BBM416 FUNDAMENTALS OF COMPUTER VISION FINAL PROJECT REPORT

Ataberk Asar, 2210356135 & Aysu Aylin Kaplan, 2200356810
Department of Computer Engineering
Hacettepe University

ABSTRACT

Developing a system for workout type identification and repetition counting, aiming to enhance accuracy and applicability in fitness analysis. This project will be helpful for people who want to improve their fitness, as well as for trainers and doctors who want to keep track of their patients' exercises.

1 INTRODUCTION

Our project aims to develop a computer vision system for workout identification and repetition counting. Traditional methods for tracking workout often rely on human supervision and wearable technologies, which can be infeasible and often prone to human error. Additionally wearable devices may not always detect exercises and count repetitions accurately, leading to incorrect data. By automating this process via computer vision, we provide more accurate feedback to the users. Main challenges include movement variation of exercises, precise movement detection to ensure accurate repetition counting, and real-time processing capability to provide immediate feedback.

2 METHOD

2.1 OVERVIEW

To count repetitions in sports activities, our project employs Google's RepNet, a deep learning model designed to identify repetitive patterns in video sequences. The approach uses self-similarity matrix based transformers to analyze video frames and predict the repetitive elements, ultimately leading to accurate repetition counting. For action recognition for the sport activity type detection, our project utilizes optical flow for frame extraction method and Long Short-Term Memory (LSTM) which is a type of recurrent neural network (RNN) architecture designed to model sequential data, capturing dependencies over long time intervals. This section describes our approach, the modifications made, and the process used to derive results.

2.2 REPNET ARCHITECTURE

RepNet relies on a deep convolutional neural network (CNN), using a modified ResNet50 backbone, to extract temporal features from video sequences. The extracted features are fed into a self-similarity matrix and subsequently processed through a multi-headed transformer architecture to estimate repetition periods. The model output provides an estimated repetition count and a within-period score, which indicates whether a frame is part of a repetitive segment.

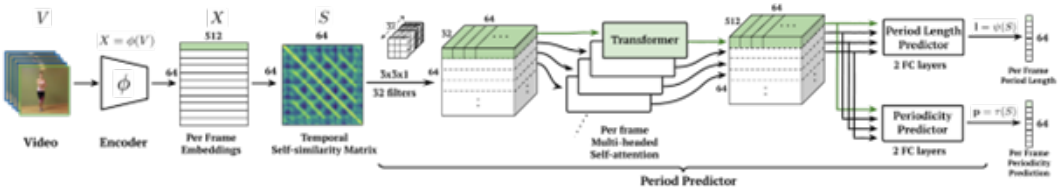

Figure 1: RepNet architecture

2.3 Workout Action Recognition Model

The model architecture combines two streams of input data: one for image features and one for optical flow, processing them through a series of convolutional layers and then merging them before feeding the result into an LSTM layer and a dense layer for classification.

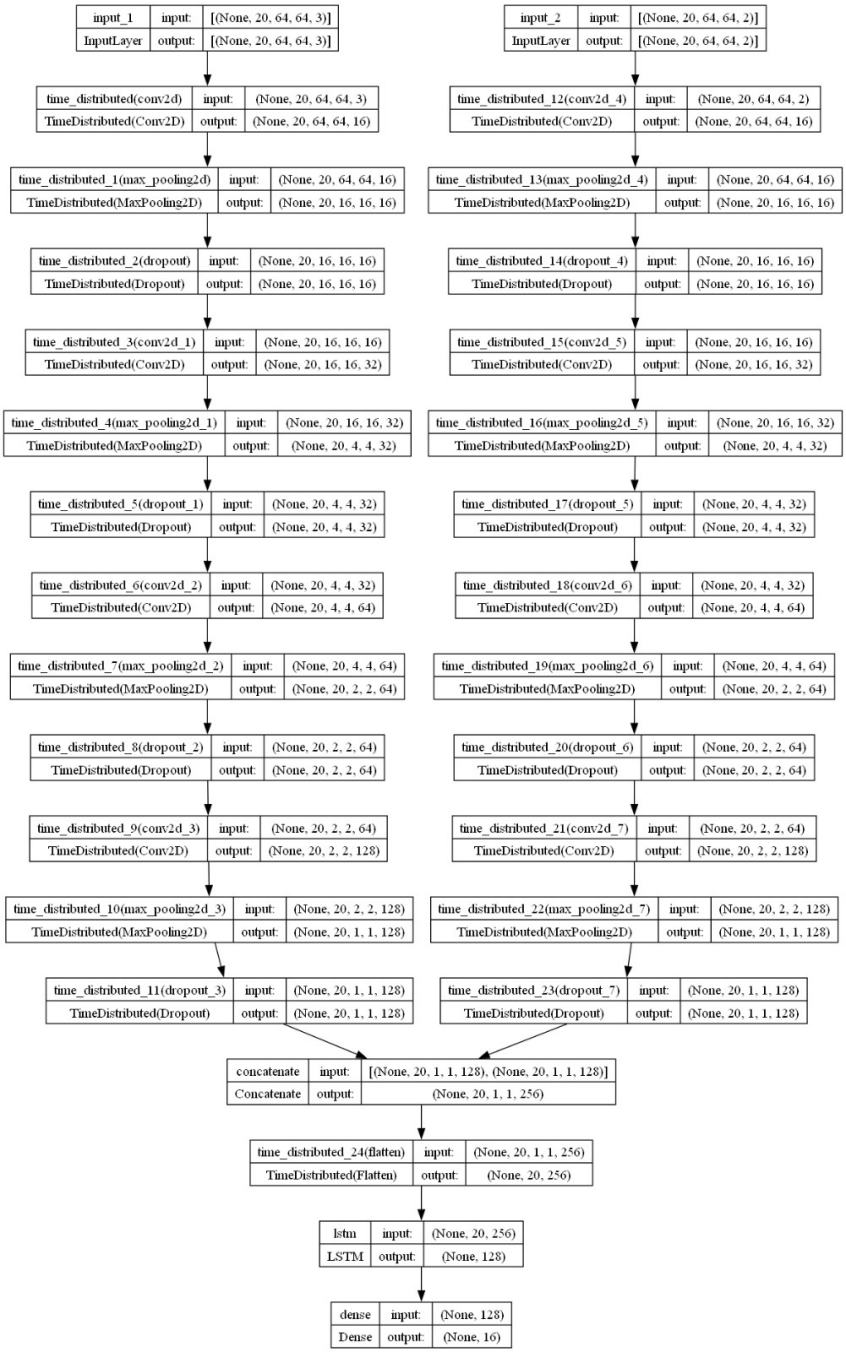

Figure 2: Workout Action Recognition Architecture

Image features capture spatial aspects of the video, while optical flow features capture the motion between consecutive frames.

The Time Distributed Convolutional Layers extract spatial features from each frame independently. These layers detect edges, textures, and patterns within each frame, reduce spatial dimensions through max-pooling, and use dropout to prevent overfitting. This is done separately for both RGB frames and optical flow frames. After extracting these features, the model concatenates them, allowing it to utilize both spatial and motion information simultaneously.

The concatenated features are then flattened and fed into an LSTM lLayer. Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to model sequential data, capturing dependencies over long time intervals. LSTM networks are especially effective for tasks where context from previous time steps is crucial.

2.4 PREPROCESSING

To prepare the video data for RepNet, frames are resized and normalized. Our pipeline processes each frame by resizing it to a standard 112x112 pixels to ensure consistency with RepNet's architecture. Additionally, pixel values are normalized to be within a range suitable for deep learning model inputs.

In the architecture of the Workout Action Recognition model, we handle two types of input data: image features and optical flow. Both inputs are sequences of images, where each image in the sequence is processed independently using convolutional neural networks (CNNs). The 'TimeDistributed' wrapper is used to apply the same convolutional operations to each frame in the sequence. Below is a detailed explanation of the preprocessing steps:

- *Image Features Input:*

```
feature_inp = Input(shape=(SEQUENCE_LENGTH, HEIGHT, WIDTH, 3))
The `feature_inp` tensor represents a sequence of RGB images,
where each image has a shape of `(HEIGHT, WIDTH, 3)`.
```

- *Optical Flow Input:*

```
optical_flow_inp = Input(shape=(SEQUENCE_LENGTH, HEIGHT, WIDTH, 2))
The `optical_flow_inp` tensor represents a sequence of optical flow
images, where each image has a shape of `(HEIGHT, WIDTH, 2)`
corresponding to the flow in the x and y directions.
```

The preprocessing steps ensure that both the spatial and temporal aspects of the input data are effectively captured and utilized by the model for accurate classification.

2.5 EXPERIMENTAL SETTINGS

To evaluate our approach, we utilize a dataset consisting of sports videos, focusing on those with clear repetitive activities such as weightlifting, jumping, and running.Repetition Action Counting Dataset(1) has been used for the training, validation and test data for the exercise action recognition model. We assess the model's performance using various metrics, including repetition accuracy, confidence score, and processing efficiency. Python, and Tensorflow used for implementation of machine learning algorithms, and Gradio used as GUI framework.

2.6 EXPERIMENTAL RESULTS

2.6.1 ACTION DETECTION

Exercise Action Detection Model trained the Repetition Action Counting dataset.

To optimize the performance of our video action recognition model, we applied hyperparameter tuning using grid search. This method involved systematically exploring a predefined set of hyperparameters, including learning rates, batch sizes, and optimizers, to identify the best combination for our model. We run our test on a subset of entire dataset.

By testing various learning rates, we aimed to find the optimal balance between convergence speed and training stability. We also experimented with various optimizers, such as Adam, RMSprop, and SGD, to identify the one that best suited our model's learning dynamics. This comprehensive grid search allowed us to fine-tune our model, enhancing its accuracy and generalization performance on the video classification task.

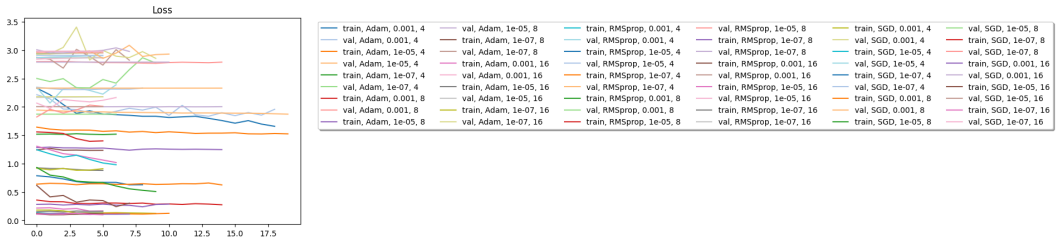

Figure 3: Loss Tests

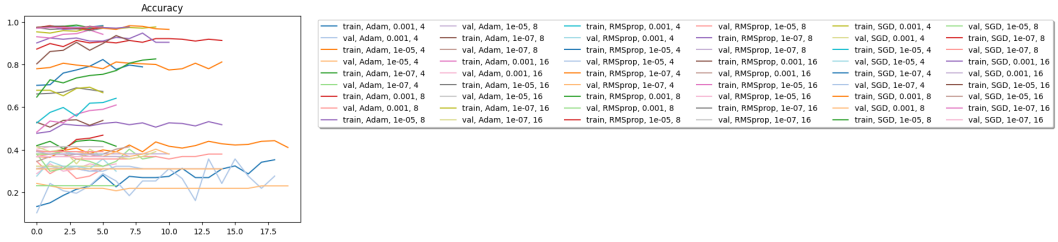

Figure 4: Accuracy Tests

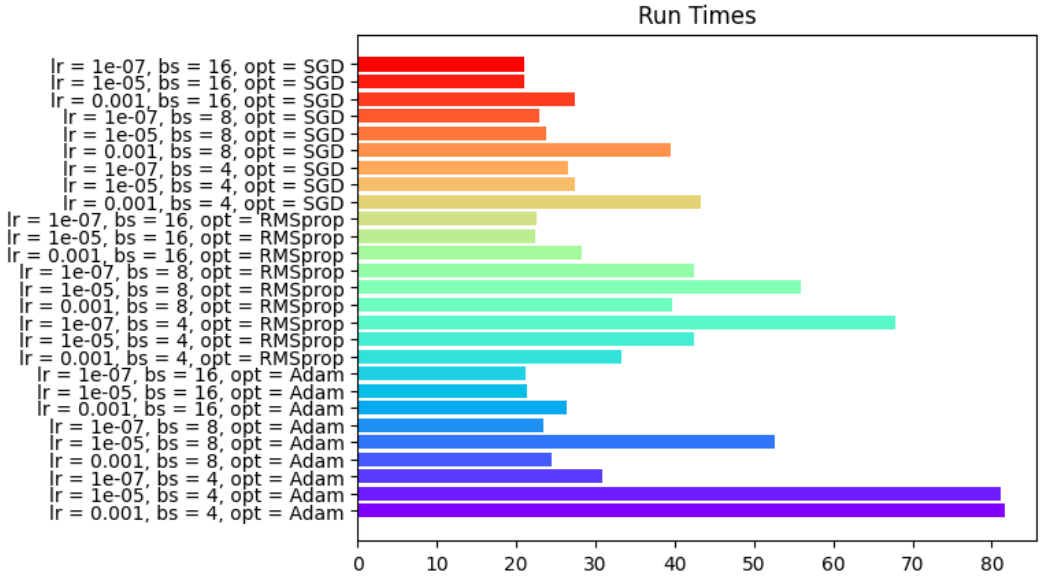

Figure 5: Runtime Tests

The actual training process involves using early stopping to prevent overfitting, and model checkpointing to save the best model based on validation performance. We use best model hyperparameters from previous grid search step.

By combining the strengths of CNNs for spatial feature extraction, optical flow for motion information, and LSTMs for temporal pattern recognition, the model can effectively recognize complex actions in video sequences.

Below is the loss and accuracy plots of the training of final action detection model.

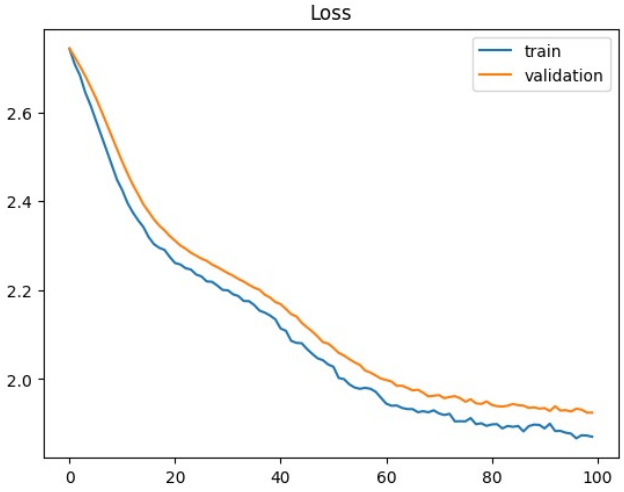

Figure 6: Loss Graph

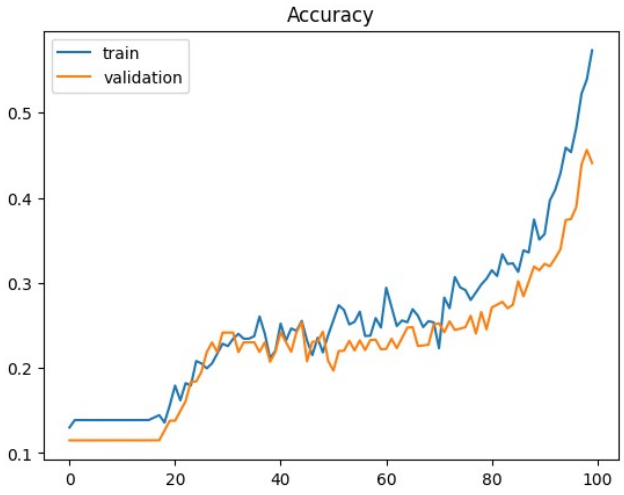

Figure 7: Accuracy Graph

2.6.2 Repetition Detection and Scoring

The core of our method involves processing the frame sequences through RepNet to obtain two crucial outputs:
Period Predictions: These indicate the estimated length of the repetitive cycles.
Within-Period Scores: These represent the likelihood of a frame belonging to a repetitive segment.

Using these outputs, we calculate:
Repetition Count: The cumulative sum of repetition rates over the video length, accounting for

stride and period predictions.
Confidence Score: A composite score based on the within-period predictions and raw period estimates. This score indicates the model's confidence in its repetition detection.

Demo Results:

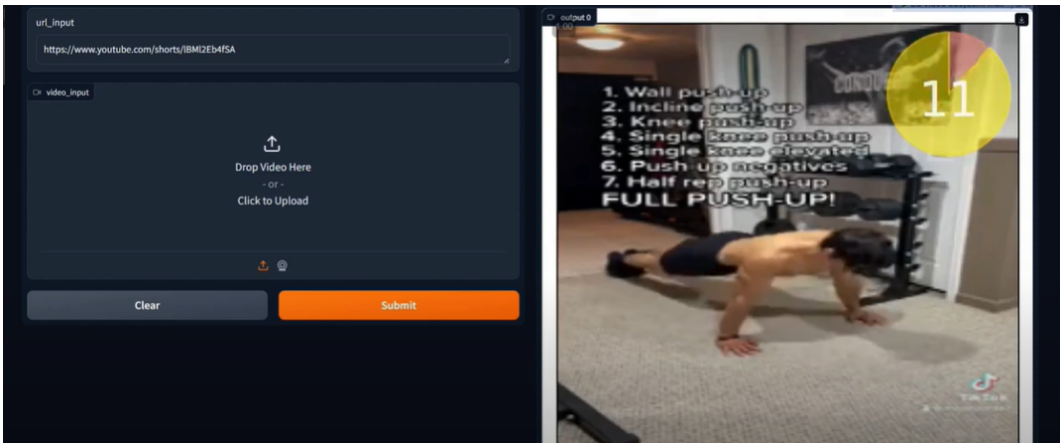

Figure 8: Repetition Counting

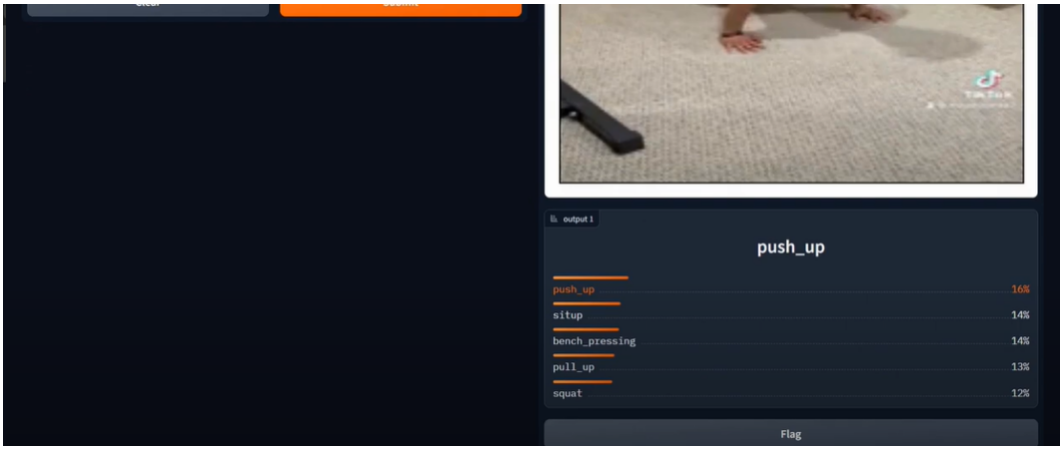

Figure 9: Action Recognition

Demo Video:
https://www.youtube.com/watch?v=A-GUBdgqG-U

Model Weights:
https://drive.google.com/file/d/1SpzsQcu1HqlhECYaFfR7HLXJ7c2fzUtu/view?usp=sharing

2.7 DISCUSSIONS/CONCLUSIONS

The project outcomes show great progress in recognizing actions in videos and also include counting repetitions using RepNet. Adding RepNet, which is made to count repeated actions in videos, gave our model more useful features. This lets the application not only classify actions but also accurately count repetitions, which is very useful for sports and exercise tracking.

The best parts of the project are the use of CNNs for finding features in each frame, LSTMs for understanding how actions change over time, and RepNet for counting repetitions. These elements work well together to improve the system's performance.

However, we faced challenges like high computational demands, long training times, and integrating different models.

To make the results even better, we could try techniques like transfer learning, better data augmentation, and more efficient optimization methods. Overall, the project is successful, showing strong action recognition and repetition counting abilities, which are important for practical uses.

REFERENCES

[1] https://svip-lab.github.io/dataset/RepCount_dataset.html

[2] https://sites.google.com/view/repnet