Video-based Action Recognition Using Deep Learning: A Final Report

Temirlan Nubay
Nazarbayev University
Astana, Kazakhstan
temirlan.nurbay@nu.edu.kz

Rakhat Meiramov
Nazarbayev University
Astana, Kazakhstan
rakhat.meiramov@nu.edu.kz

Abstract

The field of computer vision has seen remarkable growth, with video-based action recognition emerging as a critical subdiscipline. This report delves into the challenges, methodologies, and practical utilities of human action recognition within video streams, seeking to contribute to the ongoing discourse in this rapidly evolving area. We build upon the foundational work of Tran et al. [8], who proposed the R(2+1)D architecture that factorizes 3D convolution into distinct spatial and temporal components, demonstrating its effectiveness in large-scale video datasets and achieving state-of-the-art results. Leveraging this novel architecture, our study applies the R(2+1)D model to the KTH and UCF Sports Action datasets, with the goal of validating the model's performance and supporting the findings of the original study.

1. Introduction

The computer vision is one of fields that have reached new heights as a result of the rapid delelopment of technology. One of the crucial and fast-developing subdisciplines within computer vision is video-based action recognition. Human action recognition in videos is a focal area of the research in computer vision due to its wide range of possible applications. Benefits range from improving security systems to enhancing human-computer interactions, from streamlining video indexing and querying to advancing content-based video analytics. This report explores the difficulties, techniques, and applications associated with recognizing human actions through video sequences. It also proposes a deep learning model aimed at achieving performance on par with the most recent advancements in this field.

Identifying actions in video sequences is not a straightforward task. The visual contents of diverse actions can often appear deceptively similar. At the same time each action is different in their performance complexity. For instance, a football kick is relatively straightforward, involving primarily the leg's motion. In contrast, a jump for a header in soccer is a complex movement involving the legs, arms, head, and the entire body. Such variances demand sophisticated techniques for accurate recognition. The real-world setting introduces additional complications. The conditions in which films are shot are rarely perfect. Cluttered backgrounds, occlusions, and viewpoint variations can obscure the true nature of actions. It's worth noting that many existing methodologies operate under certain assumptions, such as minimal scale or viewpoint changes. However, these assumptions frequently fail in real-world scenarios, underscoring the need for more adaptive approaches [3].

One of the challenges is camera motion, often inherent in real-world video sequences. Such motion can dramatically reduce recognition performance. Addressing this issue involves a common approach of applying video stabilization as a preprocessing step before action recognition. However, achieving flawless video stabilization remains unattainable in numerous practical situations, potentially resulting in the loss of crucial information [2].

Another issue is the large variability in actions. When different subjects are performing the same action, they do not have the same appearance and their movements can be quite different for the same action. Even for a person performing the same action multiple times, each performance can be quite different from the previous one. Therefore, robust classification is an important issue in the human action recognition problem [2]. Nevertheless there have been many successful attempts of action recognition by developing models that have high accuracy and efficiency.

Deep learning models, particularly Convolutional Neural Networks (CNNs), have become the cornerstone of many advanced methods in this domain. The R(2+1)D architecture stands out among them by innovatively splitting 3D convolutions into 2D spatial and 1D temporal convolutional processes within a residual framework. This decomposition allows us to understand both spatial and temporal aspects of video data, facilitating the detection of complex action sequences despite the presence of variability and performance differences. By alternating between spatial and temporal convolutions, the R(2+1)D model avoids the complexity of 3D convolutions, but still captures useful data, achieving state-of-the-art performance on multiple benchmarks [8].

The subsequent sections of the report are structured as follows: The section 2 examines prior research and the challenges it addresses. The 3rd section provides an overview of the datasets in use. The 4th section outlines methodology, by introducing baseline and main models. The 5th one presents evaluation metrics used in this project. The section 6 discusses the performance of the models. The 7th section gives error analysis and section 8 conludes the project and offers suggestions for future work.

2. Literature Review

Many different strategies and architectures have been presented to effectively detect and understand human actions. Particularly, deep learning models have evolved into the core of many cutting-edge approaches in this field. The performance of some models on datasets are shown in Table 1.

Method	KTH	UCF sports action
Ji *et al*. [3]	90.2%	N/A
Ali and Wang [1]	94.3%	N/A
Le *et al*. [4]	93.9%	86.5%
Cho *et al*. [2]	94.2%	89.7%
Weinzaepfel *et al*. [9]	N/A	90.5%

Table 1: Average accuracy of the methods on KTH and UCF Sport action

Ali and Wang introduced a unique variant of deep neural networks known as the Deep Brief Network (DBN) [1]. The structure of DBN includes multiple hidden unit layers inter-connected, assisting in learning features specific to action recognition. By employing the Discrete Cosine Transform (DCT) technique, they optimized feature extraction and data learning. The Support Vector Machine (SVM) classifier was used for classification. To speed up learning time, they incorporated the Fast Fourier Transform (FFT) technique, allowing images to be converted into the frequency domain. Their experiments on the KTH dataset showcased an impressive accuracy rate surpassing 94.3% [1].

Le *et al*. combined different network models into a singular deep architecture to increase its performance [4]. Based on two foundational ideas of convolution and stacking in CNN architecture, they designed a deep model that integrates Independent Subspace Analysis (ISA) and Principal Component Analysis (PCA). ISA's primary function is to learn features from unlabeled video data directly by being trained on smaller input patches. Thereafter, a larger section of the input image undergoes the convolution process with ISA. For dimensionality reduction, PCA is applied, and the resulting responses are used as an input layer for another ISA. The state-of-the-art in human activity recognition was advanced by this architecture [4].

Ji *et al*. proposed three-dimensional convolutional neural network (3D-CNN) for recognizing human actions [3]. The architecture uses 3D kernels during the convolutional stages. Such a design allows it to capture motion features across both spatial and temporal dimensions. By applying it to consecutive frames in a video, it can seamlessly extract multiple features. Their model acts directly on raw inputs, bypassing the need for handcrafted features that many traditional methods relied upon.The application of their models in real-world airport surveillance videos demonstrated a great performance compared to their baseline methods [3].

Cho *et al*. presented a method for unsupervised identification and classification of actions, emphasizing the use of local motions [2]. It combines local and full motion descriptors, applies group sparsity to select important features, and uses the multiple kernel method to improve action recognition. Experimental results demonstrate that this approach outperforms existing methods and is highly efficient in terms of feature selection [2].

Weinzaepfel *et al*. introduced an effective spatio-temporal action localization technique in videos [9]. The approach begins by detection of potential action regions in video frames and scoring them using static and motion features from CNNs. High-scoring regions are tracked throughout the video, incorporating both specific instance and general class detectors. The tracks are further evaluated using a spatio-temporal motion histogram and CNN features. Temporal boundaries of actions are determined through a sliding-window approach [9].

Finally, a paper by Tran *et al*. presented an analysis of spatiotemporal convolutions in the context of action recognition [8]. The study primarily focuses on contrasting the effectiveness of 2D and 3D Convolutional Neural Networks (CNNs) in video-based action recognition tasks. Authors introduced an R(2+1)D architecture that factorizes 3D convolution into separate spatial and temporal components. The experiments, which were conducted on huge datasets like Sports-1M, Kinetics, UCF101, and HMDB51, showed that "(2+1)D" Residual neural network (ResNets) achieve state-of-the-art performance. Our study builds upon this by applying the R(2+1)D model to the KTH [6] and UCF Sports Action [5], [7] datasets, aiming to support these findings.

3. Datasets

3.1. Original datasets

- KTH [6]:
    1. Size: 600 clips
    2. Class Distribution:
        (a) Walking: 100 clips
        (b) Running: 100 clips
        (c) Jogging: 100 clips
        (d) Hand-waving: 100 clips
        (e) Hand-clapping: 100 clips
        (f) Boxing: 100 clips
- UCF Sports Action [5], [7]:
    1. Size: 153 clips
    2. Class Distribution:
        (a) Diving-side: 14 clips
        (b) Golf-Swing-Back: 5 clips
        (c) Golf-Swing-Front: 8 clips
        (d) Golf-Swing-Side: 8 clips
        (e) Kicking-Front: 10 clips
        (f) Kicking-Side: 10 clips
        (g) Lifting: 6 clips
        (h) Riding-Horse: 12 clips
        (i) Run-Side: 13 clips
        (j) SkateBoarding-Front: 12 clips
        (k) Swing-Bench: 20 clips
        (l) Swing-SideAngle: 13 clips
        (m) Walk-Front: 22 clips

3.2. Preprocessing

Two datasets were merged into a single dataset with following steps:

- Overlapping Classes Merging:
    1. 'run-side' (from UCF) + 'running' (from KTH) = 'running' (in the merged dataset).
    2. 'walk-front' (from UCF) + 'walking' (from KTH) = 'walking' (in the merged dataset).
- Structural Reorganization:
    1. The structure of the classes from UCF was converted to match the structure of KTH for uniformity.
    2. 7 clips from the 'Diving-Side' category in UCF were excluded because they consisted solely of frames, lacking actual video content, to maintain consistency with the KTH dataset during feature extraction.
    3. Classes named Golf-Swing-Front, Golf-Swing-Side, and Golf-Swing-Back will be consolidated into one unified Golf-Swing class since they depict the same action from varied perspectives.
    4. Kicking-Front and Kicking-Side classes were unified for the same reason.

We additionally executed supplementary video preprocessing as part of the main approach:

1. **Data Type Conversion**: Convert video to floating-point data type.
2. **Frame Processing**:
    - Limit processing to `min(self.frame_count, number of frames)`.
    - Convert grayscale frames to RGB by repeating the channel.
    - Permute frame dimensions to `[channels, height, width]`.
    - Resize frames to `self.resize` using bilinear interpolation.
3. **Padding**: Pad frames list to `self.frame_count` with the last frame if needed.
4. **Stacking and Permuting**: Stack frames and permute to `[channels, frames, height, width]`.
5. **Output**: Return the processed video tensor.

4. Methodology

4.1. Baseline Model

This section provides a comprehensive overview of the baseline action recognition framework that has been implemented. It breaks down the individual stages from feature extraction to final classification, explaining the rationale and procedures behind each step. As depicted in Figure 1, the process begins with the extraction of SIFT descriptors, which capture essential local image features invariant to scale and rotation. These descriptors are then encoded into VLAD vectors to form a compact yet descriptive representation of the images. Finally, an SVM classifier is trained on these vector

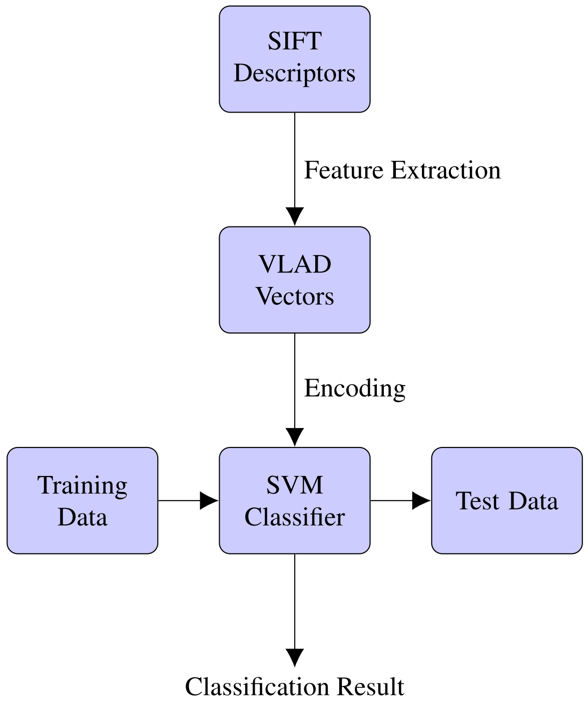


Figure 1: Baseline Classification Pipeline

representations to differentiate between various actions. This approach is a classical one in the field of computer vision and image classification, especially before the widespread adoption of deep learning methods.

4.1.1 Feature Extraction using SIFT

The SIFT descriptors are resistant to changes in image rotation and scaling, making them pertinent for video action identification.

For every video, we divided it into frames and utilized OpenCV's SIFT method to get the keypoints and descriptors for each frame. Subsequently, we compiled the descriptors into an array for every video. The resulting array is an input for the next step below.

4.1.2 Encoding image descriptors with VLAD

To create a baseline action recognition framework, the need to efficiently transform variable-length descriptors into a consistent, fixed-size representation for classification was needed. We had two choices: BoVW (Bag of Visual Words) and VLAD (Vector of Locally Aggregated Descriptors).

While both methodologies hinge on the principles of K-means clustering to create a codebook of representative visual features, there's a marked computational difference that steered our decision-making process.

BoVW, by its design, requires a substantially larger computational power and time for optimal representation. A cluster size hovering around 400 is often touted as ideal for capturing the nuances of the visual space. This considerable size, while comprehensive, increases the computational overhead, especially when paired with the requisite quantization step that maps each descriptor to its nearest visual word.

Contrastingly, VLAD offers a more compact representation without significantly compromising on the quality of the information encapsulated. Remarkably, VLAD achieves commendable performance with as few as 64 clusters. Instead of simple quantization, VLAD encodes the difference between the descriptors and their associated cluster centroids, which inherently carries more information. This differential encoding, coupled with a smaller codebook, makes VLAD a more computationally efficient alternative.

Based on our computational resources, our inclination towards VLAD emerged as a judicious choice. Its ability to capture intricate visual details with a reduced cluster size made it an ideal fit, ensuring both efficiency and representational accuracy.

4.1.3 Classification using SVM

In the development of our baseline action recognition framework, the Support Vector Machines (SVM) algorithm emerged as the chosen classifier.

Using the `scikit-learn` library we used a classifier version of SVM encapsulated within its `svm` module. Specifically, our SVM classifier is anchored on the `SVC` class.

```
from sklearn.svm import SVC
classifier = SVC(kernel='linear')
```

Parameters are pivotal in shaping the classifier's behavior. We used a linear kernel given its efficacy for high-dimensional datasets. The regularization parameter, 'C', was set to a default value of 1, balancing the trade-off between minimal training error and margin maximization.

Training the classifier entails the `fit` method, consuming both the training data and the corresponding labels.

```
classifier.fit(train_data, train_labels)
```

For prediction on novel data, the `predict` method was invoked.

```
predictions
    = classifier.predict(test_data)
```

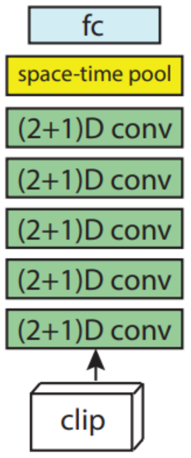
Integrating SVM with the efficiency of `scikit-learn` culminated in a robust classification mechanism. The entire pipeline, spanning from feature extraction to classifications, was targeted for optimal baseline performance, courtesy of the detailed feature vectors from VLAD encoding and SVM's inherent strengths.

4.2. Main approach

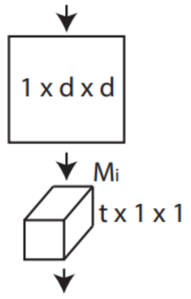
Our study primarily focused on usig a Residual Network R(2+1)D model, which is pretrained on the Kinetics dataset, further training it on a merged dataset. The R(2+1)D model is an innovative form of CNN that offers a unique approach in processing spatiotemporal data by decomposing 3D convolutions into spatial and temporal parts.

4.2.1 R(2+1)D Architecture

The R(2+1)D architecture modifies the traditional 3D convolutional filters, typically sized as $N_{i-1} \times t \times d \times d$, by replacing them with a "(2+1)D" convolutional block. This block consists of M_i 2D convolutional filters of size $N_{i-1} \times 1 \times d \times d$, followed by Ni temporal convolutional filters of size $M_i \times t \times 1 \times 1$. This design effectively separates the spatial and temporal aspects of convolution, allowing the model to more accurately capture the dynamics of motion in video data. The hyperparameter M_i, calculated as $M_i = \left\lfloor \frac{td^2 N_{i-1}}{d^2 N_{i-1} + tN_i} \right\rfloor$, is critical in determining the dimensionality of the intermediate subspace where the signal is projected between spatial and temporal convolutions. It ensures that the total number of parameters in the "(2+1)D" block approximates that of a full 3D convolution. The overall architecture is presented in Fig 2a. It shows the data flow from the input clip through multiple (2+1)D convolutional layers, leading up to a space-time pooling layer, and culminating in a fully connected (fc) layer. The figure 2b illustrates the decomposition process within a (2+1)D convolutional block. It starts with the input represented by a 3D tensor of dimensions 1 x d x d, which is indicative of a single channel spatial feature map. This tensor is then convolved with Mi number of temporal convolutional filters of size t x 1 x 1, capturing the temporal dynamics over the spatial feature map.



(a) The overall architecture of the R(2+1)D model, illustrating the flow of input.



(b) Detailed view of the (2+1)D convolutional block

Figure 2: The R(2+1)D model. Adopted from [8].

5. Evaluation Metrics

For model assessment, we adopted the following metrics and methodologies:

- **Train/Test split**: In our framework, we split the dataset into training and testing sets in an 80:20 ratio using a stratified split approach. This technique ensures that the proportion of each class in the dataset is maintained in both the training and testing sets, providing a representative sample for model training and evaluation. The rationale behind using a stratified split is to preserve the class distribution, which is particularly important for handling imbalanced datasets where certain classes are underrepresented.
- **Accuracy**: We employed top-1 (acc@1), top-3 (acc@3), and top-5 (acc@5) accuracy metrics, providing a nuanced understanding of its predictive capabilities across different levels of classification precision.
- **Confusion Matrix**: A table that illustrates the true versus predicted classifications of the model.
- **Per-class Accuracy**: Offers insights into the model's performance for individual action classes.

Employing these metrics and methodologies, we can gain a comprehensive understanding of the classifier's performance nuances.

6. Results and Analysis

This section outlines the hyperparameters employed in training the R(2+1)D model:

- **Train Ratio**: 80
- **Frame Count**: 16 video frames are input to the model.

- **Resize Dimensions**: Video frames are resized to 224×224.
- **Batch Size**: Varied batch sizes of 1, 2, 4, 8, 16 are used.
- **Number of Epochs**: Training duration ranges from 10 to 70 epochs.
- **Learning Rate**: Set at 0.001.
- **Model**: The pre-trained R(2+1)D model on the KINETICS400 dataset is utilized.
- **Loss Function**: Cross Entropy Loss.
- **Optimizer**: Adam.

During the optimization of the R(2+1)D model, our initial strategy involved exploring various hyperparameters. Time constraints, however, led us to focus primarily on different batch sizes due to their significant impact on training dynamics and model performance.

Batch sizes of 1, 2, 4, 8, and 16 were extensively tested. Models with batch sizes of 2 and 16 showed optimal results. Models with batch sizes of 1, 4, and 8 plateaued at around 30% accuracy. In contrast, the model with a batch size of 2 steadily improved, peaking at 96% training accuracy by the 55th epoch. The model with a batch size of 16 achieved peak accuracy faster, by the 25th epoch, suggesting that larger batch sizes provide more stable gradient estimates and hence more efficient training.

These experiments emphasize the importance of batch size in neural network training. While our focus was on batch sizes, the time required for these experiments restricted our ability to explore other hyperparameters. Nonetheless, the insights from these experiments are invaluable, showing that both smaller and larger batch sizes can be optimally used under different conditions for our specific model and dataset.

6.1. Comparative Analysis of Model Performances

The comparative evaluation of models, as detailed in Table 2, indicates that Models 2 and 3 (R(2+1)D with batch sizes 2 and 16, respectively) surpassed the baseline Model 1 (SVC). Key observations include:

- Model 2 improved overall Top-1 accuracy by 21.48% over the baseline, while Model 3 enhanced it by 24.83%.
- Model 3 achieved perfect Top-3 and Top-5 accuracies, demonstrating superior predictive capability.
- The larger batch size in Model 3 led to more efficient learning, with higher accuracy in fewer epochs.
- Both Models 2 and 3 showed significant accuracy improvements across classes, validating the effectiveness of the R(2+1)D architecture and batch size optimization.

These findings confirm the advancements in Models 2 and 3 in comparison with baseline model. The confusion matrices for Models 2 and 3 (Figure 3 and Figure 4, respectively) provide further insights into their classification capabilities:

- Model 2 excels in classes like 'boxing', 'handclapping', and 'handwaving' but lacks accuracy in 'Diving-Side' and 'Lifting'.
- Model 3 nearly perfectly classifies most classes, particularly 'Diving-Side' and 'Lifting', but confuses similar classes such as 'jogging' and 'running'.
- Both models struggle with visually similar classes, indicating a need for improved discriminative learning.
- The larger batch size in Model 3 correlates with its higher accuracy, suggesting better generalization.

These observations highlight Model 3's superior performance while also pointing out areas for improvement in differentiating visually similar activities. Future model iterations could benefit from enhanced feature extraction and data augmentation strategies.

7. Error analysis

7.1. Experiments and Observations

A focused evaluation was conducted using Model 2 (R(2+1)D with batch size 2) first and then Model 3 (R(2+1)D with batch size 16), which had previously shown promising training accuracy. Our testing involved self-recorded videos. Our dataset was limited to six classes due to practical constraints: boxing, handclapping, handwaving, jogging, running, and walking, with four instances each (See Fig. 7).

7.2. Analysis

The confusion matrices from Model 2 (see Fig. 5) and Model 3 (see Fig. 6) reveal some insights into the models' prediction behaviors:

As Table 3 illustrates, Model 3 demonstrated exceptional performance for the classes of boxing, handclapping, and handwaving, achieving 100% accuracy

Class	Model 1 (SVC)			Model 2 (r(2+1)d batch=2)			Model 3 (r(2+1)d batch=16)		
	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
Overall	59.73%	93.29%	99.33%	81.21%	93.96%	97.99%	84.56%	100.00%	100.00%
Boxing	80.00%	95.00%	100.00%	100.00%	100.00%	100.00%	95.00%	100.00%	100.00%
Diving-Side	100.00%	100.00%	100.00%	0.00%	0.00%	0.00%	100.00%	100.00%	100.00%
Golf-Swing	25.00%	75.00%	100.00%	75.00%	100.00%	100.00%	75.00%	100.00%	100.00%
Handclapping	65.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Handwaving	55.00%	95.00%	100.00%	90.00%	100.00%	100.00%	95.00%	100.00%	100.00%
Jogging	35.00%	95.00%	100.00%	90.00%	100.00%	100.00%	40.00%	100.00%	100.00%
Kicking	25.00%	75.00%	75.00%	25.00%	25.00%	25.00%	75.00%	100.00%	100.00%
Lifting	100.00%	100.00%	100.00%	0.00%	0.00%	0.00%	100.00%	100.00%	100.00%
Riding-Horse	50.00%	50.00%	100.00%	50.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Running	52.17%	91.30%	100.00%	65.22%	78.26%	100.00%	86.96%	100.00%	100.00%
SkateBoarding-Front	50.00%	50.00%	100.00%	100.00%	100.00%	100.00%	50.00%	100.00%	100.00%
Swing-Bench	100.00%	100.00%	100.00%	25.00%	75.00%	100.00%	100.00%	100.00%	100.00%
Swing-SideAngle	100.00%	100.00%	100.00%	66.67%	100.00%	100.00%	100.00%	100.00%	100.00%
Walking	68.00%	96.00%	100.00%	76.00%	84.00%	96.00%	88.00%	100.00%	100.00%

Table 2: Accuracies of models across different classes

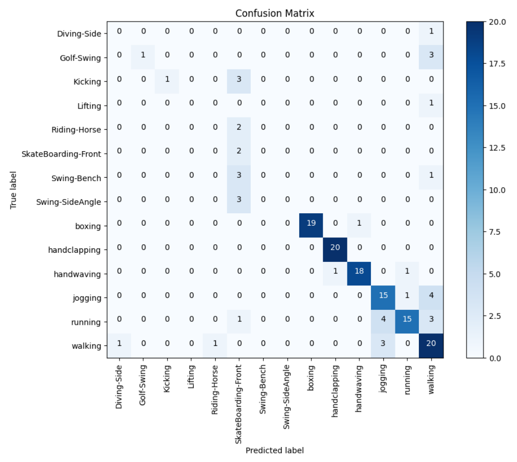


Figure 3: Confusion matrix for Model 2 (r(2+1)d with batch size 2).

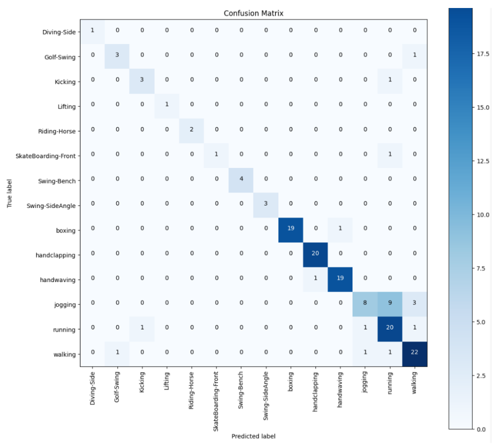


Figure 4: Confusion matrix for Model 3 (r(2+1)d with batch size 16).

across the Top-1, Top-3, and Top-5 metrics. However, it showed lower accuracy in the jogging and walking classes, with Top-1 accuracies of 25% and 50%, respectively. This disparity could be explained by the low variations in these actions, particularly when carried out by various people with different performance styles.

Model 2 also performed well in the classes of boxing and handclapping with 100% accuracies. It faced challenges in the running and walking classes, where it achieved only 50% Top-1 accuracy, indicating potential confusion between these actions. The primary takeaway from these experiments is the realization that the complexity of human actions cannot be encapsulated solely by model architecture or batch size during training. Factors such as intra-class variation, filming conditions, and the subjective nature of human movement play significant roles in a model's ability to generalize from training to real-world application. Based on the confusion matrices and the accuracy data, Model 3 (r(2+1)d with batch size 16) generally shows a stronger performance compared to Model 2 (r(2+1)d with batch size 2).

7.3. Potential Errors

Potential errors in the method may have arisen from the limited datase. While our models have demonstrated a significant capability to recognize and classify a subset of human actions accurately, the disparity in performance across similar classes highlights the need for larger, more varied datasets and potentially more complex model tuning to achieve a universally robust action recognition system.

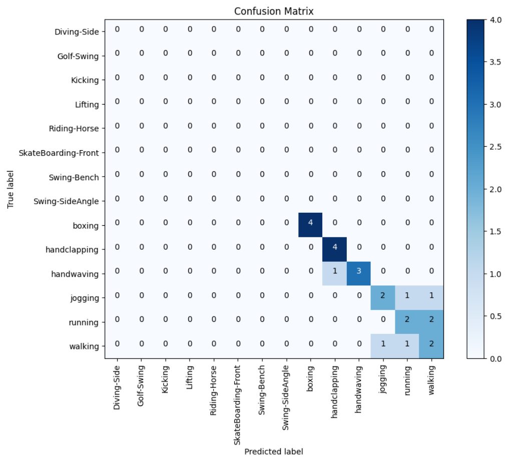


Figure 5: Experiment: Confusion matrix for Model 2 (r(2+1)d with batch size 2).

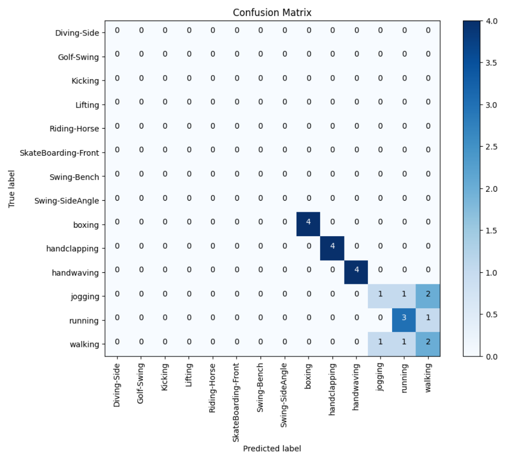


Figure 6: Experiment: Confusion matrix for Model 3 (r(2+1)d with batch size 16).

	Model 2 (r(2+1)d batch=2)			Model 3 (r(2+1)d batch=16)		
Class	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
Boxing	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Handclapping	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Handwaving	75.00%	75.00%	100.00%	100.00%	100.00%	100.00%
Jogging	50.00%	75.00%	75.00%	25.00%	100.00%	100.00%
Running	50.00%	50.00%	50.00%	75.00%	100.00%	100.00%
Walking	50.00%	75.00%	75.00%	50.00%	100.00%	100.00%

Table 3: Accuracies of Model 2 and Model 3 across selected classes

(a) Clapping. (b) Boxing.

Figure 7: Self-recorded videos.

8. Conclusion and Future Work

This project studied the r(2+1)d architecture. It was trained with different setups. Model 3, with a larger batch size, generally outperformed Model 2, displaying higher accuracy and efficiency in learning. Our experiments, though limited to six classes due to practical constraints, provided valuable insights into the models' performance, highlighting the influence of batch size on the training process and the models' ability to distinguish between various actions. Despite the limitations imposed by the small and self-generated dataset, the models demonstrated a robust capability to generalize from training to testing conditions.

For future work, the experimental dataset may include a more diverse range of actions and scenarios. This will enable us to further test the models' generalization capabilities and improve their robustness. Additionally, following areas can be explored:

- Incorporating more sophisticated data augmentation techniques to enhance the models' ability to learn from limited datasets.
- Experimenting with different architectures and hybrid models to improve the distinction between similar actions.
- Investigating the impact of other hyperparameters, such as learning rate and weight decay, on model performance.

The code of the project is available on github [1]

[1]https://github.com/RakhatMM/CV-HumanActionRecognition

References

[1] K. H. Ali and T. Wang. Learning features for action recognition and identity with deep belief networks. In *2014 International Conference on Audio, Language and Image Processing*, pages 129–132. IEEE, 2014.

[2] J. Cho, M. Lee, H. J. Chang, and S. Oh. Robust action recognition using local motion and group sparsity. *Pattern Recognition*, 47(5):1813–1825, 2014.

[3] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[4] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR 2011*, pages 3361–3368, 2011.

[5] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[6] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36. IEEE, 2004.

[7] K. Soomro and A. R. Zamir. Action recognition in realistic sports videos. In *Computer vision in sports*, pages 181–208. Springer, 2015.

[8] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[9] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3164–3172, 2015.