Video Classification and Similar Video Recommendation System

Submitted for partial fulfillment of the requirements

for the award of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

by

Vaddella Mohan	-	**20BQ1A4259**
Sheela Sai Kumar	-	**20BQ1A4251**
Thota Hemanth Krishna	-	**20BQ1A4256**
Kakarla Bhargav	-	**20BQ1A4226**

Under the guidance of

Mrs. K. Deepika, **M. Tech**

Assistant Professor

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Permanently Affiliated to JNTU Kakinada, Approved by AICTE
Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified
NAMBUR (V), PEDAKAKANI (M), GUNTUR – 522 508
Tel no: 0863-2118036, url: www.vvitguntur.com

April 2024

DECLARATION

We, V. Mohan, S. Sai Kumar, T. Hemanth Krishna, K. Bhargav, hereby declare that the Project Report entitled **"Video Classification and Similar Video Recommendation system"** done by us under the guidance of Mrs. K. Deepika, Assistant Professor, Computer Science Engineering - Artificial Intelligence & Machine Learning at Vasireddy Venkatadri Institute of Technology is submitted for partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering - Artificial Intelligence and Machine Learning. The results embodied in this report have not been submitted to any other University for the award of any degree.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE (S)

ACKNOWLEDGEMENT

We take this opportunity to express my deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express my deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B.Tech programme.

We express my sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B.Tech programme.

We express my sincere gratitude to **Dr. K. Suresh Babu**, Professor & HOD, Computer Science Engineering-Artificial Intelligence & Machine Learning, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express my sincere gratefulness to our Guide **Mrs. K. Deepika**, Assistant Professor, Computer Science Engineering-Artificial Intelligence & Machine Learning for her insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator **Mr. N. Balayesu**, Assistant Professor, Computer Science Engineering-Artificial Intelligence & Machine Learning, for his valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

We would like to take this opportunity to express my thanks to the **Teaching and Non-Teaching** Staff in the Department of Computer Science Engineering-Artificial Intelligence & Machine Learning, VVIT for their invaluable help and support.

Name (s) of Students:

Vaddella Mohan (20BQ1A4259)

Sheela Sai Kumar (20BQ1A4251)

Thota Hemanth Krishna (20BQ1A4256)

Kakarla Bhargav (20BQ1A4226)

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
Permanently Affiliated to JNTUK, Kakinada, Approved by AICTE
Accredited by NAAC with 'A' Grade, ISO 9001:20008 Certified
Nambur, Pedakakani (M), Guntur (Gt) -522508
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE& MACHINE LEARNING

CERTIFICATE

This is to certify that this **Project Report** is the bonafide work of **V. Mohan, S. Sai Kumar, T. Hemanth Krishna, K. Bhargav** bearing Reg. No. **20BQ1A4259, 20BQ1A4251, 20BQ1A4256, 20BQ1A4226** respectively who had carried out the project entitled **"Video Classification and Similar Video Recommendation system"** under our supervision.

Project Guide
(Mrs. K. Deepika, Assistant Professor)

Head of the Department
(Dr. K. Suresh Babu, Professor)

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

ANN	Approximation Nearest Neighbors
API	Application Programming Interface
CNN	Convolution Neural Network
CSS	Cascading Style Sheets
DL	Deep Learning
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
ISRO	Indian Space Research Organization
JSON	JavaScript Object Notation
LRCN	Long Term Recurrent Neural Network
LSTM	Long Short-Term Memory
ML	Machine Learning
RNN	Recurrent Neural Network

ABSTRACT

The digital age has created mountains of video content, making organization and retrieval challenging. Our research tackles this issue by developing a video classification model for the Indian Space Research Organization (ISRO) using Long-term Recurrent Convolutional Networks (LRCN). The 'Annoy' algorithm also serves as a recommendation system, suggesting similar videos based on user interest. Our proposed video classification system for identifying action sequences utilizes two key methodologies. LRCN Model: The system breaks down videos into frames, extracts spatial features with CNN, and captures temporal context with LSTM. Annoy Algorithm: The system pre-maps existing ISRO videos using their extracted features. The similar videos are then recommended to the user for a better experience. Our LRCN video classification model excelled with a 67% accuracy on the ISRO dataset. Training and Validation Loss curves demonstrate a trend, suggesting the model successfully minimized error during training and generalizes well to unseen data. Our research demonstrates the power of Neural Networks like LRCN in Video Classification. To unlock their full potential: Include diverse ethnicities for broader representation and improved model accuracy in activity recognition. Continuously expand datasets, refine models, and explore new applications to push the boundaries of video category recognition technology. Extend the model beyond the current five categories to enhance its scope.

Keywords: Convolutional Neural Network (CNN),Long-term Recurrent Convolutional Network (LRCN),Long Short- Term Memory(LSTM).

CHAPTER 1
INTRODUCTION

1.1 WHAT IS DEEP LEARNING?

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. Neural networks are modeled after the structureand function of the human brain and consist of layers of interconnected nodes that process and transform data.

The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data. Deep Learning algorithms can automatically learn and improve from data without the need for manual feature engineering.

Deep Learning has achieved significant success in various fields, including image recognition, natural language processing, speech recognition, and recommendation systems. Some of the popular Deep Learning architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Deep Belief Networks (DBNs).

Training deep neural networks typically requires a large amount of data and computational resources. However, the availability of cloud computing and the development of specialized hardware, such as Graphics Processing Units (GPUs), has made it easier to train deep neural networks.

1.2 IMPORTANCE OF DEEP LEARNING IN VIDEO CLASSIFICATOIN:

Feature Learning:

- Deep learning models, particularly Convolutional Neural Networks (CNNs), can automatically learn hierarchical features from raw video frames.

- By learning features directly from the data, deep learning models can adapt to the specific characteristics of the video content, leading to more accurate and robust classifiers.

Temporal Understanding:

- Deep learning models, such as RNNs and LSTMs, are capable of capturing temporal dependencies in video sequences.
- This allows them to understand actions and events that unfold over time, which is essential for tasks like action recognition and event detection in videos.
- By modeling temporal relationships, deep learning models can make more informed decisions about the content and context of video data, improving classification accuracy.

Scalability:

- Deep learning models can scale to large datasets and complex video data.
- This scalability is crucial for developing classifiers that can handle diverse video content, such as videos of varying lengths, resolutions, and quality.
- By leveraging the scalability of deep learning, researchers and developers can create robust classifiers that can generalize well to new and unseen video data.

Accuracy:

- Deep learning models have demonstrated state-of-the-art performance in video classification tasks.
- Compared to traditional machine learning approaches, deep learning models can achieve higher accuracy rates, especially in complex video classification tasks.
- This high level of accuracy makes deep learning models well-suited for applications where precise video classification is required, such as surveillance, healthcare, and entertainment.

Adaptability:

- Deep learning models can adapt to new and unseen video data, making them suitable for real-world applications where video content may vary widely.
- This adaptability allows deep learning models to continuously improve their classification performance as they encounter new video data, leading to more reliable and robust classifiers over time.

Versatility:

- Deep learning models can be applied to a wide range of video classification tasks, including action recognition, event detection, and scene understanding.
- This versatility makes deep learning a powerful tool for video analysis, as it can be used to solve various video classification problems with a single, flexible framework.
- By leveraging the versatility of deep learning, researchers and developers can address diverse video classification challenges, making deep learning a valuable technology for advancing the field of video analysis.

1.3 DEEP LEARNING TECHNIQUES:

There are various deep learning Techniques that bring Machine Learning to a new level, allowing robots to learn to discriminate tasks utilizing the human brain's neural network.

1. **Convolutional Neural Networks (CNNs):** Typically used for image recognition and classification tasks, CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input data.
2. **Recurrent Neural Networks (RNNs):** These are designed to recognize patterns in sequences of data, such as text or speech, and are commonly used in natural language processing (NLP) tasks.
3. **Long Short-Term Memory (LSTM):** A special type of RNN that is capable of learning long-term dependencies, making it particularly useful for tasks involving sequences with long gaps between relevant input events.
4. **Generative Adversarial Networks (GANs):** A framework for training generative models, where two neural networks, the generator and the discriminator, are trained simultaneously to produce high-quality synthetic data.
5. **Autoencoders:** Neural networks designed to learn efficient representations of input data, typically used for tasks such as data compression, denoising, and dimensionality reduction.
6. **Deep Reinforcement Learning:** A combination of deep learning and reinforcement learning, used to train agents to perform tasks in environments where they receive feedback in the form of rewards or punishments.

7. **Capsule Networks:** A type of neural network architecture that aims to improve upon the limitations of CNNs in recognizing hierarchical relationships within objects in images.
8. **Attention Mechanisms:** These mechanisms allow neural networks to focus on specific parts of the input, improving performance in tasks such as machine translation and image captioning.
9. **Transfer Learning:** A technique where a pre-trained neural network is used as a starting point for a new task, often leading to faster training and better performance, especially when the new task has limited training data.
10. **Neuroevolution:** An approach to training neural networks using evolutionary algorithms, where the networks' weights and architectures are evolved over multiple generations to optimize performance on a given task.

1.4 DEEP LEARNING APPLICATIONS:

Deep learning has a wide range of applications across various domains. Some of the key application zones include:

Computer Vision: Deep learning is extensively used in computer vision tasks such as image classification, object detection, segmentation, and image generation. Applications include facial recognition, autonomous vehicles, medical image analysis, and surveillance systems.

Natural Language Processing (NLP): Deep learning is used in NLP for tasks such as speech recognition, language translation, sentiment analysis, and text generation. Applications include virtual assistants, machine translation, and text summarization.

Healthcare: Deep learning is revolutionizing healthcare with applications in medical image analysis, disease diagnosis, personalized treatment planning, and drug discovery. It is used in areas such as radiology, pathology, genomics, and clinical decision support systems.

Finance: Deep learning is used in finance for fraud detection, risk assessment, algorithmic trading, and customer service automation. It helps financial institutions analyze large volumes of data to make informed decisions and improve customer experiences.

Manufacturing and Industry: Deep learning is used in manufacturing for predictive maintenance, quality control, supply chain optimization, and robotics. It helps improve efficiency, reduce downtime, and ensure product quality.

Autonomous Systems: Deep learning is crucial for autonomous systems such as self- driving cars, drones, and robots. It enables these systems to perceive and understand their environments, make decisions, and navigate safely.

Gaming and Entertainment: Deep learning is used in gaming for character animation,game testing, and player behavior analysis. It is also used in entertainment for content recommendation, personalization, and content generation.

Marketing and Sales: Deep learning is used in marketing and sales for customer segmentation, personalized advertising, sales forecasting, and customer relationship management. It helps businesses target the right audience and improve sales performance.

Agriculture: Deep learning is used in agriculture for crop monitoring, yield prediction,disease detection, and farm management. It helps farmers optimize their operations and improvecrop productivity.

1.5 AIM:

The aim of the project is to classify a input video to which category does the given video belongs to and also give recommended videos based on the input.

1.6 PROCESS:

This Project allows users to register or sign in, upload images or videos, which are stored in an S3 bucket with their username. Uploaded videos/images are classified using a pretrained deep learning model, and the result, a category, is displayed. Additionally, the webapp shows recommended videos based on the classification result.

1.7 Features:

- This project classify the input video to category its belongs to.
- Our project provides better predictions on this ISRO dataset.
- It also provides the similar kind of videos based on the ANN algorithm.
- We provide faster and better results.

1.8 EXISTING SYSTEM:

Human action recognition using UCF101 dataset involves analyzing video sequences to classify human actions or activities. This task is similar to video classification in the context of recognizing actions or events in videos. Both tasks often use a combination of convolutional

neural networks (CNNs) and recurrent neural networks (RNNs), such as long short-term memory (LSTM) networks, to extract features from video frames and model temporal dependencies in the data. CNNs are typically used to extract spatial features from individual frames, while LSTMs or other RNN variants are used to capture the temporal dynamics of the video sequence. By combining these two types of networks, researchers can achieve state-of-the-art performance in tasks such as human action recognition and video classification.

1.9 LRCN ARCHITECTURE:

The Long-term Recurrent Convolutional Network (LRCN) is proposed by Jeff Donahue et al. in 2016. It is a combination of CNN and RNN, end-to-end trainable and suitable for large-scale visual understanding tasks such as video description, activity recognition and image captioning. The recurrent convolutional models are deeper in that they learn compositional representations in space and time, when the previous models assumed a fixed visual representation or perform simple temporal averaging for sequential processing. And it can be trained or learn temporal dynamics and convolutional perceptual representations as it is directly connected to convolutional network.

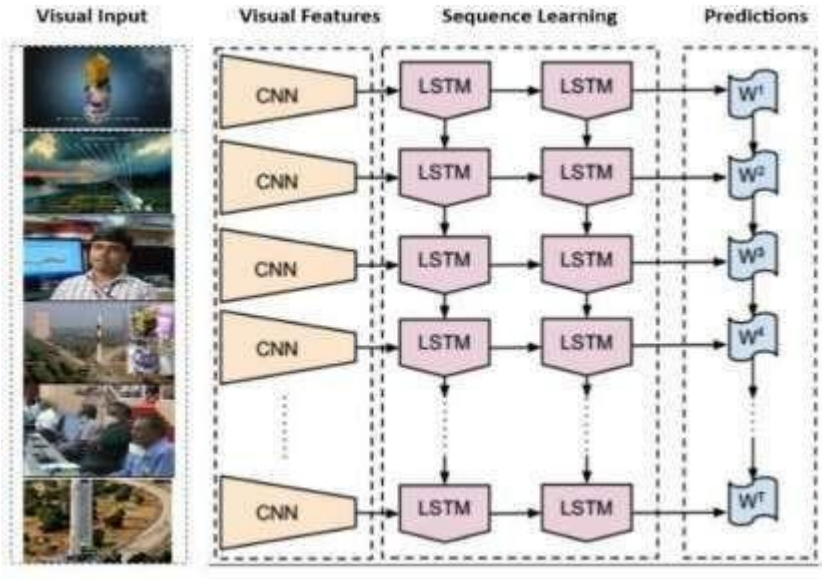


Fig:1.1 LRCN Architecture

Above figure is an example architecture of LRCN. As it is described in the figure, LRCN processes the variable-length visual input with a CNN. And their outputs are fed into a stack of recurrent sequence models which is LSTM in the figure. The final output from the sequence

models is a variable-length prediction. This makes LRCN is proper models to handle tasks with time-varying inputs and output, such as activity recognition, image captioning and video description.

LRCN = CNN +LSTM

CNN:

A convolutional neural network (CNN) is a category of machine learning model, namely a type of deep learning algorithm well suited to analyzing visual data. CNNs -- sometimes referred to as convnets -- use principles from linear algebra, particularly convolution operations, to extract features and identify patterns within images. Although CNNs are predominantly used to process images, they can also be adapted to work with audio and other signal data.

CNN architecture is inspired by the connectivity patterns of the human brain -- in particular, the visual cortex, which plays an essential role in perceiving and processing visual stimuli. The artificial neurons in a CNN are arranged to efficiently interpret visual information, enabling these models to process entire images. Because CNNs are so effective at identifying objects, they are frequently used for computer vision tasks such as image recognition and object detection, with common use cases including self-driving cars, facial recognition and medical image analysis.

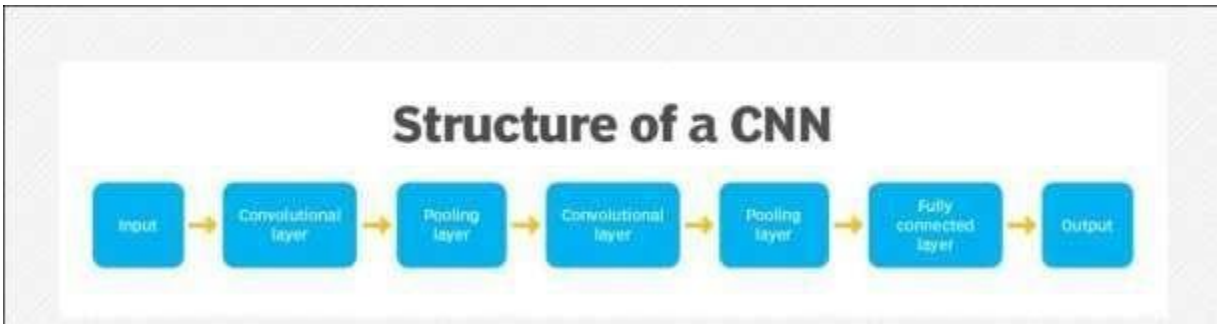


Fig 1.2 Structure of CNN

CNNs use a series of layers, each of which detects different features of an input image. Depending on the complexity of its intended purpose, a CNN can contain dozens, hundreds or even thousands of layers, each building on the outputs of previous layers to recognize detailed patterns.

The process starts by sliding a filter designed to detect certain features over the input image, a process known as the convolution operation (hence the name "convolutional neural network"). The result of this process is a feature map that highlights the presence of the detected features

in the image. This feature map then serves as input for the next layer, enabling a CNN to gradually build a hierarchical representation of the image.

Initial filters usually detect basic features, such as lines or simple textures. Subsequent layers' filters are more complex, combining the basic features identified earlier on to recognize more complex patterns. For example, after an initial layer detects the presence of edges, a deeper layer could use that information to start identifying shapes.

LSTM:

Long Short-Term Memory (LSTM) networks have emerged as a pivotal component in the realm of action recognition, and their adoption has transformed the field. These specialized recurrent neural networks (RNNs) were introduced to address the limitations of traditional RNNs when dealing with sequential data, making them ideal for modeling the temporal dynamics inherent in videos. In action recognition, the context of motion and change over time plays a critical role. Unlike static image classification tasks, where CNNs often suffice, recognizing actions demands a deep understanding of how visual patterns evolve within video sequences.

LSTMs excel at capturing the temporal dependencies in data by virtue of their unique memory cells. Each cell can store information over extended periods, ensuring that past observations significantly influence predictions, and potentially even distant events can impact the recognition process. This temporal modeling ability aligns perfectly with the challenges posed by action recognition, where the nuances of an action might evolve gradually, spanning several frames. For instance, when differentiating between “running” and “jumping,” the sequence of leg movements and body postures over time is crucial, and LSTMs are adept at capturing these subtleties.

Moreover, LSTMs offer flexibility in handling sequences of varying lengths, a common occurrence in action recognition datasets. Actions can unfold at different speeds, and LSTM networks can naturally adapt to these dynamics. This adaptability, coupled with their robustness to handle long-range dependencies, has led to their widespread adoption in action recognition research. The output from an LSTM layer can encapsulate an abstract representation of the entire video sequence, which can then be used for making accurate action predictions. In essence,

LSTMs serve as the temporal memory of the model, bridging the gap between the spatial features extracted by convolutional neural networks (CNNs) and the final action recognition decision.

LSTMs are a vital cog in the action recognition machinery due to their ability to model sequential data effectively. They bring the element of time into the recognition process, making them indispensable for tasks where temporal dynamics matter. As we continue to explore the nuances of human actions through video data, LSTMs will likely remain a cornerstone of this exciting field, helping us unlock new possibilities in applications like surveillance, human-computer interaction, and beyond.

Applications of LRCN Model:

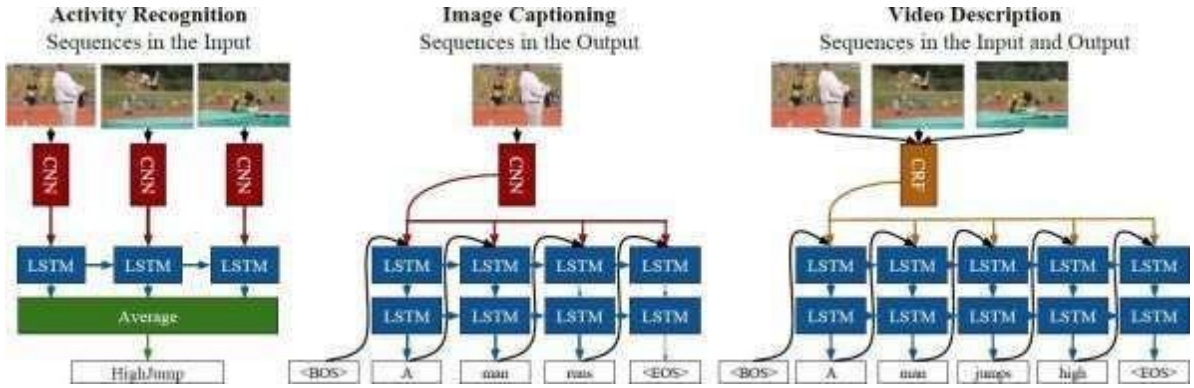


Fig 1.3 Applications of LRCN model

1.10 ANNOY LIBRARY:

Annoy (Approximate Nearest Neighbours Something Something) is a C++ library with Python bindings to search for points in space that are close to a given query point. It also creates large read-only file-based data structures that are mmapped into memory so that many processes may share the same data.

There's a couple of other libraries to do approximate nearest neighbour search, including FLANN, etc. Other libraries may be both faster and more accurate, but there are one major difference that sets Annoy apart: it has the ability to use static files as indexes. In particular, this means you can share index across processes. Annoy also decouples creating indexes from loading them, so you can pass around indexes as files and map them into memory quickly. Another nice thing of Annoy is that it tries to minimize memory footprint so the indexes are quite small.

Why is this useful? If you want to find nearest neighbours and you have many CPU's, you only need the RAM to fit the index once. You can also pass around and distribute static files to use in production environment, in Hadoop jobs, etc. Any process will be able to load (mmap) the index into memory and will be able to do lookups immediately.

We use it at Spotify for music recommendations. After running matrix factorization algorithms, every user/item can be represented as a vector in f-dimensional space. This library helps us search for similar users/items. We have many millions of tracks in a high-dimensional space, so memory usage is a prime concern.

Summary of features

- Euclidean distance (squared) or cosine similarity (using the squared distance of the normalized vectors)
- Works better if you don't have too many dimensions (like <100)
- Small memory usage
- Lets you share memory between multiple processes
- Index creation is separate from lookup (in particular you can not add more items once the tree has been created)
- Native Python support

Right now it only accepts integers as identifiers for items. Note that it will allocate memory for max(id)+1 items because it generally assumes you will have items 0 … n.

How does it work

Using random projections and by building up a tree. At every intermediate node in the tree, a random hyperplane is chosen, which divides the space into two subspaces.

We do this k times so that we get a forest of trees. k has to be tuned to your need, by looking at what tradeoff you have between precision and performance. In practice k should probably be on the order of dimensionality.

1.11 PROPOSED SYSTEM:

The main objective is to categorize videos, by extracting video frames and generating feature vectors to identify action sequences. For improved spatial and temporal properties, a suggested deep neural network integrates Long Short Term Memory and Convolutional Neural Network models.

By removing characteristics from frames and taking into account variables like frame width, height, and video sequence duration, video categorization is accomplished.Beginning with a time-distributed 2D convolutional layer, Following the initial convolutional layer, time-distributed max-pooling layers and dropout layers are strategically inserted.

Additional convolutional layers with increasing filter sizes further enhance the model's capacity for feature extraction. A flatten layer prepares the output for temporal analysis by a

Long Short-Term Memory (LSTM) layer, which captures intricate temporal dependencies within the video sequences.

The architecture culminates in a dense layer equipped with a softmax activation function, generating predictions for various classes. The ISRO Dataset is used for video classification. The results show that in terms of prediction accuracy, the LRCN methodology performs better than both ConvLSTM and conventional CNN methods.

Furthermore, the proposed method offers improved temporal and spatial stream identification accuracy. The findings show that a 67% prediction of the action sequence's probability is made.A long with Classification we also Recommend few videos based upon the input videos or images.

CHAPTER 2
REVIEW OF LIERATURE

1. Several studies have been conducted on video segmentation using visual features. Andrej Karpathy and others. [1] used a new dataset of 1 million YouTube videos in 487 categories to conduct a comprehensive empirical study of convolutional neural networks (CNNs) for large video classifications The authors investigated different approaches enhanced the ability of CNN to capture local spatial and temporal information in time domain and f proposed a multiresolution architecture as a possible way to speed up training. The performance of the proposed spatio-temporal network shows a significant improvement from 55.3% to 63.9% compared to the robust feature- based baseline models but from 59.3% to 60.9% better performance compared to the single-frame model of the Improvement. Transfer learning was also applied to the UCF101 Action Recognition dataset.

2. The CNN (RCNN) iterative model was proposed by Ming Liang et al. al [2] for object detection by adding recursive combinations to each convertible layer. Even though the input parameters are fixed, the behaviour of RCNN clusters changes over time, so that the behaviour of neighbouring clusters influences the behaviour of each cluster This characteristic provides the ability of the model to provide information about context will be included, which is important for improving the discovery process. CIFAR-10, CIFAR-100, MNIST, and SVHN object recognition data sets were used to evaluate the model. On every dataset that is taken into consideration, it is found that RCNN performs better than cutting-edge models with less trainable parameters. These results demonstrate the benefits of conventional object recognition programs over sophisticated programs exclusively.

3. For gesture recognition in videos, Pigou, L. Et al. [3] presented a novel give-up-to-quit trainable neural network architecture that combines bidirectional recurrence and temporal convolutions with deep systems. The significance of repetition and its necessity, which includes temporal convolutions that result in significant profits, have been examined by the writers. We tested several approaches with the newly discovered outcomes from the Montalbano gesture reputation dataset.

4. Caleb Andrew et al. conducted surveys on the algorithms used in video classification techniques[4] to see whether technique is more effective at predicting gestures or motions and

can be applied to the categorization of action films. In deep learning models for visual sequence modeling, LSTM is crucial for the recognition of gestures and actions. Even though there are numerous movie categories, emerging methods frequently combine well-known categories to increase classification accuracy.

5. A revolutionary neural network incorporating advanced convolutional layers and the remarkable long short-term memory (LSTM) was brought forth by Xia, K., et al. [5]. In order to gauge the true potential of this model, three publicly accessible datasets, namely UCI, WISDM, and OPPORTUNITY, were employed. Astonishingly, the model boasted an accuracy 95.78 percent in the dataset from UCI-HAR, 95.85 percent in the dataset from WISDM,, and a notable 92.63 percent in the OPPORTUNITY dataset.

CHAPTER 3

PROPOSED SOLUTION

3.1 OVERVIEW:

The proposed solution comprises three main phases. Firstly, it involves training a deep learning model, which likely entails data collection, preprocessing, model selection, and training. Secondly, there's the creation of a front-end web application, which would be the user interface for interacting with the trained model. This front-end could include features like uploading images or videos, displaying classification results, and showing recommended content. Lastly, the backend integration would involve connecting the trained model with the front-end through an API. This integration would enable the front-end to send requests to the model, receive predictions, and display them to the user.

3.2 DATASET DESCRIPTION:

The ISRO Video Dataset had been given by the isro organization as a part of the Smart india Hackathon 2023. It contains videos divided into five different categories: Animation, PersonCloseUp, Graphics, OutdoorLaunchPad, and IndoorControlRoom and the dataset contains 150 videos, with 30 videos in each category. The average length of the videos in the data set is about 5 seconds.

Total Categories	5
Total Videos	150
Videos per Category	30
Minimum Duration	4 sec
Maximum Duration	23 sec

Table 3.1: Dataset Description

Fig 3.1: ISRO videos 5 videos for each Category

The animation part consists of animated content, with visual effects and conceptual elements. The PersonCloseUp range focuses primarily on individuals in close-up shots. Graphics include videos that incorporate computer-generated images or graphics, which provide complex data or mental representations. The OutdoorLaunchPad section contains outdoor launches, spacecraft, rockets, or activities related to space exploration, providing insight into the fun side of space exploration Finally, the IndoorControlRoom section contains control room videos, monitoring stations, or similar furniture with Shan and the behind- the-scenes side of his scientific endeavours.

3.3 PROCESS:

The flow chart representation is as follows:

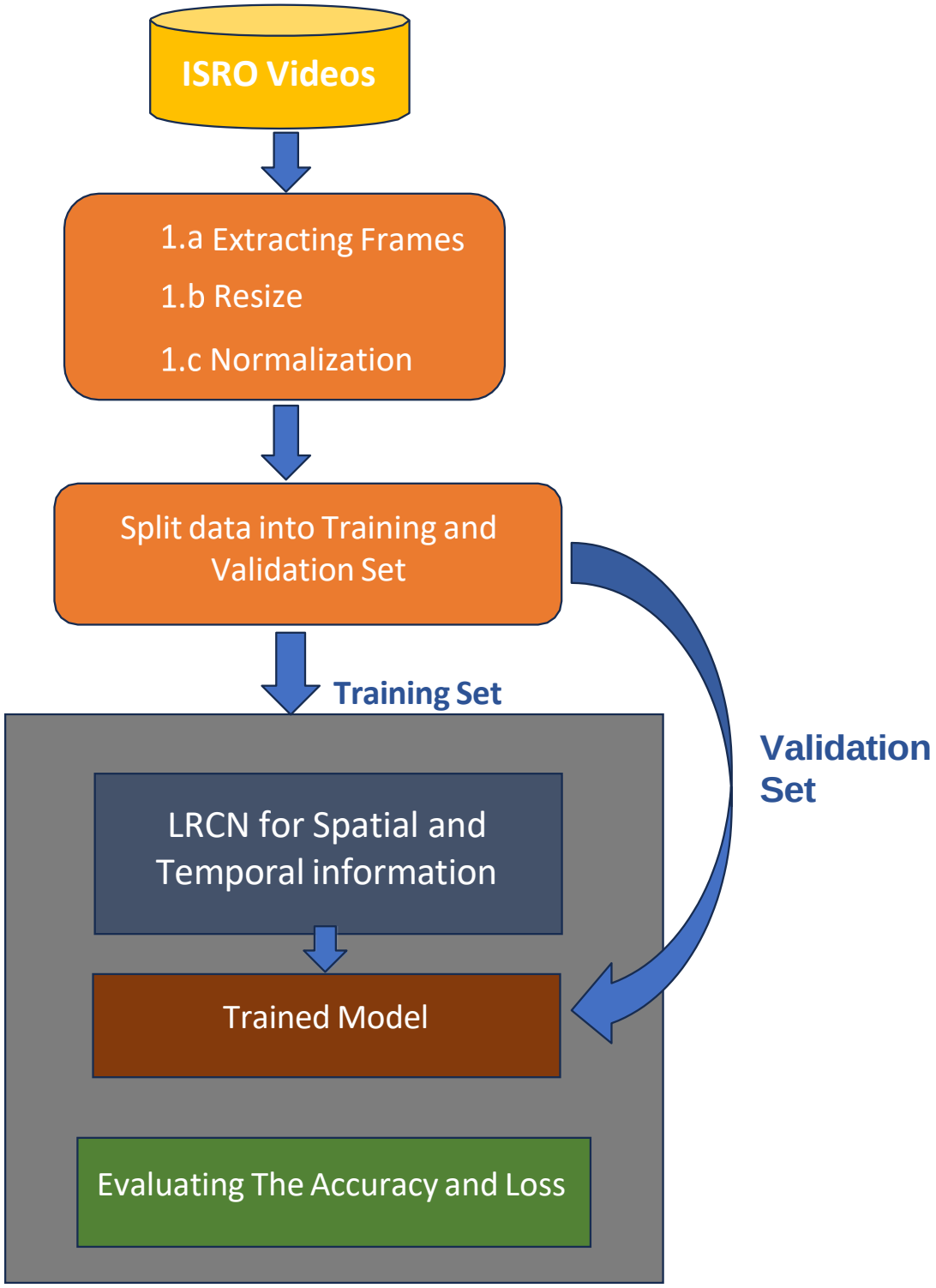


Fig:3.2 Flowchart of Proposed System

Step-1: Loading Dataset:

To load the ISRO video dataset in Google Colab, you first need to mount your Google Drive and then navigate to the dataset directory. Here's how you can do it:

```
from google.colab import drive

drive.mount('/content/drive')

# Navigate to the dataset directory

cd /content/drive/MyDrive/ISRO-Videoclassification
```

To create dataset we create a function named with create_dataset

```
def create_dataset():

  # Declared empty lists to store the features, labels, and video file path values

  features = []

  labels = []

  video_files_path = []

  # Iterating through all the classes mentioned in the classes list

  for class_index ,class_name in enumerate(CLASSES_LIST):

  # display the name of the class

    print(f"Extracting Data of class: {class_name}")

    files_list = os.listdir(os.path.join(DATASET_DIR,class_name))

    for file_name in files_list:

      # get the complete video path

      video_file_path = os.path.join(DATASET_DIR, class_name , file_name)

      # Extract the frames of the video file

      frames = frames_extraction(video_file_path)
```

```
        # Check if the extracted frames are equal to the SEQUENCE_LENGTH specified above
        # So, ignore the video having frames less than the SEQUENCE_LENGTH
        if len(frames) == SEQUENCE_LENGTH:
            # Append the data to their respective lists
            features.append(frames)
            labels.append(class_index)
            video_files_path.append(video_file_path)
    # Convertion the list to numpy arrays
    features = np.asarray(features)
    labels = np.array(labels)
    # Return the frames ,class index , video file path
    return features ,labels , video_files_path
```

The create_dataset() function iterates through each class in a predefined list of classes (CLASSES_LIST) and extracts frames from videos belonging to each class. For each video, it checks if the number of extracted frames matches a specified sequence length (SEQUENCE_LENGTH). If so, it appends the frames, the class index, and the video file path to their respective lists (features, labels, video_files_path). Finally, it converts the lists to NumPy arrays and returns them as the dataset's features, labels, and video file paths.

Step-2: Pre-processing:

Preprocessing a video dataset typically involves several steps to prepare the data for training a model.

1. Frame extraction
2. Frame Resizing
3. Normalization

For above three steps we create a function named as feature_extraction.

```
def frames_extraction(video_path):
  # Declare the list to store video frames
  frames_list = []
  # Read the video file using the VideoCapture object
  video_reader = cv2.VideoCapture(video_path)
  # Get the total no.of frames in the video
  video_frames_count =
  int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT)) # Calculate the
  interval after which frames will be added to the list skip_frames_window =
  max(int(video_frames_count/SEQUENCE_LENGTH),1)
  #Ex: no.of.Frames = 7,sequence= 3 ==> 7/3 = 2.5 ~= 2sec( select each frame after 2
seconds)
  # Iterate through the video frames
  for frame_counter in
    range(SEQUENCE_LENGTH):# Set the current
    frame position of the video
    video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter *
skip_frames_window)
    # reading the frame from video
    success , frame = video_reader.read()
    if not success:
      break
    # Resize the frame to fixed height and width
    resized_frame = cv2.resize(frame, (IMAGE_HEIGHT,IMAGE_WIDTH))
```

```
    # Nomalize the resized frame
    normalized_frame = resized_frame/255
    # Append the normalized frame into the frames list
frames_list.append(normalized_frame) video_reader.release()
return frames_list
```

The frames_extraction() function takes a video file path as input and extracts frames from the video. It reads the video using OpenCV's VideoCapture object, calculates the interval for frame extraction based on the total number of frames and a predefined sequence length. For each frame, it resizes it to a fixed height and width, normalizes the frame by scaling pixel values between 0 and 1, and appends the normalized frame to a list. The function returns a list of resized and normalized frames from the video.

Step-3: creating model:

For the model creation, we create function called with create_LRCN_model()

```
def create_LRCN_model():
    # We will use a Sequential model for model construction.
    model = Sequential()
    # Define the Model Architecture.
    model.add(TimeDistributed(Conv2D(16, (3, 3), padding='same',activation ='relu'),
       input_shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3)))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Conv2D(32, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.2)))
```

```
model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2))))
model.add(TimeDistributed(Dropout(0.2))) model.add(TimeDistributed(Conv2D(64,
(3, 3), padding='same',activation = 'relu')))
model.add(TimeDistributed(MaxPooling2D((2,
2))))model.add(TimeDistributed(Flatten()))
model.add(LSTM(32))
model.add(Dense(len(CLASSES_LIST), activation =
'softmax'))# Return the constructed LRCN model.
return model
```

The create_LRCN_model() function builds a Long-term Recurrent Convolutional Network (LRCN) model using Keras' Sequential API. The model consists of a sequence of layers, starting with a TimeDistributed layer to apply Conv2D and MaxPooling2D operations to each frame in the input sequence of frames. This is followed by additional TimeDistributed Conv2D and MaxPooling2D layers, each with a Dropout layer for regularization. The output of these layers is flattened and fed into an LSTM layer for capturing temporal dependencies across frames. Finally, a Dense layer with a softmax activation function is used for classification into the classes specified in CLASSES_LIST. The function returns the constructed LRCN model.

Step-4:Compiling and Training the model:

To compile our model we should some prequests such as optimizer, learning rate, Loss function

```
optimizer_type = "Adam"
optimizer_lr = 0.001
loss = "categorical_crossentropy"
```

```
metrics = ["accuracy"]

optimizer = Adam(learning_rate=optimizer_lr)

LRCN_model.compile(loss=loss, optimizer=optimizer, metrics=metrics)



# Start training the model with data generators and regularizationbatch_size = 4

epochs = 40

er_stop_patience = 20

validation_split_ratio = 0.2

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

reduce_lr = ReduceLROnPlateau(factor=0.1, patience=5)#

Start training the model

LRCN_model_training_history = LRCN_model.fit(features_train, labels_train,
epochs=epochs,shuffle=True, callbacks=[early_stopping, reduce_lr],

validation_data=(features_test, labels_test))
```

The code snippet compiles and trains the previously constructed LRCN model (create_LRCN_model). It sets the optimizer type to Adam with a learning rate of 0.001 and specifies categorical crossentropy as the loss function and accuracy as the metric for evaluation. The model is compiled using these settings. Training is then initiated with a batch size of 4, over 40 epochs, and with early stopping and learning rate reduction callbacks configured. The training data is split into training and validation sets with a ratio of 0.2 for validation. The training history is stored in LRCN_model_training_history, which contains information about the model's performance metrics over the training epochs.

Step-4: Model Evaluation:

For the evaluation of our trained model keras.model provide a method called evaluate() which takes testing data and their labels as parameters for getting testing accuracy.

```
# Evaluate the model on the test set
test_loss, test_accuracy = LRCN_model.evaluate(features_test, labels_test)
# Evaluate the model on the train set
train_loss , train_accuracy = LRCN_model.evaluate(features_train,labels_train)
print(f'Test accuracy: {test_accuracy}')
print(f'Train accuracy: {train_accuracy}')
```

3.4 CONCLUSION:

With the results of the above test_accuracy and train_accuracy we will state that how well our model has been trained on the training dataset and how well it is performing on the unseen dataset.

CHAPTER 4
UML DIAGRAMS

4.1 UseCase Diagram:

Represents the interactions between users (actors) and the system, showing the various use cases and how they are related.

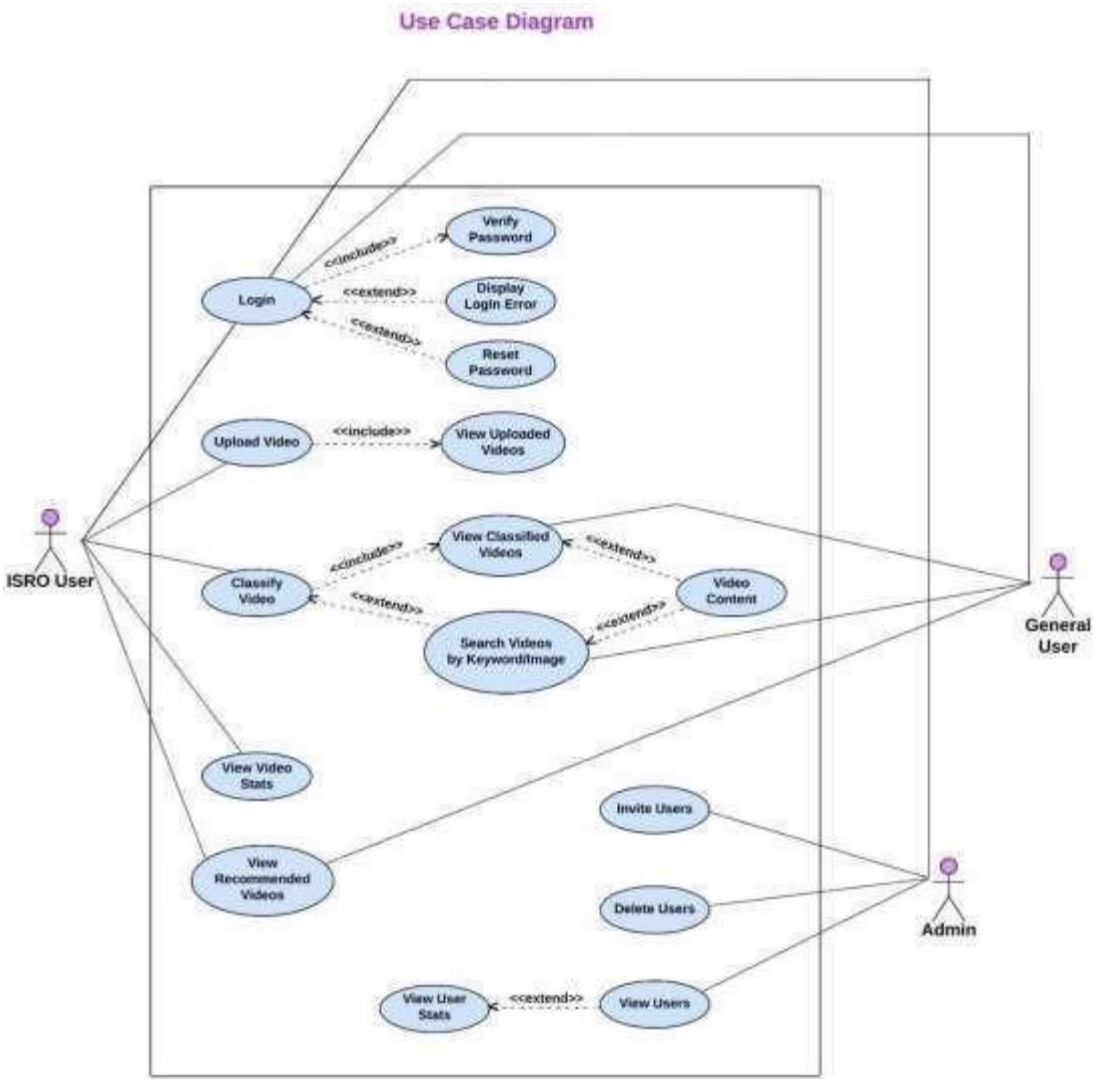


Fig 4.1 Usecase Diagram

4.2 Class Diagram:

Describes the structure of a system by showing the classes, their attributes, methods, and relationships.

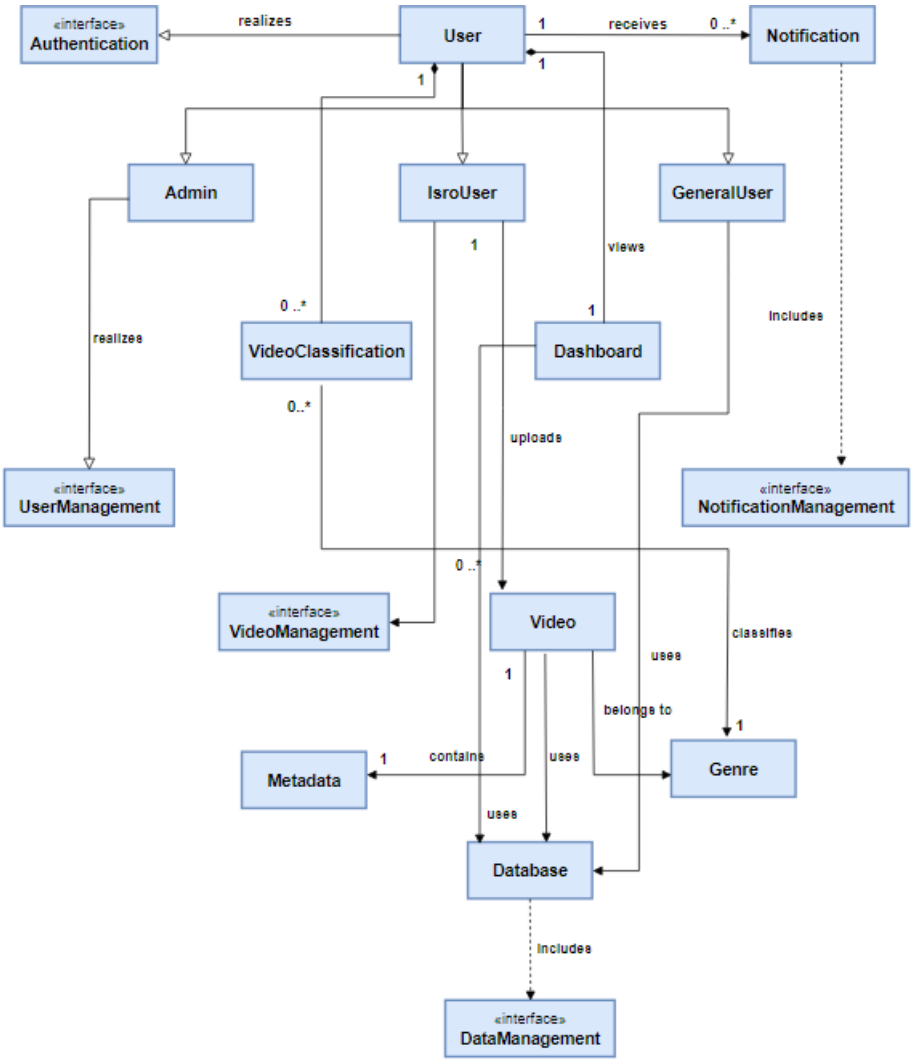


Fig 4.2 Class Diagram

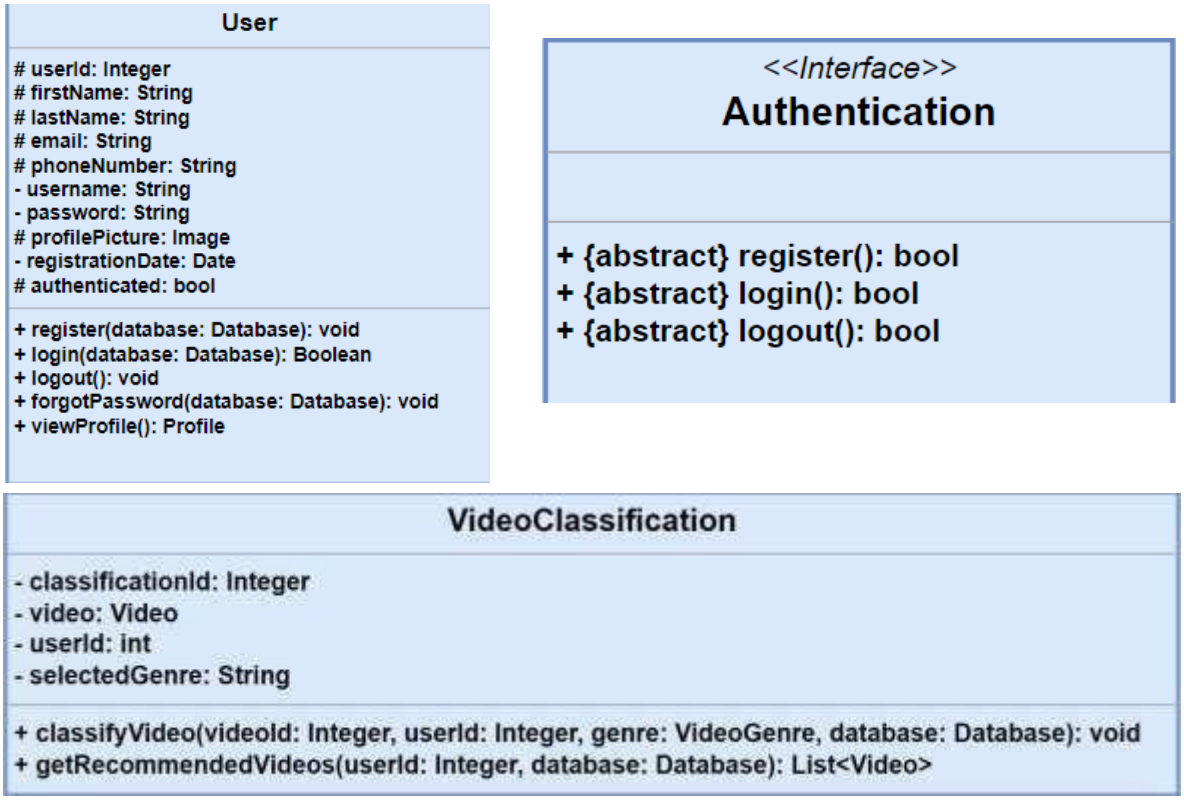


Fig 4.3 Individual Class Representation

4.3 Object Diagram :

Represents a snapshot of objects and their relationships at a particular time within a system.

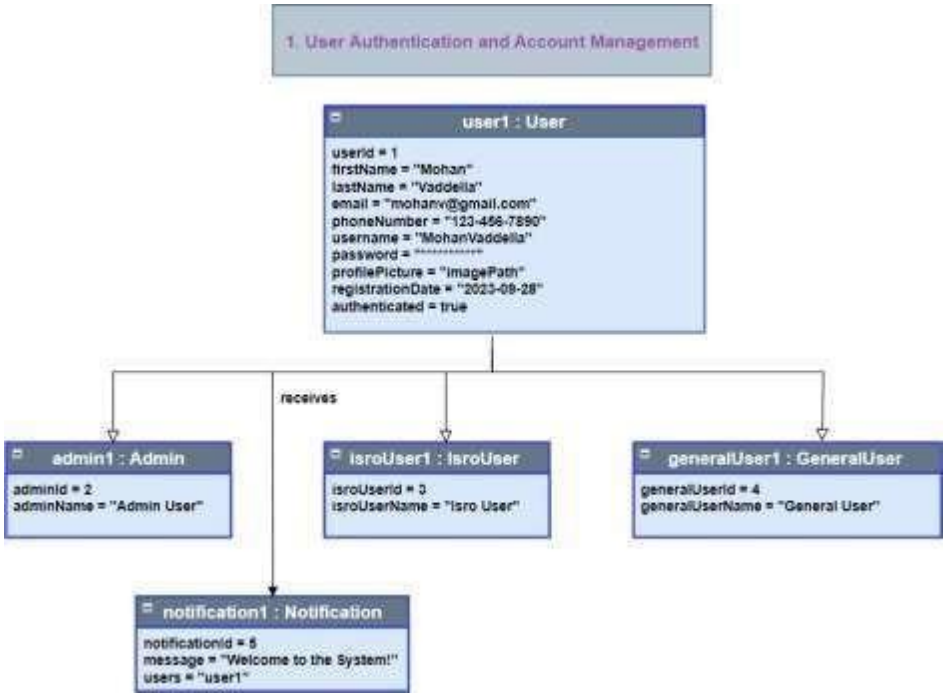


Fig 4.4 Object Diagram

4.4 Sequence Diagram:

Shows how objects interact in a particular scenario of a use case, focusing on the order of messages exchanged between objects.

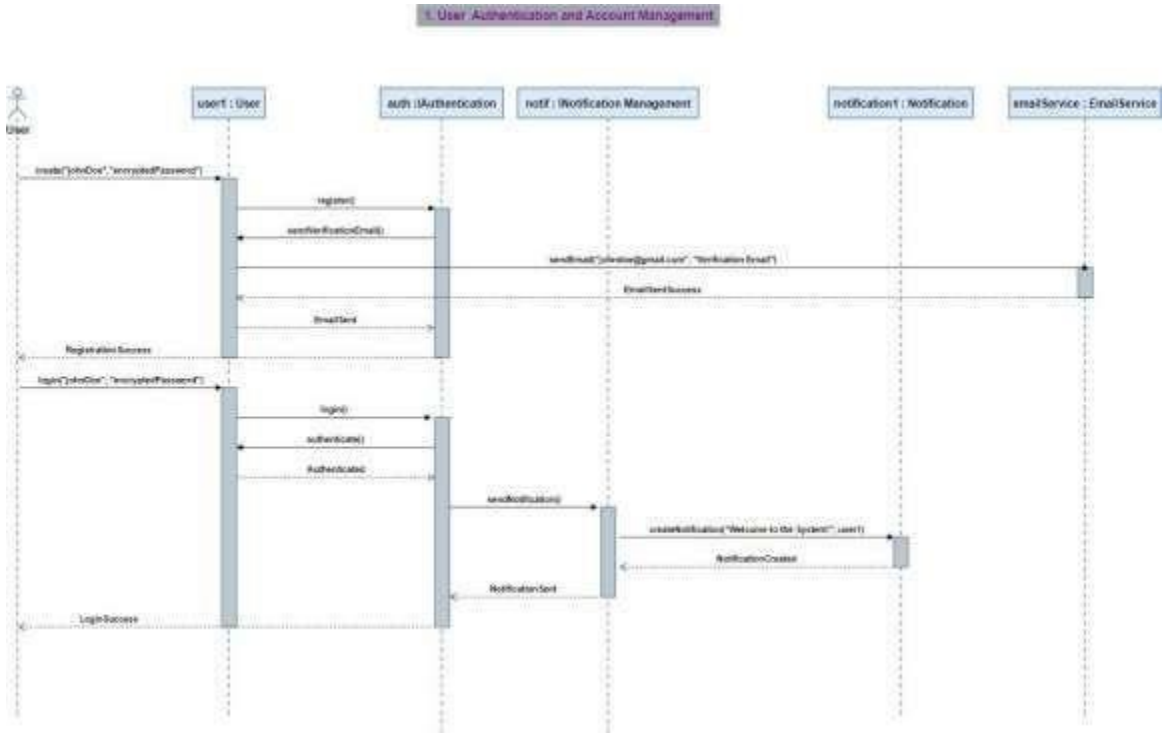


Fig 4.5 Sequence Diagram

4.5 Statechart Diagram:

Represents the states of an object and the transitions between these states in response to events.

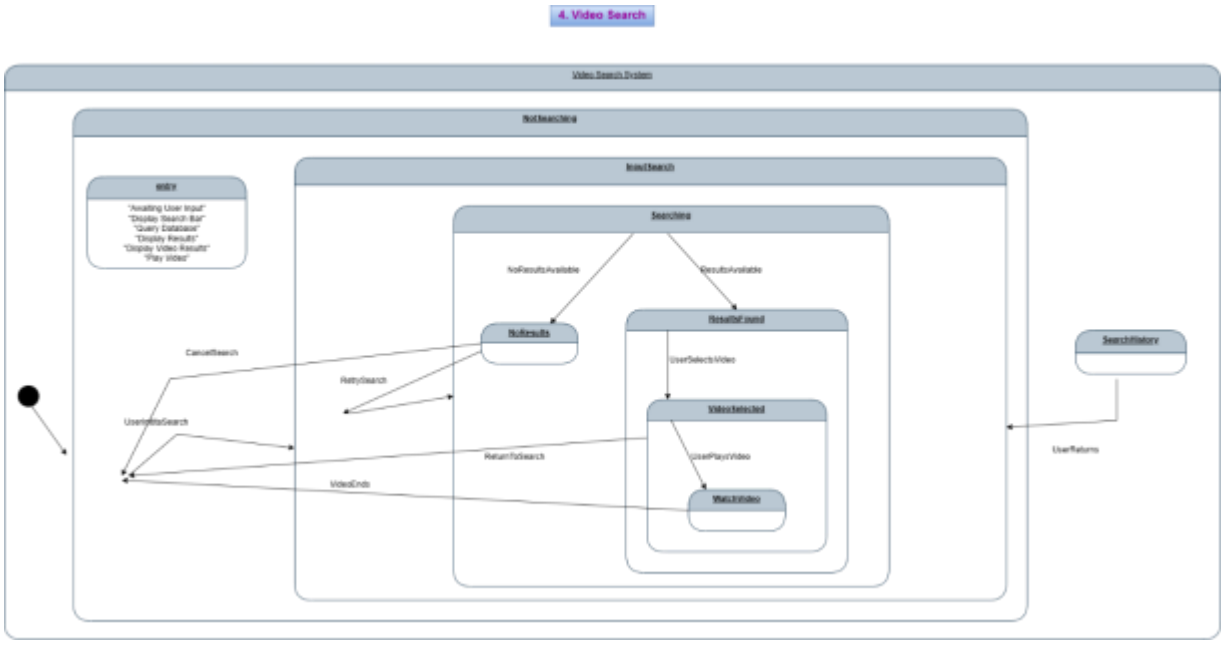


Fig 4.6 Statechart Diagram

4.6 Activity Diagram:

Represents the flow of control or the flow of objects in a system, typically used to model workflows or business processes.

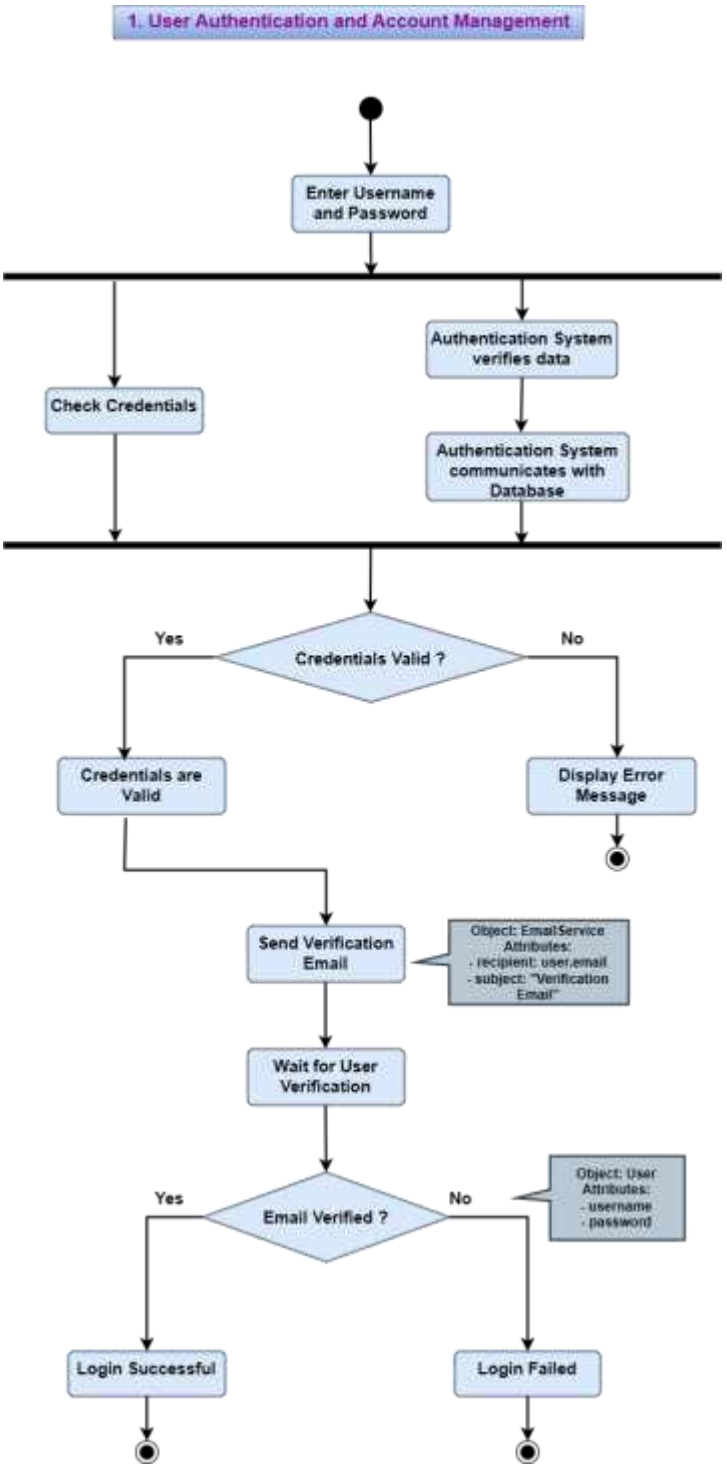


Fig 4.7 Activity Diagram

4.7 Collaboration Diagram:

Similar to sequence diagrams, but focuses more on the relationships among objects and how they collaborate to achieve a specific task.

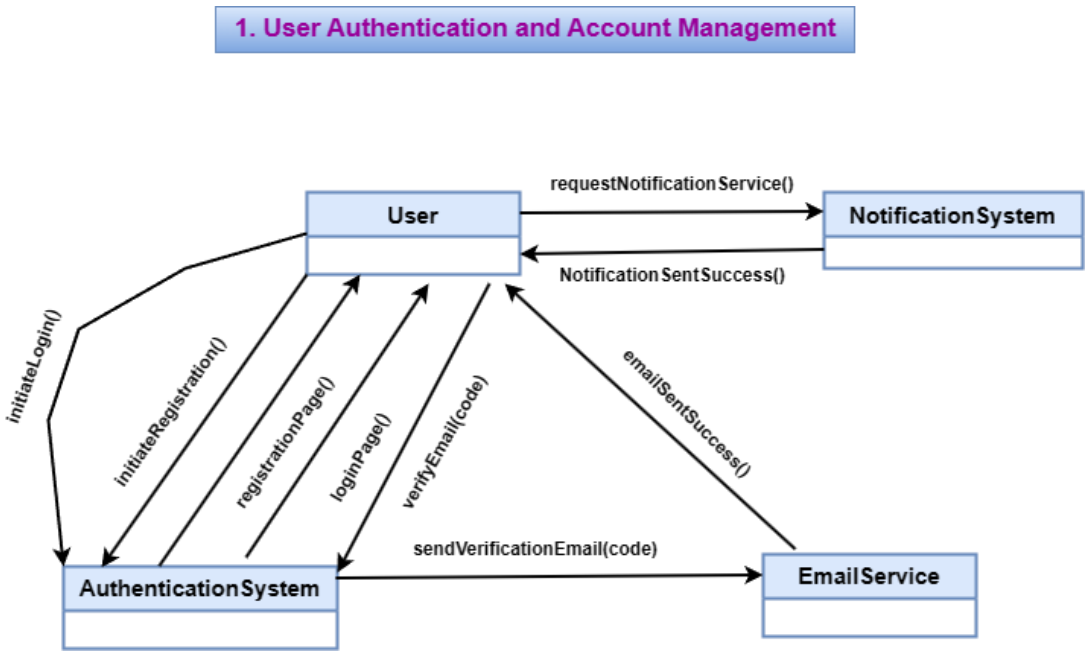


Fig 4.8 Collaboration Diagram

4.8 Component Diagram:

Shows the components of a system, their dependencies, and how they are wired together at a higher level of abstraction.

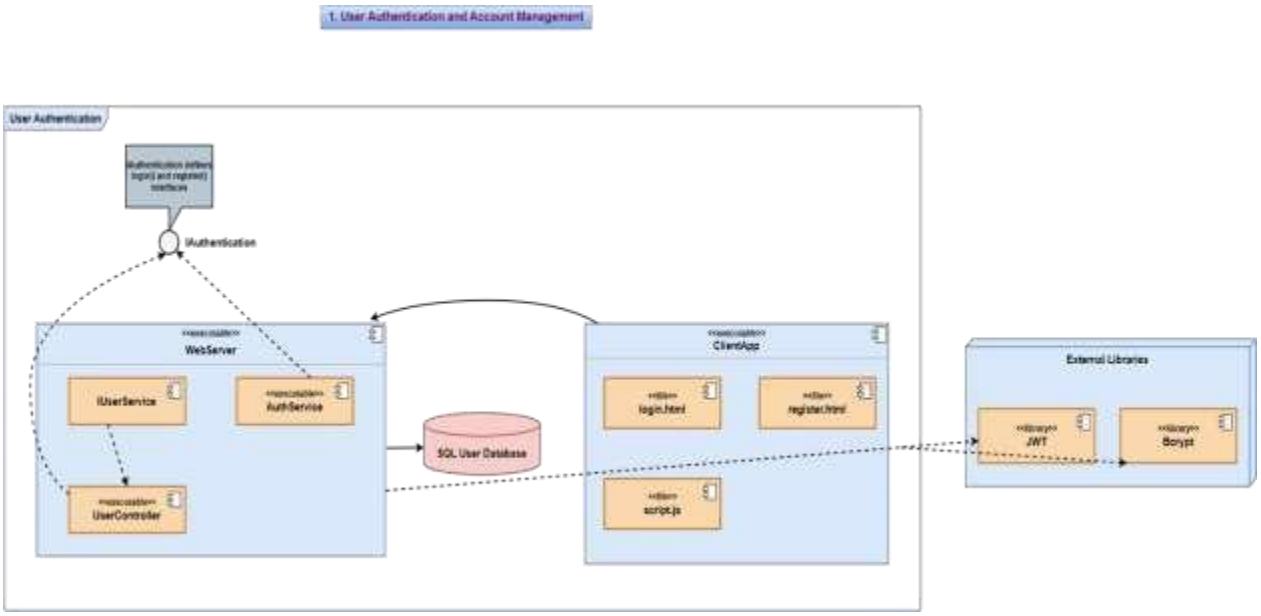


Fig 4.9 Component Diagram

4.9 Deployment Diagram:

Describes how software components are deployed on hardware components, showing the physical configuration of a system.

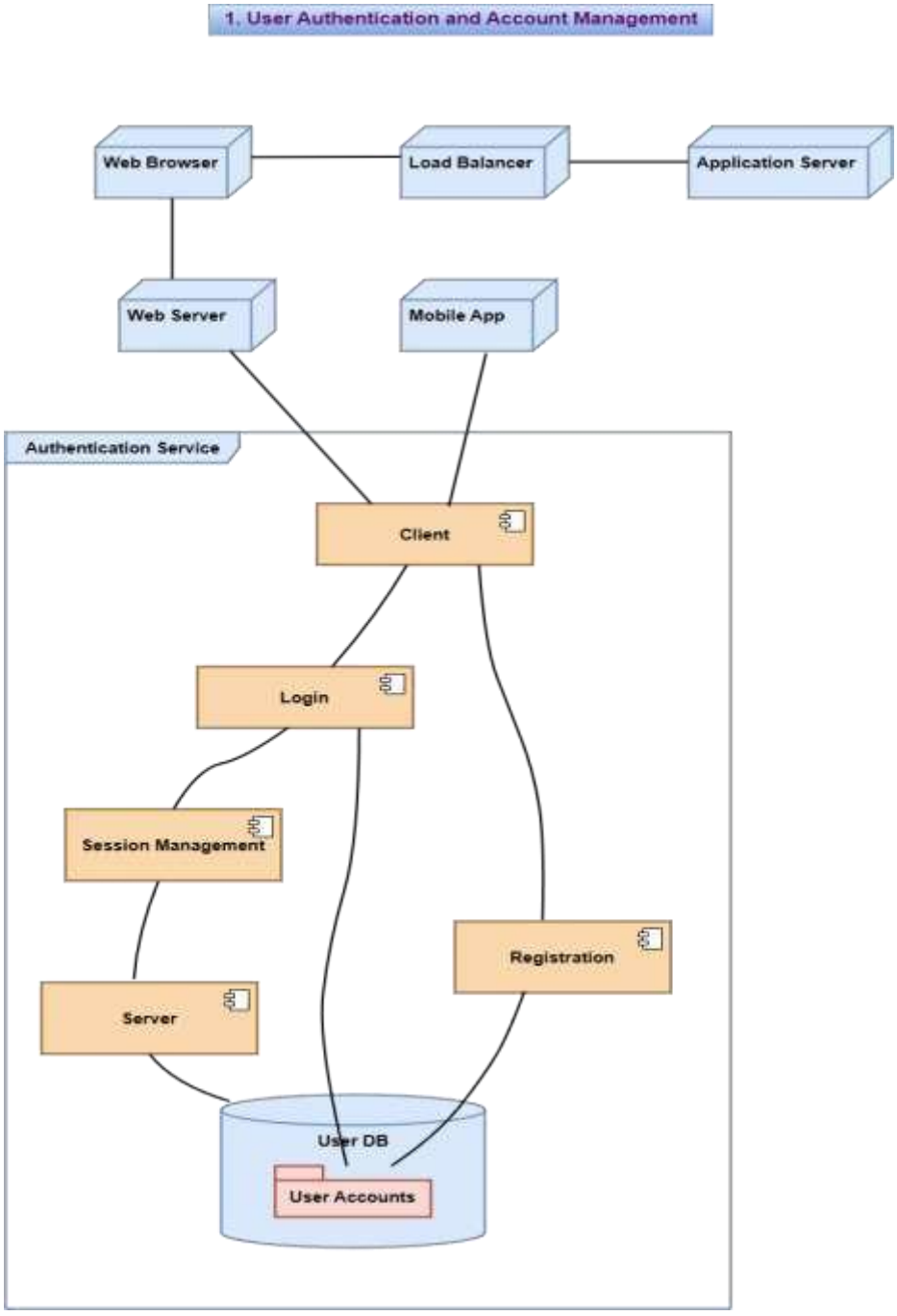


Fig 4.10 Deployment Diagram

CHAPTER 5

IMPLEMENTATION

5.1 DEEP LEARNING IMPLEMENTATION:

We will see here what are the libraries had been used for training our deep learning model.

5.1.1 ANACONDA:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager, called **Anaconda Navigator**, so it eliminates the need to learn to install each library independently.

The open source packages can be individually installed from the Anaconda repository with the **conda install** command or using the **pip install** command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together.

Custom packages can be made using the **conda build** command and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, mac OS and Linux.

5.1.2 Google Colab*:*

Google Colab is a cloud-based Jupyter notebook environment provided by Google that allows you to write, execute, and share Python code directly in your browser. It offers several features that make it popular among data scientists and machine learning practitioners:

1. **Free Access to GPU and TPU:** Google Colab provides free access to powerful GPUs and TPUs, which are essential for accelerating the training of deep learning models. This is particularly useful for tasks that require heavy computational resources, such as image classification, natural language processing, and reinforcement learning.
2. **Easy Collaboration:** Google Colab allows you to share your notebooks with others, making it easy to collaborate on projects. You can grant different levels of access to your collaborators, such as view-only, comment-only, or edit access. This feature is similar to Google Docs, making it familiar and intuitive for users.
3. **Integrated with Google Drive:** Colab is tightly integrated with Google Drive, allowing you to easily access and save your notebooks and data files. This integration makes it convenient to work with files stored in Google Drive, as you can directly access them from your Colab notebook without the need for manual uploads or downloads.
4. **Pre-installed Libraries:** Google Colab comes with many popular data science and machine learning libraries pre-installed, such as TensorFlow, PyTorch, and scikit-learn. This saves you the time and effort of installing these libraries manually, allowing you to focus on your analysis and model development.
5. **Interactive Visualizations:** Colab supports the use of libraries like matplotlib and seaborn for creating interactive visualizations directly in your notebook. This allows you to explore your data visually and gain insights quickly, enhancing your data analysis workflow.
6. **Support for Markdown**: Colab supports Markdown cells, which allow you to add formatted text, images, and links to your notebook. This is useful for documenting your code, explaining your thought process, and providing context to your analysis. Markdown cells make your notebooks more readable and understandable to others who may be reviewing or collaborating on your project.

Because of all the features of colab we had used this platform for training our deep learning model. The interface of google colab looks like below fig

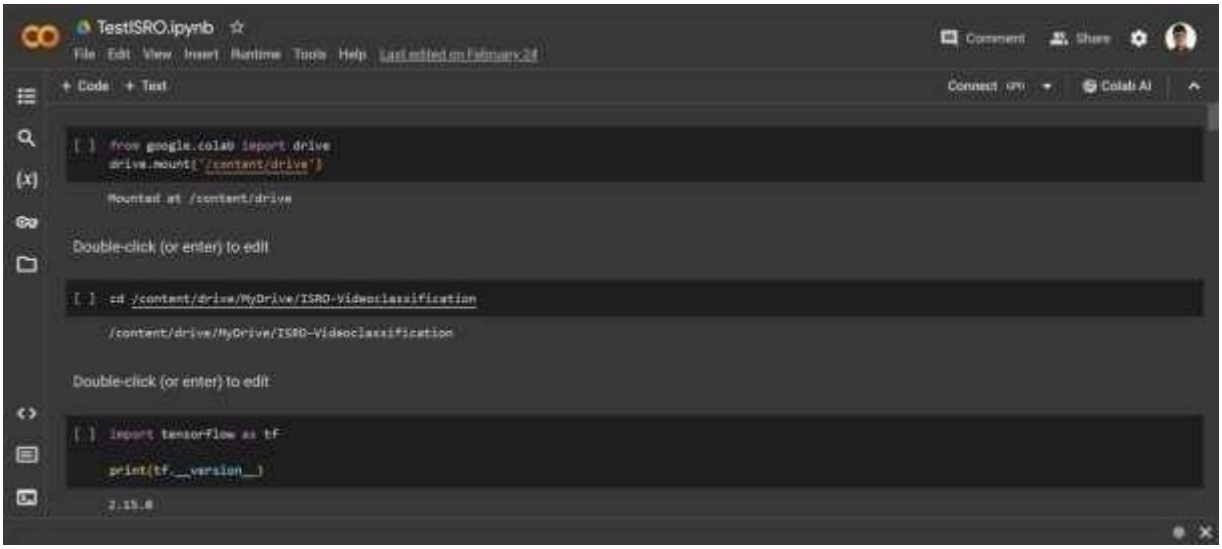


Fig 5.1 Google Colab Interface

Once if we got connect to the kernal then we can run all the python commands , the beauty of colab is we can mount our dataset from various online sources such as Google Drive or Kaggle This will save lot of internet and local memory storage when compared with the jupyter notebook. And the visualizations of the results and the errors rectification directly it recommends the solution available from the stackoverflow which helps the developers to save their time.

5.1.3 Tensorflow

TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand framework.

TensorFlow uses a dataflow graph to represent computational processes. This graph defines the operations and dependencies between them, allowing TensorFlow to efficiently execute computations on CPUs, GPUs, or other accelerators.

.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Important features of TensorFlow:

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

TensorFlow (TF) is a popular choice for deep learning models due to several reasons:

1. **Performance**: TensorFlow is optimized for performance, especially when using GPUs and TPUs. It can efficiently handle large-scale deep learning tasks, making it suitable for training complex models on large datasets.
2. **Flexibility**: TensorFlow provides a flexible architecture that allows you to build a wide range of machine learning models, from simple feedforward neural networks to complex deep learning architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs).
3. **Scalability**: TensorFlow is designed to scale to large datasets and distributed environments. It can be used to train models on clusters of GPUs or TPUs, allowing you to take advantage of parallel processing to speed up training.
4. **Ecosystem**: TensorFlow has a rich ecosystem of libraries, tools, and resources that complement the core framework. This includes high-level APIs like Keras, which make it easy to build and train models, as well as libraries for data preprocessing, model evaluation, and deployment.
5. **Community Support:** TensorFlow has a large and active community of developers and researchers who contribute to its development and provide support through forums,

tutorials, and other resources. This community-driven development model ensures that TensorFlow remains up-to-date with the latest developments in deep learning.

Installation of tensorflow:

To install TensorFlow in local computer for the better organization of our project we create a new virtual environment in anaconda and then install the TensorFlow in the created environment.

1. **Create a New Anaconda Environment:** Open Anaconda Navigator and create a new environment. You can do this by clicking on the "Create" button, giving your new environment a name, and selecting the Python version you want to use (e.g., Python 3.7). Or we can create it from anaconda shell

```
conda create -n <env-name>
```

2. **Activate the New Environment**: Once the new environment is created, activate it by clicking on the "Play" button next to its name in Anaconda Navigator, or by running the following command in your terminal or Anaconda Prompt:

```
conda activate <environment_name>
```

3. **Install TensorFlow:** With your new environment activated, you can install TensorFlow using conda or pip. For the CPU-only version of TensorFlow, use:

```
Conda install tensorflow
```

Alternatively, you can use pip to install TensorFlow:

```
pip install tensorflow
```

4. Verify Installation: To verify that TensorFlow has been installed correctly, you can try importing it in a Python script or Jupyter notebook:

```
Import tensorflow as tf
Print(tf.__version__)
```

If TensorFlow is installed correctly, this should print the version number of TensorFlow installed in your environment.

5.1.4 Numpy:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. Numpy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

Importance of Numpy in Deep learning:

1. **Efficient Storage:** Video datasets often contain large amounts of data, including multiple frames per second for each video. NumPy's ability to efficiently store and manipulate multi-dimensional arrays allows for effective handling of these large datasets.
2. **Preprocessing:** NumPy provides a wide range of functions for preprocessing video data, such as resizing frames, normalizing pixel values, and extracting features. These preprocessing steps are essential for preparing video data for deep learning models.
3. **Frame Extraction:** Video datasets are typically stored as video files, which need to be processed frame by frame. NumPy's array manipulation capabilities make it easy to extract and manipulate individual frames from video files.
4. **Feature Extraction:** NumPy provides functions for extracting features from video frames, such as histograms of pixel intensities, optical flow, and motion vectors. These features are crucial for training deep learning models to recognize patterns in video data.
5. **Integration with Deep Learning Frameworks:** Deep learning frameworks like TensorFlow and PyTorch are built on top of NumPy and accept input data in the form of NumPy arrays. This seamless integration makes NumPy an essential tool for working with video datasets in deep learning.
6. **Array Broadcasting:** NumPy's array broadcasting feature allows for efficient computation across arrays of different shapes and sizes. This is particularly useful when performing operations on video frames that may have varying dimensions.
7. **Data Augmentation:** NumPy provides functions for data augmentation, such as flipping, rotating, and cropping video frames. Data augmentation is essential for increasing the diversity of the training data and improving the generalization of deep learning models.

5.1.5 PANDAS:

Pandas is a powerful and popular Python library for data manipulation and analysis. It provides data structures like DataFrame and Series that are ideal for working with structured data, including video datasets in deep learning. Pandas is important for deep learning with video datasets for several reasons:

1. **Data Loading and Preparation**: Pandas can read data from various sources, such as CSV files, Excel files, and databases, making it easy to load video dataset metadata and annotations. It also provides tools for cleaning, preprocessing, and transforming data, which is essential for preparing video datasets for deep learning models.
2. **Data Exploration:** Pandas offers functionalities for exploring and understanding the structure of video datasets. You can quickly summarize and visualize the data to gain insights, such as the distribution of video durations, frame rates, or labels.
3. **Data Integration:** Pandas integrates well with other Python libraries used in deep learning, such as NumPy and TensorFlow. It can convert Pandas DataFrames to NumPy arrays, which are compatible with deep learning frameworks, allowing for seamless integration of data preprocessing pipelines.
4. **Time Series Handling:** Video datasets often involve time series data, where each frame is a data point indexed by time. Pandas provides robust support for handling time series data, making it easy to manipulate and analyze video frames over time.
5. **Data Filtering and Selection:** Pandas provides powerful methods for filtering and selecting subsets of data, which is useful for selecting specific videos or frames based on criteria such as labels, timestamps, or other metadata.
6. **Data Aggregation and Grouping:** Pandas allows for data aggregation and grouping operations, which are helpful for summarizing video datasets based on different attributes, such as grouping videos by labels or time intervals.

5.1.6 OPEN CV:

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java,

etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV. This OpenCV tutorial will help you learn the Image-processing from Basics to Advance, like operations on Images, Videos using a huge set of Opencv-programs and projects.

We had used this opencv module for the preprocessing of videos and to load the videos

Video Loading and Preprocessing: OpenCV provides functions to load and preprocess videos, which is crucial for preparing video data for training deep learning models. It can read video files or capture video streams from cameras, making it versatile for various applications.

Frame Extraction: Videos are essentially a sequence of frames. OpenCV allows you to extract frames from videos, which is essential for tasks like object detection, action recognition, or any task where analyzing individual frames is necessary.

Data Augmentation: Data augmentation is critical for training deep learning models effectively. OpenCV provides various image manipulation functions that can be applied to frames extracted from videos to create augmented datasets for training.

```
import cv2

import os

# Path to the video file

video_path = "path_to_your_video_file.mp4"

# Directory to save the extracted frames

output_dir = "path_to_output_directory"

os.makedirs(output_dir, exist_ok=True)

# Open the video file

cap = cv2.VideoCapture(video_path)

# Initialize frame counter
```

```
frame_count = 0

# Loop through the video frames

while True:

    # Read a frame from the video

    ret, frame = cap.read()

    # Break the loop if no frame is returned

    if not ret:

        break

    # Preprocess the frame (e.g., resize, normalize, etc.)

    # Example: frame = cv2.resize(frame, (224, 224))

    # Save the frame as an image file

    frame_path = os.path.join(output_dir, f"frame_{frame_count}.jpg")

    cv2.imwrite(frame_path, frame)

    # Increment frame counter

    frame_count += 1

    # Display the frame (optional)

    # cv2.imshow("Frame", frame)

# Release the video capture object and close the window (if any)

cap.release()

cv2.destroyAllWindows()
```

Above code snippet opens a video file, extracts frames, preprocesses them (which can include resizing, normalization, etc., as needed for your application), and saves them as individual image files.

5.2 FRONTEND IMPLEMENTATION:

To enhance user interaction with our deep learning model, we've developed a frontend web application. Users can easily create individual accounts, sign in to their personal spaces, and

seamlessly interact with our deep learning model by uploading images and videos. The application displays the model's results and recommends videos based on the classification. Users can watch the recommended videos, play them, and share them with colleagues, facilitating collaborative engagement with the content.

Key functionalities of our website include:

1. User signup and signin
2. Profile update
3. Forgot password
4. OTP verification
5. Reset password
6. Video streaming
7. Video uploading
8. Video sharing
9. Analytics
10. Obtaining classified results
11. Displaying recommended videos

To accomplish above functionalities we used ReactJS framework for frontend development and styling purpose we had used the tailwindCSS.

5.2.1 REACTJS:

React is a JavaScript library for building user interfaces.React is used to build single-page applications.React allows us to create reusable UI components.

In React, JSX (JavaScript XML) is a syntax extension that allows you to write HTML-like code within JavaScript. JSX files in React contain components that define the UI elements of your application. Here's a basic example of a JSX file:

```
import React from 'react';
function App() {
  return (
    <div>
      <h1>Hello, world!</h1>
      <p>This is a JSX file in React.</p>
    </div>
```

```
  );
}
export default  App;
```

In this example, the and <h1> elements are written in JSX, which is then transpiled by Babel (a JavaScript compiler) into regular JavaScript code that React can understand. This makes it easier to write and understand complex UI components in React.

The below are jsx files to accomplish the above functional requirements

1. **Analytics.jsx:**

   This component displays the number of videos uploaded by each user, calculated by counting the videos in their respective AWS S3 bucket folders. For every user, a unique folder is created in the S3 bucket, and the count of videos in that folder is shown on the analytics page. This provides insights into user activity and content contribution within the application.

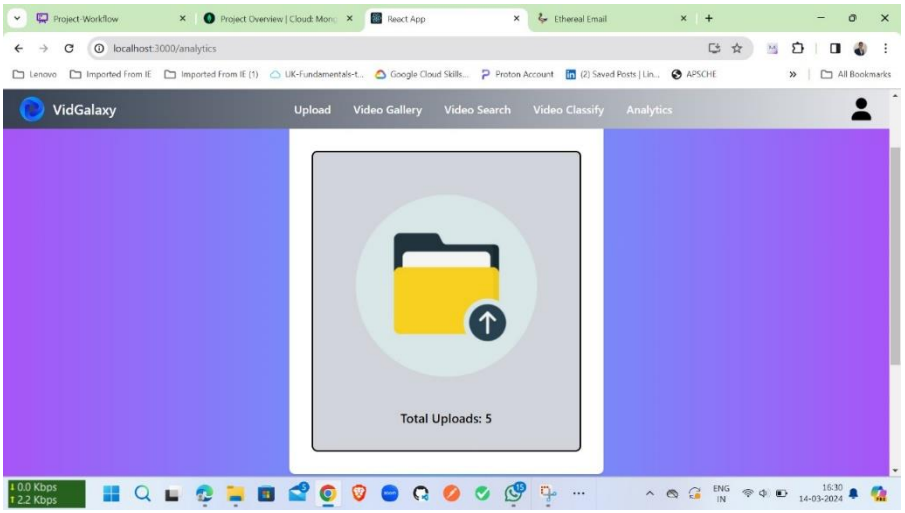


Fig 5.2 Illustration of Analytics component

2. **App.jsx:**

   The App.jsx file serves as the main entry point for the React application. It may define the overall layout and structure of the application, including the routing logic and the placement of various components. This file is crucial for initializing the application and rendering its initial state.

3. **AvatarDropdown.jsx:**

This component provides a dropdown menu that appears when a user clicks on their profile avatar in the header. The dropdown typically contains options such as viewing the user's profile, signing out of the application, and possibly other user-related actions. It enhances user experience by providing quick access to important features.

4. **ChangePassword.jsx:**

This component contains the form and logic for allowing users to change their password. It would include fields for entering the current password, new password, and confirmation, along with validation logic to ensure the password change process is secure and user-friendly.

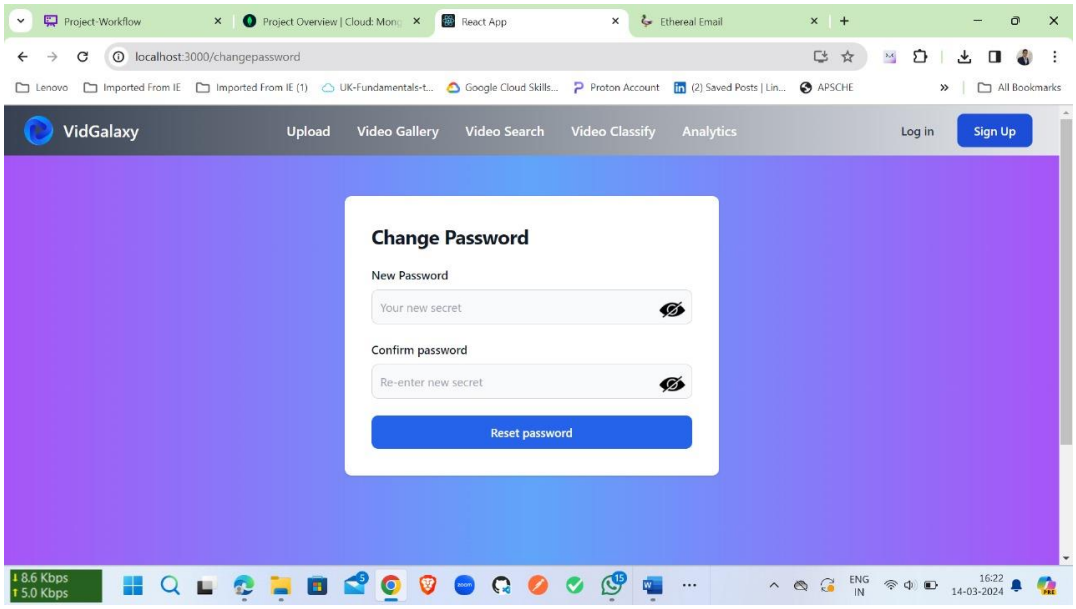


Fig 5.3 Illustration of Change Password component

5. **Footer.jsx**:

The Footer.jsx component is responsible for rendering the footer section of the application. It typically includes information such as copyright notices, links to important pages, and possibly contact information. The footer is a standard part of web design that provides additional navigation and information to users.

6. **ForgotPassword.jsx:**

This component contains the form and logic for handling the “forgot password” functionality. It would include fields for entering the user's email address and logic for sending a password reset email. This feature is important for user convenience.

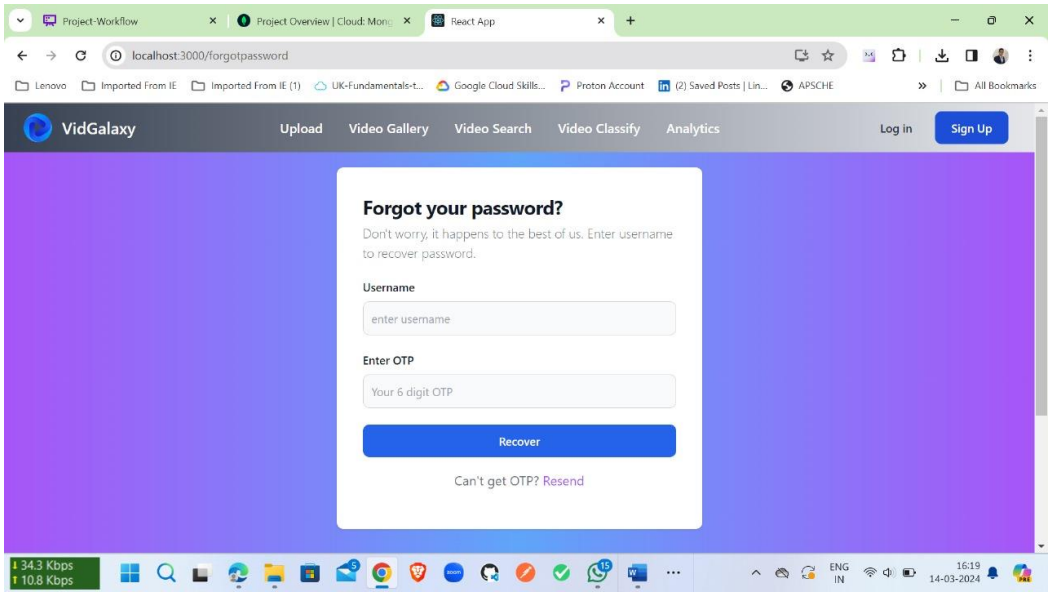


Fig 5.4 Illustration of ForgotPassword component

7. **Header.jsx:**

The Header.jsx component renders the header section of the application, which typically includes the application's logo, navigation menu, and possibly other elements such as a search bar or user profile icon. The header is a key part of the user interface, providing navigation and branding elements.

8. **Home.jsx:**

The Home.jsx component represents the home page of the application. It include introductory content, featured items and Header which contains functionalities like

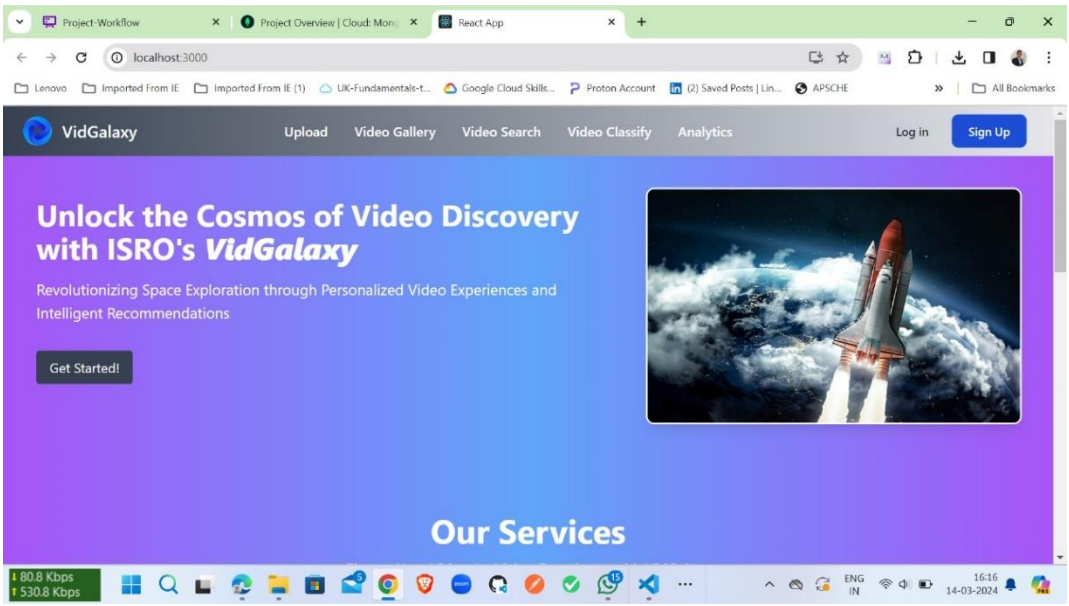


Fig 5.5 Illustration of Home Component

Upload, Video Gallery, VideoSearch, Login, signup. The home page is often the first page that users see when they visit the application.

9. **Login.jsx:**

This component contains the form and logic for the user login functionality. It would include fields for entering the username and password, along with logic for validating the user's credentials and handling the login process. This component is essential for user authentication.

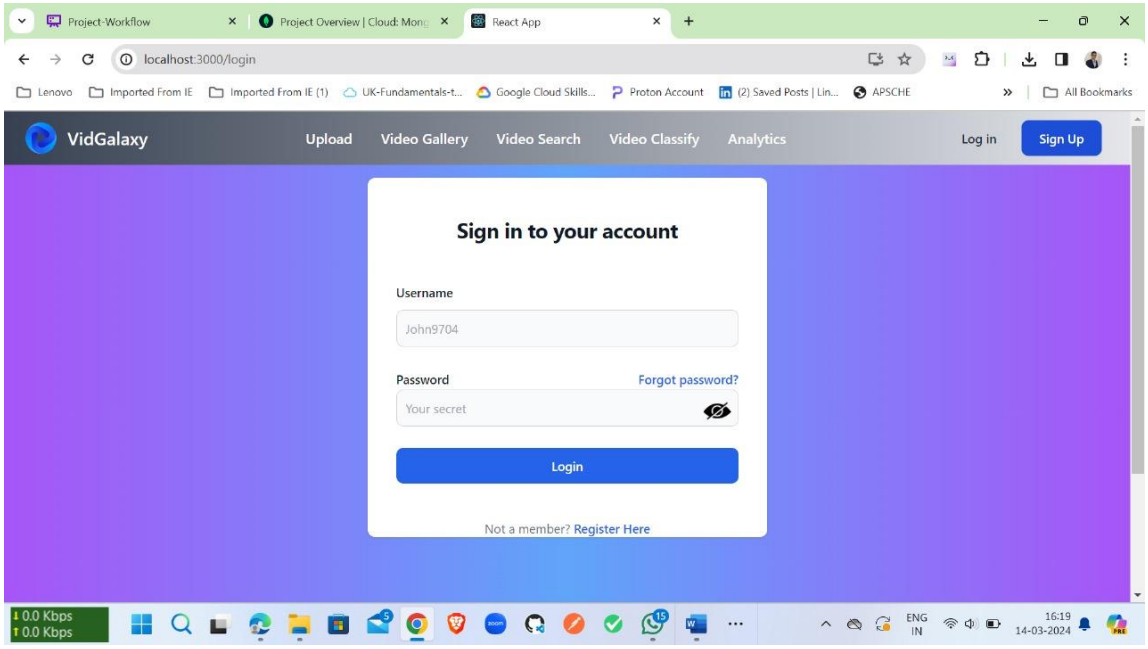


Fig 5.6 Illustration of Login component

10. NotFound.jsx:

The NotFound.jsx component is displayed when a user navigates to a URL that does not exist in the application. It typically provides a friendly message informing the user that the page they are looking for could not be found, along with possibly suggesting some alternative actions or pages to visit.

11. **Profile.jsx:**

This component likely displays the user's profile information, such as their name, email address, and Username. It may also include the ability for the user to edit their profile information, such as updating their First name and LastName. This component is important for user account management.

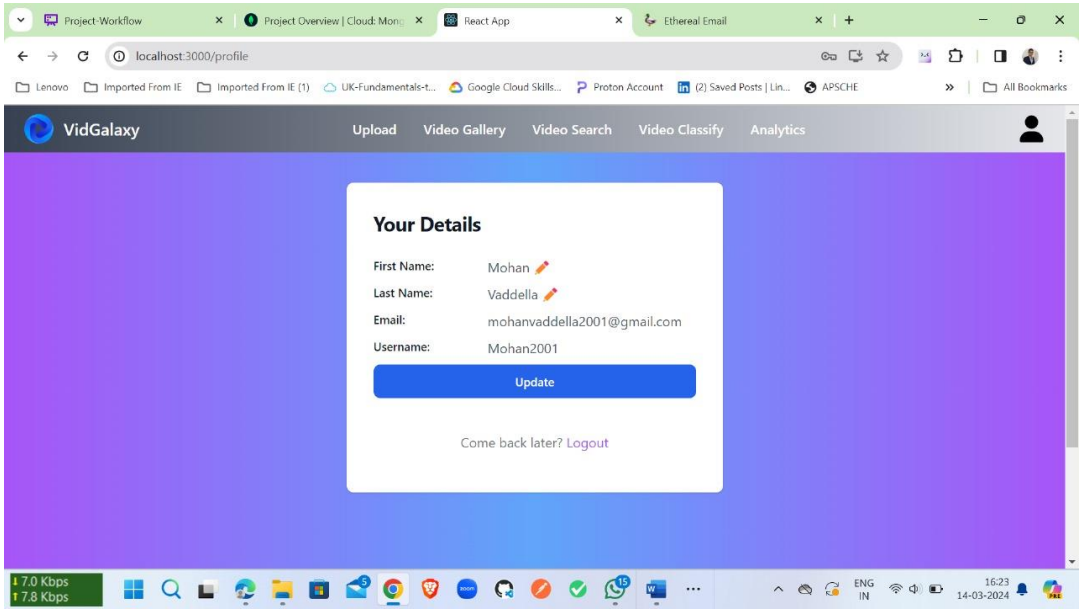


Fig 5.7 Illustration of Profile Component

12. Register.jsx:

The Register.jsx component contains the form and logic for user registration. It would include fields for entering the user's name, email address, password, Username and confirm Password. This component is essential for allowing new users to create accounts in the application.

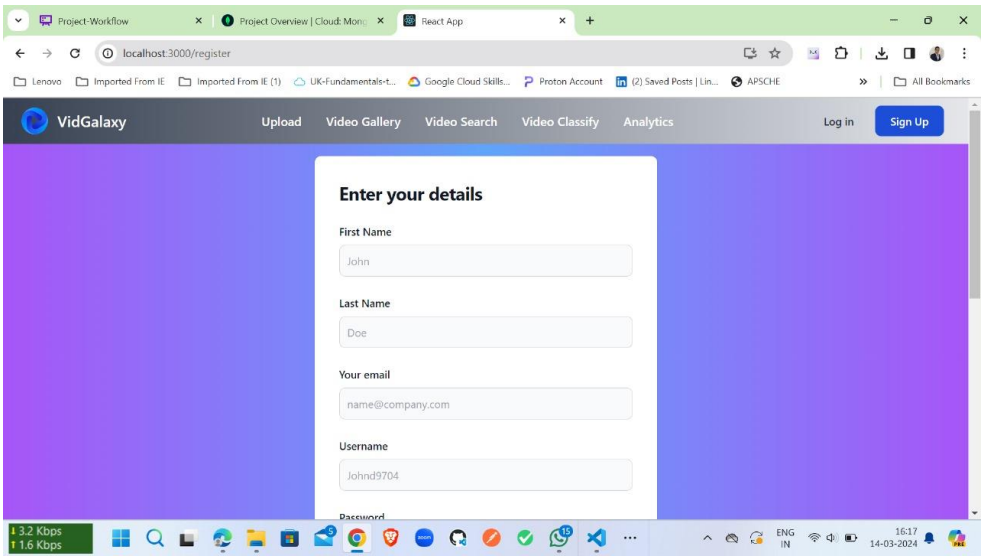


Fig 5.8 Illustration of Register component

13. VideoClassify.jsx:

This component redirects the user to a Django server running a deep learning model for video classification. By using window.location.href, the component navigates the user's

browser to the specified URL, where the deep learning model is hosted. This allows the user to interact with the video classification functionality of the application.

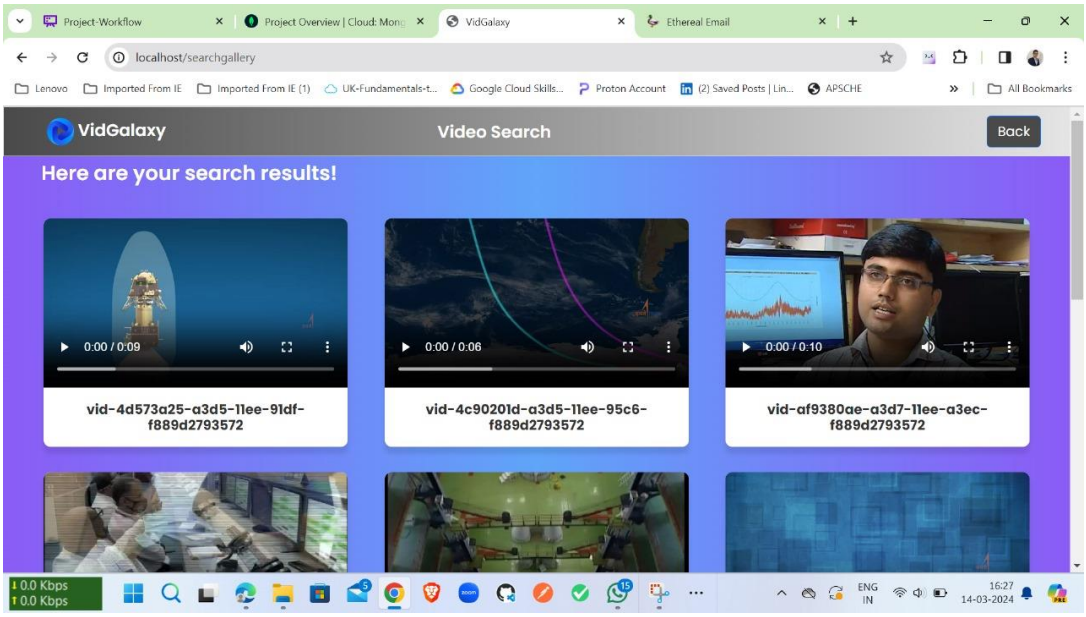


Fig 5.9 Illustration of VideoClassify Component

14. VideoGallery.jsx:

This component represents a gallery of videos in the application. It includes Title of the videos, along with viewing the videos in full screen and sharing the videos by clicking on the share button we will get an sharing url. This component is important for showcasing videos uploaded by users or curated by the application.

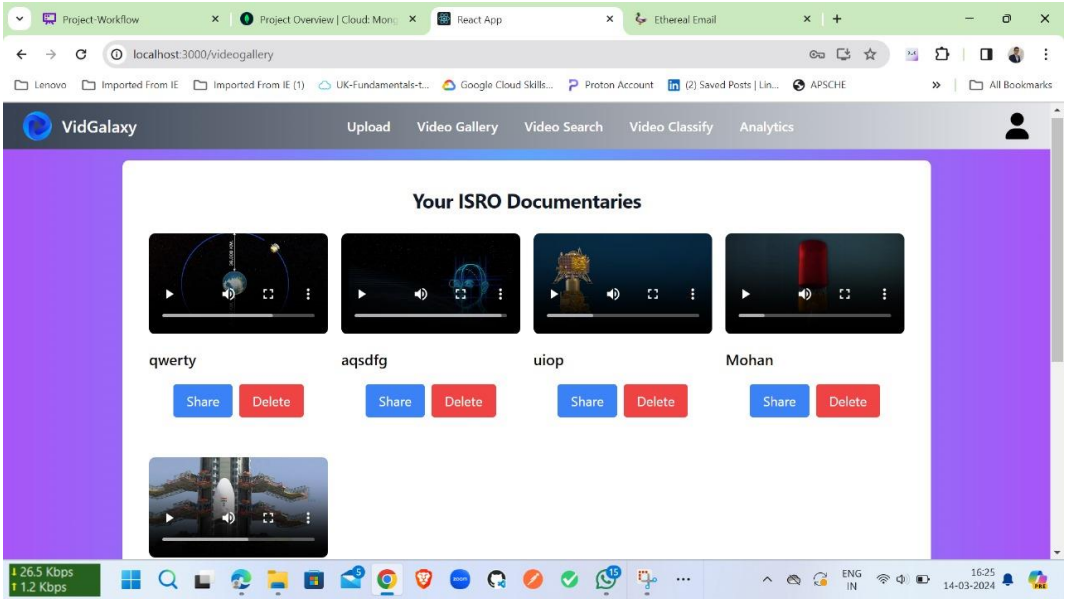


Fig 5.10 Illustration of Videogallery component

15. VideoSearch.jsx:

This component redirects the user to a search page in the Django server, using window.location.href. This allows the user to search for videos based on the uploaded videos or by image.

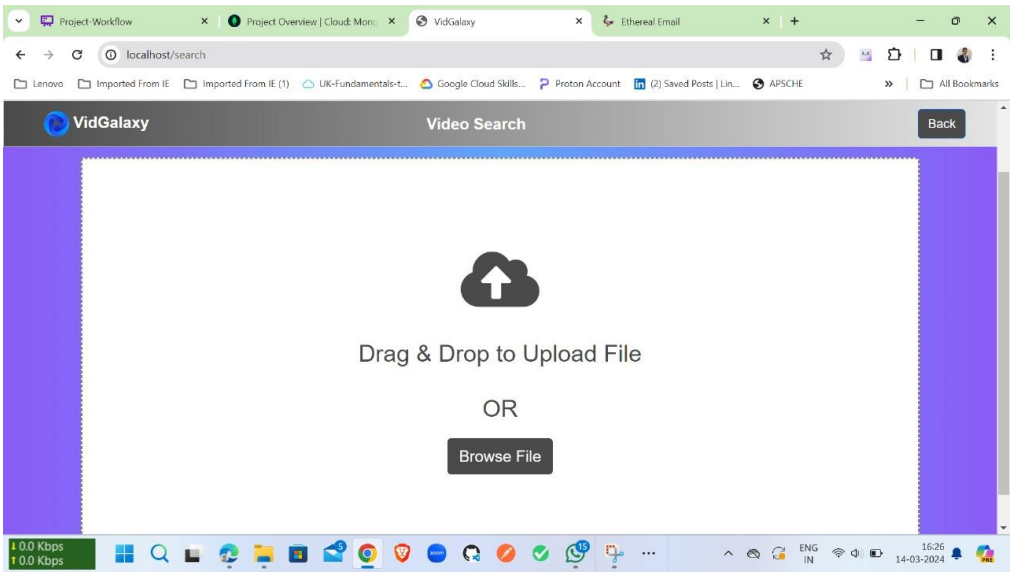


Fig 5.11 Illustration of VideoSearch Component

16. VideoUpload.jsx:

This component contains the logic and interface for users to upload videos to the application. It would include a form for selecting the video file to upload, along with options for adding a title. This component is crucial for allowing users to contribute content to the application.

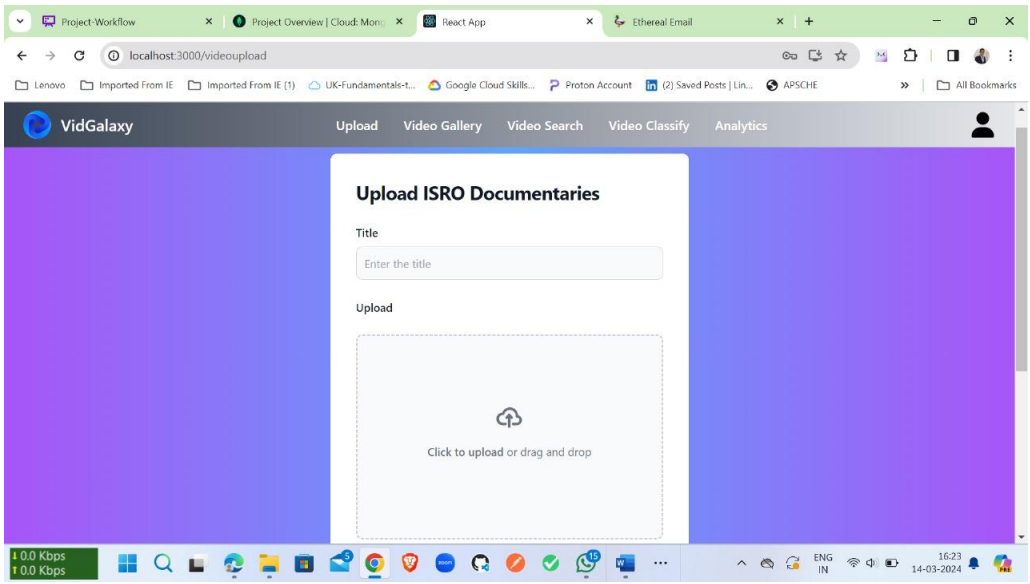


Fig 5.12 Illustration of VideoUpload component

5.2.2 TAILWIND CSS:

Tailwind CSS is basically a Utility first CSS framework for building rapid custom UI. It is a highly customizable, low-level CSS framework that gives you all of the building blocks that you need. Also, it is a cool way to write inline styling and achieve an awesome interface without writing a single line of your own CSS.

As we know there are many CSS frameworks but people always choose the fast and easy framework to learn and use in the project. Tailwind has come with inbuilt a lot of features and styles for users to choose from and is also used to reduce the tendency of writing CSS code and create a beautiful custom UI. It will help you to overcome the complicated task. Tailwind CSS creates small utilities with a defined set of options enabling easy integration of existing classes directly into the HTML code.

Installation of Tailwind CSS:

Basically Tailwind is available on npm and you can install it using the following command:

```
npm install tailwindcss
```

After that create ad Tailwind configuration file using the following command:

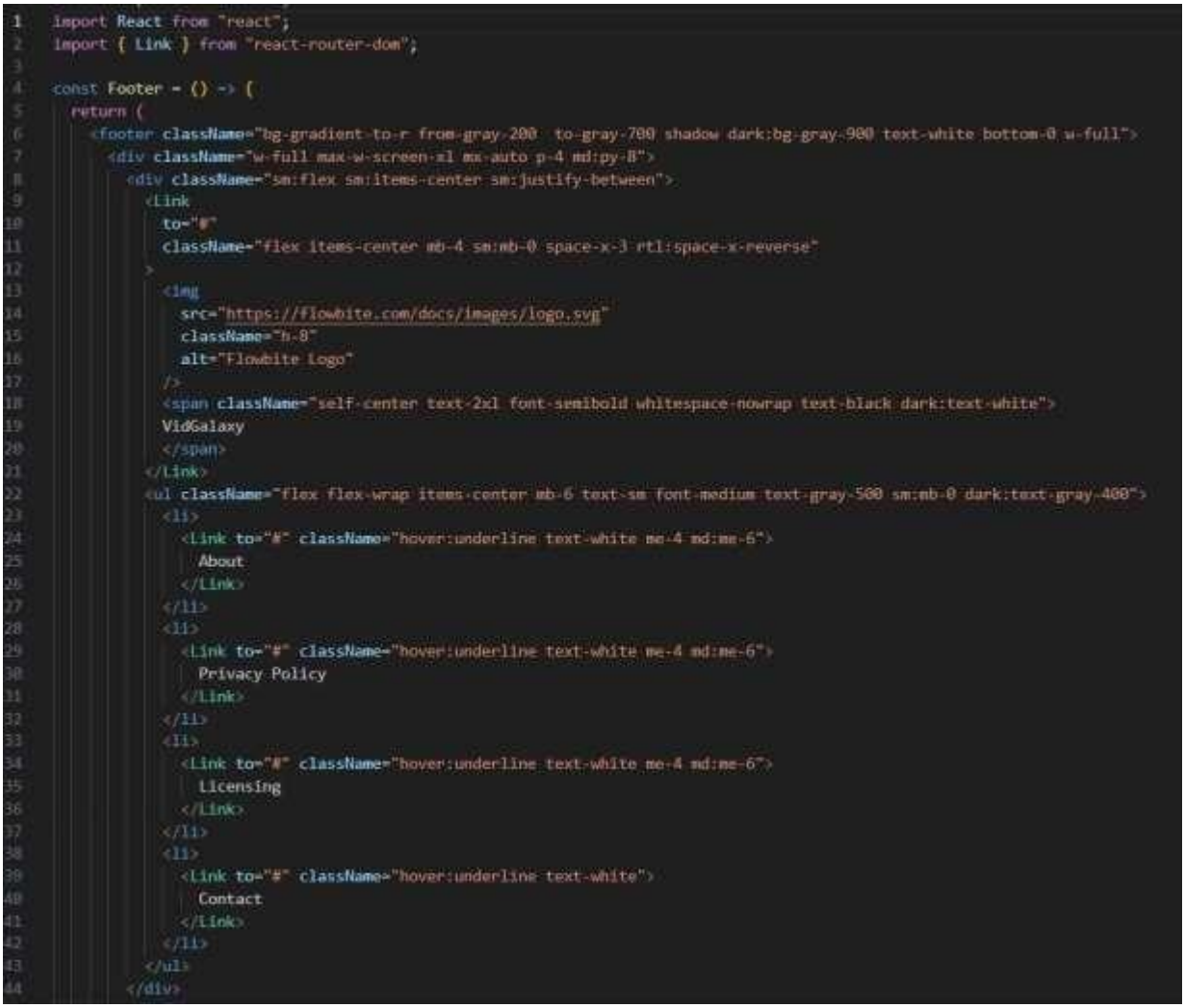
```
npm tailwind init {name of file}
```

Example :

Tailwind CSS is used in the Footer component to style the layout and appearance of the footer section of the web application. Tailwind CSS is a utility-first CSS framework that provides a set of pre-defined classes for styling elements, making it easy to create responsive and visually appealing designs.

In the below component, Tailwind CSS classes are used to set the background gradient (bg-gradient-to-r), text color (text-white), padding (p-4 md:py-8), flexbox layout (flex, items-center, justify-between), and spacing (mb-4 sm:mb-0, space-x-3 rtl:space-x-reverse, me-4 md:me-6, my-6 border-gray-200 sm:mx-auto dark:border-gray-700 lg:my-8).

Tailwind CSS helps maintain consistency in design and speeds up the development process by providing a predefined set of styles and utilities that can be easily applied to elements.



```
import React from "react";
import { Link } from "react-router-dom";

const Footer = () => (
  return (
    <footer className="bg-gradient-to-r from-gray-200  to-gray-700 shadow dark:bg-gray-900 text-white bottom-0 w-full">
      <div className="w-full max-w-screen-xl mx-auto p-4 md:py-8">
        <div className="sm:flex sm:items-center sm:justify-between">
          <Link
            to="#"
            className="flex items-center mb-4 sm:mb-0 space-x-3 rtl:space-x-reverse"
          >
            <img
              src="https://flowbite.com/docs/images/logo.svg"
              className="h-8"
              alt="Flowbite Logo"
            />
            <span className="self-center text-2xl font-semibold whitespace-nowrap text-black dark:text-white">
            VidGalaxy
            </span>
          </Link>
          <ul className="flex flex-wrap items-center mb-6 text-sm font-medium text-gray-500 sm:mb-0 dark:text-gray-400">
            <li>
              <Link to="#" className="hover:underline text-white me-4 md:me-6">
                About
              </Link>
            </li>
            <li>
              <Link to="#" className="hover:underline text-white me-4 md:me-6">
                Privacy Policy
              </Link>
            </li>
            <li>
              <Link to="#" className="hover:underline text-white me-4 md:me-6">
                Licensing
              </Link>
            </li>
            <li>
              <Link to="#" className="hover:underline text-white">
                Contact
              </Link>
            </li>
          </ul>
        </div>
```

Fig 5.13 Demonstrating TailwindCSS

5.3 DATABASE:

To store the user details like username and password we need an database, not only for storing the details for authorization of an user we need a database to provide security to the user information and user activities we need database.

5.3.1 MongoDB Atlas:

MongoDB Atlas is a cloud service by MongoDB. It is built for developers who'd rather spend time building apps than managing databases. This service is available on AWS, Azure, and GCP.

It is the worldwide cloud database service for modern applications that give best-in-class automation and proven practices guarantee availability, scalability, and compliance with the foremost demanding data security and privacy standards. We can use MongoDB's robust

ecosystem of drivers, integrations, and tools to create faster and spend less time managing our database.

Connecting our React project to MongoDB Atlas, you'll need to follow these steps:

1. **Create a MongoDB Atlas Account and Cluster:**

- Go to the MongoDB Atlas website (https://www.mongodb.com/cloud/atlas) and sign up for an account.
- Create a new cluster (or use an existing one) and follow the setup instructions.

2. **Get Connection String:**

- In your Atlas dashboard, click on "Connect" for your cluster.
- Select "Connect your application."
- Copy the connection string.

3. **Install MongoDB Node.js Driver:**

In your React project directory, install the MongoDB Node.js driver:

```
npm install mongodb
```

4. **Create a MongoDB Connection File:**

- Create a new file, for example, mongo.js, in your project's src directory.

  Paste the following code into the file, replacing <your_connection_string> with your actual connection string:

```
import { MongoClient } from 'mongodb';

const uri = '<your_connection_string>';

const client = new MongoClient(uri, {

useNewUrlParser: true,

useUnifiedTopology: true,

});

export async function connectToDatabase() {
```

```
  try {
    await client.connect();
    console.log('Connected to MongoDB Atlas');
    return client.db('your-database-name');
  } catch (error) {
    console.error('Error connecting to MongoDB Atlas:', error);
    throw error;
  }
}
```

5. Use the MongoDB Connection in Your React Components:

- Import the connectToDatabase function wherever you need to access the database.
- Use it to connect to the database and perform operations.

Example usage in a component:

```
import React, { useEffect } from 'react';
import { connectToDatabase } from './mongo.js';
function MyComponent() {
  useEffect(() => {
    async function fetchData() {
      const db = await connectToDatabase();
      const collection = db.collection('your-collection-name');
      const result = await collection.find({}).toArray();
      console.log(result);
    }
    fetchData();
```

```
  }, []);

  return <div>My Component</div>;

}

export default MyComponent;
```

Replace 'your-database-name' with your actual database name and 'your-collection-name' with the name of the collection you want to query.

Below figure shows that they are two clusters in our atlas users and videos, in users we can see the users who had signed in, we store id, username, password, email, firstname, lastname, privilege.

In the videos cluster we have the video tile, id, and url of the video which had been stored in s3 bucket.

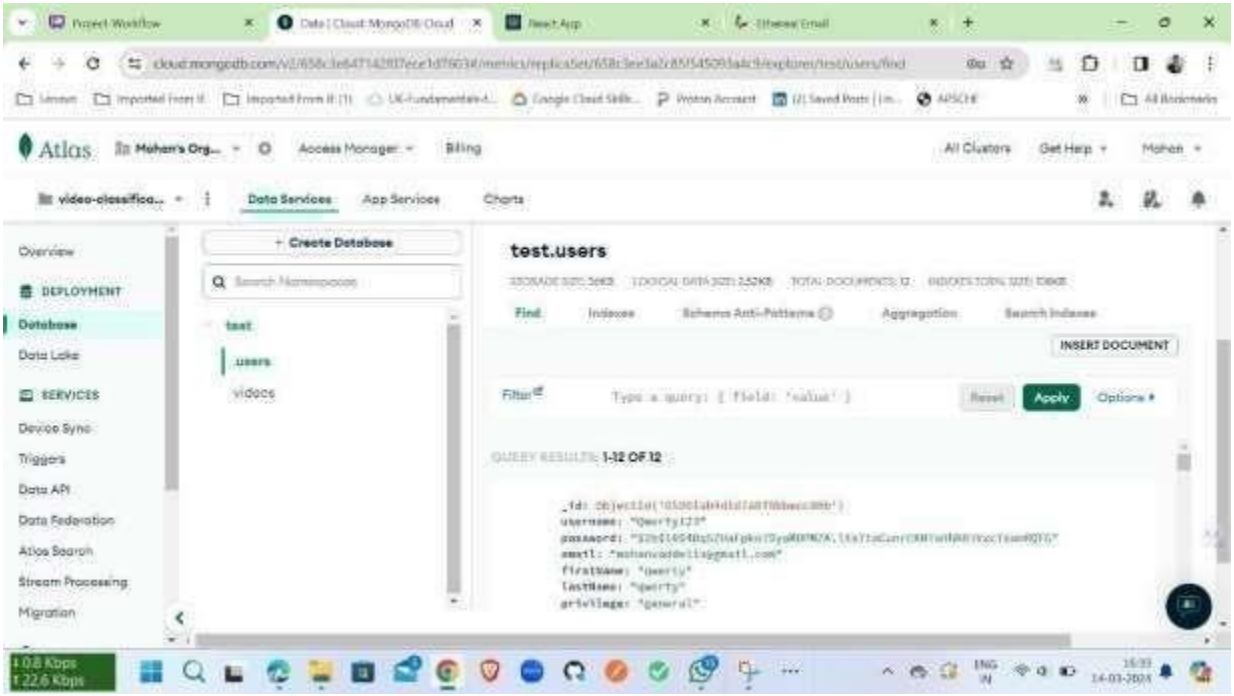


Fig 5.13 MongoDB User Collection

5.4 Backend Implementation:

In this project, we implemented the backend in two parts:

Backend for the Frontend Web Application:

- This backend handles user authentication, file uploads, and other frontend-related functionality.
- It communicates with the frontend to provide data and services.

Backend for the Deep Learning Model:

- We used Django to create a separate backend for the deep learning model.
- This backend handles the training, inference, and management of the deep learning model.
- It provides APIs for the frontend to interact with the model, such as sending images for classification and receiving the results.

These two backends work together to provide a complete solution for the web application, with Node.js and Express.js handling the frontend-related tasks and Django handling the deep learning model-related tasks.

5.4.1 NodeJS:

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to run JavaScript code outside of a web browser. It is built on Chrome's V8 JavaScript engine and provides an event-driven, non-blocking I/O model that makes it lightweight and efficient for building scalable network applications. Node.js is commonly used for building web servers, but it can also be used for a variety of other applications, including command-line tools, desktop applications, and even Internet of Things (IoT) devices.

One of the key features of Node.js is its package ecosystem, npm, which is one of the largest ecosystems of open-source libraries. npm allows developers to easily share and reuse code, making it quick and efficient to build applications. Node.js also has a strong community and active development, which ensures that it stays up-to-date with the latest trends and best practices in web development. Overall, Node.js is a powerful and versatile platform for building a wide range of applications, from small, lightweight services to large-scale, real-time applications.

5.4.2 Express JS:

Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. It is designed to make it easy to build

web servers and APIs with Node.js by providing a simple and intuitive API for defining routes, handling requests and responses, and managing middleware.

One of the key features of Express.js is its middleware system, which allows developers to easily add functionality to their applications. Middleware functions can be used for tasks such as logging, authentication, and error handling, making it easy to modularize and reuse code.

Express.js also provides a powerful routing system that allows developers to define routes based on HTTP methods and URL paths. This makes it easy to create RESTful APIs and handle complex routing logic in a clean and organized way.

Another key feature of Express.js is its support for templating engines, which allows developers to dynamically generate HTML content based on data from their application. Express.jssupports a variety of templating engines, including Pug, EJS, and Handlebars, making it flexibleand adaptable to different project requirements.

Overall, Express.js is a lightweight and flexible framework that simplifies the process of building web applications with Node.js. Its simplicity, flexibility, and robust feature set make it a popular choice for developers building web applications and APIs with Node.js.

5.4.3 Python:

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

There are two major Python versions: Python 2 and Python 3. Both are quite different.

Features of Python :

Python offers a plethora of advantages for machine learning professionals and enthusiasts alike, especially when working with machine learning models using the Python language.

1. **Easy-to-read syntax:**

Python's syntax is designed to be intuitive and straightforward, making it a popular programming language that is easy to read. Object-oriented programming provides developers with a logical method to organize, process and plan code accordingly.

The easy-to-read syntax of Python not only makes it accessible to beginners, but also allows for faster development and debugging. With Python, code is more legible and easier to debug, making it easier to identify and rectify mistakes and promptly develop new features. This user-friendly nature of Python has contributed significantly to its widespread adoption in the machine learning community.

2. Extensive libraries and frameworks:

One of the key factors that sets Python apart from other programming languages is its comprehensive library ecosystem. Python offers a wide range of libraries and frameworks specifically designed for machine learning, making it easier for developers to implement ML algorithms. Some popular Python libraries for machine learning include:

1. **NumPy**: NumPy is a fundamental Python library for efficient numerical computations and array operations.
2. **Scikit-learn:** Scikit-learn is a comprehensive machine learning library that offers a wide range of tools for various tasks, including classification, regression, clustering, and more.
3. **Pandas:** Pandas is a powerful library for data analysis and manipulation, providing intuitive data structures like DataFrames and Series.
4. **TensorFlow:** TensorFlow is a cutting-edge deep learning library known for its distributed computing capabilities and robust ecosystem.
5. **Theano:** Theano is a Python library designed for fast numerical computation, particularly useful for training deep learning models.
6. **Keras:** Keras is an easy-to-use deep learning API that acts as an interface for TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK), simplifying the creation and training of neural networks.
7. **PyTorch**: PyTorch is a dynamic deep learning library with a flexible computation graph, making it ideal for developing and training complex neural networks.

These libraries and Python frameworks provide powerful capabilities for data analysis, machine learning, and deep learning, allowing developers to focus on solving complex tasks without

having to reinvent the wheel. With this great library ecosystem, Python has become an indispensable tool for machine learning engineers, data scientists, and researchers alike.

3. Cross-platform compatibility:

Python's cross-platform compatibility enables developers to create code that can be utilized across various platforms, such as Windows, Mac, and Linux. This flexibility facilitates the development of applications that can be used on different operating systems without the need to rewrite source code. Thus, it allows developers to utilize the same code for different platforms, saving time and effort.

However, cross-platform compatibility does come with its share of challenges. Different platforms may have different versions of Python installed, which can lead to compatibility issues when running code on different platforms. To overcome these challenges, it is essential to ensure that the code is written in a manner that is compatible with all supported versions and that it is tested on all platforms to guarantee that it functions as anticipated.

4. Scaling and Performance:

Python is widely renowned for its scalability and exceptional performance in machine learning. Its versatility, user-friendly nature, and extensive libraries make it an ideal choice for scaling ML operations. With libraries like NumPy, pandas, and TensorFlow, Python empowers complex operations on massive datasets, demonstrating its high scalability. Its proficiency in handling big data contributes to its widespread adoption.

However, Python's performance does pose challenges. As an interpreted language, Python is relatively slower compared to languages like C++ or Java. In addition, distributed computing frameworks like Apache Spark and Dask greatly enhance Python's performance in ML applications.

Overall, Python's rich array of libraries, ease of use, and scalability make it a robust choice for machine learning.

Because of above features we picked the python program language for our project development.

5.4.4 Django:

Django is a Python-based web framework that allows you to create efficient web applications quickly. It is also called batteries included framework because Django provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc.

Django gives you ready-made components to use such as:

1. It's very easy to switch databases in the Django framework.
2. It has a built-in admin interface which makes it easy to work with it.
3. Django is a fully functional framework that requires nothing else.
4. It has thousands of additional packages available.
5. It is very scalable.

Features of Django

- **The versatility of Django**: Django can build almost any type of website. It can also work with any client-side framework and can deliver content in any format such as HTML, JSON, XML, etc. Some sites which can be built using Django are wikis, social networks, new sites etc.
- **Security**: Since the Django framework is made for making web development easy, it has been engineered in such a way that it automatically do the right things to protect the website. For example, In the Django framework instead of putting a password in cookies, the hashed password is stored in it so that it can't be fetched easily by hackers.
- **Scalability**: Django web nodes have no stored state, they scale horizontally – just fire up more of them when you need them. Being able to do this is the essence of good scalability. Instagram and Disqus are two Django based products that have millions of active users, this is taken as an example of the scalability of Django.
- **Portability**: All the codes of the Django framework are written in Python, which runs on many platforms. Which leads to run Django too in many platforms such as Linux, Windows and Mac OS.

The trained Model from the Colab is saved in our local storage with model.h5 this model is been loaded into our backend using the tensorflow load model method.

Django Project setup:

1. **Install Django:** If you haven't already installed Django, you can do so using pip:

```
pip install Django
```

2. **Create a Django Project:** Run the following command to create a new Django project:

```
django-admin startproject projectname
```

Replace **projectname** with the name of your project.

3. **Create a Django App:** Inside your project directory, create a new Django app using the following command:

```
python manage.py startapp appname
```

Replace **appname** with the name of your app.

4. **Configure Settings:** Open the **settings.py** file inside your project directory and add your app to the **INSTALLED_APPS** list:

```
INSTALLED_APPS = [ ... 'appname', ]
```

5. **Database Configuration:** Configure your database settings in the **settings.py** file. By default, Django uses SQLite. For other databases, you'll need to install the respective database adapter and update the settings accordingly.

6. **Run Migrations:** Run the initial database migrations to set up your database schema:

```
python manage.py migrate
```

7. **Create a Superuser:** Create a superuser to access the Django admin interface:

```
python manage.py createsuperuser
```

8. **Start the Development Server:** Start the Django development server to test your project:

```
python manage.py runserver
```

9. **Access the Admin Interface:** Open a web browser and go to **http://127.0.0.1:8000/admin/**. Log in with the superuser credentials created earlier to access the Django admin interface.

10. **Create Views, URLs, and Templates:** Define your views, URLs, and templates to build the functionality of your Django app.

5.4.5 Jinja2:

Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.

It includes:

- Template inheritance and inclusion.
- Define and import macros within templates.
- HTML templates can use autoescaping to prevent XSS from untrusted user input.
- A sandboxed environment can safely render untrusted templates.
- Async support for generating templates that automatically handle sync and async functions without extra syntax.
- Templates are compiled to optimized Python code just-in-time and cached, or can be compiled ahead-of-time.
- Exceptions point to the correct line in templates to make debugging easier.
- Extensible filters, tests, functions, and even syntax.

Usage of Jinja2 in Django Project:

1.after installing Jinja2 by pip install Jinja2

2. add it to requirements.txt. Then add following template engine setting to TEMPLATES variable in settings.py

```
TEMPLATES = [
        {
                "BACKEND": "django.template.backends.jinja2.Jinja2",
                "DIRS": [],
                "APP_DIRS": True,
                "OPTIONS": {
                        "environment": "your-app.jinja2.environment"
                }
        },
        {
                "BACKEND": "django.template.backends.django.DjangoTemplates",
```

```
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ]
        }
    }
]
```

3. remember also to change Jinja2 options environment variable to the correct app name (change "your-app" to whatever your application folder is called).

The options environment enables us to use certain functions in templates that we will setup next. Create jinja2.py file to your application folder (same place where the settings.py was) and add the following:

```
from jinja2 import Environment
from django.urls import reverse
from django.contrib.staticfiles.storage import staticfiles_storage
# for more later django installations use:
# from django.templatetags.static import static
Def environment(**options):
    env = Environment(**options)
    env.globals.update({
        "static": staticfiles_storage.url,
        "url": reverse
    })
    return env
```

This enables us to use Django template tags like {% url "index" %} or {% static "path/to/static/file.js" %} in our Jinja2 templates. As you can see, these template tags use the actual functions provided by Django.

In Django’s:

{% url "index" variable %}

is equivalent to:

{{ url("index", args=[variable]) }}

5.5 Cloud Services:

For a real time project we should need to use the cloud services

5.5.1 Amazon S3:

Amazon S3 (Simple Storage Service) is a scalable object storage service offered by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data from anywhere on the web. S3 is commonly used for backup and storage of data, hosting static websites, and serving as a content delivery network (CDN).

Key features of Amazon S3 include:

Scalability: S3 can scale to accommodate virtually unlimited amounts of data. It automatically scales storage capacity, bandwidth, and request handling based on your needs.

Durability and Availability: S3 is designed for 99.999999999% (11 9's) durability and 99.99% availability of objects over a given year. This high durability ensures that your data is protected against hardware failures and other issues.

Security: S3 offers multiple layers of security to protect your data. You can manage access permissions using IAM (Identity and Access Management) roles, bucket policies, and access control lists (ACLs). Additionally, data is encrypted at rest and in transit.

Data Lifecycle Management: S3 provides features to manage the lifecycle of your data, including transitioning objects between storage classes, setting expiration dates for objects, and automatically deleting older versions of objects.

Versioning: S3 supports versioning, which allows you to preserve, retrieve, and restore every version of every object stored in a bucket. This feature is useful for maintaining a history of changes to your data.

Storage Classes: S3 offers different storage classes to optimize costs and performance based on your needs. These include Standard, Intelligent-Tiering, Standard-IA (Infrequent Access), One Zone-IA, Glacier, and Glacier Deep Archive.

Static Website Hosting: You can use S3 to host static websites, serving HTML, CSS, JavaScript, and other static content directly from your bucket.

Data Transfer Acceleration: S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and your S3 bucket.

In our application, we utilize two S3 buckets for different purposes:

1. **Categorical Videos Bucket:**

- This S3 bucket is used for storing videos categorized for display in the Recommended Videos section.
- The videos stored in this bucket are organized based on their categories or tags.
- This bucket helps in efficiently retrieving and displaying videos based on user preferences and interests.

2. **User Uploaded Videos Bucket:**

In our application, we have a dedicated S3 bucket for storing videos uploaded by users. To ensure proper organization and management, each user's uploaded videos are stored in a separate folder or with a unique identifier within the bucket.

This approach allows us to efficiently manage and retrieve videos based on user activity, ensuring a seamless user experience.

- This S3 bucket is dedicated to storing videos uploaded by users of our application.
- Each user's uploaded videos are stored in a separate folder or with a unique identifier to ensure proper organization and management.

- This bucket allows users to upload, manage, and share their own videos within the application.

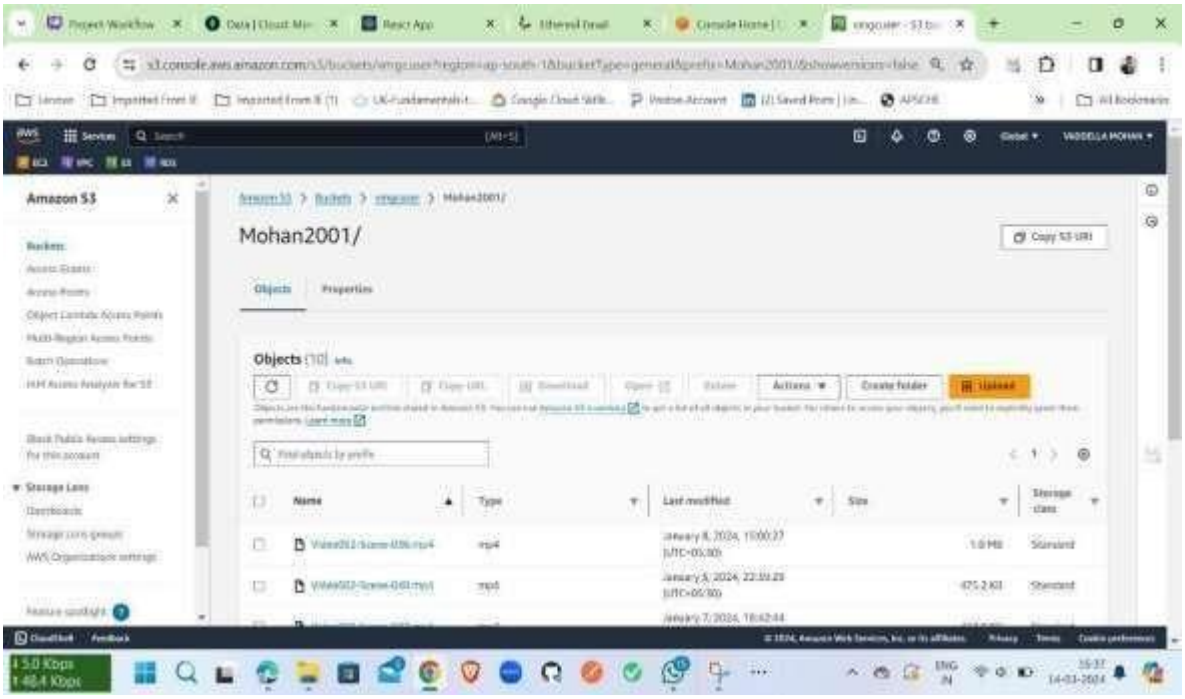


Fig 5.14 S3 Bucket user details

From Above figure Mohan2001 is the username of a user and he had uploaded 10 videos. Likewise for every individual user same like above a new folder with name of username will be created and all the uploaded videos will be stored in that specific folder.

This below is the policy attached to the IAM user to access vmgcuser bucket:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Statement1",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "s3:*",
   "Resource": "arn:aws:s3:::vmgcuser/*"
  }
 ]
}
```

5.5.2 Amazon IAM:

AWS IAM (Identity and Access Management) is a service that helps you securely control access to AWS resources. IAM allows you to manage users, groups, and permissions to access AWS services and resources. With IAM, you can create and manage AWS users and groups, and use permissions to allow or deny their access to AWS resources. IAM also enables you to set up roles for users, services, and resources to ensure secure access within your AWS environment.

Key features of AWS IAM include:

1. **Users and Groups:** You can create IAM users for individuals who need access to the AWS Management Console or AWS services programmatically. Users can be grouped together for easier management of permissions. **Roles**: IAM roles are used to delegate access to AWS services, resources, and APIs. Roles are often used to grant permissions to applications running on Amazon EC2 instances, AWS Lambda functions, or other AWS services.
2. **Permissions**: IAM uses policies to define permissions. Policies are JSON documents that specify what actions are allowed or denied on which resources. Policies can be attached to users, groups, or roles.
3. **Multi-factor Authentication (MFA):** IAM supports MFA, adding an extra layer of security to user sign-ins and API calls.
4. **Identity Federation:** IAM supports identity federation, allowing you to use external identities (such as Active Directory or Facebook) to grant access to AWS resources without having to create IAM users.
5. **Access Analyzer:** IAM Access Analyzer helps you identify resources that can be accessed from outside your AWS account, making it easier to adhere to security best practices.
6. **Integration with AWS Services:** IAM integrates with many AWS services, such as Amazon S3, Amazon RDS, and AWS Lambda, allowing you to control access to these services using IAM policies.

5.5.3 Amazon EC2:

EC2 stands for Elastic Compute Cloud. EC2 is an on-demand computing service on the AWS cloud platform. Under computing, it includes all the services a computing device can offer to you along with the flexibility of a virtual environment.

It also allows the user to configure their instances as per their requirements i.e. allocate the RAM, ROM, and storage according to the need of the current task. Even the user can dismantle the virtual device once its task is completed and it is no more required.

For providing, all these scalable resources AWS charges some bill amount at the end of every month, the bill amount is entirely dependent on your usage. EC2 allows you to rent virtual computers.

The provision of servers on AWS Cloud is one of the easiest ways in EC2. EC2 has resizable capacity. EC2 offers security, reliability, high performance, and cost-effective infrastructure so as to meet the demanding business needs.

Amazon Web Service EC2 is a web service which is provided by the AWS cloud which is secure, resizable, and scalable. These virtual machines are pre-configured with the operating systems and some of the required software.

The other advantage of AWS EC2 is that you need to pay only for how much you use it is like the pay-as-you-go model.

Steps to create an Ec2 instance:

1. **Sign in to the AWS Management Console**: Go to https://aws.amazon.com/ and sign in to the AWS Management Console.
2. **Navigate to the EC2 Dashboard:** In the AWS Management Console, navigate to the EC2 dashboard by selecting "Services" in the top navigation bar, and then selecting "EC2" under the "Compute" section.
3. **Click on "Launch Instance":** On the EC2 dashboard, click on the "Launch Instance" button to start the instance creation process.
4. **Choose an Amazon Machine Image (AMI):** Select an AMI that suits your requirements. You can choose from a variety of pre-configured AMIs, including

Amazon Linux, Ubuntu, Windows Server, and more.

5. **Choose an Instance Type:** Select an instance type based on your workload requirements. Instance types vary in terms of compute, memory, and storage capacity.
6. **Configure Instance Details:** Configure additional settings for your instance, such as the number of instances, network settings, and IAM role (if required).
7. **Add Storage:** Specify the size and type of storage for your instance. You can add additional EBS volumes if needed.
8. **Add Tags (Optional):** Add tags to your instance to easily identify it in the AWS Management Console. Tags are key-value pairs that can be used for organization and resource management.
9. **Configure Security Group:** Create a new security group or select an existing one. Security groups control inbound and outbound traffic to your instance.
10. **Review Instance Launch:** Review the configuration of your instance and make any necessary changes. Click on "Launch" to proceed.
11. **Create a Key Pair:** If you haven't already created a key pair, you'll be prompted to create one. A key pair is required to connect to your instance securely using SSH (for Linux instances) or RDP (for Windows instances).
12. **Launch Instance:** Click on "Launch Instances" to launch your EC2 instance. Once the instance is launched, you can view its status on the EC2 dashboard..

After launching an EC2 instance, you can install Docker on the instance and then build your Docker image to deploy your web application. Here's a detailed guide on how to do this:

1. Connect to Your EC2 Instance:

- Use SSH to connect to your EC2 instance. You'll need the key pair you created when launching the instance.
- Example: ssh -i your-key.pem ec2-user@your-instance-ip

2. Update the Package Repository:

- It's a good practice to update the package repository to ensure you install the latest versions of Docker.
- Run: sudo yum update -y

3. Install Docker:

- Install Docker using the yum package manager**.**

Run:

```
sudo amazon-linux-extras install docker -y
sudo service docker start
```

4. Add the ec2-user to the Docker Group:

- This allows the ec2-user to execute Docker commands without using sudo.

Run:

```
sudo usermod -a -G docker ec2-user
```

5. Verify Docker Installation:

- Check that Docker is installed and running correctly.

Run:

```
docker --version and docker info
```

6. Build Your Docker Image:

- Navigate to your project directory where your Dockerfile is located.
- Build your Docker image using the Dockerfile.
- Run: `docker build -t your-image-name` .

7. Run Your Docker Container:

- Once your image is built, you can run a Docker container from it.
- Run: docker run -d -p your-host-port:container-port your-image-name
- Replace your-host-port with the port on your EC2 instance you want to expose (e.g., 80 for HTTP).
- Replace container-port with the port your application is listening on inside the Docker container.

8. Access Your Web Application:

- Once your container is running, you should be able to access your web application

through the public IP address or public DNS of your EC2 instance.

- Open a web browser and navigate to http://your-ec2-public-ip or http://your-ec2-public-dns

5.5.4 Amazon VPC:

With Amazon Virtual Private Cloud (Amazon VPC), you can launch AWS resources in a logically isolated virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

The following diagram shows an example VPC. The VPC has one subnet in each of the Availability Zones in the Region, EC2 instances in each subnet, and an internet gateway to allow communication between the resources in your VPC and the internet.

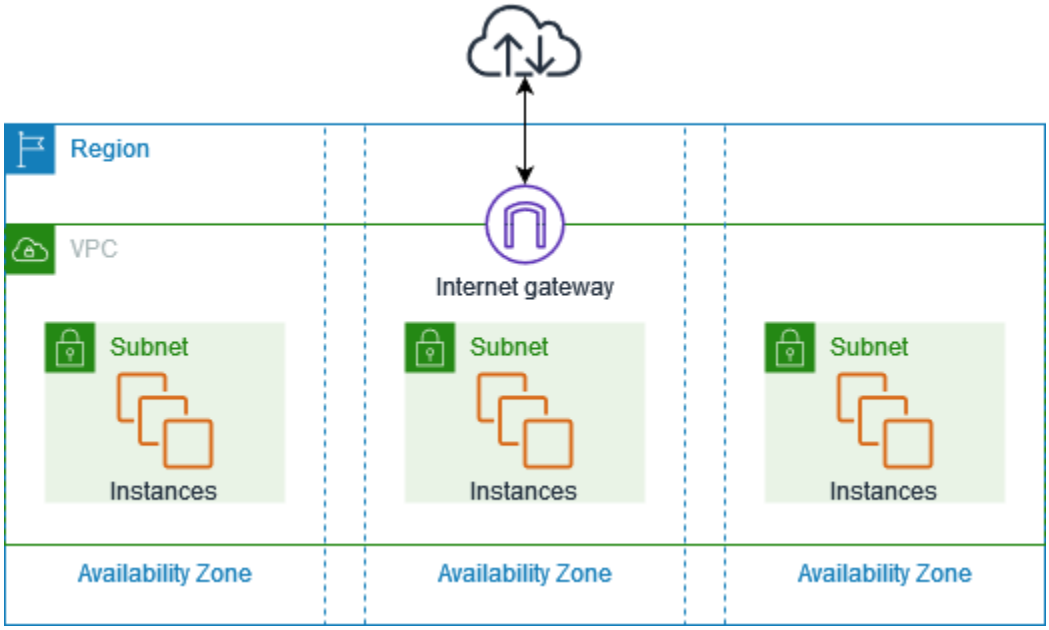


Fig 5.15 VPC Demonstration

Features:

The following features help you configure a VPC to provide the connectivity that your applications need:

1. **Virtual private clouds (VPC):** A VPC is a virtual network that closely resembles a traditional network that you'd operate in your own data center. After you create a VPC, you can add subnets.
2. **Subnets:** A subnet is a range of IP addresses in your VPC. A subnet must reside in a

single Availability Zone. After you add subnets, you can deploy AWS resources in your VPC.

3. **IP addressing:** You can assign IP addresses, both IPv4 and IPv6, to your VPCs and subnets. You can also bring your public IPv4 addresses and IPv6 GUA addresses to AWS and allocate them to resources in your VPC, such as EC2 instances, NAT gateways, and Network Load Balancers.
4. **Routing:** Use route tables to determine where network traffic from your subnet or gateway is directed.
5. **Gateways and endpoints:** A gateway connects your VPC to another network. For example, use an internet gateway to connect your VPC to the internet. Use a VPC endpoint to connect to AWS services privately, without the use of an internet gateway or NAT device.
6. **Peering connections:** Use a VPC peering connection to route traffic between the resources in two VPCs.
7. **Traffic Mirroring:** Copy network traffic from network interfaces and send it to security and monitoring appliances for deep packet inspection.
8. **VPC Flow Logs:** A flow log captures information about the IP traffic going to and from network interfaces in your VPC.

4.5.5 Amazon EKS:

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service provided by AWS. It allows you to run Kubernetes clusters without the need to manage the underlying infrastructure. Here's an overview of Amazon EKS:

1. **Kubernetes Management**: Amazon EKS simplifies the process of deploying, managing, and scaling Kubernetes clusters. It provides a fully managed Kubernetes control plane that you can interact with using the standard Kubernetes API.
2. **High Availability:** Amazon EKS ensures high availability of your Kubernetes clusters by distributing them across multiple Availability Zones within a region. This helps in minimizing downtime and improving the resilience of your applications.
3. **Security:** Amazon EKS integrates with AWS Identity and Access Management (IAM) for authentication and authorization, allowing you to control access to your clusters. It also supports encryption of data in transit and at rest to enhance the security of your

clusters.

4. **Networking:** Amazon EKS integrates with Amazon Virtual Private Cloud (VPC) to provide networking capabilities for your clusters. You can use VPC networking features such as security groups and network access control lists (ACLs) to control inbound and outbound traffic to your clusters.
5. **Scaling**: Amazon EKS allows you to scale your clusters dynamically based on workload requirements. You can use the Kubernetes Horizontal Pod Autoscaler (HPA) to automatically scale the number of pods in your cluster based on metrics such as CPU utilization or custom metrics.
6. **Monitoring and Logging**: Amazon EKS integrates with AWS CloudWatch for monitoring and logging. You can use CloudWatch to monitor the performance of your clusters and collect logs from your applications running on Kubernetes.
7. **Integration with Other AWS Services**: Amazon EKS integrates with other AWS services such as Amazon EBS, Amazon S3, and AWS Identity and Access Management (IAM), allowing you to leverage the full capabilities of the AWS ecosystem in your Kubernetes clusters.

5.6 Dev Tools;

For Developing this project various tools had used, below listed had been used

5.6.1 VS Code:

Visual Studio Code (famously known as VS Code) is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

Features

VS Code supports a wide array of programming languages from Java, C++, and Python to CSS, Go, and Dockerfile. Moreover, VS Code allows you to add on and even creating new extensions including code linters, debuggers, and cloud and web development support.

The VS Code user interface allows for a lot of interaction compared to other text editors. To

simplify user experience, VS Code is divided into five main regions:

- The activity bar
- The side bar
- Editor groups
- The panel
- The status bar

For developing Both frontend and backend we used this VS code Editor

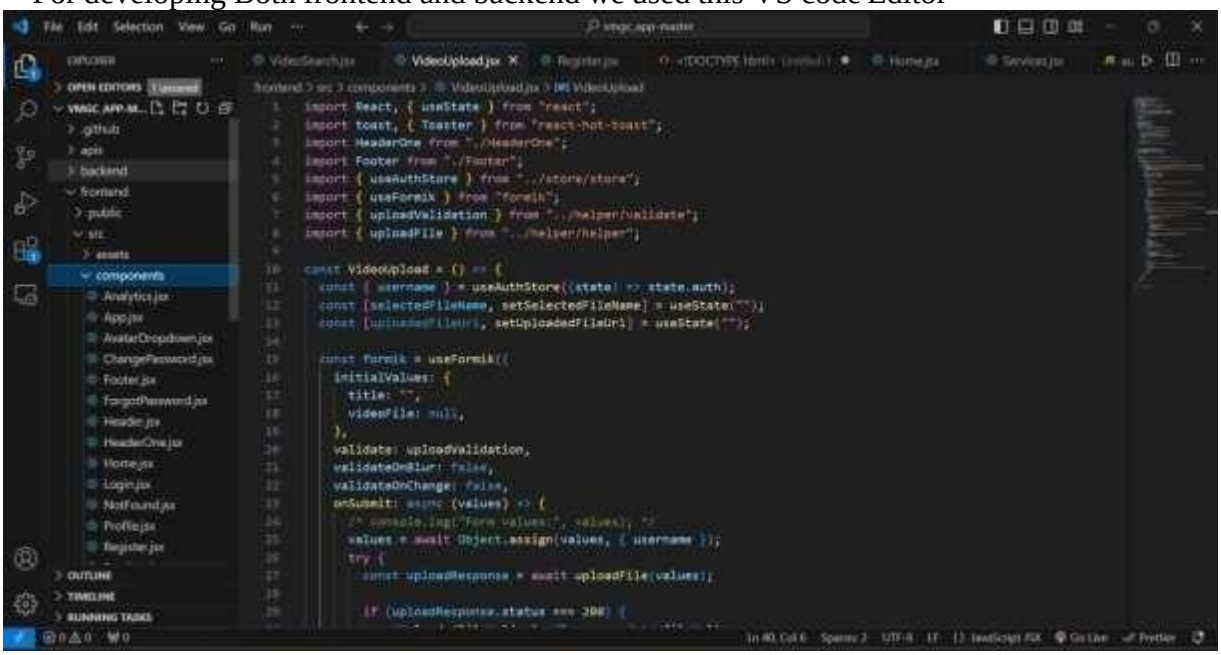


Fig 5.16 VS Code Editor

Above figure illustrates how files are organized in the editor, showcasing its ease of access. Additionally, the editor supports numerous extensions, enhancing its functionality. It also includes an integrated terminal for seamless command-line access.

5.1.1 Postman API:

Postman is a popular API development tool that simplifies the process of creating, testing, and documenting APIs. It provides a user-friendly interface for sending HTTP requests to a server and viewing the responses.

Working with GET Requests

Get requests are used to retrieve information from the given URL. There will be no changes done to the endpoint.

We will use the following URL for all examples in this Postman tutorial https://jsonplaceholder.typicode.com/users

In the workspace

1. Set your HTTP request to GET.
2. In the request URL field, input link
3. Click Send
4. You will see 200 OK Message
5. There should be 10 user results in the body which indicates that your test has run successfully.

Like wise we can check for the POST Request also.

With over 4 million users nowadays, Postman Software has become a tool of choice for the following reasons:

1. Accessibility – To use Postman tool, one would just need to log-in to their own accounts making it easy to access files anytime, anywhere as long as a Postman application is installed on the computer.
2. Use of Collections – Postman lets users create collections for their Postman API calls. Each collection can create subfolders and multiple requests. This helps in organizing your test suites.
3. Collaboration – Collections and environments can be imported or exported making it easy to share files. A direct link can also be used to share collections.
4. Creating Environments – Having multiple environments aids in less repetition of tests as one can use the same collection but for a different environment. This is where parameterization will take place which we will discuss in further lessons.
5. Creation of Tests – Test checkpoints such as verifying for successful HTTP response status can be added to each Postman API calls which help ensure test coverage.
6. Automation Testing – Through the use of the Collection Runner or Newman, tests can be run in multiple iterations saving time for repetitive tests.
7. Debugging – Postman console helps to check what data has been retrieved making it easy to debug tests.

8. Continuous Integration – With its ability to support continuous integration, development practices are maintained.

5.1.2 Docker:

Docker is a powerful tool used for developing, packaging, and deploying applications efficiently. Docker is a container management service. Docker was released in 2013. It is open-source and available for different platforms like Windows, macOS, and Linux. Docker is quickly shipping, testing, and deploying code.

So that it reduces your delay between writing code and running it in production. You can create self-contained environments known as containers. That can run consistently on different platforms.

Containerizing our application:

After Developing the project we follow the below steps

1. **Dockerfile:** Create a **Dockerfile** in the root of your project. This file defines how your application should be packaged into a Docker image. Here's a basic example for a Python application using Django:

```
# Use an official Python runtime as a parent image
FROM python:3.8
# Set the working directory in the container
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt
# Make port 8000 available to the world outside this container
EXPOSE 8000
# Define environment variable
ENV NAME World
# Run app.py when the container launches
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

2. **Build the Docker Image:** Open a terminal and navigate to the directory containing your Dockerfile. Run the following command to build the Docker image:

```
docker build -t myproject .
```

Replace **myproject** with the name you want to give to your Docker image.

3. **Run the Docker Container:** Once the image is built, you can run a Docker container based on this image:

```
docker run -p 8000:8000 myproject
```

This command maps port 8000 of the container to port 8000 on your host machine. Adjust the port mapping as needed for your application.

4. **Verify:** Open a web browser and navigate to **http://localhost:8000** to verify that your application is running correctly inside the Docker container.

5.1.3 kubernetes:

Kubernetes is a container management technology developed in Google lab to manage containerized applications in different kind of environments such as physical, virtual, and cloud infrastructure. It is an open source system which helps in creating and managing containerization of application. This tutorial provides an overview of different kind of features and functionalities of Kubernetes and teaches how to manage the containerized infrastructure and application deployment.

Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters. It is capable of creating container centric infrastructure.

Features of Kubernetes:

- Continues development, integration and deployment
- Containerized infrastructure
- Application-centric management
- Auto-scalable infrastructure

- Environment consistency across development testing and production
- Loosely coupled infrastructure, where each component can act as a separate unit
- Higher density of resource utilization
- Predictable infrastructure which is going to be created

5.1.4 FastAPI:

FastAPI is a modern Python web framework, very efficient in building APIs. FastAPI has been developed by Sebastian Ramirez in Dec. 2018. FastAPI 0.68.0 is the currently available version. The latest version requires Python 3.6 or above. It is one of the fastest web frameworks of Python.

The first step in creating a FastAPI app is to declare the application object of FastAPI class.

```
from fastapi import FastAPI

app = FastAPI()
```

This **app** object is the main point of interaction of the application with the client browser. The uvicorn server uses this object to listen to client's request.

The next step is to create path operation. Path is a URL which when visited by the client invokes visits a mapped URL to one of the HTTP methods, an associated function is to be executed. We need to bind a view function to a URL and the corresponding HTTP method. For example, the **index()** function corresponds to **'/'** path with **'get'** operation.

```
@app.get("/")

async def root():

  return {"message": "Hello World"}
```

The function returns a JSON response, however, it can return **dict, list, str, int,** etc. It can also return Pydantic models.

Save the following code as main.py

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
```

```
async def index():
    return {"message": "Hello World"}
```

Start the uvicorn server by mentioning the file in which the FastAPI application object is instantiated.

```
uvicorn main:app --reload
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [28720]
INFO: Started server process [28722]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

Open the browser and visit http://localhost:/8000. You will see the JSON response in the browser window.

Fig 5. 16 FastAPI response

5.2 Deployment:

To deploy our project on AWS using Docker, Kubernetes, and EC2, we followed these general steps:

1. **Dockerize Our Application**: We created a Dockerfile to define our application's environment and dependencies. This Dockerfile included instructions to build our application into a Docker image.

2. **Build Docker Image:** We built our Docker image using the Docker CLI. This image contained our application and all its dependencies, making it portable and easy to deploy.
3. **Push Docker Image to a Registry:** We pushed our Docker image to a container registry like Docker Hub, AWS ECR, or a private registry. This allowed us to store and manage our Docker images securely.
4. **Set Up Kubernetes Cluster:** We set up a Kubernetes cluster on AWS using Amazon EKS (Elastic Kubernetes Service) or another Kubernetes distribution. This cluster provided the infrastructure to deploy and manage our Docker containers.
5. **Deploy Kubernetes Manifests:** We created Kubernetes manifest files (YAML files) to define the resources needed to deploy our application, such as pods, deployments, services, and ingresses. These files specified how Kubernetes should deploy and manage our Docker containers.
6. **Apply Kubernetes Manifests:** We applied the Kubernetes manifest files using the kubectl apply command. This command instructed Kubernetes to create the necessary resources to deploy our application.
7. **Expose Our Service:** If our application required external access, we created an ingress resource to expose our service to the internet. This allowed users to access our application using a public URL.
8. **Monitor and Scale:** After deploying our application, we monitored its performance and scalability using Kubernetes' built-in monitoring and scaling features. This allowed us to ensure that our application was running smoothly and could handle varying levels of traffic.

Overall, deploying our project on AWS using Docker, Kubernetes, and EC2 involved containerizing our application, setting up a Kubernetes cluster, deploying our application using Kubernetes manifests, and managing its performance and scalability. This approach provided a scalable and reliable environment for running our application on AWS.

CHAPTER 6

RESULTS

Based on the ISRO video Dataset the trained model got an testing accuracy of 67% and training accuracy as 73% .The "Training Loss" score reveals the degree of model error on the particular dataset it was trained on, whereas the "Validation Loss" score denotes the degree of error on unseen, novel data.

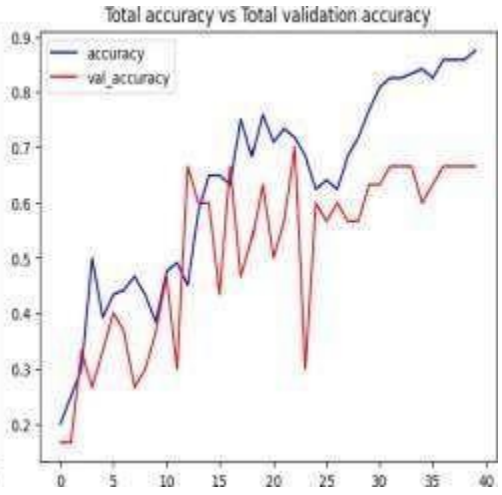


Fig- 6.1: Total Loss vs. Total Validation Loss **Fig-6.2: Total Accuracy vs. Total Accuracy Loss**

A few misclassifications were observed from the above fig-7 such as a graphic video is identified as OutdoorLaunchPad and OutdoorLaunchPad is misclassified as Graphics. This misclassification reduces the accuracy of our LRCN model.

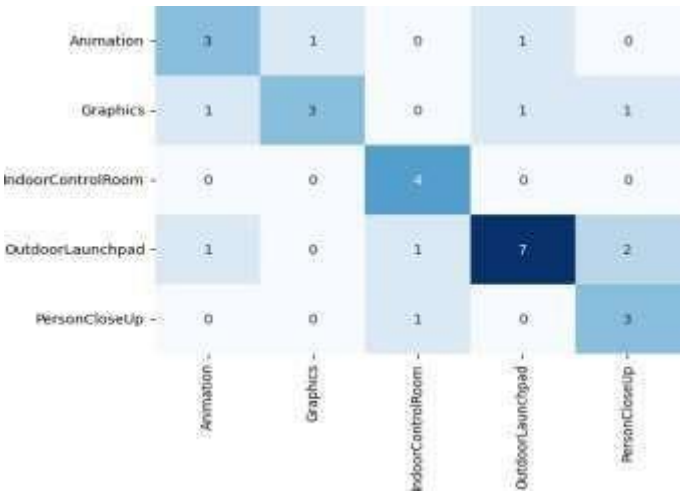


Fig 6.3 Confusion matrix

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

This Project shows the role of neural networks in video classification, underscoring the growing importance of such technologies. Utilizing Anaconda, TensorFlow, and Keras showcases the pivotal role of neural networks in this domain. It's suggested to integrate these technologies into educational programs for broader understanding.

The LRCN model achieved a notable accuracy of 67%. And also seen how annoy algorithm is well suitable for the videos recommendation.

For future research, enhancing the ISRO dataset by including diverse ethnic backgrounds is essential for comprehensive representation. Adapting to such datasets will likely boost model accuracy, advancing activity recognition research. And here we had worked only on five categories so we can extend this model with the few other categories.

Moreover, these models hold immense potential for practical applications. One such application could be their integration into Adult content detection and skipping that particular time line.

By analysing the upcoming video frames from the video we can identify the adult content scenes based on the user interest he may skip the content by one click. conclusion, by expanding and improving the dataset, refining the models, and exploring practical applications, the aim is to continue making significant contributions to the advancements in video category recognition technology.

CHAPTER 8

REFERENCES

[1] Sanketh, S., Thomas, L., Rahul, S., Karpathy, A., George, T., & Li, F. (2014). Convolutional Neural Networks for Large-Scale Video Classification. In the IEEE Computer Vision and Pattern Recognition (CVPR) 2014 Conference Proceedings, Columbus, OH, USA (pp. 1725–1732).

[2] (2015) Ming, L. and Xiaolin, H. Convolutional Neural Network with Recurrent Architecture for Object Identification. Volume 3367–Volume 3375 of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, Boston, Massachusetts, USA.

[3] Dieleman, S., Van Herreweghe, M., Pigou, L., Van Den Oord, A., & Dambre, J. (2016). Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. 126(2-4), 430–439, International Journal of Computer Vision.

[4] Rex, F., and Caleb, A. (2018). A Survey on Action Recognition-Based Video Classification. 7(2.31), 89-93, International Journal of Engineering & Technology.

[5] In 2020, Xia, Huang, and Wang published a book. LSTM-CNN Structure for Identifying Human Activity. Access IEEE, 8, 56855–56866.

[6] Zhong-Qiu, Z., Xindong, W., Peng, Z., & Shou-Tao, X. (2019). A Review on Deep Learning for Object Detection. IEEE Transactions on Learning Systems and Neural Networks, 30 (11), 1–21.

[7] Russo, M. A., Jo, K.-H., & Kurnianggoro, L. (2019). Classification of Sports Videos with Combination of Deep Learning Models and Transfer Learning. In Cox'Bazar, Bangladesh, proceedings of the 2019 International Conference on Electrical, Computer, and Communication Engineering (ECCE), (pp. 1–5).

[8] Hood, S., Sayankar, B., and Baisware, A. (2019). Recent Developments in Video Data-Based Human Action Recognition are reviewed. The 9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19), held in Nagpur, India in 2019, has published its proceedings (1–5).

[9] https://sds-aau.github.io/M3Port19/portfolio/ann/

APPENDIX

Conference Presentation Certificate:


ieeeforum
Paper ID
IE-AICS-CHNN-110324-7821
International Conference on
Advances in Artificial Intelligence and Computer Science
Certificate
This is to certify that Vaddella Mohan has presented a paper entitled "Video Classification and Similar Videos Recommendation System" at the International Conference on Advances in Artificial Intelligence and Computer Science(ICAAICS) held in Chennai, India on 11th March, 2024.
Associated with
UGC Care
Scopus
Industrial Electronics and Electrical Engineers Forum
Industrial Electronics and Electrical Engineers Forum


ieeeforum
Paper ID
IE-AICS-CHNN-110324-7821
International Conference on
Advances in Artificial Intelligence and Computer Science
Certificate
This is to certify that Seeja Sai Kumar has presented a paper entitled "Video Classification and Similar Videos Recommendation System" at the International Conference on Advances in Artificial Intelligence and Computer Science(ICAAICS) held in Chennai, India on 11th March, 2024.
Associated with
UGC Care
Scopus
Industrial Electronics and Electrical Engineers Forum
Industrial Electronics and Electrical Engineers Forum


ieeeforum
Paper ID
IE-AICS-CHNN-110324-7821
International Conference on
Advances in Artificial Intelligence and Computer Science
Certificate
This is to certify that Thota Hemanth Krishna has presented a paper entitled "Video Classification and Similar Videos Recommendation System" at the International Conference on Advances in Artificial Intelligence and Computer Science(ICAAICS) held in Chennai, India on 11th March, 2024.
Associated with
UGC Care
Scopus
Coordinator
Industrial Electronics and Electrical Engineers Forum
Chairman
Industrial Electronics and Electrical Engineers Forum


ieeeforum
Paper ID
IE-AICS-CHNN-110324-7821
International Conference on
Advances in Artificial Intelligence and Computer Science
Certificate
This is to certify that Kakarla Bhargav has presented a paper entitled "Video Classification and Similar Videos Recommendation System" at the International Conference on Advances in Artificial Intelligence and Computer Science(ICAAICS) held in Chennai, India on 11th March, 2024.
Associated with
UGC Care
Scopus
Coordinator
Industrial Electronics and Electrical Engineers Forum
Chairman
Industrial Electronics and Electrical Engineers Forum


Proceedings of
INTERNATIONAL CONFERENCE
CHENNAI, INDIA
11th MARCH 2024
Organized by
ieeeforum
(Industrial Electronics and Electrical Engineers Forum)
Co-Organized by
Institute of
Research and Journals
Integrated Research and Innovation
In Association with
ISSRD
International Society for Scientific Research and Development
iFEARP World
irfsr
International Research Forum for Scientific Research

VIDEO CLASSIFICATION AND SIMILAR VIDEOS RECOMMENDATION SYSTEM

[1]VADDELLAMOHAN, [2]SEELASAIKUMAR, [3]THOTAHEMANTHKRISHNA, [4]KAKARLABHARGAV, [5]K.DEEPIKA

[1,2,3,4]Student,Dept.ofCSE(AI &ML),VasireddyVenkatadriInstituteofTechnology,AndhraPradesh,India.
[5]Assistant Professor, Dept. of CSE (AI &ML), VasireddyVenkatadri Institute of Technology, Andhra Pradesh,India.
E-mail: [1]mohanvaddella@gmail.com, [2]sai680513@gmail.com, [3]khemanth1107@gmail.com,
[4]kakarlabhargav220@gmail.com,[5]deepikak@vvit.net

Abstract - This work's main goal is to categorize videos, by extracting video frames and generating feature vectors to identify action sequences. For improved spatial and temporal properties, a suggested deep neural network integrates Long Short Term Memory and Convolutional Neural Network models. By removing characteristics from frames and taking into account variables like frame width, height, and video sequence duration, video categorization is accomplished. Beginning with a time-distributed 2D convolutional layer, Following the initial convolutional layer, time-distributed max-pooling layers and dropout layers are strategically inserted. Additional convolutional layers with increasing filter sizes further enhance the model's capacity for feature extraction. A flatten layer prepares the output for temporal analysis by a Long Short-Term Memory (LSTM) layer, which captures intricate temporal dependencies within the video sequences. The architecture culminates in a dense layer equipped with a softmax activation function, generating predictions for various classes. The ISRO Dataset is used for video classification. The results show that in terms of prediction accuracy, the LRCN methodology performs better than both ConvLSTM and conventional CNN methods. Furthermore, the proposed method offers improved temporal and spatial stream identification accuracy. The findings show that a 67% prediction of the action sequence's probability is made. Along with Classification, we also Recommend few videos based upon the input videos or images.

Keywords - Convolutional Neural Network (CNN), Long-term Recurrent Convolutional Network (LRCN), Long Short-Term Memory (LSTM).

I. INTRODUCTION

The rise of manufacturing in the digital age has reached unprecedented levels. Platforms like YouTube produce great videos and regular in-house content. They are often faced with the challenge of sorting through the amount of digital content. The research aims to address the challenge of developing classification models using Long-Term Recurrent Convolutional Networks (LRCN). It tries to better categorize videos, providing users with introductions and similar content. Our goal is to simplify the recovery process and provide a more efficient way to communicate large-scale digital video.

Our video classification system focuses only on content developed by the Indian Space Research Organization (ISRO). ISRO's video content is varied, including satellite launches, mission updates, and behind-the-scenes footage. Thus, a systematic approach to organizing and categorizing these videos is needed. Our research seeks to use LRCN and provide a solution to develop a robust model that can accurately classify ISRO-related videos. This targeted classification system is expected to enhance the accessibility and searchability of ISRO's extensive video archives and feed users seeking organization-related content the specific needs of the Organization.

By incorporating LRCN into our video classification model, we aim to achieve higher accuracy in discriminating between different content classifiers. This approach promises to help develop an effective monitoring system not only in ISRO but also as a comprehensive tool for video content processing in various industries. Through this research, we provide a seamless experience and easy-to-use in an ever-expanding industries. We want to use it so that video classification models can evolve.

Apart from Classification, we are also providing similar video for the user by using the Annoy Algorithm, which is like a recommendation system. This makes the user easier to look up at similar content as he/she was interested in. Many of the platforms like YouTube and Netflix are famous for their movie Recommendation Systems. Here with this Annoy Algorithm, we are providing the similar videos, we discuss it in deep in a while.

II. RELATEDWORK

1. Several studies have been conducted on video segmentation using visual features. Andrej Karpathy and others. [1] used a new dataset of 1 million YouTube videos in 487 categories to conduct a comprehensive empirical study of convolutional neural networks (CNNs) for large video classifications. The authors investigated different approaches enhanced the ability of CNN to capture local spatial and temporal information in time domain and proposed a multiresolution architecture as a possible way to speed up training. The performance of the proposed spatio-temporal network shows a

significant improvement from 55.3% to 63.9% compared to the robust feature-based baseline models but from 59.3% to 60.9% better performance compared to the single-frame model of the Improvement. Transfer learning was also applied to the UCF101 Action Recognition dataset.

2. The CNN (RCNN) iterative model was proposed by Ming Liang et al. [2] for object detection by adding recursive combinations to each convertible layer. Even though the input parameters are fixed, the behavior of RCNN clusters changes over time, so that the behavior of neighboring clusters influences the behavior of each cluster. This characteristic provides the ability of the model to provide information about context will be included, which is important for improving the discovery process. CIFAR-10, CIFAR-100, MNIST, and SVHN object recognition datasets were used to evaluate the model. On every dataset that is taken into consideration, it is found that RCNN performs better than cutting-edge models with fewer trainable parameters. These results demonstrate the benefits of conventional object recognition programs over sophisticated programs exclusively.

3. For gesture recognition in videos, Pigou, L. Et al. [3] presented a novel give-up-to-quit trainable neural network architecture that combines bidirectional recurrence and temporal convolutions with deep systems. The significance of repetition and its necessity, which includes temporal convolutions that result in significant profits, have been examined by the writers. We tested several approaches with the newly discovered outcomes from the Montalbano gesture reputation dataset.

4. Caleb Andrew et al. conducted surveys on the algorithms used in video classification techniques [4] to see whether the technique is more effective at predicting gestures or motions and can be applied to the categorization of action films. In deep learning models for visual sequence modeling, LSTM is crucial for the recognition of gestures and actions. Even though there are numerous movie categories, emerging methods frequently combine well-known categories to increase classification accuracy

5. A revolutionary neural network incorporating advanced convolutional layers and the remarkable long short-term memory (LSTM) was brought forth by Xia, K., et al. [5]. In order to gauge the true potential of this model, three publicly accessible datasets, namely UCI, WISDM, and OPPORTUNITY, were employed. Astonishingly, the model boasted an accuracy of 95.78 percent in the dataset from UCI-HAR, 95.85 percent in the dataset from WISDM, and a notable 92.63 percent in the OPPORTUNITY dataset.

III. DATASET

The dataset we have been working with is ISRO dataset. It contains videos divided into five different categories: Animation, Person Close Up, Graphics, Outdoor Launch Pad, and Indoor Control Room and the dataset contains 150 videos, with 30 videos in each category. The average length of the videos in the dataset is about 5 seconds.

Fig-1:ISRO videos: 5 videos for each category

The animation part consists of animated content, with visual effects and conceptual elements. The Person Close Up range focuses primarily on individuals in close-up shots. Graphics include videos that incorporate computer-generated images or graphics, which provide complex data or mental representations. The Outdoor Launch Pad section contains outdoor launches, spacecraft, rockets, or activities related to space exploration, providing insight into the fun side of space exploration. Finally, the Indoor Control Room section contains control room videos, monitoring stations, or similar furniture with Shan and the behind-the-scenes side of his scientific endeavors.

TotalActions	5Categories
TotalVideos	150
VideosperGroup	30
MinimumDuration	4sec
MaximumDuration	23sec
FramesRate	6Per second

Table1:Detailed view of Input

IV. PROPOSEDAPPROACH

The entire thing in this is, First the input video is converted into frames with a rate of 5 per second which is crucial for the Identifying of the continuous changes in the video. Now the each frame must follow the preprocessing stage, and the result of the preprocessing step is then forwarded to the LRCN Architecture and the architecture will be discussed in below. The result of the LRCN model is an vector representation of the input video then used for the

annoy algorithm which will identify by plotting the input video with the pre-trained models. So when we plot we can identify the nearest plotting are the Similar videos as the input video.

4.1 DatasetPreprocessing:

The initial step will be to import the ISRO video dataset, which will then be split into three sets: a training set, a test set, and a validation set, with ratios of 0.8, 0.1, and 0.1, respectively. During the pre-processing stage, a frame is first extracted from the video, then it is shrunk to measure 64 by 64, and finally, the pixel values are normalized. The training set will be put into the CNN and LRCN models after the dataset has been pre-processed. The model will then be tested using test data. Lastly, a comparison of accuracy will determine whether the model performs better.

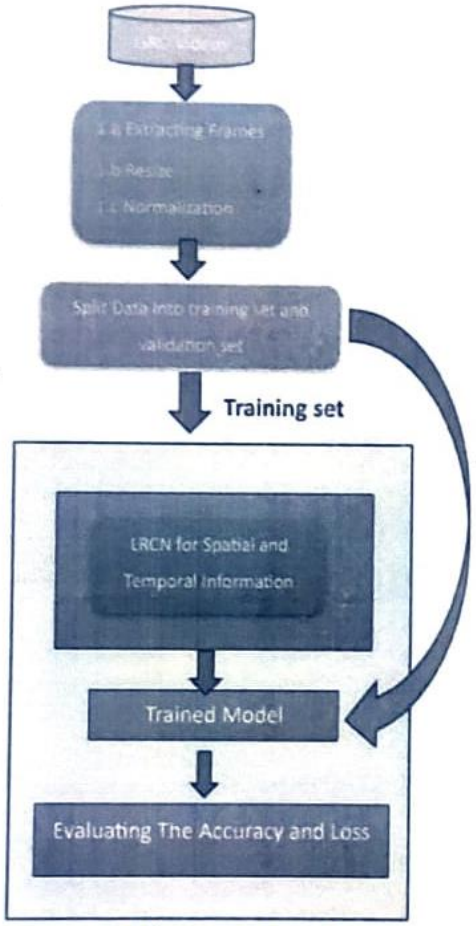


Fig-2:Illustration of Proposed Model

4.2 The Proposed Long-term Recurrent Convolutional Network For Video Classification

A sophisticated deep learning algorithm, Long-Term Recurrent Convolutional Networks (LRCN), can extract sequences of scene features from sequences of images. The versatility of the LRCN, which was first introduced [6], has been demonstrated in applications such as event recognition, image priming, and video annotation processing. This is possible because LRCN offers the flexibility of many recurrent neural networks (RNNs) are implemented. LRCN consists of two main components. Convolutional Neural Network (CNN), and RNN.

The main goal of a CNN encoder is to extract spatial features and transform them into one-dimensional vectors. The resulting vector is then used as input to the RNN encoder to make temporal dynamics decisions. Using shared loads allows the network to handle multiple frames simultaneously. The encoder-decoder design provides greater flexibility in the choice of architecture for CNNs and RNNs, since they operate independently and, because the returning neuron can receive input vectors of variable length, can be layered. The final output of any planar CNN architecture has served as input, regardless of its dimensions.

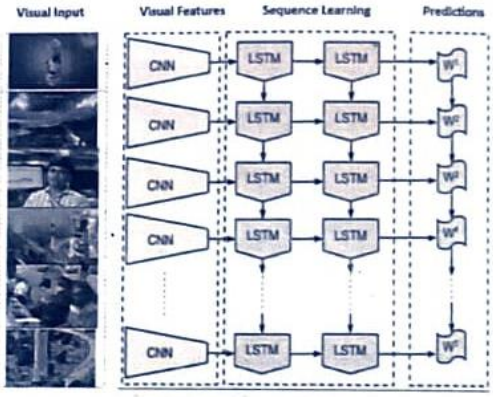


Fig-3: LRCN Architecture

4.3 Model size and Parameters of LRCN:

The Long-term Recurrent Convolutional Network (LRCN) combines Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), emphasizing Long Short-Term Memory (LSTM) networks. The version uses time-distributed CNN layers to process video frames. These CNN layers have 4 blocks, each with a Conv2D layer, BatchNormalization, MaxPooling2D, and Dropout layer. The Conv2D layers have filters starting from 32 to 128, the usage of a 3x3 kernel. MaxPooling2D reduces spatial dimensions, whilst Dropout layers counteract overfitting. After processing through CNNs, the output is flattened and exceeded to an LSTM layer with 64 units, ideal for coping with frame sequences through time. The version ends with a Dense layer using softmax activation for category, matching the number of trouble instructions.

V. ANNOY FOR SIMILAR VIDEOS RECOMMENDATION

A C++ library called ANNOY (Approximate Nearest Neighbour's Oh Yeah) has Python bindings that allow it to find points in space that are near a query point, such as a particular point of interest. Additionally, it builds sizable memory-mapped data structures that allow multiple processes to share the same set of data. Finding the points that are closest to a particular instance is the goal of the nearest neighbour search, a similarity issue.The nearest neighbours search determines the instance q that is closest to an instance p under a measurement function, given a set of n examples P = {p1, ..., pn} in some metric space X. Nearest neighbours search computes the closest instance q to an instance p under some measurement function.

The goal of the approximation closest neighbour search is to expedite the computation of the nearest neighbours, even while the exact method cannot be improved. Encouraging errors is the only way to speed up the calculation. Random projections and trees are the basis of the ANNOY (Approximate NearestNeighbour's Oh Yeah) algorithm. When using ANNOY, datasets with up to 100–1000 dense dimensions can be searched. The nearest neighbours are determined by dividing the collection of points in half. Until each set has k elements, this process is repeated. Generally speaking, k should be about 100 (see illustration below).

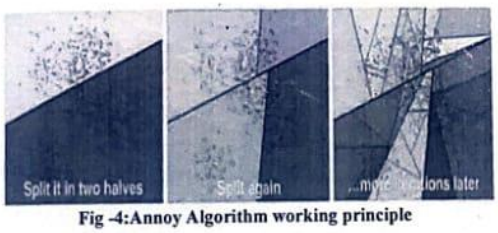


Fig -4:Annoy Algorithm working principle

Points whose distance from the query is at most c times the distance from the query to its nearest points can be returned using an approximate nearest neighbour search method. This method has the advantage that an estimated nearest neighbour is frequently nearly as good as an exact one. Specifically, minor variations in the distance should not be significant if the measure of distance effectively represents the concept of user quality. The output of an ANN's categorization indicates a class membership. Based on the majority vote of its neighbours, an object is classified. The object is classified into the class that its k closest neighbours share the greatest amount of. An integer k that is positive and usually tiny.

This is our ISRO dataset. The 150 films will also be plotted in the same way as the figure above, and each time a new input video data point is pointed, the annoy algorithm will produce 5 data points that fall under the same region according to the k value, for example, if k=5, then the annoy algorithm will provide 5 data points which fell under the same region.

VI. RESULTS

Throughout this research work, we work on one model is being evaluated on the previous mentioned ISRO dataset. We take care about the dataset avoiding the any sort of merging during the experiment process. The dataset was divided in a way that 80% was used means out of 150 means 120 videos had been used for the training of the model and the remaining 20% means 30 videos had been used for the testing the model. During the model initial phase, the validation split parameter was set to 0.2, representing that 20% of the training data was utilized for validationpurpose.

The results of our LRCN model described below Table Results represents that the LRCN model exhibits most impressive accuracy achieved is 67% on the new Video ISRO dataset. The "Training Loss" score reveals the degree of model error on the particular dataset it was trained on, whereas the "Validation Loss" score denotes the dégree of error on unseen, novel data. The fundamental purpose of training a machine learning model is to minimize its loss, which entails reducing the amount of error committed by the model.

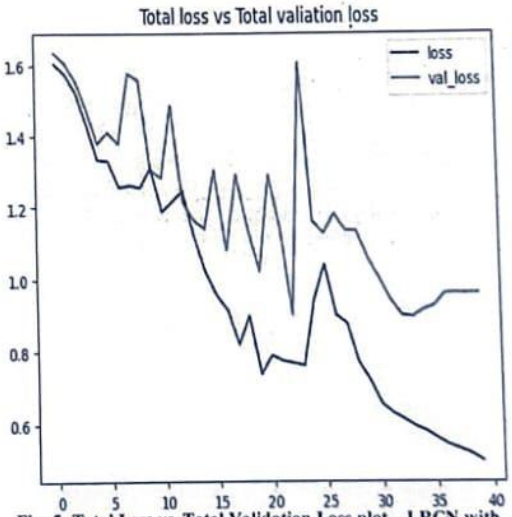


Fig- 5: Total Loss vs. Total Validation Loss plot – LRCN with the ISRO dataset.

"Training Accuracy" shows the model's performance on its training data, while "Validation Accuracy" indicates its capability on new data. A high validation accuracy is essential for effective model generalization, avoiding mere data memorization.

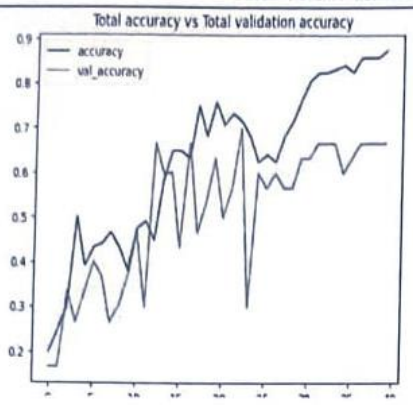


Fig-6: Total Accuracy vs. Total Accuracy Loss plot – LRCNwiththeISROdataset.

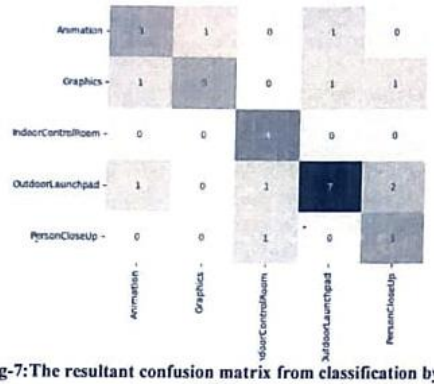


Fig-7:The resultant confusion matrix from classification by LRCN

A few misclassifications were observed from the above fig-7 such as a graphic video is identified as Outdoor Launch Pad and Outdoor Launch Pad is misclassified as Graphics. This misclassification reduces the accuracy of our LRCN model. The below fig-8 is an output where the given input video had been identified their belonging category and we can see the category label on the left corner with red text respectively.

Fig-8: Different Video Category Recognition Results.

VII. CONCLUSION

This study emphasizes the role of neural networks in video classification, underscoring the growing importance of such technologies. Utilizing Anaconda, TensorFlow, and Keras showcases the pivotal role of neural networks in this domain. It's suggested to integrate these technologies into educational programs for broader understanding. The LRCN model achieved a notable accuracy of 67%. And also seen how anoyalgorithm is well suitable for the videos recommendation. For future research, enhancing the ISRO dataset by including diverse ethnic backgrounds is essential for comprehensive representation. Adapting to such datasets will likely boost model accuracy, advancing activity recognition research. And here we had worked only on five categories so we can extend this model with the few other categories. Moreover, these models hold immense potential for practical applications. One such application could be their integration into Adult content detection and skipping that particular timeline. By analyzing the upcoming video frames from the video we can identify the adult content scenes based on the user interest he may skip the content by one click. In conclusion, by expanding and improving the dataset, refining the models, and exploring practical applications, the aim is to continue making significant contributions to the advancements in video category recognition technology.

REFERENCE

[1] Sanketh, S., Thomas, L., Rahul, S., Karpathy, A., George, T., & Li, F. (2014). Convolutional Neural Networks for Large-Scale Video Classification. In the IEEE Computer Vision and Pattern Recognition (CVPR) 2014 Conference Proceedings, Columbus, OH, USA (pp. 1725–1732).

[2] (2015) Ming, L. and Xiaolin, H. Convolutional Neural Network with Recurrent Architecture for Object Identification. Volume 3367–Volume 3375 of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, Boston, Massachusetts, USA.

[3] Dieleman, S., Van Herreweghe, M., Pigou, L., Van Den Oord, A., &Dambre, J. (2016). Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. 126(2-4), 430–439, International Journal of Computer Vision.

[4] Rex, F., and Caleb, A. (2018). A Survey on Action Recognition-Based Video Classification. 7(2.31), 89- 93, International Journal of Engineering & Technology.

[5] In 2020, Xia, Huang, and Wang published a book. LSTM-CNN Structure for Identifying Human Activity. Access IEEE, 8, 56855–56866.

[6] Mehr, H. D., &Polat, H. (2019). Human Activity Recognition in Smart Home With Deep Learning Approach. https://doi.org/10.1109/sgcf.2019.8782290

[7] Zhong-Qiu, Z., Xindong, W., Peng, Z., &Shou-Tao, X. (2019). A Review on Deep Learning for Object Detection. IEEE Transactions on Learning Systems and Neural Networks, 30(11), 1–21.

[8] Russo, M. A., Jo, K.-H., &Kurnianggoro, L. (2019). Classification of Sports Videos with Combination of Deep Learning Models and Transfer Learning. In Cox'Bazar, Bangladesh, proceedings of the 2019 International Conference on Electrical, Computer, and Communication

Engineering (ECCE), (pp. 1–5).

[9] Hood, S., Sayankar, B., and Baisware, A. (2019). Recent Developments in Video Data-Based Human Action Recognition are reviewed. The 9th International Conference on Emerging Trends in Engineering and Technology-Signal and Information Processing (ICETET-SIP-19), held in Nagpur, India in 2019, has published its proceedings (1–5).

[10] Mengyun L. Deep Learning-Based Video Classification Technology. In Parallel and Distributed Systems (ISPDS), 2020 International Conference on Information Science, Xi'an, China, Proceedings (pp. 154-157).

[11] Deb, K., Dhar, P. K., Sarma, M. S., &Koshiba, T. (2021). Deep Learning Method for Classifying Traditional Sports Videos from Bangladesh. 11(5) Applied Sciences, 2149.

[12] In 2020, Roubleh, A. A., and Khalifa, O., O. Deep learning for video-based human activity recognition. The 7th International Conference on Electronic Devices, Systems, and Applications (ICEDSA2020) was held in Shah Alam, Malaysia, and the proceedings are available on pages 020023-1 through 020023-8

★★★