CONTENT BASED VIDEO RETRIEVAL USING TRANSFER LEARNING

Submitted in partial fulfilment of the requirements for the award of the

degree of

MASTERS OF COMPUTER APPLICATIONS

2023-2025

By

KARTHIKA R (38223040)
SARAGA S (38223057)
SHAHBAZ AHMAD (38223058)

DEPARTMENT OF COMPUTER APPLICATIONS

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGYCOCHIN-22

NOVEMBER 2024

DEPARTMENT OF COMPUTER APPLICATIONS
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
COCHIN-22

BONAFIDE CERTIFICATE

This is to certify that the project report **“CONTENT BASED VIDEO RETRIEVAL USING TRANSFER LEARNING”** submitted in partial fulfilmentof the requirements for the award of the Degree of Master of Computer Applications is a recordof bonafide work done by **KARTHIKA R (38223040), SARAGA S (38223057), SHAHBAZ AHMAD(38223058),** during the period from July 2024 to November 2024 of their study in the Department Of Computer Applications at Cochin University Of Science And Technology,under my supervision and guidance.

Head of the Department	**Signature of the Examiner**
Department Of Computer Applications	Department Of Computer Applications
CUSAT	CUSAT

DEPARTMENT OF COMPUTER APPLICATIONS
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY COCHIN-22

CERTIFICATE

This is to certify that the report entitled "**CONTENT BASED VIDEO RETRIEVAL USING TRANSFER LEARNING** '' submitted in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications is a record of bonafide work done by **KARTHIKA R (38223040), SARAGA S (38223057), SHAHBAZ AHMAD(38223058),** during the period from 06/07/2024 to 12/11/2024 of their study in the Department of Computer Applications under my supervision and guidance.

Signature of the Guide
Department Of Computer Applications
CUSAT

DECLARATION

We hereby declare that the report entitled **“CONTENT BASED VIDEO RETRIEVAL USING TRANSFER LEARNING ”** in partial fulfilment of the requirements for the award of theDegree of Master of Computer Applications is a record of bonafide work done by us during the period from 06/07/2024 to 12/11/2024 under the supervision and guidance of **Dr.ARUN K S**, Department of Computer Applications, Cochin University of Science andTechnology, Cochin-22.

Place: CUSAT

KARTHIKA R (38223040)

SARAGA S (38223057)

SHAHBAZ AHMAD (38223058)

ACKNOWLEDGEMENT

We thank God almighty for bestowing his blessings upon us to proceed and complete the work. We have great pleasure in acknowledging the help given by various individuals throughout the project work. This project itself is an acknowledgement to the inspiration, drive and technical assistance contributed by many individuals.

We express our sincere and heartfelt gratitude to Dr. M V Judy, Head of the Department and faculty members for being helpful and cooperative during the period of the project.

We also express our deep gratitude to our guide Dr. Arun K S, Assistant Professor in the Department of Computer Applications for her valuable guidance and timely suggestionsthat helped in the completion of this project in time.

We extend our sincere thanks to all the teaching and non-teaching staff of the Department of Computer Applications for providing the necessary facilities and help. Without the support of any of them this project would not have been a reality.

Place: CUSAT
Date:11/11/2024

KARTHIKA R (38223040),
SARAGA S (38223057),
SHAHBAZ AHMAD (38223058)

TABLE OF CONTENTS

ABSTRACT

This is basically a content-based video retrieval system identifying similar videos in a dataset by analyzing visual features extracted from each video. The outline of the main components follows below:

Feature Extraction:

Applying a pre-trained ResNet50 model without classification layers, the system retrieves a feature vector for every frame of the video to describe its visual content. The system aggregates these features into one comprehensive feature vector representing all the visual characteristics in the video.

Feature Storage and Retrieval:

The feature vectors along with their corresponding video file paths will be stored in a compressed file. This will enable quicker access to the pre-computed features for similarity search when needed, hence bypassing the iteration process.

Similarity matching:

The system will use the similarity matching concept through which a cosine similarity between the feature vector of an input video and the feature vectors corresponding to the stored video is computed. It then returns back the video ranked by the system as the top k based on the similarity scores.

Performance Evaluation:

Now, to measure the retrieval accuracy, precision and recall are used. Precision is the number of relevant videos retrieved, whereas the sensitivity of a video retrieval system is established by measuring recall, which denotes how many of the relevant videos within the dataset were retrieved.

Pipeline Workflow:

The system can run in a streamlined workflow that reads video paths, labels and features from a CSV file, computes or loads features in addition to doing similarity retrievals. It will return a ranked list of similar videos along with performance metrics such that users can validate the effectiveness of the system

INTRODUCTION

In the near past, and with the vast growth of video content across a wide range of communication channels, there is a growing need for effective methods of video searching and retrieval based on their contents. Keyword searching might not represent the video's visual and contextual information, especially when the videos do not provide structured metadata and descriptions. Alternative approaches exist that would allow for similarity-based searches of video content through content-based video retrieval systems, which analyze the video's visual features.

In this work, thus, we design a content-based video retrieval system based on the identification of similar videos in a dataset by extracting and comparing visual features. We exploit the unique feature representations generated from every video by a pre-trained deep learning model, ResNet50. These feature representations are then utilized in the evaluation of similarity between pairs of videos. The methodology of the project involves extraction of frame level features with averaging in order to achieve a single representation per video. Obtained feature vectors are stored for efficient retrieval. Ranking videos in the dataset based on how similar they are to an input video is achieved through cosine similarity.

This system shows how deep learning models changed video search and retrieval to enable similarity-based searches, as deep learning models do not rely on added tags or descriptions. The project also includes a simple evaluation mechanism to calculate the precision and the recall of the system's retrieval performance. Thus, the approach establishes a basis for more complex content-based video retrieval applications in media archives and video recommendation engines across different domain

THEORETICAL BACKGROUND

This content-based video retrieval project is based on key deep learning principles; foremost, CNNs and transfer learning for the development of a system that identifies visually similar videos. This system focuses the content of the videos instead of its metadata. Thus, deep feature extraction and similarity measurement techniques are used in order to be efficient in retrieval.

1.Deep Learning and Convolutional Neural Networks (CNNs)

Deep learning, refers to models that consist of many layers or "deep" structures, capable of learning complex representations of data. Among the primary architectures applied to computer vision, CNNs are particularly effective in analyzing image and video data because of the possibility of capturing spatial hierarchies through convolutional layers.

- **CNN Architecture:** CNNs involve many layers consisting of convolutional and/or subsampling layers with filters (or kernels) scanning the input image and learning spatial features at different levels. The earlier layers focus on detecting straightforward features like edges; deeper layers identify more complex patterns and objects.

- **Feature Maps:** In each convolutional layer, a set of filters generates "feature maps" which highlight certain characteristics, such as shapes or textures or a pattern that represent multi-level understanding of the image.

- **Role in Video Analysis:** A CNN can be applied frame by frame to video data to extract features that capture the visual characteristics within each frame. If a deep CNN is used-for example, ResNet50-the high-quality representations can be achieved, and our system can recognize and retrieve visually similar videos.

2. Transfer Learning

Transfer learning is a deep learning technique in which a pre-trained model is reused for a new but related task. We do not train a CNN from scratch-it is very resource-intensive and computationally expensive, requiring huge amounts of data and computational power-we instead use a model like ResNet50 pretrained on ImageNet to avail learned features to analyze video frames.

- **ResNet50 Model:** ResNet50 is a 50-layer CNN which, by so-called "residual" connections, makes it efficiently capable of learning even with very many layers to reduce issues like vanishing gradients. This model has been trained on millions of images and has learned generalized features that can transfer pretty well to new datasets.
- **Feature Extraction Layer:** Removing the classification layer would make ResNet50 a feature extractor rather than a classifier. For each frame of video, ResNet50 outputs a feature vector-a numerical representation of key visual characteristics such as texture, shape, and color patterns.

3. Preprocessing and Frame Representation

Videos are also complex data structures in a form of frame sequences. Video data can only be fed into a deep learning framework if proper pre-processing happens. It aims at creating video data for which deep learning-based models can quickly and precisely extract features from the video streams.

- **Frame Sampling:** Not every frame needs to be processed. There are other methods of choosing some important frames periodically which would save space but may give redundant information; thus, sometimes there must be a compromise between these.
- **Resizing:** Resize all frames to 224 x 224 pixels since this is the size of input that ResNet50 calls for. Frame resizing does amount to standardization across frames, so the model runs them effortlessly.

- **Normalization:** This is another important preprocessing step in ResNet50 in relation to the application of the input normalization function to the frames. The pixel values are adjusted to a suitable range for the model. This normalization ensures that input frames match what ResNet50 was trained on originally.
- **Feature Average:** The feature averaging module combines the frame-level feature vectors, extracted after feature extraction from individual frames for obtaining a single vector representing the overall video content. This module decreases data complexity and provides stability in the representation of the overall visual content of the video.

RESULT

This project was a success as deep learning based feature extraction via a pre-trained ResNet50 model was able to pick and collect videos that had the same content from the given dataset. It processed all the frames of each video by creating a particular, unique feature vector that captured every view element. Averaging such frame-level vectors effectively allowed the system to represent each video using a single composite feature vector. The compression storage optimized both the storage efficiency and the speed at which the vectors could be retrieved and hence, it would easily handle large video datasets.

For video retrieval, the system applied cosine similarity to rank videos in order of how similar they are to an input video's features. The demo showed that the system correctly returned videos of similar content, often placing closely related videos high up in the rankings. The application of pre-trained layers from ResNet50 granted the system the ability to capture rich visual patterns of different types of video content, making it adaptable to many visual styles and categories.

Precision and recall metrics were used to evaluate the retrieval accuracy of the system. High precision scores imply that most the top-ranked retrieved videos are highly relevant to the input video. Good recall scores further confirm that the system could retrieve a significant proportion of relevant videos within the dataset. The mentioned metrics are indicative of robustness in obtaining correct results, especially when working on datasets which are organized based on perceptually distinct visual categories.

Reducing both computational and storage requirements by minimizing it to a single averaged feature vector per video-permitted effective real-time retrieval even for larger video datasets. The system is thus highly scalable and competitive for applications in media management, video recommendation platforms, and archival systems.

NEED FOR SYSTEM

Increasing video content in various sectors such as media education surveillance and web service-based online platforms has indeed highlighted the need for effective content-based video retrieval systems traditional search mechanisms by metadata often fail or return partial results due to a lack of tagging with intuitive words and usually miss similar visually similar videos thus this project addresses these limitations by using a deep learning-based content-based video retrieval system that takes full advantage of visual similarity in identifying and retrieving videos such a system uses feature extraction from a pre-trained model in capturing the visual characteristics of videos and organizing them for accurate retrieval this enables users to enjoy higher experiences in media applications faster access to relevant content in large video databases and supports fields such as security and forensics where fast and efficient video analysis becomes very important being content-based proves more scalable and future-proof in comparison to metadata-based systems aligned with the evolution in deep learning and prepared for adaptive and efficient video search capabilities

Background Study

ResNet50

ResNet50 is a deep convolutional neural network(CNN) that was developed by Microsoft Research in 2015. It's a variant of the popular ResNet , which stands for “ Residual Network. ” The “ 50 ” in the name refers to the number of layers in the network, which is 50 layers deep.

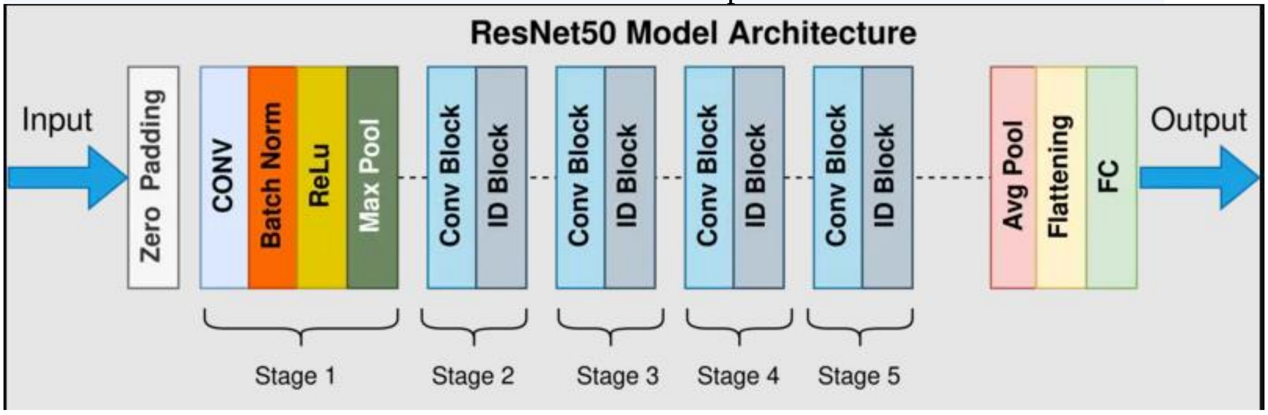
ResNet50 is a important image bracket model that can be trained on large datasets and achieve state- of- the- art results. One of its crucial inventions is the use of residual connections, which allow the network to learn a set of residual functions that collude the input to the asked affair. These residual connections enable the network to learn important deeper infrastructures than was preliminarily possible, without suffering from the problem of deleting.

The architecture of ResNet50 is divided into four main corridor the convolutional layers, the identity block, the convolutional block, and the completely connected layers. The convolutional layers are responsible for rooting features from the input image, while the identity block and convolutional block are responsible for processing and transubstantiating these features. Eventually, the completely connected layers are used to make the final bracket.

The convolutional layers in ResNet50 correspond of several convolutional layers followed by batch normalization and ReLU activation. These layers are responsible for rooting features from the input image, similar as edges, textures, and shapes. The convolutional layers are followed by maximum pooling layers, which reduce the spatial confines of the point maps while conserving the most important features.

The identity block and convolutional block are the crucial structure blocks of ResNet50. The identity block is a simple block that passes the input through a series of convolutional layers and adds the input back to the affair. This allows the network to learn residual functions that collude the input to the asked affair. The convolutional block is analogous to the identity block, but with the addition of a 1x1 convolutional subcaste that's used to reduce the number of pollutants before the 3x3 convolutional subcaste.

The final part of ResNet50 is the completely connected layers. These layers are responsible for making the final bracket. The affair of the final completely connected subcaste is fed into a softmax activation function to produce the final class chances.

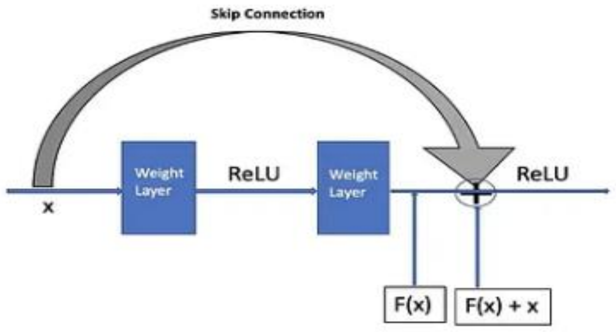


```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, Sequential
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import pathlib


dataset =
"https://storage.googleapis.com/download.tensorflow.org/example_images/flower_
photos.tgz"
directory = tf.keras.utils.get_file('flower_photos', origin=dataset,
untar=True)
data_directory = pathlib.Path(directory)


# Define the image size and batch size
img_height, img_width = 180, 180
batch_size = 32


# Create the training and validation datasets
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)
validation_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)



# Plot some sample images from the dataset
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(6):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(classnames[labels[i]])
        plt.axis("off")




# Create the ResNet50 model and set the layers to be non-trainable
resnet_model = Sequential()
pretrained_model = tf.keras.applications.ResNet50(include_top=False,
input_shape=(img_height, img_width, 3),pooling='avg',weights='imagenet')

for layer in pretrained_model.layers:
    layer.trainable = False
resnet_model.add(pretrained_model)



# Add fully connected layers for classification
resnet_model.add(layers.Flatten())
resnet_model.add(layers.Dense(512, activation='relu'))
resnet_model.add(layers.Dense(5, activation='softmax'))



# Compile and train the model
resnet_model.compile(optimizer=Adam(lr=0.001),
loss='categorical_crossentropy', metrics=['accuracy'])
history = resnet_model.fit(train_ds, validation_data=validation_ds, epochs=10)
```

Continuing with Evaluation and model inference:

```
# Evaluate the ResNet-50 model
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['validation_accuracy'])
plt.axis(ymin=0.4, ymax=1)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()



# Model Inference
# Preprocess the sample image
import cv2
image = cv2.imread(str(roses[0]))
image_resized = cv2.resize(image, (img_height, img_width))
image = np.expand_dims(image_resized, axis=0)


# Make predictions
image_pred = resnet_model.predict(image)


# Produce a human-readable output label
image_output_class = class_names[np.argmax(image_pred)]
print("The predicted class is", image_output_class)
```

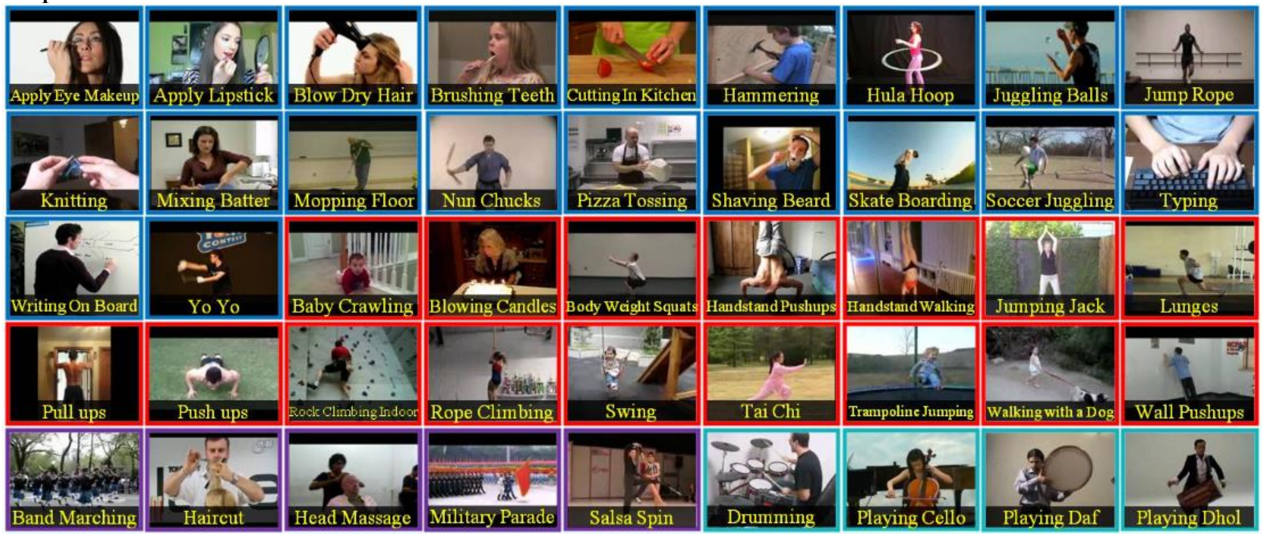
How it solved the problem of vanishing gradients:

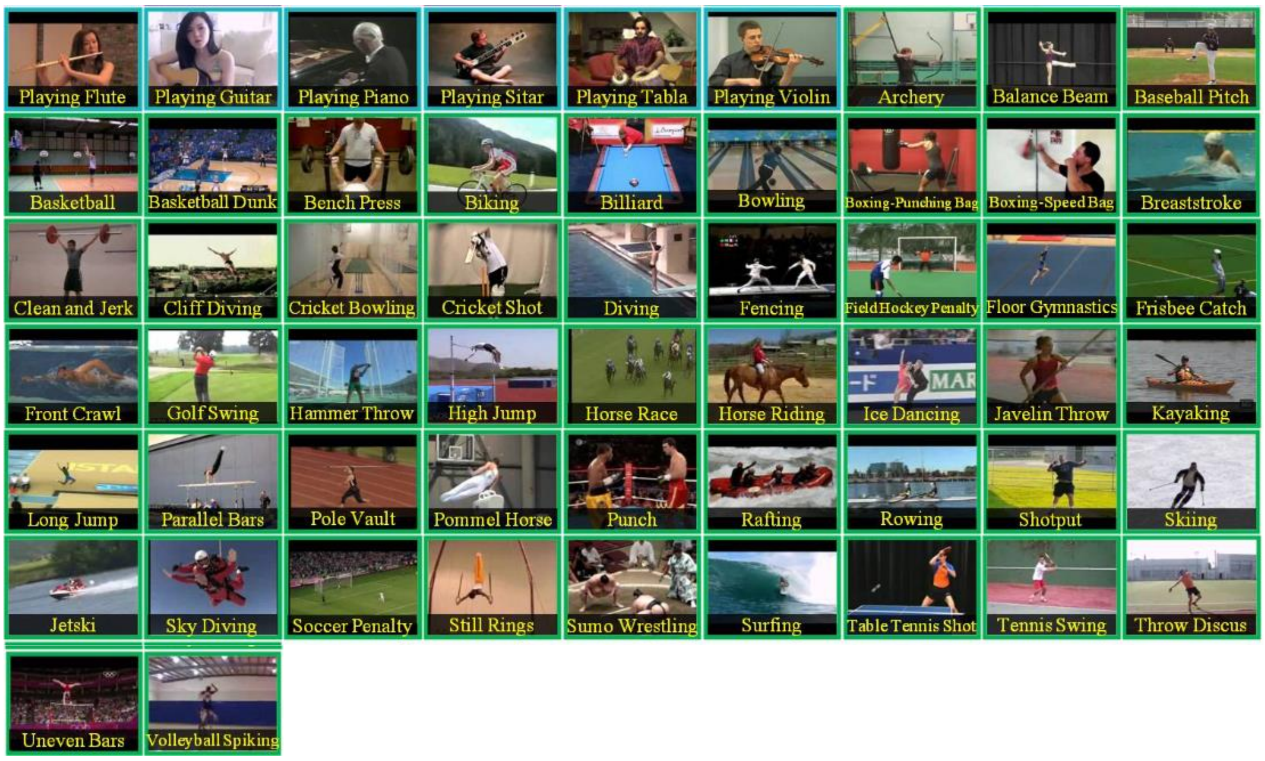


Skip Connections

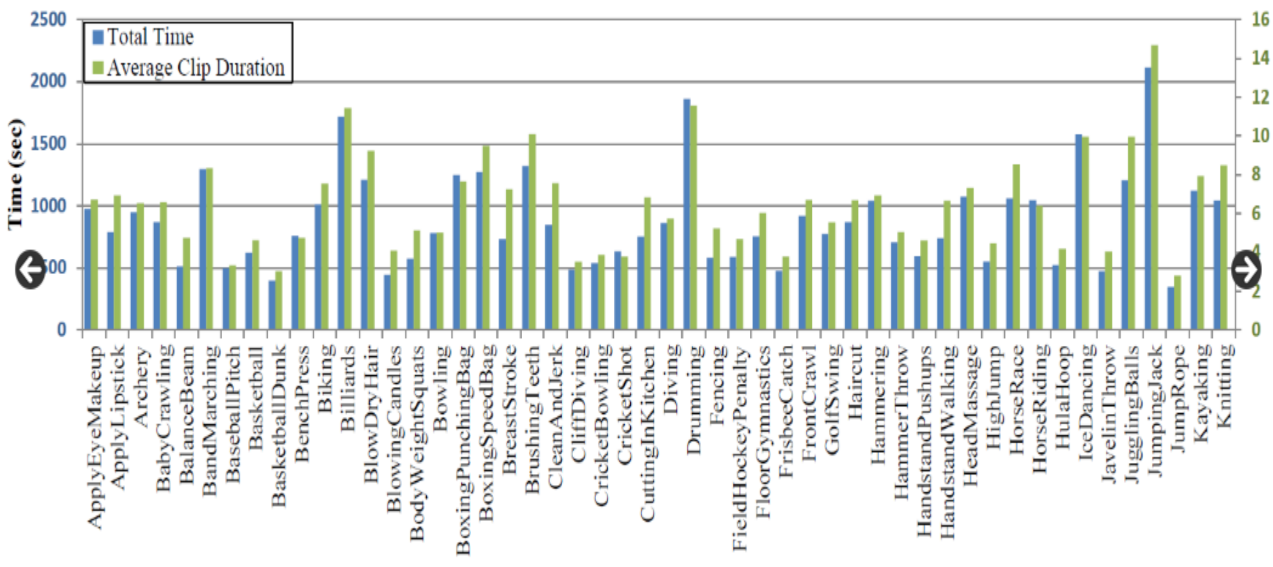
UCF101

UCF101 is an action recognition data set of realistic action videotape, collected from YouTube, having 101 action orders. This data set is an extension of UCF50 data set which has 50 action orders.
With 13320 videotape from 101 action orders, UCF101 gives the largest diversity in terms of conduct and with the presence of large variations in camera stir, object appearance and disguise, object scale, standpoint, cluttered background, illumination conditions, etc, it's the most grueling data set to date. As utmost of the available action recognition data sets are n't realistic and are offered by actors, UCF101 aims to encourage farther exploration into action recognition by learning and exploring new realistic action orders.
Data Set Details
The videotape in 101 action orders are grouped into 25 groups, where each group can correspond of 4- 7 videotape of an action. The videotape from the same group may partake some common features, similar as analogous background, analogous standpoint, etc.
The action orders can be divided into five types
• mortal- Object Interaction
• Body- stir Only
• Human- Human Interaction
• Playing Musical Instruments
• Sports



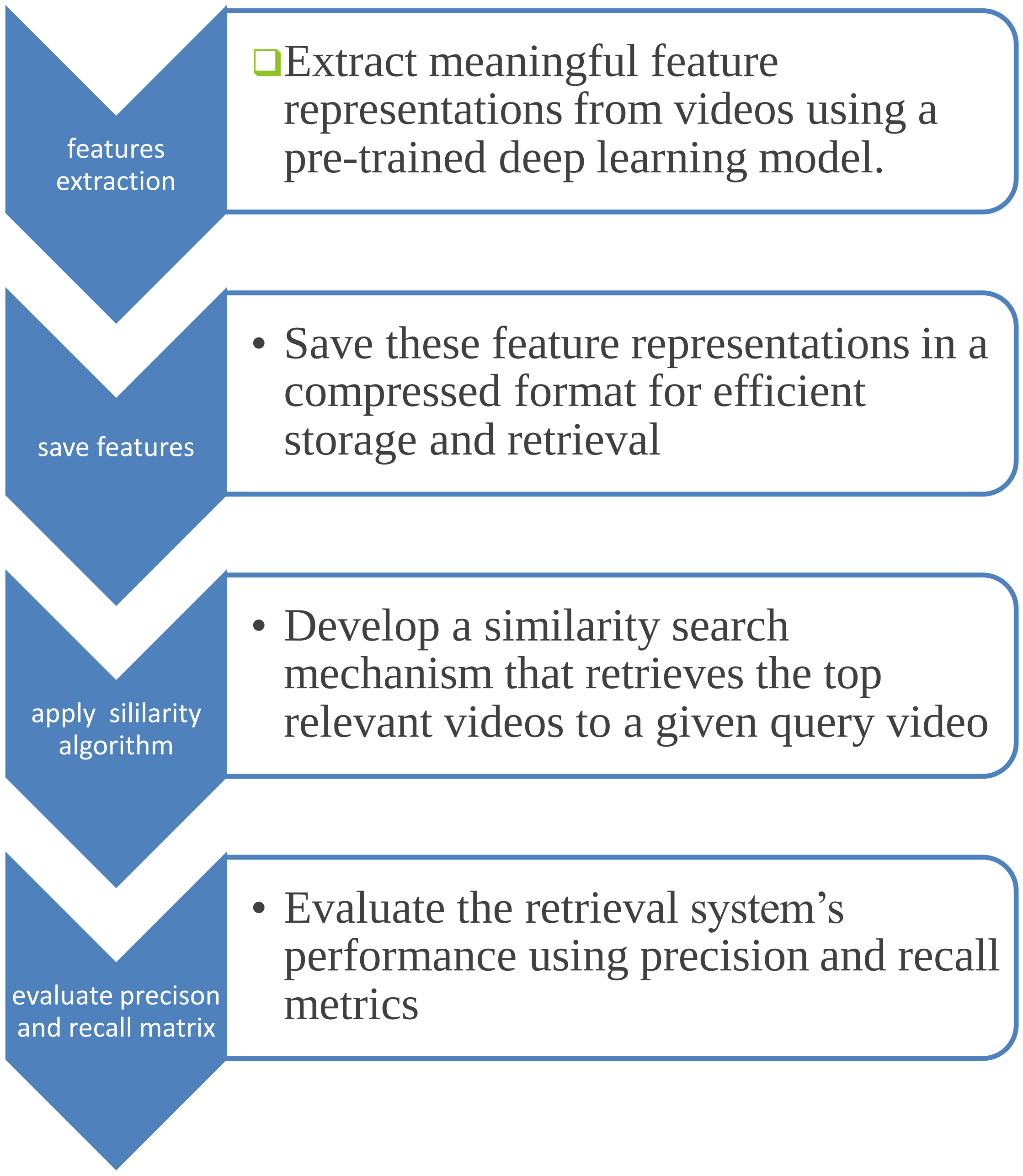

Playing Flute
Playing Guitar
Playing Piano
Playing Sitar
Playing Tabla
Playing Violin
Archery
Balance Beam
Baseball Pitch
Basketball
Basketball Dunk
Bench Press
Biking
Billiard
Bowling
Boxing-Punching Bag
Boxing-Speed Bag
Breaststroke
Clean and Jerk
Cliff Diving
Cricket Bowling
Cricket Shot
Diving
Fencing
Field Hockey Penalty
Floor Gymnastics
Frisbee Catch
Front Crawl
Golf Swing
Hammer Throw
High Jump
Horse Race
Horse Riding
Ice Dancing
Javelin Throw
Kayaking
Long Jump
Parallel Bars
Pole Vault
Pommel Horse
Punch
Rafting
Rowing
Shotput
Skiing
Jetski
Sky Diving
Soccer Penalty
Still Rings
Sumo Wrestling
Surfing
Table Tennis Shot
Tennis Swing
Throw Discus
Uneven Bars
Volleyball Spiking

Statistics


2500
2000
1500
1000
500
0
Time (sec)
16
14
12
10
8
6
4
2
0
Total Time
Average Clip Duration
ApplyEyeMakeup
ApplyLipstick
Archery
BabyCrawling
BalanceBeam
BandMarching
BaseballPitch
Basketball
BasketballDunk
BenchPress
Biking
Billiards
BlowDryHair
BlowingCandles
BodyWeightSquats
Bowling
BoxingPunchingBag
BoxingSpeedBag
BreastStroke
BrushingTeeth
CleanAndJerk
CliffDiving
CricketBowling
CricketShot
CuttingInKitchen
Diving
Drumming
Fencing
FieldHockeyPenalty
FloorGymnastics
FrisbeeCatch
FrontCrawl
GolfSwing
Haircut
Hammering
HammerThrow
HandstandPushups
HandstandWalking
HeadMassage
HighJump
HorseRace
HorseRiding
HulaHoop
IceDancing
JavelinThrow
JugglingBalls
JumpingJack
JumpRope
Kayaking
Knitting

Problem Formulation

Methodology


features extraction
Extract meaningful feature representations from videos using a pre-trained deep learning model.
save features
Save these feature representations in a compressed format for efficient storage and retrieval
apply sililarity algorithm
Develop a similarity search mechanism that retrieves the top relevant videos to a given query video
evaluate precison and recall matrix
Evaluate the retrieval system's performance using precision and recall metrics

1. Feature Extraction:

- Purpose This step aims to prize meaningful visual and temporal features from videotape that can be used to represent the videotape's content.
- Trained deep literacy model, similar as a 3D CNN, is used to reuse each videotape

frame.

- The model excerpts features that prisoner the appearance, stir, and spatial-temporal connections within the videotape.
- These features are also added up to represent the entire videotape.

2. point Representation and storehouse

- Purpose The uprooted features need to be stored efficiently for latterly reclamation.
- The point representations are compressed to reduce storehouse space.
- They're saved in a database along with metadata about the videotape(e.g., title, description, markers).

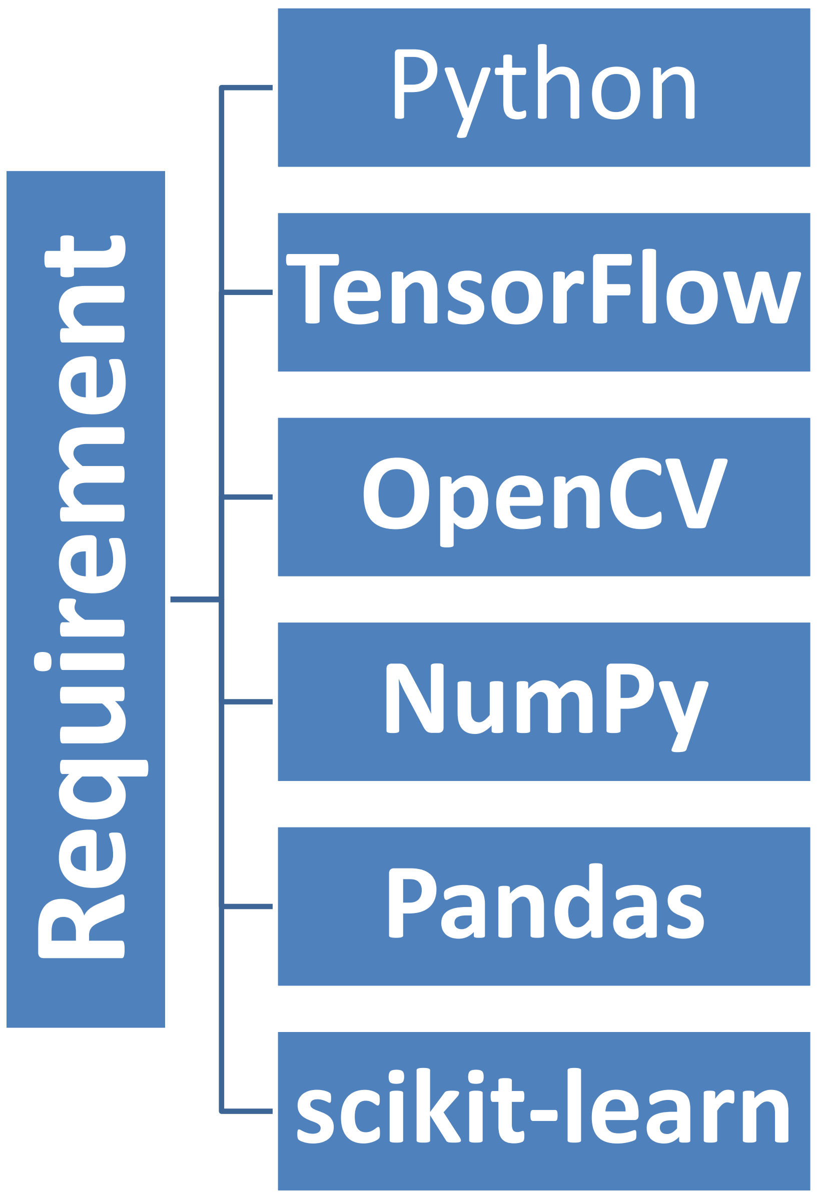
3. Similarity Hunt

- Purpose This step involves chancing videotape analogous to a given query videotape.
- When a stoner submits a query videotape, its features are uprooted.
- The uprooted features are compared to the features of all videotape in the database using a similarity metric(e.g., cosine similarity).
- The top- ranked videotape grounded on similarity are recaptured and presented to the stoner.

4. Evaluation

- Purpose The performance of the CBVR system needs to be assessed.
- Standard information reclamation criteria like perfection and recall are used to estimate the system's delicacy.
- Precision measures the proportion of recaptured videotape that are applicable, while recall measures the proportion of applicable videotape that are recaptured.
- The system's performance is estimated on a test dataset with ground verity markers.

Requirement



Python

- part Core programming language for enforcing the entire system.
- Model development and training
- Data processing and manipulation
- point birth and representation

- Similarity hunt and reclamation
- System evaluation

TensorFlow

- lading and exercising model-trained ResNet- 50 model
- rooting features from videotape frames
- Potentially fine- tuning the model on the UCF101 dataset
- Role Deep learning frame for feature extraction

OpenCV

- Role Computer vision library for videotape processing.
- Reading and writing videotape lines
- rooting individual frames from videotape
- Resizing frames to a harmonious size for processing

NumPy

- Role Numerical calculating library for effective array operations.
- Handling and manipulating numerical data(e.g., point vectors, similarity matrices)
- Efficiently storing and recycling large datasets

Pandas

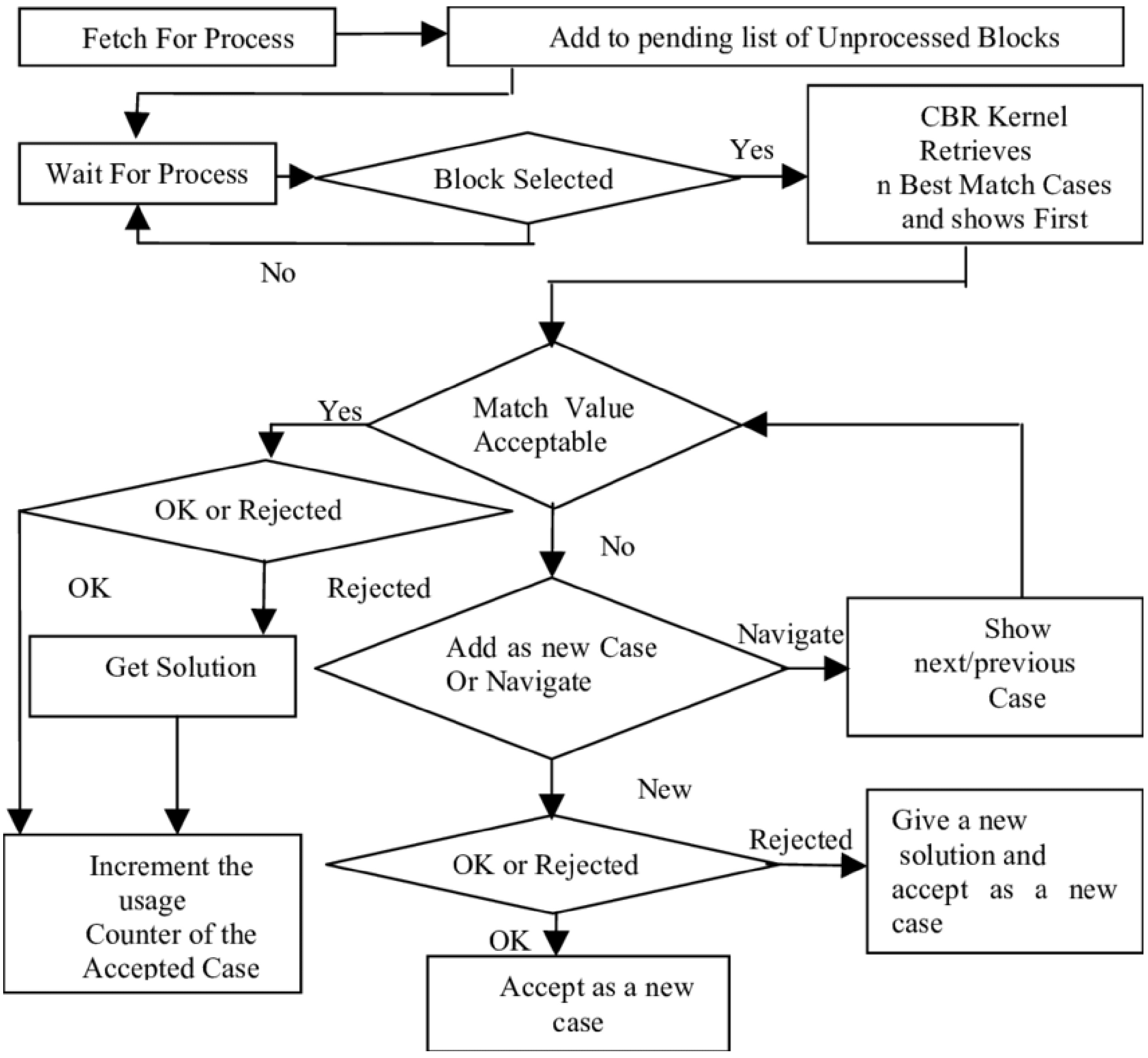
- Role Data analysis and manipulation library.
- Reading and loading metadata stored in CSV lines.
- Managing and manipulating the metadata data.

scikit- learn

- part Machine literacy library.
- furnishing the cosine similarity function to measure the similarity between point vectors.
- Cosine similarity is a crucial element in chancing videotape analogous to a given query videotape.

Design

Flow Chat


Fetch For Process
Add to pending list of Unprocessed Blocks
Wait For Process
Block Selected
Yes
No
CBR Kernel Retrieves n Best Match Cases and shows First
Match Value Acceptable
Yes
No
OK or Rejected
OK
Rejected
Get Solution
Add as new Case Or Navigate
Navigate
Show next/previous Case
New
Increment the usage Counter of the Accepted Case
OK or Rejected
Rejected
Give a new solution and accept as a new case
OK
Accept as a new case

Discussions

Code

```python
import os
import numpy as np
import pandas as pd
import tensorflow as tf
import cv2
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

```python
# Load the pre-trained ResNet50 model, excluding the top layer
model = tf.keras.applications.ResNet50(weights='imagenet', include_top=False,
pooling='avg')
```

```python
print("Model type:", type(model))
```

```
Model type: <class 'keras.src.models.functional.Functional'>
```

```python
def preprocess_frame(frame):
    frame = cv2.resize(frame, (224, 224))  # Resize frame to 224x224
    frame = tf.keras.applications.resnet50.preprocess_input(frame)  # Preprocess
for ResNet50
    return frame
```

```python
def extract_features_from_video(video_path, model):
    # Open the video file
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print(f"Error: Unable to open video file {video_path}")
        return None  # Return None if the video file can't be opened

    features = []

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Preprocess the frame and add batch dimension
        frame = preprocess_frame(frame)
        frame = np.expand_dims(frame, axis=0)

        # Extract features using ResNet50
        feature = model.predict(frame)
        features.append(feature.flatten())

    cap.release()

    # Check if features list is empty
    if not features:
        print(f"Warning: No frames extracted from video file {video_path}")
        return None  # Return None if no frames were processed

    # Average features across all frames
    video_features = np.mean(features, axis=0)
    return video_features
```

```python
def save_feature_vectors(dataset_video_paths, feature_vectors_file):
    """
    Extract features for the dataset videos and save them to a file.
    """
    features = []
    video_paths = []

    for video_path in dataset_video_paths:
        feature = extract_features_from_video(video_path, model)
        if feature is not None:
            features.append(feature)
            video_paths.append(video_path)

    # Save the feature vectors and corresponding video paths
    np.savez_compressed(feature_vectors_file, features=features,
video_paths=video_paths)
    print(f"Feature vectors saved to {feature_vectors_file}")
```

```python
def load_feature_vectors(feature_vectors_file):
    """
    Load the saved feature vectors and corresponding video paths.
    """
    data = np.load(feature_vectors_file)
    features = data['features']
    video_paths = data['video_paths']
    return features, video_paths
```

```python
def find_similar_videos(input_video_path, stored_features, stored_video_paths,
dataset_labels, model, top_k=5):
    # Extract features from the input video
    input_features = extract_features_from_video(input_video_path, model)
    if input_features is None:
        print(f"Error: Unable to extract features from the input video
{input_video_path}")
        return [], [], []  # Return empty lists if input features can't be
extracted
```

```
    # Compute cosine similarity between input video and dataset videos
    similarities = cosine_similarity([input_features], stored_features).flatten()

    # Get the indices of the top_k most similar videos
    top_indices = np.argsort(similarities)[-top_k:][::-1]
    top_similar_videos = [stored_video_paths[i] for i in top_indices]
    top_similar_labels = [dataset_labels[i] for i in top_indices]  # Get labels
of top retrieved videos

    return top_similar_videos, top_similar_labels, top_indices
```

```
def load_video_paths_and_labels_from_csv(csv_file_path):
    df = pd.read_csv(csv_file_path)
    return df['Video File'].tolist(), df['Label'].tolist()
```

```
def calculate_precision_recall(input_label, dataset_labels, top_indices):
    # Count relevant videos in top retrieved videos
    relevant_retrieved = sum(1 for i in top_indices if dataset_labels[i] ==
input_label)
    total_relevant = dataset_labels.count(input_label)
    total_retrieved = len(top_indices)

    # Calculate precision and recall
    precision = relevant_retrieved / total_retrieved if total_retrieved > 0 else
0
    recall = relevant_retrieved / total_relevant if total_relevant > 0 else 0

    return precision, recall
```

```
def plot_confusion_matrix(input_label, top_similar_labels, all_labels):
    # Generate the confusion matrix
    y_true = [input_label] * len(top_similar_labels)
    y_pred = top_similar_labels

    # Pass all known labels to ensure matrix shape is consistent
    cm = confusion_matrix(y_true, y_pred, labels=all_labels)

    # Plot the confusion matrix
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=all_labels)
    disp.plot(cmap=plt.cm.Blues)
    plt.title("Confusion Matrix for Top Retrieved Videos")
    plt.xticks(rotation=90)
    plt.show()
```

```
# Example usage
csv_file_path = 'UCF101_labeled_data_sampled.csv'  # Path to the CSV file
containing video paths and labels
```

```python
feature_vectors_file = 'video_features_sampled.npz'  # File to save/load feature
vectors
input_video_path = ' v_IceDancing_g02_c03.avi'  # Path to the input video
input_video_label = ' IceDancing'  # Label of the input video
top_k = 5

# Load dataset video paths and labels from the CSV file
dataset_video_paths, dataset_labels =
load_video_paths_and_labels_from_csv(csv_file_path)

# Extract and save feature vectors (if not already done)
if not os.path.exists(feature_vectors_file):
    save_feature_vectors(dataset_video_paths, feature_vectors_file)

# Load the stored feature vectors and video paths
stored_features, stored_video_paths = load_feature_vectors(feature_vectors_file)

# Find the top 5 similar videos
similar_videos, top_similar_labels, top_indices =
find_similar_videos(input_video_path, stored_features, stored_video_paths,
dataset_labels, model, top_k)
print("Top 5 similar videos:", similar_videos)
```

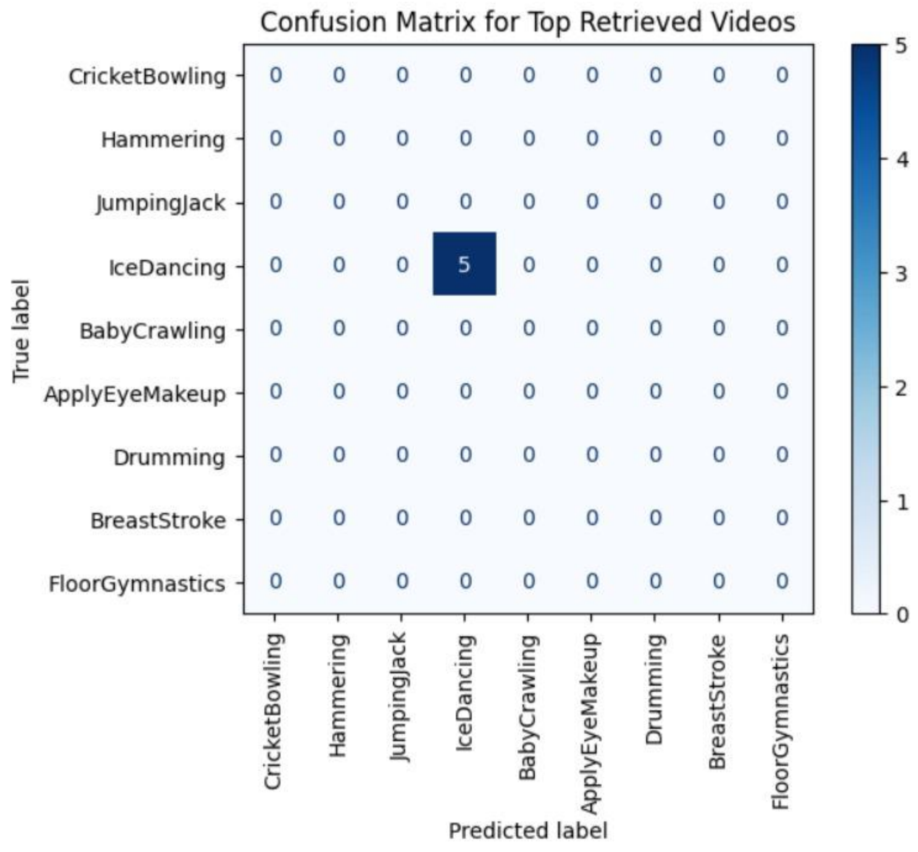
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 163ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 150ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 385ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 518ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 473ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 490ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 487ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 418ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 518ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 494ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 472ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 483ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 255ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 330ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 500ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 455ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 488ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 432ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 504ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 481ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 472ms/step
...
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 488ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 504ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 486ms/step
```

```
Top 5 similar videos: [np.str_('UCF-sampled/ IceDancing/
v_IceDancing_g02_c02.avi'), np.str_('UCF-sampled/ IceDancing/
v_IceDancing_g02_c03.avi'), np.str_('UCF-sampled/ IceDancing/
v_IceDancing_g02_c04.avi'), np.str_('UCF-sampled/ IceDancing/
v_IceDancing_g02_c04.avi'), np.str_('UCF-sampled/ IceDancing/
v_IceDancing_g02_c05.avi') ]
```
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# Example usage
# Get unique labels from dataset_labels to use as all_labels
all_labels = list(set(dataset_labels))
similar_videos, top_similar_labels, top_indices =
find_similar_videos(input_video_path, stored_features, stored_video_paths,
dataset_labels, model, top_k)
plot_confusion_matrix(input_video_label, top_similar_labels, all_labels)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 165ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 158ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 156ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 261ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 457ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 461ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 488ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 472ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 467ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 403ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 457ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 488ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 433ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 472ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 448ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 480ms/step
...
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 476ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 450ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 495ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 464ms/step
```
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
Precision: 1.00
Recall: 1.00
```



```python
# Calculate precision and recall
precision, recall = calculate_precision_recall(input_video_label, dataset_labels,
top_indices)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Precision: 1.00
Recall: 1.00
```

Conclusion

• videotape hunt machines This model can be used to search for analogous videotape in a large collection grounded on their content.
• videotape recommendation systems It can recommend videotape to druggies grounded on their watch history or the content of the videotape they're presently watching.
• videotape categorization It can help classify videotape into different classes grounded on their similarity labeled exemplifications.
• videotape anomaly discovery by comparing a videotape to a set of normal videotape, the model can potentially identify unusual or anomalous videotape content.
Overall, this law provides a good starting point for erecting a video similarity model using TensorFlow. You can further ameliorate the model
• Training a custom videotape bracket model on your specific dataset.
• Exploring different point birth ways and distance criteria.
• exercising ways like videotape segmentation for better point representation.

Reference

https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f

https://blog.roboflow.com/what-is-resnet-50/

https://www.crcv.ucf.edu/data/UCF101.php

https://www.crcv.ucf.edu/data/UCF101.php#Results_on_UCF101

https://www.kaggle.com/datasets/pevogam/ucf101