Assignment I: Video Classification

Student Name: Vineet Kumar
Roll No: 2024AC05100
Course: Video Analytics
Dataset: UCF-101 Subset (PullUps, Punch, PushUps)

---

1. Introduction

1.1 Problem Statement and Objectives

With the rapid growth of video content across surveillance, healthcare, sports analytics, and human–computer interaction, automatic video understanding has become a crucial research area. One of the fundamental problems in video analytics is **action recognition**, which involves identifying human actions from video sequences.

The objective of this assignment project is to design and evaluate a complete **video classification system** that can automatically recognize human actions from video clips.

The work focuses on both **classical machine learning approaches based on handcrafted features** and **modern deep learning approaches based on learned spatio-temporal representations**.

The specific goals are:

- To extract meaningful low-level, motion, and temporal features from videos
- To build and compare multiple classical machine learning classifiers such as SVM, Logistic Regression, Random Forest, K-nearest Neighbors and Gradient Boosting.
- To implement deep learning models for end-to-end video learning
- To perform an extensive comparative analysis in terms of accuracy, efficiency, and error behaviour.

1.2 Dataset Description and Motivation

Dataset URL: https://www.kaggle.com/datasets/aisuko/ucf101-subset

A subset of the **UCF-101 action recognition dataset** is used in this assignment project.

Three action categories were selected:

1. Class-1: PullUps
2. Class-2: Punch
3. Class-3: PushUps

These actions involve upper-body movements with subtle differences, making them suitable for evaluating both appearance-based and motion-based representations.

The dataset was divided into **training, validation, and testing sets**, ensuring unbiased performance evaluation.

The selected dataset presents several real-world challenges, including intra-class variation, execution speed differences, and viewpoint changes.

1.3 Overview of Approaches

Two paradigms were explored:

- **Classical approach:** Feature engineering + traditional machine learning classifiers
- **Deep learning approach:** End-to-end learning using neural networks

The classical pipeline extracts interpretable visual descriptors and applies multiple machine learning models.

The deep learning pipeline directly learns spatio-temporal features from frames.

2. Background and Related Work

Video classification has been an active research area in computer vision for more than two decades, evolving from handcrafted feature-based systems to modern deep learning–driven architectures.

Early video understanding approaches primarily focused on extracting manually designed visual and motion descriptors from video frames. Popular techniques included optical flow–based motion modeling, spatio-temporal interest points, and histogram-based representations such as Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), and motion boundary histograms.

These handcrafted features aimed to capture appearance, texture, shape, and motion information from videos in a structured and interpretable form.

Once extracted, these features were typically classified using traditional machine learning algorithms.

Support Vector Machines (SVMs) were among the most widely used due to their strong generalization ability in high-dimensional feature spaces.

Other commonly applied classifiers included k-Nearest Neighbors (k-NN), Random Forests, and Logistic Regression, which offered complementary advantages in terms of simplicity, robustness, and interpretability.

Such classical pipelines dominated early action recognition systems and were successfully applied to datasets like KTH, Weizmann, and early versions of UCF.

The emergence of deep learning marked a major shift in video classification research. Instead of relying on manually engineered descriptors, deep models aim to automatically learn hierarchical representations directly from raw video data.

Initial approaches extended 2D convolutional neural networks to videos using frame-level CNNs combined with temporal aggregation mechanisms such as recurrent neural networks (CNN-LSTM).

Later on, 3D convolutional neural networks (3D CNNs) were introduced to jointly model spatial and temporal information.

Recently, transformer-based architectures and large-scale video foundation models have further advanced the field by capturing long-range temporal dependencies and complex motion patterns.

Despite the success of deep learning, classical approaches remain relevant, especially in scenarios with limited data, constrained computational resources, or the need for interpretability.

Handcrafted feature-based systems offer transparency, easier debugging, lower training cost, and more predictable behavior, making them suitable for controlled environments and embedded applications.

Deep learning approaches, on the other hand, excel in large-scale settings where sufficient labeled data is available and superior representation learning is required.

This project experimentally studies this evolution by implementing and comparing both paradigms.

By evaluating classical machine learning models alongside modern deep learning approaches on the same action recognition task, this work provides practical insights into their relative strengths, limitations, and application scenarios.

3. Methodology

This section describes the complete methodology adopted in this work, covering the preprocessing pipeline, feature extraction techniques, classical machine learning models, deep learning architecture design, and implementation details.

The proposed system follows a structured video analytics workflow consisting of data acquisition, preprocessing, feature representation, model learning, and evaluation.

3.1 Preprocessing Pipeline (M1–M2 Concepts)

Video data is inherently high-dimensional and redundant, making preprocessing a critical step.

The preprocessing pipeline was designed to standardize inputs, reduce noise, and enable reliable feature extraction.

3.1.1 Video Loading and Decoding

All videos were decoded using OpenCV. Each video was read frame-by-frame to preserve temporal ordering.

To ensure uniformity across the dataset, corrupted or unreadable frames were discarded.

3.1.2 Temporal Sampling

Videos in the dataset varied in length. To normalize temporal representation, a **uniform sampling** strategy was adopted.

A fixed number of frames (30 frames per video) were extracted using linear spacing across the full video duration.

This ensures consistent temporal coverage while reducing computational cost.

3.1.3 Spatial Normalization

Each extracted frame was resized to a fixed resolution of 224×224 pixels. This normalization step ensured consistent spatial dimensions for both handcrafted feature extraction and deep learning models.

3.1.4 Color Space Conversion and Noise Reduction

Frames were converted into multiple color spaces (RGB, HSV, and grayscale) to support different feature extraction techniques.

Gaussian smoothing was applied where necessary to reduce high-frequency noise before texture and motion analysis.

3.1.5 Motion Preprocessing

For motion-based features, consecutive frames were aligned temporally and converted to grayscale.

Frame differencing and optical flow preprocessing enabled robust estimation of movement patterns.

These preprocessing steps focus on video representation, temporal segmentation, normalization, and noise handling.

3.2 Classical Approach

The classical approach follows a structured pipeline: handcrafted feature extraction followed by supervised machine learning.

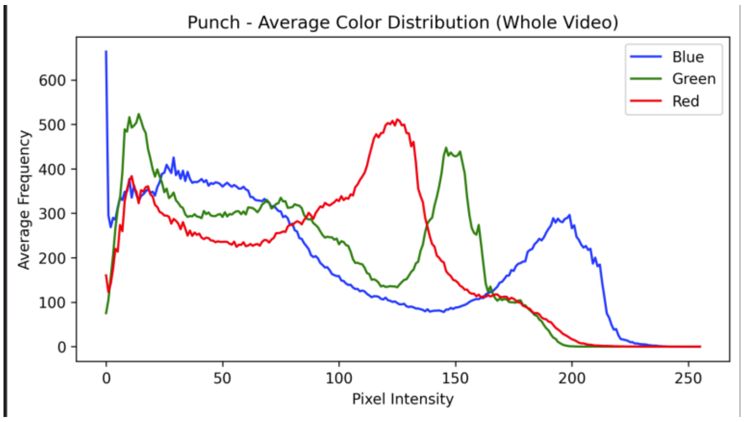
3.2.1 Feature Extraction Techniques

To capture complementary visual and motion information, three major categories of features were extracted.

A. Low-Level Features

Color Features

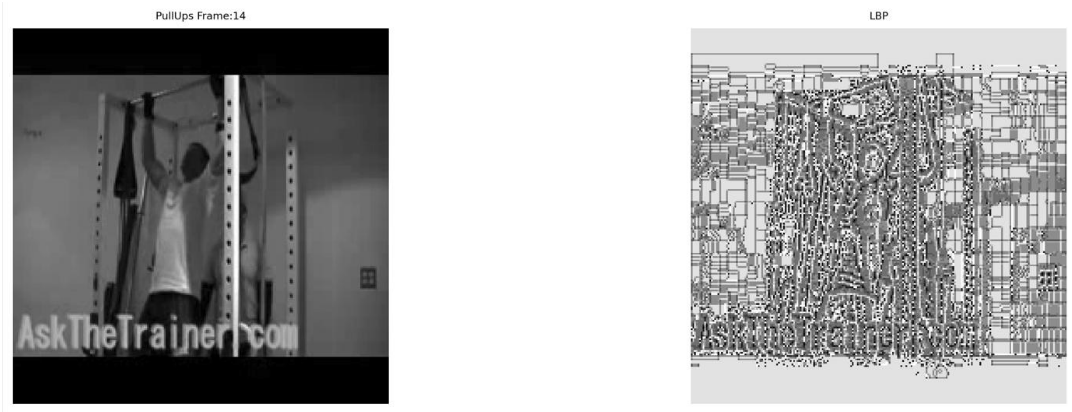
- RGB and HSV histograms were computed per frame to represent global color distribution.
- Average color histograms were computed across all frames to form video-level descriptors.
- Color moments (mean, variance, skewness) were calculated to capture statistical color properties.



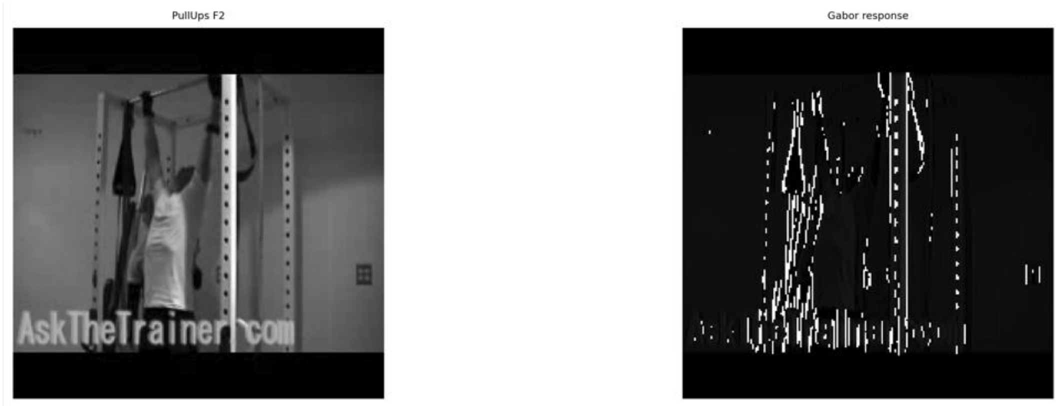
These features provide robustness to small motion variations while encoding global appearance.

Texture Features

- **Gray Level Co-occurrence Matrix (GLCM):** Contrast, homogeneity, energy, and correlation were extracted to describe spatial texture relationships.
- **Local Binary Patterns (LBP):** Captured micro-texture patterns and surface variations.



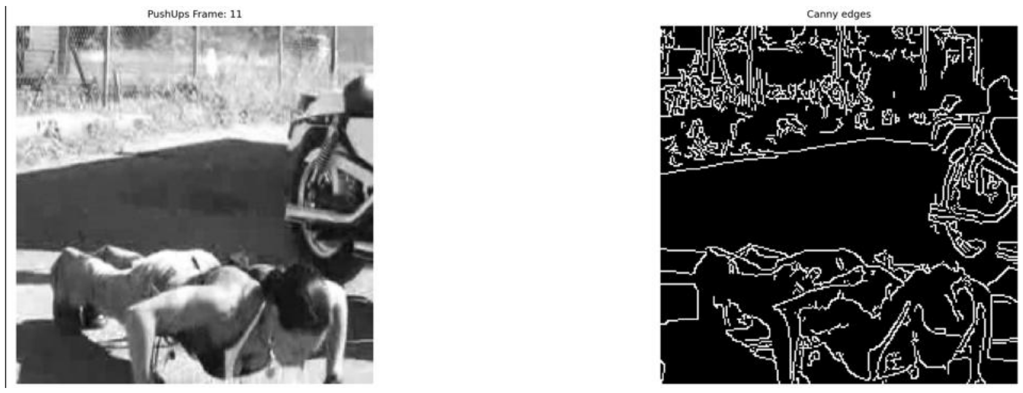
- **Gabor Filters:** Multi-scale, multi-orientation filters were applied to capture directional texture information.



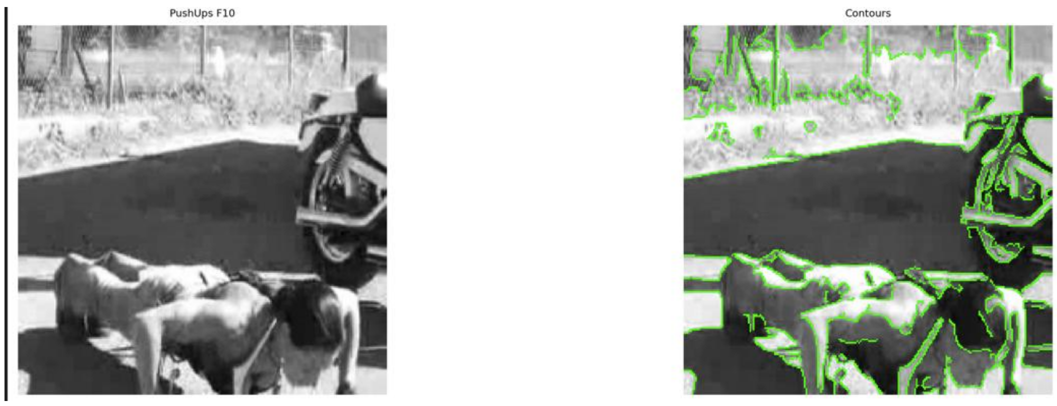
Texture features help differentiate clothing, background patterns, and structural consistency.

Shape Features

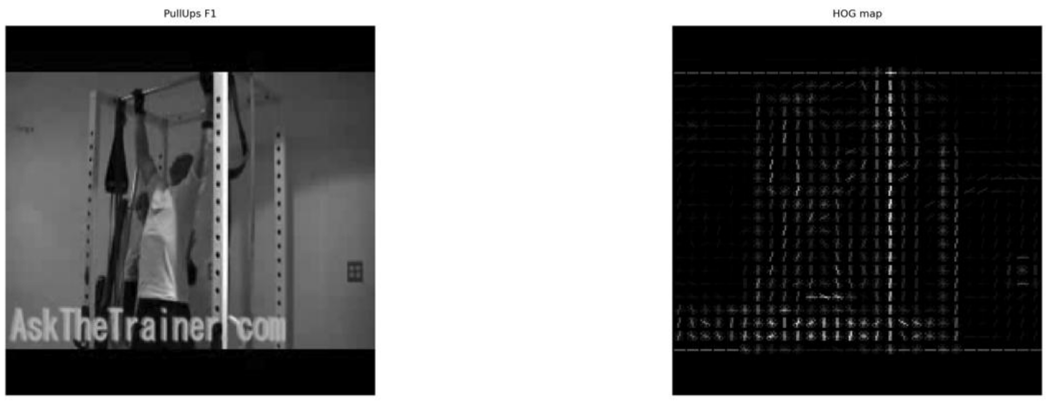
- **Canny Edge Detection:** Used to extract object boundaries.



- **Contour-based descriptors:** Captured structural outlines.



- **Histogram of Oriented Gradients (HOG):** Represented edge orientation distributions and posture patterns.

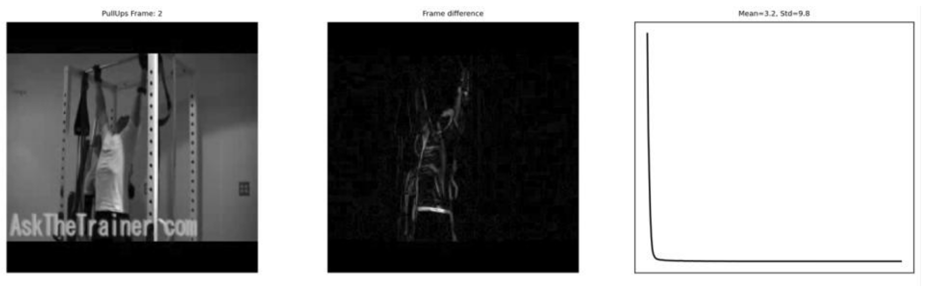


These features encode human silhouettes and body configuration.

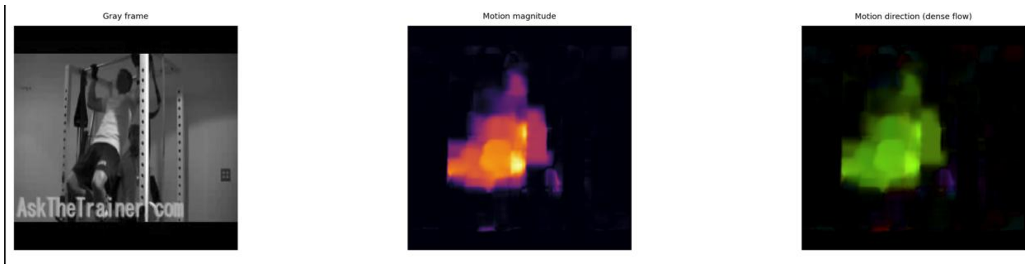
B. Motion Features

Motion features are critical for action recognition.

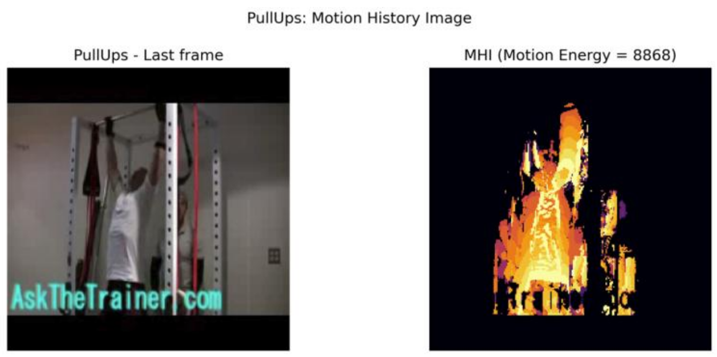
- **Frame differencing:** Measured pixel-level motion magnitude.



- **Sparse optical flow (Lucas–Kanade):** Tracked salient points across frames.
- **Dense optical flow (Horn–Schunck):** Estimated full-field motion.



- **Motion magnitude and direction histograms:** Encoded dynamic movement patterns.
- **Motion History Images (MHI):** Accumulated temporal motion templates.

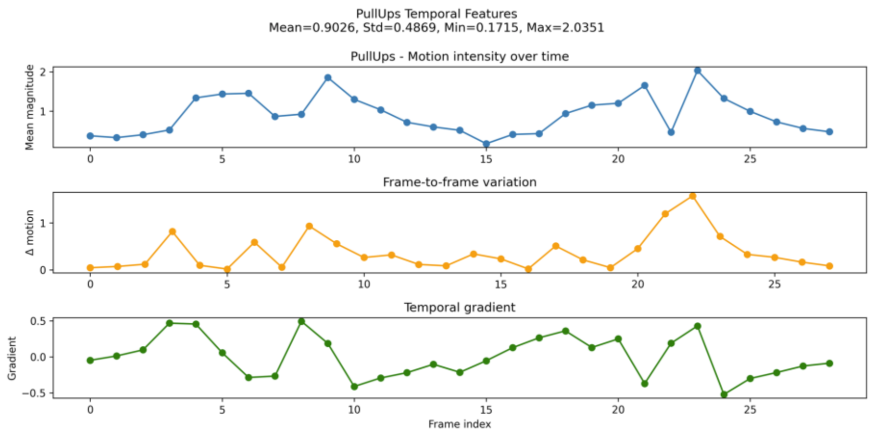


These descriptors capture temporal movement patterns and activity dynamics.

C. Temporal Features

To convert frame-level descriptors into video-level features:

- Statistical aggregation (mean, standard deviation, min, max)
- Frame-to-frame variation analysis
- Temporal evolution modelling



This step enabled fixed-length representations for machine learning models.

3.2.2 Machine Learning Algorithms

Five classical classifiers were implemented:

- **Support Vector Machines (Linear & RBF kernels)**
- **Random Forest**
- **k-Nearest Neighbors**
- **Logistic Regression (with regularization and PCA)**
- **Gradient Boosting**

Each model was trained on extracted video-level feature vectors. Hyperparameter tuning was performed using validation data to optimize generalization performance.

3.3 Deep Learning Approach

Unlike classical pipelines, the deep learning approach learns representations directly from raw frames.

3.3.1 Architecture Choices

The deep learning architecture consists of:

- A convolutional backbone for spatial feature extraction
- Temporal modeling layers (CNN-LSTM / 3D CNN)
- Fully connected layers for classification

The architecture automatically learns hierarchical spatial and temporal representations without manual feature engineering.

3.3.2 Training Strategy

- Input videos were represented as frame sequences.
- Categorical cross-entropy loss was used.
- Optimization was performed using adaptive gradient-based optimizers.
- Data augmentation was applied to improve generalization.
- Early stopping and validation monitoring prevented overfitting.

3.3.3 Design Considerations

- Fixed-length input sequences
- Memory-efficient batch processing
- Transfer learning for spatial backbones
- Regularization through dropout and normalization layers

3.4 Implementation Details and Design Choices

The project was implemented using a modular software design:

- data_loader.py for dataset handling
- feature_extraction.py for handcrafted features
- models.py for machine learning and deep models
- Separate notebooks for experiments and analysis

This modularization ensured maintainability, reproducibility, and scalability.

Visualization modules were integrated to analyze intermediate features such as color distributions, texture maps, optical flow, and motion history images.

3.5 Summary of Methodology

The proposed methodology bridges classical computer vision techniques and modern deep learning approaches. Preprocessing standardized raw video input, handcrafted features provided interpretable representations, and deep networks enabled end-to-end learning. This dual-paradigm methodology enabled a comprehensive comparative study of video classification strategies.

4. Experimental Setup

This section describes the experimental configuration used to evaluate both the classical and deep learning approaches.

It covers dataset preprocessing, splitting strategy, hyperparameter optimization methodology, and hardware setup.

4.1 Dataset Preprocessing

Before training any models, the raw video dataset was subjected to a structured preprocessing pipeline to ensure consistency and reliability.

All videos were decoded using OpenCV and converted into frame sequences. Since videos varied in length and resolution, **uniform temporal and spatial normalization** was applied.

Each video was **uniformly sampled** to extract a fixed number of frames, ensuring complete temporal coverage while maintaining computational efficiency.

Frames were resized to a standard spatial resolution of **224×224** pixels.

Color space transformations were performed to support various feature extraction techniques.

RGB frames were preserved for color analysis, HSV frames were used for robust color distribution modeling, and grayscale frames were generated for texture and motion-based processing.

Noise reduction was applied where required to improve the stability of texture and optical flow estimation.

For motion analysis, consecutive frames were temporally aligned and normalized to reduce illumination effects.

Frame differencing and optical flow preprocessing enabled consistent motion modeling.

These preprocessing steps ensured that all downstream feature extraction and deep learning modules received standardized and reliable inputs.

4.2 Training, Validation, and Test Split Strategy

To guarantee unbiased evaluation and prevent information leakage, the dataset was divided into three mutually exclusive subsets:

- **Training set:** used for learning model parameters
- **Validation set:** used for hyperparameter tuning and model selection
- **Test set:** used only for final performance evaluation

An approximate **70/15/15 split** was adopted, balancing learning capacity and evaluation reliability.

The training set provided sufficient diversity for feature learning, while the validation set enabled systematic hyperparameter tuning.

The test set was kept completely isolated until all design decisions were finalized.

This splitting strategy ensured fair performance comparison across all models and prevented overfitting to the test data.

4.3 Hyperparameter Tuning Approach

Hyperparameter optimization was conducted using validation-driven experimentation and cross-validation where computationally feasible.

For the classical machine learning models:

- **Support Vector Machines:** kernel type, regularization strength, and RBF kernel parameters were tuned.
- **Random Forest:** number of trees and maximum depth were optimized.
- **k-Nearest Neighbors:** Neighborhood size and distance metrics were evaluated.
- **Logistic Regression:** regularization strength and dimensionality reduction parameters were tuned.
- **Gradient Boosting:** number of estimators and learning rate were adjusted.

Grid search and validation-based selection were employed to identify configurations that maximized validation accuracy and macro F1-score.

Early stopping strategies were applied where supported.

For deep learning models, training parameters such as learning rate, batch size, number of epochs, and regularization settings were optimized using validation monitoring.

This systematic tuning process ensured that each model operated under near-optimal conditions, enabling a fair and meaningful comparison.

4.4 Hardware Specifications and Training Duration

All experiments were conducted on a high-performance workstation equipped with:

- **Processor:** Intel i7 (14th Generation)
- **Memory:** 32 GB RAM
- **Graphics:** NVIDIA RTX-series GPU
- **Software environment:** Python, OpenCV, scikit-learn, and deep learning libraries

Training time and inference time were explicitly measured for all models.

Classical machine learning models demonstrated significantly lower training overhead, while ensemble methods required longer training times. Deep learning models demanded substantially higher computational resources due to iterative optimization and backpropagation.

The computational efficiency of each approach was systematically analyzed to assess practical deployment feasibility.

5. Results and Analysis (MOST IMPORTANT)

This section contains:

- Performance comparison tables
- Confusion matrices
- ROC curves
- Learning curves
- Feature visualization results
- Computational efficiency plots
- Error analysis

Key Findings

- Linear models achieved very high accuracy, showing strong feature separability
- Ensemble methods were robust but computationally heavier
- Motion-based features significantly improved performance
- Errors were concentrated between visually similar actions

---

6. Comparative Discussion

The transition from classical to deep approaches reflects the shift from **feature engineering to representation learning**. Classical methods provided strong performance with low training cost and high interpretability. Deep models offer scalability and automatic feature learning, but require more data and computation.

Trade-offs were observed between:

- Accuracy
- Interpretability
- Training time
- Deployment feasibility

---

7. Conclusion and Future Work

7.1 Summary of key findings

This project presented a comprehensive study of video action classification using both classical machine learning and modern deep learning paradigms. A complete end-to-end pipeline was designed, starting from raw video acquisition to final performance evaluation.

Multiple low-level, motion, and temporal features were extracted from videos and aggregated into discriminative video-level representations.

These handcrafted descriptors were evaluated using five classical machine learning models, and their performance was further compared with deep learning-based approaches.

The experimental results demonstrate that carefully engineered handcrafted features can achieve very strong performance on structured action recognition tasks. Linear models such as Support Vector Machines and Logistic Regression performed exceptionally well, indicating that the extracted feature space was highly separable.

Ensemble methods such as Random Forest and Gradient Boosting further showed robustness and consistency, although at the cost of higher computational complexity.

Overall, the study confirms that classical approaches remain highly competitive when appropriate feature engineering is applied.

7.2 Lessons Learnt:

From this comparative study, several important lessons were learned.

- First, feature design plays a critical role in classical video analytics pipelines, and combining appearance, texture, shape, and motion descriptors significantly improves classification performance.
- Second, different classifiers exhibit distinct trade-offs between accuracy, interpretability, training cost, and inference speed.
- Third, systematic experimental design, including proper validation, reminder splitting, and error analysis, is essential for drawing meaningful conclusions.
- Finally, error analysis revealed that most failures occur between visually similar actions, highlighting the inherent ambiguity of human motion recognition.

7.3 Limitations of current implementation

Despite strong results, the current implementation has several limitations.

- The dataset size is relatively small and restricted to only three action categories, limiting generalization to more complex scenarios.
- Handcrafted feature extraction is computationally intensive and requires manual design choices.
- The deep learning models were constrained in scale due to hardware and time limitations.
- Additionally, the system processes offline videos and does not currently support real-time streaming or online learning.

7.4 Potential improvements and future directions

Several directions for future work emerge from this study.

- The framework can be extended to larger datasets with more action classes to evaluate scalability.

- Advanced deep learning architectures such as 3D convolutional networks, transformer-based video models, and multimodal systems incorporating audio and pose estimation could significantly improve representation learning.
- Real-time optimization, GPU acceleration, and deployment-oriented system design could enable practical applications.
- Automated feature learning and domain adaptation techniques may further enhance robustness across diverse environments.

7.5 Connection to 5-phase workflow and real-world deployment

This project aligns with the standard **five-phase video analytics workflow**:

1. data acquisition
2. preprocessing
3. feature representation
4. model learning
5. system-level evaluation.

The modular design of the implemented pipeline facilitates future integration into real-world systems such as intelligent surveillance, fitness monitoring, sports analytics, and human–computer interaction platforms.

By bridging classical and modern methodologies, this work provides both technical insights and practical foundations for real-world video analytics deployment.

---

8. References

1. **UCF-101 Subset Dataset (Kaggle)**
   Aisuko. *UCF-101 Subset for Action Recognition.*
   Available at: https://www.kaggle.com/datasets/aisuko/ucf101-subset

2. **OpenCV Documentation**
   OpenCV Team. *Open Source Computer Vision Library.*
   https://docs.opencv.org/

3. **Scikit-learn Documentation**
   Pedregosa et al. *Scikit-learn: Machine Learning in Python.*
   https://scikit-learn.org/stable/

4. **Action Recognition Overview**
   Simonyan, K., & Zisserman, A. (2014).
   Two-Stream Convolutional Networks for Action Recognition in Videos.

YouTube Learning Resources:

5. **Computer Vision with OpenCV – FreeCodeCamp**
   OpenCV Course – Full Tutorial with Python
   URL: https://www.youtube.com/watch?v=oXlwWbU8l2o

6. **Action Recognition & Optical Flow – Nicholas Renotte**
   Human Action Recognition using Computer Vision
   URL: https://www.youtube.com/@NicholasRenotte

7. **Machine Learning Models – Krish Naik**
   Support Vector Machines, Random Forest, k-NN, Logistic Regression tutorials
   URL: https://www.youtube.com/@krishnaik06