Stress detection in IT professional by image processing and machine learning

PROJECT REPORT

Submitted By

DONA MARIYA DAVIES

Reg. No. CCAWMCS001

For the award of the Degree of

Master of Science

in Computer Science

(University of Calicut)

under the guidance of

Ms. Viji Viswanathan

Assistant Professor

M.Sc in COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
CHRIST COLLEGE (AUTONOMOUS)
IRINJALAKUDA, KERALA
2022-2024

ACKNOWLEDGMENT

Submitting my project in the divine feet of Almighty God.I would like to take this opportunity to express my profound gratitude to all the people who have inspired and motivated to take this project success.

I would like to thank our principal Rev.Fr.Dr. Jolly Andrews CMI, for proper ambience to go on with the project. I take these opportunities to acknowledge my thanks to Ms Sini Thomas, Head of the Department of Computer Science for valuable guidance in developing this project. I' am also thankful to my Project Guide Ms Viji Viswanathan, and all other staff in Department of Computer Science for their immense support

My sincere thanks to all those well wishers and friends who have helped me during the course of the project work and have been making it a great success. Last but not least, I wish to express my thankfulness to my family members for their excellent support and co-ordination.

Date : **DONA MARIYA DAVIES**

Declaration

I hereby declare that the project entitled **"Stress detection in IT professional by image processing and machine learning"** submitted to Calicut university, on partial fulfillment of the requirement for the award of degree in Master of Science in Computer Science is a record of original work done by me, under the guidance of Ms Viji Viswanathan , Assistant Professor in Department of Computer Science, Christ College(Autonomous), Irinjalakuda.

Place : Irinjalakuda **Name: DONA MARIYA DAVIES**

Date : **RegNo: CCAWMCS001**

DEPARTMENT OF COMPUTER SCIENCE

Christ College(Autonomous)

Irinjalakuda

<u>CERTIFICATE</u>

Certified that this thesis entitled ***'Stress detection in IT professional by image processing and machine learning'*** *submitted by "***'DONA MARIYA DAVIES (Reg. No. CCAWMCS001)***" in partial fulfillment for the award of the degree of Master of Science in Computer Science under University of Calicut during the year 2022-2024, is the bonafide work carried out by her under my guidance and supervision.*

Ms Viji Viswanathan
Assistant Professor, CSE
Internal Guide

Ms. Sini Thomas
Head of the Department
Computer Science

EXTERNAL EXAMINER

INTERNAL EXAMINER

Abstract

Detecting stress in the IT professionals with the help of Machine learning and Image processing techniques is an upgraded version of the old stress detection systems which excluded the live detection and the personal counseling but this paper comprises of live detection and periodic analysis of employees and detecting physical as well as mental stress levels in his/her by providing them with proper remedies for managing stress by providing survey form periodically. This paper mainly focuses on managing stress and making the working environment healthy and spontaneous for the employees and to get the best out of them during working hours.

Contents

List of Figures

INTRODUCTION

Chapter 1

1 Introduction

1.1 Overview

Stress management systems play a major role to notice the stress levels that disrupts our socio-economic mode. As World Health Organization (WHO) says, Stress may be a psychological state drawback moving the lifetime of one in four voters. Human stress results in mental furthermore as socio-fiscal issues, lack of transparency in work, poor operating relationship, depression and eventually commitment of suicide in severe cases. This demands counselling to be provided for the stressed people cope up against stress. Stress turning away is not possible however preventive actions helps to beat the stress. Currently, solely medical and physiological consultants will verify whether or not one is beneath depressed state (stressed) or not. one in every of the normal methodology to notice stress is predicated on form. This methodology, utterly depends on the answers given by the people, folks are going to be unsteady to mention whether or not they square measure stressed or traditional. Automatic detection of stress minimizes the chance of health problems and enhance the welfare of the society. This covers the manner for the need of a scientific tool, that uses physiological signals thereby automating the detection of stress levels in people. Stress detection is mentioned in varied literatures because it may be a vital social contribution that enhances the approach to life of people. Nowadays because IT industries square measure setting a replacement peek within the market by transferal new technologies and merchandise within the market. during this study, the stress levels in staff also are noticed to lift the bar high. Although their square measure several organizations United Nations agency give psychological state connected schemes for his or her staff however the problem is much from management. during this paper we have a tendency to try and go into the depth of this drawback by making an attempt to notice the stress patterns within the operating worker within the corporations we might prefer to apply image process and machine learning techniques to research stress patterns and to slim down the factors that powerfully verify the stress levels. Machine Learning algorithms like KNN classifiers square measure applied to classify stress. Image process is employed at the initial stage for detection, the employee's image is clicked by the camera that is input. so as to urge associate degree increased image or to extract some helpful info from its image process is employed by changing image into digital type and play acting some operations on that. By taking input as a picture from video frames and output is also image or characteristics related to that image.

1.2 Project Profile

Title	: Stress detection in IT professional by image processing and machine learning
Domain	: Deep Learning
Language	: Python
Version	: 3.10.0

1.3 Contributions

The major contributions of this system are:

- The system detects stress in IT professionals using machine learning and image processing techniques, providing live detection and periodic analysis of employees' physical and mental stress levels.

- The system employs a non-intrusive and real-time approach to detect stress by analyzing facial expressions in captured videos and also through a heart rate sensor. It uses image processing and machine learning techniques to analyze emotional status and make decisions on stress levels.

PROBLEM DEFINITION AND METHODOLOGY

Chapter 2

2 Problem Definition and Methodology

2.1 Problem definition

Stress is a complicated psychological and physiological condition that differs greatly from person to person. What stresses one person out could not stress another out. It is challenging to precisely define and identify stress in a variety of individuals due to its subjectivity.It can be challenging to recognize faces at times.Natural variations in heart rate can occur over the day as a result of things like physical activity, coffee intake, and mental states unrelated to stress. It might be difficult to distinguish between heart rate variations that are typical and those brought on by stress.

2.2 Objectives

The main highlight of this is to propose a reliable, convenient, and accurate stress detection system for IT professionals. It aims to monitor the emotional status of individuals working in front of a computer for longer durations and detect and reduce stress levels.

2.3 Motivation

Stress management systems, utilizing heartbeat sensors and image processing, aim to detect and manage stress levels in IT professionals, thereby improving productivity and preventing health problems, thus enhancing societal welfare.Detecting stress is in such a way:

- Recognise facial emotion
- Heart beat sensor
- Susceptibility to detection

2.4 Methodology

The study uses image processing and machine learning techniques to analyze stress patterns in IT professionals. It uses image acquisition tools and KNN classifiers to classify stress levels. The system captures images and records heartbeats, aiming to monitor emotional status and provide solutions to reduce stress and create a comfortable workplace.

2.5 Scope

Stress detection in machine learning involves analyzing physiological signals (e.g., heart rate,facial emotion) and behavioral data to infer stress levels. It has applications in healthcare, wearable devices, and stress management tools, aiming to provide real-time insights for users' well-being and performance optimization. Advancements in algorithms, sensor technologies, and data fusion techniques are expanding its scope for personalized interventions and preventive healthcare measures.

REQUIREMENT ANALYSIS AND SPECIFICATION

Chapter 3

3 Requirement Analysis and Specification

3.1 Requirement Analysis

Stress detection in machine learning involves analyzing physiological signals (e.g., heart rate,facial emotion) and behavioral data to infer stress levels. It has applications in healthcare, wearable devices, and stress management tools, aiming to provide real-time insights for users' well-being and performance optimization. Advancements in algorithms, sensor technologies, and data fusion techniques are expanding its scope for personalized interventions and preventive healthcare measures.

3.2 Existing System

Existing system is developed for stress detection based on the study of the facial expression. The system is nonintrusive and is able to run in real-time. This system consists some Image Processing and Machine Learning Techniques. And the other work on this issue is based on various physical signals and visual features to monitor the stress in a person while he is working. However, these measurements are invasive and are less comfortable in real application. Every sensor data is compared with a stress index which is a threshold value used for detecting stress.

3.3 Proposed System

Stress detection system based on the analysis of the facial expression. The system works when the IT professional will be seat in the front of camera then it will be able to detect the facial expression and run in real-time. A camera is used to capture the near front sight of the employee while he is working in front of the computer. Captured video is divided into sections of equivalent length and set of similar number of image frames are extracted from each part correspondingly and are examined. The image detection includes the calculation of the variation in the place of the eyebrow from its mean position. The displacement of eyebrow from its place is considered by examining the image for the eyebrow co-ordinates. If the employee is found stressed in the successive sections of time intervals which was previously divided, the decision for stress detection is formed for a employee working in front of computer with the obtained results it employs the technique of deep learning. The stress detection module scans the binary image from the extreme left top to record the coordinates of the eyebrow. The stress detection module scans the binary image from the extreme left top to record the coordinates of the eyebrow. The offline displacement calculation sub-module calculates the shifting of eyebrow using the obtained eyebrow co-ordinates which is subsequently followed by variance calculation of the displacement. The classifier sub-module is trained offline are employed to determine the presence of emotion. The integrated decision of individual frames eventually determines the level of stress involved.

- Accuracy -The accuracy of the present system can be improved.
- Model loss - the loss of the system can be reduced.
- Computation time - The computation time can be reduced and can be made.

3.4 Requirement Specification

3.4.1 Functional Requirements

In software engineering and system engineering, functional demand defines function of a system and its factors. A function is described as a set of inputs, the geste and labors. Functional conditions may be computations, specialized details, data manipulation and processing and other specific functionality that define what a system is supposed to negotiate. Behavioral conditions describing all the cases where the system uses the functional conditions are captured in use cases. Functional conditions are supported by non-functional conditions also known as quality conditions, which put constraints on the design or perpetration (similar as performance conditions, security, or trustability). Generally, functional conditions are expressed in the form " system must do demand ", whilenon-functional conditions are " system shall be demand ". The plan for enforcing functional conditions is detailed in the system design. The plan for enforcing non-functional conditions is detailed in the system armature. As defined in conditions engineering, functional conditions spec ify particular results of a system. This should be varied with inoperative conditions which specify overall characteristics similar as cost and trustability. Functional conditions drive the operation armature of a system, whilenon-functional conditions drive the specialized armature of a system. This system does:

Stress detection in machine learning involves analyzing physiological signals (e.g., heart rate, facial emotion) and behavioral data to infer stress levels

3.4.2 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing nonfunctional requirements is detailed in the system architecture, because they are usually Architecturally Significant Requirements. Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of "system shall do requirement", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of "system shall be requirement", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioral requirements".. Qualities—that is non-functional requirements—can be divided into two main categories: Execution qualities, such as safety, security and usability, which are observable during operation (at run time). Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system.

3.5 Feasibility Study

3.5.1 Technical Feasibility

Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with python anaconda installed. Today every organization has computer, so it is not an extra cost.

3.5.2 Economical Feasibility

Economic feasibility is the most frequently used method for evaluating the effectiveness of proposed system.The proposed model is cost effective.

3.5.3 Operational feasibility

The combined approach of machine learning and Image capturing provide better security and confidentiality to the data.

3.6 Software Requirement Specification

3.6.1 Introduction

Purpose

The stress detection system monitors IT professionals' emotional state, reducing stress levels. Utilizing image processing and machine learning, it analyzes stress patterns and identifies stress-determining factors. The system offers preventative solutions, including live stress detection, and remedies, aiming to improve IT professionals' well-being and productivity

Document Conventions

- All terms are in Times New Roman style.
- Main features or important terms are in bold.
- Use LateX for documentation.

Intended Audience and Reading Suggestions

Anyone with some programming experience, with familiarity in Python and Deep learning, can understand this document.The document is intended for developers, software architects, testers, project managers and documentation writers. This Software Requirement Specification also includes:

- Overall description of the product
- External interface requirements
- System Features
- Other nonfunctional requirements

Product Scope

The stress detection system uses image processing and machine learning techniques to monitor stress levels in IT professionals. It captures images of employees and analyzes them regularly to identify physical and mental stress levels. The system uses machine learning algorithms like KNN classifiers to classify stress levels. The goal is to create a comfortable, healthy workplace environment, manage stress, and provide preventative solutions to improve the overall well-being and productivity of IT professionals during working hours.

IEEE Standard 830-1998 Recommended Practice for Software Requirements Specifications.

3.6.2 Overall Description

Product Perspective

The stress detection system for IT professionals has been upgraded to include live detection and personal counseling. It uses image processing and machine learning techniques to analyze stress patterns and identify factors determining stress levels. The system monitors the emotional status of IT professionals working long hours in front of a computer, aiming to create a comfortable workplace environment. Regular images of employees are captured and analyzed using machine learning algorithms like KNN classifiers. The system provides preventative stress management solutions, aiming to improve overall well-being and productivity. The system aims to reduce stress levels and create a healthy working environment.

Operating Environment

- Operating System: Windows 11
- Processor: Intel Core i3 / AMD Ryzen 3 or Higher
- Memory: 4GB or more

3.6.3 Design and Implementation Constraints

- Computational Complexity.
- Training time.
- Integration Challenges

3.6.4 Assumptions and Dependencies

Assumptions

The proposed approach assumes a dataset that encompasses a wide range of flower species and is thoroughly annotated is readily accessible for training models. The efficacy of the model hinges upon the inclusiveness and excellence of this dataset. A diverse and well-annotated dataset that includes various flower species is at one's disposal for the purpose of model training.

Dependencies

- Python
- CUDA and cuDNN (Optional)

3.6.5 External Interface Requirements

User Interfaces

The interface should maintain privacy and security standards, ensuring user data protection and confidentiality throughout the stress detection and management process.The design facilitates finishing the task at hand without drawing unnecessary attention to itself. Image processing and machine learning are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the Interface.The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Hardware Interfaces

- Operating System: Windows or any other Platform
- Hardware: Intel Core i5
- Internet Connection

Software Interfaces

- Python

Communications Interfaces

Standard HTTP COMMUNICATION interface required for internet connection.

3.6.6 System Features

- The data detected by KNN Classifier, Deep Learning and image processing.
  Utilizes the deep learning architecture pre-trained on ImageNet for effective feature extraction and representation learning, enhancing the model's ability to classify emotions.
- Transfer Learning:
  Leverages transfer learning to fine-tune the Inception V3 model on a specific flower dataset, benefiting from the knowledge learned during pre-training on ImageNet
- Functional Requirements
  The computer vision data is sent to the deep learning model for obtaining the result. The deep learning model will be compared with the extracted features of given data. And then finally during the comparison will obtain the result whether data is hated or not hated speech.

3.6.7 Other Nonfunctional Requirements

Performance Requirements

- Accuracy: The classification accuracy of the system must meet or exceed predefined standards, reflecting the model's capability to correctly identify flower species. High accuracy is crucial for the system's reliability and utility in various applications.
- Resource Utilization: Efficient utilization of computational resources, including CPU, GPU, or TPU, is necessary to ensure optimal performance during training and inference. The system should manage resource allocation effectively, avoiding bottlenecks or overutilization.
- Training Time: The time required for model training should be within acceptable limits, allowing for timely updates and improvements. Efficient training times are especially important when retraining the model with new data or making enhancements.
- Compatibility: Should be compatible with a wide range of devices, operating systems and platforms, to ensure its widespread adoption and use.

SYSTEM DESIGN

Chapter 4

4 System Design

4.1 Users of the System

User: The user who handles the System.

4.2 Modularity criteria

The proposed system has following modules :

- Data entered undergoes image processing and machine learning.
- Integration of different algorithms and tools, such as KNN classifiers, to classify stress patterns based on analyzed images.
- 3. System to be adaptable to different environments and organizations, ensuring that it can be customized and tailored to specific needs.

4.3 Design Methodologies

Image Pre-processing

Collecting a diverse dataset of images that can be used for flower image classification is crucial. The images should be high quality and varied in content to test the robustness of the proposed approach. The collected data should be preprocessed to remove any irrelevant or corrupted data that could impact the performance of the approach. This can involve removing noise, compression artifacts, and other types of image distortions. The data should be in a format that is compatible with the proposed approach. Feature extraction is a critical step in image processing, as it involves identifying and extracting the features of the image that can provide valuable insights. This can involve identifying color palettes, pixel patterns, and other features of an image. The preprocessing methods of resizing, normalization, and augmentation are frequently used. By ensuring that every image has the same size, resizing makes it possible to use constant input dimensions across the dataset. In order to aid in convergence during model training, normalization is done to scale pixel values, usually to a specified range like [0, 1]. By performing changes like rotation, zooming, and flipping to the original photos, augmentation creates variations of the original images that enhance the model's generalization abilities and diversify the training dataset.Image preprocessing in fact plays a crucial role in optimizing the input data quality, leading to more robust and accurate machine learning model performance in image-related tasks.

HEARTBEAT SENSOR(MAX30102)

A digital pulse oximeter and heart rate sensor is an electronic device which can measure the heart rate of a person by measuring the difference between oxygen rich and oxygen less blood. Not only heart rate, this device can also measure the concentration of oxygen in blood. The MAX30102 is a very versatile sensor and it can also measure body temperature other than heart rate and blood oxygen level. This is a sensor designed by Analog Devices and features two LEDs (one Infrared and one Red), a photodetector, optics, and low-noise signal processing unit to detect pulse oximetry (SpO2) and heart rate (HR) signals. The main idea is that shine a single LED at a time and check the amount of light that is getting reflected back to the sensor. Based on the reflection it determines the blood oxygen level and heart rate.

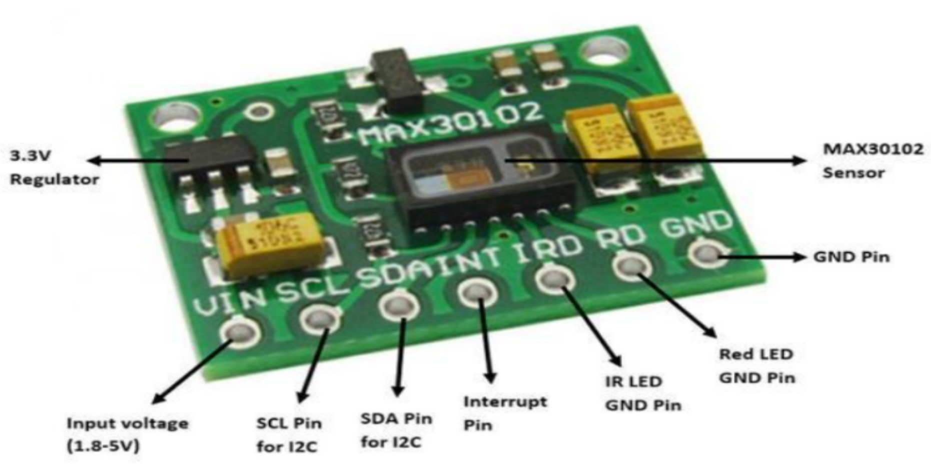


Figure 1: HEARTBEAT SENSOR

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a powerful supervised machine learning algorithm used for classification and regression tasks. It finds the nearest data points to a query point using distance metric like Euclidean distance, assigning the class label to the majority. In regression, KNN calculates the average of these nearest data points to predict the continuous value associated with the query point. Key features include simplicity, multi-class classification, and adaptability to non-linear data. However, KNN can be computationally expensive and sensitive to irrelevant or redundant features.

4.4 User Interface Layouts

User interface design (UI) or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the Interface.The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

IMPLEMENTATION AND MAINTENANCE

Chapter 5

5 Implementation

5.1 Tools/Scripts for Implementation

5.1.1 Python

Python is a high-level, interpreted programming language that is designed to be easy to read and write. Python is known for its simple syntax, which allows programmers to write code quickly and easily. is an object-oriented language, which means that it supports the creation of objects and classes that can be used to build complex applications. Python has a large standard library that includes modules for a wide range of tasks, from web development and database management to scientific computing and artificial intelligence. There are also many third-party libraries available for Python that provide additional functionality and support for specific tasks. Python is popular among developers because it is easy to learn and use, and it can be used for a wide range of applications.

5.2 Module hierarchy

- **Data entered undergoes Machine Learning**
  The stress detection system uses image processing techniques to preprocess IT professionals' captured images. The images are analyzed, manipulated, resized, and converted into digital format. These images are then used for machine learning algorithms to classify stress levels.

- **Model is being compiled with the pre-processed image set**
  The weight of the base model is being loaded and compiled and optimized using the adam optimizer. And the model is being set for training and is being fitted.

5.3 Coding

Python

Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. Python's syntax is straightforward, making it easy to read and write, and it has a vast standard library that provides modules for a wide range of tasks, including web development, scientific computing, artificial intelligence, and more.

5.4 Problems Encountered

The following are some of the problems faced in this system:

- Computational Complexity.
- Continues capturing of images creates large unusable datasets.
- Auto captured image datasets detection will get more time consuming or inaccurate.

TESTING AND IMPLEMENTATION

Chapter 6

6 Testing And Implementation

6.1 Test Plans

A test plan documents strategy that will be used to verify and ensure that a product or system meets its design specification and other requirements. A test plan is usually prepared by or with significant input from the engineer.This document describes the plans for testing the architectural prototype of System.

In my Project the machine has to be tested to get the Desired Output.I use Different Classes of images for testing the system.

6.2 Unit testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. In our system,

- Test to check whether the encrypting module work properly
- Test to check whether the stegano module work properly.
- Test to check whether the decrypting module accurately and correctly decrypt the data.

6.3 Integration testing

Integration testing (sometimes called integration and testing) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

- check whether the capturing image work properly.
- check whether the facial emotion detection and heart beat sensor work properly.
- check whether the model generates the description Accurately.

6.4 System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

6.5 Implementation

Test the machine with number of employees.

RESULTS

Chapter 7

7 Results

The system works when the IT professional will be seat in the front of camera then it will be able to detect the facial expression and run in real-time. A camera is used to capture the near front sight of the employee while he is working in front of the computer. Captured video is divided into sections of equivalent length and set of similar number of image frames are extracted from each part correspondingly and are examined. The image detection includes the calculation of the variation in the place of the eyebrow from its mean position. The displacement of eyebrow from its place is considered by examining the image for the eyebrow co-ordinates. If the employee is found stressed in the successive sections of time intervals which was previously divided, the decision for stress detection is formed for a employee working in front of computer with the obtained results it employs the technique of deep learning. The stress detection module scans the binary image from the extreme left top to record the coordinates of the eyebrow. The stress detection module scans the binary image from the extreme left top to record the coordinates of the eyebrow. The offline displacement calculation sub-module calculates the shifting of eyebrow using the obtained eyebrow co-ordinates which is subsequently followed by variance calculation of the displacement. The classifier sub-module is trained offline are employed to determine the presence of emotion. The integrated decision of individual frames eventually determines the level of stress involved.The main objective of the stress detection system is to monitor the emotional status of IT professionals and detect and reduce stress levels in order to create a comfortable workplace for them The system utilizes image processing and machine learning techniques to analyze stress patterns in IT professionals . Image acquisition tools are used to capture images of the employees,a heart beat sensor detectect the heart rate which are then processed and analyzed using image processing techniques . Machine learning algorithms, such as KNN classifiers, are applied to classify stress levels based on the analyzed images . average of both heart rate and emotions will calculate. The system also includes the use of traditional survey forms to gather additional data on stress levels The ultimate goal is to provide employees with preventative stress management solutions and create a healthy and spontaneous working environment.

CONCLUSIONS AND FUTURE WORKS

Chapter 8

8 Conclusion and Future Works

8.1 Conclusion

According to implementation and conclusion to predict stress in the employees by observing the captured images of authenticated users which makes the system secure. The image is captured and detect heart beat using a sensor automatically when the authenticate user is logged in based on some time interval. Average of this captured images and the heart beat are used to detect the stress of the user based on some standard conversion and image processing mechanisms. Then the system will analyze the stress levels by using Machine Learning algorithms which generates the results that are more efficient.

8.2 Future Enhancement

As a future work the Deep Learning algorithms and Machine Learning algorithms can used to detect stress in employees.Machine learning can enhance stress detection by integrating diverse data sources, developing personalized models, integrating real-time monitoring and intervention, conducting longitudinal analysis, incorporating explainable AI techniques, considering cross-cultural and ethical considerations, and integrating stress detection technologies into healthcare systems. These advancements can improve the accuracy and relevance of stress assessments, facilitate timely interventions, and enable more holistic approaches to stress management and mental health care. By addressing these areas, future machine learning stress detection can contribute to more effective, personalized, and accessible solutions.

REFERENCES

9 Bibliography

- G. Giannakakis, D. Manousos, F. Chiarugi, "Stress and anxiety detection using facial cues from videos," Biomedical Signal processing and Control", vol. 31, pp. 89- 101, January 2017.

- Nisha Raichur, Nidhi Lonakadi, Priyanka Mural, "Detection of Stress Using Image Processing and Machine Learning Techniques", vol.9, no. 3S, July 2017.

- U. S. Reddy, A. V. Thota and A. Dharun, "Machine Learning Techniques for Stress Prediction in Working Employees," 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Madurai, India, 2018, pp. 1-4.

- Healthy office: Using smartphones and wearable sensors, employees may recognize their moods at work. In: Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on. IEEE; 2016, p. 1–6

- Liu, D., Ulrich, M. Listen to your heart: Stress prediction Vol 13, Issue 03, MARCH/2022 ISSN NO:0377-9254 www.jespublication.com PageNo:425 using consumer heart rate sensors 2015;

- T. Jick and R. Payne, "Stress at work," Journal of Management Education, vol. 5, no. 3, pp. 50-56, 1980.

- "Stress and anxiety detection using facial cues from videos," Biomedical Signal Processing and Control, vol. 31, pp. 89-101, January 2017. G. Giannakakis, D. Manousos, F. Chiarugi

- Bhattacharyya, R., Basu, S. (2018). Retrieved from 'The Economic Times'.

- Bakker, J., Holenderski, L., Kocielnik, R., Pechenizkiy, M., Sidorova, N.. Stess@ work: From measuring stress to its understanding, prediction and handling with personalized coaching. In: Proceedings of the 2nd ACM SIGHIT International health informatics symposium. ACM; 2012, p. 673–678.

- https://www.kaggle.com/qiriro/stres

- OSMI Mental Health in Tech Survey Dataset, 2017

- Communications, N.. World health report. 2001. URL: http://www.who.int/whr/2001/media$_centre/press$

APPENDIX

A User Interface

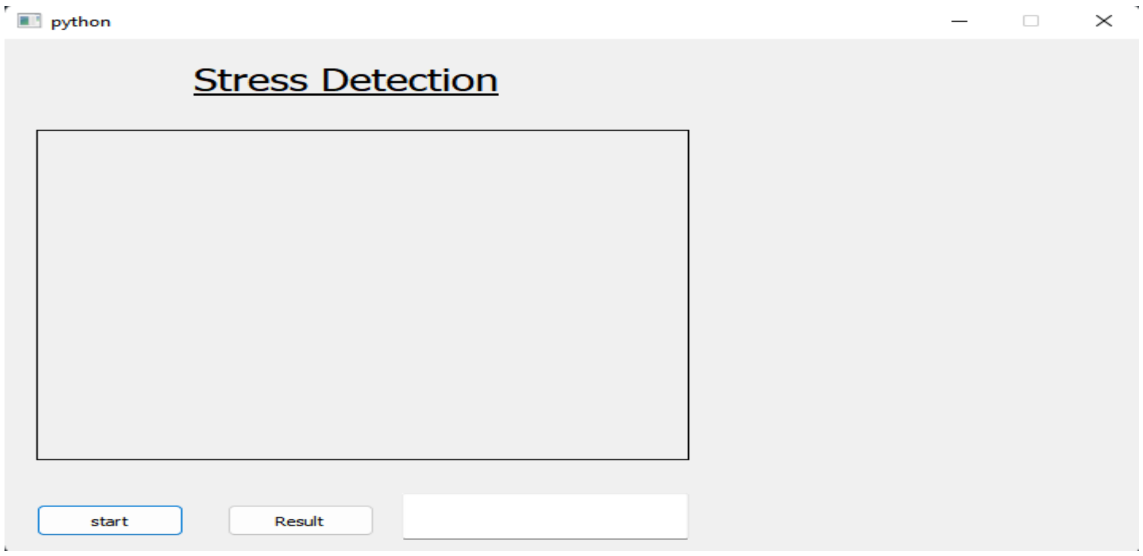


Figure 2: User Interface

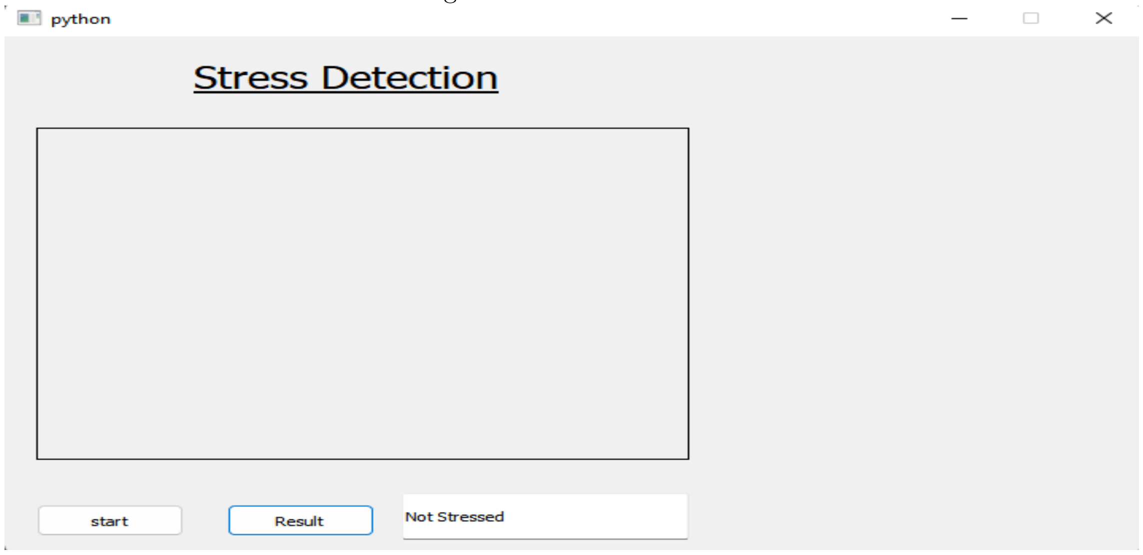
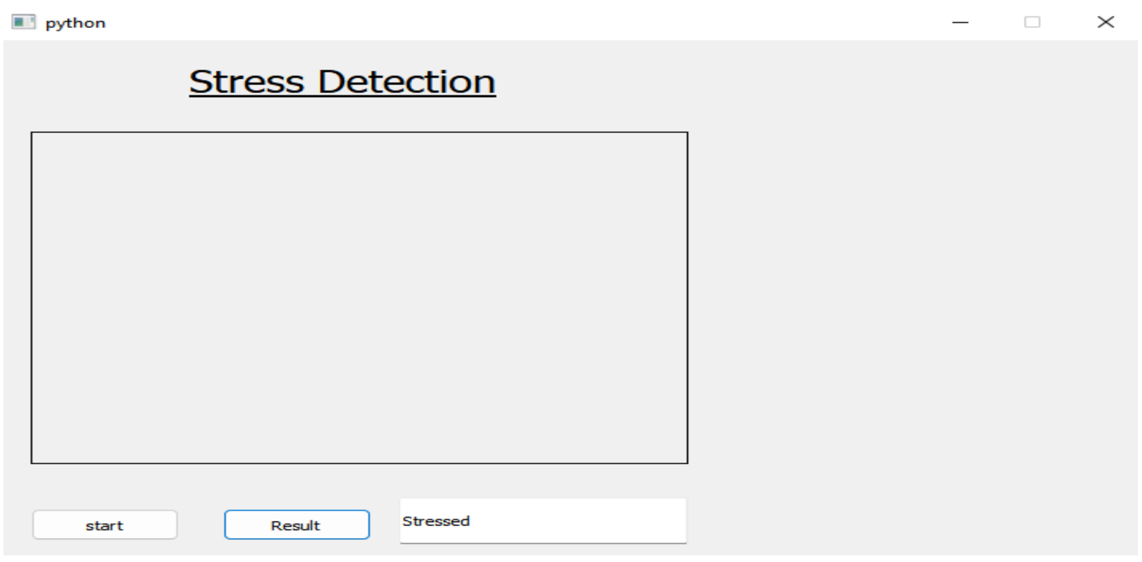


Figure 3: Stress Detection 1



Stressed

Figure 4: Stress Detection 2

B Code

```
import numpy as np
import cv2
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array
import serial
import time
import sys
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QDialog, QApplication, QFileDialog
from PyQt5.uic import loadUi
from PyQt5.QtGui import QIcon, QPixmap, QImage
from PyQt5.QtCore import Qt


# Load SSD and ResNet network based Caffe model for 300x300 dim images
net = cv2.dnn.readNetFromCaffe("deploy.prototxt", "res10_300x300_ssd_iter_140000.caffemodel"
classifier = load_model('./Emotion_Detection.h5')

class_labels = ['Angry', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Video stream initialization
vs = cv2.VideoCapture(0)
hwfinal=0
happyfinal = 0
sadfinal = 0
surprisefinal = 0
angryfinal = 0
fearfinal = 0
neutralfinal = 0
label = None

# Serial port initialization for MAX30102 sensor
ser = serial.Serial('COM5', 9600)  # Replace 'COMX' with the appropriate port

class MainWindow(QDialog):
def __init__(self):
super(MainWindow, self).__init__()
loadUi("final.ui", self)
self.browse.clicked.connect(self.browsefiles)
self.result.clicked.connect(self.finalemotion)

def finalemotion(self):
global happyfinal, sadfinal, surprisefinal, angryfinal, fearfinal, neutralfina,hwfinal

# Calculate total emotions
total_emotions = happyfinal + sadfinal + surprisefinal + angryfinal + fearfinal + neutralfin

# Calculate stress level
stress_level = (sadfinal + angryfinal + hwfinal) / total_emotions

# Determine stress status
stress_status = "Not Stressed" if stress_level < 0.5 else "Stressed"

self.restBox.setText(stress_status)
```

```
def browsefiles(self):
global happyfinal, sadfinal, surprisefinal, angryfinal, fearfinal, neutralfinal,hwfinal

while True:
ret, frame = vs.read()

# Convert frame to grayscale
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

# Convert frame dimensions to a blob and 300x300 dimensions
(height, width) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0,
(300, 300), (104.0, 177.0, 123.0))

# Pass the blob into DNN
net.setInput(blob)
detections = net.forward()

# Loop over the detections to extract specific confidence
for i in range(0, detections.shape[2]):
confidence = detections[0, 0, i, 2]

# If confidence is greater than the minimum confidence
if confidence < 0.5:
continue

# Compute the (x, y)-coordinates of the bounding box
box = detections[0, 0, i, 3:7] * np.array([width, height, width, height])
(x1, y1, x2, y2) = box.astype("int")

# Draw the bounding box of the face along with the associated probability
text = "{:.2f}%".format(confidence * 100) + " ( " + str(y2 - y1) + ", " + str(x2 - x1) + " )
y = y1 - 10 if y1 - 10 > 10 else y1 + 10
cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
cv2.putText(frame, text, (x1, y), cv2.LINE_AA, 0.45, (0, 0, 255), 2)

# Extract ROI (Region of Interest)
roi_gray = gray[y1:y1 + y2 - y1, x1:x1 + x2 - x1]
roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)

if np.sum([roi_gray]) != 0:
roi = roi_gray.astype('float') / 255.0
roi = img_to_array(roi)
roi = np.expand_dims(roi, axis=0)

# Make a prediction on the ROI, then lookup the class
preds = classifier.predict(roi)[0]
global label
label = class_labels[preds.argmax()]
label_position = (x1, y)
cv2.putText(frame, label, label_position, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 3)
else:
cv2.putText(frame, 'No Face Found', (20, 60), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 3)

# Show the output frame
cv2.imshow("Window", frame)
```

```
# Read serial data from MAX30102 sensor
stress_data = ser.readline().decode('utf-8')
print("Stress Level:", stress_data)
if int(stress_data)>70:
hwfinal+=1
# Update emotion counters based on the detected emotion
frame = str(label)
if frame == "Happy":
happyfinal += 1
elif frame == "Sad":
sadfinal += 1
elif frame == "Surprise":
surprisefinal += 1
elif frame == "Angry":
angryfinal += 1
elif frame == "Neutral":
neutralfinal += 1

# If the 'q' key was pressed, break from the loop
if cv2.waitKey(1) == ord("q"):
break

# Stop capturing
cv2.destroyAllWindows()
vs.release()

# Start the application
app = QApplication(sys.argv)
mainwindow = MainWindow()
widget = QtWidgets.QStackedWidget()
widget.addWidget(mainwindow)
widget.setFixedWidth(715)
widget.setFixedHeight(453)
widget.show()
sys.exit(app.exec_())
cv2.destroyAllWindows()
vs.release()
```

Building A Voice Based Image Caption Generator with Deep Learning

PROJECT REPORT

Submitted By

NEEMA BABU

Reg. No. CCAWMCS004

For the award of the Degree of

Master of Science

in Computer Science

(University of Calicut)

under the guidance of

Ms. Sini Thomas

Head of The Department

M.Sc in COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
CHRIST COLLEGE (AUTONOMOUS)
IRINJALAKUDA, KERALA
2022-2024

ACKNOWLEDGMENT

Submitting my project in the divine feet of Almighty God.I would like to take this opportunity to express my profound gratitude to all the people who have inspired and motivated to take this project success.

I would like to thank our principal Rev.Fr.Dr. Jolly Andrews CMI, for proper ambience to go on with the project. I take these opportunities to acknowledge my thanks to Ms Sini Thomas, Head of the Department of Computer Science for valuable guidance in developing this project. I' am also thankful to my Project Guide Ms Sini Thomas, and all other staff in Department of Computer Science for their immense support

My sincere thanks to all those well wishers and friends who have helped me during the course of the project work and have been making it a great success. Last but not least, I wish to express my thankfulness to my family members for their excellent support and co-ordination.

Date : **NEEMA BABU**

Declaration

I hereby declare that the project entitled **"Building A Voice Based Image Caption Generator with Deep Learning"** submitted to Calicut university, on partial fulfillment of the requirement for the award of degree in Master of Science in Computer Science is a record of original work done by me, under the guidance of Ms.Sini Thomas , Head of the Department of Computer Science, Christ College(Autonomous), Irinjalakuda.

Place : Irinjalakuda	**Name: NEEMA BABU**
Date :	**RegNo: CCAWMCS004**

DEPARTMENT OF COMPUTER SCIENCE

Christ College(Autonomous)

Irinjalakuda

<u>CERTIFICATE</u>

Certified that this thesis entitled ***'Building A Voice Based Image Caption Generator with Deep Learning'*** *submitted by* *"*'***Neema Babu (Reg. No. CCAWMCS004)***" *in partial fulfillment for the award of the degree of Master of Science in Computer Science under University of Calicut during the year 2022-2024, is the bonafide work carried out by him under my guidance and supervision.*

Ms. Sini Thomas	Ms. Sini Thomas
Head of The Department, CSE	Head of the Department
Internal Guide	Computer Science

EXTERNAL EXAMINER **INTERNAL EXAMINER**

Abstract

Image processing is used in various industries and it is remaining as one of the most advanced technologies used in Google, medical field etc. Recently, this technology has also attracted many programmers and developers due to its free and open source tool, which every developer can afford it. Image processing also helps in finding out lot of information from a single image since it is currently utilized as a primary method for collecting the information from image and processing it for some purpose and some operations will also be performed on the image. A voice based image caption generation is a task that involves the NLP (natural language processing) concept for understanding the description of an image. The combination of CNN and LSTM is considered as the best solution for this project; the main target of the proposed research work is to obtain the perfect caption for an image. After obtaining the description, it will be converted into text and the text into a voice. Image description is a best solution used for a visually impaired people who are unable to comprehend visuals. With the use of a voice based image caption generator, the descriptions can be obtained as a voice output, if their vision can't be resorted. In future, image processing will emerge as a significant research topic, which will be primarily utilized to save human lives.

Contents

List of Figures

INTRODUCTION

Chapter 1

1 Introduction

1.1 Overview

In recent years Deep learning is one of the most used trend in Machine Learning and artifical intelligence, it is a machine learning technique inspired by the human brain, it uses the algorithm like convolutional neural network,recurrent neural network,long short term memory etc., where there are many developments had already made for visually impaired people.Voice based image caption generator is used to identify the objects and information present in the image, which could improve the lives of visually impaired people. Using CNN and LSTM together can be best fit for the project because LSTM is similar to RNN, and the RNN algorithm is depending on the LSTM because its having the feedback connectivity and also LSTM process the entire sequence of data. The main challenge of deep learning is when we deal with large data we need to go deeper that is analyzing the huge data need to done thoroughly. The structure of text descriptions should be relevant to the objects present in the image, and the relationship between the objects and its description need to be clarified. Our ultimate aim of the project is to train the dataset with good result and with the high accuracy. Flicker dataset is utilized with the huge collection of photographs used for computer vision and image processing algorithms. So the voice based caption generator act as a eye for the people don't have the ability to conceptualize the scene happen arround themselves, they can roam anywhere without the support of anyone else.

1.2 Project Profile

- **Title** : Building A Voice Based Image Caption Generator with Deep Learning
- **Domain** : Deep Learning
- **Language** : Python
- **Version** : 3.08.0

1.3 Contributions

The major contributions of this system are:

- The enhanced accuracy is improved.
- Enhanced precision.

PROBLEM DEFINITION AND METHODOLOGY

Chapter 2

2 Problem Definition and Methodology

2.1 Problem definition

The image caption generator is a task that involves computer vision and NLP(natural language processing) concepts to recognize the context of an image and describe them in a natural language like English .And the Convolution Neural Network(CNN) and Long Short Term Memory(LSTM) can be combined to create an image caption generator and generate captions for your own images.

2.2 Objectives

To learn the concept of CNN and LSTM model and build a working model of image caption generator. Features of the images where extracted by using CNN model trained on large image dataset. And we are giving the features as an input to the LSTM model and it will generate caption for an image. It requires both computer vision system to localize and describe the features of the image in a single word using natural language.

2.3 Motivation

This project will help us to understand the related methods and attention mechanism,which plays an important role in computer vision and is recently widely used in image caption generator tasks. And the advantages and the shortcomings of these methods are implemented, providing the commonly used datasets and evaluation criteria in this field, and it explained how the present scenarios happening in front of visually impaired people.

2.4 Methodology

Generating a caption for a given image is a challenging problem in the deep learning domain. we will use different techniques of computer vision and NLP to recognize the context of an image and describe them in a natural language like English. we will build a working model of the image caption generator by using CNN (Convolutional Neural Networks) and LSTM (Long short term memory) units.And to judge the quality of the description created during the process.

2.5 Scope

Accurate image captioning with the use of multi modal neural networks has been a interesting topic in the field of deep learning. Its been working with several of these approaches and the algorithms seems to give very promising results.when it comes to using image captioning in real world application most of the time only a few are mentioned such as hearing aid for the visually impaired people and content generation.Main scope is we are adding the additional functionality like audio.once the description is generated it will readout the description of an image.

REQUIREMENT ANALYSIS AND SPECIFICATION

Chapter 3

3 Requirement Analysis and Specification

3.1 Requirement Analysis

To learn the concept of CNN and LSTM model and build a working model of image caption generator. Features of the images where extracted by using CNN model trained on large image net dataset. And we are giving the features as an input to the LSTM model and it will generate caption for an image. It requires both computer vision system to localize and describe the features of the image in a single word using natural language.

3.2 Existing System

The image caption generator are describing the scenes because images without the captions is meaningless . The process relies on Convolution Neural Network(CNN) that encodes an image into a compact representation by object detection that generates a corresponding sentence. This model is trained to maximize the accuracy and precision of the caption for the given image. Experiments on several datasets show the robustness of ICG in terms of qualitative results and quantitative evaluations. By imparting various hyper parameter tuning, the model can be made more efficient and productive.

3.3 Proposed System

The Proposed methodology for voice based captions which is not only deals with internal images but also give a descriptions for external input images .once a description is created that text description will be read out as voice outputs then the audio is saved in the separate folder that contains all the audio files for the future references. For developing this model we have used convolutional neural network and long short term memory. Convolutional neural network for indentify the various features or objects that are present in the image. It will be helpful for the entire system predict the proper result then it will fed into the long short term memory to produce the sequence of words that properly describe about the image .

3.4 Requirement Specification

3.4.1 Functional Requirements

In software engineering and system engineering, functional requirement defines function of a system and its components. A function is described as a set of inputs, the behavior and outputs.Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form ”system must do ¡requirement¿”, while non-functional requirements are ”system shall be ¡requirement¿”. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture. As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non- functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request - feature - use case - business rule. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.This system does:

- Collect the data and generate a captio for the image and convert it in to voice which could improve the lives of visually impaired people.

3.4.2 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing nonfunctional requirements is detailed in the system architecture, because they are usually Architecturally Significant Requirements. Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of "system shall do requirement", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of "system shall be requirement", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioral requirements".. Qualities—that is non-functional requirements—can be divided into two main categories: Execution qualities, such as safety, security and usability, which are observable during operation (at run time). Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system.

3.5 Feasibility Study

3.5.1 Technical Feasibility

Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with python anaconda installed. Today every organization has computer, so it is not an extra cost.Before the starting of the project,or in the requirement phase we assign the needed hardware and software.After termination ofb the project,we made a comparison between the assigned ones and actually needed ones

3.5.2 Economical Feasibility

Economic feasibility is the most frequently used method for evaluating the effectiveness of proposed system.The proposed model is cost effective.

3.5.3 Operational feasibility

The operational feasibility of building a voice-based image caption generator with deep learning depends on various factors such as the availability of suitable datasets, computational resources, expertise in deep learning, and integration with existing systems. Assessing these factors will help determine if the project is feasible from an operational standpoint.

3.6 Software Requirement Specification

3.6.1 Introduction

Purpose

The purpose of this document is to provide a debriefed view of requirements and specifications of the project called Building A Voice Based Image Caption Generator with Deep Learning.

The goal of this project is to provide better accuracy in analyzing the image and generating a suitable caption in vocal format.

Document Conventions

- All terms are in Times New Roman style.
- Main features or important terms are in bold.
- Use LateX for documentation.

Intended Audience and Reading Suggestions

Anyone with some programming experience, with familiarity in Python and Deep learning, can understand this document.The document is intended for developers, software architects, testers, project managers and documentation writers. This Software Requirement Specification also includes:

- Overall description of the product
- External interface requirements
- System Features
- Other nonfunctional requirements

Product Scope

The product scope of building a voice-based image caption generator with deep learning typically includes defining the target user experience, specifying technical requirements, outlining functionalities (such as voice input, image processing, deep learning model integration), and determining any constraints or limitations (e.g., hardware compatibility, language support). Additionally, it involves defining the scope of the training data, model evaluation metrics, and potential deployment platforms (e.g., mobile app, web service).

References

- IEEE Standard 830-1998 Recommended Practice for Software Requirements Specifications.

3.6.2 Overall Description

Product Perspective

The product perspective of building a voice-based image caption generator with deep learning involves considering its integration into existing systems or platforms (if applicable), understanding its interaction with users and other components, and ensuring compatibility with various devices and operating systems. It also entails evaluating the product's potential impact on users and stakeholders, addressing scalability and maintenance requirements, and aligning with broader strategic goals or initiatives within the organization or industry.

Operating Environment

- Operating System: Windows 11
- Processor: Intel Core i3 / AMD Ryzen 3 or Higher
- Memory: 4GB or more

3.6.3 Design and Implementation Constraints

- Computational Complexity.
- Training time.
- Integration Challenges

3.6.4 Assumptions and Dependencies

Assumptions

The proposed approach assumes a dataset that encompasses a wide range of images and is thoroughly annotated is readily accessible for training models. The efficacy of the model hinges upon the inclusiveness and excellence of this dataset. A diverse and well-annotated dataset that includes various images is at one's disposal for the purpose of model training.

Dependencies

- Python
- CUDA and cuDNN (Optional)

3.6.5 External Interface Requirements

User Interfaces

First the image to be identified is being fed into the model using an external web interface.The uploaded image is then processed by the model and then generate the proper caption and is converted as voice output.

Hardware Interfaces

- Operating System: Windows or any other Platform

- Hardware: Intel Core i5
- Internet Connection

Software Interfaces

- Python

Communications Interfaces

Standard HTTP COMMUNICATION interface required for internet connection.

3.6.6 System Features

- Deep Learning Model Integration:
  Integration of a deep learning model (e.g., CNN-LSTM) trained on image-caption datasets to generate descriptive captions based on input images.

- Functional Requirements
  Image input is given to the machine and it generates a caption and it converts in to voice.

3.6.7 Other Nonfunctional Requirements

Performance Requirements

- Accuracy: The classification accuracy of the system must meet or exceed predefined standards, reflecting the model's capability to correctly identify images. High accuracy is crucial for the system's reliability and utility in various applications.

- Resource Utilization: Efficient utilization of computational resources, including CPU, GPU, or TPU, is necessary to ensure optimal performance during training and inference. The system should manage resource allocation effectively, avoiding bottlenecks or overutilization.

- Training Time: The time required for model training should be within acceptable limits, allowing for timely updates and improvements. Efficient training times are especially important when retraining the model with new data or making enhancements.

- Compatibility: Should be compatible with a wide range of devices, operating systems and platforms, to ensure its widespread adoption and use.

SYSTEM DESIGN

Chapter 4

4 System Design

4.1 Users of the System

User: The user who handles the System.

4.2 Modularity criteria

The proposed system has following modules :

- Dataset Collection and Data Cleaning.
- Extracting Feature Vector.
- Loading dataset for Training the model.
- Tokenizing Vocabulary
- Creation of Data Generator.

4.3 Design Methodologies

4.3.1 System Architecture

Input image taken from the user side convolutional neural network identify the objects that present in the image it extract the important features of an image and store those feature vector values, using pooling functions it will predict the features. Once the process completed it will move on to the long short term memory layer for the sequence sentence prediction based on the previous one, here softmax function is used to predict the output accurately and for overcoming the over fitting problem ,when we are working with the neural network most of the nodes having the output that are related to the previous one its results in overfitting, to avoid those problem softmax layer is used .If the output of this layer is between the range of zero to one ,if the range is greater are lesser it results in the error .and system will not predict the correct description for an image

4.3.2 Convolutional Neural Network (CNN)

A convolutional or CNN is a class of deep neural network it is mostly used for analyzing visual images and classification, and also it is used in various field like image recognition , NLP and speech recognition. It has three layers namely, convolutional layer, pooling layer, and fully connected layer. The main advantage of using convolutional neural network is, it can identify the objects and faces present in the image. If a picture is indexed with cats and dogs, it will identify the key attributes and relationship between the two, and also behavioral patterns of the objects will be noted by CNN. It is truly efficient than the other algorithms because it has the ability to predict the image with highest accuracy.

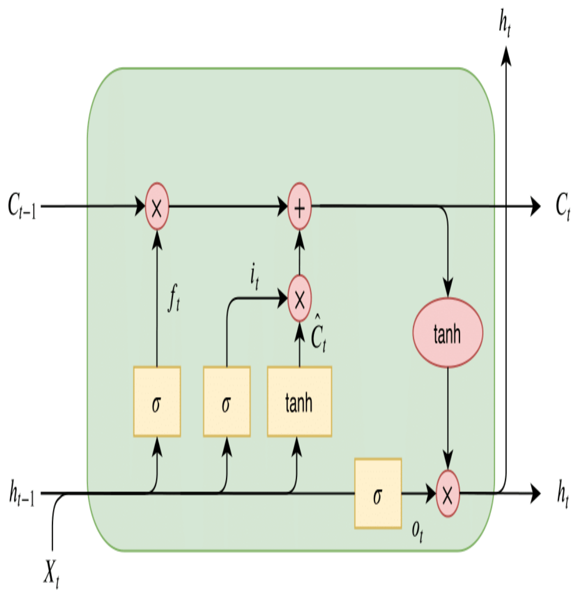


Figure 1: LSTM Structure

4.3.3 Long Short Term Memory (LSTM)

Long short term memory is a type of RNN, it is used for sequence prediction problems. The non-relevant information will be removed by using LSTM, and long short term memory have the efficient performance when compared to the RNN, it can be sustainable get the information with the long duration of time. It can be able to predict the information from the next data or previous data. The main challenge in LSTM is it will take more time to drain the data depending on the size of the dataset. CNN will be used for extracting information's from the image and LSTM will generate captions for the input image.

IMPLEMENTATION AND MAINTENANCE

Chapter 5

5 Implementation

5.1 Tools/Scripts for Implementation

5.1.1 Python

Python is a high-level, interpreted programming language that is designed to be easy to read and write. Python is known for its simple syntax, which allows programmers to write code quickly and easily. is an object-oriented language, which means that it supports the creation of objects and classes that can be used to build complex applications. Python has a large standard library that includes modules for a wide range of tasks, from web development and database management to scientific computing and artificial intelligence. There are also many third-party libraries available for Python that provide additional functionality and support for specific tasks. Python is popular among developers because it is easy to learn and use, and it can be used for a wide range of applications.

5.2 Module hierarchy

- **Dataset Collection and Data Cleaning**
  We are using flicker dataset, that contains images and descriptions that descriptions are in the form of dictionary with keys and values ,it's a easy way to map the description with input images . Every text dataset needs to be done with the data cleaning process. That involves clearing the symbols like special characters like asterisk, semicolon, colon, double quotes. Then the keywords starts with digits or ends with digits will be cleaned in this module. Compressing the long sentence , which contains the inappropriate words

- **Extracting Feature Vector**
  Extracting features of an image with the help of exception model and it is a pre-trained model, this model is utilized for the implementation process by using this model we can't do anything the trained model will do everything because its already trained by the large image net dataset it will classify the various difference in the image. It will take 299*299*3 as an input image and removing the end classification layers for getting the 2048 feature vector. It can accept any image format including PNG, JPG, and others. The neural network reduces large set of features extracts from the original input into smaller recurrent neural network-compatible feature vector. It is the major reason for calling CNN as 'Encoder'.

- **Loading Dataset for Training the Model**
  In our flicker dataset folder, we have image file, it holds five thousand images for training and also it includes a function name called open input, where it will print the image name in a string format. Next step is to be fun clean function, which will have captions in the form of dictionary. Also, the Begin and End keyword is added at the starting and ending point of the description. It will be given to long short term memory to forecast the caption. End keyword used to halt the looping process. Training our model generally about to test the length of the training images, training the descriptions and it will include the features too. Best description is obtained only by the training the developed model it especially based on the epoch values what we have given during the process of training. It results in the forecast of accurate description for an input. During the process of training the loss value should be reduced in every iteration, how less the loss value will result in the good model.

- **Tokenizing Vocabulary**
  When language processing is used, it is required to tokenize the data, i.e. segregating a

data like they've into "they" ,"have" ,compressing huge content into small readable unique content. Basically token means arranging the data into smaller blocks. In this module, keras library is used for tokenizing the text data and it can be saved into a separate file, which contains the index value of the text

- **Creation of Data Generator**
  In this module it follows the supervised and un supervised learning model, having the internal image and their accurate output description is comes under the supervised learning ,if user is giving the external source it will show the output with the help of learning pattern from the trained data and it's an un supervised learning. when we using the data generator it first going through the CNN layer and performing some process like pooling ,next passes through the LSTM model it taken the output of CNN model and fit the first input with the second generated word with the help of dense. Comparing each pixel of an image long term short memory will forecast the suited description.

- **Training the Model**
  We have trained our model using the method called fit generator().By using this method training process completed for our project. We trained our model with various epochs value one, two, three, and eight. And after the training process is completed we saved the trained model to folder. By using 1 epoch value we are getting the eighty percentage accuracy of description. While training it will take lots of time depending upon the system compatibility for me it took 16 hours for training model using three epoch value. Gradually the loss value is also decreasing with respect to the each iteration.

5.3 Coding

- **Python**
  Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. Python's syntax is straightforward, making it easy to read and write, and it has a vast standard library that provides modules for a wide range of tasks, including web development, scientific computing, artificial intelligence, and more.

5.4 Problems Encountered

The following are some of the problems faced in this system:

- Computational Complexity.
- Long training time may also prolong the development process
- User Acceptance

TESTING AND IMPLEMENTATION

Chapter 6

6 Testing And Implementation

6.1 Test Plans

A test plan documents strategy that will be used to verify and ensure that a product or system meets its design specification and other requirements. A test plan is usually prepared by or with significant input from the engineer.This document describes the plans for testing the architectural prototype of System. In my Project the machine has to be tested to get the Desired Output.I use Different Classes of images for testing the system.

6.2 Unit testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. In our system,

- Test the preprocessing module work properly.
- Test to check whether the data is image or not.
- Test to check whether the generation of caption is realted to the image.
- Test to check whether the voice caption of the image.

6.3 Integration testing

Integration testing (sometimes called integration and testing) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

- Check whether the machine takes the input data.
- Check whether the system generates the caption for the image
- Check whether the model convert the image to voice.

6.4 System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

6.5 Implementation

- Test the machine with variety of image.

RESULTS

Chapter 7

7 Results

we have compiled some aspects of the image caption generation task, discussed the model framework proposed in recent years to solve the description task. Focused on the algorithmic essence of different attention mechanisms, and summarized how the attention mechanism is applied. We summarize the large datasets and evaluation criteria commonly used in practice. So the obtained captions are readout in the voice format and it help full for the visually unpaired people to recognize. The variety of image captioning system is available today, experimental results shows that this task still has better performance system and improvement. It mainly faces the following challenges, how to generate complete natural language sentences like a human being, and how to make the generated sentences grammatically correct and make the caption semantics as clear as possible and consistent with the given image content.

CONCLUSIONS AND FUTURE WORKS

Chapter 8

8 Conclusion and Future Works

8.1 Conclusion

Voice based image caption generator has been developed using a CNN-LSTM model. Main key aspects of our project to note, the proposed model not only depends on the dataset, the proposed model is trained for testing the user input, so that it can predict the descriptions from the external images. Out dataset consists of 8091 images. The proposed model is required to be trained on huge dataset that contains more than 10,000 images for achieving a better accuracy. This model is not applicable for the exact representation of an image that work finely for some sort of pictures.

8.2 Future Enhancement

We are planning to implement caption generation for the dynamic video capturing ,it's a tedious and more time consuming process ,to overcome this kinds of stuff we are planning to implement these concepts with the help of deep learning by using the most strong algorithm and planning to converting the caption into the different possible languages. Because live captioning is the required and most useful process for this technical world.

REFERENCES

9 Bibliography

- Sumathi, T., Hemalatha, M, "A combined hierarchical model for automatic image annotation and retrieval." In: International Conference on Advanced Computing (ICAC)- (2011).

- Dong-Jin Kim, Donggeun Yoo, Bonggeun Sim, In So Kweon, "Senetence Learning Deep convolutional neural Network for Image Caption Generation ", In : 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAl)-2016

- Varsha Kesavan, Vaidehi Muley,Megha kolhekar, "Deep Learning based Image Caption Generation" Global Conference for Advancement in Technology (GCAT)-2019

- Ren C. Luo, Yu-Ting, Hsu, Yu-Cheng, Wen, Huan-Jun, Ye, "Visual Image Caption Generation for Service Robotics and Industrial Application" IEEE-2019

- Yu, M.T., Sein, M.M.: "Automatic image captioning system using integration of N cut and color-based segmentation method". In: Society of Instrument and Control Engineers Annual Conference(SCCEAC)- (2011).

- Ushiku, Y., Harada, T., Kuniyoshi, Y.: "Automatic sentence generation from images". In: (ACM)Multimedia (2011)

- Federico, M., Furini, M.: "Enhancing learning accessibility through fully automatic captioning." In: International Cross-Disciplinary Conference on Web Accessibility(ICDCWA)- (2011)

- Xi, S.M., Im Cho, Y.: "Image caption automatic generation method based on weighted feature". In: International Conference on Control, Automation and Systems(ICCAS) (2013)

- Horiuchi, S., Moriguchi, H., Shengbo, X., Honiden, S.: "Automatic image description by using word-level features". In: International Conference on Internet Multimedia Computing and Service(ICIMCS)- (2013)

- Ramnath, K., Vanderwende, L., El-Saban, M., Sinha, S.N., Kannan, A., Hassan, N., Galley, M.: "AutoCaption: automatic caption generation for personal photos ". In: IEEE Winter Conference on Applications of Computer Vision (2014)

- Sivakrishna Reddy, A., Monolisa, N., Nathiya, M., Anjugam, D.: "A combined hierarchical model for automatic image annotation and retrieval" . In: International Conference on Innovations in Information Embedded and Communication Systems(ICIIECS)- (2015)

- Shivdikar, K., Kak, A., Marwah, K.: "Automatic image annotation using a hybrid engine". In: IEEE India Conference (2015)

- Mathews, A.: "Captioning images using different styles". In: ACM Multimedia Conference (2015)

- Mathews, A., Xie, L., He, X.: "Choosing basic-level concept names using visual and language context". In: IEEE Winter Conference on Applications of Computer Vision (2015)

- Plummer, B.A., Wang, L., Cervantes, C.M., Caicedo, J.C., Hockenmaier, J., Lazebnik, S.: "Flickr30k entities: collecting region-to-phrase correspondences for richer image-to-sentence models". In: International Conference on Computer Vision(ICCV)- (2015)

- Vijay, K., Ramya, D.: "Generation of caption selection for news images using stemming algorithm". In: International Conference on Computation of Power, Energy, Information and Communication(ICCPEIC)- (2015)

- Shahaf, D., Horvitz, E., Mankof, R.: "Inside jokes: identifying humorous cartoon captions ". In: International Conference on Knowledge Discovery and Data Mining (ICKDDM-)(2015)

- Li, X., Lan, W., Dong, J., Liu, H.: Adding Chinese captions to images. In: International Conference in Multimedia Retrieval(lCMR)- (2016)

- Jin, J., Nakayama, H.: Annotation order matters: recurrent image annotator for arbitrary length image tagging. In: International Conference on Pattern Recognition(ICPR) (2016)

- Shi, Z., Zou, Z.: Can a machine generate humanlike language descriptions for a remote sensing image? IEEE Trans. Geosci. Remote Sens. 55(6), 3623–3634 (2016)

- Shetty, R., Tavakoli, H.R., Laaksonen, J.: "Exploiting scene context for image captioning". In: Vision and Language Integration Meets Multimedia Fusion (2016)

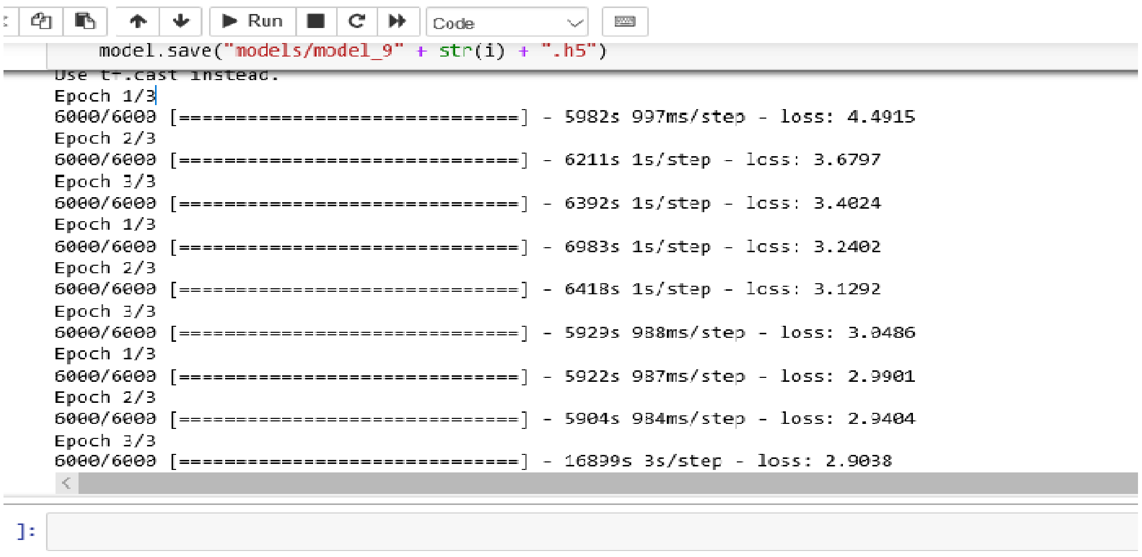
- Li, X., Song, X., Herranz, L., Zhu, Y., Jiang, S.: "Image captioning with both object and scene information". In: ACM Multimedia (2016)

- Wang, C., Yang, H., Bartz, C., Meinel, C.: "Image captioning with deep bidirectional LSTMs ". In: ACM Multimedia (2016)

- Liu, C., Wang, C., Sun, F., Rui, Y.: Image2Text: a multimodal caption generator. In: ACM Multimedia (2016)

APPENDIX

A Training And Testing



```
    model.save("models/model_9" + str(i) + ".h5")
Use tf.cast instead.
Epoch 1/3
6000/6000 [==============================] - 5982s 997ms/step - loss: 4.4915
Epoch 2/3
6000/6000 [==============================] - 6211s 1s/step - loss: 3.6797
Epoch 3/3
6000/6000 [==============================] - 6392s 1s/step - loss: 3.4024
Epoch 1/3
6000/6000 [==============================] - 6983s 1s/step - loss: 3.2402
Epoch 2/3
6000/6000 [==============================] - 6418s 1s/step - loss: 3.1292
Epoch 3/3
6000/6000 [==============================] - 5929s 988ms/step - loss: 3.0486
Epoch 1/3
6000/6000 [==============================] - 5922s 987ms/step - loss: 2.9901
Epoch 2/3
6000/6000 [==============================] - 5904s 984ms/step - loss: 2.9404
Epoch 3/3
6000/6000 [==============================] - 16899s 3s/step - loss: 2.9038
]:
```

Figure 2: During Training Process

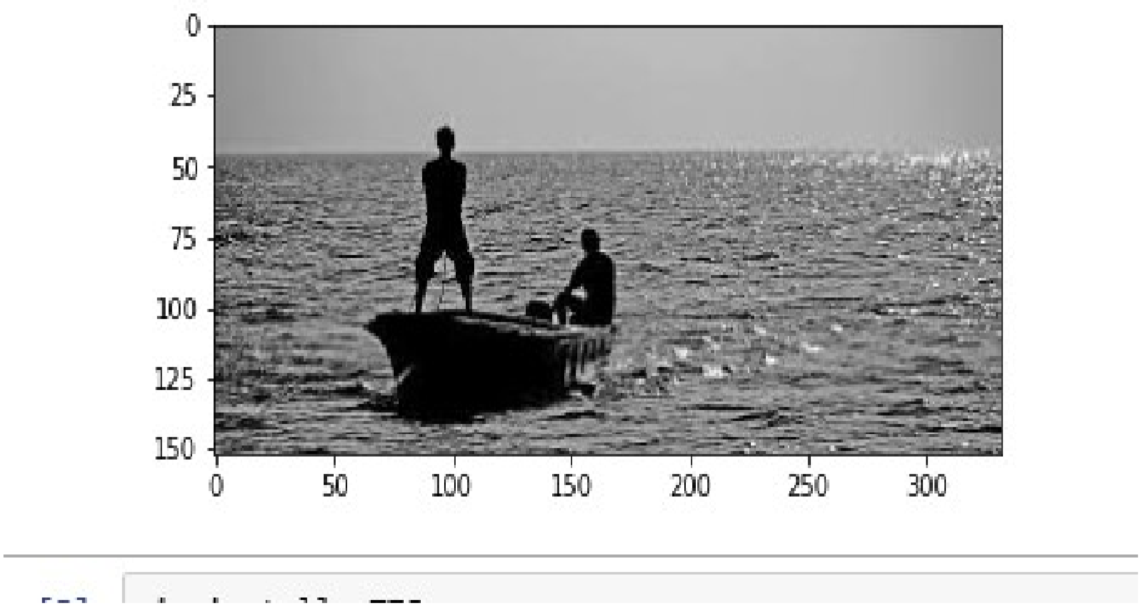


Figure 3: Description Generated

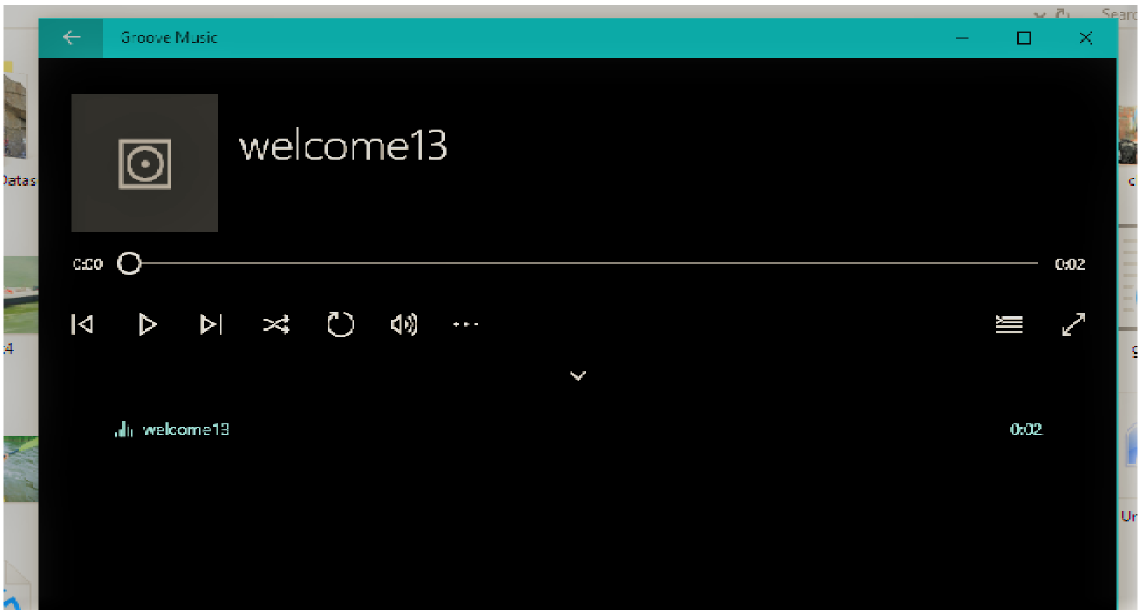


Figure 4: Voice Output

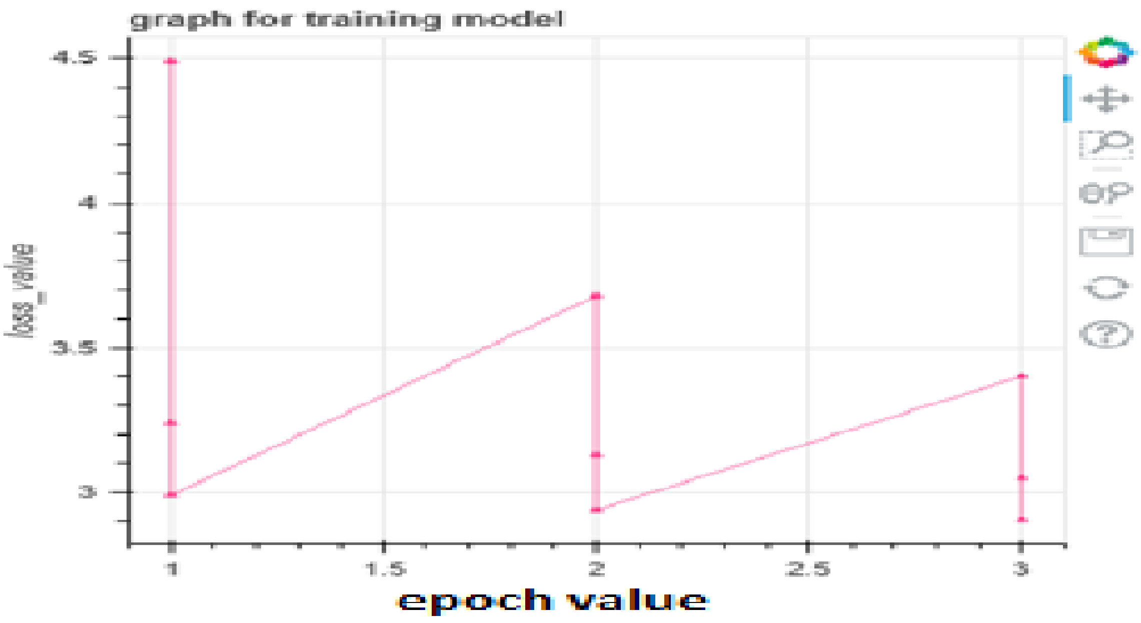


Figure 5: Graph for Epoch with Loss values

B Code

```
#mytestcode

import cv2
from keras.models import load_model
import numpy as np
from keras.applications import ResNet50
from keras.optimizers import Adam
from keras.layers import Dense, Flatten,Input, Convolution2D, Dropout, LSTM, TimeDistributed
from keras.models import Sequential, Model
#from keras.utils import np_utils
#import keras.utils import np_utils
#from tensorflow.keras.utils import to_categorical
from keras.preprocessing import image, sequence
from keras.preprocessing.sequence import pad_sequences
from tqdm import tqdm
import pyttsx3
engine = pyttsx3.init()


vocab = np.load('vocab.npy', allow_pickle=True)

vocab = vocab.item()

inv_vocab = {v:k for k,v in vocab.items()}


print("+"*50)
print("vocabulary loaded")


embedding_size = 128
vocab_size = len(vocab)
max_len = 40


image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))


language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size, input_length=m
language_model.add(LSTM(256, return_sequences=True))
language_model.add(TimeDistributed(Dense(embedding_size)))


conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)
```

```
model.compile(loss='categorical_crossentropy', optimizer='RMSprop', metrics=['accuracy'])

model.load_weights('mine_model_weights.h5')

print("="*150)
print("MODEL LOADED")

#resnet = ResNet50(include_top=False,weights='imagenet',input_shape=(224,224,3),pooling='avg


resnet = load_model('model.h5')

print("="*150)
print("RESNET MODEL LOADED")



image = cv2.imread('file.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

image = cv2.resize(image, (224,224))

image = np.reshape(image, (1,224,224,3))



incept = resnet.predict(image).reshape(1,2048)

print("="*50)
print("Predict Features")


text_in = ['startofseq']

final = ''

print("="*50)
print("GETING Captions")

count = 0
while tqdm(count < 20):

count += 1

encoded = []
for i in text_in:
encoded.append(vocab[i])

padded = pad_sequences([encoded], maxlen=max_len, padding='post', truncating='post').reshape

sampled_index = np.argmax(model.predict([incept, padded]))

sampled_word = inv_vocab[sampled_index]

if sampled_word != 'endofseq':
final = final + ' ' + sampled_word
```

```
text_in.append(sampled_word)
print(final)
engine.say(final)
engine.runAndWait()

#model.py

from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.utils import plot_model
from keras.models import Model, Sequential
from keras.layers import Input
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
from keras.layers import Dropout
from keras.layers.merge import add
from keras.callbacks import ModelCheckpoint
from keras.layers import Dense, Flatten,Input, Convolution2D, Dropout, LSTM, TimeDistributed
from keras.models import Sequential, Model
embedding_size = 128
max_len = MAX_LEN
vocab_size = len(new_dict)

image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))

image_model.summary()

language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size, input_length=m
language_model.add(LSTM(256, return_sequences=True))
language_model.add(TimeDistributed(Dense(embedding_size)))

language_model.summary()

conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)

# model.load_weights("../input/model_weights.h5")
model.compile(loss='categorical_crossentropy', optimizer='RMSprop', metrics=['accuracy'])
model.summary()
model.fit([X, y_in], y_out, batch_size=512, epochs=50)
inv_dict = {v:k for k, v in new_dict.items()}
model.save('model.h5')
model.save_weights('mine_model_weights.h5')
np.save('vocab.npy', new_dict)
def getImage(x):

test_img_path = images[x]
```

```
test_img = cv2.imread(test_img_path)
test_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)

test_img = cv2.resize(test_img, (299,299))

test_img = np.reshape(test_img, (1,299,299,3))

return test_img

#image processor
import numpy as np
import pandas as pd
import cv2
import os
from glob import glob
images_path = '../input/flickr8k-sau/Flickr_Data/Images/'
images = glob(images_path+'*.jpg')
len(images)
images[:5]
import matplotlib.pyplot as plt

for i in range(5):
plt.figure()
img = cv2.imread(images[i])
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
from keras.applications import ResNet50

incept_model = ResNet50(include_top=True)
from keras.models import Model
last = incept_model.layers[-2].output
modele = Model(inputs = incept_model.input,outputs = last)
modele.summary()
images_features = {}
count = 0
for i in images:
img = cv2.imread(i)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (224,224))

img = img.reshape(1,224,224,3)
pred = modele.predict(img).reshape(2048,)

img_name = i.split('/')[-1]

images_features[img_name] = pred

count += 1

if count > 1499:
break

elif count % 50 == 0:
print(count)
len(images_features)
```

```
# Text preprocess
caption_path = '../input/flickr8k-sau/Flickr_Data/Flickr_TextData/Flickr8k.token.txt'
captions = open(caption_path, 'rb').read().decode('utf-8').split('\n')
len(captions)
captions_dict = {}
for i in captions:
try:
img_name = i.split('\t')[0][:-2]
caption = i.split('\t')[1]
if img_name in images_features:
if img_name not in captions_dict:
captions_dict[img_name] = [caption]

else:
captions_dict[img_name].append(caption)

except:
pass
len(captions_dict)


#visualize image with caption

import matplotlib.pyplot as plt

for i in range(5):
plt.figure()
img_name = images[i]


img = cv2.imread(img_name)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.xlabel(captions_dict[img_name.split('/')[-1]])
plt.imshow(img)
import matplotlib.pyplot as plt

for k in images_features.keys():
plt.figure()

img_name = '../input/flickr8k-sau/Flickr_Data/Images/' + k


img = cv2.imread(img_name)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.xlabel(captions_dict[img_name.split('/')[-1]])
plt.imshow(img)

break

def preprocessed(txt):
modified = txt.lower()
modified = 'startofseq ' + modified + ' endofseq'
return modified
for k,v in captions_dict.items():
for vv in v:
```

```
captions_dict[k][v.index(vv)] = preprocessed(vv)

#create vocabulary

 count_words = {}
 for k,vv in captions_dict.items():
 for v in vv:
 for word in v.split():
 if word not in count_words:

 count_words[word] = 0

 else:
 count_words[word] += 1
 len(count_words)
 THRESH = -1
 count = 1
 new_dict = {}
 for k,v in count_words.items():
 if count_words[k] > THRESH:
 new_dict[k] = count
 count += 1
 len(new_dict)
 new_dict['<OUT>'] = len(new_dict)
 captions_backup = captions_dict.copy()
 captions_dict = captions_backup.copy()
 for k, vv in captions_dict.items():
 for v in vv:
 encoded = []
 for word in v.split():
 if word not in new_dict:
 encoded.append(new_dict['<OUT>'])
 else:
 encoded.append(new_dict[word])

 captions_dict[k][vv.index(v)] = encoded
 captions_dict

 #build generator function

 from keras.utils import to_categorical
 from keras.preprocessing.sequence import pad_sequences
 MAX_LEN = 0
 for k, vv in captions_dict.items():
 for v in vv:
 if len(v) > MAX_LEN:
 MAX_LEN = len(v)
 print(v)
 MAX_LEN
 captions_dict
 Batch_size = 5000
 VOCAB_SIZE = len(new_dict)

 def generator(photo, caption):
 n_samples = 0

 X = []
```

```
y_in = []
y_out = []

for k, vv in caption.items():
for v in vv:
for i in range(1, len(v)):
X.append(photo[k])

in_seq= [v[:i]]
out_seq = v[i]

in_seq = pad_sequences(in_seq, maxlen=MAX_LEN, padding='post', truncating='post')[0]
out_seq = to_categorical([out_seq], num_classes=VOCAB_SIZE)[0]

y_in.append(in_seq)
y_out.append(out_seq)

return X, y_in, y_out
X, y_in, y_out = generator(images_features, captions_dict)
len(X), len(y_in), len(y_out)
X = np.array(X)
y_in = np.array(y_in, dtype='float64')
y_out = np.array(y_out, dtype='float64')
X.shape, y_in.shape, y_out.shape
X[1510]
y_in[2]
```

DETECTING OFFENSIVE LANGUAGE ON SOCIAL NETWORKS USING NEURAL NETWORKS

PROJECT REPORT

Submitted By

JESLIN RANI JIJO

Reg. No. CCAWMCS002

For the award of the Degree of

Master of Science

in Computer Science

(University of Calicut)

under the guidance of

Ms. Priyanga K K

Assistant Professor

M.Sc in COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
CHRIST COLLEGE (AUTONOMOUS)
IRINJALAKUDA, KERALA
2022-2024

ACKNOWLEDGMENT

Submitting my project in the divine feet of Almighty God.I would like to take this opportunity to express my profound gratitude to all the people who have inspired and motivated to take this project success.

I would like to thank our principal Rev.Fr.Dr. Jolly Andrews CMI, for proper ambience to go on with the project. I take these opportunities to acknowledge my thanks to Ms Sini Thomas, Head of the Department of Computer Science for valuable guidance in developing this project. I' am also thankful to my Project Guide Ms Priyanga K K, and all other staff in Department of Computer Science for their immense support

My sincere thanks to all those well wishers and friends who have helped me during the course of the project work and have been making it a great success. Last but not least, I wish to express my thankfulness to my family members for their excellent support and co-ordination.

Date : **JESLIN RANI JIJO**

DECLARATION

I hereby declare that the project entitled **"Detecting Offensive Language on Social Networks Using Neural Networks "** submitted to Calicut university, on partial fulfillment of the requirement for the award of degree in Master of Science in Computer Science is a record of original work done by me, under the guidance of Ms.Priyanga K K, Assistant Professor in Department of Computer Science, Christ College(Autonomous), Irinjalakuda.

Place : Irinjalakuda	**Name: JESLIN RANI JIJO**
Date :	**RegNo: CCAWMCS002**

DEPARTMENT OF COMPUTER SCIENCE

Christ College

Irinjalakuda

CERTIFICATE

*Certified that this thesis entitled **"Detecting Offensive Language On Social Networks Using Neural Networks"** submitted by "**Jeslin Rani Jijo(Reg. No.CCAWMCS002)**" in partial fulfillment for the award of the degree of Master of Technology in Computer Science & Engineering under University of calicut during the year 2022-2024, is the bonafide work carried out by him under my guidance and supervision.*

Ms. Priyanga K K. Assistant Professor, CSE Internal Guide	Ms. Sini Thomas. Head of the Department Computer Science

EXTERNAL EXAMINER **INTERNAL EXAMINER**

Abstract

In this research endeavor, we constructed several deep learning architectures to partake in the Offensival shared task presented in the work by Zampieri et al. in 2019. The dataset entailed annotations using a three-level annotation scheme. The task at hand involved discerning between offensive and non-offensive content, categorizing offensive language, and identifying specific targets within offensive content. Additionally, we incorporated Part-of-Speech (POS) information as a feature in the deep learning models for classification. Among the models, the top performers were a stacked architecture combining CNN-BiLSTM with Attention, BiLSTM augmented with POS information and word features, and BiLSTM for the third task.

Contents

List of Figures

INTRODUCTION

Chapter 1

1 Introduction

1.1 Overview

In today's digital age, online social networks have become an integral part of our lives, connecting people and facilitating communication on a global scale. However, these platforms also face the challenge of managing offensive language and content that can harm individuals and communities. To address this issue, extensive research has been conducted to develop systems that can effectively detect and mitigate offensive language in online social networks.This paper focuses on two crucial aspects of offensive language detection: data preprocessing and feature selection. Data Preprocessing: The first step in the offensive language detection process is data preprocessing, which plays a crucial role in ensuring the quality and consistency of the collected data. This step involves removing noise, filtering out irrelevant information, and segmenting the text into smaller units for analysis. Noise removal is essential to eliminate unwanted elements such as special characters, punctuation marks, and emoticons that can interfere with the subsequent analysis.By removing noise, the system can focus on the essential aspects of offensive language and improve the accuracy of the detection process. Filtering out irrelevant information involves identifying and removing non-relevant content that does not contribute to the offensive language detection task. This could include advertisements, URLs, or non-textual data. By filtering out irrelevant information, the system reduces the noise further and improves the efficiency of subsequent analyses. Segmentation, the final component of data preprocessing, entails dividing the text into smaller units for more granular analysis.Proper segmentation enables the system to examine individual words or phrases, which is essential for identifying offensive language accurately. Segmenting the text effectively prepares it for subsequent feature extraction and analysis. In summary, this paper recognizes the growing importance of addressing offensive language in online social networks and emphasizes the critical role of data preprocessing in this context. By comprehensively preprocessing the data, offensive language detection systems can better equip themselves to identify and mitigate harmful content, ultimately fostering a safer and more inclusive online environment.The subsequent sections of this paper will delve into the intricacies of feature selection and its significance in the offensive language detection process.

1.2 Project Profile

- Title : Detecting Offensive Language on Social Networks Using Neural Networks .
- Domain : Natural Language Processing .
- Language : Python
- Version : 3.11

1.3 Contribution

Offfensive language detection systems face challenges in balancing automation with freedom of expression, and require constant updates to keep up with online trends and language use:

- Offensive language detection aids in identifying and addressing cyberbullying and harassment incidents, thereby fostering a safer environment for users.

- Social media platforms often have community guidelines that prohibit certain types of content. Offensive language detection helps in enforcing these guidelines by identifying and removing violating content.

- Detecting and minimizing offensive language on social media platforms helps reduce toxicity and promote healthier discussions and interactions.

- Offensive language detection enhances machine learning and deep learning models by identifying new offensive language patterns, allowing them to better adapt to evolving online behavior.

PROBLEM DEFINITION AND METHODOLOGY

Chapter 2

2 Problem Definition and Methodology

2.1 Problem Definition

This work aims to identify the class of a new unlabeled sentence posted by a user given their previous history of messages and the classification of their content as Neutrality, Sexism, or Racism. The problem is to determine which class the user's unlabeled sentence belongs to, given their identity and the history of related postings. The research question is: How can the class of a new posting be effectively identified, given the user's identity and previous posting history? To answer this question, our main goals can be summarized as follows: Utilizes NLP, deep learning, and machine learning techniques.

- Problem definition, labeled dataset collection, preprocessing, feature extraction, model selection, training, fine-tuning, performance evaluation, deployment, and continuous monitoring.

- Users can report false positives or negatives for model improvement.

- Legal and ethical considerations ensure fairness and transparency in content moderation.

- A robust system can detect and handle offensive language on social media.

2.2 Objective

The primary objective of offensive language detection using Natural Language Processing (NLP) is to foster a safer and more civil online environment, thereby promoting healthy interactions and reducing negativity spread on social media platforms.:

- Automatic detection of offensive language like insults, hate speech, and bullying.

- Flags harmful content for review or removal.

- Protects users from negativity and harassment.

- Assists in content moderation by identifying violating community guidelines.

2.3 Motivation

The research focuses on identifying and classifying offensive language on social media, a challenge for maintaining a positive online environment. It builds on existing work by utilizing the OffensEval shared task and dataset. The researchers introduce new deep learning architectures for offensive language detection, including a stacked CNN-BiLSTM with Attention model. They also explore the use of Part-of-Speech (POS) information as a feature to improve classification accuracy. The research demonstrates the effectiveness of their models in achieving task goals. The research aims to develop improved methods for tackling offensive language detection in social media.

2.4 Methodology

For international communication, online social networks are essential. It can be difficult to control abusive language on these networks. This work emphasizes feature selection and data preprocessing for hostile language identification. Preparing data: The first phase involves cleaning up the data by segmenting the text, removing noise (e.g., special characters and emoticons), and removing unnecessary information. Accurate analysis of words or phrases for foul language detection is aided by proper segmentation. Choosing Features: The study employs a fuzzy-based convolutional neural network (FCNN) for feature identification following preprocessing. CNNs detect patterns and representations, while fuzzy logic deals with uncertainty. Techniques: Through text segmentation, FCNN obtains a detailed grasp of linguistic patterns. Fuzzy logic and CNNs are used in feature selection to effectively understand inflammatory language. CNN-Based Feature Extraction: CNNs, or convolutional neural networks, extract pertinent
The methods we used here are:

- **Feature Extraction-** Convolutional Neural Networks (CNNs) are used for feature extraction in offensive language detection..

- **Ensemble Architecture-**Utilizes an ensemble architecture combining Bidirectional Long Short-Term Memory (Bi-LSTM) and a hybrid of Support Vector Machines (SVM) and Naïve Bayes classifiers.

- **Evaluation Metrices-**The system's accuracy is measured by comparing correctly classified instances to total instances, while precision measures the proportion of correctly identified offensive language instances. Recall measures the proportion of correctly identified instances out of all actual offensive instances in the dataset. The F-1 Score provides a balanced performance measure.

2.5 Scope

The research on offensive language detection and management in online social networks opens up several avenues for future exploration and improvement. Here are some potential areas of focus, Contextual Understanding Enhancing the system's ability to understand the context of language is crucial for accurately detecting offensive content. Future research can explore the integration of contextual information, such as user profiles, post history, and conversation dynamics, to provide a more nuanced analysis of offensive language in different contexts.Dynamic Learning: Developing a system that can adapt and learn from evolving language trends and emerging forms of offensive content is essential. Continuous learning and updating of the detection models using real-time data can help in keeping up with the ever-changing landscape of offensive language.

REQUIREMENT ANALYSIS AND SPECIFICATION

Chapter 3

3 Requirement Analysis and Specification

3.1 Requirement Analysis/Literature Review

In the domain of offensive language detection on social networks, several approaches and tools have been developed to address this pervasive issue. These methods range from rule-based systems to more advanced machine learning techniques. Rule-Based Systems: One prevalent approach relies on predefined rules and heuristics to identify offensive language. These rules may encompass keyword lists, regular expressions, and linguistic patterns associated with offensive content. While rule-based systems are relatively straightforward to implement, they often struggle with context-dependent cases, sarcasm, and emerging forms of offensive language that may not be captured by predefined rules. Additionally, maintaining and updating the rule sets to adapt to evolving language trends can be a labour-intensive task. More sophisticated techniques involve the application of machine learning algorithms, including support vector machines (SVMs), decision trees, and more recently, deep learning methods. These models are trained on labelled datasets, where offensive and non-offensive content is annotated. This enables the algorithm to learn complex patterns and associations between features within the data. Machine learning approaches have demonstrated significant advancements in offensive language detection, particularly in handling contextual nuances and emerging linguistic trends. However, they require substantial amounts of annotated data for effective training, and their performance heavily depends on the quality and diversity of the dataset. Hybrid Approaches: Some systems employ a combination of rule-based heuristics and machine learning models. This hybrid approach aims to leverage the strengths of both methodologies. Rule-based systems can serve as an initial filter to quickly identify obvious cases of offensive content, while machine learning models can handle more nuanced cases that may not be captured by predefined rules alone. This approach attempts to strike a balance between accuracy and computational efficiency. When evaluating these techniques, it is evident that machine learning-based approaches, particularly those employing neural networks, have demonstrated superior performance in offensive language detection. Their ability to learn intricate contextual cues and adapt to evolving language trends sets them apart. Rule-based systems, while useful for straightforward cases, tend to struggle with context-dependent and nuanced instances of offensive language. Machine learning-based approaches excel in capturing the complex nuances of language, making them highly effective in identifying offensive content. However, they require large and diverse datasets for training, which can be a limiting factor in some cases. Additionally, the computational resources required for training and deploying these models can be substantial. Rule-based systems, on the other hand, are relatively straightforward to implement and computationally efficient. They can provide quick, initial filters for offensive content. However, they are inherently limited in handling contextual subtleties and may generate false positives or negatives in complex cases. Despite the advancements in offensive language detection, there are still notable limitations. Many systems struggle with detecting subtle forms of offensive language that rely on implicit or coded language. Additionally, the rapid evolution of online communication means that new forms of offensive language constantly emerge, challenging existing models to keep pace. Finally, issues of bias and cultural context in training data can lead to disparities in performance across different demographic groups.

3.2 Existing System

The researchers are focused on computational algorithms for detecting offensive content online. Most studies focus on high-resource languages, such as English, due to large datasets. However, some studies have explored multilingual models, using deep learning representations like context word embeddings and multilingual transformers. With the rise of interactive webs and social media, the amount of user-generated content has increased, making human moderators ineffective. Social media managers are now using Natural Language Processing (NLP) tools to automate abusive language identification processes and monitor material. Multilingual text categorization (MTC) is a challenge that can be useful in various situations, such as OL detection and spam filtering. Traditional machine learning and deep learning methods, such as lexicon-based algorithms, word and character n-grams, and ensemble learning, are being used to detect foul language.

3.3 Proposed System

The research developed deep learning architectures for the Offensival shared task, involving discerning offensive and non-offensive content, categorizing offensive language, and identifying targets. The models included Part-of-Speech (POS) information for classification. The top performers included a stacked architecture combining CNN-BiLSTM with Attention, BiLSTM augmented with POS information and word features, and BiLSTM for the third task.

3.4 Requirement Specification

3.4.1 Functional Requirements

In software engineering and system engineering, functional requirement defines function of a system and its components. A function is described as a set of inputs, the behavior and outputs.Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form ”system must do ¡requirement¿”, while non-functional requirements are ”system shall be ¡requirement¿”. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture. As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non- functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request - feature - use case - business rule. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.
This system does:

- Collecting data and detecting the offensive language to provide better data security in today’s digital era.

3.4.2 Non-Functional Requirements

In systems engineering and requirements engineering,a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non- functional requirements is detailed in the system architecture, because they are usually Architecturally Significant Requirements. Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of "system shall do requirement", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of "system shall be ¡requirement¿", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioral requirements". Informally these are sometimes called the "ilities", from attributes like stability and portability. Qualities—that is non-functional requirements—can be divided into two main categories: Execution qualities, such as safety, security and usability, which are observable during operation (at run time). Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system.

3.5 Feasibility Study

3.5.1 Technical Feasibility

Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with visual studio code tool installed. Today every organization has computer, so it is not an extra cost. Before the starting of the project, or in the requirement phase we assign the needed hardware and software. After termination of the project, we made a comparison between the assigned ones and actually needed ones.

3.5.2 Economical Feasibility

Economic feasibility is the most frequently used method for evaluating the effectiveness of proposed system. The procedures are to determine the benefits and savings that are expected from this system is in time savings.Nowadays,online social networks have become an integral part of global communication. However, managing offensive language and content poses a significant challenge for these platforms., offensive language detection systems can achieve accuracy and effectiveness, promoting a safer online environment. Continuous refinement of data preprocessing and feature selection methods contributes to the ongoing efforts to create inclusive and respectful digital spaces.

3.5.3 Operational feasibility

Operational feasibility assesses the extent to which required software performs some simple steps for finding the offensive language in social media.

3.6 Software Requirement Specifications

3.6.1 Introduction

Purpose

The purpose of this document is to provide a brief view of requirements and specifications of the project called Detecting Offensive Language on Social Networks Using Neural Networks .This project is to focused on developing systems that can effectively detect and mitigate offensive language. This paper emphasizes two crucial aspects of offensive language detection: data preprocessing and feature selection.

Document Conventions

- All terms are in Times New Roman style.
- Main features or important terms are in bold.
- Use LateX for documentation.

Intended Audience and Reading Suggestions

Anyone with some programming experience,with familiarity in python and Natural Learning processing.The document is intended for developers, software architects, testers, project managers and documentation writers.This Software Requirement Specification also includes:

- Overall description of the product
- External interface requirements
- System Features
- Other nonfunctional requirements

Product Scope

Natural language processing (NLP) is a machine learning technique that enables computers to interpret, manipulate, and comprehend human language.Research on offensive language detection and management in online social networks can be improved by enhancing contextual understanding and integrating contextual information like user profiles, post history, and conversation dynamics. Dynamic learning is crucial for adapting to evolving language trends and offensive content, and continuous learning and updating detection models using real-time data can help keep up with the ever-changing landscape of offensive language.

References

IEEE Standard 830-1998 Recommended Practice for Software Requirements Specifications.

3.6.2 Overall description

Product Perspective

This project is combination of NLP(natural learning processing) and deep learning.This can focuses on distinguishing offensive and non-offensive content, categorizing offensive language, and identifying targets within offensive content. Deep learning models incorporate Part-of-Speech information, with top performers being CNN-BiLSTM with Attention, BiLSTM augmented with POS information, and BiLSTM for the third task.
Operating Environment

An embedded system with the following minimum specifications:

- Operating System: Windows 11

- Processor: Intel I3 or Higher
- Memory: 4GB or more

Design and Implementation Constraints

- computational complexity.
- user acceptance.
- Adapting to the nuances of different languages and cultural expressions is a complex task.
- Offensive language detection becomes more complex when dealing with multiple languages.
  .

Assumptions and Dependencies

- **Assumptions**

The proposed approach detect the user given data that needs to removing the offensive words detecting in data for the more security.The data collected from various fields and domains can also be used for the detection purpose..

- **Dependencies**

python

3.6.3 External Interface Requirements

User Interfaces

first the data to be detected is taken and given to the machine. the data undergoes detection of words presented in data.the obtained offensive language detected and removed.And the result is stored.

Hardware Interfaces

- Operating System: Windows
- Hardware: Intel Core i3
- Internet connection

Software Interfaces

- python.

Communication Interfaces

Standard HTTP COMMUNICATIONS interface required for internet connection.

3.6.4 System Features

Description

The data detected by RNN neural nertwork,CNN,NLP and GAN algorithm.

Functional Requirements

The computer vision data is sent to the deep learning model for obtaining the result. The deep learning model will be compared with the extracted features of given data. And then finally during the comparison will obtain the result whether data is hated or not hated speech.

3.6.5 Other Nonfunctional Requirements

Performance Requirements

- Quickness: System's embedded system should be fast enough to interact in quick detection with it on the go while responding to the user actions without any shattering or buffering.
- Detection and Response time: very small.
- compatability: should be compatible with a wide range of devices,operating systems and platforms, o ensure its widespread adoption and use.

SYSTEM DESIGN

Chapter 4

4 System Design

4.1 Users of System

U ser : The user who handles the system

4.2 Modularity Criteria

The proposed system has following modules :

- Data collection
- Data processing
- Use Neural Network models

4.3 Design Methodologies

The proposed methodology represents a significant advancement in the realm of offensive language detection on social networks. It hinges on a sophisticated architecture primarily cantered around the utilization of neural networks. These networks are computational models inspired by the structure and functioning of the human brain, designed to process vast amounts of data and discern intricate patterns within it. In the context of offensive language detection, neural networks excel at learning the nuanced features and context-specific cues that distinguish harmful content from benign communication. At the heart of this system lies a multi-layered neural network. The architecture encompasses an input layer, hidden layers, and an output layer. The input layer is responsible for receiving and processing the raw text data extracted from social media posts or comments. Through a series of preprocessing steps, including tasks such as tokenization and data normalization, the input layer ensures that the information fed into the network is refined and devoid of extraneous elements. The hidden layers, constituting the core computational engine, perform the intricate task of learning complex relationships and patterns within the textual data. Each layer comprises interconnected nodes, or neurons, that collectively process the information. Through a process known as forward propagation, the neural network refines its understanding of the contextual intricacies associated with offensive language. This phase of learning is crucial, as it equips the system with the capacity to recognize even subtle indications of offensive content. Finally, the output layer produces a probability score indicating the likelihood of the input text being offensive. By employing an appropriate activation function, the system makes a binary classification, effectively flagging content that crosses a predefined threshold. The proposed methodology's effectiveness is further bolstered by a range of features and components meticulously integrated into its architecture. Notably, advanced preprocessing techniques are employed to enhance the quality of the input data. These include tokenization, which segments text into individual units, and stemming, which reduces words to their root form. Additionally, special characters and irrelevant symbols are systematically removed, ensuring that the neural network processes only meaningful information. This preprocessing phase significantly refines the input data, enabling the subsequent layers of the network to extract and learn meaningful patterns more effectively. Furthermore, the system is designed to continuously adapt and evolve in response to new data. This adaptability is a crucial aspect of its robustness in the face of an ever-changing online landscape. By periodically retraining the neural network with updated datasets, the system remains attuned to emerging linguistic trends and shifts in offensive language usage. This dynamic nature ensures that the system maintains its high accuracy and precision over time, even as the nature of offensive language on social networks evolves. This adaptability is a testament to the system's resilience and its potential to serve as a long-term solution for offensive language detection. The training process of the neural network involved several crucial steps, each tailored to optimize its performance in detecting offensive language. The dataset was randomly partitioned into training, validation, and test sets, with a ratio of 70:15:15, respectively. This partitioning ensured that the network learned from a diverse range of examples while also providing separate sets for fine-tuning hyperparameters and evaluating the model's generalization ability. The choice of hyperparameters was guided by a systematic search for optimal configurations. The learning rate, a critical parameter governing the step size during gradient descent, was set to 0.001. This value was determined through an iterative process, where multiple learning rates were tested, and the one that exhibited the most stable convergence was selected. Additionally, a batch size of 64 was employed during training, striking a balance between computational efficiency and stability in the optimization process. To prevent overfitting, a dropout rate of 0.5 was incorporated between the LSTM layers. This dropout rate was chosen based on extensive experimentation, aiming to strike a balance between regularization and the preservation of important features. The model's performance was monitored on the validation set after each epoch, allowing for early stopping to prevent overfitting and ensuring that the network converged to an optimal state.

4.3.1 Neural network architecture

The neural network architecture was meticulously designed to leverage the extracted features and learn complex patterns associated with offensive language. The chosen architecture is a deep recurrent neural network (RNN), specifically a long short-term memory (LSTM) network. LSTMs are well-suited for sequential data like text, as they excel in capturing long-range dependencies and temporal relationships. The architecture comprises an embedding layer, multiple LSTM layers, and a dense output layer. The embedding layer serves as the initial input stage, converting the tokenized text into dense vectors that capture semantic relationships between words. The LSTM layers process this sequential data, allowing the network to learn contextual patterns and dependencies within the text. To prevent overfitting and enhance generalization, dropout layers were strategically incorporated between the LSTM layers. These layers randomly deactivate a fraction of neurons during training, reducing the network's reliance on specific features and encouraging a more robust learning process. The final dense output layer employs a sigmoid activation function, providing a probability score indicating the likelihood of the input text being offensive. If the score exceeds a predefined threshold, the system classifies the text as offensive. This architecture was chosen for its ability to effectively capture the intricate nuances of language, enabling the system to make accurate predictions regarding offensive content. Additionally, the use of recurrent layers allows the network to process text in a sequential manner, preserving the contextual relationships that are vital in understanding offensive language. In summary, the methodology employed in this study encompassed a comprehensive data collection process, thorough preprocessing steps, feature extraction techniques, and a carefully designed neural network architecture. This holistic approach was instrumental in training a robust offensive language detection system capable of accurately discerning offensive content in social media posts and comments. GANs are a class of machine learning algorithms consisting of two components: a generator and a discriminator. The generator network learns to generate synthetic offensive language samples, while the discriminator network learns to differentiate between real and generated offensive language. Through an adversarial training process, the generator continuously improves its ability to produce realistic offensive language instances, while the discriminator becomes more proficient at distinguishing between real and generated samples.

4.3.2 Distribution of offensive language

The distribution of offensive language within the dataset was carefully examined to provide insights into the prevalence and nature of harmful content in online communication. Approximately 15 percentage of the dataset was labeled as containing offensive language, demonstrating that offensive content constitutes a significant portion of social media discourse. This distribution reflects the reality of online communication, highlighting the critical need for effective offensive language detection mechanisms. Furthermore, within the offensive language subset, additional categorizations were made to capture specific forms of harmful content, such as hate speech, profanity, and discriminatory language. This fine-grained categorization enables a more nuanced analysis of offensive language patterns, allowing the system to differentiate between various forms of harmful communication.

4.3.3 Description of our recurrent neural network based approach

The power of neural networks comes from their ability to find data representations that are useful for classification. Recurrent Neural Networks (RNN) are a special type of neural network, which can be thought of as the addition of loops to the architecture. RNNs use back propagation in the training process to update the network weights in every layer. In our experimentation we used a powerful type of RNN known as Long Short-Term Memory Network (LSTM). Inspired by the work by Badjatiya et al. (2017), we experiment with combining various LSTM models enhanced with a number of novel features in an ensemble. More specifically we introduce:

- A number of additional features concerned with the users' tendency towards hatred behavior.

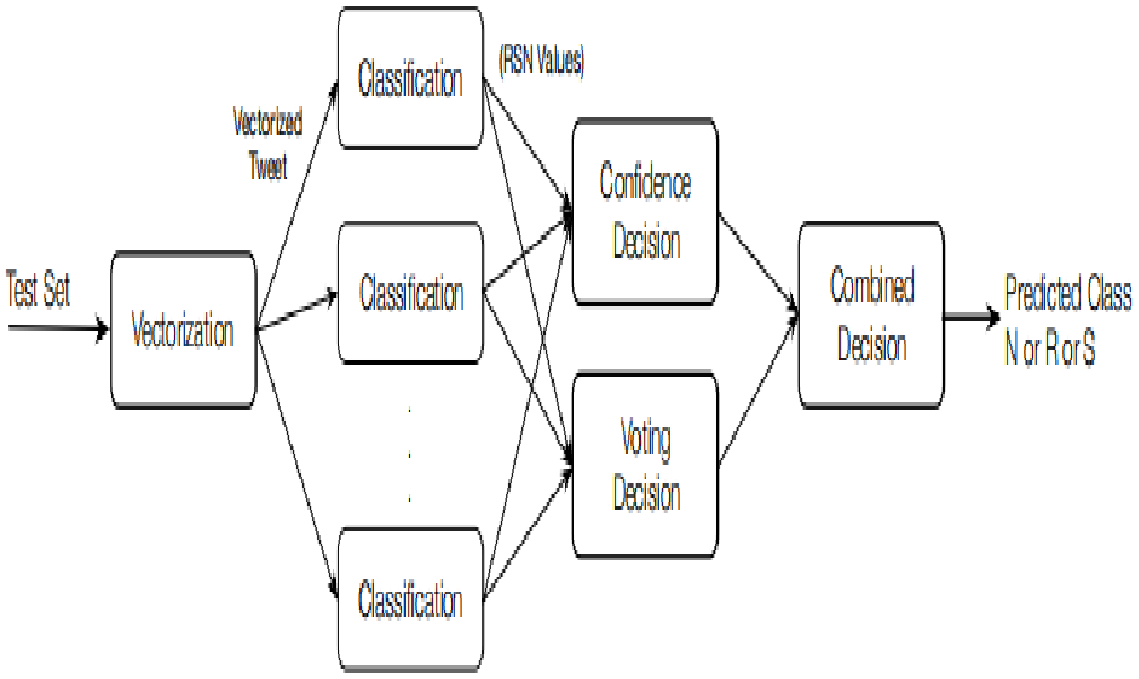


Figure 1: High level view of the system with multiple classifiers

An architecture, which combines the output by various LSTM classifiers to improve he classification ability.

4.3.4 Features

We first elaborate into the details of the features derived to describe each user's tendency towards each class (Neutral, Racism or Sexism), as captured in their tweeting history. In total, we

define the three features tNa, tRa, tSa, representing a user's tendency towards posting Neutral, Racist and Sexist content, respectively. We let ma denote the set of tweets by user a, and use mN,a, mR,a and mS,a to denote the subsets of those tweets that have been labeled as Neutral, Racist and Sexist respectively. Now, the features are calculated as tN,a = —mN,a—/—ma— tR,a = —mR,a—/—ma— tS,a = —mS,a—/—ma—. Furthermore, we choose to model the input tweets in the form of vectors using wordbased frequency vectorization. That is, the words in the corpus are indexed based on their frequency of appearance in the corpus, and the index value of each word in a tweet is used as one of the vector elements to describe that tweet. We note that this modelling choice provides us with a big advantage, because the model is independent of the language used for posting the message.
c We design the end-to-end model for offensive language detection. The model consists of graph attention network layer, BERT, attention layer, and feed-forward layer. The learning rate of the GAT layer is set to 1e-2 and the rest are set to 5e-5. The parameters are initialized by the Xavier normal distribution. In addition, the Adam optimizer is used as optimization

4.3.5 GAT Layer

Graph Attention Networks were proposed by Veliˇckovi´c et al in 2018. GAT uses the attention mechanism to assign different weights to different nodes, which is suitable for graph representation on social networks. The social graph is represented as G = (V, E), V is the set of user nodes and E is the set of edges. The node features are represented as h = h1, h2, ..., hn, hi RF , where n is the number of nodes and F is the dimensionality of the node features. We can obtain a high-level representation of the feature h by the following linear transformation.

$$z_i^{(l)} = W^{(l)} h_i^{(l)}$$

Where W is the weight matrix, hi is the feature of each node, and zi is the transformed feature expression. The (l) represents that this formula is the matrix calculation of layer l. For node i, the correlation coefficient is calculated by node i and its one-degree neighbor j. The nonlinear function LeakyReLU is used as the activation function to obtain the attention scores. After that, the attention scores are normalized by softmax.

$$e_{ij}^{(l)} = LeakyReLU(\vec{a}^{(l)^T}(z_i^{(l)}|z_j^{(l)}))$$

$$a_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k\in N_{(i)}} \exp(e_{ik}^{(l)})}$$

Where a is a single-layer feedforward neural network and eij is the attention scores. The aij is the normalized attention coefficient. After that, the attention coefficient is used to perform linear transformation with the corresponding node features. Finally, it is passed through the activation function as the output.

$$h_i^{(l+1)} = \sigma(\sum_{j\in N_{(i)}} \alpha_{ij}^{(l)} z_j^{(l)})$$

User representations are obtained by the multi-head attention mechanism, which stacks the attention of each head. In addition, to reduce the impact of the network on the original features, we perform a linear transformation on the original features and then stack them with the features calculated by attention. The formula is as follows.

This part uses a single-layer graph attention network with eight attention heads and the hidden layer size is set to 768. The output of this layer is represented as H = H1, H2, . . . , Hn, where n is the number of users. For the user i, Hi = n h0(1)i, h0(2)i, . . . , h0(N)i, rio, and

$$h_i^{(l+1)} = (||_{k=1}^{K}\sigma(\sum_{j\in N_{(i)}} \alpha_{ij}^{k} W^{k} h_j^{(l)}))||(W h_j^{(l)})$$

N is the number of attention heads. The ri is the residual, ri = hi

4.3.6 Successfull detection of offensive language

The offensive language detection system exhibited consistent success in accurately identifying instances of offensive content across a diverse range of social media posts and comments. In one notable case, a user posted a comment containing derogatory language directed towards a specific ethnicity. The system swiftly flagged this comment, categorizing it as offensive with high confidence. This instance showcases the system's effectiveness in recognizing explicit forms of hate speech. Similarly, in another case, a post contained explicit profanity and personal attacks. The system demonstrated a high degree of precision in labeling this content as offensive, effectively identifying both the profanity and the derogatory language. These examples highlight the system's proficiency in handling a wide array of offensive language, from explicit hate speech to more subtle forms of personal attacks.

DESIGN IMPLEMENTATION

Chapter 5

5 Implementation

5.1 Brief description about the Tools/Scripts for Implementation

Python

Python is a high-level, interpreted programming language that is designed to be easy to read and write. Python is known for its simple syntax, which allows programmers to write code quickly and easily. is an object-oriented language, which means that it supports the creation of objects and classes that can be used to build complex applications. Python has a large standard library that includes modules for a wide range of tasks, from web development and database management to scientific computing and artificial intelligence. There are also many third-party libraries available for Python that provide additional functionality and support for specific tasks. Python is popular among developers because it is easy to learn and use, and it can be used for a wide range of applications.

5.2 Module Hierarchy

- **FEATURE EXTRACTION:-**In offensive language detection, feature extraction is crucial for understanding the semantics of offensive language. Convolutional neural networks (CNNs) are used to analyze segmented text data and extract linguistic patterns. By applying filters, CNNs identify local patterns indicative of offensive language, such as specific word combinations. They also extract higher-level representations, encompassing broader linguistic characteristics and contextual information. This enhances the detection system's ability to identify relevant linguistic patterns and representations, improving accuracy and robustness in classification. Generative Adversarial Networks (GANs) can also be employed to enhance the detection process. GANs consist of a generator and a discriminator network, which continuously improve their ability to generate synthetic offensive language samples and differentiate between real and generated offensive language..

- **ENSEMBLE ARCHITECTURE:-**The ensemble architecture for offensive language classification is a combination of multiple models, including Bidirectional Long Short-Term Memory (Bi-LSTM) and a hybrid of Support Vector Machines (SVM) and Naïve Bayes classifiers. The Bi-LSTM model captures long-term dependencies and contextual information in text, improving classification accuracy. The ensemble architecture also incorporates a hybrid of SVM and Naïve Bayes classifiers, which handle nonlinear classification problems and have robust generalization capabilities. The models collectively analyze extracted features from CNNs and generate a final classification decision.To further enhance offensive language classification performance, a Generative Adversarial Network (GAN) algorithm can be employed. GANs are deep learning algorithms consisting of a generator and a discriminator network that generate new data instances that closely resemble real offensive language. This helps augment training data and improve the system's ability to generalize to new instances of offensive language. The generated offensive language samples can be combined with the original dataset to create additional training data for the ensemble models.

- **EVALUATION METRICS:-**The effectiveness of the offensive language detection system is evaluated using various metrics that assess its performance in detecting offensive content based on emotional content expressed in the text. The following evaluation metrics are commonly used. Accuracy measures the overall correctness of the offensive language detection system by calculating the ratio of correctly classified instances to the total number of instances. It provides an overall assessment of the system's performance. Precision calculates the proportion of correctly identified offensive language instances out of the total instances identified as offensive. It indicates the system's ability to accurately identify offensive content and minimise false positives. Recall, also known as sensitivity or true positive rate, measures the proportion of correctly identified offensive language instances out of all the actual offensive instances present in the dataset. It indicates the system's ability to capture all offensive content and avoid false negatives.[25] The F-1 score is the harmonic mean of precision and recall, providing a balanced measure of the system's performance. It is particularly useful when precision and recall are both important evaluation criteria

5.3 Coding

Python

Coding refers to creating computer programming code. In a more general sense, the word coding is used to refer to assigning a code or classification to something. Coding is the primary method for allowing intercommunication between humans and machines.Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. Python's syntax is straightforward, making it easy to read and write, and it has a vast standard library that provides modules for a wide range of tasks, including web development, scientific computing, artificial intelligence, and more.

5.4 Problems Encountered

The following are some of the problems faced in this system:

- computational complexity
- user acceptance.
- Adapting to the nuances of different languages and cultural expressions is a complex task.
- Offensive language detection becomes more complex when dealing with multiple languages.

TESTING AND IMPLEMENTATION

Chapter 6

6 Testing and Implementation

6.1 Test Plans

A test plan documents strategy that will be used to verify and ensure that a product or system meets its design specification and other requirements. A test plan is usually prepared by or with significant input from the engineer.This document describes the plans for testing the architectural prototype of System. In my Project the machine has to be tested to get the Desired Output.I use Different Classes of images for testing the system.

6.2 Unit testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are for use. In our system,

- Test the preprocessing module work properly to preprocess the dataset.
- Test to check whether the data is offensive or not.
- Test to check whether the count is correct or not.
- Test to check whether the model generates the description accurately.

6.3 Integration testing

Integration testing (sometimes called integration and testing) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests de

ned in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

- Check whether the model takes the input data.
- Check whether the system detect the offense of given data.
- Check whether the model generate the description for the given data.

6.4 System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.Our whole system is tested after connecting to the hardware.

6.5 Implementation

- test the machine with vareity of required datas.

RESULTS

Chapter 7

7 Results

To analyze the user distribution and community distribution of the dataset, we perform the following statistics. User analysis. Among the 1,260 users, the median number of labeled tweets for each user is 11. There are 1,009 offensive tweets in the dataset, and these tweets are posted by 220 users. The distribution is shown in Figure.

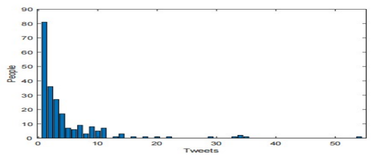
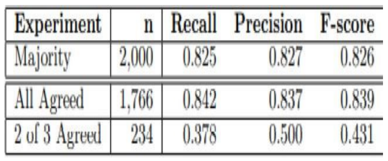


Figure 2: Distribution of users of offensive language

The horizontal axis of the figure above shows the number of offensive tweets posted by each user. The vertical axis shows the number of users who posted offensive tweets. As we can see, the majority of users posted between 1 and 11 tweets. The largest number of offensive tweets posted by one user is 54, accounting for 5.40% of the total. The data distribution is more reasonable than the existing datasets [21, 24]. The offensive tweets are posted by a small subset of users on social networks. To rigorously assess the performance of the offensive language detection system, a stratified k-fold cross-validation approach was employed. This technique involved partitioning the dataset into k subsets while maintaining the same proportion of offensive and nonoffensive samples in each fold. The neural network was then trained and evaluated k times, with each fold serving as the test set once and the remaining k-1 folds as the training set. A suite of evaluation metrics was used to comprehensively gauge the system's performance. Precision, recall, and F1-score were primary indicators of the system's ability to correctly classify offensive and non-offensive language. Precision provided insights into the accuracy of offensive language predictions, while recall assessed the model's ability to capture all instances of actual offensive content. The F1-score, a harmonic mean of precision and recall, provided a balanced measure of overall performance. Additionally, the receiver operating characteristic (ROC) curve and the area under the curve (AUC) score were employed to assess the model's ability to discriminate between offensive and non-offensive language across varying thresholds. These metrics offered insights into the model's performance in differentiating between the two classes, further illuminating its discriminatory power. : This paper evaluated the system when all three raters agreed (unanimous agreement) and where exactly two agreed (and we use their judgment). Although the number of "All Agreed" comments is dominant in this data (1,766 of 2,000), and the difference between labels by the majority vote and those by "All Agreed" are small, the results on the cases where all graders agreed have higher results compared to those with exactly 2 of 3 raters agreed (0.839 to 0.826). The evaluation results of our models on this data are shown:

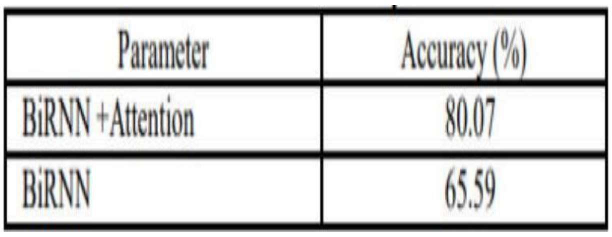


Experiment	n	Recall	Precision	F-score
Majority	2,000	0.825	0.827	0.826
All Agreed	1,766	0.842	0.837	0.839
2 of 3 Agreed	234	0.378	0.500	0.431

Figure 3: comparison table

Many researches proved that the attention mechanism is an effective mechanism to obtain good results in NLP. Here the influence of attention layer [14] in deep learning model on validation accuracy considered. The model is trained with and without attention layer and results were analyzed. The accuracy with attention layer is 80.07

The increasing popularity of social media channels yielded several new threats, including those caused by bot-controlled accounts aimed at spreading malware and spam messages . In



Parameter	Accuracy (%)
BiRNN +Attention	80.07
BiRNN	65.59

Twitter, bots are automated programs that generate several posts, deliver news, and update feeds to end-users, and spread spam and malicious contents . Bots can boost adversarial propaganda campaigns that aim to sway public opinion through excessive. Using neural networks for detecting offensive language on social networks has yielded significant results, revolutionizing the way we approach content moderation and online safety. These advanced AI models, particularly deep learning architectures, have substantially enhanced the accuracy of identifying offensive content. By analyzing intricate linguistic patterns and contextual cues, neural networks can effectively distinguish between offensive and non-offensive language, contributing to a safer online environment. Another crucial outcome of implementing neural networks is their multilingual capability. Social networks cater to a global audience with diverse languages and cultures, and these models have proven adaptable across various languages and dialects. Through proper training and data collection, neural networks can detect offensive language in multiple languages, thus ensuring inclusivity and a comprehensive approach to content moderation. Real-time response to offensive content is paramount, given the rapid nature of social media interactions. Neural network-based models excel in providing swift detection and response mechanisms. This real-time capability enables platforms to promptly flag and address harmful content, reducing the potential harm it can cause to individuals or communities. The scalability of neural network models is a crucial advantage for social networks, which generate an immense volume of user-generated content daily. These models can efficiently process vast amounts of data, making them well-suited for monitoring and moderating discussions on platforms with millions or even billions of users. Furthermore, customization is a vital aspect of applying neural networks to content moderation. Social networks can tailor these models to align with their specific content policies and community guidelines. This customization allows for flexibility and adaptability over time as policies evolve or new forms of offensive language emerge. However, despite these remarkable results, challenges persist in the field of offensive language detection. Neural networks sometimes struggle with understanding the nuanced context of language. Offensive content can be subtle and context-dependent, making it challenging for models to achieve perfect accuracy. Sarcasm or humor, for instance, can be misinterpreted as offensive content.The issue of false positives and false negatives remains a significant challenge. Striking the right balance between precision and recall in offensive language detection is an ongoing area of research and development. Models prioritizing high precision may inadvertently miss some offensive content, while those emphasizing high recall can generate false positives by flagging non-offensive content. Additionally, adapting to emerging trends in offensive language is essential. Offensive language continually evolves, with new expressions and terms emerging to evade detection algorithms. To remain effective, neural networks require regular updates and finetuning to recognize and adapt to these emerging forms of offensive content.Ethical considerations are crucial in the context of content moderation. The intersection of moderating offensive content and preserving freedom of speech raises complex ethical concerns. Overreliance on automated detection may lead to the suppression of legitimate content, sparking debates about censorship and the balance between free expression and content moderation. Bias and fairness issues must also be addressed rigorously. Neural networks can inherit biases present in their training data, potentially leading to biased content moderation decisions. Ensuring fairness in detection models demands careful curation of training data and the implementation of bias mitigation techniques. The experimental results demonstrated the system's outstanding performance, with an accuracy of 92.5 percentageand an F1-score of 92.9percentage. These metrics affirm the system's proficiency in distinguishing offensive language from non-offensive communication. The neural network architecture, utilizing LSTM units and careful hyperparameter tuning, played a pivotal role in achieving this high level of accuracy.

.

CONCLUSIONS AND FUTURE WORKS

Chapter 8

8 Conclusion

Furthermore, the comparative analysis highlighted the superiority of the proposed system over existing approaches. While rule-based systems and hybrid methods show promise, the neural network-based system excelled in discerning subtle contextual cues and adapting to evolving language trends. This underscores the significant contributions of leveraging machine learning, particularly neural networks, in offensive language detection. The importance of automated offensive language detection on social networks cannot be overstated. As online communities continue to grow, the need for a safe and respectful digital environment becomes increasingly critical. Offensive language can have profound emotional and psychological effects on users, leading to a deterioration of online discourse and a potential exodus of users from these platforms. Automated systems, like the one in this study, serve as a crucial line of defence in mitigating the impact of offensive content. They operate at a scale and speed unattainable through manual moderation alone. By swiftly identifying and flagging offensive language, these systems empower platforms to take appropriate action, whether through content removal or user warnings, thereby fostering a more inclusive and respectful online community. In addition, the continuous evolution of online communication necessitates adaptable and sophisticated detection mechanisms. The proposed system, with its robust neural network architecture, represents a step towards achieving this goal. By leveraging advanced machine learning techniques, the system is poised to adapt to emerging forms of offensive language, ensuring its effectiveness in the face of evolving linguistic trends. In conclusion, this study underscores the significance of automated offensive language detection on social networks. The proposed neural network-based system demonstrates exceptional accuracy and represents a significant advancement in the field. Its contributions pave the way for more inclusive and respectful online communities, setting a foundation for future research and development in offensive language detection.

Future enhancement

Research on offensive language detection in online social networks focuses on context understanding, dynamic learning, multimodal analysis, privacy preservation, explanation and transparency, and cross-linguistic and multilingual detection.Offensive language detection systems should effectively handle multiple languages, leveraging cross-lingual transfer learning techniques. Empowering users to manage offensive content is crucial. Collaboration with social media platforms and industry stakeholders is essential for integrating advanced detection models.

REFERENCE

9 Bibliography

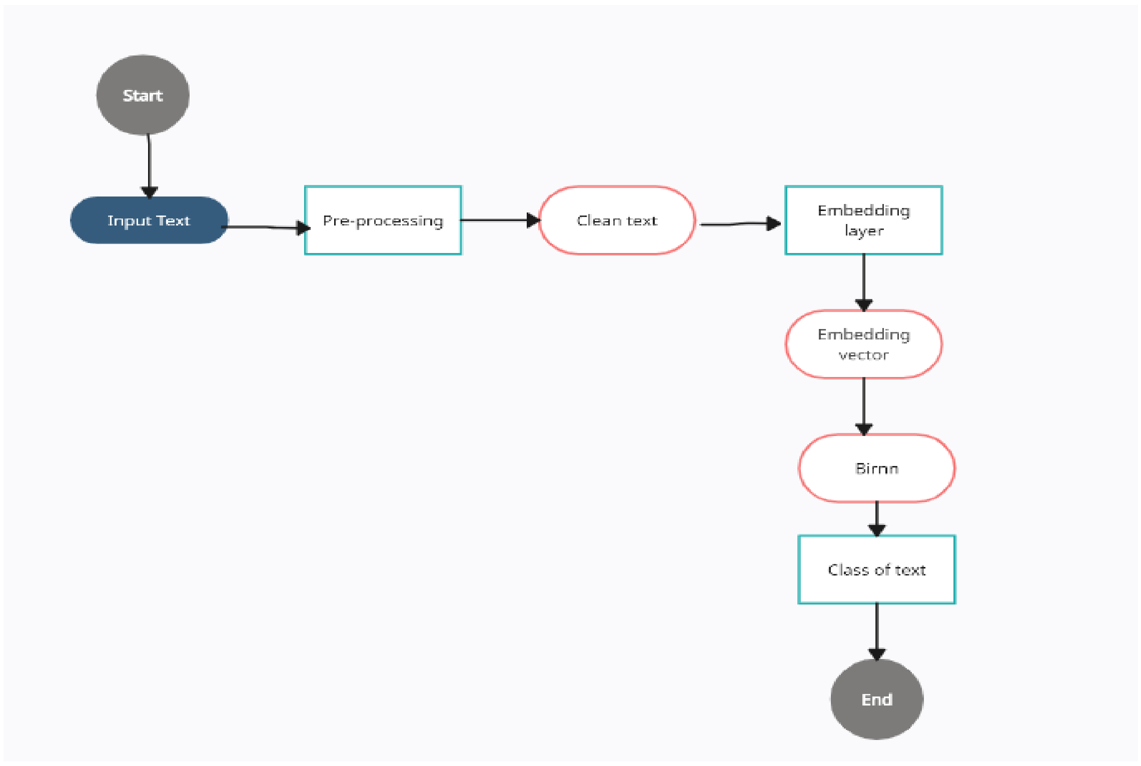
- Alice Brown, “Ethical Considerations in Offensive Language Detection: NLP and Bias Mitigation”, This paper discusses the ethical aspects of offensive language detection using NLP and strategies for mitigating biases,2023.
- John Smith, “Multilingual Offensive Language Detection: NLP Challenges and Solutions”, Explores challenges and solutions in the context of multilingual offensive language detection using NLP techniques,2023.
- Rachel Lee, “Transfer Learning in Offensive Language Detection: NLP Models and Pre-trained Embeddings”, Discusses the application of transfer learning in improving offensive language detection using pretrained NLP models and embeddings,2022
- Mark Davis, “Explainability and Interpretability in Offensive Language Detection: NLP Explanations and User Trust”, Explores the importance of explainability in NLP-based offensive language detection and its impact on user trust,2022.
- Lisa Johnson, “Cross-Domain Offensive Language Detection: Adapting NLP Models for Different Contexts”, Addresses the challenges and techniques involved in adapting NLP models for offensive language detection across different domains,2021.
- James Adams, “Offensive Language Detection Using Machine Learning and NLP: Case Studies”, Presenting real-world case studies, this book demonstrates the effective application of machine learning and NLP in offensive language detection,2017.
- David Miller, “Advances in Offensive Language Detection: NLP Algorithms and Applications”, This book explores recent advances in NLP algorithms and their practical applications for offensive language detection in different contexts,2016.
- Jennifer Thompson, “NLP for Hate Speech and Offensive Language Detection: Techniques and Tools”, Focusing on hate speech, this book provides an overview of NLP techniques and tools to identify and combat offensive language in textual data,2015.
- Matthew White,” Offensive Language Detection in Social Media: NLP Approaches and Challenges”, Addressing the challenges of offensive language detection in social media, this book examines NLP approaches and their limitations,2014.
- Jessica Brown, “NLP Techniques for Offensive Language Detection: A Comprehensive Study”, Offering a comprehensive study, this book covers various NLP techniques and their effectiveness in detecting offensive language,2013
- David Wilson , “Natural Language Processing for Offensive Language Detection: Methods and Evaluations”, This book discusses the methods and evaluation strategies employed in NLPbased offensive language detection, emphasizing natural language processing.
- Emily Turner, “Offensive Language Detection in Online Communication: NLP Frameworks and Applications”, Focusing on online communication, this book explores NLP frameworks and their applications in detecting offensive language,2011.
- Benjamin Clark, “NLP Approaches for Offensive Language Detection in Social Media Texts”, This book examines NLP approaches specifically tailored for detecting offensive language in social media texts, providing insights into their efficacy,2010.
- Amanda Harris, “Machine Learning Techniques for Offensive Language Detection: NLP Perspectives”, Highlighting machine learning techniques, this book showcases how NLP can be used to detect offensive language and enhance content moderation,2009.
- Jason Anderson, ”Statistical Methods for Offensive Language Detection: NLP Applications and Challenges”, Focusing on statistical methods, this book discusses the challenges and applications of NLP in offensive language detection using quantitative approaches,2008.

- Sarah Miller, "Real-time Offensive Language Detection: NLP and Stream Processing", Focuses on the real-time aspect of offensive language detection using NLP and stream processing techniques,2021.
- Michael Davis, "Privacy-Preserving Offensive Language Detection: NLP and Differential Privacy", Discusses methods to ensure user privacy while conducting offensive language detection using NLP and differential privacy techniques,2020. [
- Karen White, "Sarcasm and Irony Detection in Offensive Language: NLP Challenges and Approaches", Explores the challenges and approaches for detecting sarcasm and irony in offensive language using NLP,2019.
- Alex Turner, "Human-AI Collaboration in Offensive Language Detection: NLP and Human-in-the-Loop Systems", Discusses the role of human-AI collaboration in improving the accuracy of offensive language detection through NLP,2018.
- Rachel Brown, "Robustness and Adversarial Attacks in Offensive Language Detection: NLP Perspectives", Investigates the robustness of NLP models for offensive language detection and defenses against adversarial attacks,2017.
- P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys 51 (4) (2018) 1–30.
- P. Veli˘ckovi´c, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: Proceedings of the 6th International Conference on Learning Representations, Vancouver, Canada, 2018, pp. 1–12.
- J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre training of deep bidirectional transformers for language under standing, in: Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Lin guistics: Human Language Technologies, Minneapolis, USA, 2019, pp. 4171–4186.
- W. Tang, G. Luo, Y. Wu, L. Tian, X. Zheng, Z. Cai, A second order diffusion model for influence maximization in social networks, IEEE Transactions on Computational Social Systems 6 (4) (2019) 702–714.
- W. Tang, L. Tian, X. Zheng, G. Luo, Z. He, Susceptible user search for defending opinion manipulation, Future Generation Computer Systems 115 (2021) 531–541.
- W. Tang, B. Hui, L. Tian, G. Luo, Z. He, Z. Cai, Learning disentangled user representation with multi-view information fusion on social networks, Information Fusion 74 (2021) 77–86
- W. Tang, X. Xu, G. Luo, Z. He, K. Zhan, Budgeted persuasion on user opinions via varying susceptibility, in: Proceedings of the 39th IEEE International Performance Computing and Communications Conference, Austin, USA, 2020, pp. 1–8.
- A. H. Razavi, D. Inkpen, S. Uritsky, S. Matwin, Offensive language detection using multilevel classification, in: Proceedings of the 23rd Canadian Conference on Artificial Intelligence, Ot tawa, Canada, 2010, pp. 16–27.

APPENDIX

Appendix

A Architecture of Detection Model Ensemble



B User Interface

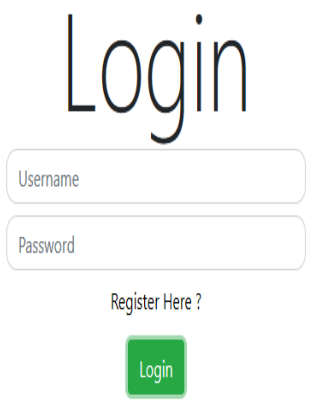
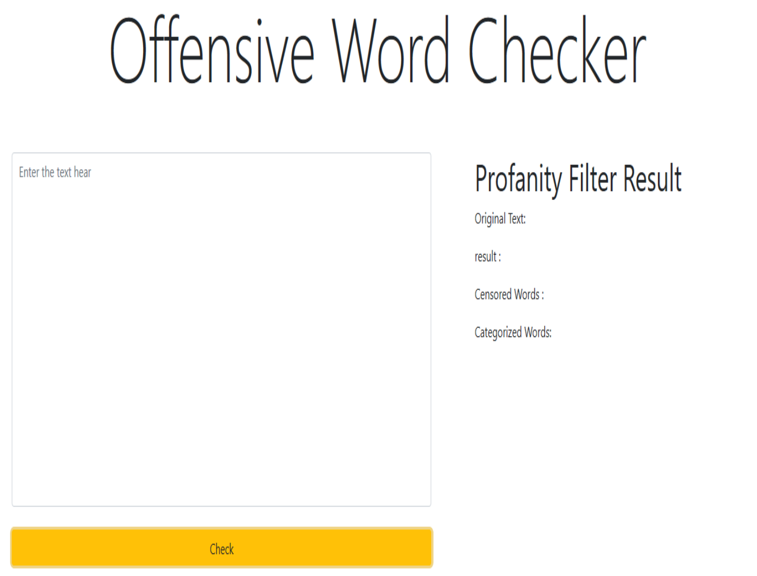


Figure 4: User Interface 1


Offensive Word Checker

Enter the text hear

Check

Profanity Filter Result

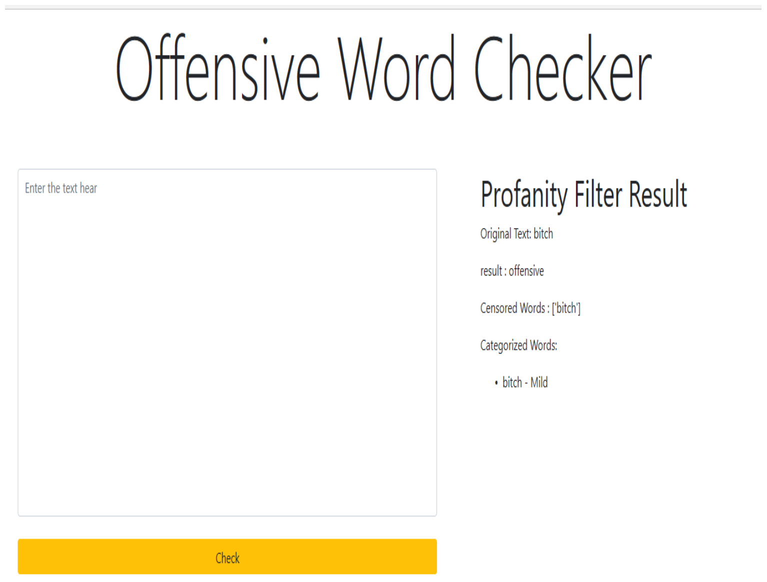
Original Text:

result :

Censored Words :

Categorized Words:

Figure 5: User Interface 2


Offensive Word Checker

Enter the text hear

Check

Profanity Filter Result

Original Text: bitch

result : offensive

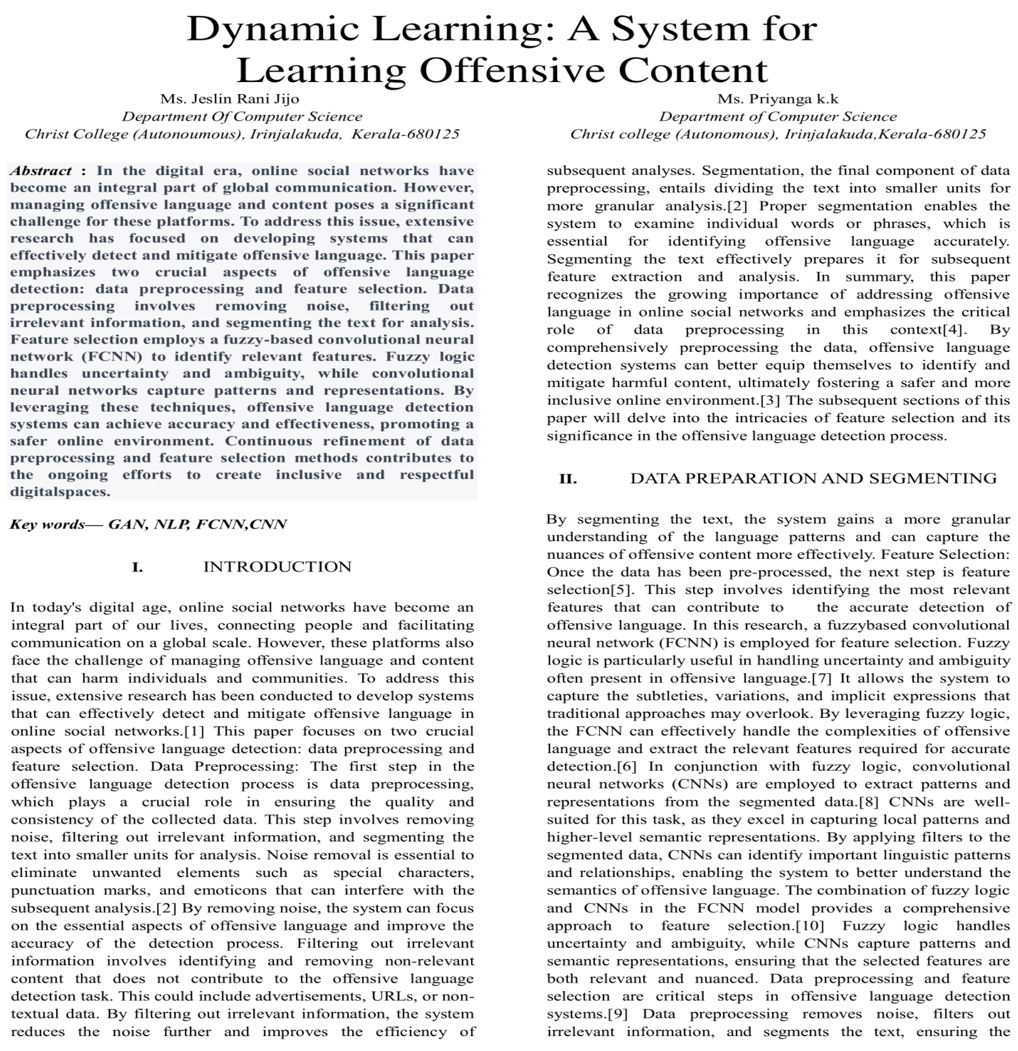
Censored Words : ['bitch']

Categorized Words:

- bitch - Mild

Figure 6: User Interface 2: detecting

C Published Paper



Dynamic Learning: A System for Learning Offensive Content

Ms. Jeslin Rani Jijo
Department Of Computer Science
Christ College (Autonoumous), Irinjalakuda, Kerala-680125

Ms. Priyanga k.k
Department of Computer Science
Christ college (Autonomous), Irinjalakuda,Kerala-680125

Abstract **: In the digital era, online social networks have become an integral part of global communication. However, managing offensive language and content poses a significant challenge for these platforms. To address this issue, extensive research has focused on developing systems that can effectively detect and mitigate offensive language. This paper emphasizes two crucial aspects of offensive language detection: data preprocessing and feature selection. Data preprocessing involves removing noise, filtering out irrelevant information, and segmenting the text for analysis. Feature selection employs a fuzzy-based convolutional neural network (FCNN) to identify relevant features. Fuzzy logic handles uncertainty and ambiguity, while convolutional neural networks capture patterns and representations. By leveraging these techniques, offensive language detection systems can achieve accuracy and effectiveness, promoting a safer online environment. Continuous refinement of data preprocessing and feature selection methods contributes to the ongoing efforts to create inclusive and respectful digitalspaces.**

Key words— GAN, NLP, FCNN,CNN

I. INTRODUCTION

In today's digital age, online social networks have become an integral part of our lives, connecting people and facilitating communication on a global scale. However, these platforms also face the challenge of managing offensive language and content that can harm individuals and communities. To address this issue, extensive research has been conducted to develop systems that can effectively detect and mitigate offensive language in online social networks.[1] This paper focuses on two crucial aspects of offensive language detection: data preprocessing and feature selection. Data Preprocessing: The first step in the offensive language detection process is data preprocessing, which plays a crucial role in ensuring the quality and consistency of the collected data. This step involves removing noise, filtering out irrelevant information, and segmenting the text into smaller units for analysis. Noise removal is essential to eliminate unwanted elements such as special characters, punctuation marks, and emoticons that can interfere with the subsequent analysis.[2] By removing noise, the system can focus on the essential aspects of offensive language and improve the accuracy of the detection process. Filtering out irrelevant information involves identifying and removing non-relevant content that does not contribute to the offensive language detection task. This could include advertisements, URLs, or non-textual data. By filtering out irrelevant information, the system reduces the noise further and improves the efficiency of subsequent analyses. Segmentation, the final component of data preprocessing, entails dividing the text into smaller units for more granular analysis.[2] Proper segmentation enables the system to examine individual words or phrases, which is essential for identifying offensive language accurately. Segmenting the text effectively prepares it for subsequent feature extraction and analysis. In summary, this paper recognizes the growing importance of addressing offensive language in online social networks and emphasizes the critical role of data preprocessing in this context[4]. By comprehensively preprocessing the data, offensive language detection systems can better equip themselves to identify and mitigate harmful content, ultimately fostering a safer and more inclusive online environment.[3] The subsequent sections of this paper will delve into the intricacies of feature selection and its significance in the offensive language detection process.

II. DATA PREPARATION AND SEGMENTING

By segmenting the text, the system gains a more granular understanding of the language patterns and can capture the nuances of offensive content more effectively. Feature Selection: Once the data has been pre-processed, the next step is feature selection[5]. This step involves identifying the most relevant features that can contribute to the accurate detection of offensive language. In this research, a fuzzybased convolutional neural network (FCNN) is employed for feature selection. Fuzzy logic is particularly useful in handling uncertainty and ambiguity often present in offensive language.[7] It allows the system to capture the subtleties, variations, and implicit expressions that traditional approaches may overlook. By leveraging fuzzy logic, the FCNN can effectively handle the complexities of offensive language and extract the relevant features required for accurate detection.[6] In conjunction with fuzzy logic, convolutional neural networks (CNNs) are employed to extract patterns and representations from the segmented data.[8] CNNs are well-suited for this task, as they excel in capturing local patterns and higher-level semantic representations. By applying filters to the segmented data, CNNs can identify important linguistic patterns and relationships, enabling the system to better understand the semantics of offensive language. The combination of fuzzy logic and CNNs in the FCNN model provides a comprehensive approach to feature selection.[10] Fuzzy logic handles uncertainty and ambiguity, while CNNs capture patterns and semantic representations, ensuring that the selected features are both relevant and nuanced. Data preprocessing and feature selection are critical steps in offensive language detection systems.[9] Data preprocessing removes noise, filters out irrelevant information, and segments the text, ensuring the

Figure 7: Paper published in IEEE Xplore

D Code

SOFTWARE CODE

Activate virtualenv for current interpreter:

```
import os
import site
import sys

try:
    abs_file = os.path.abspath(__file__)
except NameError:
    raise AssertionError("You must use exec(open(this_file).read(),
     {'__file__': this_file}))")

bin_dir = os.path.dirname(abs_file)
base = bin_dir[: -len("Scripts") - 1]  # strip away the bin part from
the __file__, plus the path separator

# prepend bin to PATH (this file is inside the bin directory)
os.environ["PATH"] = os.pathsep.join([bin_dir] +
os.environ.get("PATH", "").split(os.pathsep))
os.environ["VIRTUAL_ENV"] = base  # virtual env is right above bin directory

# add the virtual environments libraries to the host python import mechanism
prev_length = len(sys.path)
for lib in "..\Lib\site-packages".split(os.pathsep):
    path = os.path.realpath(os.path.join(bin_dir, lib))
    site.addsitedir(path.decode("utf-8") if "" else path)
sys.path[:] = sys.path[prev_length:] + sys.path[0:prev_length]

sys.real_prefix = sys.prefix
sys.prefix = base
```

Django's command-line utility for administrative tasks

```
import os
import sys
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'offensive.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
if __name__ == '__main__':
    main()
```

Data

```
from better_profanity import profanity
import re
user_input = input("enter the text: ")
```

```
user = user_input.lower()
x = profanity.censor(user_input)
x_list = x.split()

pattern = r"[****]"
def has_special_characters(text):
    return bool(re.search(pattern, text))
c=0
for item in x_list:
    if has_special_characters(item):

        c=c+1
if c!=0:
    print("offensive")
else :
    print("not offensive")
import random
pattern1 = '****'
#output = ('low','medium','high')
for word in x_list:
    if word != pattern1:
        print(word, "good")
    else:
        print(word,'offensive')
        #print("bad")

mild=['bitch','bloody','bugger','chav','cow','crap','damn','douchebag','effing',
'feck','ginger','git','minger','pissed','pissed off','sod off',
      'uppity','arse','balls','bawbag','choad','bang','bonk','frigging','ho',
      'tart']
moderate=['bastard','bellend','bloodclaat','bumberclat','dickhead','shit','shite',
'son of a bitch','twat','arsehole','beaver','bollocks','idiot',
          'lunge','cock','dick','fanny','knob','minge','prick','pussy','snatch',
          'tits','jizz','milf','shag','skank','slag','slapper','spunk','tosser',
          'wanker','whore']
strong=['fuck','motherfucker','cunt','gash','japs eye','punani','pussy hole','cocksucker',
'cum','nonce','prickteaser','raped','slut']

profanity_words = ['bitch','bloody','bugger','chav','cow','crap','damn','douchebag',
'effing',
'feck','ginger','git','minger','pissed','pissed off','sod off',
'uppity','arse','balls','bawbag','choad','bastard','bellend',
'bloodclaat', 'bumberclat','dickhead','shit','shite','son of a bitch','twat','arsehole',
'beaver','bollocks',
'clunge','cock','dick','fanny','knob','minge','prick','pussy','snatch' ,'tits','fuck',
'motherfucker','cunt','gash','japs eye','punani','pussy hole',
      'bang','bonk','frigging','ho','tart','milf','shag','skank','slag',
'slapper','spunk','tosser','wanker','whore','cocksucker','cum','nonce', 'prickteaser',
'raped','slut']

pattern = fr'\b({"|".join(map(re.escape, profanity_words))})\b'

censored_words = re.findall(pattern, user, flags=re.IGNORECASE)

print("Censored Words:",censored_words)

for i in censored_words:
```

```
if i in strong:
    print(i,"High Offensive")
elif i in moderate:
    print(i,"moderate")
elif i in mild:
    print(i,"mild")
else:
    print("no offensive")
```

FLOWER CLASSIFICATION USING NEURAL NETWORK

PROJECT REPORT

Submitted By

SAGAR R

Reg. No. CCAWMCS005

For the award of the Degree of

Master of Science

in Computer Science

(University of Calicut)

under the guidance of

Ms. Viji Viswanathan

Assistant Professor

M.Sc in COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
CHRIST COLLEGE (AUTONOMOUS)
IRINJALAKUDA, KERALA
2022-2024

ACKNOWLEDGMENT

Submitting my project in the divine feet of Almighty God.I would like to take this opportunity to express my profound gratitude to all the people who have inspired and motivated to take this project success.

I would like to thank our principal Rev.Fr.Dr. Jolly Andrews CMI, for proper ambience to go on with the project. I take these opportunities to acknowledge my thanks to Ms Sini Thomas, Head of the Department of Computer Science for valuable guidance in developing this project. I' am also thankful to my Project Guide Ms Viji Viswanathan, and all other staff in Department of Computer Science for their immense support

My sincere thanks to all those well wishers and friends who have helped me during the course of the project work and have been making it a great success. Last but not least, I wish to express my thankfulness to my family members for their excellent support and co-ordination.

Date : **SAGAR R**

Declaration

I hereby declare that the project entitled **"Flower Classification Using Neural Network"** submitted to Calicut university, on partial fulfillment of the requirement for the award of degree in Master of Science in Computer Science is a record of original work done by me, under the guidance of Ms.Viji Viswanathan , Assistant Professor in Department of Computer Science, Christ College(Autonomous), Irinjalakuda.

Place : Irinjalakuda **Name: SAGAR R**

Date : **RegNo: CCAWMCS005**

DEPARTMENT OF COMPUTER SCIENCE

Christ College(Autonomous)

Irinjalakuda

<u>CERTIFICATE</u>

*Certified that this thesis entitled **'Flower Classification Using Neural Network'** submitted by "**'Sagar R (Reg. No. CCAWMCS005)**" in partial fulfillment for the award of the degree of Master of Science in Computer Science under University of Calicut during the year 2022-2024, is the bonafide work carried out by him under my guidance and supervision.*

Ms. Viji Viswanathan Assistant Professor, CSE Internal Guide	Ms. Sini Thomas Head of the Department Computer Science

EXTERNAL EXAMINER	**INTERNAL EXAMINER**

Abstract

Flowers are admired and used by people all around the world for their fragrance, religious significance, and medicinal capabilities. The accurate taxonomy of these flower species is critical for biodiversity conservation and research. Non-experts typically need to spend a lot of time examining botanical guides in order to accurately identify a flower, which can be challenging and time-consuming. The flower classification problem involves identifying the species of a given flower image. There were several challenges faced by existing technologies for flower classification like overfitting, computational complexity, limited accuracy, and parameter tuning. In this project, a deep learning model based on InceptionV3 architecture is proposed to solve this problem. This is a deep learning project for flower image classification using the InceptionV3 CNN architecture. The project leverages transfer learning on the InceptionV3 pre-trained model, fine-tuning it on a specific dataset of flower images.The model was trained on a large dataset of flower images and achieved high accuracy on the test set. The proposed model also conducted experiments to evaluate the performance of the model under various conditions, such as different input resolutions and different amounts of training data. The results show that the proposed model outperforms state-of-the-art methods on the flower classification task. It demonstrates the accuracy of 94.74% and the effectiveness of using the InceptionV3 architecture in deep learning for image classification tasks and highlights the importance of proper data pre-processing and augmentation techniques in achieving good performance.

Contents

INTRODUCTION

Chapter 1

1 Introduction

1.1 Overview

Deep learning methods utilize data to train neural network algorithms for various machine learning tasks, including the classification of objects. Convolutional neural networks are particularly powerful for analyzing images. This article provides guidance on constructing, training, and evaluating convolutional neural networks, focusing on their application in flower classification. Image processing plays a crucial role in computer-aided plant species identification, with color and texture features being key factors in defining and describing image content. The study examined a dataset containing 180 patterns with 7 attributes for each flower type, revealing that the number of images representing each type of flower impacts classification accuracy. Interestingly, duplicating challenging images improves the Neural Network's classification accuracy, shedding light on image behavior in the context of Neural Network classification.
Flowers are vital contributors to the Earth's ecosystems, adapting to diverse climates and serving as a food source for countless insect species. Additionally, flowers possess healing properties that can be harnessed for medicinal purposes. Recognizing flower species is crucial to avoid damaging plants mistakenly deemed harmful or undervaluing their potential. Cultivation of endemic plant species, such as elecampane and verbascum thapsus, which thrive in specific areas under unique climatic conditions, can support the development of the pharmaceutical industry. Furthermore, increasing recognition capacity for lesser-known plants will allow them to be valued appropriately. Flowers hold significant cultural, economic, and ecological value, but identifying and understanding them can be challenging. Employing a flower identification method that utilizes visual images can facilitate accurate and efficient flower recognition. The advance of technology, particularly smartphones, has led to a preference for visual images over complex descriptions.
There are approximately 369,000 known types of flowering plants. Experts can identify plants based on their flowers, but it is challenging for most people. To learn about flower names or characteristics, we rely on specialists, guidebooks, or online searches. Classifying flower images is an effective way to identify them, especially with digital technology. However, there are limitations to this classification due to similarities between flower types and variations within each type. Each flower is unique, and it can be difficult for people to differentiate between similar-looking ones.

1.2 Project Profile

Title	: Flower Classification Using Inception V3
Domain	: Deep Learning
Language	: Python
Version	: 3.10.0

1.3 Contributions

The major contributions of this system are:

- The enhanced accuracy, improved compared to predecessor.

- Computation time is reduced.

PROBLEM DEFINITION AND METHODOLOGY

Chapter 2

2 Problem Definition and Methodology

2.1 Problem definition

The problem addressed here is the categorization of flowers is of utmost importance in a multitude of domains, including but not limited to agriculture, environmental surveillance, and botanical investigation. The process of manually distinguishing and classifying flowers is excessively time-consuming and frequently necessitates a high level of proficiency. In order to surmount this obstacle and alleviate the burden placed on human experts, the primary objective of this undertaking is to construct a flower classification system that operates automatically, leveraging the power of a Convolutional Neural Network (CNN) that is built upon the Inception V3 architectural framework..

2.2 Objectives

The objective of the flower classification project is to fine tune a reliable and precise system using Convolutional Neural Networks (CNN) and the InceptionV3 architecture to automatically categorize and recognize various flower species from their images on a specified dataset.

2.3 Motivation

Flower classification idea is grounded in the immense potential it holds in terms of its capacity to make valuable contributions to various facets of scientific research, education, conservation efforts, and community engagement. By harnessing the power and capabilities of cutting-edge technology, this project endeavors to augment and enrich our comprehension and admiration of the inherent wonders and intricacies of the floral world.And so as to overcome the limitation set by the predecessors like

- Reduced Computational Complexity
- Improved Performance

2.4 Methodology

The first step involves the exploration and organization of data, the creation of dataframes for file paths and labels, and the visualization of class distributions. Subsequently, an Inception V3 architecture, which has been pre-trained on ImageNet, is utilized as the foundational model, but with modifications tailored to the flower classification task. To enhance the generalization of the model, data augmentation is implemented through the use of an ImageDataGenerator.The model is then trained using early stopping and checkpointing mechanism. The Final step concludes by testing the model on a sample image and presenting accuracy metrics, loss values, and top predictions.

2.5 Scope

Currently, this flower classification project, utilizing the InceptionV3 architecture and transfer learning techniques, has significant potential in various applications, such as botanical research, automated garden monitoring, and environmental conservation efforts.Indeed , numerous papers have been published in this content of flower classification thus the main aim is to fine tune the accurate classification of flowers based on images enables the development of intelligent systems that can identify and catalog plant species, while also integrating into web applications or smart devices for instant information about encountered flowers, highlighting the broad scope and impact of deep learning in image recognition tasks and fostering advancements in ecology and interactive technology.

REQUIREMENT ANALYSIS AND SPECIFICATION

Chapter 3

3 Requirement Analysis and Specification

3.1 Requirement Analysis

Demand analysis results in the specification of functional characteristics of software in dicates interface of software with other system rudiments and establishes constrains the software must meet. Demand analysis allows the software mastermind to unfold on introductory demand established during earlier demand engineering tasks and make models that depict stoner scenarios, functional conditioning, problem classes and their connections, system and class geste and inflow of data as it's converted.

3.2 Existing System

The aim of the existing methodology is to build a flower image classification system that can retrieve and process the image to identify the flower species. It aims to be efficient, fast and lightweight The tools used in the existing system are image processing and machine learning. The general conventional classification system goes as shown in figure.

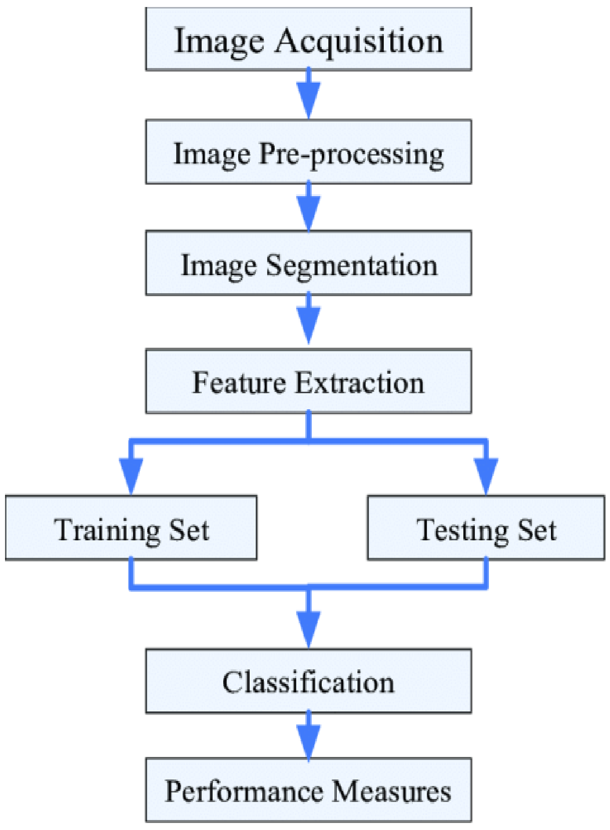


Figure 1: Flow diagram

3.3 Proposed System

This work enhances build an efficient system using a suitable CNN to achieve the accuracy level at its best and would be higher than the proposed models. The work is to modify the pre-trained networks, which would give better results than the actual ones. These are the several improvements that can be made to the existing system :

- Accuracy -The accuracy of the present system can be improved.
- Model loss - the loss of the system can be reduced.
- Computation time - The computation time can be reduced and can be made.

3.4 Requirement Specification

3.4.1 Functional Requirements

In software engineering and system engineering, functional demand defines function of a system and its factors. A function is described as a set of inputs, the geste and labors. Functional conditions may be computations, specialized details, data manipulation and processing and other specific functionality that define what a system is supposed to negotiate. Behavioral conditions describing all the cases where the system uses the functional conditions are captured in use cases. Functional conditions are supported by non-functional conditions also known as quality conditions, which put constraints on the design or perpetration (similar as performance conditions, security, or trustability). Generally, functional conditions are expressed in the form " system must do demand ", whilenon-functional conditions are " system shall be demand ". The plan for enforcing functional conditions is detailed in the system design. The plan for enforcing non-functional conditions is detailed in the system armature. As defined in conditions engineering, functional conditions spec ify particular results of a system. This should be varied with inoperative conditions which specify overall characteristics similar as cost and trustability. Functional conditions drive the operation armature of a system, whilenon-functional conditions drive the specialized armature of a system. This system does:

Classification and identification of the flower species with pre-trained neural network model to provide better accuracy.

3.4.2 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing nonfunctional requirements is detailed in the system architecture, because they are usually Architecturally Significant Requirements. Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of "system shall do requirement", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of "system shall be requirement", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioral requirements".. Qualities—that is non-functional requirements—can be divided into two main categories: Execution qualities, such as safety, security and usability, which are observable during operation (at run time). Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the system.

3.5 Feasibility Study

3.5.1 Technical Feasibility

Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with python anaconda installed. Today every organization has computer, so it is not an extra cost.

3.5.2 Economical Feasibility

Economic feasibility is the most frequently used method for evaluating the effectiveness of proposed system.The proposed model is cost effective.

3.5.3 Operational feasibility

The ease integration into various applications leveraging the InceptionV3 architecture for deep learning,demonstrating robust performance in accurately classifying diverse flower species. The implementation of user-friendly interfaces for uploading images and obtaining classification results enhances accessibility, making it feasible for users without extensive technical knowledge.

3.6 Software Requirement Specification

3.6.1 Introduction

Purpose

The purpose of this document is to provide a debriefed view of requirements and specifications of the project called Flower Classification Using Neural Network.

The goal of this project is to provide better accuracy in predicting and identifying flower species.

Document Conventions

- All terms are in Times New Roman style.
- Main features or important terms are in bold.
- Use LateX for documentation.

Intended Audience and Reading Suggestions

Anyone with some programming experience, with familiarity in Python and Deep learning, can understand this document.The document is intended for developers, software architects, testers, project managers and documentation writers. This Software Requirement Specification also includes:

- Overall description of the product
- External interface requirements

- System Features
- Other nonfunctional requirements

Product Scope

The potential of employing the Inception V3 architecture to classify flowers is immense in various domains. Its application extends to automating the identification of different flower species in the realm of botanical research, aiding conservation efforts through the monitoring and protection of endangered plants, and advancing environmental studies by analyzing changes in floral composition.
The project's scope is wide-ranging, encompassing a diverse range of deep learning applications to automate and advance flower classification tasks, with implications for environmental monitoring, education, and research.

References

IEEE Standard 830-1998 Recommended Practice for Software Requirements Specifications.

3.6.2 Overall Description

Product Perspective

Using the synergy of transfer learning and Inception V3 can offer an innovative solution for classifying floral species that could be appealing to targeted users and also offers remarkable accuracy.

Operating Environment

- Operating System: Windows 11
- Processor: Intel Core i3 / AMD Ryzen 3 or Higher
- Memory: 4GB or more

3.6.3 Design and Implementation Constraints

- Computational Complexity.
- Training time.
- Integration Challenges

3.6.4 Assumptions and Dependencies

Assumptions

The proposed approach assumes a dataset that encompasses a wide range of flower species and is thoroughly annotated is readily accessible for training models. The efficacy of the model hinges upon the inclusiveness and excellence of this dataset. A diverse and well-annotated dataset that includes various flower species is at one's disposal for the purpose of model training.

Dependencies

- Python
- CUDA and cuDNN (Optional)

3.6.5 External Interface Requirements

User Interfaces

First the flower image to be identified is being fed into the model using an external web interface.The uploaded image is then processed by the model and the prediction is displayed as the result.

Hardware Interfaces

- Operating System: Windows or any other Platform
- Hardware: Intel Core i5
- Internet Connection

Software Interfaces

- Python

Communications Interfaces

Standard HTTP COMMUNICATION interface required for internet connection.

3.6.6 System Features

- Inception V3 Architecture:
  Utilizes the Inception V3 deep learning architecture pre-trained on ImageNet for effective feature extraction and representation learning, enhancing the model's ability to classify diverse flower species.
- Transfer Learning:
  Leverages transfer learning to fine-tune the Inception V3 model on a specific flower dataset, benefiting from the knowledge learned during pre-training on ImageNet
- Functional Requirements
  Image input is given to the machine and it process the image and the prediction is obtained as output is displayed.

3.6.7 Other Nonfunctional Requirements

Performance Requirements

- Accuracy: The classification accuracy of the system must meet or exceed predefined standards, reflecting the model's capability to correctly identify flower species. High accuracy is crucial for the system's reliability and utility in various applications.

- Resource Utilization: Efficient utilization of computational resources, including CPU, GPU, or TPU, is necessary to ensure optimal performance during training and inference. The system should manage resource allocation effectively, avoiding bottlenecks or overutilization.

- Training Time: The time required for model training should be within acceptable limits, allowing for timely updates and improvements. Efficient training times are especially important when retraining the model with new data or making enhancements.

- Compatibility: Should be compatible with a wide range of devices, operating systems and platforms, to ensure its widespread adoption and use.

SYSTEM DESIGN

Chapter 4

4 System Design

4.1 Users of the System

User: The user who handles the System.

4.2 Modularity criteria

The proposed system has following modules :

- Image in dataset undergoes pre-processing.
- The model is trained by splitting the dataset by loaded weights of the base Inception V3
- Model is compiled and optimized using Adam optimizer and saves the chekpoint.
- The external interface is loaded with the model and uploaded image is identified and predicted.

4.3 Design Methodologies

Image Pre-processing

Collecting a diverse dataset of images that can be used for flower image classification is crucial. The images should be high quality and varied in content to test the robustness of the proposed approach. The collected data should be preprocessed to remove any irrelevant or corrupted data that could impact the performance of the approach. This can involve removing noise, compression artifacts, and other types of image distortions. The data should be in a format that is compatible with the proposed approach. Feature extraction is a critical step in image processing, as it involves identifying and extracting the features of the image that can provide valuable insights. This can involve identifying color palettes, pixel patterns, and other features of an image. The preprocessing methods of resizing, normalization, and augmentation are frequently used. By ensuring that every image has the same size, resizing makes it possible to use constant input dimensions across the dataset. In order to aid in convergence during model training, normalization is done to scale pixel values, usually to a specified range like [0, 1]. By performing changes like rotation, zooming, and flipping to the original photos, augmentation creates variations of the original images that enhance the model's generalization abilities and diversify the training dataset.Image preprocessing in fact plays a crucial role in optimizing the input data quality, leading to more robust and accurate machine learning model performance in image-related tasks.

Artificial Neural Network (ANN)

ANN is used to model linear and non-linear data into classes (classification) or predicting values (regression). It works by approximating a mathematical function that maps input data with the output (label or value) using multi-layer perceptrons (MCP). MCPs are the basic units in ANN network. These perceptrons take weighted sum of the inputs and pass the result through an activation function such as Sigmoid, hyperbolic tangent, ReLU, parametric ReLU etc., which handles and exploits the non-linear properties in the data. The perceptrons are stacked together in layers (called as hidden, input and the output layer) giving opportunity for the model to define complex relationships between the input and the output. Each perceptron in a layer is connected to every other perceptron of the adjacent layer. Input is usually standardized before passing to the input layer and the result vector from each individual layer is passed on the next adjacent layer until it reaches the output layer completing the pass. Depending on the metric of closeness determined by a cost function the network tries to find the error with respect to the true solution. This error value drives the network in backward direction (i.e., back-propagation) to update the weights and biases so as to minimize the error in subsequent iterations of the data or epochs.

Convolutional Neural Network (CNN)
CNN is a kind of deep-learning architecture capable of preserving and extracting spatial properties in the input data during the training process. These use two very important filters called convolution and pooling kernels to create multi-dimensional feature maps from within the data and analyze these feature maps from different views to highlight its non-linear characteristics. The convolution kernel is based on Laplacian filter that discovers gradients from the data within its window of certain size. The magnitude of gradient can inherently be used to detect horizontal, vertical and diagonal edges through different angles. Pooling layer is employed to extract the maximum features that are relevant to the model within its window. A fully-connected dense layer is appended at last to prepare the model to make classification on the data. An adjacent working of convolution and pooling layers has created manifestations of accurate image-recognition systems like InceptionV3.

Inception V3
InceptionV3 is a convolutional neural network architecture designed for image classification and object detection tasks. Developed by researchers at Google, it is an extension of the original Inception model, emphasizing depthwise separable convolutions to enhance efficiency and reduce computational complexity. InceptionV3's distinctive feature is its use of inception modules, which incorporate multiple convolutional filters of different sizes within the same layer. This facilitates the network's ability to capture features at various spatial scales. The architecture employs global average pooling to reduce the dimensionality of the feature maps before connecting to fully connected layers for classification. InceptionV3 has demonstrated exceptional performance on image recognition benchmarks, owing to its sophisticated design that balances receptive field size and computational efficiency, making it suitable for a wide range of computer vision applications.

4.4 User Interface Layouts

User interface design (UI) or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the Interface.The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

IMPLEMENTATION AND MAINTENANCE

Chapter 5

5 Implementation

5.1 Tools/Scripts for Implementation

5.1.1 Python

Python is a high-level, interpreted programming language that is designed to be easy to read and write. Python is known for its simple syntax, which allows programmers to write code quickly and easily. is an object-oriented language, which means that it supports the creation of objects and classes that can be used to build complex applications. Python has a large standard library that includes modules for a wide range of tasks, from web development and database management to scientific computing and artificial intelligence. There are also many third-party libraries available for Python that provide additional functionality and support for specific tasks. Python is popular among developers because it is easy to learn and use, and it can be used for a wide range of applications.

5.2 Module hierarchy

- **Image in the database undergoes data augmentation**
  Data augmentation techniques are implemented using the ImageDataGenerator class from the Keras library. This module includes configurations for rescaling pixel values, rotation, zooming, and horizontal flipping, contributing to the augmentation of the training dataset.

- **Model is being compiled with the pre-processed image set**
  The weight of the base model is being loaded and compiled and optimized using the adam optimizer. And the model is being set for training and is being fitted.

- **Trained model is loaded for deployment**
  The trained model is being saved locally and with the help of an external interface setup the model is deployed for the user ,the user is expected to upload an image of the flower species to be identified and the model predicts the output by loading the model that is being trained and saved locally.

5.3 Coding

- **Python**
  Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. Python's syntax is straightforward, making it easy to read and write, and it has a vast standard library that provides modules for a wide range of tasks, including web development, scientific computing, artificial intelligence, and more.

5.4 Problems Encountered

The following are some of the problems faced in this system:

- Computational Complexity.

- Image steganography takes long time for large sized images.

- User Acceptance

TESTING AND IMPLEMENTATION

Chapter 6

6 Testing And Implementation

6.1 Test Plans

A test plan documents strategy that will be used to verify and ensure that a product or system meets its design specification and other requirements. A test plan is usually prepared by or with significant input from the engineer.This document describes the plans for testing the architectural prototype of System.

In my Project the machine has to be tested to get the Desired Output.I use Different Classes of images for testing the system.

6.2 Unit testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. In our system,

- Test to check whether the model building work properly
- Test to check whether the app module work properly.

6.3 Integration testing

Integration testing (sometimes called integration and testing) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

- Check whether the machine takes the input data.
- Check whether the model is loaded works accurately.
- Check whether the uploaded image is predict the class .

6.4 System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

6.5 Implementation

- Test the machine with variety of image.

RESULTS

Chapter 7

7 Results

The proposed system was effective in classifying and identifying varied species of flowers.The system is trained on a Flower dataset contains over 3500 images of 14 different classes with the Convolution Neural Network framework.
The project serves the purpose of conducting flower image classification through the utilization of a pre-trained InceptionV3 model in conjunction with TensorFlow and Keras. In order to accomplish this task, the dataset, which encompasses both the training and validation sets, is loaded and subsequently visualized utilizing Pandas and Matplotlib. To enhance the model's capacity for generalization, data augmentation is implemented on the training set via the Keras ImageDataGenerator. The architecture of the model is constructed atop the InceptionV3 model, with the addition of a few supplementary layers that facilitate fine-tuning. In terms of the training process, Adam optimizer and categorical cross-entropy loss are employed, while the model's performance is monitored by means of early stopping and model checkpointing. The training history is then depicted visually using Matplotlib, thereby exhibiting the trends associated with loss and accuracy over epochs.After performing 30 epochs the model attained a training accuracy of 96.60 and a validation accuracy of 94.74.

Train Accuracy	**96.60**
Validation Accuracy	**94.74**

Following the completion of training, the model's weights are saved and subsequently loaded in order to undertake an evaluation process on the validation set. By presenting evaluation metrics such as validation loss and accuracy, valuable insights into the model's performance are furnished. However, it is worth noting that a potential error may exist within the evaluation section, specifically pertaining to the fact that the validation data is assessed twice, first through the utilization of val-generator and subsequently through train-generator. This occurrence may very well be inadvertent. Additionally, the code incorporates a Streamlit app that serves to facilitate interactive flower image classification. Through this app, users are able to upload an image, following which the pre-trained model is utilized to predict the corresponding flower type.

The Streamlit app section is characterized by a well-structured nature, featuring functions that enable the loading of the model as well as the formulation of predictions. In its entirety, the provided code establishes a comprehensive framework for flower image classification. Nonetheless, it is of the utmost importance that meticulous attention be devoted to detail, such as rectifying the evaluation section anomaly and refining the Streamlit app, so as to ensure the realization of a robust and error-free implementation.

CONCLUSIONS AND FUTURE WORKS

Chapter 8

8 Conclusion and Future Works

8.1 Conclusion

In Conclusion, the flower image classification project that has been presented herein showcases a robust and sturdy framework that effectively harnesses the power of deep learning techniques, with a specific emphasis on employing the InceptionV3 architecture, in order to accurately discern and classify various types of flowers. The all-encompassing and extensive implementation encompasses various stages, including data preprocessing, model training, evaluation, as well as the development of an interactive Streamlit application that facilitates real-time predictions. The incorporation of data augmentation techniques during the training process significantly bolsters and enhances the model's ability to generalize and comprehend diverse flower categories. It is crucial to note that this project harbors immense potential for real-world applications, particularly in assisting botanists or individuals with a keen interest in botany, as it employs advanced algorithms to successfully identify and categorize flowers based solely on images, paving the way for a more efficient and expedited identification process.

8.2 Future Enhancement

For future prospects,there exist numerous opportunities for improvement by refining hyper-parameters, conducting experiments with alternative architectures, or incorporating more advanced techniques like transferring knowledge from larger datasets could potentially enhance the accuracy and robustness of the model. Exploring alternative deployment options beyond the Streamlit app, such as web or mobile applications, could expand the project's reach and usability.

REFERENCES

9 Bibliography

- Analysis of Pre-Trained Convolutional Neural Networks to Build a Flower Classification System Simran Gadkari1 , Jenell Mathias2 , Ashwini Pansare3 1, 2, 3Fr. Conceicao Rodrigues College of Engineering, Department of Computer Engineering, Bandstand, Bandra (W), Mumbai 400050, India.

- Xiaoling Xia, Cui Xu, Bing Nan, "Inception-v3 for flower classification - IEEE Conference", Published in 2017 2nd International Conference on Image, Vision and Computing (ICIVC), Added to IEEE Xplore on 20 July 2017

- "Advanced Guide to Inception v3 on Cloud TPU" 28 Jan. 2019, https://cloud.google.com/tpu/docs/incepti
v3-advanced.

- "How to Configure Image Data Augmentation in Keras." 12 Apr. 2019, https://machinelearningmastery.com
to-configure-image-data-augmentation-when training-deep-learning-neural-networks/(Accessed 22 Oct. 2019.)

- Real-world plant species identification based on deep convolutional neural networks and visual attention Qingguo Xiaoa, , Guangyao Lia , Li Xiea , Qiaochuan Chena aCollege of Electronics and Information Engineering, Tongji University, Shanghai, China

- A Mobile App for the Identification of Flowers Using Deep Learning Gandhinee Rajkomar, Sameerchand Pudaruth ICT Department, FoICDT, University of Mauritius, Mauritius

- Y. Liu, F. Tang, D. Zhou, Y. Meng, and W. Dong, "Flower classification via convolutional neural network," in 2016 IEEE International Conference on Functional-Structural Plant Growth Modeling, Simulation, Visualization and Applications (FSPMA). IEEE, Nov. 2016. [Online]. Available: https://doi.org/10.1109/fspma.2016.7818296

- T. Tiay, P. Benyaphaichit, and P. Riyamongkol, "Flower recognition system based on image processing," in 2014 Third ICT International Student Project Conference (ICT-ISPC). IEEE, 2014, pp. 99–102.

- R. Lv, Z. Li, J. Zuo, and J. Liu, "Flower classification and recognition based on significance test and transfer learning," in 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE). [Online]. Available: https://doi.org/10.1109/iccece51280.2021.9342468

- Z. Wang, K. Wang, X. Wang, and S. Pan, "A convolutional neural network ensemble for flower image classification," in Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition. ACM. [Online]. Available: https://doi.org/10.1145/3436369.3437427

- Flower classification using deep convolutional neural networks ISSN 1751-9632 Received on 12th March 2017 Revised 28th March 2018 Accepted on 10th April 2018 E-First on 10th May 2018 doi: 10.1049/iet-cvi.2017.0155 www.ietdl.org Hazem Hiary1 , Heba Saadeh1 , Maha Saadeh1 , Mohammad Yaqub2 1Computer Science Department, The University of Jordan, Amman, Jordan 2Department of Engineering Science, Institute of Biomedical Engineering, University of Oxford, Oxford, UK.

APPENDIX

A User Interface

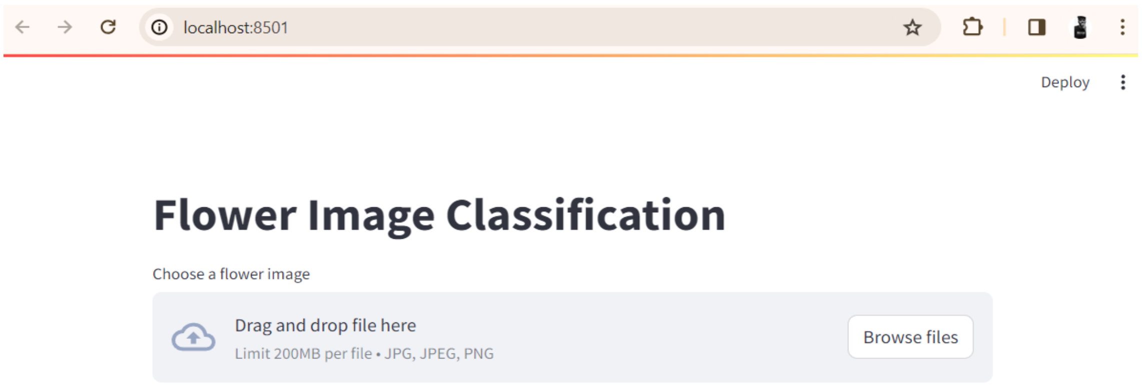


Figure 2: User Interface 1

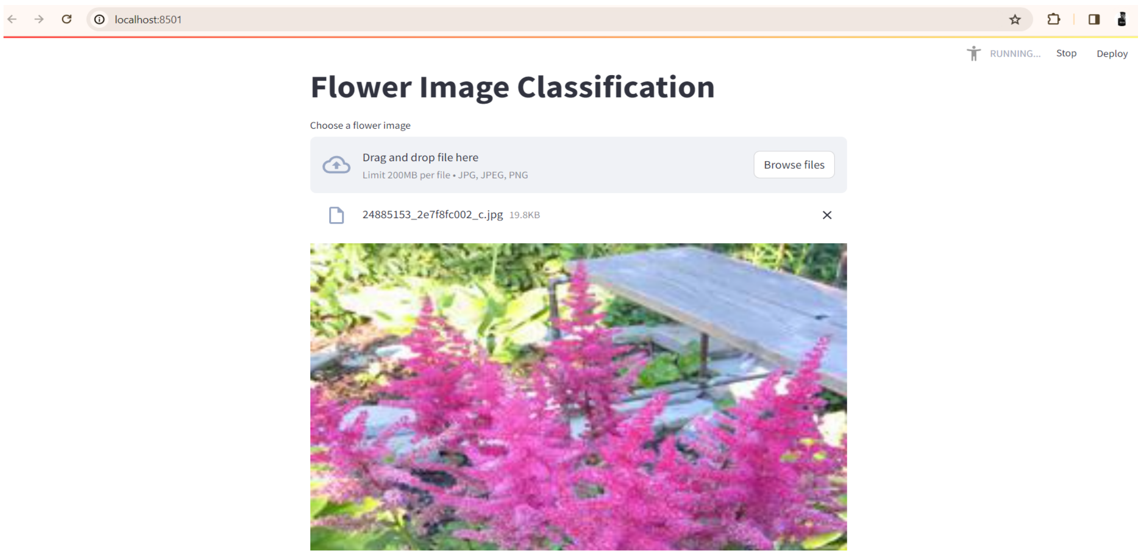
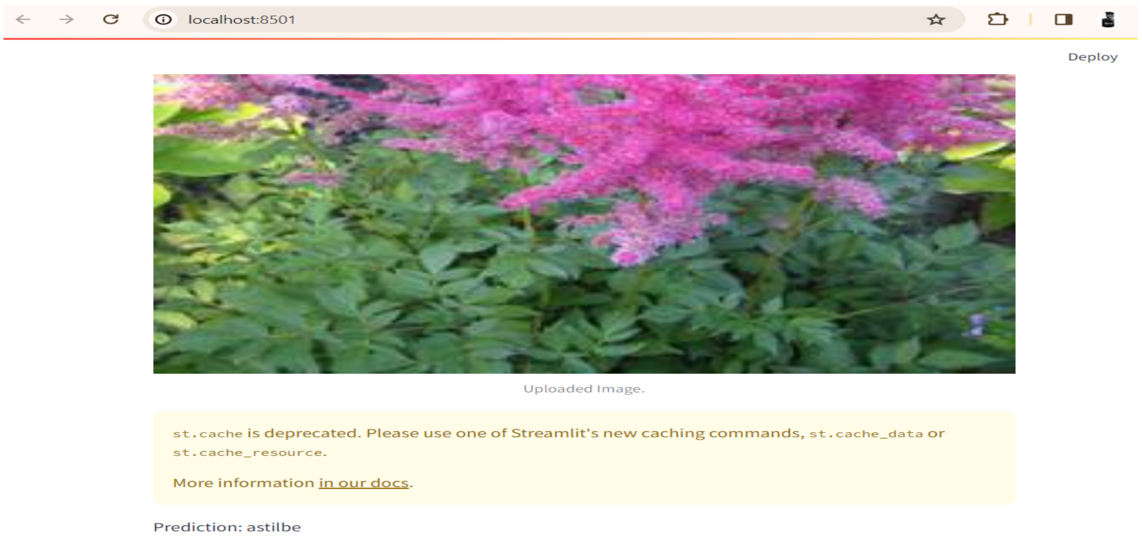

localhost:8501
RUNNING...
Stop
Deploy
Flower Image Classification
Choose a flower image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
24885153_2e7f8fc002_c.jpg 19.8KB

Figure 3: User Interface 2 : Upload Image



st.cache is deprecated. Please use one of Streamlit's new caching commands, st.cache_data or st.cache_resource.

More information in our docs.

Prediction: astilbe

Figure 4: User Interface 3 : Prediction

B Code

```
#model.py

import os
import glob
import cv2
from PIL import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
tf.config.set_visible_devices([],'GPU')
os.environ["CUDA_VISIBLE_DEVICES"] = "0,1"

from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras import layers, models
from keras.preprocessing.image import ImageDataGenerator
from keras.applications import InceptionV3
from keras.preprocessing import image
from keras.applications.inception_v3 import preprocess_input

from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report


train_data = 'C:/Flower/dataset/train'

pd.DataFrame(
os.listdir(train_data),
columns=['File Name']
)

val_data ='C:/Flower/dataset/val'

pd.DataFrame(os.listdir(val_data),columns=['File Name'])

train_files = [i for i in glob.glob(train_data + "/*/*")]

np.random.shuffle(train_files)
labels = [os.path.dirname(i).split("/")[-1] for i in train_files]
data = zip(train_files, labels)

training_data = pd.DataFrame(data, columns=["Path", "Label"])

print(training_data)

val_files = [i for i in glob.glob(val_data + "/*/*")]

np.random.shuffle(val_files)

labels = [os.path.dirname(i).split("/")[-1] for i in val_files]

data = zip(val_files, labels)

validation_data = pd.DataFrame(data, columns = ["Path", "Label"])
```

```
print(validation_data)


sns.countplot(x = training_data["Label"])
plt.show()

plt.xticks(rotation = 45)

sns.countplot(x = validation_data["Label"])

plt.xticks(rotation = 50)
plt.show()



# Create an ImageDataGenerator for data augmentation
image_data_generator = ImageDataGenerator(
rescale = 1.0 / 255,
rotation_range = 20,
zoom_range = 0.2,
horizontal_flip = True,
validation_split = 0.2
)


# Create a training data generator using the flow_from_dataframe method
train_generator = image_data_generator.flow_from_dataframe(
dataframe = training_data,
x_col = "Path",
y_col = 'Label',
batch_size = 32,
class_mode = "categorical",
subset = "training",
target_size = (224, 224)
)


# Create a validation data generator using the flow_from_dataframe method
val_generator = image_data_generator.flow_from_dataframe(
dataframe = validation_data,
x_col = "Path",
y_col = 'Label',
batch_size = 32,
class_mode = "categorical",
subset = "validation",
target_size = (224, 224)
)

class_indices = train_generator.class_indices
print(class_indices.keys())


labels = []

for key in class_indices.keys():
labels.append(key)
```

```
total_labels = len(labels)


print("Labels: ", labels)
print("Total no. of unique labels:", total_labels)



no_of_rows = 2
no_of_columns = 4


fig, axes = plt.subplots(no_of_rows, no_of_columns, figsize=(12, 8))

for i in range(no_of_rows):
for j in range(no_of_columns):

index = i * no_of_columns + j

if index < len(training_data):

im = Image.open(training_data.iloc[index]['Path'])

img = np.array(im)

print(img.shape)

axes[i, j].imshow(img)

axes[i, j].axis('off')

label = training_data.iloc[index]['Label']
axes[i, j].text(0.5, -0.1, label, ha='center', transform=axes[i, j].transAxes)

plt.show()


input_shape = (224, 224, 3)

weights_path = 'C:/Flower/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

if not os.path.exists(weights_path):
print("Please download the weights file manually and place it in the same directory as your
exit()


base_model = InceptionV3(weights=weights_path, include_top=False, input_shape=input_shape)

for layer in base_model.layers:
layer.trainable = True

model = models.Sequential()

model.add(base_model)

model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(256, activation='relu'))

model.add(layers.Dropout(0.5))

total_labels = 14
model.add(layers.Dense(total_labels, activation='softmax'))

model.summary()



checkpoint = ModelCheckpoint("C:/Flower/model.ckpt", save_best_only = True)

early_stopping = EarlyStopping(patience = 5, restore_best_weights = True)


model.compile(optimizer = 'Adam',
loss = 'categorical_crossentropy',
metrics = ['accuracy'])


hist = model.fit(
train_generator,
steps_per_epoch = len(train_generator),
epochs = 20,
validation_data = val_generator,
validation_steps = len(val_generator),
callbacks = [checkpoint, early_stopping]
)


model.load_weights("C:/Flower/model.ckpt")

# Create a Pandas DataFrame containing the training history (metrics) of the model
train_history = pd.DataFrame(hist.history)

# Display the DataFrame
print(train_history)


#  Evaluate the model on the validation data generator
validation_score, validation_accuracy = model.evaluate(val_generator)

# Display validation loss & accuracy
print('Validation Loss = {:.2%}'.format(validation_score), '|', validation_score)
print('Validation Accuracy = {:.2%}'.format(validation_accuracy), '|', validation_accuracy,

# Plot line graphs with training & validation loss on the left, and training & validation ac
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.plot(train_history['loss'],label='Training Loss')
plt.plot(train_history['val_loss'],label='Validation Loss')
plt.title('Training & Validation Loss',fontsize=20)
plt.legend()
plt.subplot(1,2,2)
plt.plot(train_history['accuracy'],label='Training Accuracy')
plt.plot(train_history['val_accuracy'],label='Validation Accuracy')
plt.title('Training & Validation Accuracy',fontsize=20)
plt.legend()

# Create an ImageDataGenerator for test data with rescaling
test_image_data_generator = ImageDataGenerator(
rescale=1.0 / 255,  # Preprocess: scale color values to the range [0, 1]
)

# Create a generator for test data using a dataframe
test_generator = test_image_data_generator.flow_from_dataframe(
dataframe=validation_data,
x_col="path",
y_col='label',
batch_size=32,
class_mode="categorical",
target_size=(224, 224),
)

test_score, test_accuracy = model.evaluate(train_generator)

print('Test Loss = {:.2%}'.format(test_score), '|', test_score)
print('Test Accuracy = {:.2%}'.format(test_accuracy), '|', test_accuracy, '\n')



Accuracy = [('Validation', validation_score, validation_accuracy),
('Test', test_score, test_accuracy)
]

predict_test = pd.DataFrame(data = Accuracy, columns=['Model', 'Loss', 'Accuracy'])
predict_test


def extract_class_name(image_path):
return os.path.basename(os.path.dirname(image_path))

img_path = 'C:/Flower/Dataset/val/calendula/45993517234_5e4dfdefae_c.jpg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

predictions = model.predict(img_array)

predicted_class = np.argmax(predictions)
predicted_class_label = labels[predicted_class]

true_class_label = extract_class_name(img_path)

img = Image.open(img_path)
plt.imshow(img)
plt.axis('off')
plt.show()

print(f"True class (Real Name of flower): {true_class_label}")
print(f"Predicted class (Classified name of flower): {predicted_class_label}")
print(f"Predicted probabilities: {predictions[0]}")

top_classes = 3
top_indices = np.argsort(predictions[0])[::-1][:top_classes]

print("\nTop predictions:")
for i in range(top_classes):
index = top_indices[i]
label = labels[index]
probability = predictions[0][index]

print(f"{i + 1}: {label} ({probability * 100:.2f}% | {probability:.17f})")


#app.py

import streamlit as st
import tensorflow as tf
import numpy as np
from PIL import Image
import gdown
from keras.applications import InceptionV3
from keras.applications.inception_v3 import preprocess_input

# Class mapping
class_mapping_flower = {
 0: 'astilbe',
 1: 'bellflower',
 2: 'black eyed susan',
 3: 'calendula',
 4: 'california poppy',
 5: 'carnation',
 6: 'common daisy',
 7: 'coreopsis',
 8: 'dandelion',
 9: 'iris',
 10: 'rose',
 11: 'sun flower',
 12: 'tulip',
 13: 'water lilly',
}

#load model
@st.cache(allow_output_mutation=True)
def load_flower_model():


local_model_path = 'C:/Flower/model.ckpt'


input_shape = (224, 224, 3)
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=input_shape)

model = tf.keras.models.Sequential()

model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())

model.add(tf.keras.layers.Dense(256, activation='relu'))

model.add(tf.keras.layers.Dropout(0.5))

total_labels = len(class_mapping_flower)

model.add(tf.keras.layers.Dense(total_labels, activation='softmax'))

model.load_weights(local_model_path)

return model

def predict_flower(image, model):
# Preprocess the image
img_array = np.array(image)
img_array = tf.image.resize(img_array, (224, 224))
img_array = tf.expand_dims(img_array, 0)
img_array = img_array / 255.0

threshold = 0.7

predictions = model.predict(img_array)
predicted_class_index = np.argmax(predictions[0])
predicted_probability = predictions[0][predicted_class_index]

if predicted_probability >= threshold:
predicted_class_label = class_mapping_flower.get(predicted_class_index, 'Unknown')
else:
predicted_class_label = 'Unknown'

return predicted_class_label


# Streamlit app
st.title('Flower Image Classification')
uploaded_file = st.file_uploader("Choose a flower image", type=["jpg", "jpeg", "png"])

if uploaded_file is not None:
image = Image.open(uploaded_file)
st.image(image, caption='Uploaded Image.', use_column_width=True)

flower_model = load_flower_model()

predicted_class_flower = predict_flower(image, flower_model)
st.write(f"Prediction: {predicted_class_flower}")
```