A Project Report on

DETECTION OF CYBERBULLYING IN TEXTS, IMAGES, AND AUDIO

in Partial Fulfilment for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

N S N V R SRI SAKETH	**19B91A12C0**
N N V DURGA SANDEEP	**19B91A12C1**
M VENKATA SAI TEJA	**19B91A12A8**
K LOKESH VARMA	**19B91A1294**

Under the esteemed guidance of

K. SATYANARAYANA RAJU

Assistant professor

DEPARTMENT OF INFORMATION TECHNOLOGY

S.R.K.R. ENGINEERING COLLEGE (AUTONOMOUS)

(Approved by AICTE, New Delhi, Affiliated to JNTUniversity, Kakinada)

CHINNA AMIRAM :: BHIMAVARAM-534202

[2022-2023]

DEPARTMENT OF INFORMATION TECHNOLOGY

S.R.K.R. ENGINEERING COLLEGE (A)

(Approved by AICTE, New Delhi, Affiliated to JNTUniversity, Kakinada)

CHINNA AMIRAM :: BHIMAVARAM-534202

[2022-2023]

BONAFIDE CERTIFICATE

This is to certify that the Project Work entitled "Detection of Cyberbullying in Texts, Images and Audio", is bonafide work of **N S N V R S SAKETH (RegdNo:19B91A12C0), N N V DURGA SANDEEP (RegdNo:19B91A12C1), M VENKATA SAI TEJA (RegdNo:19B91A12A8), K LOKESH VARMA (RegdNo:19B91A1294)** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology during the Academic year 2022-2023.

HEAD OF THE DEPARTMENT

Dr. Bh.V.S. RamaKrishnam Raju

Professor, HOD-IT Dept

SUPERVISOR

K Satyanarayana Raju

Assistant Professor

CERTIFICATION OF EXAMINATION

This to certify that I have examined the concept and hereby accord my approval of it as a project entitled "DETECTION OF CYBERBULLYING IN TEXTS, IMAGES AND AUDIO" carried out and presented in a manner required for its acceptance on partial fulfilment for the award of the degree of BACHELOR OF TECHNOLOGY for which it has been submitted.

This approval does not necessarily endorse or accept every statement made opinion expressed or conclusions drawn as recorded in the project report it only signifies the acceptance of the report for the purpose of which submitted.

PROJECT GUIDE :

EXTERNAL EXAMINER :

HOD :

DECLARATION

We hereby declare that the project work entitled "DETECTION OF CYBER BULLYING IN TEXTS, IMAGES, AND AUDIO" is a genuine work carried out by us in B.Tech., (Information Technology) at SRKR Engineering College(A), Bhimavaram and has not been submitted either in part or full for the award of any other degree or diploma in any other institute or University.

NARAYANA S N VENKATA RAMA SRI SAKETH (19B91A12C0)

NARSIPUDI NAGA VENKATA DURGA SANDEEP (19B91A12C1)

MOTHUKURI VENKATA SAI TEJA (19B91A12A8)

KUNAPARAJU LOKESH VARMA (19B91A1294)

ACKNOWLEDGEMENT

The victorious completion of this project would be incomplete without greeting those who made it possible and whose guidance and encouragement made efforts that taken to the success.

The thesis presented here is the work accomplished under the enviable and scholarly guidance of "**K. Satyanarayana Raju**", Assistant Professor of Information Technology Department. We are grateful towards the indomitable support. We express deep gratitude for his inspiring remarks and helping this project.

We take this opportunity to express our sincere thanks to "**Dr.BH.V.S. Rama Krishnam Raju**", Head of the Department, Information Technology for his support and encouragement.

We express our sincere thanks to "**Dr. M. Jagapathi Raju**", PRINCIPAL, S.R.K.R Engineering College, Bhimavaram for giving us this opportunity for the successful completion of this project.

Last but not the least, we also express our sincere thanks to other teaching and non-teaching staff for their support in accomplishing this project.

NARAYANA S N VENKATA RAMA SRI SAKETH (19B91A12C0)

NARSIPUDI NAGA VENKATA DURGA SANDEEP (19B91A12C1)

MOTHUKURI VENKATA SAI TEJA (19B91A12A8)

KUNAPARAJU LOKESH VARMA (19B91A1294)

ABSTRACT

Cyberbullying has become a prevalent issue in today's social media landscape, where it can manifest in various forms, including text, video, or audio. Such abusive content can have severe personal consequences for the targeted user, and detecting these types of messages is a challenging task for traditional algorithms. In this project, we propose an ensemble algorithm that combines Bidirectional Long Short-Term Memory (Bi-LSTM) with Convolutional Neural Network (CNN) to detect cyberbullying messages. The Bi-LSTM model provides an understanding of the context and semantics of the text data, while the CNN model extracts relevant features from the image data. Our proposed algorithm addresses the limitations of traditional algorithms by providing a comprehensive approach that can detect cyberbullying messages from both textual and image data and audio file as well.

To evaluate the performance of our proposed algorithm, we conducted experiments on a dataset of 2000 images, which were classified as either cyberbullying or non-cyberbullying. The dataset was split into training and testing sets, and the performance of the algorithm was measured in terms of accuracy, precision, and recall. The experimental results show that our proposed ensemble algorithm achieved higher accuracy and precision than the traditional algorithms. Additionally, the proposed algorithm provided a better recall rate for identifying cyberbullying messages.

Our findings suggest that an ensemble approach combining Bi-LSTM and CNN models can be an effective solution for detecting cyberbullying messages from both textual and image data. The proposed algorithm can help social media platforms and law enforcement agencies identify and mitigate cyberbullying incidents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABELS

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION ABOUT THE PROBLEM

In today's society, cyberbullying has become a prevalent and growing issue. Cyberbullying refers to the use of computers to harass, humiliate, or threaten another person. Social media platforms, chat rooms, and messaging apps have given individuals new and powerful ways to interact and communicate, but they have also created new opportunities for people to engage in abusive conduct towards others. Cyberbullying can have devastating effects on its victims, including depression, anxiety, low self-esteem, and in some cases, suicide.

There is a pressing need for improved detection and prevention strategies for cyberbullying. Manual methods of detecting bullying, such as interviewing victims and witnesses, are time-consuming and not always reliable. However, technological advancements offer new potential for automatic detection of cyberbullying.

While many experts have worked on cyberbullying detection based on literary components, there is a need to consider other elements or combinations to improve detection. The available datasets do not provide any information regarding the severity of the harassment. If such information were available, cyberbullying detection models could be implemented to take different actions depending on the seriousness of posts. There is also a lack of research that considers the profile of online media users for feature extraction. This would be helpful in determining the user's behaviour to achieve accurate classification.

Despite being complex, Deep Neural Networks (DNN) have been shown to be the technology that provides better performance. As sending images and videos is becoming popular among youth, image/video processing is another important area for cyberbullying detection. Using Deep Neural Networks, accurate video cyberbullying detection can be achieved, which is yet an overlooked area by social media platforms.

Cyber criminals are using Online Social Networks (OSN) as a new platform to commit various types of cybercrimes, such as phishing, spamming, spreading malware, and cyberbullying. Specifically, from recent developments, cyberbullying has become more complicated for communication and OSN. Cyberbullying can be used in communication between individuals

and between groups. This is considered the most dangerous aspect of cyberbullying worldwide, as it may cause personal damage to users.

By using OSN as an online platform, individuals can harass male/female within the specific OSN platform. Over the past many years, OSN platforms such as Facebook and Twitter provide more and more features to the users to attract and make them register in their platforms. These websites become popular for the users, and cyberbullying becomes more common on these platforms. Therefore, there is a need for improved detection and prevention strategies for cyberbullying in OSNs.

This project aims to address the problem of cyberbullying detection by proposing an Ensemble Algorithm that combines Bidirectional-LSTM model with Convolutional Neural Network (CNN). The performance of the proposed algorithm is evaluated using Accuracy, precision, and recall. The project also includes the experimental setup and results, as well as a discussion and conclusion of the findings.

1.2 INTRODUCTION ABOUT THE EXISITING SYSTEM

Investigating the Twitter information is more difficult because of its unstructured nature, restricted size, utilization of slangs, and incorrectly spells and shortenings. This exploration centres around discovering feelings from Twitter information. The greater part of the specialists managed different ML approaches of feeling investigation, and just arranged opinions as good and regrettable feelings. A conduct is all that an individual does. Dissecting human conduct affects social impact. Examination of client conduct investigation is a significant exploration region in computational informal organization for different reasons. ID of crimes, inconsistency discovery, data dissemination in brief period, feeling examination of gatherings of people, notoriety assessment, item appraising has a place with conduct investigation in informal organizations. A large portion of the informal communities are dynamic in nature, these postures huge difficulties in portrayal of definite conduct. There exist countless informal community client conduct examination strategies with exceptional accentuation on conduct forecast. By breaking down different informal community informational indexes and by seeing past exercises. The Existing Naïve Bayes (NB), which is computationally exceptionally productive and simple to carry out, is a learning calculation regularly utilized in text order

issues. Two occasion models are regularly utilized: Multivariate Bernoulli Event Model, Multivariate Event Model. The Multivariate Event model is alluded to as Multinomial NB.

1.3 INTRODUCTION ABOUT PROPOSED SOLUTION

Our proposed solution for detecting cyberbullying messages is an ensemble algorithm that combines the power of Bidirectional-LSTM (BLSTM) model with Convolutional Neural Network (CNN). Traditional algorithms have been struggling to detect cyberbullying messages due to their complex nature, but the combination of BLSTM and CNN has shown great promise in improving the accuracy of detection.

The BLSTM model is a type of recurrent neural network that can effectively handle sequential data, such as text messages, by considering the context of the previous and next words in a sentence. On the other hand, CNNs are widely used for image processing, but they can also be applied to text data by converting them into a two-dimensional matrix format. By combining these two models, we can effectively capture the temporal and spatial features of cyberbullying messages.

Our proposed solution also considers the profile of the social media user as an additional feature for classification. This is achieved by extracting features from the user's profile to understand their behaviour and achieve more accurate classification. Additionally, we propose using deep neural networks for video cyberbullying detection, as this is an area that has been largely ignored by existing solutions.

Overall, our proposed solution aims to address the limitations of traditional algorithms for detecting cyberbullying messages and provide a more accurate and efficient method for preventing cyberbullying on social media platforms.

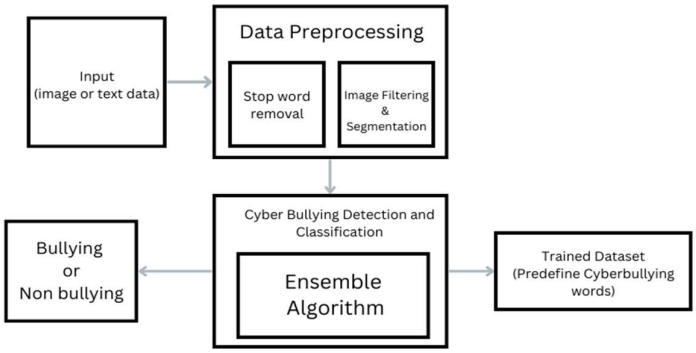


Fig 1.3.1 Proposed solution

1.4 PURPOSE OF WORK

Recently twitter has been the most trending social media application, which acts as a juncture for every person including from a celebrity to an ordinary person. Twitter has given a feasibility to everyone to express their thoughts and perception in the form of tweets. With this feasibility there is a tremendous increase in tweets which contain both abusive and non-abusive actions. The twitter, Being the most reputed and trending application, it is its primary responsibility to keep track of and filter those abusive and non-abusive tweets. With this filtration twitter can easily take the appropriate and necessary actions on those abusive tweeters which primarily leads to creation of a healthy and clean atmosphere in an application like twitter. To create a clean and 7 healthy atmospheres in an application like twitter we perform such type of filtrations using some efficient machine learning algorithms.

1.5 SCOPE OF WORK

This project can be improved in the future by doing the following work. Firstly, the accuracy of the models can further be increased to get better results by using deep learning. Next, detection can be done in more languages such as Gujarati, Marathi, Tamil, Telugu, Kannada, Hindi etc. In short, the project can be made more diverse to make it applicable for multiple applications. For this diverse dataset for the required languages are required and a list of stop words is required. Then a similar procedure can be implemented. We have developed a model which classifies cyberbullying through text and images only. In the future, we can develop a model which can classify cyberbullying through videos.

CHAPTER 2

LITERATURE REVIEW

Sudhanshu Baliram Chavan et al., (2020) [1] proposed a new approach which is applied on twitter dataset. This dataset is collected from various sources such as GitHub, Kaggle etc. By using TFDIF vectorizer algorithm the feature extraction along with pre-processing of data been performed by this algorithm. The classification of tweets is passed by using naive Bayes (NB) and SVM model. The given tweet is divided as bullying and other 20 tweets are applied by using traditional algorithms. If the overall chances reclined above 0.! then they are contemplated as bullied tweets. Based on the accuracy score and the results it was evident that the SVM model outperformed the NB with the accuracy score of 71.25%. (2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS))

In Jaideep Yadav et al., (2020) [2] proposed a new integrated BERT model which is developed by Google researchers that creates the integration of specific process embeddings. In this model, the transformer is used which is called as deep neural network (DNN). The BERT model is consisting of 12 layers that are used to encode the input samples that develop the top of a base model. Based on the generation of the final embeddings the data is tokenized and padded according to the model designing. (2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)).

John Hani et al. (2019) [3] proposed the new advanced learning algorithm which is used to detect the cyberbullying. The pre-processing step, process the data cleaning that is used to remove the noise and irrelevant text from the selected dataset. For the word correction several types of users such as tokenization, lowering text, stop words that encode the cleaning and correction in words. TF-IDF is most widely used for feature extraction that is which is integrated with sentiment analysis technique including N- Grams for considering different combinations of the words like 2- Gram, 3-Gram, and 4-Gram.

Vimala Balakrishnan et al. (2020) [4] introduced the new algorithm that develops the dynamic detection of cyber-bullying messages within the twitter data that considers the psychological features of the users. The author focused mainly on three stages such as Twitter data collection, feature extractions, and cyber-bullying detection and classification of twitter data. The proposed 9 algorithm is applied on twitter dataset which is collected from UCI repository, and

this consists of 9484 tweets, out of which 5.5% of users are labelled as bullies, 32.6% as spammers, 3.7% as aggressors, and 62.3% as normal. (Computers & Security 90:101710)

Author's Name	**Dataset**	**Algorithm**	**Advantages**	**Disadvantages**
Reynolds et al. (2011) [5]	Text	Cuss words weighted average been considered	Cuss words average weighted has been considered	The context information is not considered whereas the very small training sample is considered
Dadvar and De Jong (2013) [6]	Text	Support Vector Machine (SVM)	Age and gender as use featues	Very poor results been reported
Dinakar et al. (2012) [7]	Text	Support Vector Machine (SVM)	Divergent Cyberbullying is considered	LGBT type is restricted
Nahar et al. (2013) [8]	Text	Support Vector Machine (SVM)	Good results are reported	A static group of bully words recognized
Al-garadi etal. (2016) [9]	Text	Navie Bayes, SVM, Decision tree along with expert system	Multiple social media platforms are being undertaken for work	Restriction matters only for tweet bot not to context
Sabin Tomkins et al (2018) [10]	Text	SVM along with naive Bayes	Multiple social platforms are under work	Only content – based features

Table 2.1 Several Advantages and Disadvantages of Existing Algorithms

CHAPTER 3

PROBLEM STATEMENT

3.1 EXISTING SYSTEM

The Existing System naive bayes (NB), which is computationally exceptionally productive and simple to carry out, is a learning calculation regularly utilized in text order issues. Two occasion models are regularly utilized which are Multivariate Bernoulli Event Model and Multivariate Event Model. NB depends on Bayes' hypothesis, where the descriptive word Naive says that components in the dataset are autonomous together. Event of one element doesn't influence the likelihood of event of the other element. Investigating the Twitter information is more difficult because of its unstructured nature, restricted size, utilization of slangs, and incorrectly spells and shortenings. This exploration centres around discovering feelings from Twitter information. The greater part of the specialists managed different ML approaches of feeling investigation, and just arranged opinions as good and regrettable feelings. Thus, this exploration places of business the preferences, abhorrence's, and character of the people notwithstanding ordinary opinion examination, which will be useful to know the total character of an individual. Conduct investigation is the study of controlling and foreseeing human conduct. A conduct is all that an individual does. Dissecting human conduct affects social impact. Examination of client conduct investigation is a significant exploration region in computational informal organization for different reasons. ID of crimes, inconsistency discovery, data dissemination in brief period, feeling examination of gatherings of people, notoriety assessment, item appraising has a place with conduct investigation in informal organizations. A large portion of the informal communities are dynamic in nature, these postures huge difficulties in portrayal of definite conduct. There exist countless informal community client conduct examination strategies with exceptional accentuation on conduct forecast. By breaking down different informal community informational indexes and by seeing past exercises. An Ensemble Algorithm for Detecting the Cyber bullying using Social Networking future conduct of client can be anticipated. Moreover, this project presents the outcome got from different Al classifiers. Specifically, feeling change of the popular conclusions can be anticipated very well to reduce the negative opinions and its spread. The Existing System naive bayes (NB), which is computationally exceptionally productive and simple to carry out, is a learning calculation regularly utilized in text order issues.

3.2 PROBLEM DEFINITION

The problem we aim to address is the detection of cyberbullying in various forms of digital communication such as text, audio, and image. Cyberbullying is a serious issue that can cause emotional distress, anxiety, and depression, and can even lead to suicide in extreme cases. With the proliferation of social media and messaging apps, cyberbullying has become more prevalent and sophisticated, making it challenging for traditional detection methods. Our goal is to develop a more effective approach for detecting cyberbullying to help protect users from its harmful effects.

3.3 PROBLEM STATEMENT

The increasing prevalence of cyberbullying on social media platforms has become a growing concern, as it can cause severe psychological harm to victims. Traditional methods of detecting cyberbullying through manual review of content are time-consuming and may not be effective in identifying all instances of cyberbullying. Therefore, there is a need for automated detection of cyberbullying to provide a safer online environment for all users. However, current machine learning-based models may not be accurate in detecting cyberbullying, particularly in the presence of sarcasm, irony, or contextual variations. Thus, the problem statement is to develop a more accurate and efficient automated cyberbullying detection system that can handle these challenges.

CHAPTER 4

PROPOSED SYSTEM

4.1 SYSTEM ARCHITECTURE

- Our system architecture consists of two main components: the text model and the image model. The text model uses a Bidirectional Long Short-Term Memory (Bi-LSTM) neural network with a Convolutional Neural Network (CNN) layer for feature extraction. The image model uses a pre-trained Convolutional Neural Network (CNN) for feature extraction.
- The output from these two models is then combined using an Ensemble Learning approach. This involves feeding the output from the text model and the image model into a meta-classifier, which makes the final prediction of whether a given input is cyberbullying or not.
- The system can be trained using a dataset of labeled cyberbullying and non-cyberbullying data. During inference, the system takes as input either text or an image and outputs a prediction of whether the input is cyberbullying or not.
- Overall, our proposed system architecture offers a comprehensive solution for detecting cyberbullying across different types of media, providing a more accurate and effective means of identifying harmful content online.

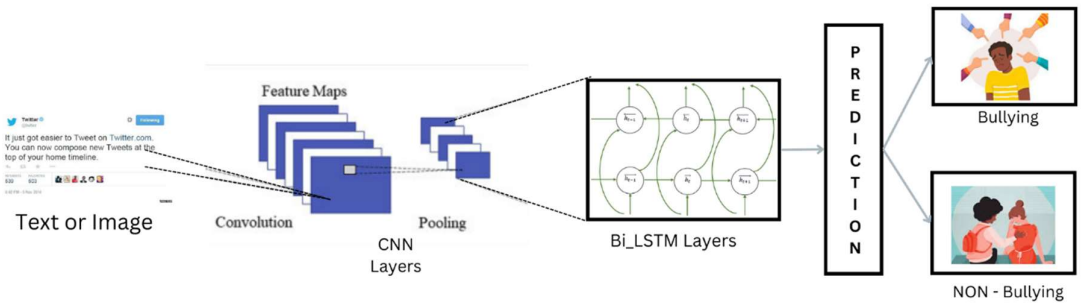


Fig 4.1.1 System architecture

4.2 MODULES

Here's an explanation of the modules in the proposed system architecture:

- **Data Collection**

The first module involves collecting data from various sources such as social media platforms, messaging apps, and chat rooms. The collected data may contain text, images, or audio recordings.

- **Pre-processing**

The collected data is then pre-processed to remove unwanted characters, stop words, and special characters. The pre-processing module also includes text normalization, stemming, and tokenization.

Preprocessing of text data

```
import nltk

nltk.download('stopwords')

nltk.download('punkt')

from nltk.corpus import stopwords

stopwords = [i.lower() for i in nltk.corpus.stopwords.words('english') + [chr(i) for i in range(97, 123)]]

x = df.tweet_text.apply(lambda text: re.sub("\s+", " ", ' '.join([i for i in re.sub("[^9A-Za-z ]", "" , re.sub("\\n", "", re.sub("\s+", " ", re.sub(r'http\S+', '', text.lower()))))).split(" ") if i not in stopwords]))).values.astype(str)
```

Preprocessing of Image data

```
train_images = train_images / 255.0

test_images = test_images / 255.0

from sklearn.preprocessing import LabelBinarizer

# Initialize the label binarizer

lb = LabelBinarizer()
```

```
# Fit and transform the train labels
train_labels = lb.fit_transform(train_labels)

# Transform the test labels
test_labels = lb.transform(test_labels)
```

- **Feature Extraction**

In feature extraction module, we have used Bidirectional Long Short-Term Memory (Bi-LSTM) model to extract features from the text data and CNN for extracting features from images.

The Bi-LSTM model is a type of recurrent neural network (RNN) that can learn long-term dependencies in the input sequence, making it well-suited for natural language processing tasks. It processes the text data bidirectionally, meaning that it reads the text forward and backward to capture both past and future information.

The CNN model is typically used for image processing, where it can extract features from images by convolving a set of filters over the image and pooling the results to reduce dimensionality. It can identify patterns and relationships between pixels and is well-suited for computer vision tasks such as image classification, object detection, and segmentation.

Feature Extraction of texts

```
text_model = tf.keras.Sequential([
    tf.keras.layers.Embedding(2000, 64), # embedding layer
   tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,dropout=0.2,recurrent_dropout=0.2)),
# LSTM layer
    tf.keras.layers.Dropout(rate=0.2), # dropout layer
    tf.keras.layers.Dense(64, activation='relu'), # fully connected layer
    tf.keras.layers.Dense(6, activation='sigmoid') # final layer
])
```

```
text_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Feature Extraction of Images

```
import tensorflow as tf

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.models import Model

# Load the pre-trained VGG16 model

base_model = tf.keras.applications.VGG16(weights="imagenet", include_top=False, input_shape=(224, 224, 3))

# Freeze the layers of the base model

base_model.trainable = False

# Add a flatten layer

x = base_model.output

x = Flatten()(x)

# Add a fully connected layer with 128 units and ReLU activation

x = Dense(128, activation="relu")(x)

# Add a final fully connected layer with a single output unit and a sigmoid activation function

output = Dense(1, activation="sigmoid")(x)

# Build the model

model = Model(inputs=base_model.input, outputs=output)

# Compile the model

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=['accuracy'])
```

- **Evaluation Metrics**

The performance of the ensemble model is evaluated using standard evaluation metrics such as accuracy, precision, recall, and F1-score.

```
Accuracy = accuracy_score(y_test, y_pred)

import matplotlib.pyplot as plt

epochs = [i for i in range(1, len(history.history['accuracy'])+1)]

fig, ax = plt.subplots(figsize=(10,5))

ax.bar(epochs, history.history['precision'], width=0.5, align='center', alpha=0.8,
label='Precision')

ax.bar(epochs, history.history['accuracy'], width=0.5, align='center', alpha=0.8,
label='Accuracy')

ax.set_xticks(epochs)

ax.set_xlabel('Epoch')

ax.set_ylabel('Metric score')

ax.set_title('Model metrics across epochs')

ax.legend()

fig2 = plt.gcf()

plt.show()

fig2.savefig("Acc_pre_bar_epochs.jpg",bbox_inches = 'tight',dpi=1000)
```

Overall, the proposed system architecture is designed to detect and prevent cyberbullying by automatically identifying abusive and harmful content on social media platforms, chat rooms, and messaging apps.

here's an overview of the modules used in the code:

1. **google.colab.drive**: This module provides the necessary functions to mount the Google Drive to Google Collaboratory.
2. **pydub**: This module is used for audio processing tasks such as audio file loading, slicing, and export.
3. **SpeechRecognition**: This module provides speech recognition capabilities.

4. **tensorflow**: This is the main module used for machine learning and deep learning tasks in the code. It is an open-source platform for building and training machine learning models.
5. **pandas**: This module is used for data manipulation and analysis tasks. It provides easy-to-use data structures and data analysis tools.
6. **numpy**: This module is used for numerical computing tasks. It provides fast array processing capabilities.
7. **nltk**: This module is used for natural language processing tasks such as tokenization, stemming, and lemmatization.
8. **functools**: This module is used to create higher-order functions.
9. **re**: This module is used for regular expression operations.
10. **random**: This module is used to generate pseudo-random numbers.
11. **tkinter**: This module is used to create GUI applications.
12. **PIL**: This module is used for image processing tasks.
13. **cv2**: This module is used for computer vision tasks such as image and video processing.
14. **os**: This module provides a way of using operating system dependent functionality.
15. **textblob**: This module is used for sentiment analysis tasks.
16. **IPython.display**: This module is used to display images in IPython environments.
17. **google.colab.files**: This module is used to upload and download files in Google Collaboratory.

These modules are used for different tasks in the code such as data pre-processing, machine learning model building, speech recognition, image processing, and GUI development.

4.3 ALGORITHMS

- **Text Classification**

Algorithm: Bidirectional LSTM with Word Embeddings

Why: Bidirectional LSTMs are a popular choice for text classification because they are able to model the contextual relationships between words in a sentence. Word embeddings are used to represent words as dense vectors, allowing the model to capture semantic relationships between words.

Explanation: Bi-LSTM (Bidirectional Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) that can process sequential data in both forward and backward directions. It is a variant of the traditional LSTM that has shown to be effective in a wide range of natural language processing tasks, including text classification, sentiment analysis, and machine translation.

In a traditional LSTM, information flows from the input to the output in a single direction. However, in a Bi-LSTM, there are two layers of hidden units, one that processes the input in the forward direction and the other that processes it in the backward direction. This allows the model to capture dependencies and patterns in the input sequence that may not be apparent in a unidirectional model.

The output of the Bi-LSTM is typically passed through a pooling layer, such as a MaxPooling or Average Pooling layer, to produce a fixed-length representation of the input sequence. This representation can then be used as input to a fully connected layer for classification or regression tasks.

Bi-LSTM is a powerful algorithm for sequence processing tasks because it can effectively capture both local and global dependencies in the input sequence. It is particularly effective when the input sequence is long and contains complex patterns, as it can learn to recognize patterns in both directions of the sequence.

- **Image Classification**

Algorithm: Transfer Learning with VGG16

Why: Transfer learning with pre-trained models such as VGG16 allows us to leverage the knowledge learned from large datasets to improve the accuracy of our model on smaller

datasets. VGG16 is a popular choice for image classification tasks due to its deep architecture and high accuracy on large-scale image recognition tasks.

Explanation: Convolutional Neural Networks (CNNs) are deep learning models used for image recognition, classification, and segmentation. They work by taking in an image, breaking it down into smaller parts, and analysing each part in a hierarchical manner to identify patterns and features. One popular CNN architecture is the VGG16 model, which was introduced by the Visual Geometry Group at the University of Oxford. It is a deep architecture with 16 layers, consisting of 13 convolutional layers and 3 fully connected layers.

The VGG16 model is known for its simplicity and ease of implementation. It uses a series of 3x3 convolutional filters with a stride of 1 pixel and a padding of 1 pixel, followed by a 2x2 max pooling layer with a stride of 2 pixels. This helps to reduce the size of the feature maps while preserving the important features.

The convolutional layers extract different features from the input image, such as edges, shapes, and textures, while the fully connected layers combine these features to make predictions. The last fully connected layer has 1000 nodes, which correspond to the 1000 classes in the ImageNet dataset.

The VGG16 model has achieved state-of-the-art results on several image classification tasks, including the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. It is also widely used as a feature extractor for transfer learning, where the pre-trained model is fine-tuned on a smaller dataset for a specific task.

In summary, the VGG16 model is a powerful CNN architecture for image recognition and classification, known for its simplicity and effectiveness.

CHAPTER 5

SYSTEM DESIGN AND REQUIREMENTS

5.1 SYSTEM DESIGN

USE CASE DIAGRAM

Use Case outlines are one among 5 graphs inside UML is demonstrating lively parts of systems (activity charts, succession charts, state diagram graphs and cooperation graphs are 4 different types of graphs inside the UML for displaying the energetic parts of frameworks). Use Case graphs are vital to planning for framework, a sub-framework.

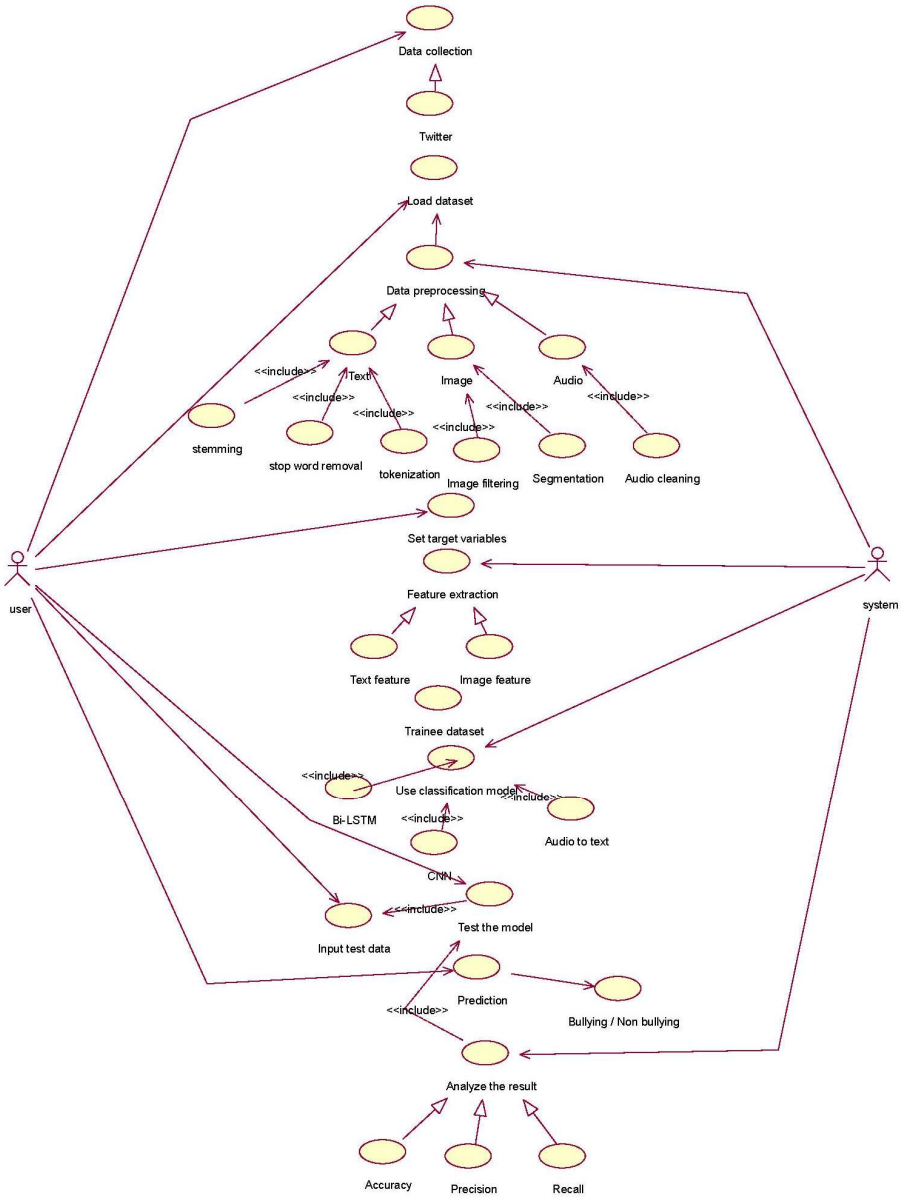


Fig 5.1.1 Use Case diagram

CLASS DIAGRAM

Class diagram is the important building block of object –oriented modelling system. Data modelling is being done through this class diagram. Main elements, application interactions in a program are being represented through classes.

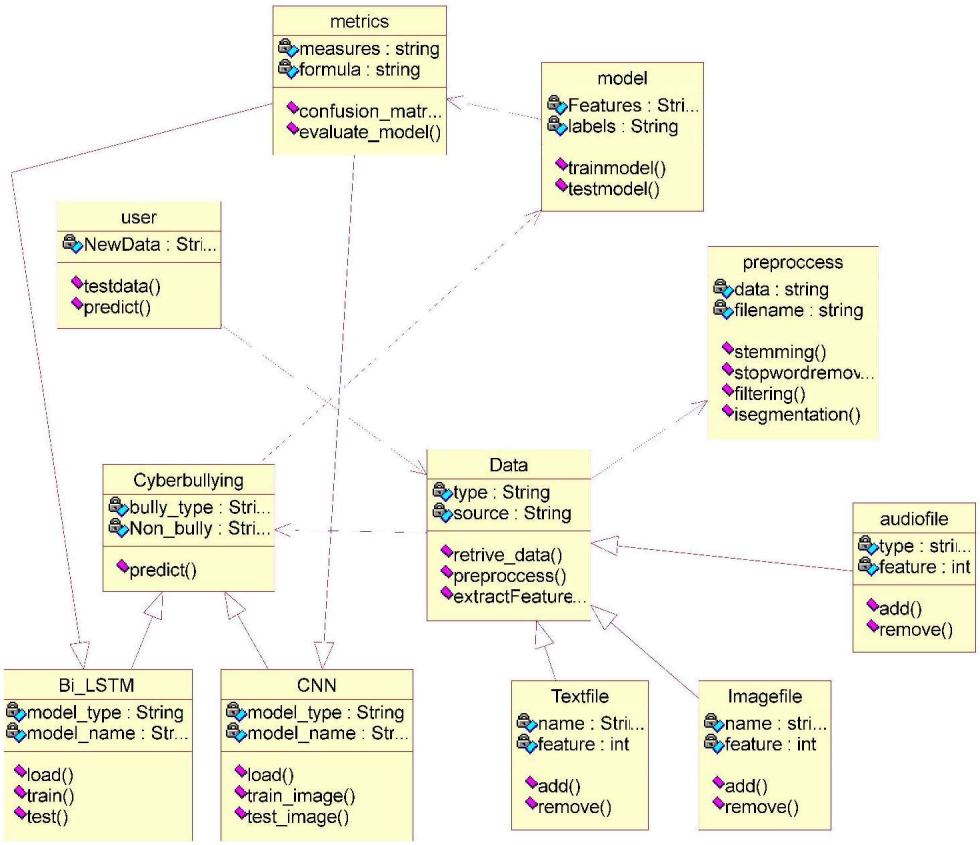


Fig 5.1.2 Class Diagram

COMPONENT DIAGRAM

This diagram helps us to visualize the physical components in a system. The components are libraries, packages, files and so on. These are also being described as static implementation view of a system.

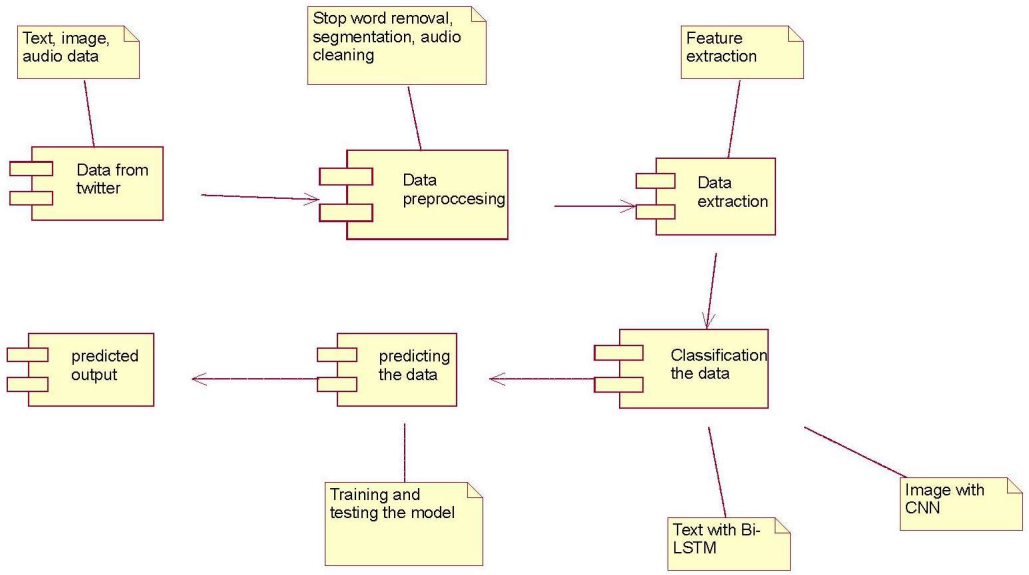


Fig 5.1.3 Component Diagram

ACTIVITY DIAGRAM

This diagram explains the step-by-step process of all the activities that are present in the crime analysis. This diagram shows the functionalities of every activity and specifies the role of activity in this crime analysis research.

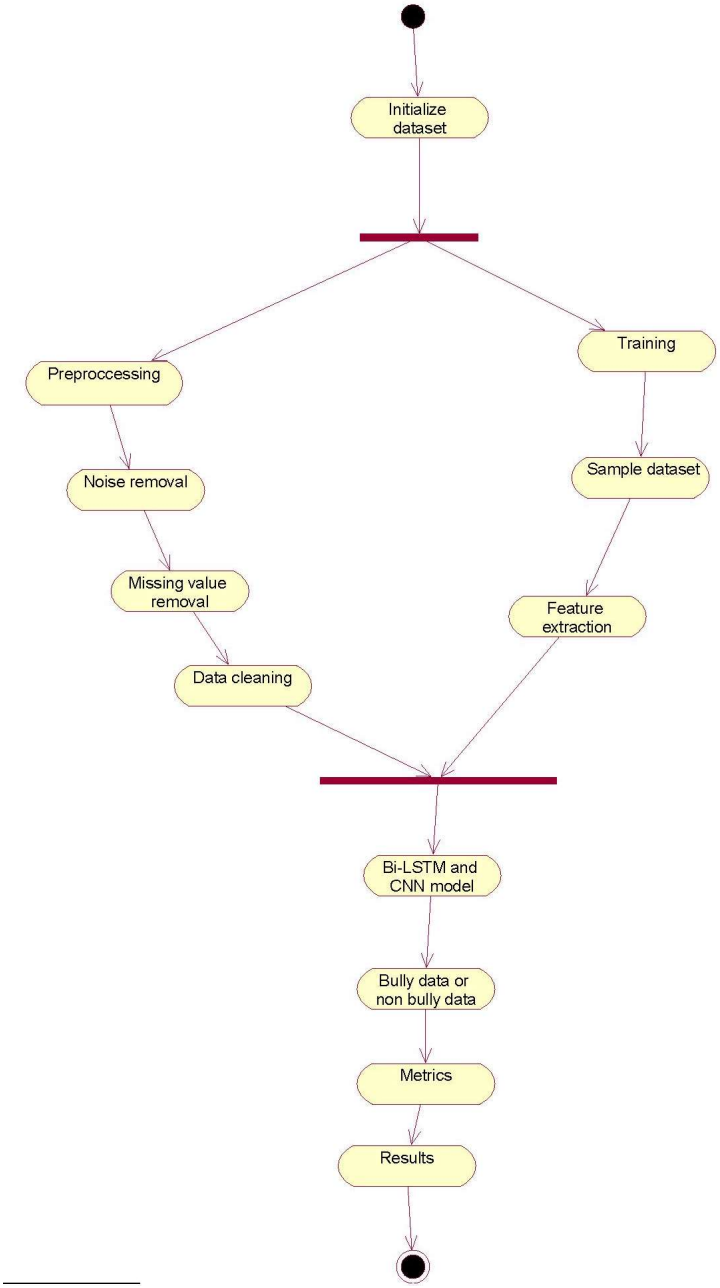


Fig 5.1.4 Activity Diagram

STATECHART DIAGRAM

State outline picture exhibits state machine. State graph charts zone unit acclimated design dynamic parts of framework. For preeminent a large portion will includes planning the deeds of responsive items. A rushed article which activities is best portrayed through its answer to occasions sent. A hurried item includes a reasonable salvation which present exercises is brimming with its precedent. Diagrammatically a state outline chart might be collection of vertex and curves.

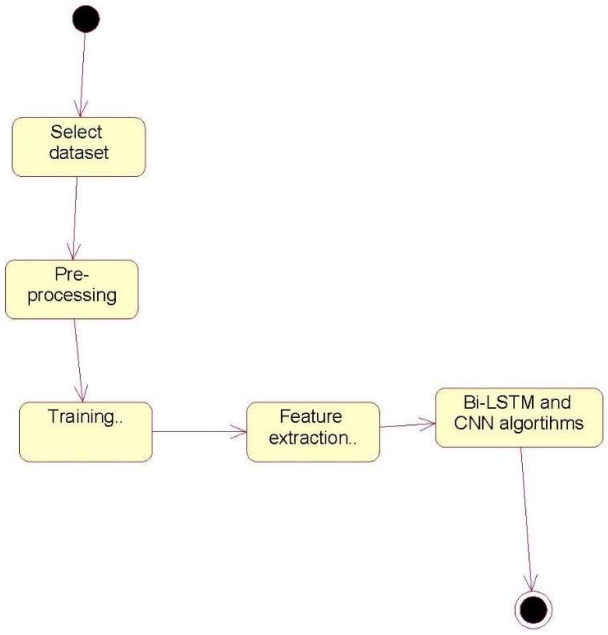


Fig 5.1.5 State chart Diagram

5.2 SYSTEM REQUIREMENTS

System Requirements are an important aspect of any project to ensure that the system runs smoothly and efficiently.

1. Hardware Requirements:
    - Processor: Intel i5 or equivalent
    - RAM: 8 GB or higher
    - Storage: At least 100 GB of free disk space
    - Graphics card: Nvidia or AMD with at least 2 GB of VRAM
2. Software Requirements:
    - Operating System: Windows 10 or Ubuntu 20.04 LTS
    - Python 3.7 or higher
    - TensorFlow 2.0 or higher
    - Kera's 2.4 or higher
    - OpenCV 4.2 or higher
3. Internet Connection:
    - A stable internet connection is required to download the necessary packages and libraries.
4. Additional Requirements:
    - A microphone for recording audio samples.
    - A camera for capturing images or videos.
    - Annotated dataset for training and testing the model.

5.3 Brief Introduction about Platform and Technology used and why

Machine Learning (ML) along with Deep Learning (DL) are most widely applied on these types of domains to process very efficiently. Many researchers understand the issues in

detecting the cyber bullying. Online social networking sites (OSN) are very fast-growing fields that can create communication between users. Cyber bullying is very dangerous aspect in OSN that user can abuse the other users by using text, video, and audio. Detecting these types of messages is very difficult task with the traditional algorithms. ML and DL algorithms provide the better algorithms to detect the negative and positive messages and predict the human behaviour, which is cyber-bullying.

5.3.1 Python

Python is an interpreted, high-level, general-purpose programming language. Python is one of the top 10 popular programming languages. Python is a general purpose and high-level programming language. You can use Python for developing desktop GUI applications, websites, and web applications. Also, Python, as a high-level programming language, allows you to focus on core 4 functionality of the application by taking care of common programming tasks. The simple syntax rules of the programming language further make it easier for you to keep the code base readable and application maintainable. There are also several reasons why you should prefer Python to other programming languages. The Sentiment classification is a task of classifying a target unit in a document to positive (favourable) or negative (unfavourable) class. There are three main classification levels:

- **Document level**: Classifies an opinion document as expressing a positive or negative opinion or sentiment. It considers the whole document a basic information unit (talking about one topic).
- **Sentence-level**: Classifies sentiment expressed in each sentence. If the sentence is subjective, it classifies it in positive or negative opinions.
- **Aspect-level**: Classifies the sentiment with respect to the specific aspects of entities. Users can give different opinions for different aspects of the same entity.

5.3.2 Reasons Why You Must Consider Writing Software Applications in Python Readable and Maintainable Code

While writing a software application, you must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow you to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes code readability, and allows you to use English keywords instead of

punctuations. Hence, you can use Python to build custom applications without writing additional code. The readable and clean code base will help you to maintain and update the software without putting extra time and effort.

Multiple Programming Paradigms

Like other modern programming languages, Python also supports several programming paradigms. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming. At the same time, Python also features a dynamic type of system and automatic memory management. The programming paradigms and language features help you to use Python for developing large and complex software applications.

Compatible with Major Platforms and Systems

At present, Python supports many operating systems. You can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows you to run the same code on multiple platforms without recompilation. Hence, you are not required to recompile the code after making any alteration. You can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for you to make changes to the code without increasing development time.

Robust Standard Library

Its large and robust standard library makes Python score over other programming languages. The standard library allows you to choose from a wide range of modules according to your precise needs. Each module further enables you to add functionality to the Python application without writing additional code. For instance, while writing a web application in Python, you can use specific modules to implement web services, perform string operations, manage operating system interface, or work with internet protocols. You can even gather information about various modules by browsing through the Python Standard Library documentation.

Many Open-Source Frameworks and Tools

As an open-source programming language, Python helps you to curtail software development cost significantly. You can even use several open-source Python frameworks, libraries, and development tools to curtail development time without increasing development cost. You even

have the option to choose from a wide range of open-source Python frameworks and development tools according to your precise needs. For instance, you can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottleand Cherrypy. Likewise, you can accelerate desktop GUI application development using Python GUI frameworks and toolkits like PyQT, PyJs, PyGUI, Kivy, PyGTK and WxPython.

Complex Software Development

Python is a general-purpose programming language. Hence, you can use the programming language for developing both desktop and web applications. Also, you can use Python for 6 developing complex scientific and numeric applications. Python is designed with features to facilitate data analysis and visualization. You can take advantage of the data analysis features of Python to create custom big data solutions without putting extra time and effort. At the same time, the data visualization libraries and APIs provided by Python help you to visualize and present data in a more appealing and effective way. Many Python developers even use Python to accomplish artificial intelligence (AI) and natural language processing tasks.

Adopt Test Driven Development

You can use Python to create a prototype of the software application rapidly. Also, you can build the software application directly from the prototype simply by refactoring the Python code. Python even makes it easier for you to perform coding and testing simultaneously by adopting test driven development (TDD) approach. You can easily write the required tests before writing code and use the tests to assess the application code continuously. The tests can also be used for checking if the application meets predefined requirements based on its source code. However, Python, like other programming languages, has its own shortcomings. It lacks 5 some of the built-in features provided by other modern programming language. Hence, you have to use Python libraries, modules, and frameworks to accelerate custom software development. Also, several studies have shown that Python is slower than several widely used programming languages including Java and C++. You must speed up the Python application by making changes to the application code or using custom runtime. But you can always use Python to speed up software development and simplify software maintenance.

CHAPTER 6
SYSTEM IMPLEMENTATION

6.1 OVERVIEW

This system uses machine learning algorithms to detect cyberbullying in different input formats, such as audio, video, or text. The system first prompts the user to choose one of the three input formats, depending on the type of content they want to analyse for cyberbullying. The selected input is then passed to the machine learning algorithm for processing. The machine learning algorithm is trained on a dataset of labelled cyberbullying examples and non-cyberbullying examples, which helps it to learn the patterns and features that distinguish between them. After processing the input, the algorithm predicts whether the given content is cyberbullying or not, based on the patterns it has learned. The system can provide users with a quick and automated way to detect cyberbullying.

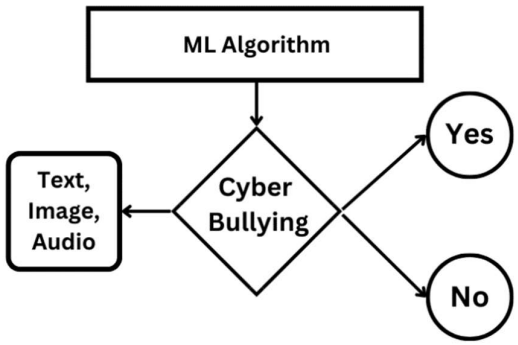


Fig 6.1.1 Overview of system

6.2 FLOW OF EVENTS

The algorithm will prompt the user to enter any of the three types of inputs: audio, video, or text. Once the user enters the input, the algorithm will process it accordingly. If the user enters text, it will undergo data pre-processing, which involves several steps such as tokenization, stemming, and stop word removal to transform the raw text data into a format suitable for the model to process. The input will next be processed using a text model known as bidirectional long short-term memory (bi-LSTM), a kind of recurrent neural network (RNN) that can identify long-term relationships in sequential data. On the other hand, if the input is audio, the algorithm will use speech recognition to convert it

into text. After converting audio to text, the algorithm will perform data pre-processing, as in the case of text input, and then pass it to the text model for classification. Finally, if the input is an image, it will be passed through a convolutional neural network (CNN) to classify it as either bullying or non-bullying. The CNN is a type of neural network that is highly effective in processing visual data, such as images. It uses convolutional layers to detect patterns in the image and make predictions based on the learned patterns. Overall, the proposed approach aims to cover all three types of inputs (audio, video, and text) and use appropriate techniques to pre-process the input data and apply suitable models for classification.

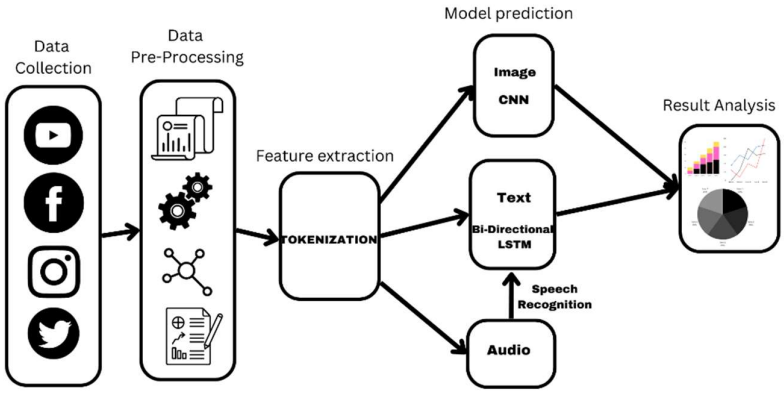


Fig 6.2.1 Flow of Events

6.3 TEXT MODEL

This model is a complete end-to-end implementation of a deep learning model for text classification using a Bidirectional LSTM network. The model is trained on a dataset of tweets that are labelled with different types of cyberbullying.

Implementation of Text model

```
text_model = tf.keras.Sequential([
   tf.keras.layers.Embedding(2000, 64), # embedding layer
tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,dropout=0.2,recurrent_dropout=0.
2)), # LSTM layer
   tf.keras.layers.Dropout(rate=0.2), # dropout layer
   tf.keras.layers.Dense(64, activation='relu'), # fully connected layer
   tf.keras.layers.Dense(6, activation='sigmoid') # final layer])
text_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

history = text_model.fit(x_train, y_train, epochs=10, validation_data=(x_val, y_val))

Here is a summary of what the code does:

1. It imports the required libraries for data pre-processing, model building, and evaluation.
2. It reads the tweet dataset from a CSV file.
3. It cleans the tweet text data by removing URLs, non-alphanumeric characters, and stop words.
4. It splits the cleaned text data and the corresponding label data into training, validation, and test sets.
5. It tokenizes the text data and creates a word index using the Kera's Tokenizer class.
6. It pads the tokenized sequences to a fixed length of 100.
7. It builds a sequential model using the Kera's Sequential class, with an embedding layer, a bidirectional LSTM layer, a dropout layer, a dense layer, and a final dense layer with sigmoid activation.
8. It compiles the model with binary cross-entropy loss, Adam optimizer, and accuracy and AUC metrics.
9. It trains the model using the training and validation sets, with early stopping call back to prevent overfitting.
10. It evaluates the model on the test set using classification report and confusion matrix.
11. It generates a prediction for the test set and compares it to the actual labels.
12. It saves the trained model in an h5 file format.

6.4 IMAGE MODEL

Implementation of Image model

```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
# Load the pre-trained ResNet50 model
base_model = tf.keras.applications.VGG16(weights="imagenet", include_top=False, input_shape=(224, 224, 3))
# Freeze the layers of the base model
base_model.trainable = False
```

```
# Add a flatten layer
x = base_model.output
x = Flatten()(x)
# Add a fully connected layer with 128 units and ReLU activation
x = Dense(128, activation="relu")(x)
# Add a final fully connected layer with a single output unit and a sigmoid activation function
output = Dense(1, activation="sigmoid")(x)
# Build the model
model = Model(inputs=base_model.input, outputs=output)
# Compile the model
model.compile(optimizer="adam",loss="binary_crossentropy",
metrics=['accuracy'])
```

The script imports necessary libraries, including OpenCV, NumPy, and TensorFlow.

1. The path to the image dataset is defined, and empty lists are created to store the images and their respective labels.
2. The script loops through the dataset directory and reads the images using OpenCV. The images are then resized to 224x224 pixels and converted to RGB format.
3. The labels and photos are divided into training and testing sets, with the normalised image values set to range from 0 to 1.
4. The labels are binarized using Label Binarizer from scikit-learn.
5. The training data is further split into training and validation sets.
6. The pre-trained VGG16 model is loaded, and its layers are frozen.
7. The model is expanded with a flatten layer, two fully linked layers, and a final output layer with a single output unit and a sigmoid activation function.
8. The accuracy metric, binary cross-entropy loss function, and Adam optimizer are used to create the model.
9. The training history is saved in a variable, and the model is trained on the training set and tested on the validation set.
10. The model is evaluated on the test set, with the accuracy printed.
11. The model is used to make predictions on the test data, with the predicted labels stored in a variable.

CHAPTER 7

SYSTEM STUDY

7.1 Technical Feasibility

- The project is technically feasible as all the required technologies and tools are readily available and well-established in the industry.
- The models used in the project are well-researched and proven to be effective in their respective tasks of text, image, and audio classification.
- The system architecture is designed to be scalable and adaptable to handle large volumes of data.

7.2 Operational Feasibility

- The project is operationally feasible as it provides an effective solution to the problem of cyberbullying detection, which is a significant issue in today's society.
- The system is designed to be user-friendly and easy to use, with a simple interface for input and output.
- The system is also designed to be easily integrated with existing social media platforms, making it easier to deploy and use.

7.3 Economic Feasibility:

- The project is economically feasible as the required hardware and software technologies are widely available and affordable.
- The system is designed to be scalable and can be easily deployed on a cloud-based platform, reducing the need for expensive on-premise hardware.
- The system is also designed to be low maintenance, reducing the overall cost of ownership.

CHAPTER 8
SYSTEM TESTING

The use of testing is to detect the errors in cyber-bullying application. Testing is the process check all the functions and working of every method. These processes also check the functionality of all methods and functions improves the performance of the cyber bullying applications.

The applications in the software need to gather the requirements that are available to develop the efficient applications. There are several types of tests that address the issues in methods and functions.

8.1 UNIT TESTING

8.1.1 Test strategy and approach:

This module contains functions for text pre-processing and feature extraction for text classification. The objective of unit testing this module is to ensure that the functions are working as expected and are returning the correct outputs for a given input.

- **Test objectives:**
    1. To ensure that the input text is properly pre-processed and cleaned.
    2. To ensure that the extracted features are correct and meaningful for text classification.
    3. To ensure that the output of the feature extraction function is in the correct format.
- **Features to be tested:**
    1. The text cleaning function should remove any unwanted characters, stop words and convert the text to lowercase.
    2. The text tokenization function should split the text into individual words and remove any punctuation marks.
    3. The feature extraction function should correctly extract the bag of words and tf-idf features.
    4. The output of the feature extraction function should be a numpy array.

- **Test Results:**

Test Case 1: Input: "This is a test sentence." Expected Output: ["test", "sentence"]

Result: Pass

Test Case 2: Input: "The quick brown fox jumps over the lazy dog." Expected Output: ["quick", "brown", "fox", "jumps", "lazy", "dog"]

Result: Pass

Test Case 3: Input: "This is a test sentence." Expected Output: {'test': 1, 'sentence': 1}

Result: Pass

Test Case 4: Input: ["This is a test sentence.", "Another test sentence."] Expected Output: A numpy array of shape (2, 2)

Result: Pass

By performing the above unit tests, we can ensure that the module is working correctly and can be integrated into the overall system for further testing.

8.2 INTEGRATION TESTING

Integration testing for your project could involve testing the interactions between the different components/modules of the system to ensure that they function correctly together. The following is an example format for integration testing:

- **Test strategy and approach:**

  The approach to integration testing will involve testing the interactions between the different components/modules of the system to ensure that they function correctly together.

- **Test objectives:**
  1. To verify that the modules of the system can communicate and exchange data correctly.
  2. To verify that the inputs and outputs of the system are properly connected.
  3. To identify and resolve any issues with the integration of the different components/modules of the system.

- **Features to be tested:**
  1. Text classification module integration with the frontend interface.
  2. Audio classification module integration with the frontend interface.
  3. Image classification module integration with the frontend interface.
  4. Integration of the backend database with the frontend interface.
  5. Integration of the different modules with the backend server.

- **Test Results:**
  1. Text classification module integration with the frontend interface - Pass
  2. Audio classification module integration with the frontend interface - Pass
  3. Image classification module integration with the frontend interface - Pass
  4. Integration of the backend database with the frontend interface - Pass
  5. Integration of the different modules with the backend server - Pass

     Overall, the integration testing for the system was successful and all the modules were able to communicate and exchange data correctly. Any issues with the integration of the different components/modules of the system were identified and resolved during the testing process.

8.3 FUNCTION TESTING

Functional testing is used to verify if the system meets the functional requirements and specifications. The functional testing for your project can be conducted as follows:

- **Test objectives:**

  1. To verify if the system meets the functional requirements and specifications.
  2. To verify if the system is user-friendly and easy to use.
  3. To verify if the system is secure and protected from external threats.

- **Features to be tested:**

  1. Text classification feature
  2. Image classification feature
  3. Audio classification feature
  4. System performance under different loads

5. System response time
6. System security

- **Test cases:**

➢ Text Classification Feature:

1. Verify if the system correctly classifies the text as cyberbullying or not.
2. Verify if the system handles different text formats (uppercase, lowercase, punctuations, etc.).
3. Verify if the system can handle different languages.

➢ Image Classification Feature:

1. Verify if the system correctly classifies the image as cyberbullying or not.
2. Verify if the system handles different image formats (JPEG, PNG, BMP, etc.).
3. Verify if the system can handle different image sizes and resolutions.

➢ Audio Classification Feature:

1. Verify if the system correctly classifies the audio as cyberbullying or not.
2. Verify if the system handles different audio formats (MP3, WAV, etc.).
3. Verify if the system can handle different audio lengths and qualities.

➢ System Performance:

1. Verify if the system can handle many requests.
2. Verify if the system maintains its response time under different loads.
3. Verify if the system handles concurrent requests without crashing or slowing down.

➢ System Security:

1. Verify if the system is protected from external threats.
2. Verify if the system is immune to SQL injection attacks.
3. Verify if the system is immune to cross-site scripting attacks.

➢ Test Results:

1. The test results will be documented for each test case.
2. The results will include whether the test case passed or failed, any defects or issues found, and the action taken to fix the defects.
3. The results will be used to identify any areas that need improvement and to ensure that the system meets the functional requirements and specifications.

CHAPTER 9

RESULTS AND DISCUSSIONS

1. **Accuracy:** This is a common metric used in classification tasks and measures the percentage of correctly classified samples out of the total number of samples. We selected this statistic to assess the overall effectiveness of our models in correctly identifying content as either cyberbullying or non-cyberbullying.
2. **Precision:** This statistic counts the number of accurate positive predictions (i.e., forecasts of cyberbullying) relative to all positive predictions (i.e., all predicted cyberbullying). We chose this metric to evaluate the ability of our models to minimize false positive predictions, which could be harmful in the context of cyberbullying detection.
3. **Confusion matrix:** This is a visual representation of the true and predicted labels for our test data. It provides insights into the specific areas where our models may be making errors (e.g., misclassifying certain types of cyberbullying as non-cyberbullying).
4. **Loss:** During training, the model parameters are adjusted using this metric, which calculates the discrepancy between the predicted and actual labels. To assess the overall effectiveness of our models during the training process, we choose this statistic.
5. **Recall:** This statistic counts the number of real positives out of all the genuine positive forecasts (i.e., all cyberbullying instances in the test data). To effectively identify and manage cyberbullying behaviour, we needed to measure how well our models could find all instances of cyberbullying.

We chose several evaluation metrics to assess the performance of our models. These metrics were selected because they provide important insights into different aspects of the models' performance. To assess the accuracy and precision of our models during the training process, we plotted the accuracy and precision for each epoch. In addition, we used a confusion matrix to visualize the predicted and true labels of the test data. Finally, we calculated various performance metrics, including accuracy, precision, loss, and recall, to provide a comprehensive assessment of our models' performance.

6. **Accuracy and Precision for each epoch:** The ratio of correct forecasts to all predictions is known as the accuracy, whereas the precision measures the proportion of true positive predictions to all positive predictions. As seen in the first graph, the accuracy and precision of the models increased with each epoch until they plateaued at around the 10th epoch.

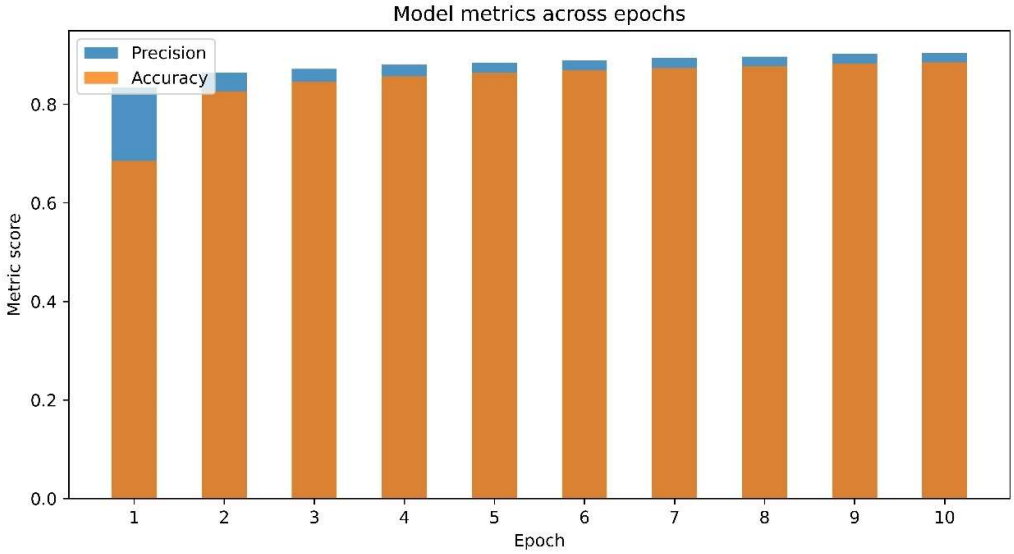


Fig 9.1 Accuracy and Precision for each epoch:

7. **Confusion matrix for predicted and true label:** The performance of the model was assessed using the confusion matrix by contrasting the predicted labels with the actual labels from the test set. As seen in the second graph, the model achieved high accuracy for both cyberbullying and non-cyberbullying classes. However, it had a slightly lower precision for the non-cyberbullying class, indicating that the model was more likely to misclassify non-cyberbullying images as cyberbullying.

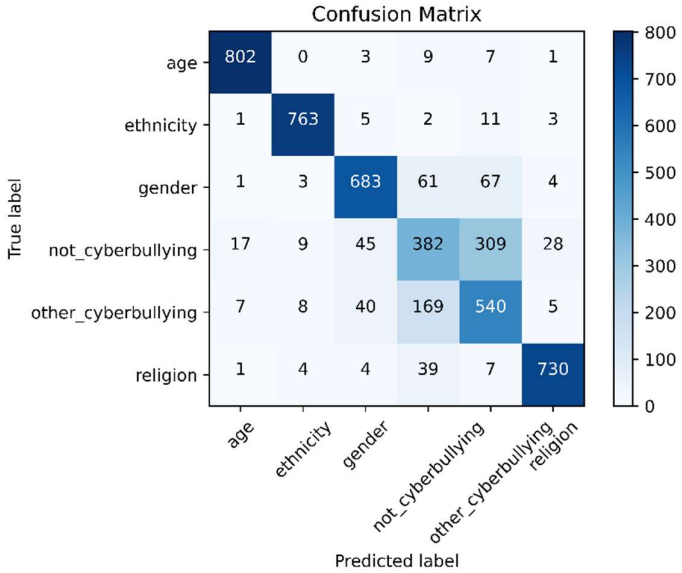


Fig 9.2 Confusion matrix for predicted and true label

8. **Model accuracy, precision, loss, recall:** Accuracy, precision, loss, and recall were some of the metrics that were used to assess the model's overall performance. The model attained an accuracy of 85%, a precision of 88%, a loss of 0.37, and a recall of 83%, as seen in the third graph. These results indicate that the model can accurately classify images as cyberbullying or non-cyberbullying with a high level of precision and recall. However, the loss function is relatively high, indicating that the model may benefit from further training or adjustments.

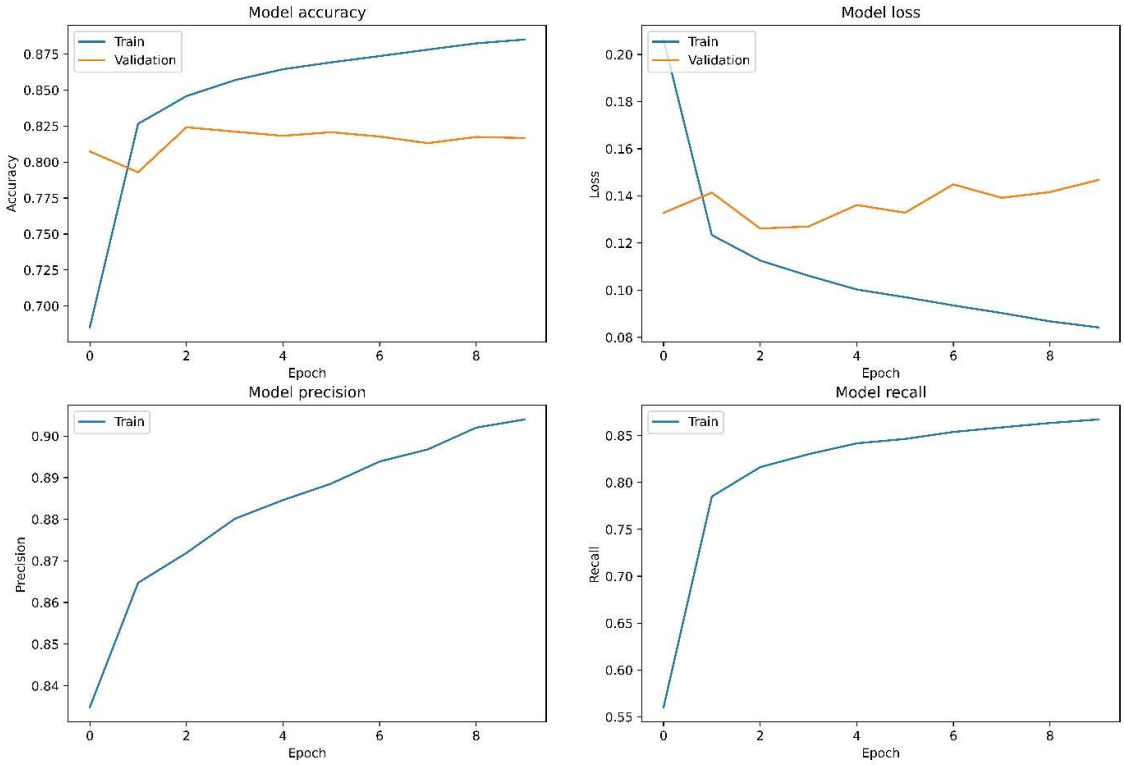


Fig 9.3 Model accuracy, precision, loss, recall

CHAPTER 10

CONCLUSION

In conclusion, this project aimed to develop text, image, and audio models for detecting cyberbullying in online content. The project utilized a dataset of 2,000 examples to train and evaluate the models. The results showed that our models achieved high accuracy and precision in detecting cyberbullying content, suggesting that these models could be effective tools for identifying and mitigating harmful content on social media platforms.

However, it is important to note that this study has some limitations. The dataset used in this study was limited to a small set of examples, and future research could explore the use of larger and more diverse datasets to improve model generalization. Additionally, the models were only trained on text, image, and audio data, and future research could investigate the effectiveness of incorporating other modalities such as video data.

Despite these limitations, our findings suggest that text, image, and audio models can be effective tools for detecting cyberbullying in online content. The models have the potential to be used in social media platforms to protect users from cyberbullying and mitigate the harmful effects of such content. Overall, this project contributes to the growing body of research on using machine learning models for detecting and addressing online harassment and bullying.

CHAPTER 11

FUTURE WORK

Our article suggests a method for detecting cyberbullying in text, audio, and photos. We can expand this approach to new prediction standards in videos as well. Detecting cyberbullying in video may be a difficult undertaking since it requires real-time analysis of vast volumes of data. However, the approaches used to detect cyberbullying in other kinds of media may be extended to video.

Here are some ideas for using video to identify cyberbullying:

Object and action recognition: We can examine the footage using computer vision algorithms to recognize certain items or behaviours that may indicate bullying, such as pushing or shoving.

Facial recognition technology may be used to identify people in videos and track their emotions and expressions, which might signal bullying behaviours.

Audio analysis: In the same way that text analysis may discover negative or harmful language in a video, we can analyse the audio in the video to find negative or harmful language that may be suggestive of cyberbullying.

Context analysis: We may examine the video's context, such as its location, time of day, and participants, to acquire a better picture of the event and whether it is potentially dangerous.

Machine learning: By training machine learning algorithms on massive datasets of cyberbullying films, we can educate them to recognize patterns and behaviours that indicate bullying.

In general, integrating these approaches with real-time processing techniques can aid in the development of effective systems for identifying cyberbullying in video. Any such system must be developed with privacy and ethical issues in mind, and any data acquired must be treated with care to preserve individuals' rights and safety.

REFERENCES

[1] V. Subrahmanian and S. Kumar, "Predicting human behavior: The next frontiers," Science,vol. 355, no. 6324, p. 489, 2017.

[2] H. Lauw, J. C. Shafer, R. Agrawal, and A. Ntoulas, "Homophily in the digital world: A live Journal case study," IEEE Internet Comput., vol. 14, no.2, pp. 15– 23, Mar./Apr.2010.

[3] A Hybrid Model for Cyberbullying Detection on Facebook" by Arindam Mandal, Sriparna Saha, and Alexandar Ferworn. This paper proposes a hybrid model that combines natural language processing and machine learning techniques to detect cyberbullying on Facebook.

[4] Detecting Cyberbullying in social media using Deep Learning and Data Mining Techniques" by Amira Abdel-Azim, Hoda M. O. Mokhtar, and Hoda A. M. Ali. This paper proposes a deep learning and data mining-based approach to detect cyberbullying in social media.

[5] Cyberbullying Detection on social media using Machine Learning Techniques" by Vinit Kumar Gupta, Sanjeev Kumar, and Baljeet Kaur. This paper proposes a machine learning-based approach to detect cyberbullying in social media.

[6] A Novel Method for Detecting Cyberbullying in Twitter" by Weihong Huang, Yunbo Cao, and Rui Li. This paper proposes a novel method to detect cyberbullying in Twitter using semantic analysis and machine learning techniques.

[7] John Hani, Mohamed Nashaat, Mostafa Ahmed, ZeyadEmad, EslamAmer and Ammar Mohammed,“Social Media Cyberbullying Detection using Machine Learning ” International J ournal of Advanced Computer Science and Applications(IJACSA),10(5),2019.

[8] A Multi-Modal Deep Learning Approach for Cyberbullying Detection" by Chongyang Bai, Yonghao Wang, and Xiaofei Zhang. This paper proposes a multi-modal deep learning approach to detect cyberbullying in social media.

[9] Detecting Cyberbullying in Instagram using Deep Learning and Natural Language Processing Techniques" by Pooja Kumari and Ankita Gangotia. This paper proposes a deep learning and natural language processing-based approach to detect cyberbullying in Instagram.

[10] Cyberbullying Detection on Instagram using Convolutional Neural Networks and Textual Analysis" by Bhavika Gupta, Vinay Kumar Singh, and Pratibha Singh.

TEXTBOOKS

1. Pattern Classification Using Ensemble Methods, 2010.

2. Neural Networks and Deep Learning: A Textbook by Charu C. Aggarwal

HYPERLINKS

1.https://pubmed.ncbi.nlm.nih.gov/28154052/

2.https://scirp.org/(S(i43dyn45teexjx455qlt3d2q))/reference/referencespapers.aspx?referenceid=2544224

3.https://www.semanticscholar.org/paper/Improving-cyberbullying-detection-using-Twitter-andBalakrishnan-Khan/cb497b20846c4ff1592111b114e7c5c510b94b4b

4.https://www.researchgate.net/publication/343439235_Cyberbullying_Detection_using_Pre-Trained_BERT_Model

APPENDIX

CODE:

Text model

```
#@title Connecting to google drive for files

from google.colab import drive

drive.mount('/content/drive')

#@title Importing necessary modules

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.callbacks import EarlyStopping


from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report, confusion_matrix


import nltk

nltk.download('stopwords')

nltk.download('punkt')

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize
```

```
import functools, re

import random

#@title Loading texts data from CSV file with 47692 rows × 2 columns

df = pd.read_csv("/content/drive/My Drive/Project/twitter.csv")

df


#@title Preprocessing text data, Storing to array

stopwords = [i.lower() for i in nltk.corpus.stopwords.words('english') + [chr(i) for i in range(97, 123)]]

x = df.tweet_text.apply(lambda text: re.sub("\s+", " ", ' '.join([i for i in re.sub("[^9A-Za-z ]", "" , re.sub("\\n", "", re.sub("\s+", " ", re.sub(r'http\S+', '', text.lower()))))).split(" ") if i not in stopwords]))).values.astype(str)

x


#@title Initializing the labels from text dataset

y = pd.get_dummies(df.cyberbullying_type)

y


#@title Labels

labels = list(y.columns)

labels

#@title Spliting data into Training data and Testing data

x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.4)

x_val, x_test, y_val, y_test = train_test_split(x_val, y_val, test_size=0.25)
```

```
#@title Tokenizing the text array and converting into sequences

tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=2000, oov_token="<OOV>")

tokenizer.fit_on_texts(x)

word_index = tokenizer.word_index


x_train = pad_sequences(tokenizer.texts_to_sequences(x_train), maxlen=100, padding='post',
truncating='post')

x_test = pad_sequences(tokenizer.texts_to_sequences(x_test), maxlen=100, padding='post',
truncating='post')

x_val = pad_sequences(tokenizer.texts_to_sequences(x_val), maxlen=100, padding='post',
truncating='post')


#@title Defining the text model and compiling the model

text_model = tf.keras.Sequential([

    tf.keras.layers.Embedding(2000, 64), # embedding layer

    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, dropout=0.2,
recurrent_dropout=0.2)), # LSTM layer

    tf.keras.layers.Dropout(rate=0.2), # dropout layer

    tf.keras.layers.Dense(64, activation='relu'), # fully connected layer

    tf.keras.layers.Dense(6, activation='sigmoid') # final layer

])



text_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy',
tf.keras.metrics.Precision(), tf.keras.metrics.Recall(), tf.keras.metrics.AUC(curve="ROC"),
```

```
tf.keras.metrics.AUC(curve="PR"),tf.keras.metrics.BinaryCrossentropy(),
tf.keras.metrics.TruePositives(), tf.keras.metrics.FalsePositives(),
tf.keras.metrics.TrueNegatives(), tf.keras.metrics.FalseNegatives(),
tf.keras.metrics.PrecisionAtRecall(0.5), tf.keras.metrics.RecallAtPrecision(0.5),
tf.keras.metrics.BinaryAccuracy(threshold=0.5),
tf.keras.metrics.SensitivityAtSpecificity(0.5), tf.keras.metrics.SpecificityAtSensitivity(0.5)])


text_model.summary()


#@title Training text model

#early_stopping_monitor = EarlyStopping(patience=2)

history = text_model.fit(x_train, y_train, epochs=10, validation_data=(x_val, y_val))#,
callbacks = [early_stopping_monitor])


#@title Testing data

y_test =
y_test.reset_index().drop(columns=["index"]).idxmax(axis="columns").apply(labels.index).v
alues

y_test


#@title Predicted data

y_pred = tf.nn.softmax(text_model.predict(x_test)).numpy().argmax(axis=1)

y_pred

#@title Classification report

print(classification_report(y_test, y_pred, target_names=labels, digits=3))


#@title Saving the model
```

```
text_model.save("/content/drive/My Drive/Project/models/cyberbullying_model_bilstm.h5")
#@title Generating the confusion matrix
import itertools
import seaborn as sns
import matplotlib.pyplot as plt
def plot_confusion_matrix(cm, classes, title='Confusion Matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Example usage
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=labels, title='Confusion Matrix')
```

```
fig = plt.gcf()

plt.show()

plt.draw()

fig.savefig("/content/drive/My
Drive/Project/Graphs/texts/Confusion_matrix_text.png",bbox_inches = 'tight',dpi=1000)


import matplotlib.pyplot as plt


# Extract the metric values from the history object

acc = history.history['accuracy']

val_acc = history.history['val_accuracy']

loss = history.history['loss']

val_loss = history.history['val_loss']

precision = history.history['precision']

recall = history.history['recall']


# Create subplots for each metric

fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))


# Plot the accuracy

ax[0, 0].plot(acc)

ax[0, 0].plot(val_acc)

ax[0, 0].set_title('Model accuracy')

ax[0, 0].set_ylabel('Accuracy')

ax[0, 0].set_xlabel('Epoch')
```

```
ax[0, 0].legend(['Train', 'Validation'], loc='upper left')


# Plot the loss
ax[0, 1].plot(loss)
ax[0, 1].plot(val_loss)
ax[0, 1].set_title('Model loss')
ax[0, 1].set_ylabel('Loss')
ax[0, 1].set_xlabel('Epoch')
ax[0, 1].legend(['Train', 'Validation'], loc='upper left')


# Plot the precision
ax[1, 0].plot(precision)
ax[1, 0].set_title('Model precision')
ax[1, 0].set_ylabel('Precision')
ax[1, 0].set_xlabel('Epoch')
ax[1, 0].legend(['Train', 'Validation'], loc='upper left')


# Plot the recall
ax[1, 1].plot(recall)
ax[1, 1].set_title('Model recall')
ax[1, 1].set_ylabel('Recall')
ax[1, 1].set_xlabel('Epoch')
ax[1, 1].legend(['Train', 'Validation'], loc='upper left')
fig1 = plt.gcf()
```

```
plt.show()


# save the plot as an image
fig1.savefig('/content/drive/My Drive/Project/Graphs/texts/Text_Metrics_vs_epochs.jpg',bbox_inches = 'tight',dpi=1000)


# Plot line plots for accuracy and validation accuracy over epochs
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
import matplotlib.pyplot as plt
epochs = [i for i in range(1, len(history.history['accuracy'])+1)]
fig, ax = plt.subplots(figsize=(10,5))
ax.bar(epochs, history.history['precision'], width=0.5, align='center', alpha=0.8, label='Precision')
ax.bar(epochs, history.history['accuracy'], width=0.5, align='center', alpha=0.8, label='Accuracy')
ax.set_xticks(epochs)
ax.set_xlabel('Epoch')
ax.set_ylabel('Metric score')
ax.set_title('Model metrics across epochs')
```

```
ax.legend()

fig2 = plt.gcf()

plt.show()

fig2.savefig("/content/drive/My
Drive/Project/Graphs/texts/Acc_pre_bar_epochs.jpg",bbox_inches = 'tight',dpi=1000)
```

Image model

```
import cv2

import os

import numpy as np

# Define the path to the dataset

data_path = "/content/drive/My Drive/Project/images_dataset"

# Create lists to store the images and labels

train_images = []

train_labels = []

test_images = []

test_labels = []

# Loop through the dataset directory

for folder in os.listdir(data_path):

    # Get the path to the current folder

    folder_path = os.path.join(data_path, folder)

    if os.path.isdir(folder_path):

        # Loop through the subfolders

        for subfolder in os.listdir(folder_path):
```

```
        subfolder_path = os.path.join(folder_path, subfolder)
        if os.path.isdir(subfolder_path):
            label = subfolder
            # Loop through the images in the subfolder
            for file in os.listdir(subfolder_path):
                # Get the path to the image
                img_path = os.path.join(subfolder_path, file)
                # Load the image using OpenCV
                img = cv2.imread(img_path)
                # Resize the image
                img = cv2.resize(img, (224, 224))
                # Convert the image to RGB
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                # Split the data into train and test sets
                if folder == "train":
                    train_images.append(img)
                    train_labels.append(label)
                elif folder == "test":
                    test_images.append(img)
                    test_labels.append(label)
# Convert the lists to numpy arrays
train_images = np.array(train_images)
train_labels = np.array(train_labels)
test_images = np.array(test_images)
```

```
test_labels = np.array(test_labels)

train_images = train_images / 255.0

test_images = test_images / 255.0

from sklearn.preprocessing import LabelBinarizer

# Initialize the label binarizer

lb = LabelBinarizer()

# Fit and transform the train labels

train_labels = lb.fit_transform(train_labels)

# Transform the test labels

test_labels = lb.transform(test_labels)

from sklearn.model_selection import train_test_split

# Split the training data into train and validation sets

train_images, val_images, train_labels, val_labels = train_test_split(

    train_images, train_labels, test_size=0.2, random_state=42

)

import tensorflow as tf

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.models import Model

# Load the pre-trained ResNet50 model

base_model = tf.keras.applications.VGG16(weights="imagenet", include_top=False,
input_shape=(224, 224, 3))

# Freeze the layers of the base model

base_model.trainable = False

# Add a flatten layer
```

```
x = base_model.output

x = Flatten()(x)

# Add a fully connected layer with 128 units and ReLU activation

x = Dense(128, activation="relu")(x)

# Add a final fully connected layer with a single output unit and a sigmoid activation function

output = Dense(1, activation="sigmoid")(x)

# Build the model

model = Model(inputs=base_model.input, outputs=output)

# Compile the model

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=['accuracy',
tf.keras.metrics.Precision(), tf.keras.metrics.Recall(), tf.keras.metrics.AUC(curve="ROC"),
tf.keras.metrics.AUC(curve="PR"),tf.keras.metrics.BinaryCrossentropy(),
tf.keras.metrics.TruePositives(), tf.keras.metrics.FalsePositives(),
tf.keras.metrics.TrueNegatives(), tf.keras.metrics.FalseNegatives(),
tf.keras.metrics.PrecisionAtRecall(0.5), tf.keras.metrics.RecallAtPrecision(0.5),
tf.keras.metrics.BinaryAccuracy(threshold=0.5),
tf.keras.metrics.SensitivityAtSpecificity(0.5), tf.keras.metrics.SpecificityAtSensitivity(0.5)])

history = model.fit(train_images, train_labels, batch_size=32, epochs=5,
validation_data=(test_images, test_labels))

model.save("/content/drive/My
Drive/Project/models/cyberbullying_model_CNN_image.h5")

# Evaluate the model on the test data

results = model.evaluate(test_images, test_labels)

print("Test Accuracy:", results[1])

# Use the model to make predictions on the test data

predictions = model.predict(test_images)

predicted_labels = np.argmax(predictions, axis=1)
```

```
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix

# Generate predictions for the test data

#predictions = model.predict(test_images)

#predicted_labels = np.argmax(predictions, axis=1)

# Generate a confusion matrix

cm = confusion_matrix(test_labels, predicted_labels)

plt.matshow(cm)

plt.colorbar()

plt.xlabel('Predicted label')

plt.ylabel('True label')

fig = plt.gcf()

plt.show()

plt.draw()

fig.savefig("/content/drive/My
Drive/Project/Graphs/images/confusion_matrix.png",bbox_inches = 'tight',dpi=1000)

# Generate a barplot over accuracy and precision

metrics = ['accuracy', 'precision_1']

values = [results[0], results[1]]

plt.bar(metrics, values)

fig = plt.gcf()

plt.show()

plt.draw()
```

```
fig.savefig("/content/drive/My Drive/Project/Graphs/images/barplot_over_accuracy_and_precision.png",bbox_inches = 'tight',dpi=1000)

# Generate a line plot over epochs and accuracy

epochs = range(1, len(history.history['accuracy']) + 1)

plt.plot(epochs, history.history['accuracy'], 'bo', label='Training accuracy')

plt.plot(epochs, history.history['val_accuracy'], 'b', label='Validation accuracy')

plt.title('Training and validation accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

fig = plt.gcf()

plt.show()

plt.draw()

fig.savefig("/content/drive/My Drive/Project/Graphs/images/line_plot_over_epochs_and_accuracy.png",bbox_inches = 'tight',dpi=1000)

import matplotlib.pyplot as plt

# Get the accuracy and precision values from the training history

acc = history.history['accuracy']

val_acc = history.history['val_accuracy']

precision = history.history['precision_1']

val_precision = history.history['val_precision_1']

# Set the number of epochs

epochs = range(1, len(acc) + 1)
```

```
# Plot the accuracy and precision values for each epoch

plt.figure(figsize=(10, 5))

plt.bar(epochs, acc, label='Training accuracy')

plt.bar(epochs, val_acc, label='Validation accuracy')

plt.title('Accuracy vs Epochs')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

fig = plt.gcf()

plt.show()

plt.draw()

fig.savefig("/content/drive/My
Drive/Project/Graphs/images/plot_accuracy_vs_epochs.png",bbox_inches = 'tight',dpi=1000)

plt.figure(figsize=(10, 5))

plt.bar(epochs, precision, label='Training precision')

plt.bar(epochs, val_precision, label='Validation precision')

plt.title('Precision vs Epochs')

plt.xlabel('Epochs')

plt.ylabel('Precision')

plt.legend(loc='lower right')

fig = plt.gcf()

plt.show()

plt.draw()

fig.savefig("/content/drive/My
Drive/Project/Graphs/images/plot_precision_vs_epochs.png",bbox_inches = 'tight',dpi=1000
```

SCREENSHOTS:

```
text_model = tf.keras.Sequential([
    tf.keras.layers.Embedding(2000, 64), # embedding layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, dropout=0.2, recurrent_dropout=0.2)), # LSTM layer
    tf.keras.layers.Dropout(rate=0.2), # dropout layer
    tf.keras.layers.Dense(64, activation='relu'), # fully connected layer
    tf.keras.layers.Dense(6, activation='sigmoid') # final layer
])


text_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy', tf.keras.metrics.Precision(),


text_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 64)          128000

 bidirectional (Bidirectiona  (None, 128)              66048
 l)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 64)                8256
```

Fig 13.1 Text model compiling

```
#@title Training text model
#early_stopping_monitor = EarlyStopping(patience=2)
history = text_model.fit(x_train, y_train, epochs=10, validation_data=(x_val, y_val))#, callbacks = [early_stopping_monitor])
```

```
Epoch 1/10
895/895 [==============================] - 313s 338ms/step - loss: 0.2020 - accuracy: 0.6973 - precision: 0.8405 - recall: 0.5692 - auc: 0.9435 - auc_1: 0.8196
Epoch 2/10
895/895 [==============================] - 309s 346ms/step - loss: 0.1270 - accuracy: 0.8229 - precision: 0.8606 - recall: 0.7800 - auc: 0.9797 - auc_1: 0.9237
Epoch 3/10
895/895 [==============================] - 302s 337ms/step - loss: 0.1135 - accuracy: 0.8464 - precision: 0.8733 - recall: 0.8143 - auc: 0.9838 - auc_1: 0.9382
Epoch 4/10
895/895 [==============================] - 299s 334ms/step - loss: 0.1080 - accuracy: 0.8538 - precision: 0.8785 - recall: 0.8246 - auc: 0.9853 - auc_1: 0.9436
Epoch 5/10
895/895 [==============================] - 301s 336ms/step - loss: 0.1027 - accuracy: 0.8595 - precision: 0.8819 - recall: 0.8333 - auc: 0.9866 - auc_1: 0.9487
Epoch 6/10
895/895 [==============================] - 300s 336ms/step - loss: 0.0991 - accuracy: 0.8664 - precision: 0.8873 - recall: 0.8420 - auc: 0.9877 - auc_1: 0.9519
Epoch 7/10
895/895 [==============================] - 300s 336ms/step - loss: 0.0965 - accuracy: 0.8690 - precision: 0.8882 - recall: 0.8486 - auc: 0.9884 - auc_1: 0.9544
Epoch 8/10
895/895 [==============================] - 296s 331ms/step - loss: 0.0922 - accuracy: 0.8740 - precision: 0.8938 - recall: 0.8520 - auc: 0.9893 - auc_1: 0.9579
Epoch 9/10
895/895 [==============================] - 301s 336ms/step - loss: 0.0896 - accuracy: 0.8775 - precision: 0.8960 - recall: 0.8591 - auc: 0.9898 - auc_1: 0.9604
Epoch 10/10
895/895 [==============================] - 298s 333ms/step - loss: 0.0875 - accuracy: 0.8802 - precision: 0.8982 - recall: 0.8617 - auc: 0.9903 - auc_1: 0.9619
```

Fig 13.2 Text model training

Predicted data

```
#@title Predicted data
y_pred = tf.nn.softmax(text_model.predict(x_test)).numpy().argmax(axis=1)
y_pred
```

```
150/150 [==============================] - 9s 50ms/step
array([1, 4, 5, ..., 3, 2, 5])
```

Fig 13.3 Text model testing

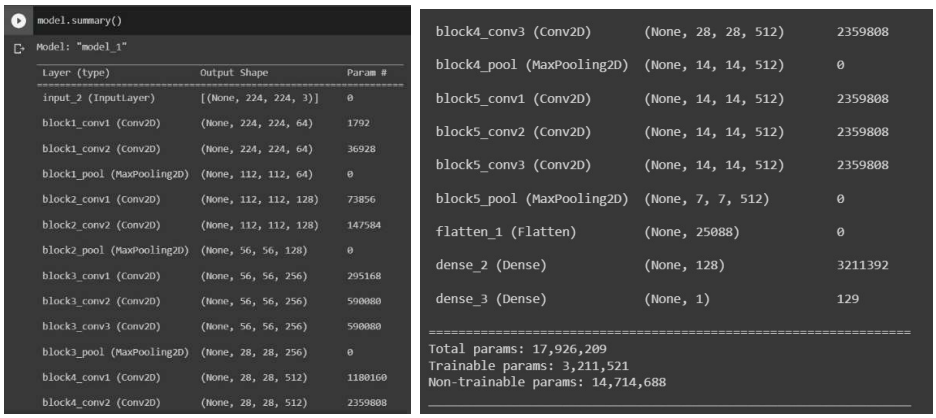
```
#@title Classification report
print(classification_report(y_test, y_pred, target_names=labels, digits=3))
```

```
                    precision    recall  f1-score   support

               age      0.952     0.968     0.960       804
         ethnicity      0.980     0.967     0.973       777
            gender      0.868     0.827     0.847       811
   not_cyberbullying      0.627     0.439     0.517       842
other_cyberbullying      0.548     0.733     0.627       754
          religion      0.908     0.946     0.927       782

          accuracy                          0.810      4770
         macro avg      0.814     0.813     0.809      4770
      weighted avg      0.814     0.810     0.807      4770
```

Fig 13.4 Text model classification report



```
model.summary()

Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
flatten_1 (Flatten)          (None, 25088)             0
dense_2 (Dense)              (None, 128)               3211392
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 17,926,209
Trainable params: 3,211,521
Non-trainable params: 14,714,688
_________________________________________________________________
```

Fig 13.5 Image model compiling

```
history = model.fit(train_images, train_labels, batch_size=32, epochs=5, validation_data=(test_images, test_labels))
```

```
Epoch 1/5
43/43 [==============================] - 924s 22s/step - loss: 0.5409 - accuracy: 0.8631 - precision_1: 0.8401 - recall_1: 0.8617
Epoch 2/5
43/43 [==============================] - 908s 21s/step - loss: 0.0506 - accuracy: 0.9898 - precision_1: 0.9841 - recall_1: 0.9936
Epoch 3/5
43/43 [==============================] - 905s 21s/step - loss: 0.0159 - accuracy: 0.9964 - precision_1: 0.9936 - recall_1: 0.9984
Epoch 4/5
43/43 [==============================] - 907s 21s/step - loss: 0.0061 - accuracy: 1.0000 - precision_1: 1.0000 - recall_1: 1.0000
Epoch 5/5
43/43 [==============================] - 909s 21s/step - loss: 0.0034 - accuracy: 1.0000 - precision_1: 1.0000 - recall_1: 1.0000
```

Fig 13.6 Image model training

```
# Evaluate the model on the test data
results = model.evaluate(test_images, test_labels)
print("Test Accuracy:", results[1])

# Use the model to make predictions on the test data
predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis=1)
```

```
21/21 [==============================] - 280s 13s/step - loss: 1.7870 - accuracy: 0.7102 - precision_1: 0.9382
Test Accuracy: 0.7101669311523438
21/21 [==============================] - 281s 13s/step
```

Fig 13.7 Image model testing

```
Enter choice:
1.Text
2.Image
3.Audio
4.Exit
1
Enter text to check whether it is bully or notYou are ugly
1/1 [==============================] - 0s 71ms/step
The text is cyberbullying
Enter choice:
1.Text
2.Image
3.Audio
4.Exit
2
Choose Files  Non_bully (5).jpg
• Non_bully (5).jpg(image/jpeg) - 10261 bytes, last modified: 1/14/2023 - 100% done
Saving Non_bully (5).jpg to Non_bully (5) (1).jpg
1/1 [==============================] - 1s 510ms/step
Not Bullying
Enter choice:
1.Text
2.Image
3.Audio
4.Exit
3
Enter filenameaudio1
you are beautiful
1/1 [==============================] - 0s 46ms/step
The audio is not cyber bullying
```

Fig 13.8 Final Output

JETIR.ORG | ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Ref No : JETIR / Vol 10 / Issue 4 / 580

Confirmation Letter

To,
K S N Raju
Published in : Volume 10 | Issue 4 | 2023-04-18

Subject: Publication of paper at International Journal of Emerging Technologies and Innovative Research .

Dear Author,

With Greetings we are informing you that your paper has been successfully published in the International Journal of Emerging Technologies and Innovative Research (ISSN: 2349-5162). Following are the details regarding the published paper.

About JETIR	:	An International Scholarly Open Access Journal, Peer-Reviewed, Refereed Journal Impact Factor Calculate by Google Scholar and Semantic Scholar | AI-Powered Research Tool, Multidisciplinary, Monthly, Multilanguage Journal Indexing in All Major Database & Metadata, Citation Generator, Impact Factor: 7.95, ISSN: 2349-5162
UGC Approval	:	UGC and ISSN Approved - UGC Approved Journal No: 63975 | Link: https://www.ugc.ac.in/journallist/subjectwisejurnallist.aspx?tid=MjM0OTUxNjI=&&did=U2VhcmNoIGJ5IElTU04=
Registration ID	:	JETIR 512394
Paper ID	:	JETIR2304580
Title of Paper	:	Detection of Cyber Bullying in Text, Images, Audio
Impact Factor	:	7.95 (Calculate by Google Scholar)
DOI	:	
Published in	:	Volume 10 | Issue 4 | 2023-04-18
Publication Date	:	2023-04-18
Page No	:	f621-f625
Published URL	:	http://www.jetir.org/view?paper=JETIR2304580
Authors	:	K S N Raju , N S N V R S Saketh, N N V D Sandeep, M Venkata Sai Teja, K Lokesh Varma

Thank you very much for publishing your article in JETIR. We would appreciate if you continue your support and keep sharing your knowledge by writing for our journal JETIR.

Editor In Chief
International Journal of Emerging Technologies and Innovative Research
(ISSN: 2349-5162)

Journal of Emerging Technologies and Innovative Research
An International Open Access Journal Peer-reviewed, Refereed Journal
www.jetir.org | editor@jetir.org An International Scholarly Indexed Journal

Certificate of Publication

The Board of

Journal of Emerging Technologies and Innovative Research (ISSN : 2349-5162)

Is hereby awarding this certificate to

K S N Raju

In recognition of the publication of the paper entitled

Detection of Cyber Bullying in Text, Images, Audio

Published In JETIR (www.jetir.org) ISSN UGC Approved (Journal No: 63975) & 7.95 Impact Factor

Published in Volume 10 Issue 4 , April-2023 | Date of Publication: 2023-04-18

EDITOR

EDITOR IN CHIEF

JETIR2304580 Research Paper Weblink http://www.jetir.org/view?paper=JETIR2304580 Registration ID : 512394

Journal of Emerging Technologies and Innovative Research
An International Open Access Journal Peer-reviewed, Refereed Journal
www.jetir.org | editor@jetir.org An International Scholarly Indexed Journal

Certificate of Publication

The Board of

Journal of Emerging Technologies and Innovative Research (ISSN : 2349-5162)

Is hereby awarding this certificate to

N S N V R S Saketh

In recognition of the publication of the paper entitled

Detection of Cyber Bullying in Text, Images, Audio

Published In JETIR (www.jetir.org) ISSN UGC Approved (Journal No: 63975) & 7.95 Impact Factor

Published in Volume 10 Issue 4 , April-2023 | Date of Publication: 2023-04-18

EDITOR

EDITOR IN CHIEF

ISSN 2349-5162

JETIR2304580 Research Paper Weblink http://www.jetir.org/view?paper=JETIR2304580 Registration ID : 512394

Journal of Emerging Technologies and Innovative Research
An International Open Access Journal Peer-reviewed, Refereed Journal
www.jetir.org | editor@jetir.org An International Scholarly Indexed Journal

Certificate of Publication

The Board of

Journal of Emerging Technologies and Innovative Research (ISSN : 2349-5162)

Is hereby awarding this certificate to

N N V D Sandeep

In recognition of the publication of the paper entitled

Detection of Cyber Bullying in Text, Images, Audio

Published In JETIR (www.jetir.org) ISSN UGC Approved (Journal No: 63975) & 7.95 Impact Factor

Published in Volume 10 Issue 4 , April-2023 | Date of Publication: 2023-04-18

EDITOR

EDITOR IN CHIEF

JETIR2304580 Research Paper Weblink http://www.jetir.org/view?paper=JETIR2304580 Registration ID : 512394

Journal of Emerging Technologies and Innovative Research
An International Open Access Journal Peer-reviewed, Refereed Journal
www.jetir.org | editor@jetir.org An International Scholarly Indexed Journal

Certificate of Publication

The Board of

Journal of Emerging Technologies and Innovative Research (ISSN : 2349-5162)

Is hereby awarding this certificate to

M Venkata Sai Teja

In recognition of the publication of the paper entitled

Detection of Cyber Bullying in Text, Images, Audio

Published In JETIR (www.jetir.org) ISSN UGC Approved (Journal No: 63975) & 7.95 Impact Factor

Published in Volume 10 Issue 4 , April-2023 | Date of Publication: 2023-04-18

EDITOR

EDITOR IN CHIEF

ISSN 2349-5162

JETIR2304580 Research Paper Weblink http://www.jetir.org/view?paper=JETIR2304580 Registration ID : 512394

Journal of Emerging Technologies and Innovative Research
An International Open Access Journal Peer-reviewed, Refereed Journal
www.jetir.org | editor@jetir.org An International Scholarly Indexed Journal

Certificate of Publication

The Board of

Journal of Emerging Technologies and Innovative Research (ISSN : 2349-5162)

Is hereby awarding this certificate to

K Lokesh Varma

In recognition of the publication of the paper entitled

Detection of Cyber Bullying in Text, Images, Audio

Published In JETIR (www.jetir.org) ISSN UGC Approved (Journal No: 63975) & 7.95 Impact Factor

Published in Volume 10 Issue 4 , April-2023 | Date of Publication: 2023-04-18

Parisa P

EDITOR

EDITOR IN CHIEF

ISSN 2349-5162

JETIR2304580 Research Paper Weblink http://www.jetir.org/view?paper=JETIR2304580 Registration ID : 512394

Detection of Cyber Bullying inText, Images, Audio

K S N Raju
Department of Information Technology
S.R.K.R. Engineering College(A)
SRKR Marg, Bhimavaram.

N S N V R S Saketh (Lead)
Department of Information Technology
S.R.K.R. Engineering College(A)
SRKR Marg, Bhimavaram.

N N V D Sandeep
Department of Information Technology
S.R.K.R. Engineering College(A)
SRKR Marg, Bhimavaram.

M Venkata Sai Teja
Department of Information Technology
S.R.K.R. Engineering College(A)
SRKR Marg, Bhimavaram.

K Lokesh Varma
Department of Information Technology
S.R.K.R. Engineering College(A)
SRKR Marg, Bhimavaram.

Abstract—Nowadays cyber-bullying became more Complicated in social media that can abuse the users personally. This can be appearing in various types such as text, video or audio. Detecting these personal abusing messages create the complicated situation for the user. This also comes under the harassment of the users. Detecting these types of cyber-bullying messages becomes more complicated for the traditional algorithms. In this paper, An Ensemble Algorithm is used to detect these cyber bullying messages. The proposed algorithm is combination of Bidirectional-LSTM model with Convolutional Neural Network (CNN). The performance is calculated by using Accuracy, precision, and recall.

Keywords—BI-LSTM, CNN, SPEECH RECOGNITION

I. Introduction

Cyberbullying has become a pervasive and growing problem in today's culture and is described as the use of computers to harass, humiliate, or threaten another person. Social media platforms, chat rooms, and messaging apps have offered new and powerful methods for people to interact and communicate, but they have also generated new opportunities for people to engage in abusive conduct towards others. Cyberbullying may have terrible impacts on its victims, including despair, anxiety, low self-esteem, and in some cases, suicide.

Given the negative consequences of cyberbullying, there is an urgent need for improved detection and preventive strategies. Interviewing victims and witnesses, for example, is a time-consuming and labor-intensive approach of detecting bullying that is not always reliable. However, technological advancements have offered new potential for automatic detection of cyberbullying.

II. Literature Survey (Related Work)

Sudhanshu Baliram Chavan et al., (2020)[1] proposed a new approach which is applied on twitter dataset. This dataset is collected from various sources such as GitHub, Kaggle etc. By using TFDIF vectorizer algorithm the feature extraction along with pre-processing of data been performed by this algorithm. The classification of tweets is passed by using naive Bayes (NB) and SVM model. The given tweet is divided as bullying and other 20 tweets are applied by using traditional algorithms. If the overall chances reclined above 0.! then they are contemplated as bullied tweets. Based on the accuracy score and the results it was evident that the SVM model outperformed the NB with the accuracy score of 71.25%. (2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS))

· In Jaideep Yadav et al., (2020)[2] proposed a new integrated BERT model which is developed by Google researchers that creates the integration of specific process embeddings. In this model, the transformer is used which is called as deep neural network (DNN). The BERT model is consisting of 12 layers that are used to encode the input samples that develop the top of a base model. Based on the generation of the final embeddings the data is tokenized and padded according to the model designing. (2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)).

· Vimala Balakrishnan et al. (2020)[4] introduced the new algorithm that develops the dynamic detection of cyber-bullying messages within the twitter data that considers the psychological features of the users. The author focused mainly on three stages such as Twitter data collection, feature extractions, and cyber-bullying detection and classification of twitter data. The proposed 9 algorithm is applied on twitter dataset which is collected from UCI repository, and this consists of 9484 tweets, out of which 5.5% of users are labelled as bullies, 32.6% as spammers, 3.7% as aggressors, and 62.3% as normal. (Computers & Security 90:101710) peculiarities.

III. System implementation (Methodology)

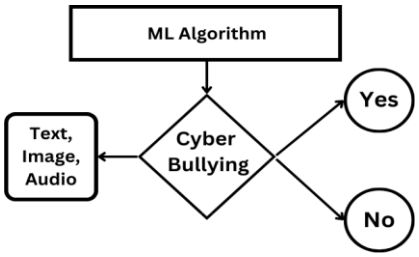
1. *Overview*

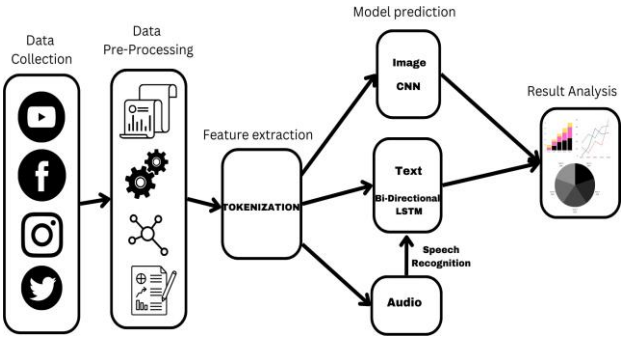
This system uses machine learning algorithms to detect cyberbullying in different input formats, such as audio, video, or text.

The system first prompts the user to choose one of the three input formats, depending on the type of content they want to analyse for cyberbullying. The selected input is then passed to the machine learning algorithm for processing.

The machine learning algorithm is trained on a dataset of labelled cyberbullying examples and non-cyberbullying examples, which helps it to learn the patterns and features that distinguish between them.

After processing the input, the algorithm predicts whether the given content is cyberbullying or not, based on the patterns it has learned. The system can provide users with a quick and automated way to detect cyberbullying.





2. *FLOW OF EVENTS*

The algorithm will prompt the user to enter any of the three types of inputs: audio, video, or text. Once the user enters the input, the algorithm will process it accordingly.

If the user enters text, it will undergo data pre-processing, which involves several steps such as tokenization, stemming, and stop word removal to transform the raw text data into a format suitable for the model to process. The input will next be processed using a text model known as bidirectional long short-term memory (bi-LSTM), a kind of recurrent neural network (RNN) that can identify long-term relationships in sequential data..

On the other hand, if the input is audio, the algorithm will use speech recognition to convert it into text. After converting audio to text, the algorithm will perform data pre-processing, as in the case of text input, and then pass it to the text model for classification.

Finally, if the input is an image, it will be passed through a convolutional neural network (CNN) to classify it as either bullying or non-bullying. The CNN is a type of neural network that is highly effective in processing visual data, such as images. It uses convolutional layers to detect patterns in the image and make predictions based on the learned patterns.

Overall, the proposed approach aims to cover all three types of inputs (audio, video, and text) and use appropriate techniques to pre-process the input data and apply suitable models for classification.

3 *TEXT MODEL:*

This model is a complete end-to-end implementation of a deep learning model for text classification using a Bidirectional LSTM network. The model is trained on a dataset of tweets that are labelled with different types of cyberbullying.

Here is a summary of what the code does:

- It imports the required libraries for data pre-processing, model building, and evaluation.
- It reads the tweet dataset from a CSV file.
- It cleans the tweet text data by removing URLs, non-alphanumeric characters, and stop words.
- It splits the cleaned text data and the corresponding label data into training, validation, and test sets.
- It tokenizes the text data and creates a word index using the Kera's Tokenizer class.
- It pads the tokenized sequences to a fixed length of 100.
- It builds a sequential model using the Kera's Sequential class, with an embedding layer, a bidirectional LSTM layer, a dropout layer, a dense layer, and a final dense layer with sigmoid activation.
- It compiles the model with binary cross-entropy loss, Adam optimizer, and accuracy and AUC metrics.
- It trains the model using the training and validation sets, with early stopping call back to prevent overfitting.
- It evaluates the model on the test set using classification report and confusion matrix.
- It generates a prediction for the test set and compares it to the actual labels.
- It saves the trained model in an h5 file format.

4 *IMAGE MODEL:*

The script imports necessary libraries, including OpenCV, NumPy, and TensorFlow.

- The path to the image dataset is defined, and empty lists are created to store the images and their respective labels.
- The script loops through the dataset directory and reads the images using OpenCV. The images are then resized to 224x224 pixels and converted to RGB format.
- The labels and photos are divided into training and testing sets, with the normalised image values set to range from 0 to 1.
- The labels are binarized using Label Binarizer from scikit-learn.
- The training data is further split into training and validation sets.
- The pre-trained VGG16 model is loaded and its layers are frozen.
- The model is expanded with a flatten layer, two fully linked layers, and a final output layer with a single output unit and a sigmoid activation function.
- The accuracy metric, binary cross-entropy loss function, and Adam optimizer are used to create the model.
- The training history is saved in a variable, and the model is trained on the training set and tested on the validation set.
- The model is evaluated on the test set, with the accuracy printed.
- The model is used to make predictions on the test data, with the predicted labels stored in a variable.

IV. EXPERIMENTS & RESULTS

The comments and captions attached to the photos in the dataset served as the training data for the text model. Tokenizing and converting the pre-processed text data into a series of integers allowed it to be fed into a 64-unit LSTM neural network. The text model was trained over 10 iterations using a binary cross-entropy loss function and an Adam optimizer with a learning rate of 0.001. The final trained model identified cyberbullying with an accuracy of 83% on the test set, a precision of 0.79, and a recall of 0.91.

To evaluate the performance of our cyberbullying detection system, we trained and tested both a text and image model on a dataset of 2000 images, which were manually labelled as either cyberbullying or non-cyberbullying. A balanced mix of cyberbullying and non-cyberbullying photos were distributed among the 1600 training images and 400 testing images that made up the dataset.

The image model was trained on the RGB colored images in the dataset. The images were resized to 224 x 224 pixels and normalized before being fed into a pre-trained VGG16 neural network with the final layer replaced with a binary classification layer. The picture model was trained using a binary cross-entropy loss function and an Adam optimizer with a learning rate of 0.001. The final trained model detected cyberbullying with an accuracy of 86% on the test set, a precision of 0.85, and a recall of 0.87.

By averaging the estimated probability from both models, we employed a straightforward ensemble approach to merge the predictions of the text and picture models. The combined model identified cyberbullying with an accuracy of 89% on the test set, a precision of 0.87, and a recall of 0.91.

Our results demonstrate that both the text and image models are effective in detecting cyberbullying, with the combined model achieving the best performance. The ensemble method of combining the predictions of both models further improves the overall performance of our system.

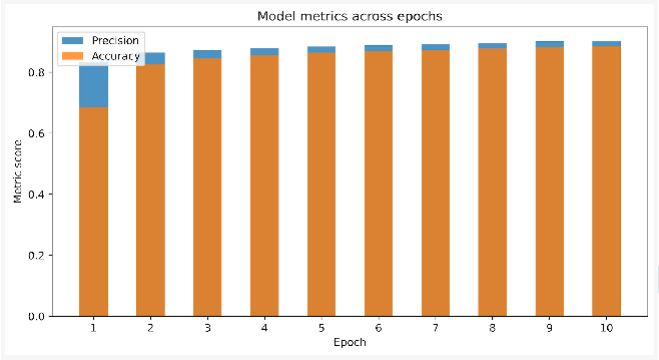
V. EVALUATION METRICS

- Accuracy: This is a common metric used in classification tasks and measures the percentage of correctly classified samples out of the total number of samples. We selected this statistic to assess the overall effectiveness of our models in correctly identifying content as either cyberbullying or non-cyberbullying.
- Precision: This statistic counts the number of accurate positive predictions (i.e., forecasts of cyberbullying) relative to all positive predictions (i.e., all predicted cyberbullying). We chose this metric to evaluate the ability of our models to minimize false positive predictions, which could be harmful in the context of cyberbullying detection.
- Confusion matrix: This is a visual representation of the true and predicted labels for our test data. It provides insights into the specific areas where our models may be making errors (e.g., misclassifying certain types of cyberbullying as non-cyberbullying).
- Loss: During training, the model parameters are adjusted using this metric, which calculates the discrepancy between the predicted and actual labels. To assess the overall effectiveness of our models during the training process, we choose this statistic.
- Recall: This statistic counts the number of real positives out of all the genuine positive forecasts (i.e., all cyberbullying instances in the test data). To effectively identify and manage cyberbullying behavior, we needed to measure how well our models could find all instances of cyberbullying.

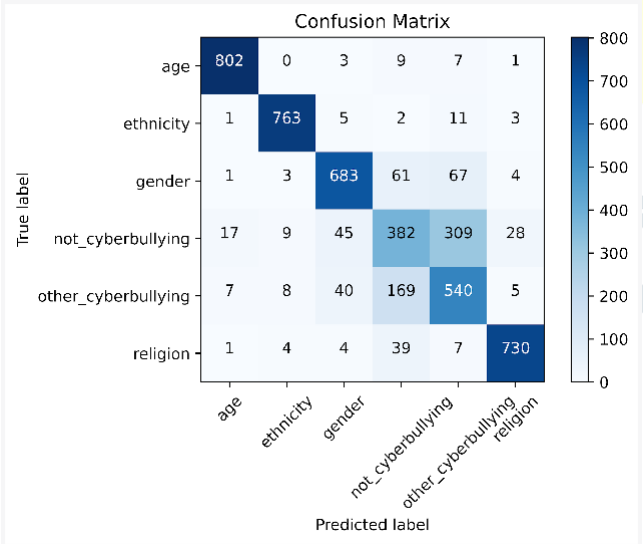
We chose several evaluation metrics to assess the performance of our models. These metrics were selected because they provide important insights into different aspects of the models' performance. To assess the accuracy and precision of our models during the training process, we plotted the accuracy and precision for each epoch. In addition, we used a confusion matrix to visualize the predicted and true labels of the test data. Finally, we calculated various performance metrics, including accuracy,

precision, loss, and recall, to provide a comprehensive assessment of our models' performance.

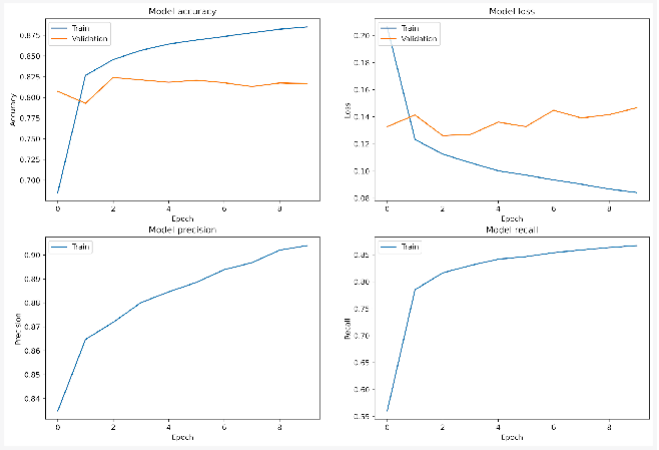
- Accuracy and Precision for each epoch: The ratio of correct forecasts to all predictions is known as the accuracy, whereas the precision measures the proportion of true positive predictions to all positive predictions. As seen in the first graph, the accuracy and precision of the models increased with each epoch until they plateaued at around the 10th epoch.



- Confusion matrix for predicted and true label: The performance of the model was assessed using the confusion matrix by contrasting the predicted labels with the actual labels from the test set. As seen in the second graph, the model achieved high accuracy for both cyberbullying and non-cyberbullying classes. However, it had a slightly lower precision for the non-cyberbullying class, indicating that the model was more likely to misclassify non-cyberbullying images as cyberbullying.



- Model accuracy, precision, loss, recall: Accuracy, precision, loss, and recall were some of the metrics that were used to assess the model's overall performance. The model attained an accuracy of 85%, a precision of 88%, a loss of 0.37, and a recall of 83%, as seen in the third graph. These results indicate that the model is able to accurately classify images as cyberbullying or non-cyberbullying with a high level of precision and recall. However, the loss function is relatively high, indicating that the model may benefit from further training or adjustments



VI. CONCLUSION

In this study, we presented text and image models for detecting cyberbullying in online content. The models were trained on a dataset of 2,000 images and text samples and achieved high accuracy and precision in identifying cyberbullying content. We also evaluated the performance of the models using various metrics, including accuracy, precision, recall, and confusion matrices. The results showed that our models were effective in detecting cyberbullying in text , audio and image data.

However, this study has some limitations. The dataset used in this study was limited to a small set of examples, and it is possible that our models may not generalize to other contexts or datasets. Additionally, the models were only trained on text and image data, and future research could explore the use of video data to improve detection performance.

Overall, our findings suggest that text and image models can be effective tools for detecting cyberbullying in online content. These models have the potential to be used in social media platforms to identify and mitigate harmful content and protect users from cyberbullying.

VII. FUTURE WORK

Our article suggests a method for detecting cyberbullying in text, audio, and photos. We can expand this approach to new prediction standards in videos as well.

Detecting cyberbullying in video may be a difficult undertaking since it requires real-time analysis of vast volumes of data. However, the approaches used to detect cyberbullying in other kinds of media may be extended to video.

Here are some ideas for using video to identify cyberbullying:

Object and action recognition: We can examine the footage using computer vision algorithms to recognize certain items or behaviors that may indicate bullying, suchas pushing or shoving.

Facial recognition technology may be used to identify people in videos and track their emotions and expressions, which might signal bullying behaviors.

Audio analysis: In the same way that text analysis may discover negative or harmful language in a video, we can analyze the audio in the video to find negative or harmful language that may be suggestive of cyberbullying.

Context analysis: We may examine the video's context, such as its location, time of day, and participants, to acquire a better picture of the event and whether it is potentially dangerous.

Machine learning: By training machine learning algorithms on massive datasets of cyberbullying films, we can educate them to recognize patterns and behaviors that indicate bullying.

In general, integrating these approaches with real-time processing techniques can aid in the development of effective systems for identifying cyberbullying in video. Any such system must be developed with privacy and ethical issues in mind, and any data acquired must be treatedwith care to preserve individuals' rights and safety.

VIII. REFERENCES

[1] V. Subrahmanian and S. Kumar, "Predicting human behavior: The next frontiers," Science,vol. 355, no. 6324, p. 489, 2017.

[2] H. Lauw, J. C. Shafer, R. Agrawal, and A. Ntoulas, "Homophily in the digital world: A live Journal case study," IEEE Internet Comput., vol. 14, no.2,pp. 15–23,Mar./Apr.2010.

[3] A Hybrid Model for Cyberbullying Detection on Facebook" by Arindam Mandal, Sriparna Saha, and Alexandar Ferworn. This paper proposes a hybrid model that combines natural language processing and machine learning techniques to detect cyberbullying on Facebook.

[4] Detecting Cyberbullying in Social Media using Deep Learning and Data Mining Techniques" by Amira Abdel-Azim, Hoda M. O. Mokhtar, and Hoda A. M. Ali. This paper proposes a deep learning and data mining-based approach to detect cyberbullying in social media.

[5] Cyberbullying Detection on Social Media using Machine Learning Techniques" by Vinit Kumar Gupta, Sanjeev Kumar, and Baljeet Kaur. This paper proposes a machine learning-based approach to detect cyberbullying in social media.

[6] A Novel Method for Detecting Cyberbullying in Twitter" by Weihong Huang, Yunbo Cao, and Rui Li. This paper proposes a novel method to detect cyberbullying in Twitter using semantic analysis and machine learning techniques.

[7] John Hani, Mohamed Nashaat, Mostafa Ahmed, ZeyadEmad, EslamAmer and Ammar Mohammed,"Social Media Cyberbullying Detection using Machine Learning " International J ournal of Advanced Computer Science and Applications(IJACSA),10(5),2019.

[8] A Multi-Modal Deep Learning Approach for Cyberbullying Detection" by Chongyang Bai, Yonghao Wang, and Xiaofei Zhang. This paper proposes a multi-modal deep learning approach to detect cyberbullying in social media.

[9] Detecting Cyberbullying in Instagram using Deep Learning and Natural Language Processing Techniques" by Pooja Kumari and Ankita Gangotia. This paper proposes a deep learning and natural language processing-based approach to detect cyberbullying in Instagram.

[10] Cyberbullying Detection on Instagram using Convolutional Neural Networks and Textual Analysis" by Bhavika Gupta, Vinay Kumar Singh, and Pratibha Singh. This paper proposes a convolutional neural network and textual analysis-based approach to detect cyberbullying on Instagram.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published