Human Activity Recognition

Project Report

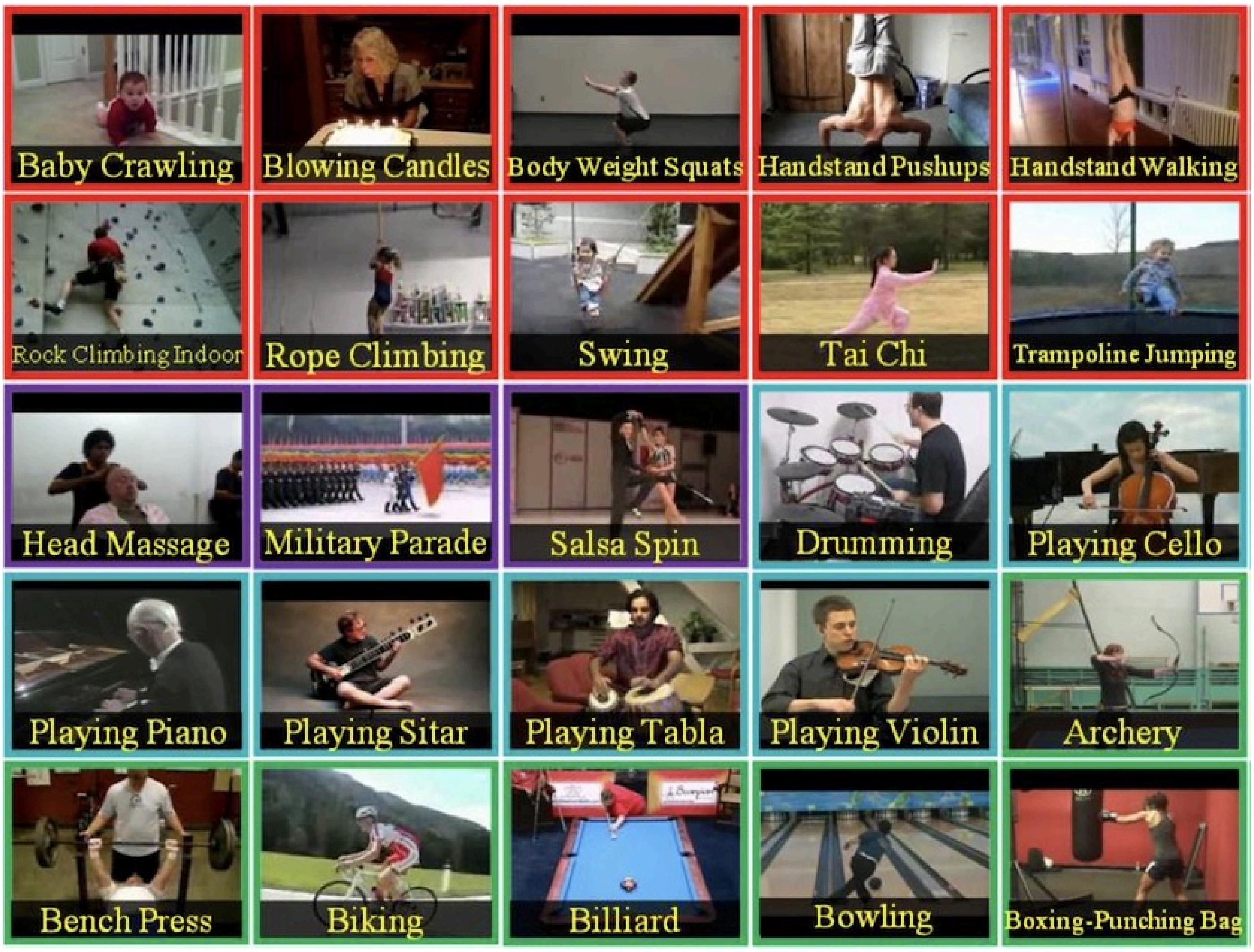
By Elamathi. R

Table of Contents

HUMAN ACTIVITY RECOGNITION

Introduction

Human Activity Recognition (HAR) is a field of artificial intelligence and computer vision that identifies and classifies human actions from video, sensor, or image data. With applications in healthcare, surveillance, sports analytics, and human-computer interaction, HAR enhances automation and assistive technologies.
Traditional HAR relied on handcrafted features and classical machine learning. Advances in deep learning introduced CNNs for spatial feature extraction, RNNs/LSTMs for temporal modeling, and transformers for capturing long-range dependencies. HAR is categorized into vision-based HAR, using deep learning on video data, and sensor-based HAR, relying on wearable sensors processed with machine learning.
Despite progress, challenges like viewpoint variations, occlusions, and environmental inconsistencies persist. Research focuses on developing robust, generalized models for improved real-time recognition and adaptability.



Problem Statement

Human Activity Recognition (HAR) is a crucial field in computer vision and artificial intelligence, aiming to automatically identify and classify human actions from video sequences. The ability to recognize activities accurately has widespread applications in healthcare, surveillance, sports analytics, human-computer interaction, and autonomous systems. However, achieving high recognition accuracy in real-world scenarios remains a challenging task due to variations in lighting, background clutter, viewpoint changes, and occlusions.
This project focuses on developing a deep learning-based HAR system capable of recognizing 10 specific human activities from video data. By leveraging custom CNN models, pretrained architectures (such as ResNet), and retrieval-augmented learning, the goal is to improve classification accuracy and make HAR models more robust and generalizable across diverse datasets.

Literature Study

Human Activity Recognition (HAR) is widely used in healthcare, security, sports analytics, human-computer interaction, and robotics. It enables patient monitoring, enhances surveillance, improves athletic performance analysis, and facilitates gesture-based interfaces. The increasing adoption of IoT sensors and deep learning has made HAR more efficient and applicable in real-world scenarios.

Recent advancements in deep learning have significantly improved HAR accuracy by extracting spatial and temporal features from video data. **H. Yang et al.,** in **Asymmetric 3D Convolutional Neural Networks for Action Recognition**, introduced 3D CNNs that capture both spatial and temporal dependencies, improving action recognition. Hybrid models combining CNNs and LSTMs further enhance performance by leveraging CNNs for spatial feature extraction and LSTMs for motion understanding. **El Mehdi Saoudi et al.,** in **Advancing Human Action Recognition**: **A Hybrid Approach Using Attention-Based LSTM and 3D CNN**, demonstrated that attention-based hybrid models improve recognition accuracy. More recently, transformer-based architectures have shown superior results in capturing long-range dependencies. **Khaled Alomar**, in **CNNs, RNNs, and Transformers in Human Action Recognition**: A Survey and a Hybrid Model, highlighted how self-attention mechanisms in transformers outperform traditional RNN-based models.

Despite these advancements, HAR faces challenges such as computational complexity, difficulty in distinguishing similar activities, and data imbalance. High computational demands limit deployment on edge devices, while minimal spatial differences make recognizing overlapping activities challenging. Class imbalance in datasets leads to biased model predictions, necessitating data augmentation techniques. Additionally, deep learning models lack transparency, making interpretation difficult. **Sanjay Jyoti Dutta et al.,** in **Human Activity Recognition: A Review of Deep Learning-Based Methods**, discuss efforts to integrate explainable AI techniques for better model interpretability. Addressing these challenges through self-supervised learning and retrieval-augmented generation can improve HAR's robustness and real-world applicability.

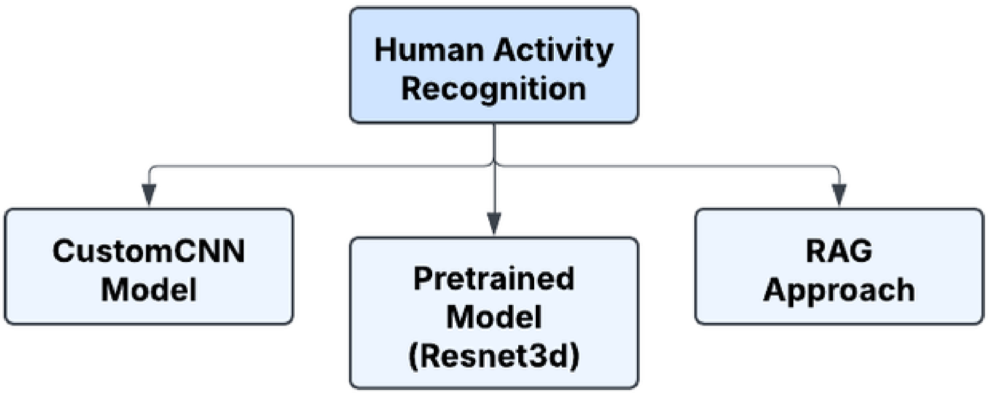
Objectives

The objective of this project is to develop an efficient Human Activity Recognition (HAR) system using deep learning. The key goals include:

- Custom CNN Model – Design and optimize a CNN architecture for accurate activity classification.
- Pretrained Model Comparison – Evaluate state-of-the-art models like ResNet against the custom CNN.
- Spatiotemporal Feature Extraction – Use 3D CNNs to capture both spatial and temporal motion patterns.
- Data Processing & Pretraining – Perform data preprocessing, frame extraction, and dataset preparation.
- Performance Optimization – Compare predictions with ground truth, refine architecture, and tune hyperparameters.
- Retrieval-Augmented Generation (RAG) – Integrate retrieval-based techniques for improved classification.

Address Challenges – Tackle issues like data imbalance, computational efficiency, and model generalization.

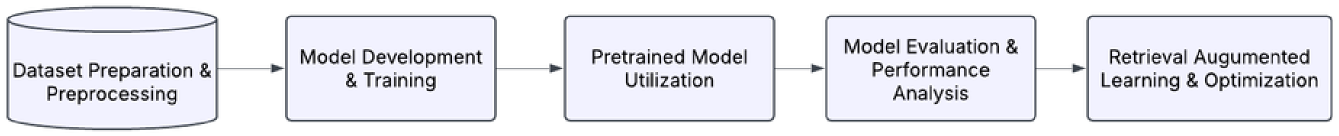
Proposed Approach



This project adopts a structured deep learning approach for Human Activity Recognition (HAR), focusing on dataset preparation, model development, evaluation, and optimization. The process begins with dataset preparation and preprocessing, where predefined activity labels are extracted from a Kaggle HAR dataset, and video sequences are converted into frames using MoviePy to create structured training data.

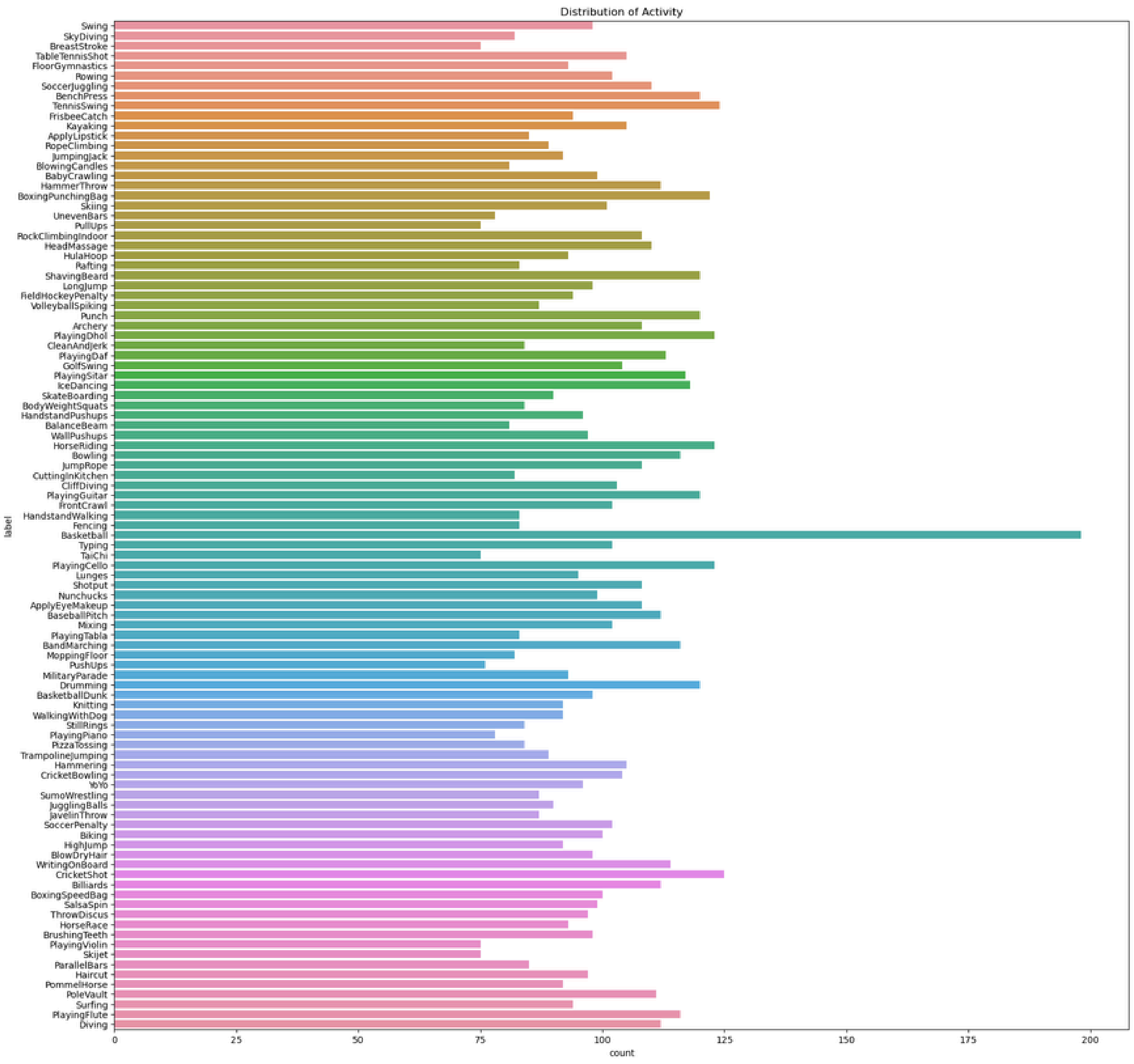
In model development and training, a custom CNN is designed and optimized to extract spatiotemporal features from video data. The model is trained and fine-tuned using deep learning frameworks to enhance its ability to recognize human activities. Additionally, pretrained models such as ResNet are implemented and fine-tuned to compare their performance with the custom CNN, ensuring the selection of the most effective architecture.

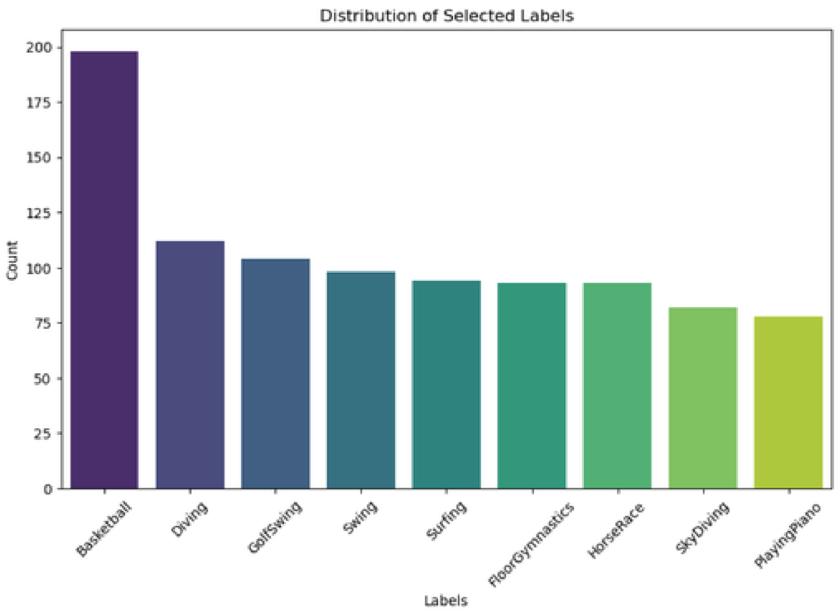
For model evaluation and performance analysis, predictions are validated against ground truth annotations, and performance is assessed using key metrics such as accuracy, precision, recall, F1-score, and confusion matrices. To further improve classification accuracy and generalization, retrieval-augmented learning (RAG) techniques are integrated, enabling the model to reference past data for better decision-making. Optimization strategies such as data augmentation and self-supervised learning are employed to improve feature representation, while computational efficiency techniques are applied to ensure real-time processing capabilities, making the HAR system more robust and scalable.

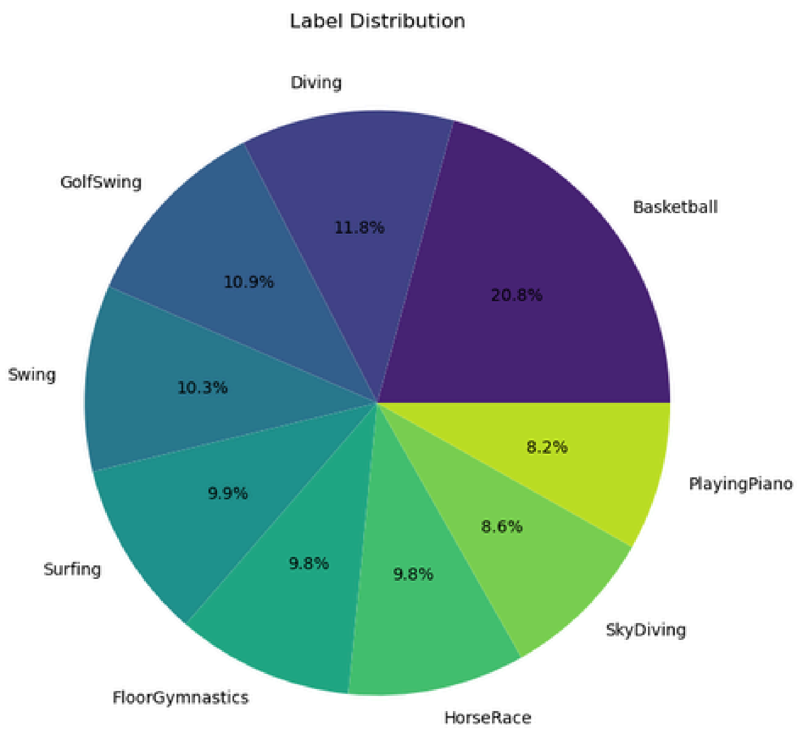

Dataset Preparation & Preprocessing
Model Development & Training
Pretrained Model Utilization
Model Evaluation & Performance Analysis
Retrieval Augumented Learning & Optimization

Dataset and Preprocessing

This project utilizes a video-based Human Activity Recognition (HAR) dataset from Kaggle, selecting 10 specific activity labels based on their distinct motion patterns: Swing, SkyDiving, FloorGymnastics, Diving, Surfing, Basketball, GolfSwing, PlayingPiano, PlayingGuitar, and HorseRace. The dataset was filtered into training, validation, and testing subsets, ensuring balanced class distribution and structured file organization for efficient processing.


Distribution of Activity
label
count
0
25
50
75
100
125
150
175
200
Swing
SkyDiving
BreastStroke
TableTennisShot
FloorGymnastics
Rowing
SoccerJuggling
BenchPress
TennisSwing
FrisbeeCatch
Kayaking
ApplyLipstick
RopeClimbing
JumpingJack
BlowingCandles
BabyCrawling
HammerThrow
BoxingPunchingBag
Skiing
UnevenBars
PullUps
RockClimbingIndoor
HeadMassage
HulaHoop
Rafting
ShavingBeard
LongJump
FieldHockeyPenalty
VolleyballSpiking
Punch
Archery
PlayingDhol
CleanAndJerk
PlayingDaf
GolfSwing
PlayingSitar
IceDancing
SkateBoarding
BodyWeightSquats
HandstandPushups
BalanceBeam
WallPushups
HorseRiding
Bowling
JumpRope
CuttingInKitchen
CliffDiving
PlayingGuitar
FrontCrawl
HandstandWalking
Fencing
Basketball
Typing
TaiChi
PlayingCello
Lunges
Shotput
Nunchucks
ApplyEyeMakeup
BaseballPitch
Mixing
PlayingTabla
BandMarching
MoppingFloor
PushUps
MilitaryParade
Drumming
BasketballDunk
Knitting
WalkingWithDog
StillRings
PlayingPiano
PizzaTossing
TrampolineJumping
Hammering
CricketBowling
YoYo
SumoWrestling
JugglingBalls
JavelinThrow
SoccerPenalty
Biking
HighJump
BlowDryHair
WritingOnBoard
CricketShot
Billiards
BoxingSpeedBag
SalsaSpin
ThrowDiscus
HorseRace
BrushingTeeth
PlayingViolin
Skijet
ParallelBars
Haircut
PommelHorse
PoleVault
Surfing
PlayingFlute
Diving


Distribution of Selected Labels
Count
0
25
50
75
100
125
150
175
200
Basketball
Diving
GolfSwing
Swing
Surfing
FloorGymnastics
HorseRace
SkyDiving
PlayingPiano
Labels



Frame Extraction and Preprocessing

Since HAR relies on spatiotemporal features, each video was processed into 16 frames per sequence to capture essential motion patterns. MoviePy and OpenCV were used to extract frames at regular intervals, with additional preprocessing steps including:

- Resizing frames to a fixed resolution of 128×128 pixels for consistency.
- Normalizing pixel values between 0 and 1 to enhance model stability.

Handling shorter videos by duplicating the last frame to maintain a fixed sequence length.

Label Encoding and Dataset Creation

To facilitate supervised learning, Label Encoding was applied, mapping each activity class to a unique numerical value. A PyTorch Dataset class (VideoDataset) was developed to:

- Dynamically load and preprocess frames from video files.
- Apply data transformations for standardized inputs.

Return frame sequences with corresponding numerical labels for model training.
Additionally, a PyTorch DataLoader was implemented for efficient batch processing, optimizing memory usage and computational speed, especially for GPU acceleration. By ensuring structured data preparation, this preprocessing pipeline enhances the HAR model's ability to learn meaningful activity patterns and generalize across various human actions.

Model Development

Custom CNN Model

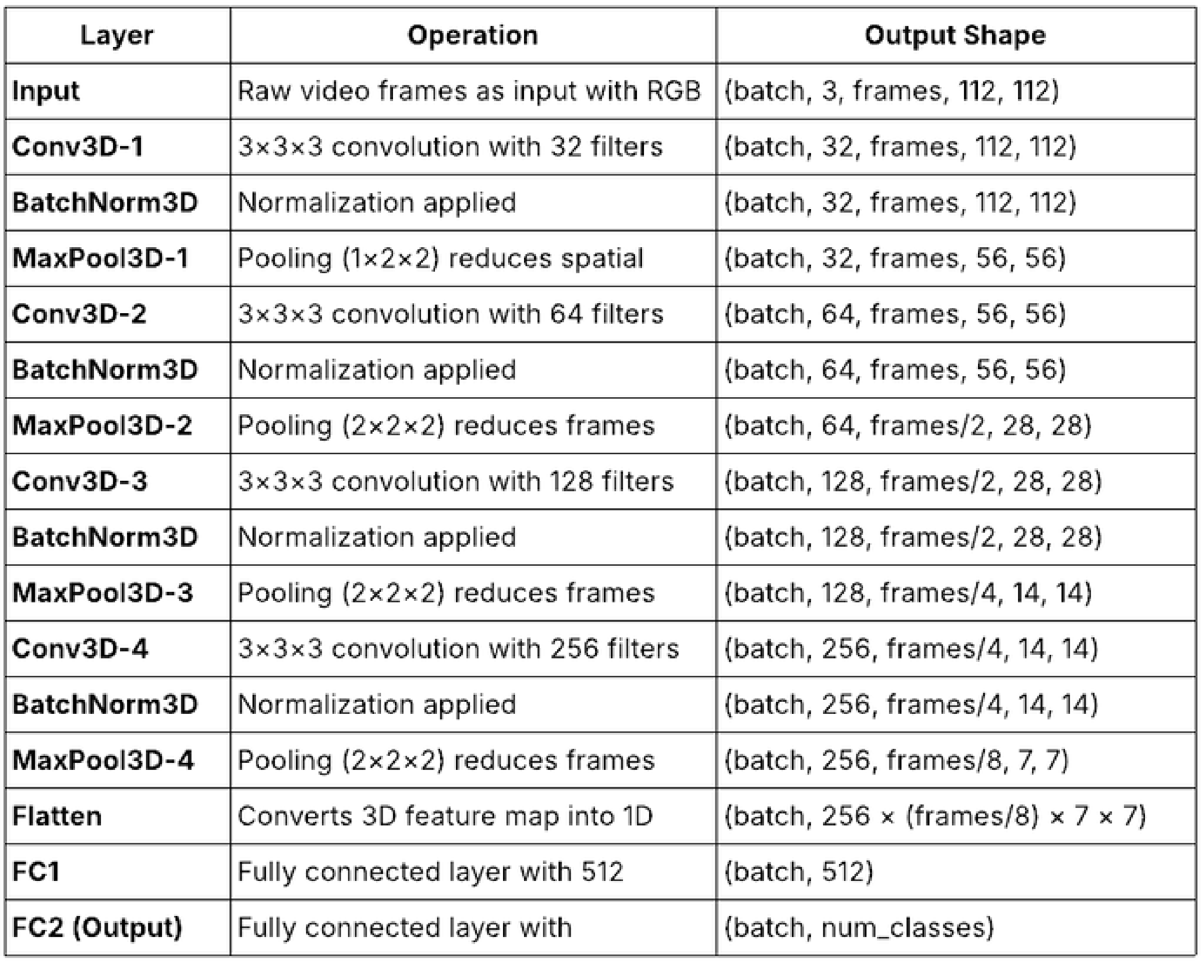
The model is designed to process video data using 3D Convolutional Neural Networks (3D-CNNs). It extracts spatial and temporal features from video frames while progressively reducing dimensions.

Architecture Details and Design Choices
The architecture is designed to efficiently process video data by leveraging 3D convolutional layers, batch normalization, max pooling, and fully connected layers to extract spatial and temporal features. The model begins with an input layer that accepts video clips with RGB channels. It then undergoes multiple convolutional and pooling operations to progressively refine feature extraction and reduce dimensionality.

The first convolutional layer applies 32 filters of size 3×3×3, capturing spatial and motion-related features. This is followed by batch normalization, which stabilizes training and normalizes feature distributions. Max pooling with a 1×2×2 kernel reduces the spatial dimensions, preserving key structural information while minimizing computational costs.

The second convolutional layer increases the number of filters to 64, enabling deeper feature extraction. The same pattern of batch normalization and pooling follows, reducing the spatial and temporal dimensions further. By the third convolutional layer, 128 filters process the data, refining feature hierarchies for action recognition or object tracking. The model continues with a fourth convolutional layer of 256 filters, further abstracting complex patterns in the video.

After feature extraction, the data is flattened into a 1D vector, making it suitable for classification. A fully connected layer with 512 neurons learns high-level feature representations, followed by a final output layer with neurons corresponding to the number of target classes. This design ensures efficient feature learning while preventing overfitting through batch normalization and max pooling operations.



Layer	Operation	Output Shape
Input	Raw video frames as input with RGB	(batch, 3, frames, 112, 112)
Conv3D-1	3×3×3 convolution with 32 filters	(batch, 32, frames, 112, 112)
BatchNorm3D	Normalization applied	(batch, 32, frames, 112, 112)
MaxPool3D-1	Pooling (1×2×2) reduces spatial	(batch, 32, frames, 56, 56)
Conv3D-2	3×3×3 convolution with 64 filters	(batch, 64, frames, 56, 56)
BatchNorm3D	Normalization applied	(batch, 64, frames, 56, 56)
MaxPool3D-2	Pooling (2×2×2) reduces frames	(batch, 64, frames/2, 28, 28)
Conv3D-3	3×3×3 convolution with 128 filters	(batch, 128, frames/2, 28, 28)
BatchNorm3D	Normalization applied	(batch, 128, frames/2, 28, 28)
MaxPool3D-3	Pooling (2×2×2) reduces frames	(batch, 128, frames/4, 14, 14)
Conv3D-4	3×3×3 convolution with 256 filters	(batch, 256, frames/4, 14, 14)
BatchNorm3D	Normalization applied	(batch, 256, frames/4, 14, 14)
MaxPool3D-4	Pooling (2×2×2) reduces frames	(batch, 256, frames/8, 7, 7)
Flatten	Converts 3D feature map into 1D	(batch, 256 × (frames/8) × 7 × 7)
FC1	Fully connected layer with 512	(batch, 512)
FC2 (Output)	Fully connected layer with	(batch, num_classes)

Training, Validation, and Testing

To ensure optimal performance in video classification, the model undergoes three critical phases: training, validation, and testing. The entire training pipeline is implemented using PyTorch, leveraging the Adam optimizer, cross-entropy loss, and batch normalization for improved learning efficiency.

Training Phase

The training phase involves multiple epochs, where the model learns from data using backpropagation and an optimization algorithm. The key steps are as follows:

Data Loading & Processing

Input video frames are loaded in mini-batches using a PyTorch DataLoader for efficient processing.

Forward Propagation

The model extracts spatiotemporal features through 3D convolutions, batch normalization, max pooling, and fully connected layers.

Loss Calculation

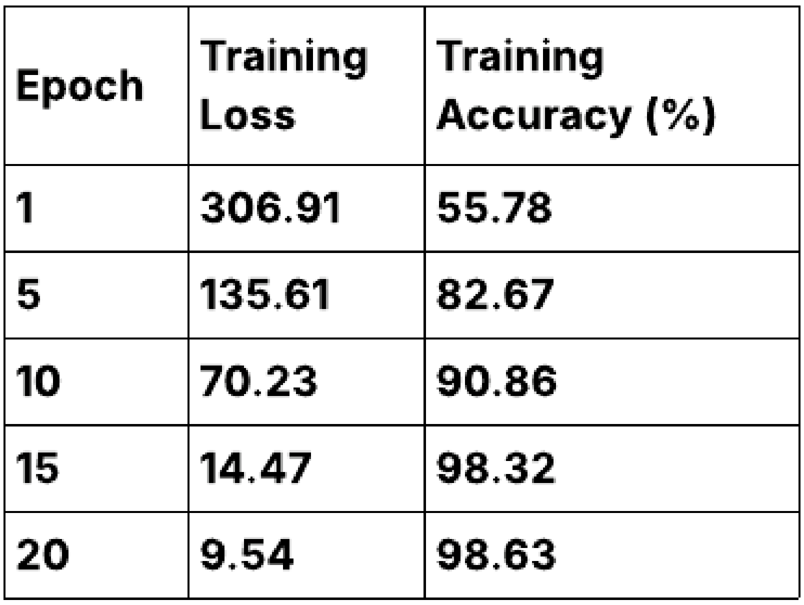
The cross-entropy loss function is used for multi-class classification, comparing model predictions with ground-truth labels.

Backpropagation & Optimization

The Adam optimizer with weight decay updates model weights based on computed loss.
optimizer.zero_grad() is called before backpropagation to clear previous gradients.The computed gradients are propagated backward, updating parameters to improve accuracy.

Training Performance Evaluation

The model's training accuracy is calculated by comparing predicted labels with actual ground-truth labels. A structured evaluation is maintained over multiple epochs to monitor progress.



Epoch	Training Loss	Training Accuracy (%)
1	306.91	55.78
5	135.61	82.67
10	70.23	90.86
15	14.47	98.32
20	9.54	98.63

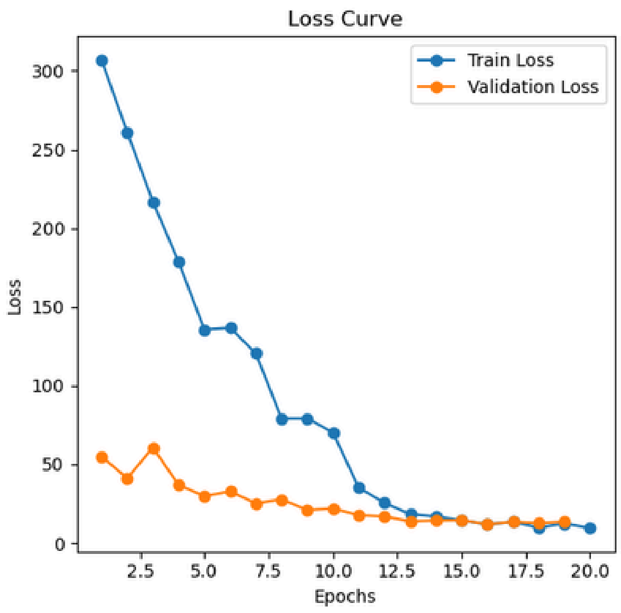
Validation Phase
The validation phase assesses the model’s ability to generalize to unseen data, ensuring it does not suffer from underfitting or overfitting.

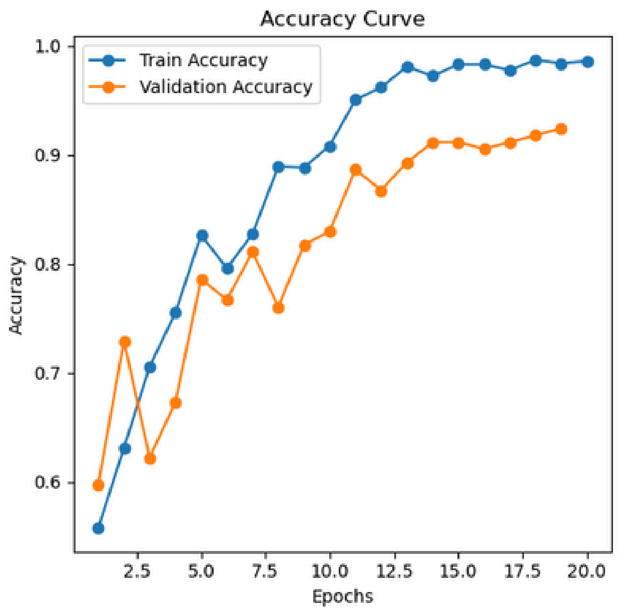
Data Processing
The validation dataset is passed through the model in evaluation mode **(model.eval())**, disabling dropout and batch normalization updates.

Loss & Accuracy Computation
The validation loss and accuracy are recorded to monitor generalization.

Learning Rate Adjustment
A learning rate scheduler adjusts the learning rate dynamically for efficient convergence.

Epoch	Validation Loss	Validation Accuracy (%)
1	55.07	59.75
5	29.88	78.62
10	21.97	83.02
15	14.58	91.19
20	13.16	90.57





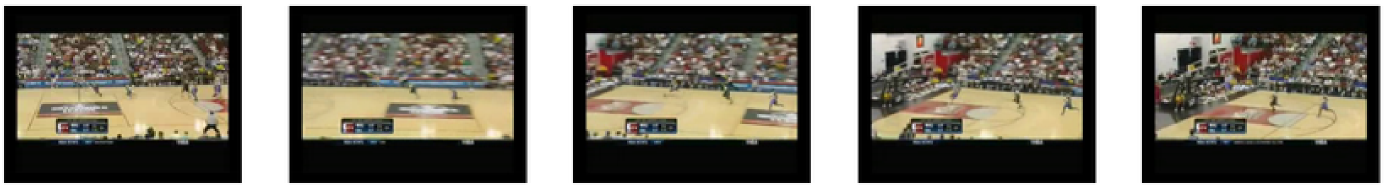
Testing Phase
The final testing phase evaluates the trained model on a completely unseen dataset, ensuring an unbiased assessment of its performance.

Final Inference on Test Data
The trained model is used to classify test samples, and predictions are compared against actual ground-truth labels.

Performance Metrics
The test accuracy is computed as the ratio of correct predictions to total test samples.

Final Test Accuracy
The model achieved a test accuracy of 93.29%, demonstrating strong classification capability.

Actual: Basketball | Predicted: Basketball



Actual: PlayingPiano | Predicted: PlayingPiano

Actual: SkyDiving | Predicted: SkyDiving

Pretrained Model (ResNet3D) for Video Classification

Using a pretrained model such as ResNet3D significantly improves efficiency in video classification tasks by leveraging feature representations learned from large-scale datasets like Kinetics-400. Training a deep learning model from scratch requires substantial computational resources, extensive labeled data, and prolonged training time, making it less feasible for many applications. By using ResNet3D, the training process is accelerated since the model has already learned fundamental spatiotemporal features, requiring only fine-tuning on the specific dataset. Additionally, pretrained models enhance generalization, reducing the risk of overfitting when working with smaller datasets. Unlike a custom 3D CNN that demands extensive training, ResNet3D can effectively adapt with limited data while maintaining high accuracy. Furthermore, its optimized deep architecture, incorporating residual connections, ensures stable gradient flow, allowing deeper networks to be trained more efficiently.

Understanding the Pretrained Model (ResNet3D)
ResNet3D, an advanced extension of the traditional ResNet architecture, is specifically designed for video-based tasks, incorporating 3D convolutions to capture both spatial (image-level) and temporal (motion-level) features. Unlike 2D CNNs that process individual frames separately, ResNet3D effectively extracts motion information across consecutive frames, making it highly suitable for video classification. Its key features include 3D convolutional layers that enhance spatiotemporal feature extraction, residual learning to mitigate the gradient vanishing problem and enable deeper network training, and batch normalization to stabilize learning and prevent overfitting. Since the pretrained ResNet3D model was originally trained on the large-scale Kinetics-400 dataset, we adapted it for our specific dataset by replacing the final fully connected layer to match the number of activity classes. This modification ensured that the model could generalize well to our dataset while maintaining the advantages of pretrained feature representations.

Training
To fine-tune ResNet3D, the model was trained using an **Adam optimizer with weight decay**, which helps prevent overfitting while ensuring stable updates. The **cross-entropy loss function** was used for multi-class classification.

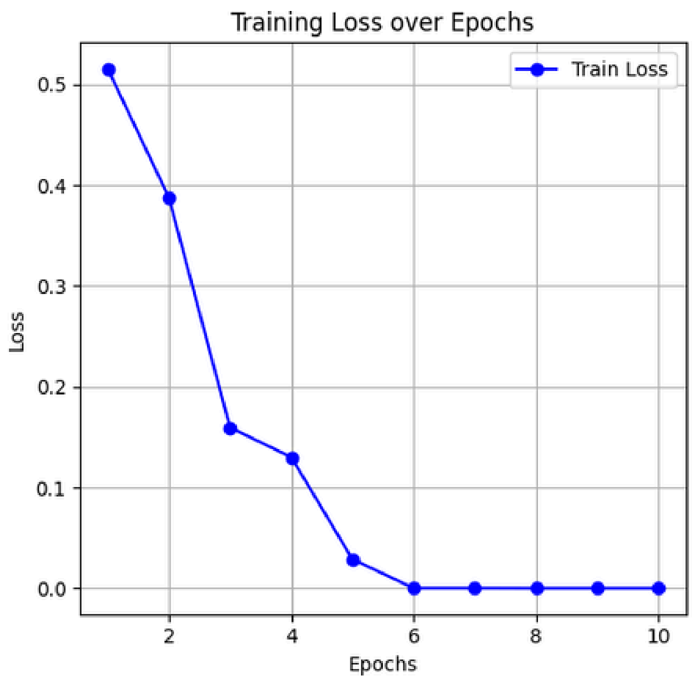
1. **Dataset Preparation:** Video data was preprocessed into short clips and structured into batches for training.
2. **Batch Normalization:** Applied after each convolution layer to stabilize training and improve convergence.
3. **Learning Rate Scheduling:** The learning rate was dynamically adjusted to enhance performance and prevent overfitting.
4. **Backpropagation & Optimization:** The model parameters were updated using backpropagation and the Adam optimizer to minimize classification errors.

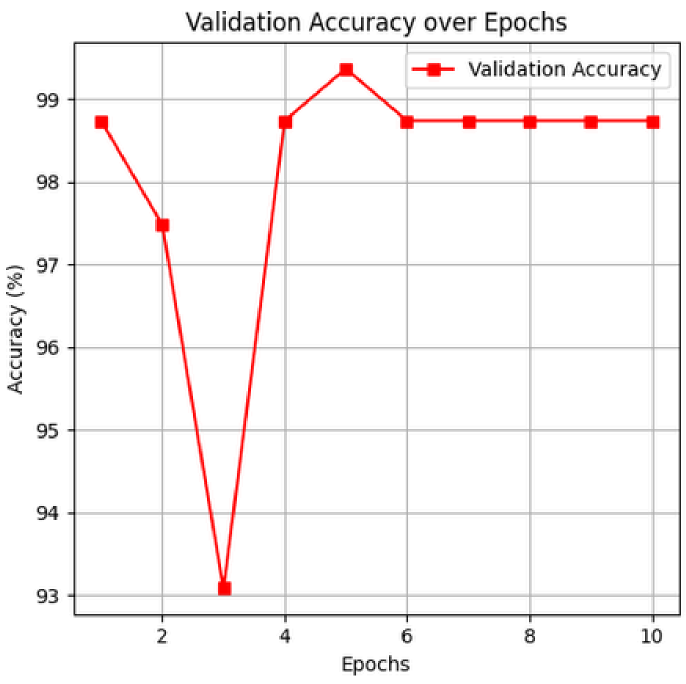
During training, accuracy and loss metrics were monitored to ensure the model was learning effectively. Over multiple epochs, the training accuracy progressively improved while the loss decreased, confirming the effectiveness of the pretrained architecture.

Validation

After each training epoch, the model was validated on a separate dataset to measure its ability to generalize to unseen data. Key performance metrics such as **validation accuracy and loss** were recorded. The validation phase ensured that the model was neither underfitting nor overfitting.

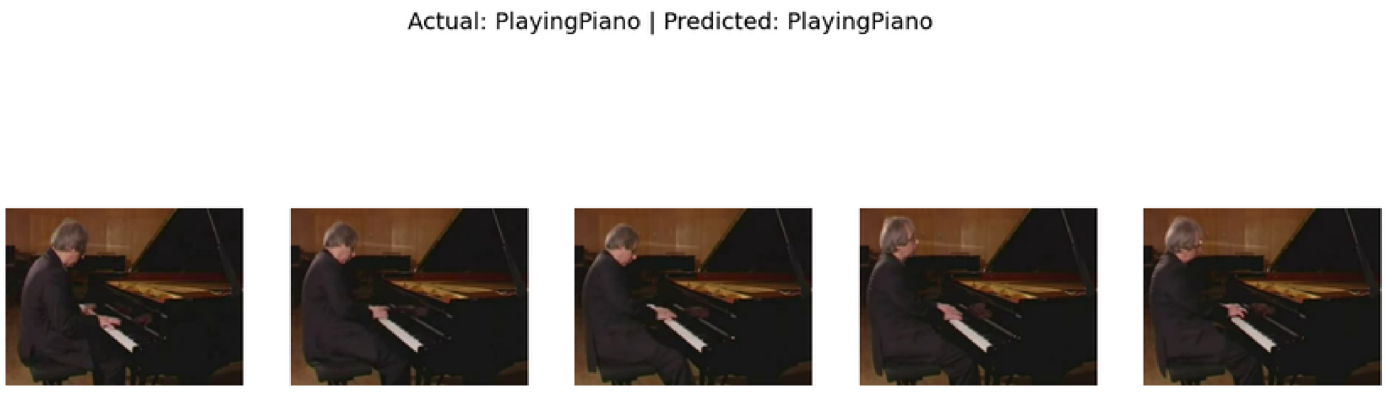
The accuracy steadily improved across epochs, demonstrating that the pretrained model effectively adapted to our dataset. The validation accuracy reached **above 98%** by the final epochs, confirming strong generalization.





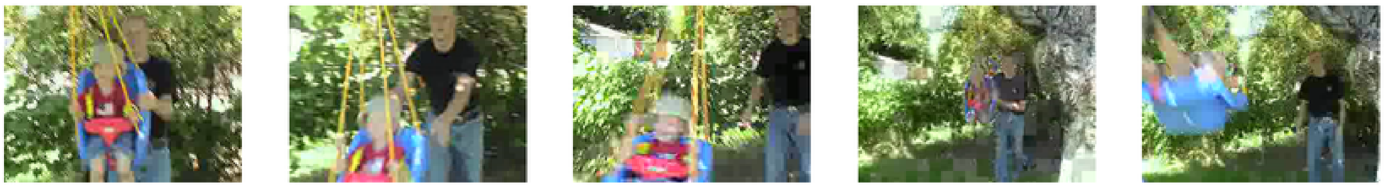
Testing Phase

Once training and validation were completed, the final model was tested on a completely unseen test dataset. This step provided a realistic measure of the model's performance in real-world scenarios. The model's predictions were compared against ground truth labels, and the final **test accuracy exceeded 98%**, demonstrating the effectiveness of fine-tuning ResNet3D for our classification task.



Actual: PlayingPiano | Predicted: PlayingPiano

Actual: Swing | Predicted: Swing



Comparison with a Custom 3D CNN

Feature	**ResNet3D (Pretrained)**	**Custom 3D CNN**
Training Time	Faster (Pretrained weights reduce epochs)	Slower (Training from scratch)
Data Requirement	Less data needed (Generalizes well)	Requires large dataset (High risk of overfitting)
Feature Extraction	Strong (Pretrained on Kinetics-400)	Weak (Learns from scratch)
Generalization	High (Well-optimized architecture)	Lower (Needs more fine-tuning)
Accuracy	>98% (After fine-tuning)	~85% (With longer training)
Computational Cost	Lower (Efficient training)	Higher (More epochs, more resources)

Implementation of Retrieval-Augmented Generation (RAG) in Human Activity Recognition (HAR)

Justification for Implementing RAG After a Pretrained Model

Traditional deep learning models, such as ResNet-50, rely solely on learned representations from training data. While these models are effective for classifying common activities, they often struggle with ambiguous, rare, or previously unseen activities. This limitation occurs because the model lacks the ability to adapt to new information beyond its training set.

RAG addresses this issue by integrating a retrieval mechanism that fetches relevant contextual information from an external knowledge base. This approach enhances prediction accuracy by incorporating real-time, domain-specific knowledge that complements the model's learned features. In the context of **Human Activity Recognition (HAR)**, RAG improves model generalization, reduces misclassification errors, and enhances interpretability by providing justifications for predictions.

Feature Extraction Using ResNet-50

To enable similarity-based retrieval, the first step is extracting **high-dimensional feature embeddings** from video frames. A pretrained **ResNet-50** model is used for this purpose, as it effectively captures spatial hierarchies and motion patterns essential for HAR.

The process involves preprocessing images by resizing, normalizing, and converting them into tensors compatible with ResNet-50. Once passed through the network, the model extracts **deep feature representations**, which are then stored as **NumPy arrays** for efficient retrieval. These embeddings serve as the foundation for similarity search in the RAG pipeline.

Feature extraction using ResNet-50 is crucial for dimensionality reduction while preserving meaningful information. By representing activities as compact feature vectors, the system can efficiently compare and retrieve similar instances based on feature similarity.

Building a FAISS Index for Efficient Retrieval

To facilitate fast and scalable similarity searches, a **FAISS (Facebook AI Similarity Search) index** is constructed using the extracted ResNet-50 embeddings. FAISS is an optimized nearest-neighbor search library that enables efficient retrieval in large-scale datasets.

The process begins with loading the precomputed embeddings and converting them into a **FAISS-compatible format**. A **L2 distance-based similarity search** is implemented to ensure accurate retrieval of visually similar activity frames. The FAISS index, along with corresponding image paths, is then stored for future queries. FAISS significantly enhances retrieval performance by enabling sub-millisecond similarity searches, making it ideal for handling millions of activity frames. This capability ensures that the system can efficiently find relevant activities for refining predictions in HAR.

Querying FAISS for Similar Frame Retrieval

With the FAISS index in place, the next step is to retrieve similar frames for any given query image. This process involves selecting a random frame from the dataset, extracting its feature embedding using ResNet-50, and performing a nearest-neighbor search in FAISS to identify the most similar activity frames.

Random frame selection is used as a benchmark to evaluate the retrieval accuracy of the FAISS index. By comparing the retrieved frames with the query frame, the system validates its ability to identify **semantically similar activities**. This capability is particularly useful for HAR, as it allows the model to reference past instances and refine its classification based on prior knowledge.

Activity Prediction Using ResNet-50

To classify activities, a **pretrained ResNet-50 model** is fine-tuned on an activity dataset. The prediction process involves preprocessing the query image, extracting deep features, and using a classification layer to assign the most probable activity label.

ResNet-50 is chosen for its high accuracy in image classification tasks and its ability to extract hierarchical representations of activities. By leveraging its feature extraction capabilities, the model effectively identifies key motion patterns and spatial configurations necessary for HAR.

Activity Knowledge Base for RAG

A structured **knowledge base** is developed to store textual descriptions of various human activities. This knowledge base serves as an additional source of contextual information, allowing the RAG system to retrieve relevant explanations when making activity predictions.

The knowledge base is integrated with a language model, such as **Gemini API**, to generate **detailed activity descriptions**. These descriptions provide interpretability by explaining why a particular activity was classified in a specific way. This integration bridges the gap between **visual recognition** and **text-based understanding**, making the HAR system more transparent and informative.

Implementing RAG in Human Activity Recognition enhances model performance by incorporating an external retrieval mechanism that dynamically refines predictions. Feature extraction using **ResNet-50** enables efficient representation of activities, while FAISS indexing ensures fast and accurate retrieval of similar frames. By integrating a **knowledge base** and **language model**, RAG provides contextual explanations, improving both classification accuracy and interpretability.
This approach significantly enhances HAR systems, making them more adaptable to real-world scenarios where new or ambiguous activities frequently arise.

Challenges Faced

- The dataset had class imbalances, leading to biased predictions.
- Training ResNet-50 on a large dataset was resource-intensive.
- Limited hardware availability impacted performance.

Potential Improvements

- A larger and more diverse dataset would enhance model robustness. High-performance GPUs would enable faster training and better fine-tuning.

- Advanced architectures like Vision Transformers (ViTs), 3D CNNs, and LSTMs could improve feature extraction and temporal learning. Hybrid models (CNN + RNN) could further enhance accuracy.
- Future research could explore multimodal learning (video, audio, and sensor data integration), self-supervised learning to reduce dependency on labeled data, and optimization for edge devices to enable real-time applications.

Future Directions for HAR Research

- Multimodal learning (combining video with audio/sensor data).
- Self-supervised learning to reduce dependence on labeled data.
- Optimizing models for edge devices for real-time applications.

Conclusion

This study explored a hybrid approach to video classification, integrating a **custom CNN model** and a **pretrained ResNet-50 model** to enhance feature extraction and classification accuracy. While the custom CNN provided a strong baseline, the transition to a pretrained model significantly improved generalization. Further improvements were achieved using **Retrieval-Augmented Generation (RAG)**, which leveraged retrieval-based techniques for more robust activity recognition. The overall approach demonstrated practical applications in real-world scenarios such as surveillance, healthcare, and sports analytics. Future research can focus on optimizing model efficiency, incorporating multimodal learning, and improving real-time performance.

References

Dutta, S. J., Boongoen, T., & Zwiggelaar, R. (2025). *Human activity recognition: A review of deep learning-based methods*. First published: 01 February 2025. https://doi.org/10.1049/cvi2.70003.

Srinivasan, V. (2025). *Activity Recognition Using Deep Learning in Videos under Clinical Setting*.
Saoudi, E. M., Jaafari, J., & Andaloussi, S. J. (2025). *Advancing human action recognition: A hybrid approach using attention-based LSTM and 3D CNN*.

Yang, H., Yuan, C., Li, B., Du, Y., Xing, J., Hu, W., & Maybank, S. J. (2019). *Asymmetric 3D convolutional neural networks for action recognition. Pattern Recognition, 85*, 1-12. https://doi.org/10.1016/j.patcog.2018.07.028.

Krishna, M. L. S., Yeleswarapu, A., R, J., & Khan, M. Z. (2025). *Human Activity Recognition Using Machine Learning Techniques*.

Alomar, K. (2025). *CNNs, RNNs and Transformers in Human Action Recognition: A Survey and a Hybrid Model*.