REAL-TIME HUMAN ACTION RECOGNITION

PMLDL Project

Kaggle: Real-Time Human Action Recognition

Authors
Vaycheslav Koshman, Yaryeva Dinara, Stepanova Anastasiia
v.koshman@innopolis.university
d.yaryeva@innopolis.university
aa.stepanova@innopolis.university
Innopolis University, Sep, 2023

Contents

1 Introduction

The field of computer vision has witnessed significant advancements in recent years, enabling machines to perceive and understand the visual world with increasing accuracy. One crucial aspect of this progress is the ability to recognize and interpret human actions in real-time. Real-time human action recognition has emerged as a fundamental research area with numerous applications, ranging from surveillance and security systems to human-computer interaction and virtual reality.

The objective of this course project is to develop a robust and efficient system for real-time human action recognition. By leveraging state-of-the-art computer vision techniques and machine learning algorithms, we aim to design a system that can accurately identify and classify various human actions in real-time video streams. This project will not only enhance our understanding of human behavior but also contribute to the development of intelligent systems capable of perceiving and responding to human actions in real-world scenarios.

In this report, we will provide a comprehensive overview of the project, including the motivation behind choosing this topic, the methodology employed, the dataset used, and the results obtained. We will also discuss the challenges encountered during the project and propose potential avenues for future research and improvement.

The report is structured as follows: In Section 2, we will review the existing literature and state-of-the-art approaches in the field of real-time human action recognition. Section 3 will outline the methodology adopted for this project, including the preprocessing steps, feature extraction techniques, and the classification algorithm employed. Section 4 will present the dataset used for training and evaluation, along with the data augmentation techniques employed. In Section 5, we will discuss the experimental setup and present the results obtained, including accuracy, precision, recall, and F1-score. Section 6 will highlight the challenges faced during the project and propose potential solutions. Finally, in Section 7, we will conclude the report by summarizing the key findings and discussing future directions for research.

Overall, this project aims to contribute to the advancement of real-time human action recognition systems, with the ultimate goal of enabling machines to understand and interpret human actions in real-world scenarios. By developing an accurate and efficient system, we hope to pave the way for a wide range of applications that can benefit from real-time human action recognition, ultimately enhancing human-computer interaction and improving the overall efficiency and safety of various domains.

2 Literature review

To start, we decided to compare different machine learning methods for addressing an existing objective. In our literature review section, we've divided the content into two parts: one covering more scientific solutions and the other focusing on practical, real-world approaches. This approach helps us explore machine learning techniques from a variety of sources.

2.1 Scientific researches

First, we started by searching for new ideas in the latest research solutions. We used well-known research databases like Google Scholar. Our focus was mainly on the most recent solutions because the field of Machine Learning evolves very quickly. Even just a few years of progress can make a big difference in a solution's quality.

According to [1, 2], the future of DL for human action recognition:

- Developing unsupervised learning models
- Deeper CNNs
- Combining different deep learning models
- Fusion of hand-crated and deep learning solutions
- Transfer learning

Let us briefly discuss some of the approaches.

In the literature of human action recognition based on DL, CNNs seem to be the most important model for learning spatio-temporal features directly from RGB videos without pre-processing. However pure CNN solutions are not able to grasp temporal relations between video frames. To cope with this, some notably innovative architectures have used 3D convolutional filters to extract motion features[3, 4].Also transfer learning can significantly improve the results of a CNN-based architecture using a pre-trained deep CNN and fine-tuning it for the current objective[5, 6, 3].

LSTM-RNNs are successful in recognizing human actions because they can use a complete history of motion frames. However, they can't work directly with raw data. So, many researchers use CNNs to extract color features and then feed them into LSTM for learning and prediction[3]. There are two ways to combine LSTM and CNN: LRCN and ConvLSTM. LRCN has CNN layers whose output goes into LSTM units. ConvLSTMs integrate time series and computer vision using special cells in an LSTM layer. Interestingly, according to several articles[4], LRCN tends to perform better than ConvLSTM in various situations.

While the inputs are normally frames, researchers performed 3D convolution operations to add the temporal information in order to recognize videos. Additionally some of the approaches could also be used to generate features for different classifiers like optical flow[7].

Transformer models are famous for their ability to handle complex tasks, and they can be made very large. Some new models like MotionBERT and UniFormer combine visual and time-related information, similar to models that understand both text and images[8, 9].

Another one transformer model is ViT. Instead of looking at a whole picture at once, it chops the image into smaller pieces called patches. These patches are then processed by a Transformer network, which helps understand how these pieces relate to each other. This approach helps ViT understand images better[10, 11].

Below, you'll find a link to the table listing all the resources we used with some notes.

link

2.2 Practical solutions

After conducting a literature review, we explored various solutions available for real-time human action recognition. To narrow down our search, we utilized the Kaggle competition platform, which yielded 1,814 results for the query ”human action”. We then selected a dataset to work with based on the analysis of 232 datasets found on the site https://paperswithcode.com/. After reviewing the datasets mentioned in the literature, we decided to use the UCF dataset, which was frequently referenced. We found over 200 notebooks related to UCF dataset on Kaggle, but only 14 had results that met our requirements. We used these works for further analysis.

2.2.1 Notebooks analysis

As mentioned in the literature review, the CNN (Convolutional Neural Network) is a crucial component of video processing models. It is worth to mention, that all the models we found utilized the Convolutional level in their structure.

The use of pure CNN for binary classification yielded excellent results, with an accuracy of 0.9952 [12]. For 10 classes, the accuracy was 0.89 [13]. However, when a 3D Convolutional Network model was used for 101 classes from the UCF-101 dataset, the accuracy dropped to 0.12 [14].

ConvLSTM emerged as the most popular solution for the human action recognition task, with an average classification result of approximately 0.806 accuracy [15, 16, 17, 18, 19].

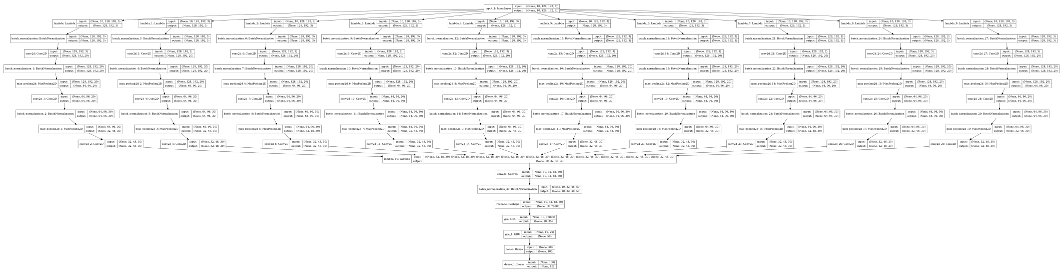
input_1: InputLayer input: [(None, 10, 128, 192, 3)] output: [(None, 10, 128, 192, 3)]
lambda: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_1: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_2: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_3: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_4: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_5: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_6: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_7: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_8: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
lambda_9: Lambda input: (None, 10, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_3: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_6: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_9: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_12: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_15: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_18: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_21: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_24: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
batch_normalization_27: BatchNormalization input: (None, 128, 192, 3) output: (None, 128, 192, 3)
conv2d: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_3: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_6: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_9: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_12: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_15: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_18: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_21: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_24: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
conv2d_27: Conv2D input: (None, 128, 192, 3) output: (None, 128, 192, 20)
batch_normalization_1: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_4: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_7: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_10: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_13: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_16: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_19: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_22: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_25: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
batch_normalization_28: BatchNormalization input: (None, 128, 192, 20) output: (None, 128, 192, 20)
max_pooling2d: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_2: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_4: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_6: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_8: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_10: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_12: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_14: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_16: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
max_pooling2d_18: MaxPooling2D input: (None, 128, 192, 20) output: (None, 64, 96, 20)
conv2d_1: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_4: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_7: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_10: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_13: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_16: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_19: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_22: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_25: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
conv2d_28: Conv2D input: (None, 64, 96, 20) output: (None, 64, 96, 30)
batch_normalization_2: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_5: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_8: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_11: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_14: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_17: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_20: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_23: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_26: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
batch_normalization_29: BatchNormalization input: (None, 64, 96, 30) output: (None, 64, 96, 30)
max_pooling2d_1: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_3: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_5: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_7: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_9: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_11: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_13: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_15: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_17: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
max_pooling2d_19: MaxPooling2D input: (None, 64, 96, 30) output: (None, 32, 48, 30)
conv2d_2: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_5: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_8: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_11: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_14: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_17: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_20: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_23: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_26: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
conv2d_29: Conv2D input: (None, 32, 48, 30) output: (None, 32, 48, 30)
lambda_10: Lambda input: [(None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30), (None, 32, 48, 30)] output: (None, 10, 32, 48, 30)
conv3d: Conv3D input: (None, 10, 32, 48, 30) output: (None, 10, 32, 48, 50)
batch_normalization_30: BatchNormalization input: (None, 10, 32, 48, 50) output: (None, 10, 32, 48, 50)
reshape: Reshape input: (None, 10, 32, 48, 50) output: (None, 10, 76800)
gru: GRU input: (None, 10, 76800) output: (None, 10, 25)
gru_1: GRU input: (None, 10, 25) output: (None, 50)
dense: Dense input: (None, 50) output: (None, 100)
dense_1: Dense input: (None, 100) output: (None, 10)

Figure 1: The scheme of a model, which combines CNN branches with GRU.

One of the most successful methods involved combining 10 parallel branches of ConvLSTM models with GRU (Gated Recurrent Unit), as shown in Figure 1. This approach achieved an average accuracy of 0.925 [18, 19].

Several models utilized pre-trained models as layers or input-to-embeddings transformers. For example, using ResNet101 and LSTM as the pre-final layer resulted in an accuracy of 0.995 [20]. Wrapping EfficientNet in a Time Distributed layer yielded an accuracy of 0.987 [21].

An unusual solution involved using a transformer from Facebook named timesformer-base-finetuned-k400 as embeddings for a model consisting of only 3 dense layers [22]. Surprisingly, this approach achieved one of the best results, with an accuracy of 0.94.

Another solution (accuracy of 0.9255) utilized a transformer with ResNet101 as the encoder, LSTM as the decoder, and an attention function [23].

3 Methodology

Based on the results of the analysis of existing solutions, we decided to repeat the results of the model (scheme on the fig 2) that combined ResNet and LSTM models [20] and maybe compare the approach with the transformer, implemented in [23].

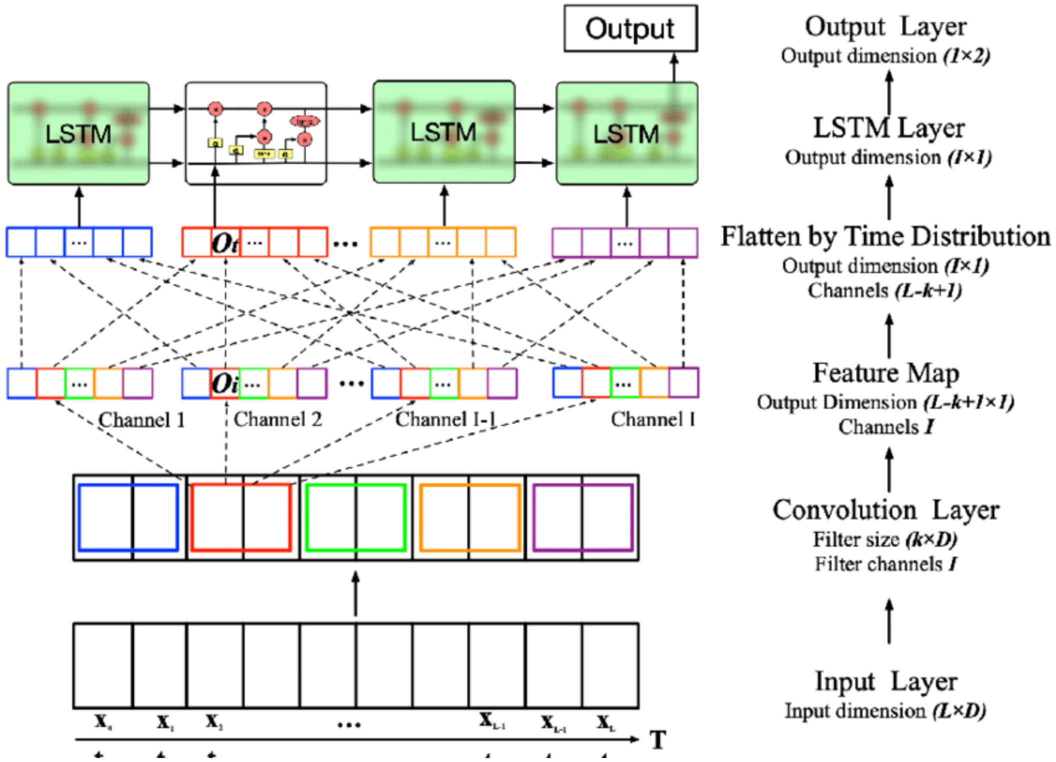

Figure 2: The scheme of a model, which combines ResNet and LSTM.

3.1 Data Preprocessing

3.2 Feature Extraction

3.3 Classification Algorithm

4 Dataset

In the project we use the UCF50[24] - Action Recognition Data Set [24], which turned out to be the most cited among the articles we used. In the dataset the videos from youtube are divided into 50 human action categories. This dataset differs from other available datasets in that it uses stock videos from YouTube, rather than those recorded specifically for the task of analyzing human movements. This introduces its own difficulties in data analysis, since the video uses completely different shooting angles, camera focal lengths, shooting points, backgrounds, scene illumination and many other parameters.

4.1 Analysis

After analyzing the dataset UCF50, we found that it is an imbalance. Each class has a different count of the videos fig. 3. The min count 100 is ‘PlayingViolin‘ and max count 197 ‘HorseRiding‘. The first main goal is to take a balanced dataset. To get it we will take random 100 videos for each class. In the result from 6657 videos in the dataset we have 5000.

The second problem is the number of frames in the video. Due to the fact that each video has a different number of frames, we will have to split them into sub-arrays with images to facilitate model calculations and its unification. It is planned to use a sequence pad from the torch library for this.

4.2 VideoDataset Class Description

The VideoDataset class accepts as input paths to video, transformations, and a balancer - the number of how many videos there should be in each class. When you specify this number, all videos belonging to this class are taken and a random sample of x videos is made. When requesting frames, the name of its class is also given. There are plans to add it to work with the number of frames to unify the number of submitted images at a time for the model.

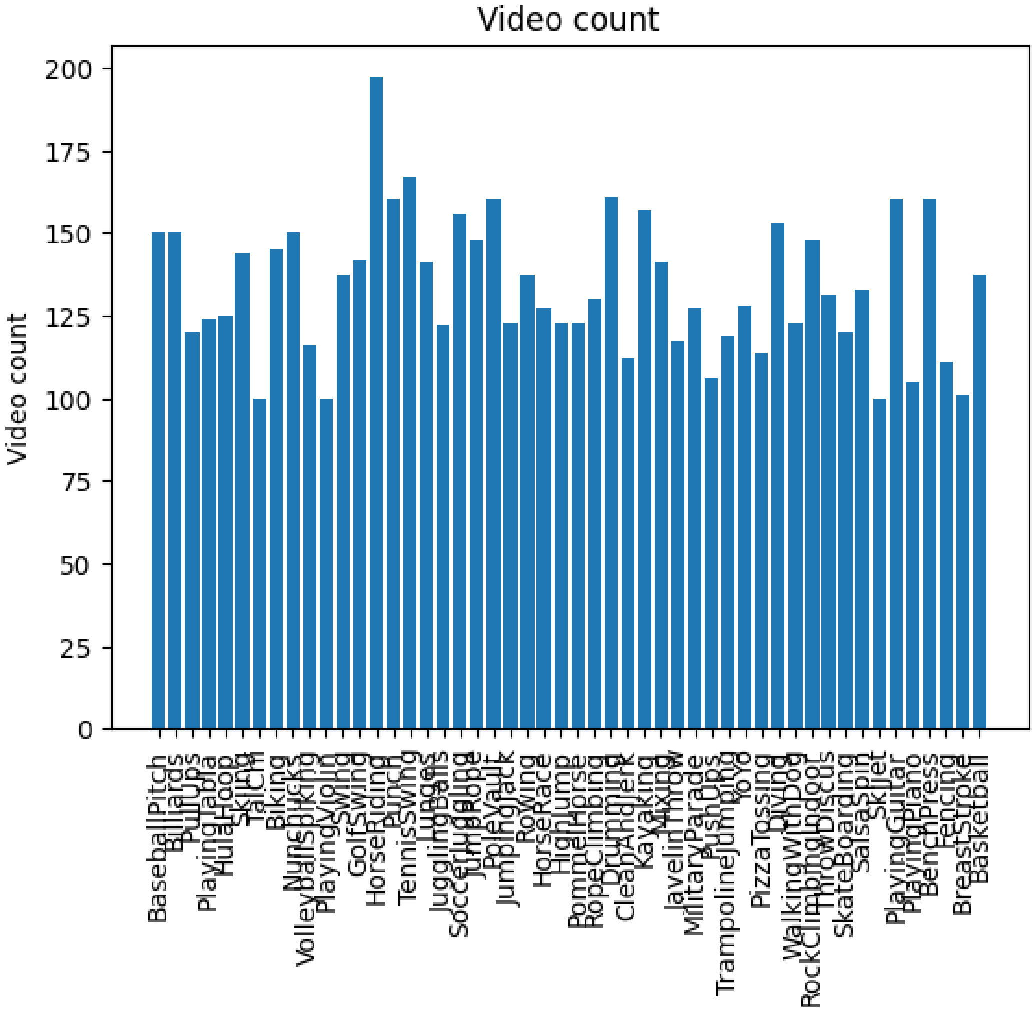

Figure 3: Number of different human actions in the UCF50 Dataset.

5 Discussion

6 Conclusion

7 Future plans for next 3 weeks

1 - get results for combined ResNet and LSTM model. 2 - Analyze results

References

[1] H. H. Pham, L. Khoudour, A. Crouzil, P. Zegers, and S. A. Velastin, “Video-based human action recognition using deep learning: A review,” 2022.

[2] D. Wu, N. Sharma, and M. Blumenstein, “Recent advances in video-based human action recognition using deep learning: A review,” 2017.

[3] R. Darelli, L. Mahathi, M. Koushik, and P. K. D. Kollu, “A deep learning framework for human action recognition on youtube videos,” 2022.

[4] D. Manimala, K. N. Nayak, R. S. Ramesh, S. Mittapalli, and V. V, “Human action recognition in videos,” 2022.

[5] T. Ahmad, J. Wu, I. Khan, A. Rahim, and A. Khan, “Human action recognition in video sequence using logistic regression by features fusion approach based on cnn features,” 2021.

[6] C. I. Orozco, E. Xamena, M. E. Buemi, and J. J. Berlles, “Human action recognition in videos using a robust cnn lstm approach,” 2020.

[7] A. Sarabu and A. K. Santra, “Human action recognition in videos using convolution long short-term memory network with spatio-temporal networks,” 2021.

[8] W. Zhu, X. Ma, Z. Liu, L. Liu, W. Wu, and Y. Wang, “Motionbert: A unified perspective on learning human motion representations,” 2023.

[9] K. Li, Y. Wang, P. Gao, G. Song, Y. Liu, H. Li, and Y. Qiao, “Uniformer: Unified transformer for efficient spatiotemporal representation learning,” 2022.

[10] M. Kulbacki, J. Segen, Z. Chaczko, J. Rozenblit, M. Kulbacki, R. Klempous, and K. Wojciechowski, “Intelligent video analytics for human action recognition: The state of knowledge.” 2023.

[11] G. Surek, L. Seman, S. Stefenon, V. Mariani, and L. Coelho, “Video-based human activity recognition using deep learning approaches,” 2023.

[12] M. U. ASAD, “Activity recognition.” [Online]. Available: https://www.kaggle.com/code/maifeeulasad/activity-recognition

[13] D. POOLLL, “Action recognition.” [Online]. Available: https://www.kaggle.com/code/deadpoolll/action-recognition

[14] HARSHRAJ22, “c3d ucf101.” [Online]. Available: https://www.kaggle.com/code/harshraj22/c3d-ucf101

[15] M. SADIK, “Ucf101 videos - action recognition.” [Online]. Available: https://www.kaggle.com/code/mouadsadik/ucf101-videos-action-recognition

[16] A. GOLE, “Human activity recognization.” [Online]. Available: https://www.kaggle.com/code/aditya9790/human-activity-recognization#Plot-the-model-metrics

[17] D. L. DSOUZA, "Action recognition -cnn + lstm." [Online]. Available: https://www.kaggle.com/code/dianalaveena/action-recognition-cnn-lstm

[18] MUAYAD, "Ucf101." [Online]. Available: https://www.kaggle.com/code/muayad/ucf101

[19] F. TAGHIYEV, "Video action recognition ucf101." [Online]. Available: https://www.kaggle.com/code/faridtaghiyev/video-action-recognition-ucf101

[20] N. M. CNG, "Pytorch: Video classification with conv2d + lstm." [Online]. Available: https://www.kaggle.com/code/nguyenmanhcuongg/pytorch-video-classification-with-conv2d-lstm#Conv-+-LSTM

[21] LONNIE, "Video classification." [Online]. Available: https://www.kaggle.com/code/lonnieqin/video-classification

[22] D. SMIRNOV, "Video plus audio." [Online]. Available: https://www.kaggle.com/code/dannysmirnov/video-plus-audio

[23] A. KUMAR, "Cnn+lstm with attention." [Online]. Available: https://www.kaggle.com/code/ankitkr1606/cnn-lstm-with-attention

[24] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of web videos," 2012.