Enhancing Action Recognition with Advanced Frame Extraction Techniques

Rufina Gafiiatullina
r.gafiiatullina@innopolis.com

Ivan Golov
i.golov@innopolis.com

Anatoly Soldatov
a.soldatov@innopolis.com

Innopolis University
Computer Vision Course 2024
Professor Karam Almaghout

Introduction

Video action recognition is a fundamental classification problem in computer vision field with diverse applications, including surveillance, healthcare, sports analytics, and etc. The ability to automatically identify actions within video sequences has transformative potential for real-time decision-making systems, patient monitoring, and performance evaluation. However, the task remains challenging due to the high computational demands of video data and the reliance on extensive, annotated datasets. These challenges are further exacerbated when resources, such as hardware and time, are limited, creating a bottleneck for the broader adoption of such technologies.

This project aims to address these limitations by developing a streamlined approach to video classification that optimizes data processing while preserving essential features for accurate recognition. By reducing the size and complexity of input data, the proposed method facilitates efficient training and classification without sacrificing model performance.

To demonstrate the viability of this approach, we utilize the UCF50 dataset, a widely used benchmark for action recognition. This dataset offers a balance between diversity and manageability, making it suitable for testing resource-efficient solutions [1]. Our work introduces an advanced frame extraction pipeline that significantly compresses video data while retaining critical temporal and spatial information necessary for classification. Unlike conventional methods that often rely on computationally intensive deep learning-based frame selection techniques, we propose a hybrid approach. This combines traditional feature-based methods such as Oriented FAST and Rotated BRIEF (ORB) [2] and Scale-Invariant Feature Transform (SIFT) [3] with state-of-the-art deep learning models, achieving a balance between computational efficiency and effectiveness.

The proposed pipeline incorporates two advanced models: VideoMAE and (2+1)D Convolutions. VideoMAE, a pre-trained transformer model, accelerates training and achieves high performance on action recognition tasks [4]. Meanwhile, modifications to

(2+1)D Convolutions enhance its ability to integrate spatial features with sequential frame data, ensuring a comprehensive understanding of video content [5].

By using this hybrid pipeline, we achieve a nearly 50% reduction in dataset size while maintaining competitive performance metrics, demonstrating the approach's effectiveness in resource-constrained scenarios.

This project makes a significant contribution to video classification scope by introducing a resource-efficient framework that integrates traditional feature extraction techniques with modern deep learning architectures. Our results highlight the potential of hybrid methodologies to address computational challenges in video processing, paving the way for more scalable and accessible solutions in real-world applications.

Related Work

The task of recognizing human actions in videos has gathered significant attention due to its potential applications in different fields. Over the years, numerous methods have been proposed to tackle the inherent spatial and temporal complexities of video data. These approaches can be broadly categorized into traditional feature-based methods, advanced frame selection techniques, and modern deep learning paradigms.

Early Approaches

Prior to the rise of deep learning, hand-crafted feature extraction techniques were widely used for action recognition. Methods like Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB) were highly effective for identifying key points and matching features across frames [2], [6]. While these techniques demonstrated robustness in static image analysis, their application to video data—particularly in action recognition—remains underexplored. Their simplicity and low computational requirements suggest potential utility when combined with modern methodologies to address the temporal aspects of videos.

Advanced Frame Selection

Efficient frame selection has become a critical preprocessing step for action recognition, particularly in scenarios involving short, trimmed videos. Approaches like SMART Frame Selection aim to balance computational cost and accuracy by using both visual and temporal features. This method integrates lightweight models, such as MobileNet for visual analysis and GloVe embeddings for textual context, using a combination of local (MLP-based) and global (attention-based) selectors to evaluate frame relevance [7]. Similarly, Key Frame Extraction (KFE) techniques, such as X^2-based Shot Boundary Detection and Histogram Difference, have been effective in identifying significant frames for downstream tasks [6]. These methods, while computationally efficient, ensure redundancy reduction and highlight the most relevant frames, enabling better action recognition performance.

The Shift to Deep Learning

The advent of deep learning revolutionized video-based tasks, allowing for automatic extraction of both spatial and temporal features. Two dominant paradigms emerged:

(1) the combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), and (2) fully spatiotemporal models like 3D CNNs [8].

In CNN-RNN architectures, CNNs extract spatial features from video frames, which are then processed by RNNs (e.g., LSTMs) to model temporal dependencies. This modular approach separates spatial and temporal feature processing, offering flexibility but requiring sequential training of different components. In contrast, 3D CNNs extend 2D convolutions into three dimensions, allowing simultaneous capture of spatial and temporal information. While 3D CNNs streamline feature extraction, their computational demands are significantly higher, making them less feasible for resource-constrained environments.

Transformer and Factorized Convolution Approaches

Recent advances in video action recognition include the use of transformer-based models and factorized convolutions. Transformer models, such as VideoMAE, use self-supervised pre-training and a novel video tube masking strategy to capture spatiotemporal relationships effectively [4]. This method has demonstrated strong performance even on smaller datasets like UCF101 and HMDB51, but its reliance on extensive computational resources for training limits its scalability.

Factorized spatiotemporal convolutions, exemplified by architectures like R(2+1)D, offer a computationally efficient alternative [5]. By decomposing 3D convolutions into separate spatial (2D) and temporal (1D) operations, R(2+1)D achieves strong performance on benchmarks such as Sports-1M and Kinetics, while reducing training complexity. This balance between computational efficiency and temporal modeling makes it a viable choice for modern video analysis pipelines.

Methodology

The field of video action recognition has evolved from traditional hand-crafted feature extraction methods to advanced deep learning architectures, each with its own trade-offs in terms of efficiency, accuracy, and computational requirements. While modern techniques like VideoMAE and R(2+1)D demonstrate state-of-the-art performance, simpler methods like SIFT and ORB remain underutilized for video tasks despite their low computational overhead.

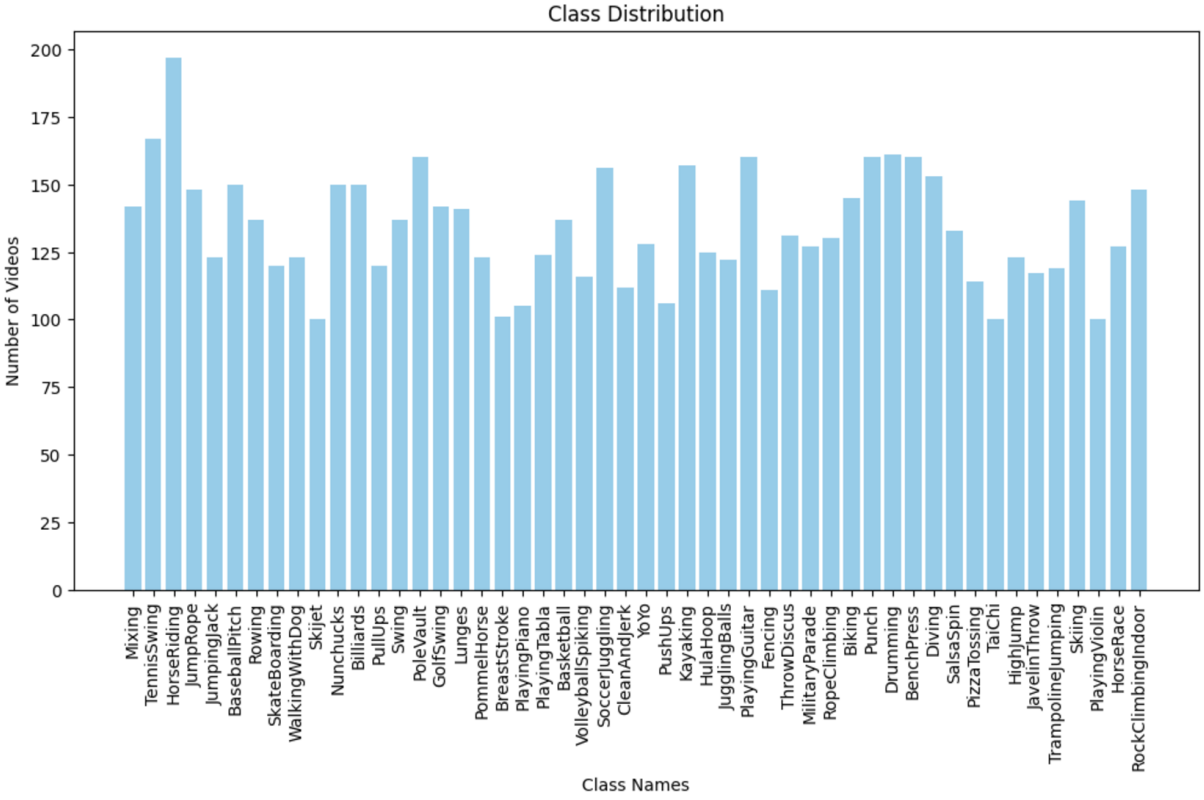
In our research, we propose leveraging these lightweight feature-based methods to develop a custom dataset transformation pipeline. This pipeline will serve as a foundation for testing advanced models like VideoMAE and (2+1)D Convolutions, allowing us to evaluate the potential of combining traditional and contemporary approaches for effective and efficient action recognition.

1. Dataset Description

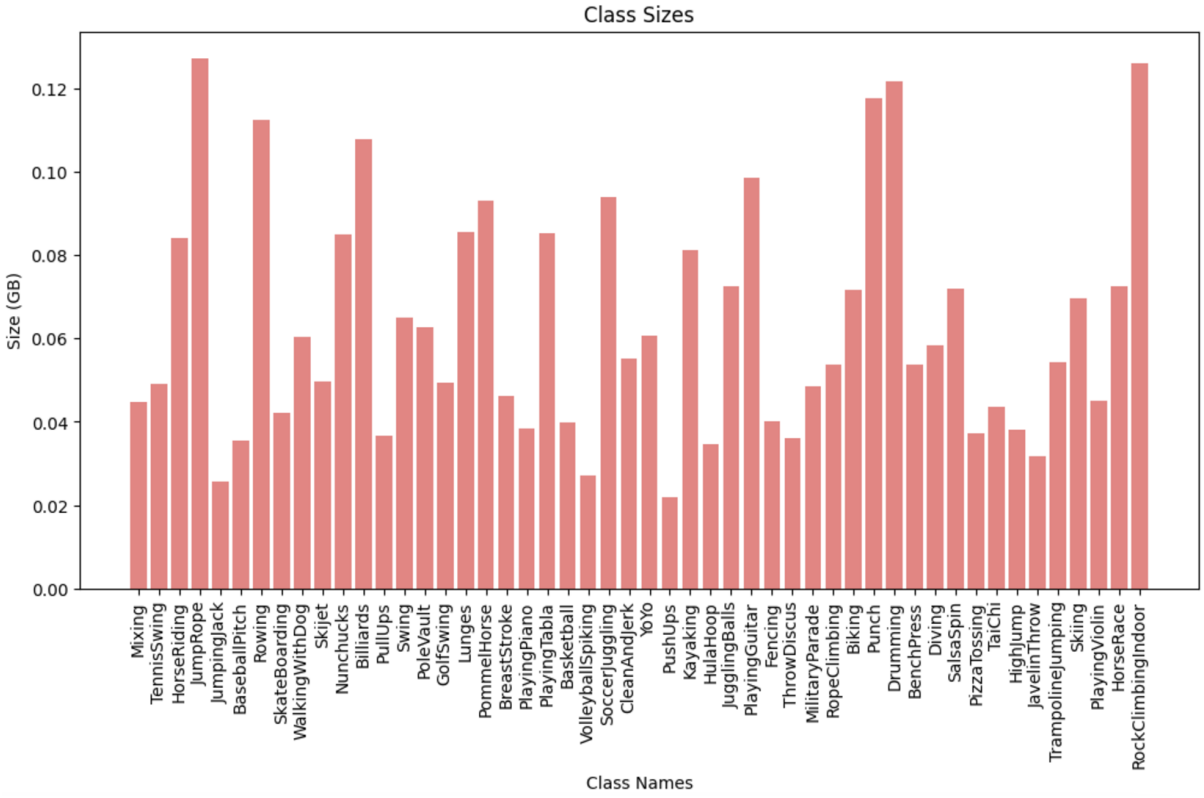
We used **UCF50** dataset that represents video action recognition benchmark consisting of 6,618 video clips that cover 50 different human action categories. It was introduced by the University of Central Florida in 2012 to facilitate research in the area of video understanding and human action recognition.

We analyzed our video classification dataset and created several plots.

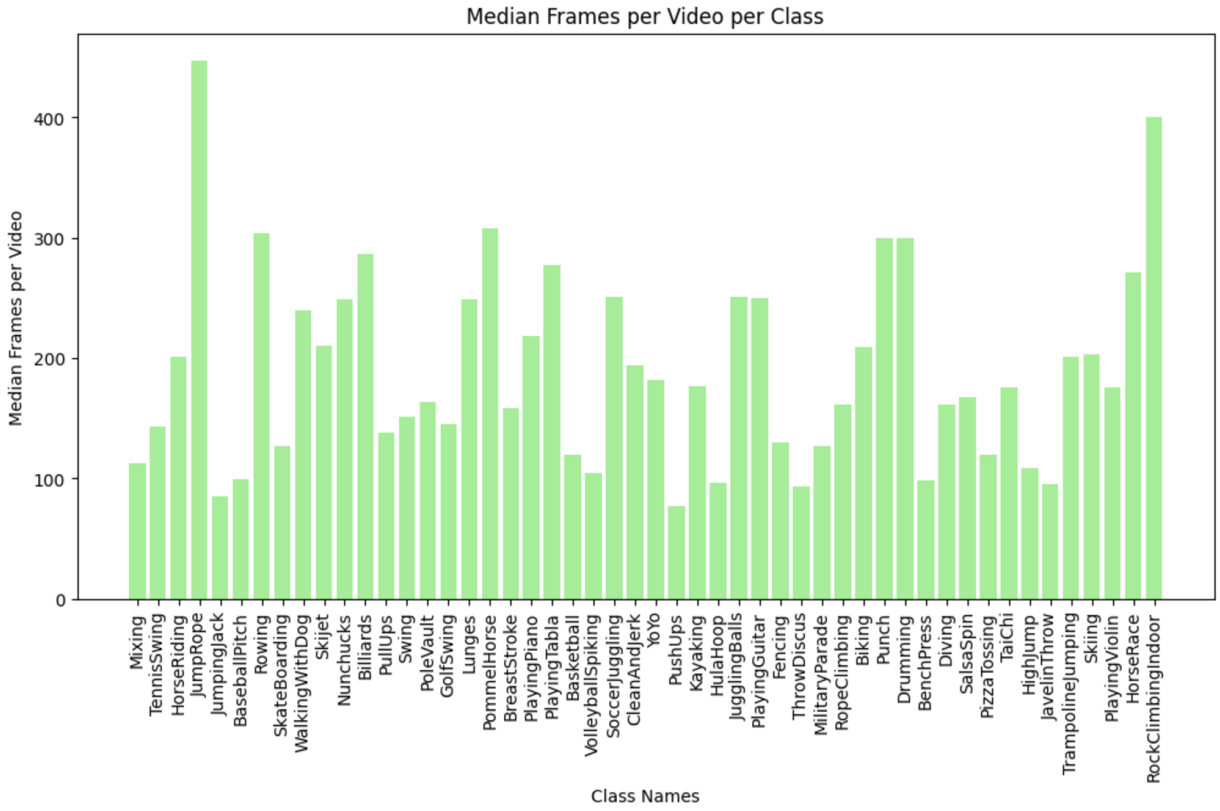
Class Distribution: A plot showing the distribution of samples across different classes to check for balance.

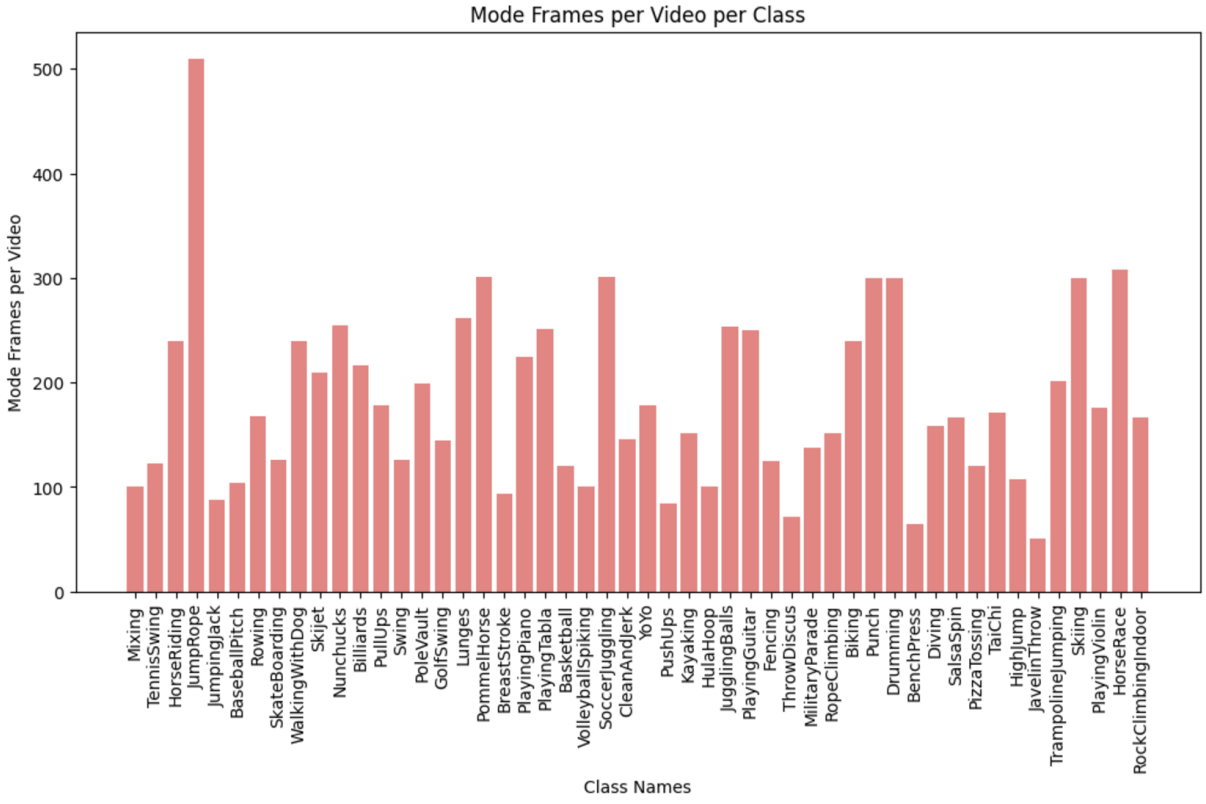


Class Sizes: A plot showing the size in GB of samples in each class, giving an idea of its storage requirements.



Median and Mode of Frames: Calculated and visualized to see the typical and most common video lengths.


Median Frames per Video per Class
Median Frames per Video
0
100
200
300
400
Mixing
TennisSwing
HorseRiding
JumpRope
JumpingJack
BaseballPitch
Rowing
SkateBoarding
WalkingWithDog
Skijet
Nunchucks
Billiards
PullUps
Swing
PoleVault
GolfSwing
Lunges
PommelHorse
BreastStroke
PlayingPiano
PlayingTabla
Basketball
VolleyballSpiking
SoccerJuggling
CleanAndJerk
YoYo
PushUps
Kayaking
HulaHoop
JugglingBalls
PlayingGuitar
Fencing
ThrowDiscus
MilitaryParade
RopeClimbing
Biking
Punch
Drumming
BenchPress
Diving
SalsaSpin
PizzaTossing
TaiChi
HighJump
JavelinThrow
TrampolineJumping
Skiing
PlayingViolin
HorseRace
RockClimbingIndoor
Class Names


Mode Frames per Video per Class
Mode Frames per Video
0
100
200
300
400
500
Mixing
TennisSwing
HorseRiding
JumpRope
JumpingJack
BaseballPitch
Rowing
SkateBoarding
WalkingWithDog
Skijet
Nunchucks
Billiards
PullUps
Swing
PoleVault
GolfSwing
Lunges
PommelHorse
BreastStroke
PlayingPiano
PlayingTabla
Basketball
VolleyballSpiking
SoccerJuggling
CleanAndJerk
YoYo
PushUps
Kayaking
HulaHoop
JugglingBalls
PlayingGuitar
Fencing
ThrowDiscus
MilitaryParade
RopeClimbing
Biking
Punch
Drumming
BenchPress
Diving
SalsaSpin
PizzaTossing
TaiChi
HighJump
JavelinThrow
TrampolineJumping
Skiing
PlayingViolin
HorseRace
RockClimbingIndoor
Class Names

2. Preprocessing techniques

(2+1)D Convolutions

We used a preprocessing pipeline implemented with PyTorch's torchvision.transforms module. This preprocessing step standardizes input sizes across the dataset. The transformation was defined as:

```
transform = T.Compose([
    T.Resize(output_size)
])
output_size=(224, 224)
```

Figure 1: Transformation pipeline for (2+1)D Convolutions Model

VideoMAE

During fine-tuning normalization, random short side scaling, random cropping and random horizontal flipping were used:

```
train_transform = Compose(
    [
        ApplyTransformToKey(
            key="video",
            transform=Compose(
                [
                    UniformTemporalSubsample(num_frames_to_sample),
                    Lambda(lambda x: x / 255.0),
                    Normalize(mean, std),
                    RandomShortSideScale(min_size=256, max_size=320),
                    RandomCrop(resize_to),
                    RandomHorizontalFlip(p=0.5),
                ]
            ),
        ),
    ]
)
```

Figure 2: Transformation pipeline for VideoMAE Model

3. Models

(2+1)D Convolutions

This paper [5] investigates the use of **(2+1)D convolutions** with residual connections for efficient spatiotemporal processing. We implemented a (2+1)D convolution that decomposes 3D convolutions into separate spatial and temporal steps, reducing the number of parameters while maintaining efficiency.

Key Points

- **3D Convolution:** Combines vectors from a 3D patch of inputs ($time \times height \times width \times channels$) to produce outputs. For a kernel size of $3 \times 3 \times 3$, this requires $27 \times \text{channels}^2$ parameters.

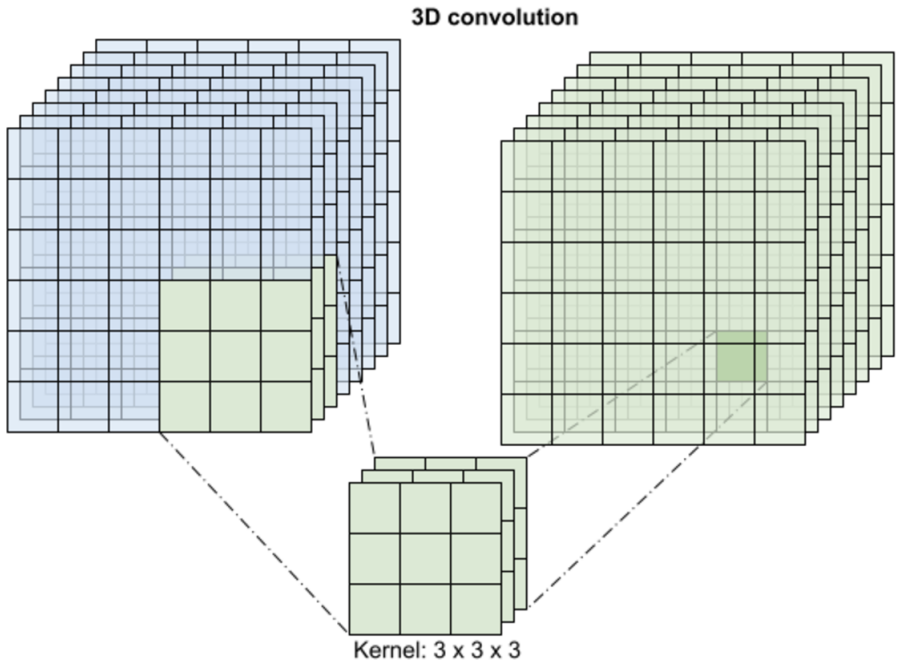


Figure 3: 3D Convolution

- **(2+1)D Convolution:** Factorizes the 3D convolution into:
  - A *spatial convolution* ($1 \times$ width $\times$ height).
  - A *temporal convolution* (*time* $\times 1 \times 1$).

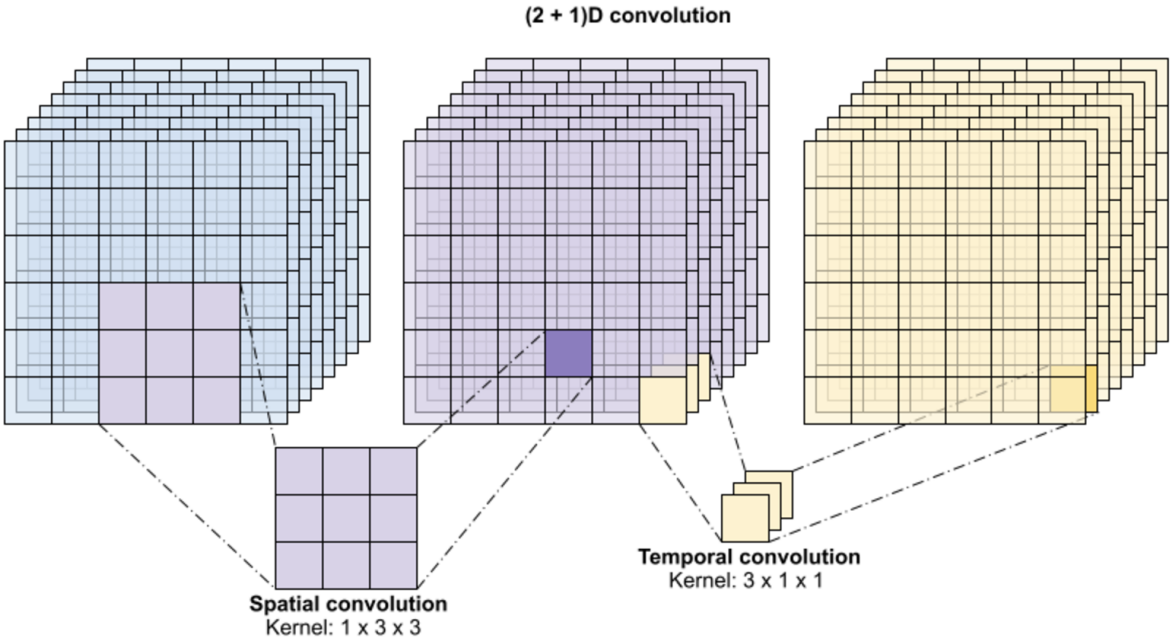


Figure 4: (2+1)D Convolution

- **Parameter Efficiency:** For the same kernel size, a (2+1)D convolution requires

$(9 \times \text{channels}^2) + (3 \times \text{channels}^2)$ parameters, less than half the parameters of a full 3D convolution.

We implements a (2+1)D Convolution with architecture in fig. 5. This approach takes the benefits of reduced parameters and improved spatiotemporal factorization. We used this model as baseline.

```
==========================================================================
Layer (type:depth-idx)                             Param #
==========================================================================
VideoModel                                         --
├─Conv2Plus1D: 1-1                                 --
│    └─Sequential: 2-1                             --
│    │    └─Conv3d: 3-1                            2,368
│    │    └─Conv3d: 3-2                            784
├─LayerNorm: 1-2                                   32
├─ReLU: 1-3                                        --
├─ResidualMain: 1-4                                --
│    └─Conv2Plus1D: 2-2                            --
│    │    └─Sequential: 3-3                        3,104
│    └─LayerNorm: 2-3                              32
│    └─Conv2Plus1D: 2-4                            --
│    │    └─Sequential: 3-4                        3,104
│    └─ReLU: 2-5                                   --
├─ResidualMain: 1-5                                --
│    └─Conv2Plus1D: 2-6                            --
│    │    └─Sequential: 3-5                        7,744
│    └─LayerNorm: 2-7                              64
│    └─Conv2Plus1D: 2-8                            --
│    │    └─Sequential: 3-6                        12,352
│    └─ReLU: 2-9                                   --
├─ResidualMain: 1-6                                --
...
==========================================================================
Total params: 442,810
Trainable params: 442,810
Non-trainable params: 0
==========================================================================
```

Figure 5: Structure of (2+1)D Convolution

VideoMAE

Video Masked Autoencoder is a deep learning model designed for video understanding tasks, particularly video classification. It builds upon the idea of Masked Autoencoders (MAE), extending it to video domain. While pretraining model learned representations by reconstructing missing pixels in a masked input without labels. It aims on both **spatial (frame content)** and **temporal (motion)** patterns from partially masked video sequences.

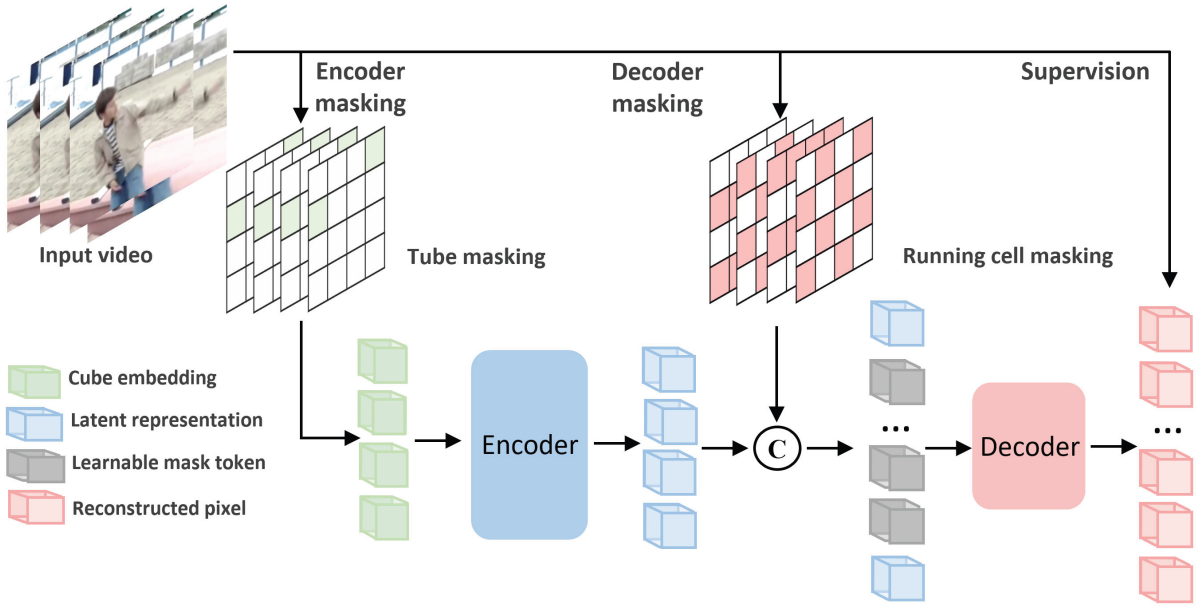


Figure 6: VideoMAE Structure

Models Comparison

	Parameters	Size (.safetensors)	Pretrained	Framework
(2+1)D Convolutions	0.4M	1.8MB	-	
VideoMAE	94.2M	330MB	Kinetics-400	

Deployment

We chose Gradio as tool for implementing user freindly interface of video classification task. It allows users to either upload their own videos or select from predefined options. It includes a dropdown menu for choosing the model version. In the backend, the chosen model is loaded and it processes the video to output the most probable class with its confidence percentage.

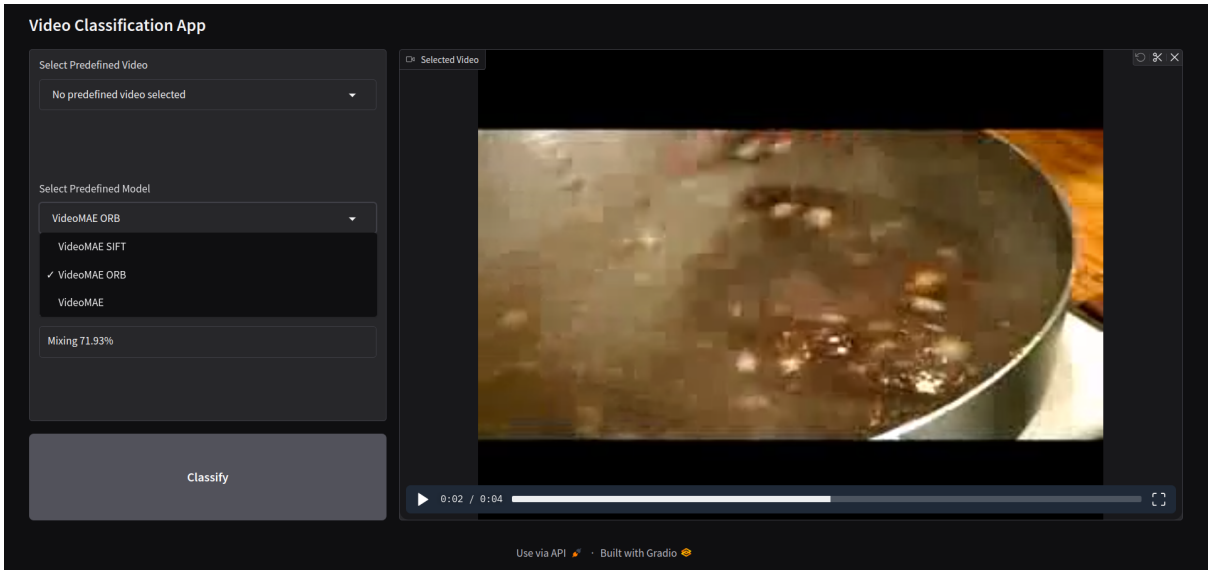


Figure 7: App usage example

GitHub Link

The project code and implementation details can be accessed at: **GitHub Repository Link**.

Project artifacts

Main project's artifacts are located on google drive.

Models Checkpoints and Dataset Versions

Model	Dataset
VideoMAE	UCF50
VideoMAE ORB	UCF50 ORB
VideoMAE SIFT	UCF50 SIFT

Experiments and Evaluation

1. Classical Frame Selection Strategies

(2+1)D Convolutions

For this model, we implemented two strategies for selecting frames from videos to ensure effective representation of temporal information:

1. Static Frame Selection

- Frames were extracted at fixed intervals from the video using a predefined step size.
- For example, with a step size of 15, every 15th frame was selected from the video.
- This method is straightforward and computationally efficient but may miss critical events if they occur between selected frames.

2. Adaptive Frame Selection

- The step size was dynamically calculated based on the total number of frames in the video and the desired number of output frames.
- The formula used was:

$$\text{frame\_step} = \lfloor \frac{\text{video\_length}}{\text{n\_frames}} \rfloor$$

- This approach ensures an evenly spaced selection of frames tailored to the video's length and the required number of frames.
- It provides a more adaptable solution compared to the static method.

Graphs of validation loss and accuracy is following:

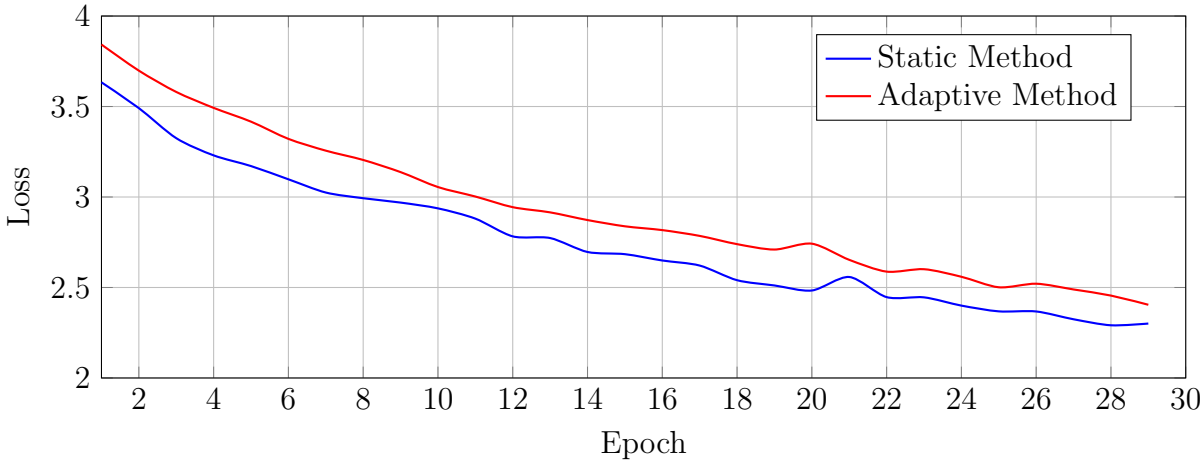


Figure 8: Validation Loss over Epochs for Two Methods

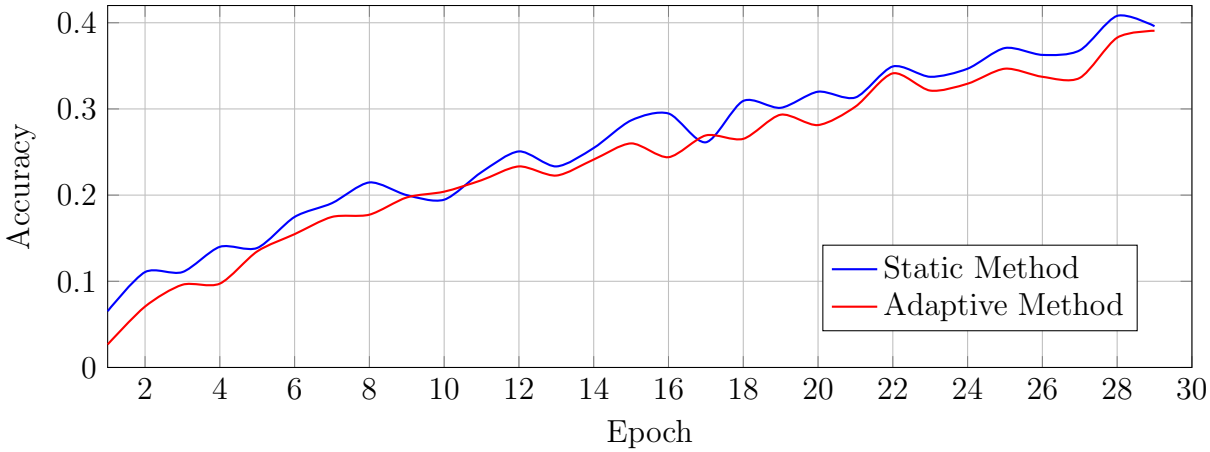


Figure 9: Validation Accuracy over Epochs for Two Methods

Using both methods for the baseline, we obtained almost the same results for the test part.

Method	Test Loss	Test Accuracy	Macro-average ROC-AUC
Static Method	2.288	0.376	0.93
Adaptive Method	2.401	0.364	0.92

Since the accuracy is relatively small, we decided to test our hypothesis about smart frame selection on the transformer.

VideoMAE

During VideoMAE fine-tuning, **random frame selection strategy** was used as augmentation mechanism. It is presented as following, it chooses initial timestemp and selects next frames randomly preserving the order.

During inference and validation, the **uniform strategy** is used, choosing predefined number of frames with equal intervals.

2. Feature-Based Frame Selection

To optimize memory usage and reduce the dataset size without losing performance, we implemented feature-based frame selection techniques using Scale-Invariant Feature Transform **(SIFT)** and Oriented FAST and Rotated BRIEF **(ORB)**. These methods enabled the extraction of keyframes that effectively represent the spatial and temporal features of each video while minimizing redundancy.

1. **Feature Extraction:**

   - **SIFT:** Detects and describes local features in frames, focusing on regions with significant texture or structure. It is robust to changes in scale, rotation, and illumination.

- **ORB:** A faster and computationally efficient alternative to SIFT, combining the FAST keypoint detector and BRIEF descriptor. It maintains robustness to scale and rotation.

2. **Frame Selection:**

   - From the entire video, frames with the most distinctive features were selected using SIFT or ORB.
   - A minimum of 30 frames was chosen to ensure sufficient representation of the video content.

3. **Code implementation:**

   - Frame selection using **ORB** is based on distance between consecutive frames:

```
keypoints, descriptors = orb.detectAndCompute(processed_frame, None)

# Compute the distance between descriptors
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(prev_descriptors, descriptors)
distances = [m.distance for m in matches]
avg_distance = np.mean(distances)

if avg_distance > threshold:
    relevant_frames.append(frame)
```

   - Frame selection using **SIFT** is based on number of the best matches between consecutive frames:

```
keypoints, descriptors = sift.detectAndCompute(processed_frame, None)

# Find the two best matches for current and previous frames
bf = cv2.BFMatcher()
matches = bf.knnMatch(prev_descriptors, descriptors, k=2)
good_matches = []
for match in matches:
    # Ensure at least two matches are returned
    if len(match) == 2:
        m, n = match
        if m.distance < 0.75 * n.distance:
            good_matches.append(m)
total_frames += 1
all_frames.append((total_frames, len(good_matches), processed_frame))
```

     After we have compared all the consecutive frames, we select the most dissimilar ones:

```
# Sort all frames by the number of good matches (ascending order)
all_frames.sort(key=lambda x: x[1])
selected_frames = all_frames[:min(len(all_frames), 2 * number_of_frames)]

# Reorder the selected frames based on their original temporal order
selected_frames.sort(key=lambda x: x[0])
relevant_frames = [frame[2] for frame in selected_frames]
```

4. **Recomposition:**

   - The selected frames were recomposed into a new video, significantly reducing the memory size of the dataset.

Advantages

- **Efficient Storage:** Selecting minimum 30 keyframes per video reduced the dataset's memory footprint without compromising essential content.
- **Feature Preservation:** SIFT and ORB ensured the selected frames preserved meaningful spatial and temporal features.

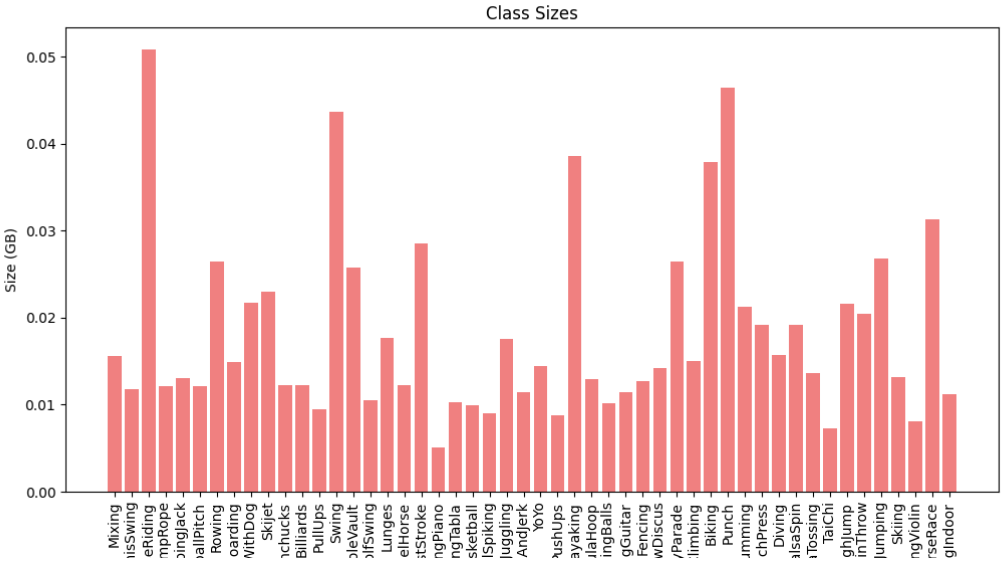


Figure 10: Size of Dataset after ORB

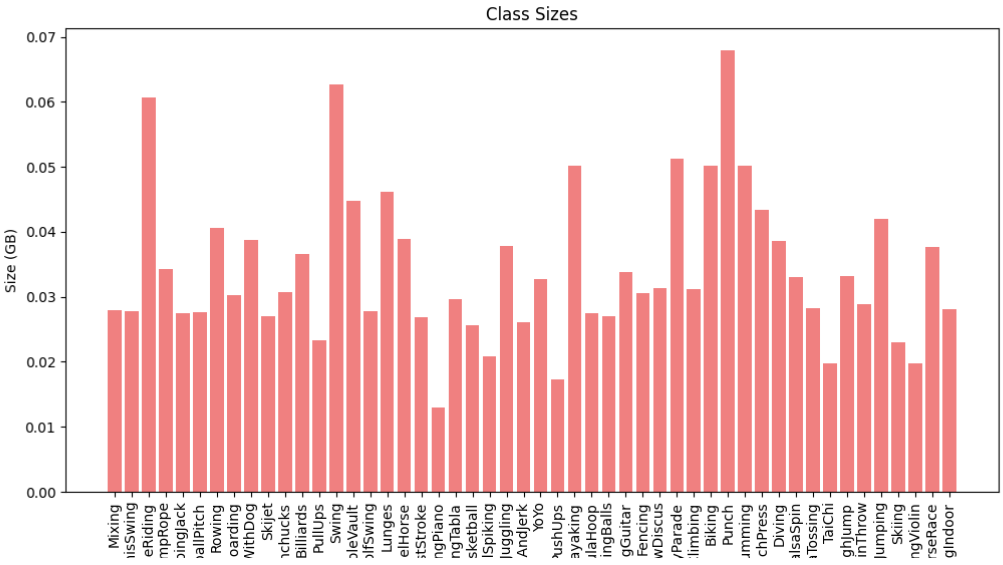


Figure 11: Size of Dataset after SIFT

The visual distinction we added in Method-Comparison google folder. These methods effectively reduced the size of the dataset as we can see in fig. 10 and fig. 11, making it suitable for resource-constrained environments by preserving the information that the data contains.

3. Evaluation metrics

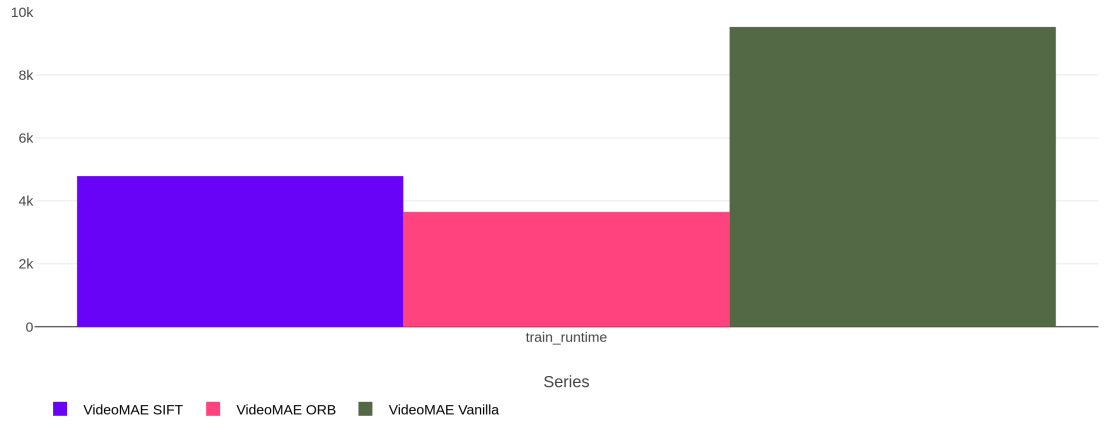
Size

As preprocessing techniques aim to reduce size with minimal information losses, the comparison of resulting dataset sizes seems reasonable:

Dataset	UCF50	UCF50 ORB	UCF50 SIFT
Size of zip file	3.23 GB	0,87 GB	1.62 GB

Train Time

With help of size reduction, we encounter less time required to train model. (in seconds)



VideoMAE Train Loss

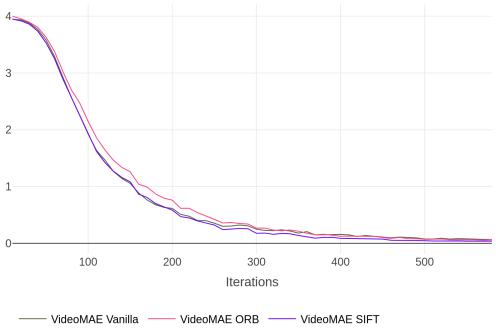


Figure 12: Based on iteration step

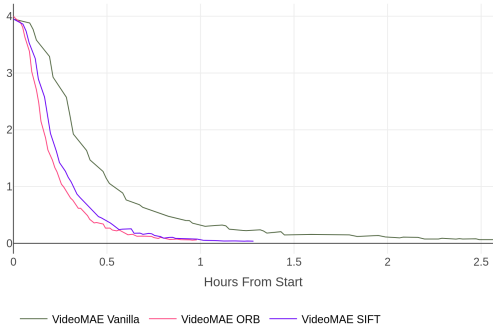


Figure 13: Based on time from start

VideoMAE Evaluation Loss

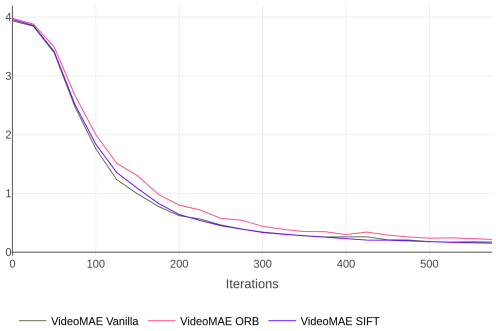


Figure 14: Based on iteration step

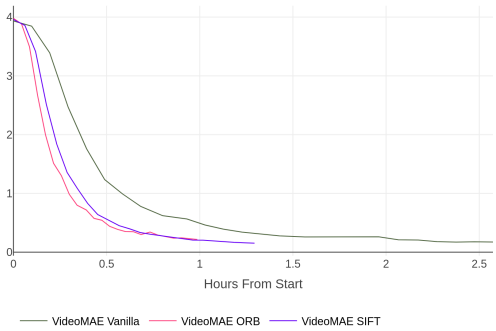


Figure 15: Based on time from start

VideoMAE Evaluation Accuracy

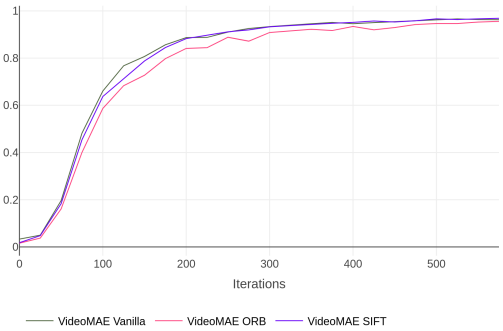


Figure 16: Based on iteration step

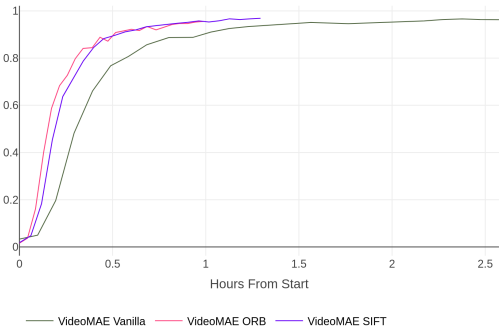


Figure 17: Based on time from start

VideoMAE Evaluation ROC-AUC metrics

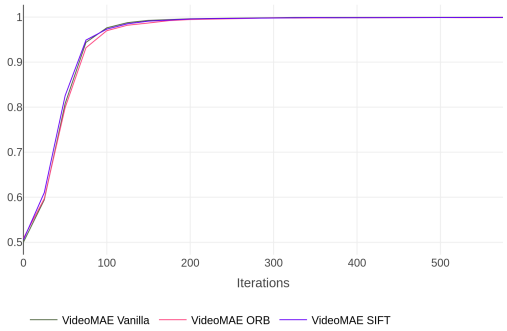


Figure 18: Based on iteration step

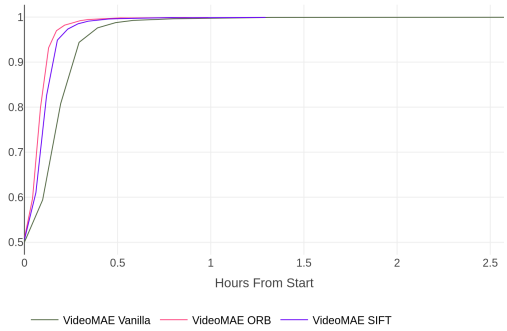


Figure 19: Based on time from start

Analysis and Observations

By selecting key frames and reducing their total amount in the dataset, processing time can be minimized without significantly degrading performance. Despite the smaller dataset, the model performance remains almost identical to that of training on the full dataset. This suggests that not all frames in a video contribute equally to model training, and selecting the most informative frames is a form of dimensionality reduction.

The ability to discard redundant or less informative frames allows for a more efficient use of computational resources, resulting in faster training and lower memory consumption, which is particularly useful for deploying models in resource-constrained environments.

Moreover, the smart selection methods allowed the models to converge faster because they start with a smaller and more relevant dataset, which likely simplifies the learning process. Faster convergence means that models reach optimal performance faster, reducing the need for lengthy training, which is a valuable advantage for both research and deployment in production.

Observations on video classification preprocessing techniques highlight that smart frame selection and dataset size reduction significantly improve efficiency. By selecting key frames and reducing redundant data, large models achieve faster and easier convergence, cutting down training time. Remarkably, this approach maintains performance metrics nearly on par with training on the full dataset, demonstrating that thoughtful preprocessing can optimize resources without sacrificing accuracy.

Conclusion

The analysis shows that using preprocessing techniques such as smart frame selection (using SIFT and ORB) and dataset reduction not only makes video classification more computationally efficient, but also maintains high performance. The ability to reduce the size of the dataset while maintaining accuracy shows that significant performance gains can be achieved through intelligent data selection, which is crucial for improving model deployment in real-world applications.

References

[1] A. Hussain, T. Hussain, W. Ullah, and S. W. Baik, "Vision transformer and deep sequence learning for human activity recognition in surveillance videos," *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, p. 3 454 167, 2022. DOI: https://doi.org/10.1155/2022/3454167.

[2] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[3] E. Karami, S. Prasad, and M. Shehata, *Image matching using sift, surf, brief and orb: Performance comparison for distorted images*, 2017. [Online]. Available: https://arxiv.org/abs/1710.02726.

[4] Z. Tong, Y. Song, J. Wang, and L. Wang, *Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training*, 2022. [Online]. Available: https://arxiv.org/abs/2203.12602.

[5] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, *A closer look at spatiotemporal convolutions for action recognition*, 2018. [Online]. Available: https://arxiv.org/abs/1711.11248.

[6] A. Makandar, D. Mulimani, and M. Jevoor, "Preprocessing step-review of key frame extraction techniques for object detection in video," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:40113593.

[7] S. N. Gowda, M. Rohrbach, and L. Sevilla-Lara, *Smart frame selection for action recognition*, 2020. [Online]. Available: https://arxiv.org/abs/2012.10671.

[8] M. Mao, A. Lee, and M. Hong, "Deep learning innovations in video classification: A survey on techniques and dataset evaluations," *Electronics*, vol. 13, p. 2732, Jul. 2024. DOI: 10.3390/electronics13142732.