# VIDEO RECOGNITION AND CLASSIFICATION USING DEEP LEARNING

A PROJECT REPORT

BY

KAPIL KUMAR BHITORIA(E23CSEU1552)
VATSAL MITTAL(E23CSEU1547)

SUBMITTED TO

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND TECHNOLOGY,
BENNETT UNIVERSITY

GREATER NOIDA, 201310, UTTAR PRADESH, INDIA

April 2025

# DECLARATION

I hereby declare that the project report titled **"Video Recognition and Classification Using Deep Learning"** is a record of my original work carried out during the course of this project. The information and data presented in this report are true to the best of my knowledge and belief.

This work has not been submitted to any other institution or university for the award of any degree, diploma, or other academic titles. All sources of information and data have been duly acknowledged.

**Kapil Kumar Bhitoria**

Enrolment No.: E23CSEU1538

**Vatsal Mittal**

Enrollment No.: E23CSEU1547

# ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to everyone who supported and guided us throughout the development of our project titled "Video Recognition and Classification Using Deep Learning."

We are especially thankful to our project mentor, Dr. Yajnaseni Dash for their continuous guidance, encouragement, and insightful feedback which played a crucial role in shaping the outcome of this work.

We are also thankful to our peers and friends for their valuable inputs and constant motivation during every phase of the project.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

**AI** – Artificial Intelligence

**API** – Application Programming Interface

**UI** – User Interface

**Jupyter** – Jupyter Notebook

# ABSTRACT

This project presents a deep learning approach for **video recognition and classification** by integrating **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)**. Video frames are extracted and resized using OpenCV, and spatial features are obtained using the pre-trained **InceptionV3** model. These features are encoded with Keras's StringLookup layer and passed through an RNN-based model for temporal sequence classification.

The model was trained with an image size of 224x224, batch size of 64, 30 epochs, and a max sequence length of 20. With a validation split of 0.3, the system achieved a **test accuracy of 76.55%**, demonstrating its potential for real-world video analysis tasks.

# 1. INTRODUCTION

## 1. Introduction

With the rapid expansion of digital content, video has become one of the most dominant forms of data on the internet. From entertainment and education to surveillance and healthcare, video data plays a vital role in modern applications. However, analyzing and understanding this data in an automated, scalable, and accurate manner remains a major challenge in the field of artificial intelligence.

Deep learning has made significant progress in image classification, object detection, and speech recognition. Extending these capabilities to videos requires models that can understand not only what appears in a single frame, but also how scenes evolve over time. This has led to the development of hybrid architectures that combine Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

## 1.1 Problem Statement

Despite advancements in deep learning, video classification still faces major hurdles. Traditional models often focus on spatial features alone, ignoring the critical temporal relationships between frames. Manual video analysis is inefficient, inconsistent, and infeasible at scale. Furthermore, most models lack the ability to generalize across diverse types of video content.

This project aims to address these challenges by building a deep learning-based solution that combines CNNs for spatial feature extraction and RNNs for modeling temporal dynamics, providing a more comprehensive understanding of video content.

# 2. BACKGROUND RESEARCH

With the surge in video data generation, traditional machine learning approaches have proven insufficient for handling the complexity of video analysis. Video classification is inherently more complex than image classification due to the added dimension of time, requiring models that can learn from both spatial content and temporal flow.

Deep learning, particularly the use of **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)**, has emerged as a powerful technique for video understanding. CNNs are effective for extracting spatial features from individual frames, while RNNs, especially GRU and LSTM variants, are capable of capturing temporal dependencies across sequences of frames. The combination of these two architectures allows for a more holistic understanding of video content.

**Proposed System**

The proposed system leverages a hybrid deep learning architecture that integrates CNN and RNN components. OpenCV is used to extract and resize video frames, which are then passed through a pre-trained **InceptionV3** CNN model for feature extraction. The resulting feature vectors are encoded using Keras's **StringLookup** layer and passed through an RNN model that includes masking, ReLU activation, and softmax layers for classification.

This design ensures the model can recognize and learn both spatial patterns within frames and temporal dynamics across frame sequences, resulting in accurate and efficient video classification.

**Goals and Objectives**

- To build an automated system capable of classifying videos using deep learning techniques.
- To utilize **CNNs** for spatial feature extraction and **RNNs** for temporal sequence modeling.
- To evaluate the effectiveness of transfer learning (using InceptionV3) in feature extraction.
- To achieve a high classification accuracy with minimal manual preprocessing.
- To demonstrate the scalability of the model for real-world video data applications.

# 3. Tracking

**Project Tracking and Deployment**

**The development of this project was structured into clear, manageable weekly milestones, allowing for steady progress and clarity throughout each phase:**

**Week 1–2:**
**Conducted background research on video classification using CNNs and RNNs, explored relevant tools, datasets, and models, and established the foundational plan.**

**Week 3:**
**Extracted frames from videos using OpenCV and resized them to ensure uniformity across the dataset. Defined the preprocessing pipeline for data preparation.**

**Week 4:**
**Implemented feature extraction using the pre-trained InceptionV3 CNN model through transfer learning, focusing on extracting relevant visual features from video frames.**

**Week 5:**
**Developed the feature encoding layer using Keras' StringLookup and integrated it into the pipeline for encoding the extracted features.**

**Week 6:**
**Built the RNN-based temporal modeling pipeline, ensuring proper sequence handling for video frame data.**

**Week 7–8:**
**Trained the hybrid CNN-RNN model, configured training parameters (batch size, sequence length, epochs), and performed initial model tuning for optimal performance.**

**Week 9:**
**Evaluated model performance with test accuracy calculations (76.55%) and validated results through rigorous testing and adjustments.**

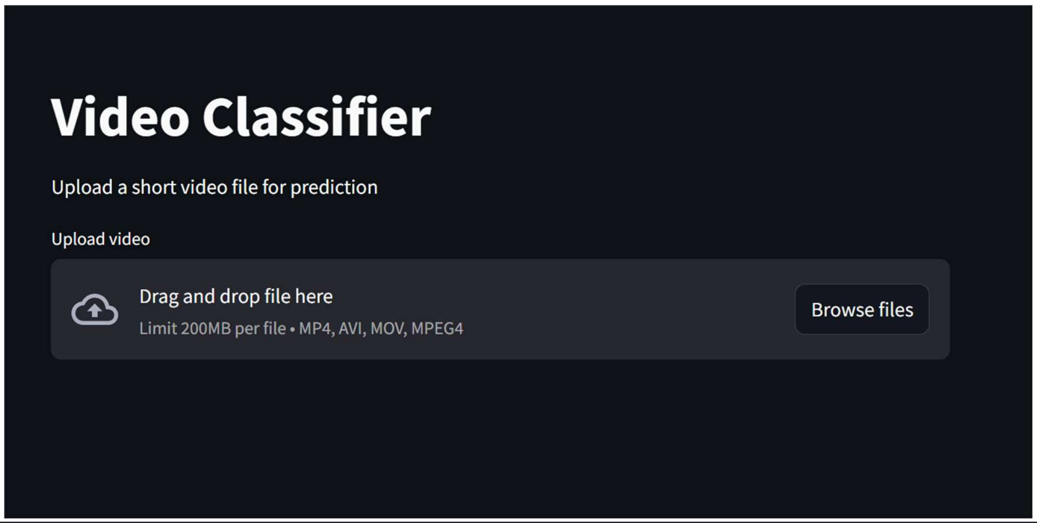
**Week 10:**
**Designed the user interface using Streamlit, integrated the trained model into the UI, and conducted thorough testing. The deployment enabled users to upload and analyze videos in real-time, view predictions with confidence scores, and interact with a user-friendly, browser-based interface. The final deployment was successful, making the project easily shareable and accessible with a functional, web-based application for video recognition.**

# 4. SYSTEM ANALYSIS AND DESIGN

The system is designed as a modular deep learning pipeline with the following key components:

- **Frame Extraction**: Video data is split into frames using **OpenCV**, and each frame is resized to 224x224 pixels.
- **Feature Extraction (CNN)**: A pre-trained **InceptionV3** model is used to extract high-dimensional feature vectors from each frame. This is part of a transfer learning approach to speed up development and improve performance.
- **Feature Encoding**: Encoded using **Keras's StringLookup** layer to prepare for sequence modeling.
- **Temporal Modeling (RNN)**: A **GRU-based RNN** model processes the sequence of encoded frame features. It uses a **mask input layer** to handle varying sequence lengths.
- **Classification**: Dense layers with **ReLU** and **Softmax** activations are used to output class probabilities.
- **Training**: The model is trained with a **batch size of 64**, **sequence length of 20**, **2048 feature size**, **30 epochs**, and **validation split of 0.3**.

## 5. USER INTERFACE



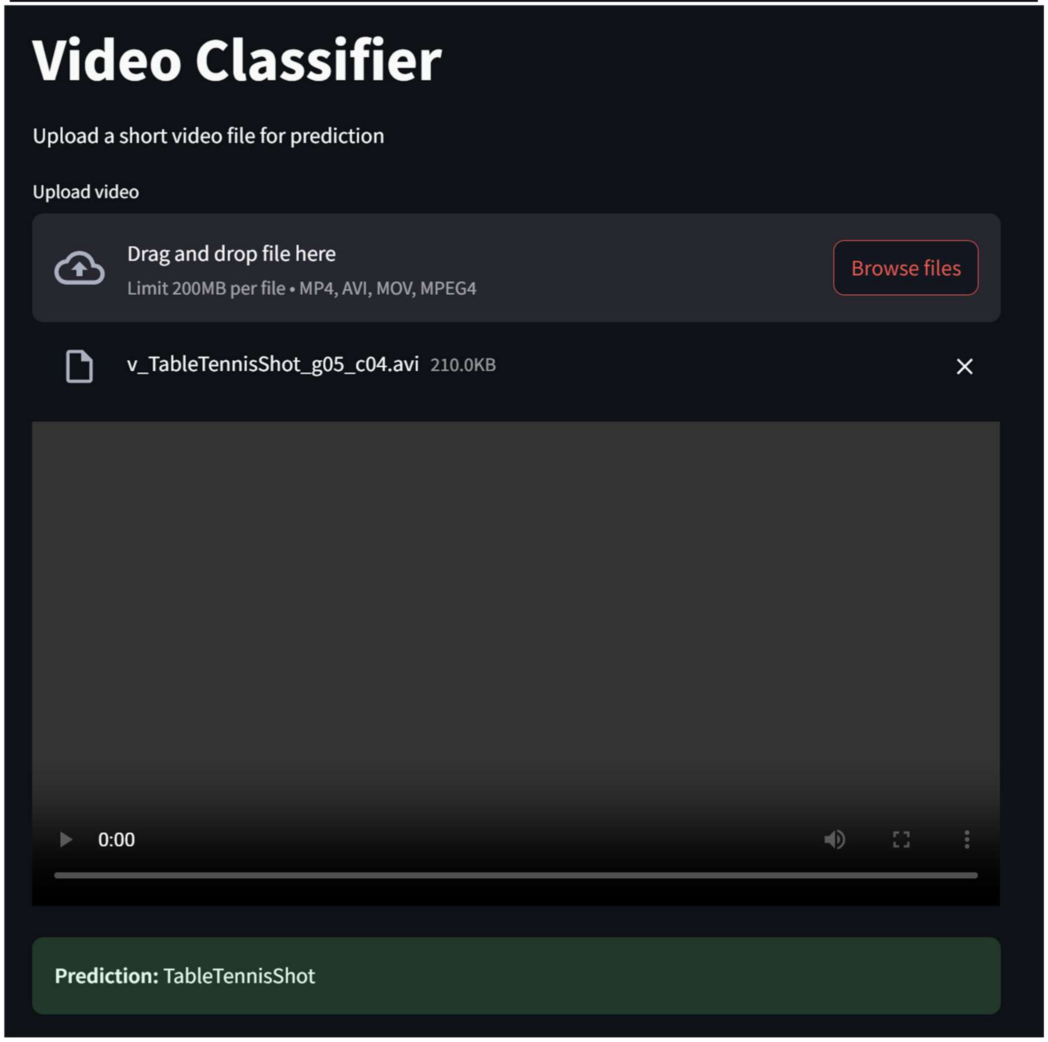
Video Classifier

Upload a short video file for prediction

Upload video

Drag and drop file here

Limit 200MB per file • MP4, AVI, MOV, MPEG4



Video Classifier

Upload a short video file for prediction

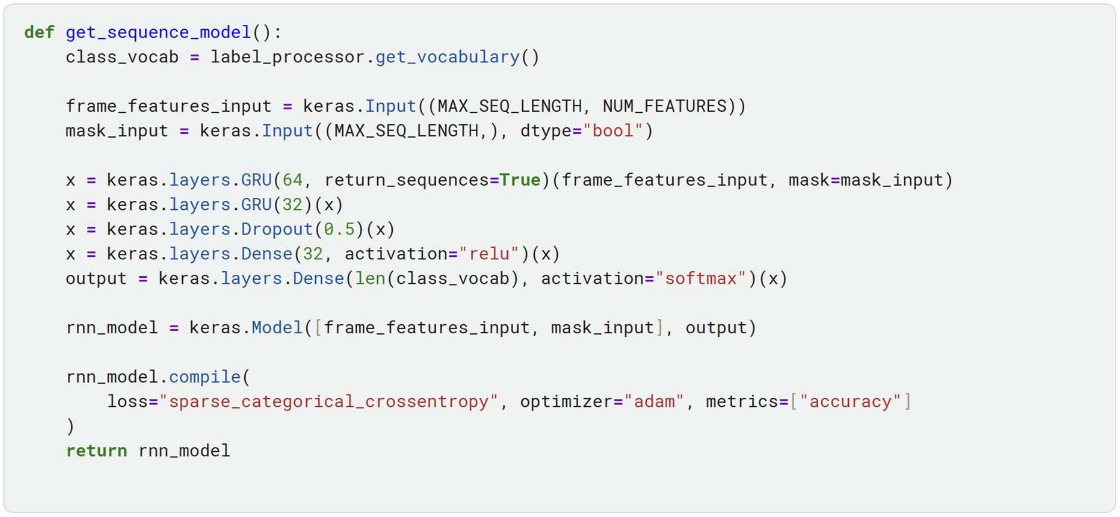
Upload video

Drag and drop file here

Limit 200MB per file • MP4, AVI, MOV, MPEG4

**Prediction:** TableTennisShot

## 6. ALGORITHMS / PSEUDO CODE OF CORE FUNCTIONALITY



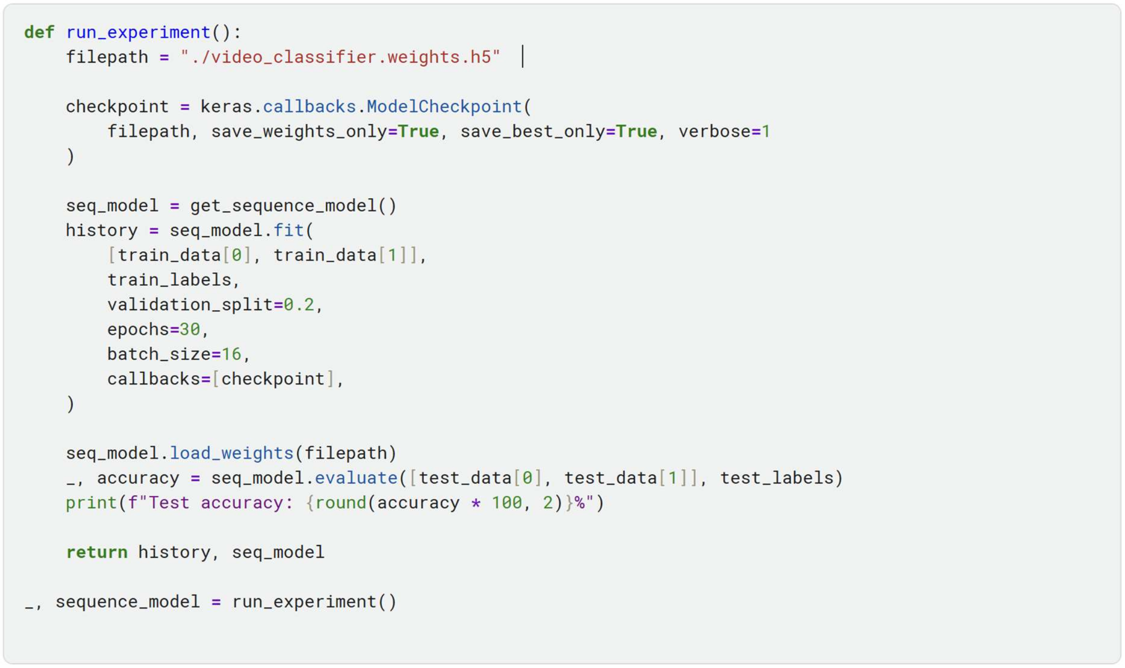
```python
def get_sequence_model():
    class_vocab = label_processor.get_vocabulary()

    frame_features_input = keras.Input((MAX_SEQ_LENGTH, NUM_FEATURES))
    mask_input = keras.Input((MAX_SEQ_LENGTH,), dtype="bool")

    x = keras.layers.GRU(64, return_sequences=True)(frame_features_input, mask=mask_input)
    x = keras.layers.GRU(32)(x)
    x = keras.layers.Dropout(0.5)(x)
    x = keras.layers.Dense(32, activation="relu")(x)
    output = keras.layers.Dense(len(class_vocab), activation="softmax")(x)

    rnn_model = keras.Model([frame_features_input, mask_input], output)

    rnn_model.compile(
        loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"]
    )
    return rnn_model
```



```python
def run_experiment():
    filepath = "./video_classifier.weights.h5"

    checkpoint = keras.callbacks.ModelCheckpoint(
        filepath, save_weights_only=True, save_best_only=True, verbose=1
    )

    seq_model = get_sequence_model()
    history = seq_model.fit(
        [train_data[0], train_data[1]],
        train_labels,
        validation_split=0.2,
        epochs=30,
        batch_size=16,
        callbacks=[checkpoint],
    )

    seq_model.load_weights(filepath)
    _, accuracy = seq_model.evaluate([test_data[0], test_data[1]], test_labels)
    print(f"Test accuracy: {round(accuracy * 100, 2)}%")

    return history, seq_model

_, sequence_model = run_experiment()
```

# 7. PROJECT CLOSURE

## 7.1 Goals / Vision

The vision of the **Video Recognition and Classification** project is to develop a robust AI-based system that can accurately classify video content in real time. By leveraging deep learning models, the system adapts to different types of video data and provides efficient and reliable classification. The ultimate goal is to improve video analysis capabilities across various industries, including healthcare, security, and entertainment, by automating the classification process.

## 7.2 Delivered Solution

A fully integrated AI solution has been successfully developed and deployed, including the following components:

- **Video Classification Model**:
  A deep learning-based model combining Convolutional Neural Networks (CNN) for feature extraction and Recurrent Neural Networks (RNN) for temporal analysis. The model achieved an accuracy of **76.55%** on test data.
- **Real-Time Video Processing**:
  Integration with OpenCV allows for seamless frame extraction and resizing, enabling efficient real-time processing of video content.
- **User Interface**:
  Deployed on Streamlit, the UI allows users to upload and analyze videos in real time. The interface presents clear, user-friendly predictions with confidence scores, making the system easy to interact with.
- **Deployment**:
  The system is deployed on Streamlit, making it accessible via a web browser without requiring any additional software installation. This ensures ease of access and sharing.

## 7.3 Remaining Work

While significant progress has been made, the following tasks remain for future improvements:

- **Model Performance Optimization**:
  Further tuning and training of the model are needed to improve accuracy and handle a broader variety of video content.
- **Expand Dataset**:
  Increasing the diversity and size of the dataset will enhance the model's generalization, especially for videos with varying content and quality.
- **Cloud Deployment**:
  Transition the deployment to cloud-based solutions to enhance scalability and ensure faster processing for larger datasets.
- **Advanced UI Features**:
  Enhance the user interface with additional features, such as multi-video uploads, batch processing, and more detailed result visualizations.
- **Extended Testing and Validation**:
  Perform additional testing with more diverse video types and user scenarios to ensure robust performance across a range of real-world conditions.

## References

1. Keras InceptionV3 Model: Keras. (2021). *InceptionV3 for image classification*. Retrieved from https://keras.io/api/applications/inceptionv3/
2. Streamlit Documentation: Streamlit. (2021). *Streamlit: Build data apps*. Retrieved from https://docs.streamlit.io/
3. OpenCV Documentation: OpenCV. (2021). *Open Source Computer Vision Library*. Retrieved from https://opencv.org/
4. TensorFlow StringLookup: TensorFlow. (2021). *StringLookup Layer*. Retrieved from https://www.tensorflow.org/api_docs/python/tf/keras/layers/StringLookup
5. Olah, C. (2015). *Understanding LSTM Networks*. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/