Video Action Recognition with UCF50 using LRCN

Shakuntala Mitra

May 1, 2025

1 Introduction

This project presents a video action classification pipeline developed using the UCF50 dataset. The model architecture is based on a Long-term Recurrent Convolutional Network (LRCN), which leverages a ResNet-based convolutional neural network for spatial feature extraction from individual frames, followed by a Long Short-Term Memory (LSTM) network to capture temporal dependencies across video sequences.

2 Data

2.1 UCF50 Dataset

The UCF50 dataset is a very widely used benchmark dataset for video action recognition, comprised of realistic video clips sourced from YouTube. It includes video clips belonging to 50 distinct action categories, with each category organized into 25 groups of action clips. Each group contains at least four video clips that often share common characteristics, such as the same subject, similar backgrounds, or consistent camera viewpoints. This structure introduces intraclass variability while preserving a level of visual consistency within the video groups. This feature makes the UCF50 dataset a suitable benchmark for evaluating both spatial and temporal modeling capabilities in video action recognition systems. The 50 action categories span a large range of human activities, including sports (e.g., Baseball Pitch, Basketball Shooting, Golf Swing), fitness exercises (e.g., Push Ups, Pull Ups, Jumping Jack), musical performances (e.g., Playing Guitar, Playing Violin, Playing Tabla), and other common daily life actions (e.g., Pizza Tossing, Mixing Batter, Walking with a Dog, YoYo).

2.2 Preprocessing and Augmentation

To prepare the UCF50 dataset for training, a two-stage data preprocessing pipeline was implemented: frame extraction, followed by image transformations.

2.2.1 Frame Extraction

Each raw video clip was processed using OpenCV to uniformly sample a fixed number of 16 frames per clip, regardless of video length. This uniform sampling ensures temporal consistency while standardizing the input sequence length across all samples. The selected frames were converted to RGB format and saved as JPEG images, grouped by action class. This structure supports efficient loading during training and aligns with the input expectations of the model.

2.2.2 Image Transformations

Following frame extraction, image transformations were applied to augment the sample diversity and enhance the generalizability of the model. The model architecture- a Long-term Recurrent Convolutional Network (LRCN)- expected frame inputs resized to 224×224 pixels.

The full **training** transformation pipeline was as follows:

1. Resize images to 224x224 resolution
2. Random horizontal flipping (50% probability)
3. Random affine transformations (translation up to 10%)
4. Pixel-value normalization using the mean and standard deviation values
5. Convert to PyTorch Tensors

The **validation and test** transformation pipeline did not include the data augmentations:

1. Resize images to 224x224 resolution
2. Pixel-value normalization using the mean and standard deviation values
3. Convert to PyTorch Tensors

The frame extraction ensured both temporal uniformity across video sequences and spatial normalization of individual frames. The training data augmentations helped the model learn more robust representations by introducing controlled variability into the training data.

3 Methods

3.1 Long-Term Recurrent Convolutional Network

The model architecture used for the video classification is a Long-Term Recurrent Convolutional Network (LRCN). This model design combines spatial and temporal modeling through the integration of the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN).

A pre-trained ResNet backbone was used to extract spatial features from each individual frame. The final fully connected layer of ResNet was replaced with an identity function to output high-dimensional features instead of class logits. The sequence of frame-wise features was passed through a multi-layer LSTM network to capture temporal dependencies. The LSTM was configured with a specified hidden size and number of layers, followed by a dropout layer for regularization. The final hidden state of the LSTM was passed through a fully connected layer to output logits corresponding to the 50 action classes.

3.2 Training Procedure

The training procedure was designed as a modular pipeline that can be configured by passing arguments to the command line. The dataset was divided into training, validation, and test subsets using stratified sampling to preserve the distribution of the classes between the splits. Cross-entropy loss was used as the loss function for multiclass classification. The Adam optimizer was used for gradient-based optimization. The 'ReduceLROnPlateau' learning rate scheduler was used to monitor the changes in the validation loss and reduce the learning rate when the improvements plateaued. The training and evaluation was performed using the GPU, but still took a very long time to complete.

Model training was carried out over 30 epochs, with a batch size of 8 and a learning rate of $3e^{-5}$. The training process demonstrated steady convergence and improving generalization (based on validation performance) throughout the process.

The model achieved substantial gains in validation accuracy early on, rising from 50.22% after the first epoch to 71.45% by epoch 4, and surpassing 85% by epoch 10. The best validation accuracy of **92.53%** was achieved in epoch 29, with a corresponding validation loss of 0.3933 and a training loss of 0.0587, indicating effective learning. Training loss steadily decreased over the epochs, while validation performance remained robust even as learning rates decreased.

The final model weights were selected based on the best validation accuracy. The high model performance demonstrated both stability and strong generalizability.

3.3 Action Classification Results

The model achieved an overall test accuracy of 91.03%, indicating strong generalization to the unseen video samples. The model demonstrated high precision and recall across most of the 50 action classes.

There were a few classes where the model achieved a perfect F1-score of 1.00, meaning that the class predictions were essentially flawless. These classes were: BenchPress, Billiards, PlayingGuitar, PlayingTabla, PlayingViolin, PommelHorse, and TrampolineJumping. Other classes for which the model performed exceptionally well were CleanAndJerk (F1: 0.97), Drumming (F1: 0.98), and BreastStroke (F1: 0.97).

The model did seem to struggle with some action classes that had more visually ambiguous features, or for which the motions bore a lot of similarity with those of other actions. For example, RopeClimbing vs. RockClimbingIndoor had lower recall. This is consistent with the known challenges of fine-grained action classification. The confusion matrices for each class indicate that most misclassifications involved only one or two incorrect predictions per class, though.

The final aggregate performance metrics are as follows:

Metric	Final Value
F1-score	0.91
Precision	0.91
Recall	0.91

Table 1: Aggregate Performance Metrics for Action Classification

The model had very balanced performance across all 50 classes, meaning that the performance did not skew towards any particular class or group of classes. The high precision and recall values shows that the model is able to reliably recognize actions, with very few false positives or missed samples.