Bouldering Video Segmentation

M1 Data Science Research Project

Centrale Lille - Université de Lille - IMT Nord Europe - Cristal Lab - Université de Rouen

Nadir Kichou [1] **Jérémie Boulanger** [2] **Ludivine Plumhans** [3] **Ludovic Seifert** [3]

Abstract

This research project focuses on temporal action segmentation in professional bouldering videos. The objective is to classify each frame of a video based on the climber's activity to enable a detailed analysis of their performance. The project leverages pre-trained networks, which are crucial for addressing the dataset's limited size by providing robust spatio-temporal feature extraction. These features are then combined with either a Multi-Layer Perceptron (MLP) or a Long Short-Term Memory (LSTM) network for classification. The dataset, collected during a bouldering competition, presents several challenges, including class imbalance, noise, and limited size.

Experimental results demonstrate that models incorporating temporal context significantly outperform frame-based approaches. Notably, the X3D-M model with an LSTM classifier achieves the highest accuracy of 86.61%. This work contributes valuable insights into effective model selection for bouldering video analysis and highlights potential improvements for real-world applications. The code and dataset used in this project are available at: https://github.com/raideno/bouldering-video-segmentation.

[1]Université de Lille [2]CRIStAL Lab, SIGMA [3]Université de Rouen.

Nadir Kichou <nadir.kichou.etu@univ-lille.fr>, Jérémie Boulanger <jeremie.boulanger@univ-lille.fr>, Ludivine Plumhans <ludivine.plumhans@univ-rouen.fr>, Ludovic Seifert <ludovic.seifert@univ-rouen.fr>.

1. Context

Bouldering Basics. It is a form of rock climbing where climbers tackle short but challenging routes, typically lasting around 4 minutes. During this time, climbers have the freedom to choose their climbing strategies, retry as many times as needed, and aim to complete the route as quickly as possible. The ultimate goal is to reach the top of the climb in the least amount of time.

Project Overview. This research project is part of a larger initiative (ANR) aimed at improving bouldering performance. The focus is on developing tools for coaches to analyze bouldering performances, allowing them to provide better advice to help climbers improve.

Existing Tools. Among these tools are: Grip Detection, Path Tracking, Performance Analysis, and more. Several models have already been developed as part of this broader effort, including the model for detecting the grips used by the climber and the climber's climbing path tracking model.

Phase Identification. These models provide valuable data, but in order to fully analyze a climber's performance, it is essential to distinguish between the different phases of a bouldering event (Climbing, Observing, Brushing). By doing so, we can apply the appropriate model to each phase and gather the corresponding statistics.

Research Focus. This brings us to the current research project. The project will focus on Temporal Action Segmentation (TAS), a technique that will enable us to identify the different phases of a bouldering event. By accurately distinguishing between phases such as observing, reading the route, and brushing the grips, we can gain deeper insights into the climber's performance. Analyzing the duration, order, and impact of these phases will allow coaches to provide more targeted guidance and improve training methods.

(a) Observing the block. (b) Climbing the block. (c) Brushing the grips. (d) Climbing the block.

Figure 1: Different phases of bouldering.

2. Problem Setup

Research Aim

This project focuses on the task of temporal action segmentation in professional bouldering videos. The goal is to classify each frame based on the climber's activity, enabling a more detailed analysis of the climber's performance throughout the video.

2.1. Video Representation

We represent each video as a sequence of frames $video_i = \{f_1, f_2, \ldots, f_{N_i}\}$, where each frame f_i corresponds to the image captured at position i in the video. Each frame f_i is a tensor with shape $(3 \times H \times W)$, where 3 denotes the number of channels, H is the height of the video, and W is the width of the video. The total number of frames in the video is denoted by N_i.

The i-th video's duration, d_i, in seconds is given by $d_i = N_i/F$, where F is the frame rate of the video. In our experiments, we fix $F = 25$ fps, which matches the frame rate used in our dataset.

2.2. Annotation Representation

Each video's annotations are provided at the frame level. Given a video $video_i$, its corresponding annotations are defined as $a_i = \{c_1, c_2, \cdots, c_{N_i}\}$, where $c_j \in C$ is the annotation corresponding to the frame f_i, and $C = \{\text{Climbing, Brushing, Observing, Stopwatch}\}$ is the set of all possible activity labels.

Alternatively, the annotations can be represented using start and end timestamps. In this format, the annotations are defined as a set of tuples:

$$\{(s_1, e_1, c_1), (s_2, e_2, c_2), \ldots\}$$

where s_j is the index of the starting frame for the j-th action segment, e_j is the index of the ending frame, and $c_j \in C$ is the corresponding action label.

Depending on the context, the starting frame s_j and ending frame e_j can alternatively be expressed as timestamps (e.g., seconds, milliseconds) or as a starting timestamp combined with a duration.

This alternative representation is more compact and especially useful when dealing with longer videos or when visualizing action intervals and it is the preferred format when annotating videos as it is easier to deal with. While the first representation provides more precision at the frame level, the second representation offers a higher-level view of the video's structure, facilitating analysis of action segments and their durations.

Note that it is easy to convert between these two representations by computing the starting and ending frames from the timestamps and vice versa.

2.3. Problem Formulation

The goal is to develop a model $\mathcal{M}$ that takes a sequence of frames as input and predicts the corresponding sequence of frame-level annotations, $\mathcal{M} : \{f_1, f_2, \ldots, f_{N_i}\} \rightarrow \{c_1, c_2, \ldots, c_{N_i}\}$.

Frame-wise Prediction. In this formulation, the model processes each frame independently and predicts its corresponding label. As the each frame is passed through a

convolutional layer or a similar architecture. The model predicts a label c_j for each frame f_j, i.e.,

$$c_j = \mathcal{M}(f_j)$$

While this approach is computationally efficient, it may struggle to capture temporal dependencies between consecutive frames.

Sequence-based Prediction. Here, the model processes a sequence of T consecutive frames and predicts a single label for the entire sequence. A larger segment length T allows the model to leverage more temporal context, improving its ability to recognize complex patterns. This can be expressed as:

$$c_{j:j+T-1} = \mathcal{M}(f_j, f_{j+1}, \ldots, f_{j+T-1})$$

In this approach, the model predicts the activity label for the entire sequence of frames, taking into account their temporal dependencies. However, this comes at the cost of increased computational complexity.

The choice between these approaches depends on the desired trade-off: a smaller segment length favors computational efficiency, while a larger segment length enhances the model's capacity to capture temporal dependencies. In the following T is taken between 4 and 32 frames depending on the model's variant.

2.4. Additional Statistical Measures

Besides the usual statistics, such as the mean and average duration, we also utilize the following two metrics, introduced in (Ding et al., 2023), to gain further insights into the data:

Repetition Score. It quantifies the degree of repetition of actions within a video. Using the notations defined earlier, it is expressed as:

$$\text{Repetition Score} = \frac{|\{c_j \mid c_j \in a_i\}|}{|a_i|}$$

where $|\{c_j \mid c_j \in a_i\}|$ denotes the number of unique action labels in the annotation sequence a_i, and $|a_i|$ represents the total number of annotated actions in the video. This score ranges from 0 to 1, where 0 indicates no repetition of actions, and 1 indicates that the same action is repeated throughout the video.

Order Variation Score. It measures the consistency of the order of actions across different videos or sequences. Using the previously defined notations, it is expressed as:

$$\text{Order Variation Score} = \frac{1}{|V|} \sum_{i,j \in V} d(a_i, a_j)$$

where $d(a_i, a_j)$ represents the pairwise distance between the annotation sequences a_i and a_j, and V is the set of all videos in the dataset. The score ranges from 0 to 1, where 0 indicates identical action order across all videos, and 1 indicates completely inconsistent action order.

2.5. Evaluation Metrics

During the project we are going to use the **Accuracy** as a metric to evaluate the model's performance. It is defined as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}.$$

Pros. Simple to compute and provides a clear overall performance indicator. **Cons.** Not informative for imbalanced classes, as high accuracy can be achieved by predicting the majority class.

Pyhton Package - TAS Helpers.

We introduce the `tas_helpers` (Temporal Action Segmentation Helpers) library, which contains implementations of various metrics and scores discussed above. Additionally, it offers utilities for visualizing video segmentations and more. The package is available at: https://github.com/raideno/tas-helpers.

3. Dataset

In this section, we present the dataset used in this project, detailing its construction, preparation, and some important statistics. We also compare our dataset to other popular datasets used in temporal action segmentation.

3.1. Raw Format

Our dataset was collected during the "Manip Chambery" bouldering event. It consists of videos filmed from two different angles of 10 climbers attempting to complete 2 distinct bouldering blocks (events) each. This resulted in a total of 20 unique climbs, or 40 videos in total, with each video having a duration of around 4 minutes.

The climbers' activities during the event are diverse. They may engage in actions such as brushing the holds, observing them, and climbing. These actions occur freely within the 4-minute duration of each video, providing a rich set of diverse activities.

As specified in 1, the videos were annotated by the climbers themselves, and the annotations were provided in raw Excel

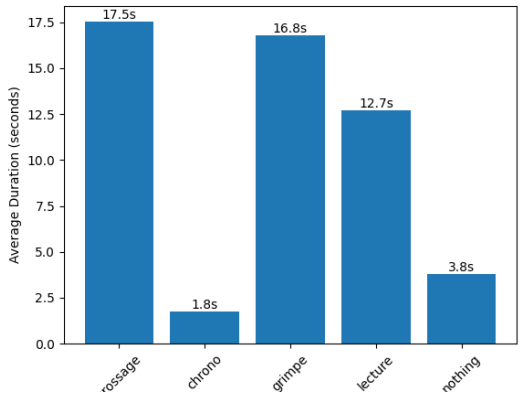

Figure 2: Average duration of actions in the dataset.

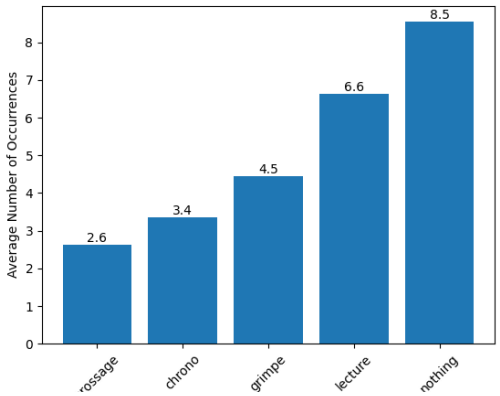

Figure 3: Average Action Count per Video.

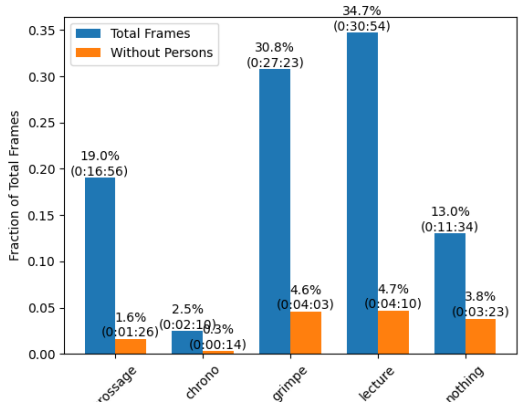

Figure 4: Distribution of actions in the dataset.

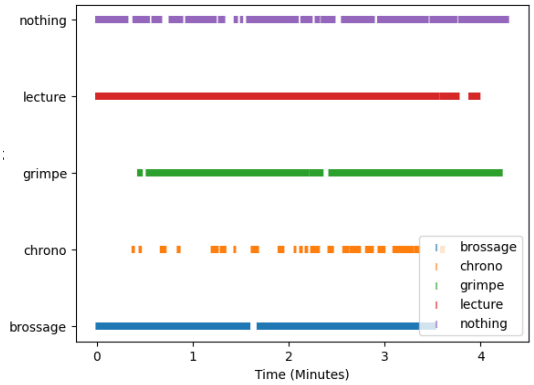

Figure 5: Temporal distribution of actions in the dataset.

format. The annotations are segment-level, meaning that for each action, we are given the start time and duration of the activity, along with the corresponding label. Out of the 20 climbs, only 11 have been annotated.

The total duration of the dataset is 2 hours and 40 minutes, of which 1 hour and 28 minutes is annotated, forming the basis of our experiments.

3.2. Chosen Structure

We decided to structure the dataset in a way that is specifically suited for temporal action segmentation and classification tasks, with a focus on both ease of use and efficiency for training. This structure is designed to allow for easy scalability as new data is added to the project, an important consideration as the dataset will be updated by individuals who may not have a background in data science.

```
dataset
+-- videos
|   +-- video-1
|   |   +-- frame-1.jpg
|   |   +-- frame-2.jpg
|   |   +-- ...
|   +-- ...
+-- annotations
|   +-- video-1.csv
|   +-- video-2.csv
|   +-- ...
```

As everything is automated by the developed tools, adding new data can be done very easily, ensuring the dataset can be expanded effortlessly in the future.

To optimize for video loading, we store each video as a sequence of JPG frames rather than the video file itself. This avoids the need for decoding during runtime, though it comes at the cost of additional storage space. This structure also makes the dataset more versatile and compatible with various libraries and tools.

Python Package - Video Dataset.

We have developed a Python package, https://github.com/raideno/video-dataset, to support this dataset structure. The package provides utilities for easily loading videos, transforming them into this frame-based format, and handling annotations at the frame level. It is highly customizable and can be adapted to work with different types of datasets.

By utilizing this structure and the provided tools, we were able to quickly load the dataset and begin training the model.

3.3. Data Exploration

In this section, we explore key aspects of our bouldering dataset to better understand its characteristics and potential challenges for model training.

Imbalance Ratio	Average Repetition Score	Order Variation Score
14.16	0.81 ± 0.04	0.49

Table 1: Dataset statistics

As shown in Figure 4, our dataset contains five action classes: cleaning ("brossage"), climbing ("grimpe"), observation ("lecture"), nothing, and break ("chrono"). The distribution is notably unbalanced, with "lecture" being the most frequent at 34.7%, followed by "grimpe" at 30.8%. The cleaning action ("brossage") represents 19.0%, while "chrono" accounts for only 2.5%. This imbalance is reflected in the dataset's imbalance ratio of **14.16** (Table 1), indicating a significant risk of the model overfitting to the dominant classes.

Figure 5 reveals that "grimpe" is less common at the beginning of videos, aligning with climbers preparing their attempts through cleaning or observation. Conversely, "lecture" and "brossage" actions are rare at the end of videos, suggesting that climbers tend to finalize their attempts rather than engage in preparation during these moments. This temporal pattern highlights the importance of incorporating time-dependent features in our model.

From Figure 3, we observe that "nothing" appears most frequently, averaging 8.5 instances per video. While this class is important for capturing pauses and transitions, its dominance risks overwhelming the model if not carefully balanced. Additionally, the low frequency of "chrono" (3.4 occurrences per video) may challenge the model's ability to accurately detect this class.

Figure 2 highlights further complexities: "brossage" has the longest average duration at 17.5 seconds, while "chrono" has the shortest at just 1.9 seconds. The brief duration of "chrono," combined with its low frequency, makes it particularly vulnerable to misclassification.

Lastly, from Figure 1 we can see that the dataset's **average repetition score of 0.81** suggests a high degree of repetitive patterns in action sequences, which may lead the model to exploit these patterns rather than learning meaningful action transitions. Additionally, the **order variation score of 0.49** reveals some variability in action order, which may require the model to adapt to less predictable sequences.

In addition to these challenges, from Figure 4 we can see that approximately 15% (4) of the frames contain no visible person, and 3% include multiple people, introducing further noise. Moreover, 13% of climbs lack proper annotations and are labeled as "nothing" by default.

These insights highlight key challenges for our model: class imbalance, varying action durations, repetitive patterns, and noisy data. Addressing these issues will be crucial for achieving robust action segmentation performance.

3.4. Dataset Limitations.

While the dataset provides valuable data for action segmentation tasks, it has several limitations:

1. The dataset size is relatively small, containing only 11 unique videos.
2. The annotations are provided in Excel format, which is not ideal for modern data science workflows.
3. The annotations themselves are not perfect, as they may be shifted in time, and the videos do not always start precisely at the beginning of the event.
4. Climbers occasionally leave the frame during the video, and in some instances, other people may enter the frame, which introduces noise to the data.
5. The dataset includes a 'nothing' class, representing periods with no relevant activity, which can further imbalance the dataset and affect model performance.

Despite these limitations, this dataset provides an essential foundation for the development and evaluation of action segmentation models in the context of bouldering.

3.5. Popular Video Datasets

While there are a number of video datasets available for action recognition and segmentation tasks, the overall number remains limited compared to image or text datasets. Video datasets are much more complex due to the added temporal dimension and therefore require a significantly larger number of samples for effective model training. For instance, while image datasets might need millions of samples, video datasets typically require tens of millions of samples due to the increased complexity.

Some of the most notable video datasets designed for temporal action segmentation include:

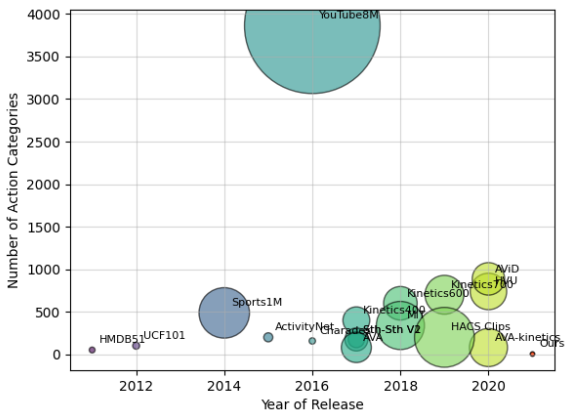

Figure 6: A visualization of popular video datasets, showcasing their relative sizes.
(Zhu et al., 2020)

Dataset	Release Year	#Samples	#Actions
HMDB51	2011	7000	51
UCF101	2012	13300	101
Sports1M	2014	1100000	487
ActivityNet	2015	28000	200
YouTube8M	2016	8000000	3862
AVA	2017	385000	80
⋮	⋮	⋮	⋮
MIT	2018	1000000	339
HACS Clips	2019	1550000	200
HVU	2020	572000	739
AViD	2020	450000	887
Ours	2024	22	4

Table 2: A list of popular video datasets in comparison with ours.
(Zhu et al., 2020)

50 Salads, GTEA, and Breakfast Datasets. These datasets focus on activities involving food preparation, such as making salads or breakfast, with fine-grained action segmentation annotations. They are widely used for temporal action segmentation benchmarks (McKenna & Stein, 2012), (Ding & Xu, 2018), (Kuehne et al., 2014).

Kinetics Family. Large-scale datasets containing YouTube clips of various human actions, ranging from sports to human-object interactions (Kay et al., 2017), (Carreira et al., 2018), (Carreira et al., 2019).

Assembly101 Dataset. Videos of people assembling and disassembling objects, captured from multiple angles, including some POV (Sener et al.).

HowTo100M Dataset. A massive dataset of instructional (tutorial) videos, covering a broad range of activities (Miech et al., 2019).

Something-Something V2. Videos of basic actions such as picking up a pen, captured from a POV perspective, focusing on fine-grained interactions (Goyal et al., 2017).

Table 2 and Figure 6 provides an overview of these datasets, including their sizes and key characteristics.

4. Related Work

In this section, we explore some of the prominent approaches to temporal action segmentation (TAS), discussing how methods have evolved over time as computational power and dataset availability have increased.

4.1. The Evolution of Video Analysis Models

The development of video analysis models has seen a significant shift as computational power and dataset size have increased. Initially, video analysis relied heavily on handcrafted feature extraction (Laptev, 2005; Wang & Schmid, 2013), where specific features such as pixels were manually chosen or extracted. With the introduction of optical flow, models began incorporating motion information (Carreira & Zisserman, 2018), improving their ability to understand temporal sequences in videos.

As 2D Convolution Neural Networks (CNNs) gained popularity, they were applied to video data for feature extraction, showing promising results for action recognition (Wang et al., 2016). Following this, the introduction of 3D Temporal CNNs (Xie et al., 2017; Hara et al., 2017; Feichtenhofer, 2020; Carreira & Zisserman, 2018; Fan et al., 2020) allowed for better understanding of the temporal dimension in videos. These methods, combined with the advent of larger datasets like Kinetics (Kay et al., 2017) and Something-Something (Goyal et al., 2017), marked a turning point in video action recognition, leveraging deep learning to handle complex temporal and spatial patterns in video data.

In recent years, transformers have also emerged as a promising approach for video analysis. Initially used for NLP tasks, transformers have demonstrated strong potential for video action recognition (Arnab et al., 2021; Oquab et al., 2023; Wang et al., 2021). However, unlike CNNs, transformers lack strong inductive biases (Dosovitskiy et al., 2020), such as locality and translation invariance, which CNNs excel at. This makes transformers more dependent on massive amounts of data to perform effectively.

The increasing availability of video datasets and computational resources has catalyzed these developments, enabling

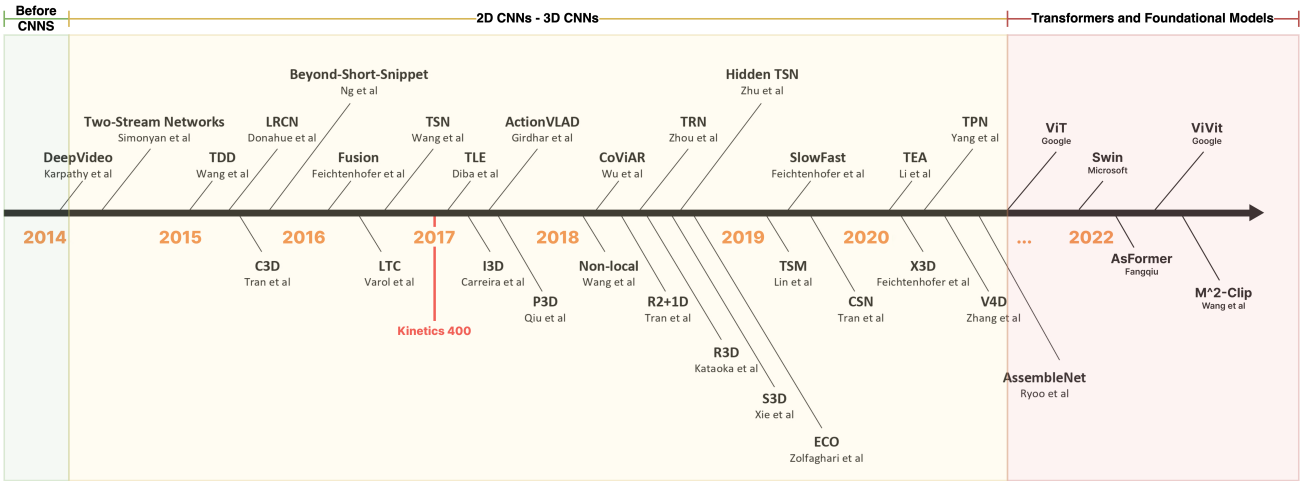

Figure 7: A timeline overview of the evolution of video analysis models. (Zhu et al., 2020)

more sophisticated models. However, video models still face significant challenges due to the vast and diverse distribution of video data. This distribution causes models to drift more rapidly compared to other data types, which makes generalization across different datasets more difficult. Consequently, large datasets are essential for training video models, and pre-trained models often struggle to generalize beyond their original training sets. Moreover, networks with a large number of parameters are particularly prone to overfitting when trained on smaller datasets (Hara et al., 2017).

4.2. Temporal Action Segmentation and Its Approaches

Temporal Action Segmentation can be approached in several ways. Here, we explore a few key perspectives (Ding et al., 2023):

1. Video Classification. In video classification, the focus is on assigning a single action label to an entire video or segment, typically using models such as 2D or 3D CNNs. This is the most popular perspective in the field, with well-established pre-trained networks available for various datasets.

2. Temporal Action Detection / Localization. In this approach, the goal is to detect the temporal boundaries of actions within a video. This task is more challenging than video classification as it requires the model to identify both the start and end of actions while also classifying them.

5. Our Approach

Given the dataset size constraint, training a neural network from scratch with our data is infeasible and unlikely to yield meaningful results. To address this, we leverage pre-trained networks trained on large-scale datasets that contain similar types of actions to those found in our data. These actions are general, composed of multiple smaller movements, and not very detailed, making them comparable to activities present in popular datasets like Kinetics.

Our strategy involves utilizing these pre-trained networks to extract feature representations from our videos, clips, and frames. Specifically, we extract features from the hidden layers of these networks — typically the layer immediately preceding the final classification layer — as they encode rich spatio-temporal information relevant to our task. These extracted features are then fed into a custom network designed for classification on our custom action labels.

This approach addresses multiple constraints simultaneously. First, it mitigates the impact of our limited dataset by capitalizing on robust features learned from extensive pre-training. Second, it reduces the computational burden, as we avoid the costly process of training a deep network from scratch. Finally, this method is well-suited to our project timeline, as designing and training a complex architecture would exceed our allocated 120 hours for this research project.

In the following sections, we describe the key components of our approach described in Figure 8 and how we train it.

5.1. Spatio-Temporal Feature Extractors

The backbone of our model is a spatio-temporal feature extractor designed to capture both spatial and temporal information from video data. We employ pre-trained models for this component, freezing their weights to retain the learned representations.

Two types of models are explored for this purpose: frame-level feature extractors and segment-level feature extractors.

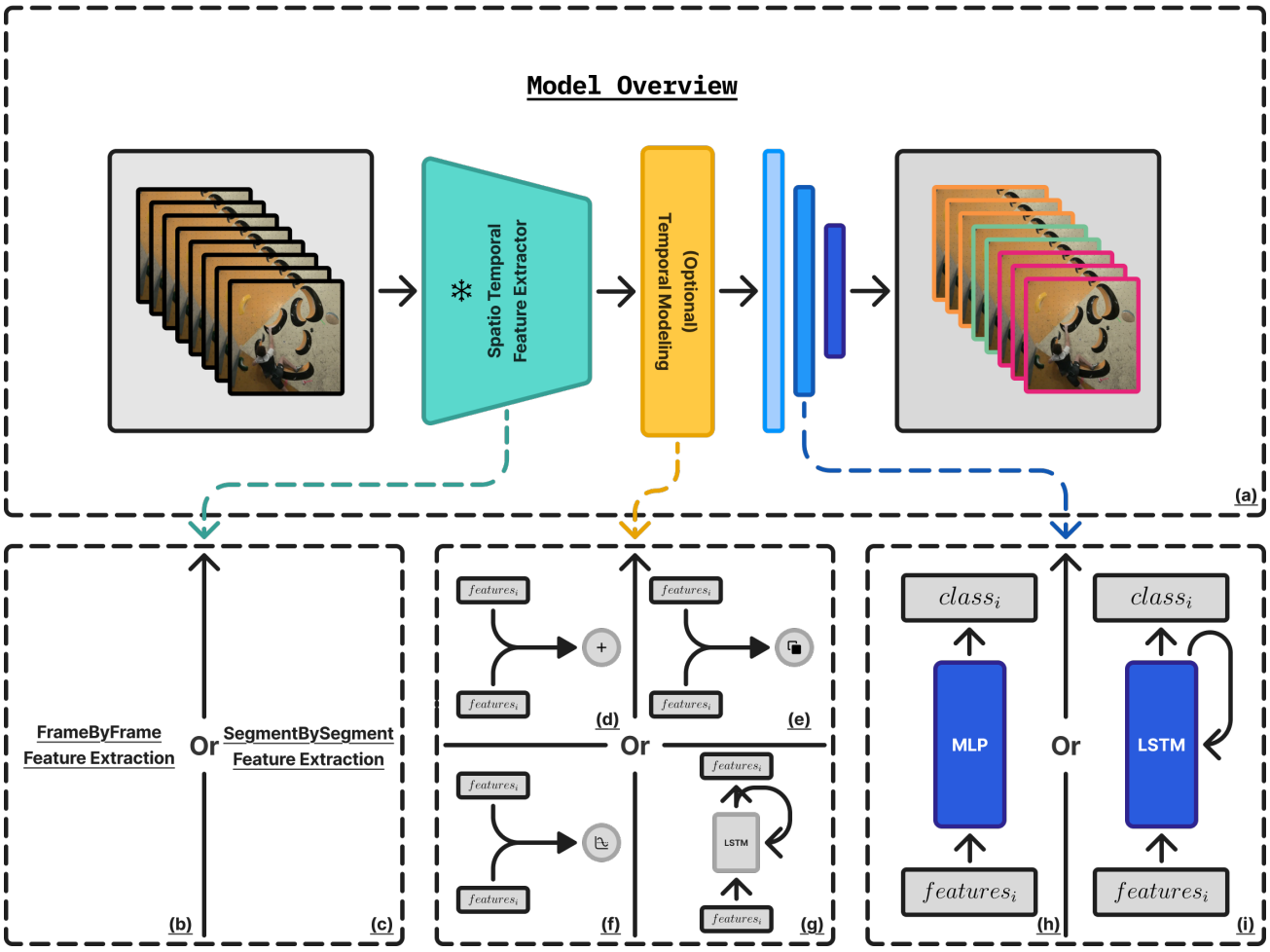

Figure 8: Model Architecture Overview.

(a) Represent a global overview of our Model's architecture. (b) and (c) represent the types of spatio temporal feature extractors considered. (d), (e), (f) and (g) represent the different ways of aggregating the frame by frame feature into a single feature representing a sequence of frames. (h) and (i) represent the two architectures of feature classifiers we considered, and MLP and an LSTM respectively.

5.1.1. Frame-Level Extractors

Frame-level extractors process individual frames independently to extract spatial features. Examples include 2D CNNs such as **ResNet**, vision transformers like **DINO**, **CLIP**, or **ViT**, and potentially pseudo handcrafted features such as object presence (e.g., brushes) and / or positional key points.

While (pseudo) handcrafted features can be interpretable and informative, they introduce biases and require additional training for object detection — to detect the presence and position of a brush for example — which is resource-intensive. Thus, our approach prioritizes automated feature extractors to maximize efficiency and robustness.

Once frame-level features are extracted, we explore several strategies to aggregate these features into a single representation for each video segment:

Averaging. The most common method, recommended in the literature (Oquab et al., 2023) for its simplicity and effectiveness. **Addition.** Similar to averaging but involves summing the features directly. **Concatenation.** Combines frame features without any additional operations. According to (Oquab et al., 2023), this method is the most effective in capturing temporal information & thus it is the one we opted to use. **Temporal Modeling.** Uses models like LSTMs or Transformers to capture temporal dependencies, offering improved contextual understanding at the cost of higher computational complexity. (Wang et al., 2021).

5.1.2. Segment-Level Extractors

Segment-level extractors process multiple consecutive frames directly, capturing both spatial and temporal patterns. Examples include 3D CNNs, which extend 2D CNN architectures to model motion and temporal dynamics, and

transformer-based models that have demonstrated strong performance on video tasks. While effective, these models are typically more computationally demanding.

A list of all the spatio-temporal feature extractors we considered can be found in the Appendix.

5.2. Temporal Context Window

The temporal context window defines the number of consecutive frames the model processes at once. Choosing an optimal window size is crucial:

Short Window. It may not capture sufficient context for accurate classification.

Long Window. It risks embedding the entire video into a single feature, increasing computational costs and reducing model flexibility.

As the temporal context window of the backbone networks is quite limited to around 16-32 frames usually, processing the whole 4 minutes of video and extracting a single feature for the entire video is both computationally costly and not practical in our case. This would be more practical for video classification tasks, where the goal is to classify the whole video. In our case, we aim to extract different time segments and classify them individually.

5.3. Temporal Sub-Sampling

To improve efficiency without sacrificing performance, we incorporate temporal sub-sampling. This technique involves reducing the frame rate before feature extraction, as consecutive frames in bouldering videos often exhibit minimal visual change. Temporal sub-sampling is a simple yet highly effective method for reducing computational load without significant performance degradation (Scheidegger et al., 2017).

By combining pre-trained feature extractors, temporal subsampling, and thoughtful hyperparameter tuning, our approach aims to achieve robust performance despite dataset limitations.

5.4. Classifier Model

For classification, we consider two different model architectures: a Multi-Layer Perceptron (MLP) and a Long Short-Term Memory (LSTM) network. Both approaches are designed to process the spatio-temporal features extracted from the pre-trained models.

MLP Approach. The MLP used in our approach consists of a simple linear layer with no activation function and a bias term. This architecture has been shown to yield the best performance in our case, as more complex network structures tend to overfit the limited training data. The simplicity of the MLP helps avoid overfitting while maintaining efficiency, making it ideal for our constraints. The output of this linear layer is fed directly into the softmax function to produce class probabilities for the classification task.

LSTM Approach. The LSTM used in our approach consists of a single LSTM layer with a hidden state size of 128. This configuration enables the model to capture temporal dependencies between frames or video segments, which is critical for action recognition tasks where context across time is crucial. The LSTM's output is passed through a linear layer (similar to the MLP) to predict the class for each segment.

Both models were validated using cross-validation to identify the most effective hyperparameters for our dataset.

5.5. Training Methodology

Backbone Model

We experimented with several backbone models for feature extraction, leveraging both frame-level and segment-level networks. Notable examples include:

DINO and **CLIP.** Vision Transformers used as frame-level feature extractors for their strong spatial representation capabilities.

X3D, **R3D**, and **Slowfast.** Segment-level extractors known for their robust spatio-temporal modeling.

Most of the backbones we use are pre-trained or fine-tuned on the Kinetics dataset, except for **CLIP**, which leverages large-scale internet data.

Loss Function

Our classification model is trained using the **cross-entropy loss** function, defined as:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log \hat{y}_i$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for class i. While cross-entropy loss effectively handles multi-class classification, alternative losses such as focal loss or label-smoothing loss have been proposed in the literature to address class imbalance and calibration issues. These approaches may be worth exploring in future work.

Training Setup

To prevent data leakage during validation, we ensured that segments from the same video were never split across training and validation sets. This avoids the model unintentionally learning video-specific patterns rather than generaliz-

able features.

Data Filtering and Preprocessing

To improve data quality and enhance model performance, we applied several preprocessing steps:

Removal of Personless Frames. Frames without visible climbers were excluded to reduce noise. This was done by detecting the presence of a person in each frame using a pre-trained object detection model (**YOLO**) (Redmon et al., 2015).

Cleaning of Low-Quality Data. Unannotated frames were removed rather than assigning them a generic "nothing" label. This improved class balance and reduced label ambiguity.

Stopwatch Class Removal. We removed segments with the stopwatch class, as they were highly imbalanced and lacked distinction. Climbers often exit the frame while checking the timer, and it varies across competitions, making it inconsistent in our dataset.

Python Package - Cached Dataset.

To accelerate training, we developed a custom Python package called **cached-dataset** - https://github.com/raideno/cached-dataset. This tool caches the transformation of a dataset to disk, significantly reducing data loading overhead during training and feature extraction. This allow us to efficiently iterate through training experiments and hyperparameter tuning.

By combining robust pre-trained feature extractors, temporal sub-sampling, and thoughtful data filtering strategies, our training methodology effectively addresses the challenges posed by our limited dataset. These strategies ensured a stable and efficient training process while maximizing model performance.

6. Results

6.1. Performance Analysis

Table 3 presents the classification accuracy of various backbone architectures combined with either MLP or LSTM classifiers for bouldering video segmentation. The results reveal several interesting patterns that provide insights into the effectiveness of different approaches for this specific task.

6.1.1. Segment vs. Frame-based Extraction

Our experiments demonstrate a clear advantage for models that incorporate temporal information through segment-level processing. As shown in Table 3, segment-based models consistently outperform frame-based models, with the top-performing models (**X3D family**) all utilizing temporal information. The **X3D-S** model achieves 85.28% accuracy with an MLP classifier, while **R3D** reaches 86.61% accuracy when combined with an LSTM. This pattern aligns with the intuitive understanding that climbing actions involve temporal dynamics that cannot be fully captured by analyzing individual frames or segments in isolation.

6.1.2. Analysis of Underperforming Models

Two backbone architectures exhibit notably lower performance compared to others:

YOLO-based Skeleton Features. The **YOLO**-based approach, which extracts skeleton key points, achieves only 65.84% accuracy with MLP and 70.61% with LSTM. This underperformance can be attributed to the similarity in climber movement dynamics across different action categories. For instance, the speed and pattern of movement during observing, brushing, and climbing activities may exhibit similar skeleton motion signatures despite being semantically distinct. A potential improvement would be to incorporate absolute spatial positions rather than positions relative to the climber's center of mass. However, this approach would introduce camera position dependency, potentially reducing generalizability across different recording setups.

S3D with HowTo100M Pre-training. The **S3D** model pre-trained on HowTo100M (**S3D-H**) shows particularly poor performance (59.52% with MLP, 43.25% with LSTM). This can be explained by the nature of the pre-training dataset, which consists primarily of instructional videos featuring fine-grained hand manipulations and subtle movements. The same architecture pre-trained on the Kinetics dataset (**S3D-K**), which contains more diverse and dynamic whole-body activities, performs substantially better (78.38% with MLP, 78.57% with LSTM). This significant performance gap highlights the critical importance of selecting appropriate pre-training datasets that align with the target domain's action characteristics.

6.1.3. Model Complexity and Performance

Interestingly, our experiments reveal that larger models do not necessarily yield better performance for this task. The **X3D-S** model (3.0M parameters) outperforms the larger

Model Information				MLP - Accuracy				LSTM - Accuracy			
Backbone	Type	#Params	#Frames (Seconds)	Overall Accuracy	Brushing Class	Reading Class	Climbing Class	Overall Accuracy	Brushing Class	Reading Class	Climbing Class
yolo	By Frame	2.9M	8 (0.32)	65.84% ± 3.93	33.63% ± 8.67	72.81% ± 7.96	78.87% ± 6.49	70.61% ± 3.62	54.79% ± 20.18	75.30% ± 9.07	80.23% ± 3.05
dino	By Frame	22.1M	8 (0.32)	80.53% ± 5.02	51.41% ± 21.74	84.98% ± 4.91	90.68% ± 4.32	82.64% ± 3.15	70.98% ± 4.63	82.68% ± 7.93	**92.76**% ± 3.45
clip	By Frame	151.3M	8 (0.32)	77.08% ± 1.88	48.76% ± 16.31	77.66% ± 9.70	91.05% ± 3.56	78.72% ± 3.04	58.30% ± 10.55	85.84% ± 5.69	89.55% ± 5.02
r3d	By Segment	31.6M	8 (0.32)	84.15% ± 4.59	63.14% ± 19.01	86.92% ± 3.79	**92.22**% ± 2.87	**86.80**% ± 3.50	83.46% ± 4.18	87.06% ± 3.67	89.71% ± 5.64
i3d	By Segment	12.7M	16 (0.64)	77.61% ± 7.96	56.66% ± 13.86	77.69% ± 6.42	89.88% ± 5.74	79.38% ± 5.68	71.62% ± 5.64	75.79% ± 16.07	89.04% ± 6.34
x3d-xs	By Segment	3.0M	4 (0.16)	81.64% ± 3.76	62.65% ± 12.53	82.48% ± 6.19	90.58% ± 4.65	84.90% ± 2.68	81.50% ± 4.99	83.73% ± 3.82	89.80% ± 3.95
x3d-s	By Segment	3.0M	16 (0.64)	84.89% ± 4.90	69.96% ± 14.65	**88.83**% ± 3.71	88.48% ± 5.21	85.97% ± 2.84	**84.88**% ± 5.92	82.70% ± 4.17	89.97% ± 5.78
x3d-m	By Segment	3.0M	16 (0.64)	**85.39**% ± 4.16	**72.03**% ± 16.10	87.13% ± 4.09	90.50% ± 4.55	85.65% ± 2.22	84.00% ± 9.41	84.45% ± 6.60	89.16% ± 2.85
x3d-l	By Segment	5.3M	16 (0.64)	84.79% ± 4.04	69.36% ± 16.20	86.55% ± 3.81	91.35% ± 2.53	85.46% ± 1.47	84.26% ± 6.07	84.41% ± 8.37	87.82% ± 4.21
s3d-k	By Segment	7.9M	16 (0.64)	78.38% ± 5.21	57.96% ± 11.23	77.66% ± 11.26	89.65% ± 2.84	78.57% ± 4.06	77.65% ± 4.85	73.37% ± 5.84	84.53% ± 7.77
s3d-h	By Segment	7.9M	16 (0.64)	59.52% ± 8.90	00.00% ± 00.00	78.96% ± 7.35	76.79% ± 15.54	43.25% ± 4.74	23.56% ± 34.56	42.93% ± 29.85	58.29% ± 43.94
slowfast	By Segment	33.6M	32 (1.28)	84.06% ± 2.94	70.79% ± 12.22	85.00% ± 2.27	90.47% ± 1.33	85.16% ± 1.80	79.18% ± 7.60	**88.54**% ± 6.68	87.82% ± 3.96
vivit	By Segment	88.6M	32 (1.28)	77.65% ± 5.19	45.44% ± 14.48	81.71% ± 8.27	90.93% ± 5.70	81.46% ± 2.21	72.71% ± 9.25	82.14% ± 6.72	87.14% ± 5.72

Table 3: Models Performance Results.

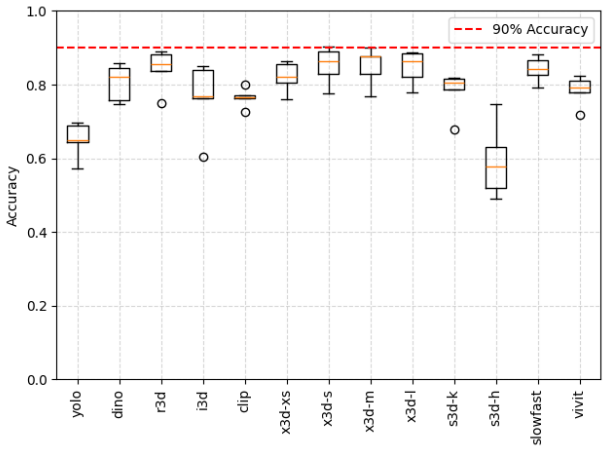

Figure 9: MLP Model Training Results

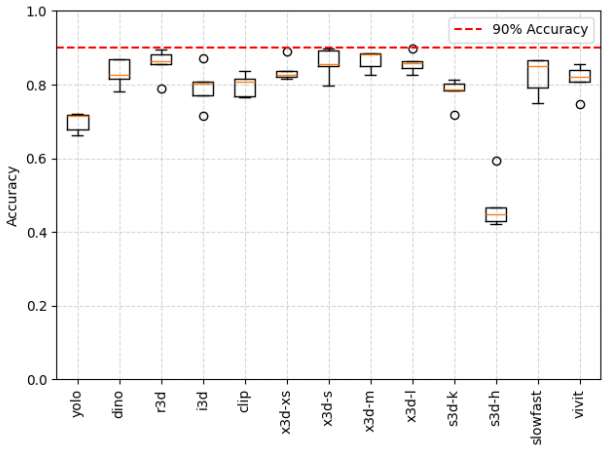

Figure 10: LSTM Model Training Results

X3D-L variant (5.3M parameters) when using an MLP classifier. Similarly, the relatively lightweight **R3D** backbone (31.6M parameters) achieves better results to much larger models such as **CLIP** (151.3M parameters) and **ViViT** (88.6M parameters). This suggests that for our specific dataset size and task complexity, model architecture design is more important than raw parameter count. The **X3D** family, designed specifically for efficient video understanding, demonstrates excellent performance-to-parameter ratios across all variants.

6.1.4. Temporal Modeling with LSTM

When comparing MLP and LSTM classifiers, we observe that LSTM generally provides modest improvements across most backbones. For instance, **DINO** shows a 2.11% improvement when switching from MLP to LSTM. However, this pattern is not universal—SlowFast actually performs 1.73% worse with LSTM compared to MLP. The inconsistent benefits of additional temporal modeling through LSTM may be due to our dataset size constraints, as larger datasets typically show more significant improvements from temporal modeling, as demonstrated in prior work (Wang et al., 2021).

Beyond improvements in accuracy, we also observe a reduction in variance across multiple runs when using LSTM classifiers. For example, the **X3D-M** model shows a standard deviation of 4.16% with MLP but only 2.22% with LSTM, representing a nearly 50% reduction in variance. This increased stability is a significant advantage in practical applications, as it indicates more reliable and consistent performance across different climbing sessions and environmental conditions.

6.2. Practical Implications

Based on our experimental results, we can draw several conclusions to guide model selection for practical bouldering video segmentation applications:

For accuracy-critical applications. The **R3D** with LSTM classifier provides the highest overall accuracy (86.80%) and represents the best choice when classification performance is the primary concern.

For resource-constrained environments. The **X3D-XS** with MLP classifier offers an excellent balance between

accuracy (81.64%) and computational efficiency, utilizing only 3.0M parameters. And this model provide the best trade-off between speed and accuracy.

These findings provide valuable insights for climbing gym operators, sports coaches, and performance analysts working with climbing videos. For coaches analyzing technique, the highest accuracy models would better distinguish between different climbing phases, while training facilities with limited computing resources could implement the lighter models for real-time feedback systems. The ability to reliably segment climbing activities enables more targeted training programs and better performance assessment for climbers at all levels.

7. Limitations & Future Directions

While our approach demonstrates promising results for bouldering video segmentation, several limitations remain to be addressed in future work.

7.1. Methodological Limitations

Our current implementation faces several methodological constraints. First, the cross-entropy loss function used may not be optimal for temporal segmentation tasks with class imbalance. Alternative losses could potentially improve performance on underrepresented classes. Second, despite the strong performance of convolution architectures, we did not extensively explore transformer-based approaches specifically designed for video understanding, such as TimeSformer (Bertasius et al., 2021) or MViT (Li et al., 2021), which have shown state-of-the-art results in related action recognition tasks.

Real-time processing remains a significant limitation of our current approach. The segment-based models that achieved the highest accuracy operate with inherent latency due to their temporal window requirements. However, for practical applications, coaches usually don't require real-time annotations.

Another limitation is that we did not explore varying temporal context sizes. Investigating different segment lengths, particularly with backbones that support flexible context sizes, would help assess the stability of our findings across different temporal resolutions.

7.2. Data-Related Challenges

Limited training data remains a fundamental constraint. Our dataset, while sufficient to demonstrate the viability of automated bouldering segmentation, lacks the scale and diversity needed for robust real-world deployment. We identify several potential approaches to address this:

Semi-supervised Learning. Using our best-performing models to pseudo-label unlabeled climbing videos, then leveraging these annotations for additional training data. This bootstrapping approach could substantially increase our effective dataset size while maintaining a manually verified validation set to avoid bias.

External Data Integration. Augmenting training with targeted external data from sources like YouTube or subsets of action recognition datasets like Kinetics that contain climbing or similar activities. This would require careful selection and possibly domain adaptation techniques to ensure relevance.

Data Augmentation. More advanced video-specific augmentation techniques, such as temporal shifts, speed variation, and generating new views from different angles, could enhance model robustness to variations in climbing styles and camera positions (Carreira & Zisserman, 2018).

7.3. Model Limitations

Our current model architecture may also benefit from additional enhancements. One potential improvement is integrating an attention mechanism to allow the model to focus on the most relevant frames or segments. This could improve performance by emphasizing key moments in the video; however, this would also increase the model's complexity, thus raising training time and computational costs.

Another promising direction is incorporating fixed or learnable positional embeddings in combination with the MLP to improve temporal awareness. This enhancement could enable the model to better understand the sequential nature of climbing activities, particularly in scenarios where the temporal structure is crucial.

Finally, subsampling segments and interpolating between classified segments may offer a practical way to reduce inference costs while maintaining reliable predictions. This technique could improve the model's efficiency, especially in resource-constrained environments or when analyzing lengthy videos, by classifying only a subset of segments and filling the gaps through interpolation.

By addressing these limitations, we aim to develop a more robust and efficient framework for automated bouldering video analysis, ultimately providing coaches with valuable insights to enhance athlete performance.

8. Conclusion

Research Contributions

This work demonstrates the feasibility of automated bouldering video segmentation using modern deep learning approaches. Our experiments revealed that segment-based models, particularly the X3D family and the R3D, outperform frame-based approaches for this task, achieving accuracies of up to 86.80%. We established that the choice of pre-training dataset significantly impacts performance, as evidenced by the stark difference between S3D models trained on HowTo100M versus Kinetics. Notably, model complexity did not necessarily correlate with performance, with lighter models often matching or exceeding their larger counterparts.

Future research should focus on improving real-time capabilities, investigating alternative loss functions, and exploring transformer-based architectures for video understanding. Semi-supervised approaches leveraging unlabeled climbing footage could address the data scarcity challenge while maintaining evaluation integrity.

Project Reflection

This project provided valuable experience in end-to-end data science development. Building a computer vision system from scratch presented numerous challenges, from data collection and annotation to model selection and evaluation. The need to balance performance requirements against computational constraints mirrors real-world AI deployment scenarios.

The most significant challenge was navigating the vast landscape of video understanding models without extensive training resources. This required careful consideration of transfer learning approaches and creative solutions for data augmentation.

This work served as a formative experience in independent research, requiring comprehensive exploration of the computer vision literature and adaptation of techniques across domains. It demanded proficiency across the entire machine learning pipeline—from data preparation to deployment considerations—providing practical skills beyond what typical structured projects offer. This holistic approach to solving a complex visual understanding problem has reinforced both technical capabilities and research methodology fundamentals.

Acknowledgment

I would like to express my sincere gratitude to my supervisor, Jeremie Boulanger, for his invaluable guidance, support, and encouragement throughout this project.

References

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. ViViT: A Video Vision Transformer. *arXiv e-prints*, art. arXiv:2103.15691, March 2021. doi: 10.48550/arXiv.2103.15691.

Bertasius, G., Wang, H., and Torresani, L. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. *arXiv preprint arXiv:1705.07750*, 2018. URL https://arxiv.org/abs/1705.07750.

Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018. URL https://arxiv.org/abs/1808.01340.

Carreira, J., Noland, E., Hillier, C., and Zisserman, A. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

Ding, G., Sener, F., and Yao, A. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Ding, L. and Xu, C. Tricornet: A hybrid temporal convolutional and recurrent network for video action segmentation. 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv e-prints*, art. arXiv:2010.11929, October 2020. doi: 10.48550/arXiv.2010.11929.

Fan, H., Li, Y., Xiong, B., Lo, W.-Y., and Feichtenhofer, C. Pyslowfast. https://github.com/facebookresearch/slowfast, 2020.

Feichtenhofer, C. X3d: Expanding architectures for efficient video recognition. *arXiv preprint arXiv:2004.04730*, 2020. URL https://arxiv.org/abs/2004.04730.

Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzyńska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., Hoppe, F., Thurau, C., Bax, I., and Memisevic, R. The 'something something' video database for learning and evaluating visual common sense. *arXiv preprint arXiv:1706.04261*, 2017. URL https://arxiv.org/abs/1706.04261.

Hara, K., Kataoka, H., and Satoh, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? *arXiv preprint*, arXiv:1711.09577, 2017.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Kuehne, H., Arslan, A. B., and Serre, T. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*, 2014.

Labs, V. Video segmentation guide. 2025. URL v7labs.com/blog/video-segmentation-guide.

Laptev, I. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, 2005. doi: 10.1007/s11263-005-1173-4. URL https://doi.org/10.1007/s11263-005-1173-4.

Li, Y., Fan, H., Yan, Z., Xiong, B., Mangalam, K., Malik, J., and Feichtenhofer, C. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021. URL https://arxiv.org/abs/2104.11227.

McKenna, S. and Stein, S. 50 salads. https://doi.org/10.15132/10000120, 2012. Creator.

Miech, A., Zhukov, D., Alayrac, J.-B., Tapaswi, M., Laptev, I., and Sivic, J. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *ICCV*, 2019.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.-Y., Xu, H., Sharma, V., Li, S.-W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. Dinov2: Learning robust visual features without supervision, 2023.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv e-prints*, art. arXiv:1506.02640, June 2015. doi: 10.48550/arXiv.1506.02640.

Scheidegger, F., Cavigelli, L., Schaffner, M., Malossi, A. C. I., Bekas, C., and Benini, L. Impact of temporal subsampling on accuracy and performance in practical video classification. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 996–1000, 2017. doi: 10.23919/EUSIPCO.2017.8081357.

Sener, F., Chatterjee, D., Shelepov, D., He, K., Singhania, D., Wang, R., and Yao, A. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *CVPR 2022*.

Wang, H. and Schmid, C. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2921–2928. IEEE, 2013. doi: 10.1109/ICCV.2013.365. URL https://doi.org/10.1109/ICCV.2013.365.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. *arXiv e-prints*, art. arXiv:1608.00859, August 2016. doi: 10.48550/arXiv.1608.00859.

Wang, M., Xing, J., and Liu, Y. ActionCLIP: A New Paradigm for Video Action Recognition. *arXiv e-prints*, art. arXiv:2109.08472, September 2021. doi: 10.48550/arXiv.2109.08472.

Xie, S., Sun, C., Huang, J., Tu, Z., and Murphy, K. Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification. *arXiv e-prints*, art. arXiv:1712.04851, December 2017. doi: 10.48550/arXiv.1712.04851.

Zhu, Y., Li, X., Liu, C., Zolfaghari, M., Xiong, Y., Wu, C., Zhang, Z., Tighe, J., Manmatha, R., and Li, M. A Comprehensive Study of Deep Video Action Recognition. *arXiv e-prints*, art. arXiv:2012.06567, December 2020. doi: 10.48550/arXiv.2012.06567.

A. Definitions

A.1. Self-Supervision

Self-supervision is a learning paradigm in which a model is trained using automatically generated labels rather than human-annotated data. This technique is commonly used to leverage large amounts of unlabeled data by designing tasks that encourage the model to learn meaningful representations.

B. Backbone Networks

YOLO

YOLO, developed by Ultralytics, is a CNN-based model capable of detecting 17 key points of a person in a frame. It is widely used for pose estimation and action recognition.

R3D

ResNet-3D (R3D) is a 3D CNN architecture proposed in (Hara et al., 2017) for video classification. It extends the original ResNet architecture to process spatio-temporal data.

I3D

Inflated 3D (I3D) is a 3D CNN network proposed in (Carreira & Zisserman, 2018) for video classification. It extends the Inception architecture to 3D by inflating 2D convolutional filters into 3D filters. The model initializes its weights using pretrained 2D CNN models to improve convergence and performance. I3D employs a two-stream approach that processes optical flow and RGB frames separately through dedicated 3D CNNs, merging the outputs for classification.

DINO

DINO, introduced in (Oquab et al., 2023), is a transformer-based vision model pre-trained on a large image dataset. It leverages a class token, whose state in the final layer serves as the feature representation. DINO demonstrates strong generalization capabilities on various visual tasks.

CLIP

CLIP is a vision transformer model designed for zero-shot learning. Trained on extensive image-text pairs, CLIP effectively classifies images based on text descriptions and vice versa.

X3D

X3D, proposed in (Feichtenhofer, 2020), is an efficient and lightweight family of 3D CNNs designed for video understanding. By expanding a 2D CNN across multiple axes, X3D reduces parameters and computations while maintaining strong performance.

S3D

S3D, described in (Xie et al., 2017), is a 3D CNN architecture designed for video-text embeddings. The model replaces certain 3D convolution steps with 2D convolutions equipped with temporal kernels. This modification improves efficiency and facilitates training on large-scale instructional video datasets.

SlowFast

SlowFast, introduced in (Fan et al., 2020), is a two-stream network that processes video at two different frame rates. The slow path captures spatial semantics and high-level features, while the fast path captures rapid motion dynamics at a higher temporal resolution.

ViViT

ViViT, proposed in (Arnab et al., 2021), extends the vision transformer architecture to video data. Each video frame is processed by a vision transformer, extracting CLS tokens that are then combined using a second transformer to capture temporal relationships and produce a final video embedding.

C. Figures and Tables

Backbone	Type	#Parameters	#Frames	Features Shape
yolo	By Frame	2.9M	8	(8, 34)
dino	By Frame	22.1M	8	(8, 384)
clip	By Frame	151.3M	8	(8, 512)
r3d	By Segment	31.6M	8	(2048, 1)
i3d	By Segment	12.7M	16	(1024, 1)
x3d-xs	By Segment	3.0M	4	(2048, 1)
x3d-s	By Segment	3.0M	16	(2048, 1)
x3d-m	By Segment	3.0M	16	(2048, 1)
x3d-l	By Segment	5.3M	16	(2048, 1)
s3d-k	By Segment	7.9M	16	(1024, 1)
s3d-h	By Segment	9.7M	16	(1024, 1)
slowfast	By Segment	33.6M	32	(2304, 1)
vivit	By Segment	88.6M	32	(768, 1)

Table 4: Detailed Descriptions of Feature Extractors.

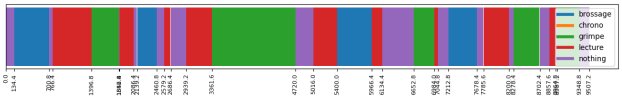

Figure 11: Example of video segmentation generated using the TAS helper Python package.