THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

DATS 6303
Deep Learning

Human Action Recognition
Project Proposal

Akshit Reddy Palle, Ritu Patel,
Dhatrija Sukasi, Ayush Meshram,
Shaik Mohammed Mujahid Khalandar

1. Problem Statement and Rationale

Human Action Recognition (HAR) is a key computer vision task that involves identifying and classifying human activities—such as walking, jumping, or clapping—from still RGB images. It has broad applications in surveillance, sports analytics, rehabilitation, and human–computer interaction.

Recognizing human actions from static images is challenging due to variations in lighting, body posture, and background context. Traditional feature-based methods often fail to capture these complex spatial patterns, making deep learning approaches far more effective.

This project aims to develop a deep learning–based image classification system that accurately distinguishes between 15 categories of human actions. By training and comparing multiple architectures, we seek to determine which models most effectively capture spatial features for reliable visual activity recognition.

2. Dataset

We will use the Human Action Recognition (HAR) Dataset sourced from Kaggle, created by Meet Nagadia.

The dataset contains over 12,000 labeled RGB images depicting individuals performing 15 distinct human activities, such as walking, clapping, cycling, and jumping. Each activity class is stored in a separate folder, making it well-suited for deep learning–based image classification tasks.

All images contain a single person performing one action, allowing for clear multi-class categorization. The dataset is sufficiently large and diverse to train and evaluate deep neural networks effectively. It will be divided into training, validation, and testing sets, with standard image preprocessing techniques applied, including resizing, normalization, and augmentation, to enhance generalization and model robustness.

3. Deep Learning Networks

3.1 Custom CNN (Baseline Model)

A simple convolutional network built from scratch will serve as our baseline. It will include multiple convolutional and pooling layers followed by a fully connected layer for 15-class classification. The model will help benchmark performance and highlight improvements achieved through transfer learning.

3.2 VGG16 (Transfer Learning)

We will fine-tune the VGG16 architecture pretrained on ImageNet. Its deep, sequential convolutional layers capture spatial features effectively, making it suitable for human pose and activity recognition. The final layer will be modified for 15-class classification.

3.3 ResNet50

We will fine-tune ResNet50, a deep residual network pretrained on ImageNet. Its skip connections allow stable training of deeper layers, improving feature learning and reducing overfitting. The final classification layer will be adjusted for 15 activity classes.

3.4 DCAM-Net

We will implement DCAM-Net, a convolutional network that integrates dual channel attention mechanisms. This model enhances feature representation by emphasizing important spatial and channel information, helping the network focus on key human pose regions. It will be trained and evaluated alongside other architectures for performance comparison.

3.5 InceptionV3

We will fine-tune InceptionV3, a pretrained convolutional architecture that captures both fine-grained and large-scale spatial features using multi-branch convolutional filters. Its ability to analyze features at multiple receptive fields makes

it effective for recognizing diverse human actions from still images.

3.6 EfficientNet / Vision Transformer

We will experiment with EfficientNet and Vision Transformer (ViT) as advanced deep learning models. EfficientNet optimizes accuracy and efficiency through compound scaling, while ViT leverages self-attention to model global spatial relationships. Both will be fine-tuned for the 15 activity classes to evaluate their performance against traditional CNN architectures.

4. Framework

Framework:

- **Pytorch** - chosen for its flexibility, strong GPU support, and availability of pretrained models through TorchVision.

Libraries:

- **NumPy, Pandas -** data manipulation and analysis.
- **OpenCV, Pillow -** image preprocessing and augmentation.
- **Matplotlib,Seaborn -** visualization of training and evaluation results.
- **scikit-learn –** computation of accuracy, precision, recall, and F1-score metrics.

5. Reference Material

The following key references will be used to support model selection and framework implementation:

1. Nagadia, M. (2023). *Human Action Recognition (HAR) Dataset*. Kaggle.

2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition (ResNet)*.

3. Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*.

4. PyTorch Documentation – official framework reference for model development.

6. Performance Evaluation and Metrics

Model performance will be evaluated using standard metrics for multi-class image classification to ensure fair and comprehensive comparison across all architectures.

6.1 Evaluation Metrics:

- **Accuracy:** Overall proportion of correctly classified images.
- **Precision, Recall, and F1-Score:** Per-class and macro-averaged measures of classification quality.
- **Confusion Matrix:** Visualization of true versus predicted labels to analyze misclassifications.
- **Training Time and Model Size:** Assessment of computational efficiency and scalability.

6.2 Evaluation Approach:

Each model will be trained and tested on the same dataset split to maintain consistency. Performance metrics will be compared to identify the most effective architecture in terms of both accuracy and efficiency.

7. Rough Schedule

Week 1 : Data preparation and preprocessing

- Download and organize the dataset.
- Perform exploratory analysis.
- Apply basic preprocessing techniques such as resizing, normalization, and augmentation.

Week 2 : Baseline Model Training

- Implement and train the Custom CNN to establish baseline performance.
- Begin fine-tuning VGG16 and ResNet50 models.

Week 3 : Advanced architectures:

- Implement and train DCAM-Net, InceptionV3, and EfficientNet/ViT.
- Adjust hyperparameters and optimize model performance.

Week 4 : Demo and deployment:

- Evaluate final models, visualize results, and deploy the best-performing model for real-time or batch inference as part of the project demonstration.