GUI displays various evaluation metrics such as accuracy, precision, recall, and F1-score, along with a confusion matrix and classification report to assess the model's performance.
- User Interaction: Users can interact with the application through buttons for uploading datasets, preprocessing, running the LSTM algorithm, and testing news detection.
- Visualization: The application provides a visual representation of the confusion matrix using heatmaps generated by Matplotlib and Seaborn libraries.

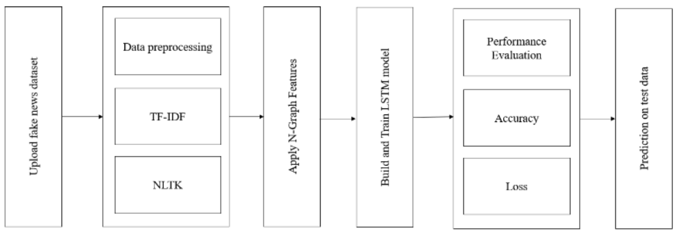


Fig. 4.1: Block diagram of proposed system.

4.2 TF-IDF Feature extraction

TF-IDF which stands for Term Frequency – Inverse Document Frequency. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document. we have a string or Bag of Words (BOW) and we have to extract information from it, then we can use this approach. The tf-idf value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. TF-IDF use two statistical methods, first is Term Frequency and the other is Inverse Document Frequency. Term frequency refers to the total number of times a given term t appears in the document doc against (per) the total number of all words in the document and The inverse document frequency measure of how much information the word provides. It measures the weight of a given word in the entire document. IDF show how common or rare a given word is across all documents.

TF-IDF can be computed as tf * idf

TF-IDF do not convert directly raw data into useful features. Firstly, it converts raw strings or dataset into vectors and each word has its own vector. Then we'll use a particular technique for retrieving the feature like Cosine Similarity which works on vectors, etc.

Terminology

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

Step 1: Term Frequency (TF): Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, “Data Science is awesome!” A simple way to start out is by eliminating documents that do not contain all three words “Data” is”, “Science”, and “awesome”, but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. The weight of a term that occurs in a document is simply proportional to the term frequency.

Step 2: Document Frequency: This measures the importance of document in whole set of corpora, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, whereas DF is the count of occurrences of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

Step 3: Inverse Document Frequency (IDF): While computing TF, all terms are considered equally important. However, it is known that certain terms, such as “is”, “of”, and “that”, may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. The IDF is the inverse of the document frequency which measures the 145 informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as “is” is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage. Now there are few other problems with the IDF, in case of a large corpus, say 100,000,000 , the IDF value explodes , to avoid the effect we take the log of idf . During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator. The TF-IDF now is at the right measure to evaluate how important a word is to a document in a collection or corpus. Here are many different variations of TF-IDF but for now let us concentrate on this basic version.

Step 4: Implementing TF-IDF: To make TF-IDF from scratch in python, let’s imagine those two sentences from different document:

first_sentence: “Data Science is the sexiest job of the 21st century”.

second_sentence: “machine learning is the key for data science”.

4.3 Natural Language Toolkit

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenization, lower case conversion, Stop Words removal, stemming, and lemmatization.

Tokenization

The breaking down of text into smaller units is called tokens. tokens are a small part of that text. If we have a sentence, the idea is to separate each word and build a vocabulary such that we can represent all words uniquely in a list. Numbers, words, etc. all fall under tokens.

Lower case conversion

We want our model to not get confused by seeing the same word with different cases like one starting with capital and one without and interpret both differently. So we convert all words into the lower case to avoid redundancy in the token list.

Stop Words removal

When we use the features from a text to model, we will encounter a lot of noise. These are the stop words like the, he, her, etc… which don’t help us and just be removed before processing for cleaner processing inside the model. With NLTK we can see all the stop words available in the English language.

Stemming

In our text we may find many words like playing, played, playfully, etc… which have a root word, play all of these convey the same meaning. So we can just extract the root word and remove the rest. 146 Here the root word formed is called ‘stem’ and it is not necessarily that stem needs to exist and have a meaning. Just by committing the suffix and prefix, we generate the stems.

Lemmatization

We want to extract the base form of the word here. The word extracted here is called Lemma and it is available in the dictionary. We have the WordNet corpus and the lemma generated will be available in this corpus. NLTK provides us with the WordNet Lemmatizer that makes use of the WordNet Database to lookup lemmas of words.

4.4 LSTM Network

Deep learning is a new research direction in the field of artificial intelligence. It is developed on the basis of shallow neural networks with the improvement of computer hardware levels and the explosive growth of the current data volume. Deep learning and shallow neural network structure are both layered. Each layer will process the data input to the model and combine low-level features into potential high-level features by learning data rules. Compared with shallow models, deep learning can express complex high dimensionality such as high-variable functions and find the true relationships within the original data better. In the 1980s, artificial neural network back propagation algorithm was born. This method can automatically learn data rules from a large amount of training data without manual intervention. At present, deep learning is the most concerned research direction in the field of artificial intelligence, which completely subverts the shallow model in traditional machine, proposes a deep learning network model, and elevates it to a new height from theory to application. CNN (convolutional neural network) and RNN are two types of classical deep learning network structures now.

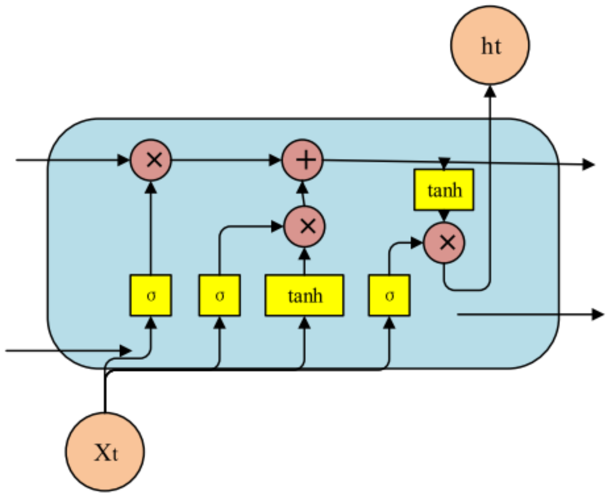


Fig. 4.2: LSTM model structure.

Because there are connections between neurons in the RNN layer, the network can learn the change law of sequence data before and after, and the internal sequence rules of data is easy to be mined. Thus RNN is widely used in the field of sequence data processing such as speech recognition and machine translation. However, this structure also has some problems. When data is transmitted backward, the problem of gradient disappearance or gradient explosion is unavoidable, which limits its processing of long-term dependencies. The LSTM network changes the way of gradient transmission during backpropagation by adding multiple special computing nodes in the hidden layer of RNN, which effectively slows the problem of gradient disappearance or gradient explosion. Its model structure is shown in figure 4.

Where h_{t-1} represents the output of the previous cell, and x_t represents the input of the current cell. σ represents the sigmoid function. The difference between LSTM and RNN is that it adds a "processor" to the algorithm to determine the usefulness of the information. The structure of this processor is called a cell. Three gates are placed in a cell, which are called *Input gate*, *Forget gate*, and *Output gate*. A piece of information enters the LSTM network, and it can be judged whether it is useful according to the rules. Only the information that meets the algorithm authentication will be left, and the non-conforming information will be forgotten through the *Forget gate*.

Forget Gate

The first step for data entering the LSTM is to decide what information should be lost and what retained. This decision is made by the Forget gate, which reads h and x and outputs a value between 0 and 1, where 1 means "complete reserved", 0 means "completely discarded". Forget gate is calculated as:

$$f_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_f\right)$$

In the formula, f_t is the calculation result of the Forget gate which is mainly used to control the retention of the information transmitted from the unit state at the previous moment to the unit state at the current moment. [ ] indicates that the two vectors are spliced, h_{t-1} is the output of the unit at the previous moment, and are the weight and bias of Forget gate, W_f and b_f are Sigmoid activation functions.

Input Gate

International journal of imaging science and engineering
ISSN: 1934--9955 www.ijise.net
Vol-20 Issue-01 April 2025

Input gate determines the addition of new information, and its operation process includes sigmoid layer and tanh layer. The sigmoid layer determines the information that needs to be updated. The calculation formula is:

$$i_t = \sigma\left(W_i * [h_{t-1}, x_t] + b_i\right)$$

In the formula, i_t is the calculation result of the input gate, and the input gate also has independent weight and bias. The role of the tanh layer is to generate a vector of candidate update information. Its calculation formula is:

$$\tilde{C}_t = \tanh\left(W_c * [h_{t-1}, x_t] + b_c\right)$$

$\tilde{C}_t$is the unit state of the current input, the unit state of the current moment is C_t, and its calculation formula is:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate

Output gate is roughly the same as the Input gate, and its operation flow includes sigmoid layer and tanh layer. The sigmoid layer determines the output part of the information, and the calculation formula is:

$$o_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right)$$

Finally get the output of the current moment h_t:

$$h_t = o_t * \tanh\left(c_t\right)$$

The forward propagation of LSTM calculates the cell state C_t and h_t the output of the current moment and completes the forward propagation calculation of the network. The backpropagation of LSTM is like the back-propagation principle of RNN. Finally, the weights and biases of all parts of the network are updated to complete the model training.

Advantages

Long Short-Term Memory (LSTM) models offer several advantages in for time series analysis of residential electrical power consumption:

— Sequential Data Handling: LSTMs are specifically designed to handle sequential data, making them well-suited for time series analysis. In the project, LSTMs can capture the temporal dependencies and patterns in power consumption data more effectively compared to traditional machine learning models.
— Long-Term Dependencies: LSTMs are capable of capturing long-term dependencies in time series data. They can learn from historical data over extended periods, which is crucial for understanding seasonal or yearly patterns in residential power consumption. This capability helps in making more accurate predictions.
— Variable Sequence Length: LSTMs can accommodate variable-length sequences, making them adaptable to datasets where the number of time steps may vary. In the project, this flexibility allows the model to consider different look-back periods when making predictions.
— Feature Engineering: LSTMs do not require extensive feature engineering. They can automatically learn relevant features from the sequential data, reducing the need for manual feature selection and engineering, which can be time-consuming.
— Handling Noisy Data: LSTMs can handle noisy time series data effectively. They are robust to outliers and missing values, which is essential when working with real-world data that may have irregularities or data gaps.
— Parallel Processing: LSTMs can be trained efficiently on modern hardware with parallel processing capabilities, enabling faster training times. This is valuable when dealing with large datasets or complex models.
— Model Interpretability: While LSTMs are often considered "black-box" models, efforts can be made to interpret their internal workings. For instance, techniques such as attention mechanisms can be added to LSTMs to provide insights into which parts of the input data are most influential in making predictions.
— Transfer Learning: Pre-trained LSTM models can be leveraged for time series analysis. Transfer learning allows the use of models trained on similar data or domains, potentially reducing the amount of data required for training and improving model performance.
— Real-Time Predictions: LSTMs can be used for real-time predictions, making them valuable for applications where timely forecasting of power consumption is critical. This can aid in proactive load management and energy optimization.

International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net
Vol-20 Issue-01 April 2025

— Scalability: LSTMs can be scaled for more complex tasks. In this project, as more features or data sources become available (e.g., weather data), LSTMs can accommodate and incorporate these additional inputs to improve predictions.

4. EXPERIMENTAL ANALYSIS

Figure 1 shows a collection of original images that are taken in low-light conditions or have poor lighting quality. These images serve as the input to the proposed image enhancement model. These images are the input images that the model will process in order to improve their visibility and quality. The purpose of this figure is to provide a visual representation of the types of images that the model is designed to enhance.

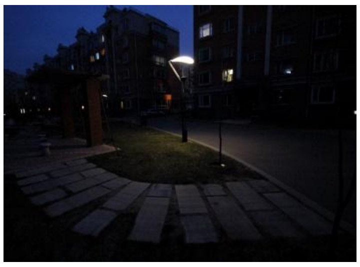


Figure 2: Sample Images

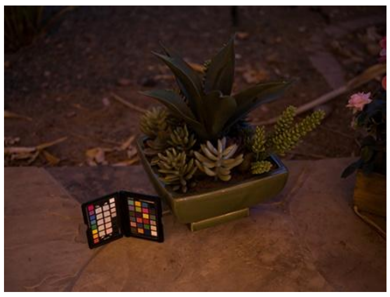
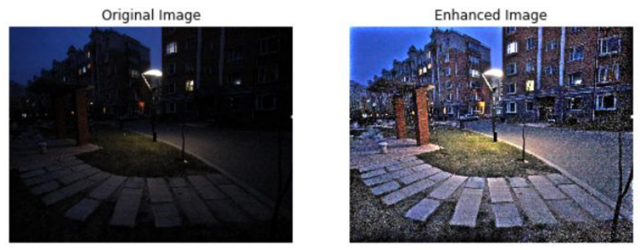
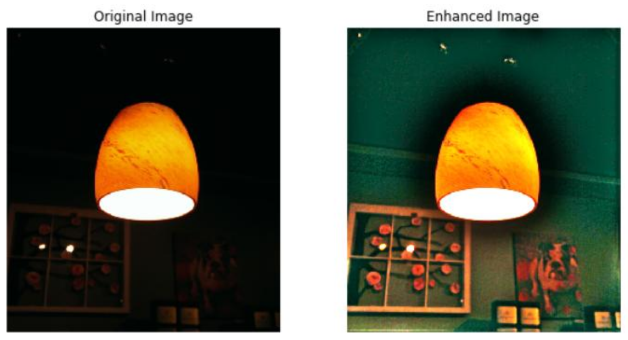


Figure3: Enhanced Image 1

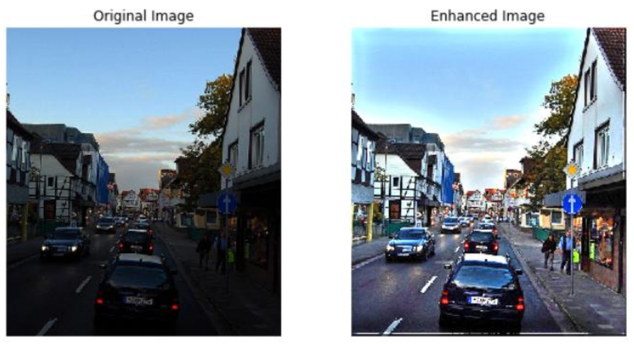


```
PSNR 10.171815771384654
SSIM 0.18386150146633054
MSE 1.0857190890301478
```

Figure 4: Enhanced Image 2



```
PSNR 13.747368386518946
SSIM 0.3436245339679396
MSE 0.9086185481481482
```



```
PSNR 11.031691901461375
SSIM 0.5199471454508887
MSE 1.0771644632584392
```

Figure 5: Enhanced Image 3

Figure 2 displays a set of images that have been processed or enhanced by the proposed image enhancement model. These are the output images that produces improved visibility and quality of these images compared to the original low-light images shown in Figure 1These metrics are numerical values that provide insights into the image quality, with higher PSNR and SSIM values and lower MSE values indicating better image quality. The purpose of this figure is to visually demonstrate the effectiveness of the proposed image enhancement model by showing the enhanced images and providing quantitative metrics that measure the improvement in image quality.

PSNR

- The peak_signal_noise_ratio function calculates the Peak Signal-to-Noise Ratio, which is a widely used metric to measure the quality of an image.
- It compares two images, typically the original and the enhanced image, and computes a value that indicates how much noise or distortion is present relative to the maximum possible quality.
- The result is a numerical value, often in decibels (dB). Higher PSNR values indicate higher image quality.

5. CONCLUSION

This groundbreaking work signifies a substantial leap forward in the realm of image processing and computer vision. LIME, with its unwavering focus on the formidable task of improving images captured in low-light environments, presents a formidable solution that not only enhances image quality but also enhances visibility. By harnessing the power of deep learning methodologies, this initiative adeptly tackles prevalent issues encountered in low-light image processing, including noise reduction, addressing insufficient contrast, and preserving crucial details that might otherwise be lost.

A key highlight lies in the remarkable versatility and adaptability exhibited by LIME. This innovative solution grants users the flexibility to finely adjust enhancement parameters, ensuring that the resultant output precisely aligns with specific requirements and individual preferences. Furthermore, the incorporation of robust quality assessment metrics such as PSNR, SSIM, and MSE facilitates a quantitative evaluation of the efficacy of the enhancement process. This meticulous approach guarantees that the enhanced images not only possess visual appeal but also uphold or even surpass the quality standards set by their original counterparts.

The far-reaching impact of the LIME project transcends disciplinary boundaries, finding resonance across a myriad of domains. In the realm of surveillance, where bolstering nighttime video quality holds paramount significance for ensuring security, LIME emerges as an invaluable tool. Similarly, in the field of astronomy, this groundbreaking initiative aids in capturing the intricate nuances of celestial bodies, such as stars and galaxies, under challenging lighting conditions. Moreover, within the realm of consumer photography, the project serves as a beacon of innovation by enhancing smartphone camera performance, particularly in dimly lit environments, thus empowering users to capture high-quality photos even amidst adverse lighting conditions, thereby enriching their photographic experiences.

While LIME has achieved significant success, there are several promising avenues for future research and development. First and foremost, optimizing the algorithm for real-time processing is a priority, especially for applications like live video enhancement, where speed is critical. Developing adaptive algorithms that can automatically adjust enhancement parameters based on image content and lighting conditions could enhance user experience and convenience.

REFERENCES

[1] A. Nematzadeh, E. Ferrara, A. Flammini, and Y. Y. Ahn, “Optimal network modularity for information diffusion,” Phys. Rev. Lett., vol. 113, no. 8, p. 088701, 2014.

[2] J. Firmstone and S. Coleman, “The cha nging role of the local news media in enabling citizens to engage in local democracies,” Journalism Pract., vol. 8, no. 5, pp. 596–606, 2014.

[3] R. A. Hackett, “Decline of a paradigm? Bias and objectivity in news media studies,” Crit. Stud. Media Commun., vol. 1, no. 3, pp. 229–259, 1984.

[4] S. Della Vigna and E. Kaplan, “The Fox News effect: Media bias and voting,” Quart. J. Econ., vol. 122, no. 3, pp. 1187–1234, 2007.

[5] M. Gentzkow and J. M. Shapiro, “Media bias and reputation,” J. Political Economy, vol. 114, no. 2, pp. 280–316, 2006.

[6] M. Karsai et al., “Small but slow world: How network topology and burstiness slow down spreading,” Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 83, no. 2, p. 025102, 2011.

[7] F. D. Gilliam, Jr., and S. Iyengar, “Prime suspects: The influence of local television news on the viewing public,” Amer. J. Political Sci., vol. 44, no. 3, pp. 560–573, 2000.

[8] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks,” Proc. Nat. Acad. Sci. USA, vol. 99, no. 12, pp. 7821–7826, Apr. 2002.

[9] J. Yang and J. Leskovec, “Overlapping community detection at scale: A nonnegative matrix factorization approach,” in Proc. 6th ACM Int. Conf. Web Search Data Mining, 2013, pp. 587–596.

[10] S. Ji, N. Satish, S. Li, and P. Dubey. (2016). “Parallelizing word2vec in shared and distributed memory.” [Online]. Available: https://arxiv.org/abs/1604.04661?context=stat.ML [11] T. Mikolov, I. C. K. Sutskever, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

[12] A. Mnih and Y. W. Teh. (2012). “A fast and simple algorithm for training neural probabilistic language models.” [Online]. Available: https://arxiv.org/abs/1206.6426

[13] B. Recht, C. Re, S. Wright, and F. Niu, “HOGWILD: A lock-free approach to parallelizing stochastic gradient descent,” in Proc. Adv. Neural Inf. Process. Syst., 2011, pp. 693–701.

[14] R. G. Miller, Jr., Survival Analysis, vol. 66, Hoboken, NJ, USA: Wiley, 2011.

[15] A.-L. Barabási. (2005). “The origin of bursts and heavy tails in human dynamics.” [Online]. Available: https://arxiv.org/abs/cond-mat/0505371

[16] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2014, pp. 701–710.

[17] D. Kempe, J. Kleinberg, and E. Tardos, “Maximizing the spread of influence through a social network,” in Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2003, pp. 137–146.

[18] E. Mossel, J. Neeman, and A. Sly. (2012). “Stochastic block models and reconstruction.” [Online]. Available: https://arxiv.org/abs/1202.1499

[19] R. M. Gomez, J. Leskovec, and B. Schölkopf, “Structure and dynamics of information pathways in online media,” in Proc. 6th ACM Int. Conf. Web Search Data Mining, 2013, pp. 23–32.

[20] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance,” J. Mach. Learn. Res., vol. 11, pp. 2837–2854, Jan. 2010.