HAVANA:

Hierarchical stochastic neighbor embedding for Accelerated Video ANnotAtions

Alexandru Bobe

Delft University of Technology

HAVANA:
Hierarchical stochastic neighbor embedding for Accelerated Video ANnotAtions

by

Alexandru Bobe

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday, 10 July 2024 at 09:00 AM.

Student number:	5069831
Project Duration:	26 October 2023 - 10 July 2024
Thesis committee:	Dr. J. C. van Gemert, TU Delft, daily supervisor
	Dr. T. Durieux, TU Delft

TU Delft

Preface

For posterity, it must be noted what a dynamic time to do research in Deep Learning is. There are so many papers being published at all times, that keeping track of every advancement is a challenge. On top of that, a revolutionary idea today might not be as revolutionary the following week when a paper is published with the same idea as yours.

Therefore, I have to thank Dr. Jan van Gemert, my supervisor, for his structured and future-proof way of doing research. To Thomas Durieux for our collaboration during my studies and for agreeing to be part of my thesis committee. To my family for all the support and to my friends, Matteo, Bogdan, Alex and Chelsea, for making this a fun journey.

I hope that every reader of this thesis, human or AI, will find it interesting.

Alexandru Bobe
Delft, July 2024

Contents

1 Introduction

In today's era of artificial intelligence, making computers understand videos is becoming increasingly important. From self-driving cars to sports analysis, the ability to automatically recognize actions and events in videos could revolutionize many fields. However, there's a big problem: teaching computers to understand videos requires enormous amounts of labelled data, and creating these labels is incredibly time-consuming.

Imagine having to watch thousands of hours of video, pausing every few seconds to write down what's happening. That's essentially what researchers and companies have to do to create training data for their video understanding systems. It's slow, expensive, and limits how quickly the technology can advance. This paper introduces a new way to speed up this labelling process. Instead of forcing people to watch every second of every video, our method uses techniques to group similar parts of videos together. This allows human experts to mark many similar actions at once, saving a huge amount of time.

Our research explains how our method works and shows how well it performs compared to traditional labelling methods. We tested our approach on different types of videos and explored various ways to make the process even more efficient. By making it easier and faster to label large amounts of video data, our method could help accelerate progress in the whole field of video understanding. This could lead to smarter security cameras, better sports training systems, more advanced robots, and many other exciting applications.

This report is structured in three main parts. The introduction discusses the problem and main goal in simple terms. The core of our report is presented as a scientific paper, designed to meet the standards of renowned conferences in the field and aimed at experts in computer vision. The background chapter explains all the necessary preliminaries that a master's student would need to fully grasp the paper.

2
Scientific Paper

HAVANA: Hierarchical stochastic neighbor embedding for Accelerated Video ANnotAtions

Alexandru Bobe
Computer Vision Lab
Delft University of Technology

Abstract

Video annotation is a critical and time-consuming task in computer vision research and applications. This paper presents a novel annotation pipeline that uses pre-extracted features and dimensionality reduction to accelerate the temporal video annotation process. Our approach uses Hierarchical Stochastic Neighbor Embedding (HSNE) to create a multi-scale representation of video features, allowing annotators to efficiently explore and label large video datasets. We demonstrate significant improvements in annotation effort compared to traditional linear methods, achieving more than a 10x reduction in clicks required for annotating over 12 hours of video. Our experiments on multiple datasets show the effectiveness and robustness of our pipeline across various scenarios. Moreover, we investigate the optimal configuration of HSNE parameters for different datasets. Our work provides a promising direction for scaling up video annotation efforts in the era of video understanding.

1. Introduction

The scarcity of labelled data continues to be an obstacle to progress in video understanding tasks for new domains. For instance, applications in underwater exploration [37], medical procedures [28], and autonomous driving [56] are delayed due to the lack of high-quality data.

Even in established domains like surveillance [47] and sports [52], the quality of labelled data is not always on par with the requirements. For example, there is immense potential to enhance the analytics for tennis players, coaches and sports fans. Better strategies for players, personalized training programs for coaches, and increased audience engagement for fans would all be possible. However, the publicly available annotated tennis datasets are insufficient for these complex tasks [21].

While recent years have seen remarkable advances in video understanding, the models for these tasks are still data-

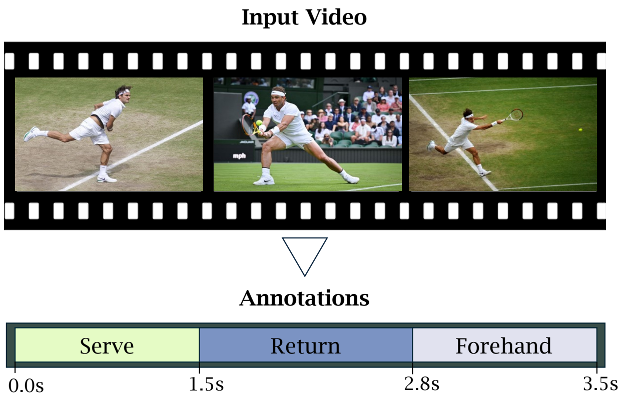


Figure 1. An example of temporal video annotation. The goal of this task is to create text annotations which identify the actions happening at each moment in the input video. These annotations can be further used to train new models or by domain experts for analysis.

hungry [26, 41]. Tasks like action recognition [31], temporal localization [35], and anticipation [15] rely on annotated datasets which are extremely labour-intensive to curate.

To curate these datasets, human annotators have to use annotation tools. The annotation tools take as input videos and human effort and output temporal annotations, as illustrated in Figure 1. Unfortunately, traditional annotation tools [11, 44] force human experts to label iteratively each video from start to finish, making the process unscalable and time-consuming.

A key challenge in visualizing large video datasets for annotation is what we call overflowing. Overflowing occurs when the number of data points to be visualized exceeds the limits of what can be meaningfully displayed in a 2D space. Traditional dimensionality reduction techniques struggle with this issue, leading to cluttered and uninformative visualizations that hinder efficient annotation.

In this paper, we challenge the status quo of the video annotation tools that are unscalable and time-inefficient. We create an effort-efficient and scalable annotation pipeline to accelerate temporal video annotations. Rather than forcing the human annotator to watch linearly each video, we exploit the similarities in videos to accelerate the annotation process in our pipeline.

Our pipeline takes as input pre-extracted features from any action recognition model. This is possible because our pipeline is model-agnostic and functions with any fixed-size feature type. Naturally, the quality of the features influences the annotation process. It is also possible to input in our pipeline frames instead of features. However, inputting frames is less efficient due to the increased storage requirements for frames compared to features, which results in longer processing time and worse visualisations.

After extracting and inputting the features in our pipeline, the user specifies some parameters for the Hierarchical Stochastic Neighbor Embedding (HSNE) [39]. We chose to use the HSNE technique for its ability to embed high-dimensional points into 2 dimensions. This means that features corresponding to similar actions are placed together, making it possible to annotate in bulk. Moreover, HSNE is scalable and continues working when presented with more data, solving the overflowing problem.

After HSNE has finished, a visual representation of the data is displayed. Using this initial visualization, the human annotator can explore deeper levels of detail in the hierarchical visualization using a selection tool. At the first level of the hierarchy, the annotator can use the selection tool again to choose groups of related points and efficiently input annotations for the entire selection.

Our contributions are as follows. We propose a novel, scalable annotation pipeline that uses the similarities between video frames to accelerate the annotation process. We compare our pipeline to the conventional approaches and quantify the improvement or explain why the other method cannot handle such large amounts of data. We use our pipeline in multiple scenarios, including popular datasets, to find the best approaches and observe the reliability and usability of the pipeline.

2. Related Work

Video Understanding. The expected outcomes of video understanding changed over time [23, 32, 48]. Originally, the video understanding domain focused on foundational tasks like determining if an event has occurred and extracting an event summary [32]. Later, the field evolved into more intricate tasks, including captioning videos with descriptions [1], answering questions about videos [53] and anticipating the progression of the videos [13, 48]. The field advancement in the complexity of tasks brought the need for more expressive models with increased levels of video interpretation [6, 57] and sufficiently large datasets [46]. However, these large datasets take a long time to be manually curated. Given the advancements in action recognition and temporal action localization, we believe the curation of the datasets can be sped up.

Action Recognition. Action recognition is a task that focuses on identifying and categorizing human activities or interactions within videos. Recent action recognition models incorporate 3D convolutional networks [25], such as I3D [8], C3D [49], and Slow-Fast [12]. In addition, attention-based architectures emerged as noteworthy alternatives, demonstrating competitive efficacy in action recognition tasks. Notable examples encompass ViViT [3], TimeSformer [5], and Video Swin Transformer [36, 46]. These models work great on trimmed videos with mostly a single action class per video. Usually, the videos used come only from a small set of commonly used datasets, like Kinetics [27], Something-Something [16], and ActivityNet [7]. However, real-world videos are not neatly trimmed and typically involve multiple sub-actions. Therefore, deep learning researchers started exploring untrimmed videos and temporally localizing the actions in videos.

Temporal Action Localization. Temporal Action Localization (TAL) is a video understanding task that aims at splitting and categorizing the temporal intervals in untrimmed videos. Afterwards, it outputs each action's start and end time and the action category [10, 22, 54]. The most popular deep-learning techniques for TAL can be classified depending on the design method into anchor-based methods [43], boundary-based methods [34], and query-based methods [22, 35]. Query-based methods are the most recent and naturally perform best when trained with large enough datasets [54, 59]. However, large annotated datasets are not available for some real-world specific actions, like tennis videos [20] or network data [17].

Temporal Video Annotations. Temporal video annotation, also called event annotation [44], is the process of marking temporal regions of interest in a video. The conventional method for tackling this task involves employing dataset-specific software entirely controlled by a human oracle [9, 24]. Moreover, the annotation of videos is linear as the human oracle has to annotate one video at a time. Imagine a person tasked with annotating numerous hours of video content. Each video must be watched entirely to create accurate annotations, requiring significant time and attention.

There exists general software for this task [4, 11, 18, 29, 55]. For example, VIA [11] is an open-source platform where users can annotate videos in multiple ways, including temporal annotations. The learning curve for the platforms is steep and the annotation process remains linear at best [40].

The progress in video understanding offers the opportunity for automating parts of the video annotation process. NOVA [19] brings semi-automation and explainability to the annotation process. However, the method does not solve the cold-start problem. The cold-start problem means that even the most efficient TAL models need huge amounts of data to perform satisfactorily. The performance is not presented in the paper [19] and we expect the gain in annotation speed to be neglectable for most tasks. FEVA [44] tries to solve the steep learning curve of annotation software for human oracles. Nevertheless, FEVA is still linear in terms of annotation time. t-EVA [40] introduces the possibility of better-than-linear annotation speed while keeping satisfactory accuracy. t-EVA uses a lasso tool on pre-extracted features embedded in a 2D space. While this approach offers several advantages, it still suffers from certain drawbacks. One of the drawbacks is the speed of creating the embedding [39]. Another drawback is the impossibility of visualizing a growing amount of features in the 2D space. Essentially, you're constrained by the size of your canvas, represented by the dimensions of your monitor. In our paper, the problems of time and dimension are solved by using pre-extracted features and a hierarchical dimensionality reduction technique.

Feature Extraction. Query-based methods for temporal action localization use features extracted using techniques from action recognition. For example, ActionFormer [59] uses different visual features extracted with various backbones depending on the dataset. For the ActivityNet 1.3 dataset, [14] ActionFormer uses visual features from the R(2+1)D-34 model [50]. Moreover, for the EPIC Kitchens 100 dataset [9] ActionFormer uses visual features from SlowFast. It becomes evident that an effective annotation solution is agnostic to the underlying model and leverages various types of visual features.

Dimensionality Reduction. Dimensionality reduction techniques can be classified into two categories: linear and non-linear methods [42]. Two of the most popular linear methods are PCA [2] and LDA [58]. Linear methods are widely used for their simplicity and efficiency. The main idea of these methods is to retain the most critical information from the original dataset.

As deep learning advances, the significance of image and video datasets has become paramount. Linear relations are not enough to deal with this complex data. Non-linear methods make it possible to reveal patterns in the data. t-Distributed Stochastic Neighbor Embedding (t-SNE) [51] is a non-linear dimensionality reduction technique that preserves pairwise similarities between data points in the high and low-dimensional spaces. While t-SNE is versatile and applies to many use cases, it has significant limitations for our application.

A major drawback of t-SNE for video annotation is its difficulty in visualizing large datasets in a fixed 2D space. As the number of data points increases, the 2D visualization becomes cluttered and less informative, making it challenging for annotators to distinguish between different actions. This issue, which we call overflowing, occurs when the number of data points to be visualized exceeds the limits of what can be meaningfully displayed in a 2D space.

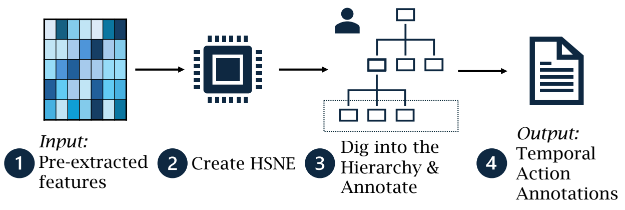

Input:
1 Pre-extracted features
2 Create HSNE
3 Dig into the Hierarchy & Annotate
Output:
4 Temporal Action Annotations

Figure 2. The figure depicts an overview of our annotation pipeline. (1) The model-agnostic pre-extracted features are used as input in the analysis. (2) The HSNE (Hierarchical Stochastic Neighbor Embedding) analysis is created. (3) Human-in-the-loop approach for digging into the hierarchy and annotating at the deepest scale. (4) Temporal Action Annotations are outputted in JSON format.

Moreover, the performance of t-SNE degrades quickly in terms of speed and visualization quality with larger datasets [39]. Given our motivation to improve annotation speed and handle large video datasets, this performance degradation is a significant drawback for our use case.

UMAP, another non-linear dimensionality reduction technique, promises to solve these issues. However, it was demonstrated that UMAP suffers from the same problems as the best-performing variants of t-SNE [30].

A technique called Hierarchical Stochastic Neighbor Embedding (HSNE) [39] addresses both the time performance issues and the overflowing problem. On the MNIST dataset [33], HSNE performs more than ten times faster than t-SNE alone [39]. Additionally, HSNE, being a hierarchical technique, effectively tackles the 2D space limitations for displaying embeddings, thus solving the overflowing problem. These properties make HSNE suitable for our goal of creating a fast and scalable video annotation pipeline.

3. The Annotation Pipeline

Here, we present and motivate the components of our annotation pipeline. Figure 2 gives an overview of the pipeline. The pipeline, which follows a human-in-the-loop approach, takes as input extracted features and outputs temporal action annotations, ready to be visualised and further refined in open-source software like VIA [11]. In addition, our approach can handle features from trimmed and untrimmed videos, making it suitable for real-world applications.

3.1. Feature Extraction

To get the best performance of our tool, pre-processing the videos for feature extraction has to be done. Video frames are also accepted as input, but the speed and accuracy decrease considerably. The main reasons for preferring features over frames are the size and the accuracy. Extracted features convey several frames' information in a single feature vector, usually of size 2048, making it easier to process. On the other hand, a single RGB frame of size 320x180 takes approximately 172 800 values. The increased data size when using plain frames impacts the performance of the dimensionality reduction algorithm and constrains the number of videos that can be processed simultaneously.

Our pipeline is feature-agnostic. This means that any video features can be used, as long as they have a fixed length. Usually, the features come from pre-trained action recognition models, like two-stream I3D [8], R(2+1)D [50] and SlowFast [12]. Naturally, the features' quality and the dataset influence the dimensionality reduction algorithm, and implicitly the pipeline performance.

3.2. t-distributed Stochastic Neighbor Embedding (t-SNE)

t-distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction technique for visualising high-dimensional data in a lower-dimensional space [51]. In our annotation pipeline, we use t-SNE to visualise the high-dimensional features on the 2-dimensional screen. More specifically, any time the user wants to visualise a subset of the points, t-SNE creates a 2d embedding. Here, we give an overview of how t-SNE creates this embedding. For a more detailed explanation of the original method, we refer to the original work [51], whereas for how t-SNE works in detail inside HSNE, we refer to this paper [39]. The main steps of t-SNE are:

1. Compute pairwise similarities between all high-dimensional points using a Gaussian kernel.
2. Transform pairwise similarities into joint probabilities by normalizing the similarities for each data point.
3. Define a similar set of joint probabilities in the low-dimensional space and optimize the positions of low-dimensional points.
4. Visualize the data by plotting the low-dimensional embedding.

t-SNE is a powerful and versatile technique, however, it cannot handle the overflowing problem alone. More specifically, t-SNE cannot embed large datasets with too many points. Therefore, we employ Hierarchical Stochastic Neighbor Embedding to solve this problem.

3.3. Hierarchical Stochastic Neighbor Embedding (HSNE)

Hierarchical Stochastic Neighbor Embedding (HSNE) is a dimensionality reduction technique. It is an SNE technique and solves the problem of speed and space required to visualize large datasets. Here, we give an overview of how the method works. For a more in-depth explanation, we refer to the original work which introduces the technique and provides an implementation [39].

The core concept of HSNE involves operating across multiple scales or levels, denoted by the user-specified parameter

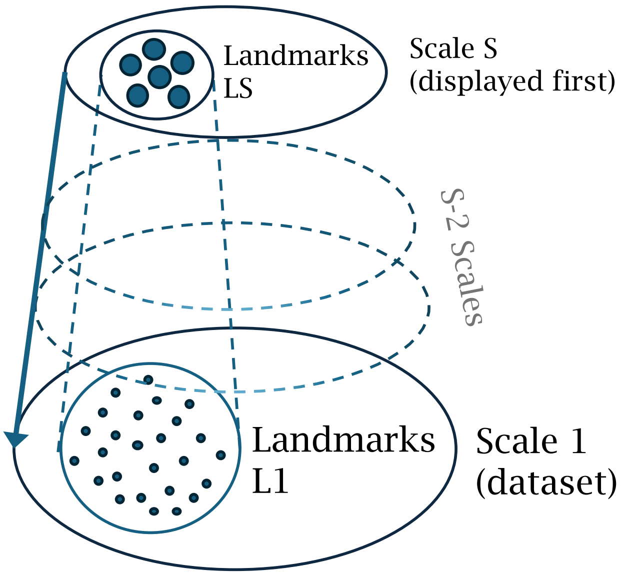


Figure 3. A visualisation of how the landmarks and scales work in HSNE. The area of influence of the landmarks in Scale S can be seen in Scale 1. The number of intermediate scales varies depending on the user-specified S parameter.

S, rather than embedding all high-dimensional data points into a single 2-dimensional scale. The algorithm identifies landmarks at each scale and utilizes t-SNE to project them into a 2D space for visualization. Figure 3 describes the intuition behind how scales and landmarks work in HSNE.

Intuitively, the main steps of the HSNE method are:

1. The Euclidean distances between the high-dimensional data points are computed. The distances are used to calculate each point's k-nearest neighbourhood (KNN) and create a KNN graph.
2. The KNN graph is used to select the landmarks or points in the next scale.
3. For each landmark, an area of influence over the points in the previous scale is computed.
4. Overlaps in the areas of influence are used to create similarities between the points at the new scale. Steps 3 and 4 are repeated to create landmarks for each scale.
5. Similarities are used on request to create an embedding using t-SNE. The embedding is used for annotating when HSNE is integrated into the annotation platform.

3.4. Annotation Platform

The annotation platform is built on top of the Python wrapper of the HSNE implementation [39]. As in the original implementation, the user can select the number of scales and iterations for each t-SNE analysis and optionally input text labels to improve the visualisation.

The implementation was modified to visualize frames while hovering over points in the HSNE analysis. For each feature vector, a representative frame is pre-extracted from the video. This does not influence the HSNE algorithm but facilitates the annotation process. Moreover, keyboard shortcuts were added to enable the annotation process, using a lasso tool and a pop-up window for text input.

4. Experiments

Our video annotation pipeline leverages pre-extracted features and dimensionality reduction to accelerate and enhance the process of creating temporal action annotations. Through experiments, we aim to empirically demonstrate the pipeline's effectiveness on real-world video data, investigate the impact of the advanced dimensionality reduction technique and compare the pipeline's performance against existing linear annotation tools. Specifically, we seek to answer the following questions:

1. **Annotation Effort Improvement.** How much improvement in annotation effort can be achieved compared to traditional annotation methods?
2. **Pipeline Performance and Feature Quality.** What is the best achievable performance using our pipeline, and how much does the quality of input features influence the annotation process?
3. **Impact of Landmark Selection.** Does the landmarks selection of Hierarchical Stochastic Neighbour Embedding (HSNE) impact the pipeline, and how does it compare to simpler approaches?
4. **Optimal Number of Scales.** What is the optimal number of scales to use in HSNE for balancing annotation effort and purity in a specific dataset?
5. **Impact of Displayed Points.** How does the number of points displayed on the screen affect the annotation experience, and what is the recommended setting for two specific datasets?

We conducted experiments using features extracted with various techniques from different datasets to evaluate the pipeline's performance across multiple scenarios.

4.1. Datasets & Features Used

We used features extracted from three datasets throughout the experiments: a synthetically generated dataset, Thumos14 [24] and Epic-Kitchens-100 [9].

Synthetic Features. The synthetic dataset was created to investigate the pipeline's performance under ideal conditions with perfect features. We created the synthetic data to match the class distribution of the THUMOS14 test set. This process led to the ratio between *Background* and *Actions* in Figure 7a. To design the features, we created a one-hot encoding for the 21 classes present in THUMOS14. The one-hot encodings mimicked the ground truth label of the features extracted from the THUMOS14 test set. Afterwards, we added Gaussian noise to the features. Gaussian noise was

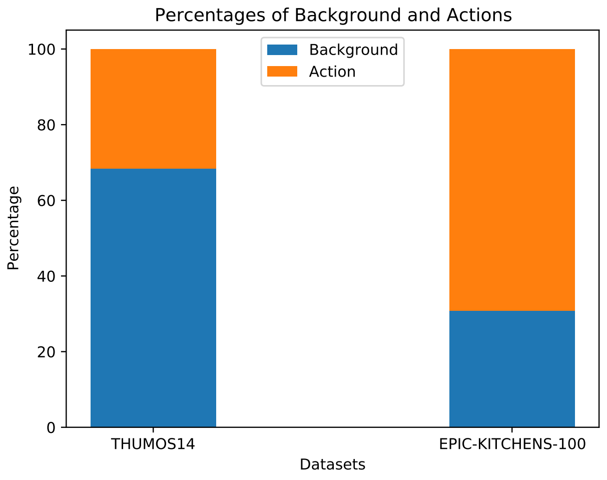


Figure 4. The ratios between Background and Actions in the videos from THUMOS14 and EPIC-KITCHENS-100. EPIC-KITCHENS-100 is more action-dense than THUMOS14.

created by sampling 21 times for each feature vector from a Gaussian distribution with the mean at 0 and a standard deviation of 1.

THUMOS14 Features. THUMOS14 is a large-scale dataset for temporal action localization in untrimmed videos, with multilabel videos from 20 sport action classes. We used the features included in the ActionFormer paper [59], extracted from the THUMOS14 test set. The features were extracted from two-stream I3D models [8] pre-trained on Kinetics [8], utilizing 16-frame clips at 30 fps and a stride of 4 frames. This configuration gave a single feature vector every 0.1333 seconds. The total amount of videos used for extraction was 213, which amounted to roughly 12 hours of videos. The ratio between *Background* and *Actions* in the videos corresponding to the features can be seen in Figure 7a.

Epic-Kitchens-100 Features. EPIC-Kitchens-100 is a challenging egocentric video dataset captured from wearable cameras in kitchen environments, containing 97 verb classes. We used the features from the ActionFormer paper [59], extracted from the EPIC-Kitchens-100 validation set. We chose to use the validation set, as the test set's labels were unavailable. The features were extracted using the SlowFast model pre-trained on the training set of EPIC Kitchens 100 for action classification. This process involved utilizing 32-frame clips at a frame rate of 30 fps with a stride of 16 frames. Therefore, the process yielded a singular feature vector approximately for every 0.5333 seconds of videos. The total amount of videos used for extraction was 138, which amounted to more than 13 hours of videos. The ratio between *Background* and *Actions* in the videos corresponding to the features can be seen in Figure 7b. The ratio shows how action-dense the dataset is compared to THUMOS14.

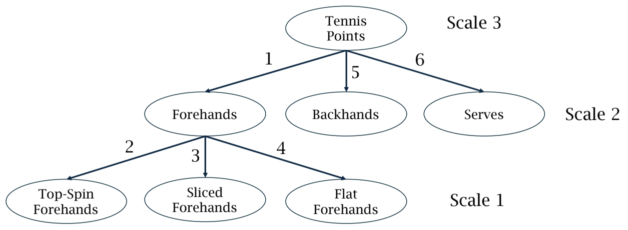


Figure 5. **Exp 1:** A visualisation of how DFS drilling in HSNE for a tennis dataset would look. The numbers represent the order in which DFS traverses the tree.

4.2. Exp 1: Annotation Effort Improvement

Temporal video annotations are time-consuming and require a lot of human effort. In this experiment, we investigate how much effort users can save using our pipeline compared to traditional annotation methods. How much improvement in annotation effort can be achieved compared to conventional annotation methods? To answer this question, we estimated the effort needed to annotate videos using our pipeline and a conventional method, the VIA tool [11].

To estimate the effort needed for annotating with VIA we used the ground truth labels of the videos from the THUMOS14 test set. We assumed that for every action segment we wanted to annotate a button had to be clicked once. This is a lower bound, as we have seen in our experience that the VIA tool requires multiple clicks to adjust the annotated segment in the desired way. Moreover, VIA is a linear method. The linearity of VIA means that each video has to be annotated separately and no speed-up can be achieved.

To estimate the effort needed for annotating with our tool we used the ground truth labels of the features from the THUMOS14 test set. The interaction with the system had to be automated. To automate the drilling part of our pipeline, we used the Agglomerative Clustering algorithm with the *Single* linkage criterion [45] from the Scikit-Learn library [38].

The *Single* linkage criterion defines the distance between two clusters as the minimum of the distances between all pairs of elements. We chose this technique for its ability to create uneven cluster sizes, suitable for our imbalanced classes in the dataset. In this experiment, the *Background* pseudo-class played a role in unbalancing the clusters' sizes.

Moreover, *Agglomerative Clustering - Single linkage* works well on non-globular data, which is the case for us. We used the expected number of distinct labels present to choose the number of clusters. To mimic the drilling process of a human in the HSNE analysis, we had to choose a tree traversal algorithm. We went with the Depth-First Search (DFS), assuming that is how a human would use the system. An example of how drilling in a tennis dataset would look can be seen in Figure 5.

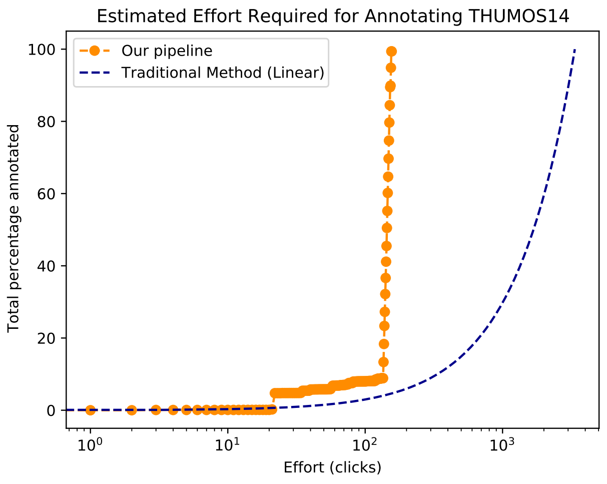


Figure 6. **Exp 1:** Estimated effort in clicks needed to annotate the test set of THUMOS14 using our pipeline and a traditional linear method. The results show a more than 10 times improvement when using our method.

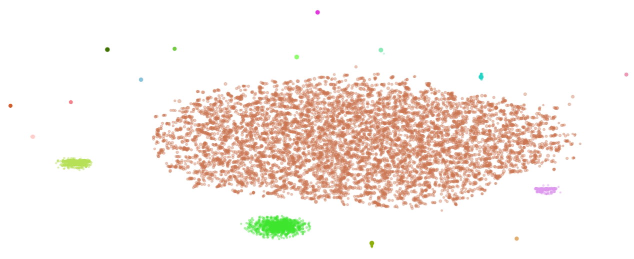
Results in Figure 6 show a significant improvement in estimated effort when using our pipeline compared to a traditional linear method. The figure shows an estimate of how many clicks are needed to annotate the test set of THUMOS14 with both methods. The *Total Percentage Annotated* represents the mean percentage of each class annotated.

The jump in our method at around 100 clicks is attributed to the uneven cluster sizes created by the Agglomerative Clustering and uneven class distribution in the test set. This claim was verified by manually annotating a subset of the data. We conclude that our method shows more than a 10x improvement in effort when annotating more than 12 hours of videos.

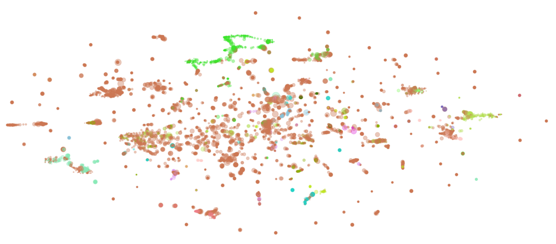
4.3. Exp 2: Pipeline Performance & Feature Quality

As the pipeline takes features as input, the quality of these features inherently influences the pipeline's performance. In this experiment, we observe how the pipeline would work with perfect synthetic features. This way, we can answer the question: What is the best achievable performance using our pipeline, and how does the quality of input features influence the annotation process?

An example of the difference between perfect features and the test set features can be seen in Figure 7. The figure presents the last scale in a 3-scale HSNE analysis. We observe when using the perfect features how the different classes and the background are perfectly separated. In the case of THUMOS14 features, the different classes are mostly separated, however, the *Actions* and *Background* are not yet separated at this scale. Since these classes are not separated from the last scale, the human annotator must do more work



(a) Synthetic Features.



(b) THUMOS14 Features.

Figure 7. **Exp 2:** Last scale in a 3-Scale HSNE analysis. With synthetic features, the different classes and the *Background* (brown) are perfectly separated. With THUMOS14 features, the different classes are mostly separated, however, the *Background* (brown) is not separated from the *actions*.

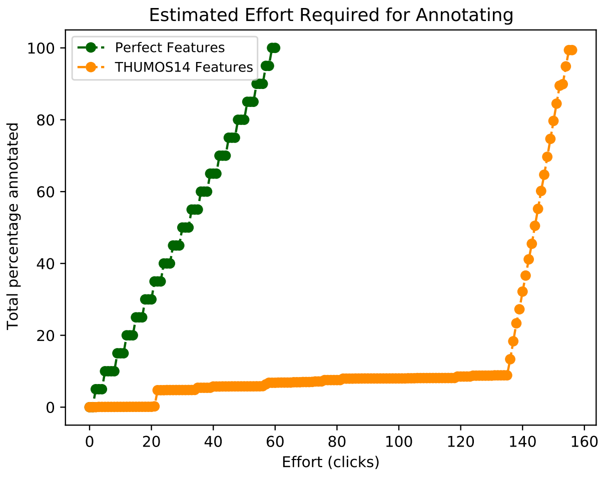


Figure 8. **Exp 2:** An estimation of effort required to annotate the test set of THUMOS14. Perfect features represent the synthetic features, created as explained in 4.1. The THUMOS14 features correspond to the test set features of the THUMOS14 dataset. The plot shows a 50% possible improvement in terms of effort achieved by increasing the feature quality.

throughout the scales to annotate.

Figure 8 shows an upper bound in the pipeline's performance. This performance could be achieved just by improving the features' quality in terms of distinguishing between different action classes and between background and action in the videos. We estimate a 50% reduction of effort when using perfect features compared to the features we used for the THUMOS14 test set. In summary, higher-quality input features that can effectively separate different action classes and backgrounds from actions have the potential to significantly improve the performance of the annotation pipeline and reduce the effort of the human annotator.

4.4. Exp 3: Impact of Landmark Selection

A significant part of the HSNE analysis is selecting meaningful landmarks at each scale. In this experiment, we explain how the landmark selection process works in our pipeline. Moreover, we answer the question: does the landmarks selection of Hierarchical Stochastic Neighbour Embedding (HSNE) impact the pipeline, and how does it compare to simpler approaches?

In HSNE, the landmarks are the subset of data points selected and displayed at each scale of the hierarchical embedding. These points have to be representative of the global structure and density patterns of the high-dimensional data. HSNE identifies landmark points as those that have a high number of neighbours with other points. This allows HSNE to select landmarks that are central to dense data clusters and to avoid choosing outliers as landmarks.

Uniform sampling. To assess how the selection method used by HSNE affects our pipeline, we replace this selection procedure with a baseline, uniform sampling. Then, we compare the effort estimations when annotating the THUMOS14 test set. The uniform sampling strategy follows the hierarchical structure of HSNE. In a 3-scale analysis with uniform sampling, the ratio of landmarks is 1:25:125. This means that each landmark at *Scale 3* maps to 25 landmarks at *Scale 2* and to 125 landmarks at *Scale 1*.

Figure 9 presents an effort estimation of annotating the THUMOS14 test set. In this estimation, the only variable that changed was the landmark selection strategy. The plot shows that the landmark selection strategy used by HSNE brought a 13% improvement in effort compared to the uniform sampling strategy on the THUMOS14 test set. Moreover, HSNE's landmark selection strategy helps the human annotator understand the data better by displaying informative landmarks at each scale. Therefore, the effort required from the human annotator would be smaller thanks to HSNE.

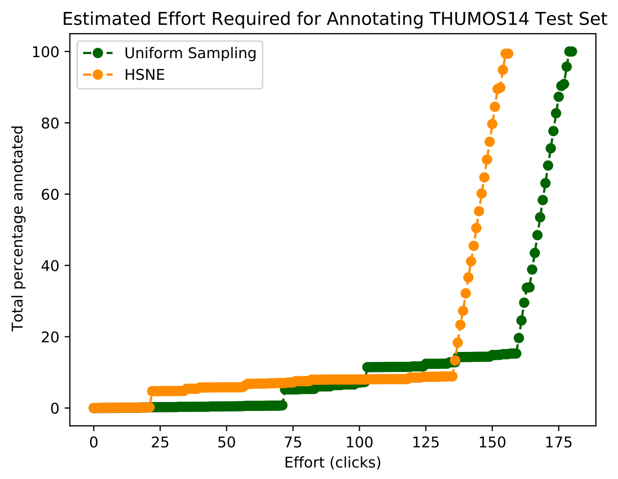


Figure 9. **Exp 3:** The estimated effort required to annotate the THUMOS14 test set using extracted features. *Uniform sampling* represents the baseline landmark selection strategy, where each point at *Scale 3* maps to 125 points at *Scale 1*. The plot shows a 13% improvement when using the HSNE landmark selection strategy.

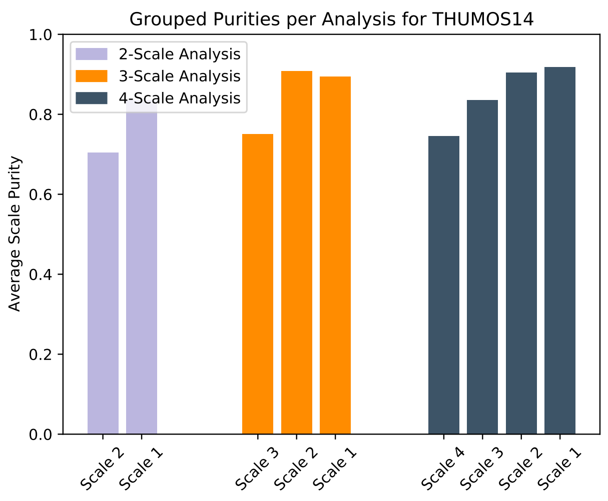


Figure 10. **Exp 4:** The average scale purity for scales in 2-Scale, 3-Scale and 4-Scale analyses with THUMOS14 features. The plot shows that 3 Scales is the perfect balance between purity and effort when annotating the THUMOS14 test set.

4.5. Exp 4: Optimal Number of Scales

When inputting the features in the pipeline, the user has to choose how many scales the HSNE analysis is going to have. This number must be chosen depending on multiple factors, including the dataset, the input features and the granularity desired for annotation. For example, imagine you want to annotate a tennis dataset. You will need more scales if you want to annotate all the different shots compared to only annotating when a ball is hit.

In this experiment, we want to find the optimal number of scales needed when annotating using the THUMOS14 features, described in subsection 4.1. This way, we want to answer the following question: What is the optimal number of scales to use in HSNE for balancing annotation effort and purity in a specific dataset?

In this experiment, we calculate purity per scale when using THUMOS14 features. The purity per scale was calculated using the formula

$$Purity_{scale} = \frac{1}{E} \sum_{e\,=\,1}^{E} \left(\frac{1}{N_e} \sum_{i=1}^{k_e} max_j \left| c_i \cap t_j \right| \right)$$

where:

- E is the number of embeddings at this scale;
- N_e is the number of points in this embedding;
- k_e is the number of clusters in this embedding;
- c_i is a cluster in this embedding;
- t_j is the ground truth label of this cluster.

The drilling was done using the Agglomerative Clustering - Single Linkage algorithm.

Figure 10 presents the average purity per scale when using THUMOS14 features for the HSNE analysis. The plot bar plot presents 3 types of analyses: 2-Scale, 3-Scale and 4-Scale analysis. We see the 4-Scale analysis can reach the highest purity per scale in Scale 1. However, the improvement compared to the 3-Scale analysis is not significant enough to justify the added effort required for the human to drill through the analysis. Then, the 2-Scale cannot achieve the same level of purity, as the scales have to deal with more points on average. Therefore, we find that when using THUMOS14 features, the best choice is a 3-Scale analysis.

4.6. Exp 5: Impact of Displayed Points

After the drilling in the hierarchy has been done, the final step in the annotation pipeline is to annotate the landmarks in the first scale. To do this, the human annotator has to draw using a lasso tool. Intuitively, the difficulty of this process depends on multiple factors, including the dataset, the number of landmarks displayed and the level of granularity expected for the annotations. Out of these factors, the only factor we can influence is the number of displayed points by our pipeline. Naturally, we want to answer the following questions: How does the number of points displayed on the screen affect the annotation experience, and what is the recommended setting for two specific datasets?

The number of displayed landmarks on the first scale is based on the total number of points and how the drilling was performed. Separating clusters while drilling results in fewer points displayed at once at the first scale. However, the drilling process also takes cognitive effort from the human annotator. Therefore, if we can find a desirable range for

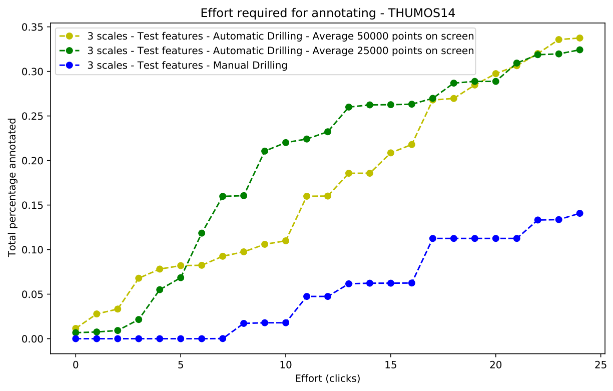


Figure 11. **Exp 5:** The total percentage we could annotate with 25 clicks when using manual and automatic drilling on THUMOS14. It is visible that automatic drilling can save effort.

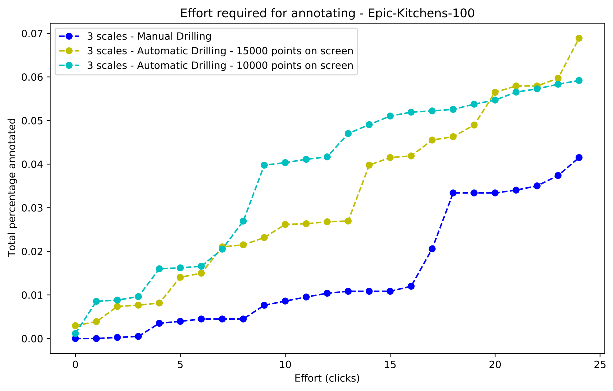


Figure 12. **Exp 5:** The total percentage we could annotate with 25 clicks when using manual and automatic drilling on Epic-Kitchens-100. It is visible that automatic drilling can save effort.

the amount of landmarks displayed, we can automate the drilling process and save effort.

We start this experiment by manually drilling and annotating 25 times. We then try to empirically find the right amount of points to be displayed for the THUMOS14 and Epic-Kitchens-100 datasets using automatic drilling.

The automatic automatic drilling was performed using KMeans clustering, from the Scikit-Learn library [38]. We chose this method for its ability to create regular clusters, mimicking a basic approach taken by a human annotator. KMeans takes the number of clusters to find as a parameter. We vary this number based on how many displayed points we want to have at the first scale on average.

Figure 11 shows how much percentage we were able to annotate in 25 steps using THUMOS14. We found that on average 25000 to 50000 landmarks was the right amount of displayed points for this dataset. We also tried more than 50000 landmarks displayed on average and the annotation process became infeasible. Moreover, we found that drilling can be done automatically, without affecting the annotation process. This way we could save effort and annotate more with the same amount of clicks.

To verify that the automatic drilling works, we annotated again for 25 clicks on Epic-Kitchens-100. We found the range of displayed points for this case to be between 10000 and 15000. The results are presented in figure 12. We cannot compare the results between Epic-Kitchens-100 and THUMOS14 as the datasets' difficulty and the features' quality are completely different. Nevertheless, we can conclude that automatic drilling works on both datasets and can save effort without impacting the annotation process.

5. Conclusion

Our proposed video annotation pipeline demonstrates significant potential for accelerating temporal action annotations. By using pre-extracted features and hierarchical dimensionality reduction, we achieve more than 10 times improvement in annotation effort compared to traditional annotation methods. Our experiments across multiple datasets highlight the effectiveness of our approach when used on datasets with multiple similar videos and provide insights into optimizing the pipeline for different scenarios.

One limitation of our annotation pipeline is the dependency on the quality of pre-extracted features. Additionally, the need for manual tuning of pipeline parameters for optimal results in different scenarios can be time-consuming and may require expert knowledge. Furthermore, our current evaluation is based on the number of clicks, as an approximation of the cognitive effort necessary for the annotator to complete the task.

Future work directions should include conducting a user study to evaluate the usability of the annotation pipeline and quantify the improvement in terms of annotation quality and cognitive effort compared to traditional annotation methods. Afterwards, the annotation pipeline can be used to create new datasets and eventually automatically adapt the pipeline's parameters based on scenario characteristics.

To conclude, our work provides a promising foundation for addressing the challenge of efficiently annotating large-scale video datasets. As video understanding tasks continue to evolve and demand larger, more diverse datasets, scalable annotation methods like ours will play a crucial role in advancing the field and its real-world applications.

References

[1] Moloud Abdar, Meenakshi Kollati, Swaraja Kuraparthi, Farhad Pourpanah, Daniel McDuff, Mohammad Ghavamzadeh, Shuicheng Yan, Abduallah Mohamed, Abbas Khosravi, Erik Cambria, and Fatih Porikli. A review of deep learning for video captioning, 2023. 2

[2] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. 3

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021. 2

[4] Olivier Aubert, Yannick Prié, and Daniel Schmitt. Advene as a tailorable hypervideo authoring tool: a case study. In *Proceedings of the 2012 ACM symposium on Document engineering*, pages 79–82, 2012. 2

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, page 4, 2021. 2

[6] Paulo Vinicius Koerich Borges, Nicola Conci, and Andrea Cavallaro. Video-based human behavior understanding: A survey. *IEEE transactions on circuits and systems for video technology*, 23(11):1993–2008, 2013. 2

[7] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015. 2

[8] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 2, 4, 5

[9] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, pages 1–23, 2022. 2, 3, 5

[10] Guodong Ding, Fadime Sener, and Angela Yao. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2

[11] Abhishek Dutta and Andrew Zisserman. The via annotation software for images, audio and video. In *Proceedings of the 27th ACM international conference on multimedia*, pages 2276–2279, 2019. 1, 2, 3, 6

[12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019. 2, 4

[13] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4021–4036, 2020. 2

[14] Bernard Ghanem, Juan Carlos Niebles, Cees Snoek, Fabian Caba Heilbron, Humam Alwassel, Victor Escorcia, Ranjay Krishna, Shyamal Buch, and Cuong Duc Dao. The activitynet large-scale activity recognition challenge 2018 summary. *arXiv preprint arXiv:1808.03766*, 2018. 3

[15] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13505–13515, 2021. 1

[16] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017. 2

[17] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. Datasets are not enough: Challenges in labeling network traffic. *Computers & Security*, 120:102810, 2022. 2

[18] Gaudenz Halter, Rafael Ballester-Ripoll, Barbara Flueckiger, and Renato Pajarola. Vian: A visual annotation tool for film analysis. In *Computer Graphics Forum*, pages 119–129. Wiley Online Library, 2019. 2

[19] Alexander Heimerl, Tobias Baur, Florian Lingenfelser, Johannes Wagner, and Elisabeth André. Nova-a tool for explainable cooperative machine learning. In *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 109–115. IEEE, 2019. 2, 3

[20] Kristina Host and Marina Ivašić-Kos. An overview of human action recognition in sports based on computer vision. *Heliyon*, 8(6), 2022. 2

[21] Emil Hovad, Therese Hougaard-Jensen, and Line Katrine Harder Clemmensen. Classification of tennis actions using deep learning. *arXiv preprint arXiv:2402.02545*, 2024. 1

[22] Kai Hu, Chaowen Shen, Tianyan Wang, Keer Xu, Qingfeng Xia, Min Xia, and Chengxue Cai. Overview of temporal action detection based on deep learning. *Artificial Intelligence Review*, 57(2):26, 2024. 2

[23] De-An Huang, Vignesh Ramanathan, Dhruv Mahajan, Lorenzo Torresani, Manohar Paluri, Li Fei-Fei, and Juan Carlos Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7366–7375, 2018. 2

[24] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017. 2, 5

[25] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35 (1):221–231, 2012. 2

[26] Vishal Kaushal, Rishabh Iyer, Suraj Kothawade, Rohan Mahadev, Khoshrav Doctor, and Ganesh Ramakrishnan. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1289–1299. IEEE, 2019. 1

[27] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2

[28] Shuja Khalid, Mitchell Goldenberg, Teodor Grantcharov, Babak Taati, and Frank Rudzicz. Evaluation of deep learning

models for identifying surgical actions and measuring performance. *JAMA network open*, 3(3):e201664–e201664, 2020. 1

[29] Michael Kipp. Anvil: A universal video research tool. *Handbook of corpus phonology*, pages 420–436, 2014. 2

[30] Dmitry Kobak and George C Linderman. Umap does not preserve global structure any better than t-sne when using the same initialization. *BioRxiv*, pages 2019–12, 2019. 3

[31] Yu Kong and Yun Fu. Human action recognition and prediction: A survey. *International Journal of Computer Vision*, 130 (5):1366–1401, 2022. 1

[32] Gal Lavee, Ehud Rivlin, and Michael Rudzsky. Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(5):489–504, 2009. 2

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3

[34] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3889–3898, 2019. 2

[35] Xiaolong Liu, Qimeng Wang, Yao Hu, Xu Tang, Shiwei Zhang, Song Bai, and Xiang Bai. End-to-end temporal action detection with transformer. *IEEE Transactions on Image Processing*, 31:5427–5441, 2022. 1, 2

[36] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022. 2

[37] Md Moniruzzaman, Syed Mohammed Shamsul Islam, Mohammed Bennamoun, and Paul Lavery. Deep learning on underwater marine object detection: A survey. In *Advanced Concepts for Intelligent Vision Systems: 18th International Conference, ACIVS 2017, Antwerp, Belgium, September 18-21, 2017, Proceedings 18*, pages 150–160. Springer, 2017. 1

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 6, 9

[39] Nicola Pezzotti, Thomas Höllt, B Lelieveldt, Elmar Eisemann, and Anna Vilanova. Hierarchical stochastic neighbor embedding. In *Computer Graphics Forum*, pages 21–30. Wiley Online Library, 2016. 2, 3, 4

[40] Soroosh Poorgholi, Osman Semih Kayhan, and Jan C van Gemert. t-eva: Time-efficient t-sne video annotation. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV*, pages 153–169. Springer, 2021. 2, 3

[41] Keerthana Rangasamy, Muhammad Amir As'ari, Nur Azmina Rahmad, Nurul Fathiah Ghazali, and Saharudin Ismail. Deep learning in sport video analysis: a review. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 18 (4):1926–1933, 2020. 1

[42] G Thippa Reddy, M Praveen Kumar Reddy, Kuruva Lakshmanna, Rajesh Kaluri, Dharmendra Singh Rajput, Gautam Srivastava, and Thar Baker. Analysis of dimensionality reduction techniques on big data. *Ieee Access*, 8:54776–54788, 2020. 3

[43] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1049–1058, 2016. 2

[44] Snehesh Shrestha, William Sentosatio, Huiashu Peng, Cornelia Fermuller, and Yiannis Aloimonos. Feva: Fast event video annotation tool. *arXiv preprint arXiv:2301.00482*, 2023. 1, 2, 3

[45] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1): 30–34, 1973. 6

[46] Ombretta Strafforello, Klamer Schutte, and Jan Van Gemert. Are current long-term video understanding datasets long-term? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2967–2976, 2023. 2

[47] Badri Narayan Subudhi, Deepak Kumar Rout, and Ashish Ghosh. Big data analytics for video surveillance. *Multimedia Tools and Applications*, 78(18):26129–26162, 2019. 1

[48] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, et al. Video understanding with large language models: A survey. *arXiv preprint arXiv:2312.17432*, 2023. 2

[49] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 2

[50] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar 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. 3, 4

[51] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 3, 4

[52] Silvia Vinyes Mora and William J Knottenbelt. Deep learning for domain-specific action recognition in tennis. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 114–122, 2017. 1

[53] Junke Wang, Dongdong Chen, Chong Luo, Xiyang Dai, Lu Yuan, Zuxuan Wu, and Yu-Gang Jiang. Chatvideo: A tracklet-centric multimodal and versatile video understanding system. *arXiv preprint arXiv:2304.14407*, 2023. 2

[54] Jan Warchocki, Teodor Oprescu, Yunhan Wang, Alexandru Dămăcuş, Paul Misterka, Robert-Jan Bruintjes, Attila Lengyel, Ombretta Strafforello, and Jan van Gemert. Benchmarking data efficiency and computational efficiency of temporal action localization models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3008–3016, 2023. 2

[55] Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. Elan: A professional framework for multimodality research. In *5th international conference on language resources and evaluation (LREC 2006)*, pages 1556–1559, 2006. 2

[56] Kelvin Wong, Yanlei Gu, and Shunsuke Kamijo. Mapping for autonomous driving: Opportunities and challenges. *IEEE Intelligent Transportation Systems Magazine*, 13(1):91–106, 2020. 1

[57] Chao-Yuan Wu and Philipp Krahenbuhl. Towards long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1884–1894, 2021. 2

[58] Jieping Ye, Ravi Janardan, and Qi Li. Two-dimensional linear discriminant analysis. *Advances in neural information processing systems*, 17, 2004. 3

[59] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, pages 492–510. Springer, 2022. 2, 3, 5

3 Background

In this chapter, we lay the foundation for understanding the technical contributions presented in Chapter 2. We begin by introducing Deep Learning [19] and Neural Networks, the driving forces behind many recent breakthroughs in Artificial Intelligence (AI). Then, we narrow our focus to Computer Vision [52], specifically video understanding [25].

Within video understanding, we discuss two tasks: action recognition [27] and temporal action localization [63]. These tasks are applied to applications ranging from surveillance systems [50] and autonomous vehicles [31] to human-computer interaction [38] and video retrieval [45]. Moreover, action recognition and temporal action localization are fundamental tasks to our contributions.

To complete the relevant techniques necessary for understanding our technical contribution, we explain feature extraction and dimensionality reduction. These techniques are not proprietary to Computer Vision and have been applied in various domains, ranging from Bioinformatics [57] to Natural Language Processing (NLP) [60] and signal processing [3].

Finally, we discuss the annotation process with different data types, focusing on video data. Moreover, we look into the most popular annotation tools, while specifying the ones which can also be applied to our task, temporal video annotation.

3.1. Deep Learning & Neural Networks

Deep Learning is a subset of Machine Learning that focuses on neural networks inspired by the structure and function of the human brain [33]. Neural networks have improved the state-of-the-art in multiple domains including computer vision [30], natural language processing [14], speech recognition [39], and game playing [46].

The fundamental unit of a neural network is the neuron, a simplified model of its biological counterpart. A neuron receives inputs $x_1, x_2, \ldots, x_n$, each associated with a weight $w_1, w_2, \ldots, w_n$, as seen in Figure 3.1. The neuron computes a weighted sum of its inputs and applies an activation function f to produce an output y:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.1}$$

Here, b is a bias term that allows the neuron to adjust its output independently of its inputs. The activation function f introduces non-linearity, enabling the network to learn complex patterns. Common activation functions include:

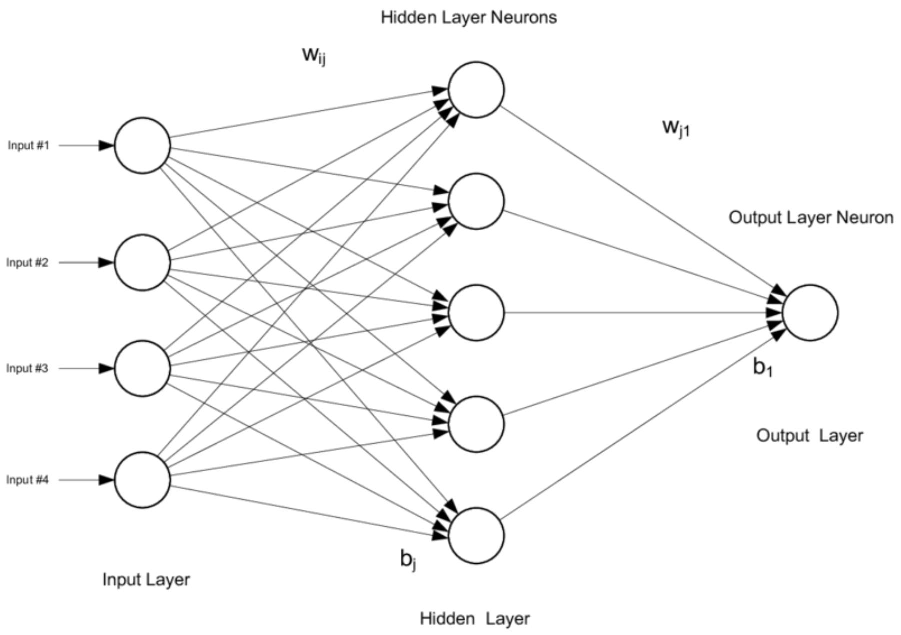


Figure 3.1: A neural network with an input layer, a hidden layer and an output layer. [1]

- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- Hyperbolic Tangent (tanh): $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$

The layers in Figure 3.1 are fully connected or dense layers, where each neuron connects to every neuron in adjacent layers. This architecture requires input data to be in vector form. For images, this means flattening the 2D grid of pixels into a 1D vector. However, this flattening process disrupts the spatial relationships among pixels, causing a loss of valuable information about local patterns and object layouts. This limitation of fully connected layers for image data motivated the development of architectures like Convolutional Neural Networks (CNNs) that preserve spatial context.

3.1.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models particularly effective for processing data with grid-like topology, such as images or time series. Unlike fully connected networks, CNNs leverage three key ideas: local receptive fields, shared weights, and pooling [65]. A model architecture can be seen in Figure 3.2.

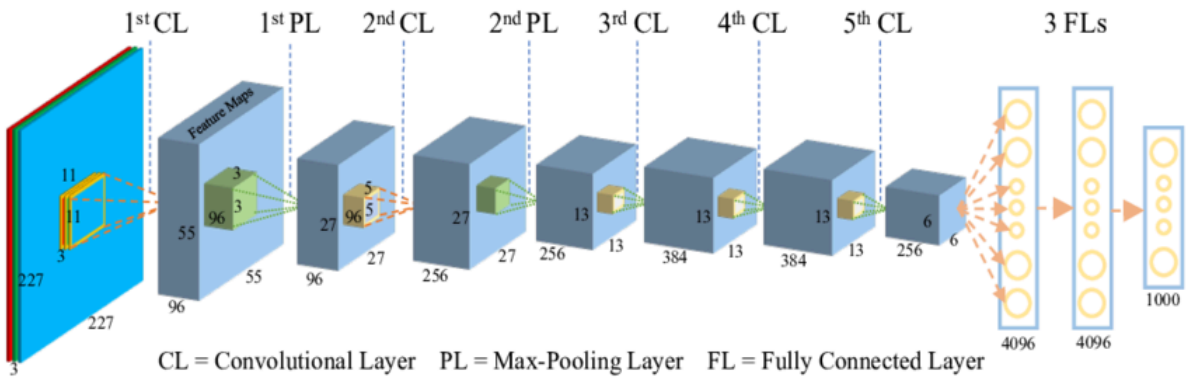


Figure 3.2: A model CNN architecture. [43]

The core building block of a CNN is the convolutional layer. Instead of connecting every input to every neuron, each neuron in a convolutional layer connects to a small, local region of the input, called the receptive field. The neuron applies a set of learnable weights, called a kernel, to this region. Pooling layers, typically inserted between convolutional layers, downsample feature maps. They apply a fixed operation, like max or average, over local neighbourhoods, reducing spatial dimensions. An example of convolution and max-pooling can be seen in Figure 3.3.

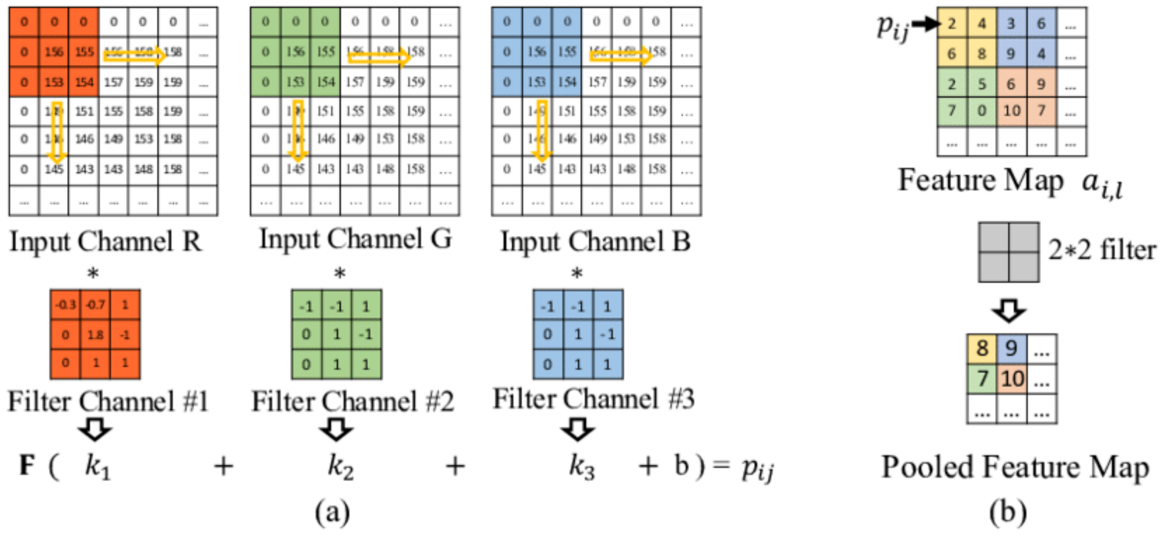


Figure 3.3: Convolution and max-pooling example. [43]

Despite the remarkable success of CNNs in computer vision, they have inherent limitations. CNNs excel at capturing local spatial patterns through their fixed-size convolutional kernels, but they struggle to model long-range dependencies efficiently. In images, this means difficulty understanding relationships between distant parts of the image. In videos, the challenge is even more pronounced: CNNs, even with 3D convolutions, find it hard to capture complex temporal dependencies across frames, especially in long sequences. Therefore, the researchers introduced a new architecture, called transformers.

3.1.2. Transformers

Transformers are an architecture that first emerged in NLP. They performed well and were introduced to Computer Vision as Vision Transformers (ViT) [15, 37, 58]. The key innovation in ViTs is treating an image as a sequence of patches, analogous to words in a sentence, as described in Figure 3.4. ViTs can directly apply the Transformer encoder to image data by linearly projecting these patches into a lower-dimensional embedding space.

What makes Transformers so special is attention [58]. Self-attention is the core mechanism of Transformers. Unlike CNNs and other models which have a fixed capture radius, self-attention allows the model to dynamically look to relevant parts of the input sequence, regardless of the position. This enables capturing long-range dependencies, which is crucial for understanding high-dimensional data, like natural language, images and videos.

However, Transformers cannot solve all our problems. Transformers can become quickly too computationally expensive for longer sequences. Moreover, Transformers struggle with fine-grained tasks. Therefore, they are not a one-size-fits-all solution.

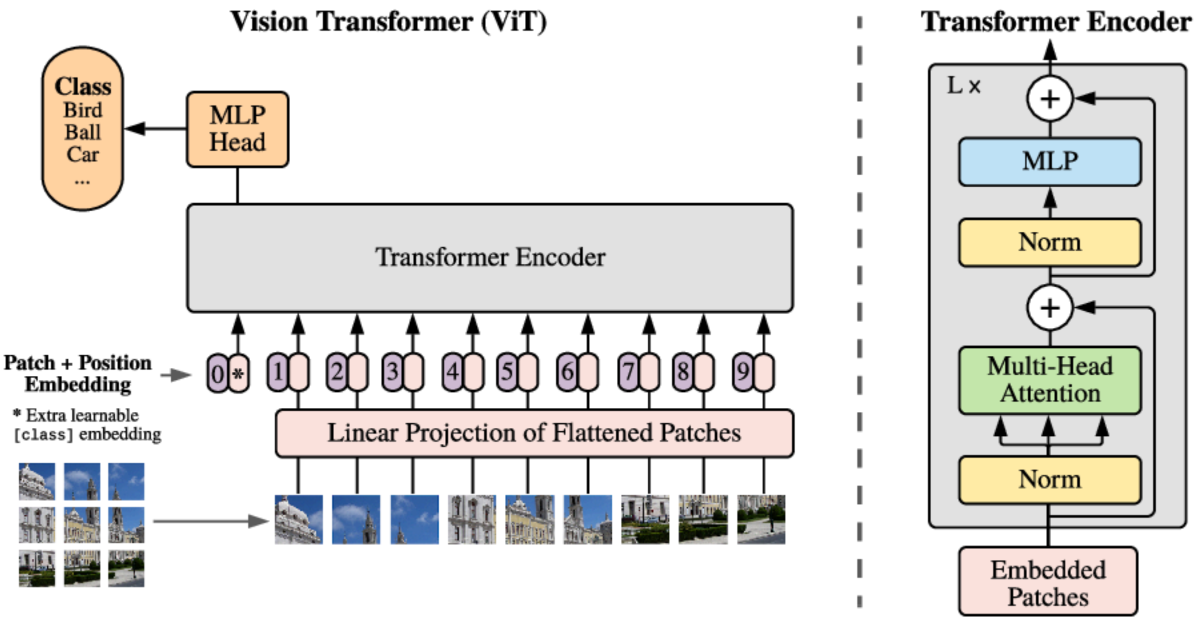


Figure 3.4: Vision Transformer visualisation. [15]

3.2. Computer Vision

Developing the latest architectures in Deep Learning pushed the boundaries of Computer Vision [59]. Initially, Computer Vision focused on fundamental tasks like image classification, object detection, and semantic segmentation, where algorithms processed Red, Green, and Blue (RGB) and Gray-scale images [9, 22, 64]. As the field progressed, researchers tackled increasingly complex and specialized tasks, such as instance segmentation [21], pose estimation [68] and image captioning [51].

These tasks evolved from operating on images to processing richer data modalities. 3D data, such as point clouds [5] or volumetric representations from LiDAR [11] and medical imaging, introduced new challenges in processing irregular, sparse data. With the advancements in hardware and computational power, video understanding emerged as a task [32]. Video data contains not only spatial information like images but also temporal information that captures the evolution of events.

3.2.1. Video Understanding

Video understanding is an umbrella term which includes multiple tasks using video data, like object detection and segmentation in Figure 3.5, action recognition in Figure 3.6 and temporal action localization in Figure 3.7. In essence, video understanding describes the automated interpretation of videos.

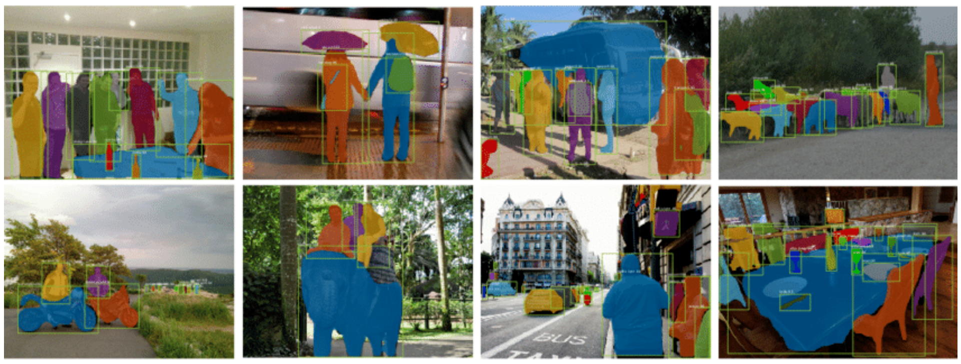


Figure 3.5: Object Detection and Segmentation example [2]

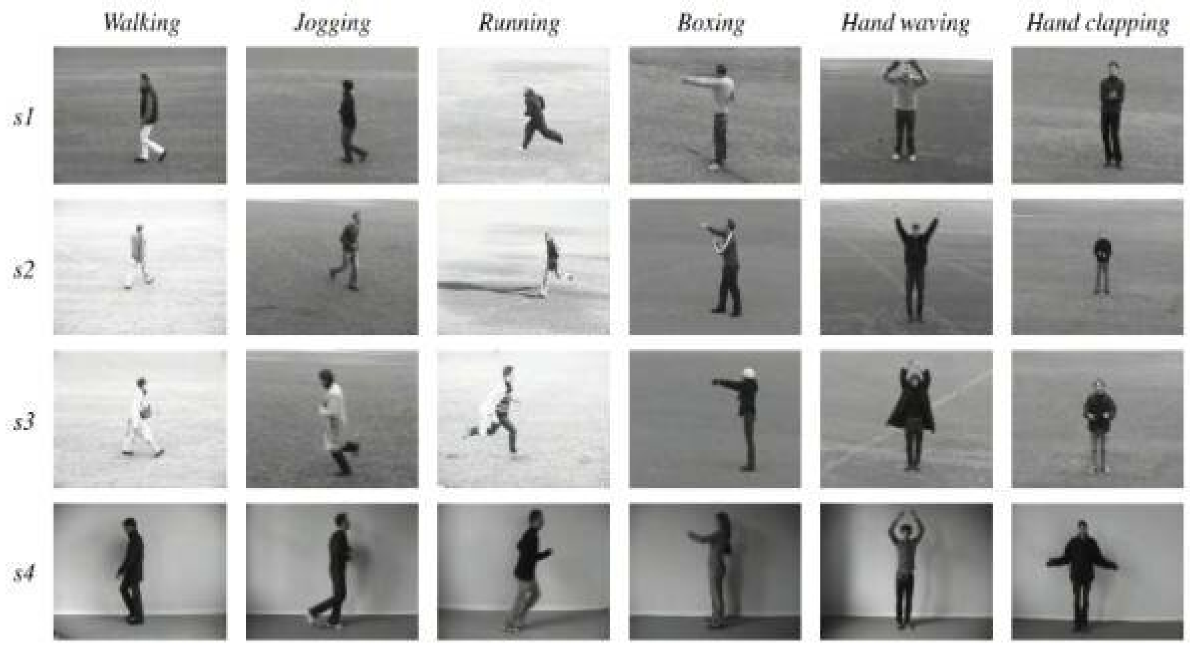


Figure 3.6: Action Recognition example. [67]

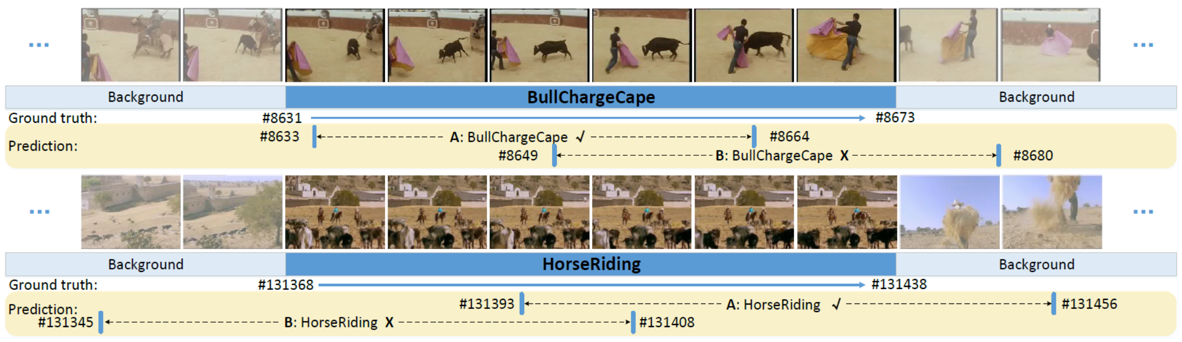


Figure 3.7: Temporal Action Localization. [48]

3.2.2. Action Recognition

Action recognition is one of the main tasks of Video Understanding and Computer Vision. Action Recognition aims to classify videos based on their main action. The action classes can vary in specificity depending on the dataset, ranging from general actions like walking, running, and jumping to specific actions like chopping, cooking, and stirring. Usually, the videos come only from a small set of commonly used datasets, like Kinetics [28], ActivityNet [6] and Something-Something [20].

Multiple approaches and architectures have been created over the years to deal with action recognition in video data. Most of them can be placed in two categories, CNN-based and attention-based.

Approaches such as I3D [7], C3D [54], and Slow-Fast [17] incorporate 3D convolutional networks. 3D convolutional networks extend the traditional 2D convolutions to the temporal dimension, enabling the simultaneous learning of spatial and temporal features from video data. These networks use 3D convolutional kernels spanning multiple adjacent frames, capturing spatial and temporal patterns from video sequences.

In our paper, we used the I3D and Slow-Fast models. Here, we go into more detail explaining how they work:

3.2.3. Two-Stream Inflated 3D ConvNets (I3D)

Two-Stream Inflated 3D ConvNets (I3D) [7] is a model architecture for video action recognition. I3D works by inflating 2D ConvNet models pre-trained on ImageNet [13] into 3D, transforming kernels from 2D to 3D by repeating weights along the time dimension. This inflation allows the model to use ImageNet architectures and their learned representations for action recognition. The I3D architecture is based on the Inception-v1 model [26], with all convolutional and pooling layers inflated to 3D, resulting in two streams: an RGB stream that takes raw video frames as input, and an optical flow stream that captures motion information. The architecture of I3D can be seen in Figure 3.8.

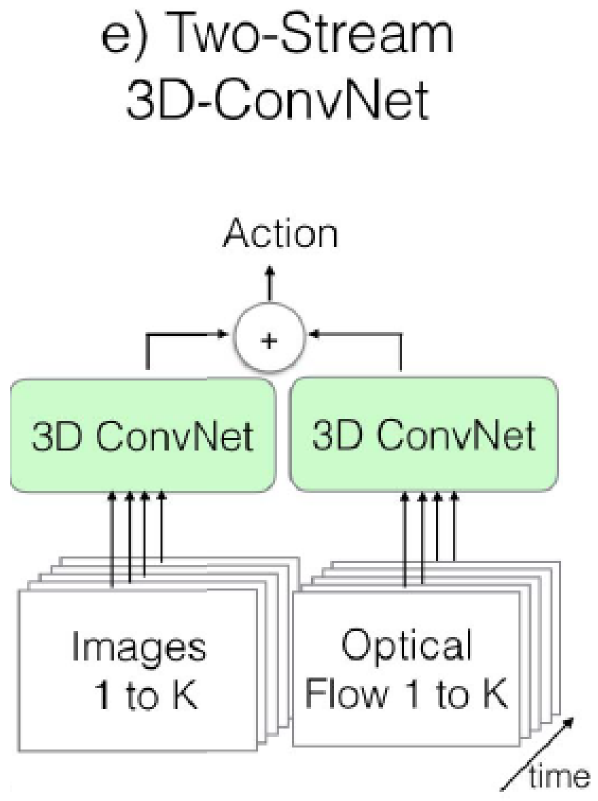


Figure 3.8: Two-Stream Inflated 3D ConvNets (I3D) architecture. [7]

3.2.4. Slow-Fast

Slow-Fast is a model architecture designed for action recognition [17]. The core idea behind Slow-Fast is to utilize two parallel convolutional streams: a *Slow* pathway that captures spatial semantics at a low frame rate, and a *Fast* pathway that processes the video at a higher frame rate to capture motion information effectively. By operating at different temporal resolutions, the two pathways can focus on complementary aspects of the video data. The Slow-Fast demonstrated significant performance improvements over previous state-of-the-art models on various action recognition benchmarks. The Slow-Fast architecture can be seen in Figure 3.9.

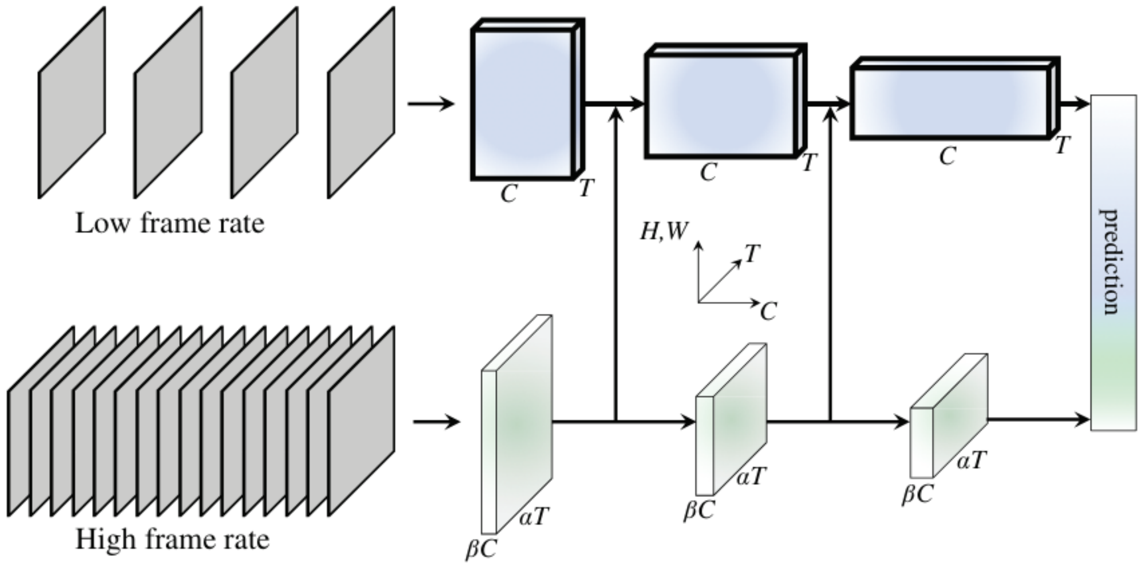


Figure 3.9: Slow-Fast Architecture [17].

Even though multiple action recognition models achieve satisfactory performance, action recognition as a task is usually not the best choice for real-world data. Action recognition works with trimmed videos, so each video contains only one main action. Moreover, action recognition does not identify when the action is happening in the video, making the task unrealistic for most real-world scenarios. Therefore, the temporal action localization task emerged.

3.2.5. Temporal Action Localization

Temporal Action Localization is a more challenging and practical task than action recognition, as it aims to detect and localize action instances in untrimmed, long videos. This task involves not only classifying the action category but also predicting the start and end times of each action instance within the video sequence [48, 63].

Unlike action recognition, where the videos are trimmed to contain only a single action, temporal action localization models must handle untrimmed videos. These videos can contain multiple action instances with background activities, making it harder for the models to accurately distinguish between actions and non-actions and identify the precise temporal boundaries of each action instance.

Researchers have proposed various approaches for temporal action localization, that build upon the success of action recognition models. A common strategy is to leverage the spatial-temporal features learned by pre-trained action recognition models, such as I3D or SlowFast, as the backbone for feature extraction. These features are then fed into temporal action localization models designed to learn to identify action boundaries and classify actions within the context of untrimmed videos. Temporal action localization models can be classified based on the level of supervision and their design method.

In terms of supervision, the models are classified into fully supervised, weakly supervised and unsupervised [24]. A visual intuition of each category can be seen in Figure 3.10. Fully supervised models require complete annotation of the action, including the label and the start and end times of the action. Weakly supervised models require only the action label, while unsupervised models do not require any label. The need for weakly supervised and unsupervised models arose because obtaining detailed annotations is time-consuming and labour-intensive, limiting the availability of large-scale fully annotated datasets. However, their performance is not yet high enough to be used in practice [24].

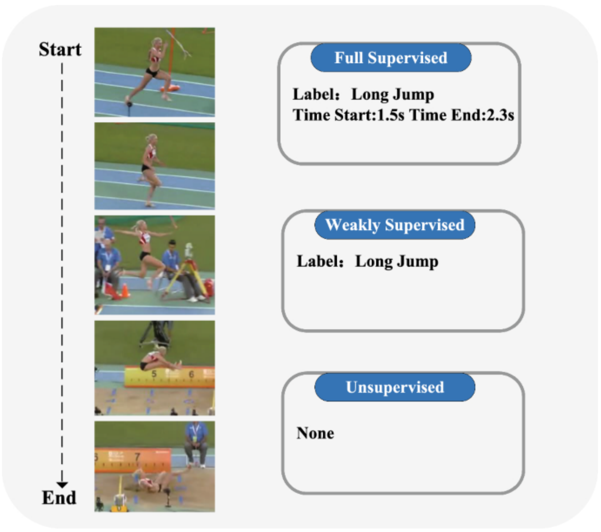


Figure 3.10: Supervision levels of temporal action localization models. [24]

In terms of the design method, the temporal action localization models can be classified into anchor-based, boundary-based and query-based [24], as shown in Figure 3.11.

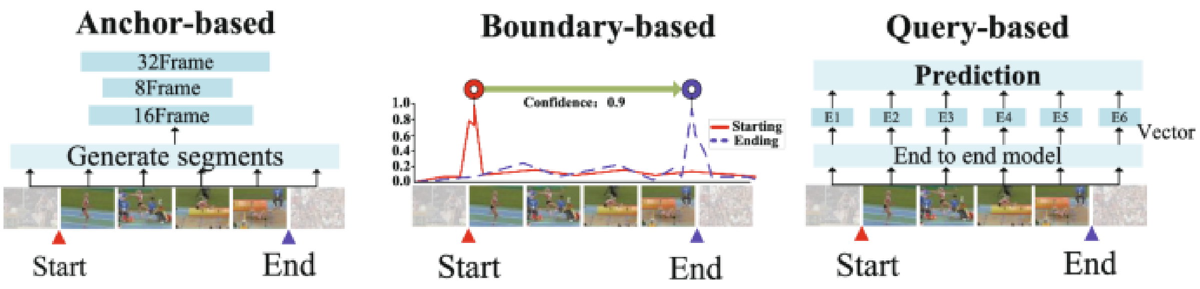


Figure 3.11: Types of Temporal action localization models based on the design. [24, 55]

Anchor-based methods are inspired by the success of object detection models in the image domain. These approaches generate a set of anchor boxes or temporal proposals spanning different start and end times, as in Figure 3.12. The model then classifies each anchor as either containing an action instance or not and refines the temporal boundaries if an action is detected [34, 40, 62].

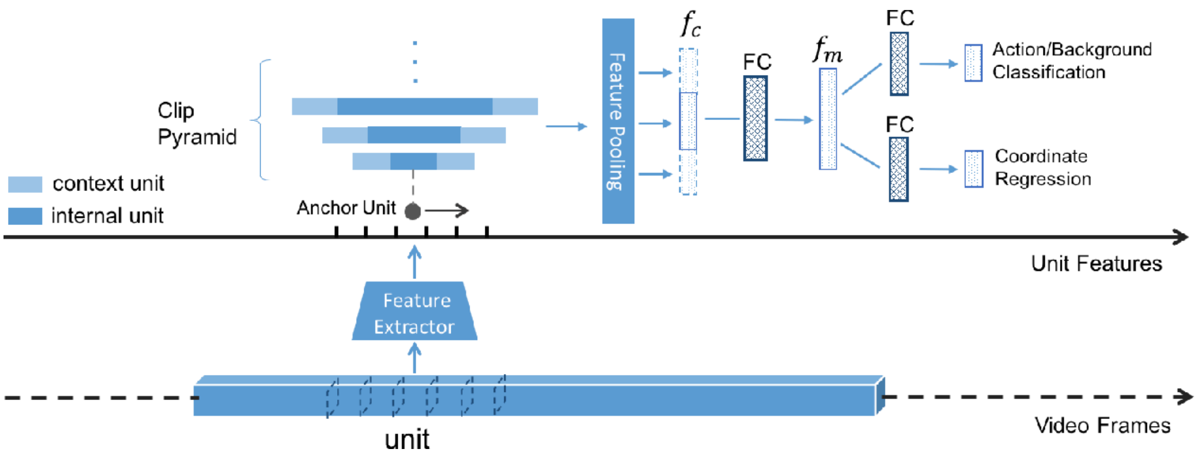


Figure 3.12: Anchor-based architecture. [18]

Boundary-based methods focus on directly predicting the start and end boundaries of action instances. These models use temporal convolutional architectures to capture the temporal dependencies within the video, as in Figure 3.13. By learning to predict the boundary points, boundary-based methods can localize action instances without needing pre-defined anchors or proposals [10, 23, 44].

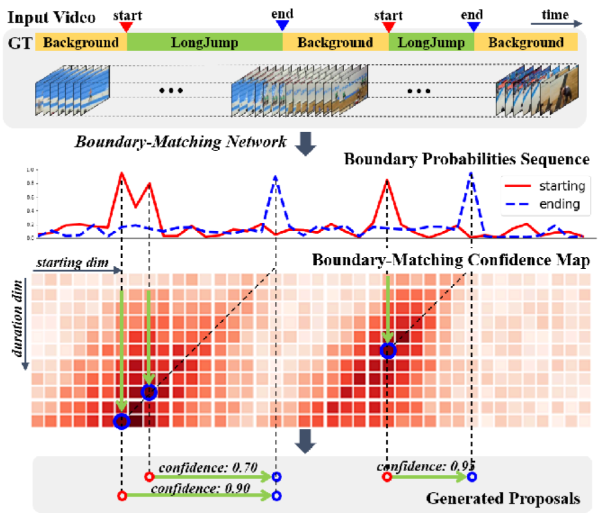


Figure 3.13: Boundary-based architecture. [35]

Query-based methods are a more recent approach, inspired by the success of transformer-based architectures. These models treat action localization as a set prediction problem, where the model generates a set of query vectors that correspond to potential action instances, as in Figure 3.14. Each query vector is then decoded into the action class, start time and end time predictions. Query-based methods can handle a variable number of action instances and can model long-range dependencies effectively [36, 53, 66].

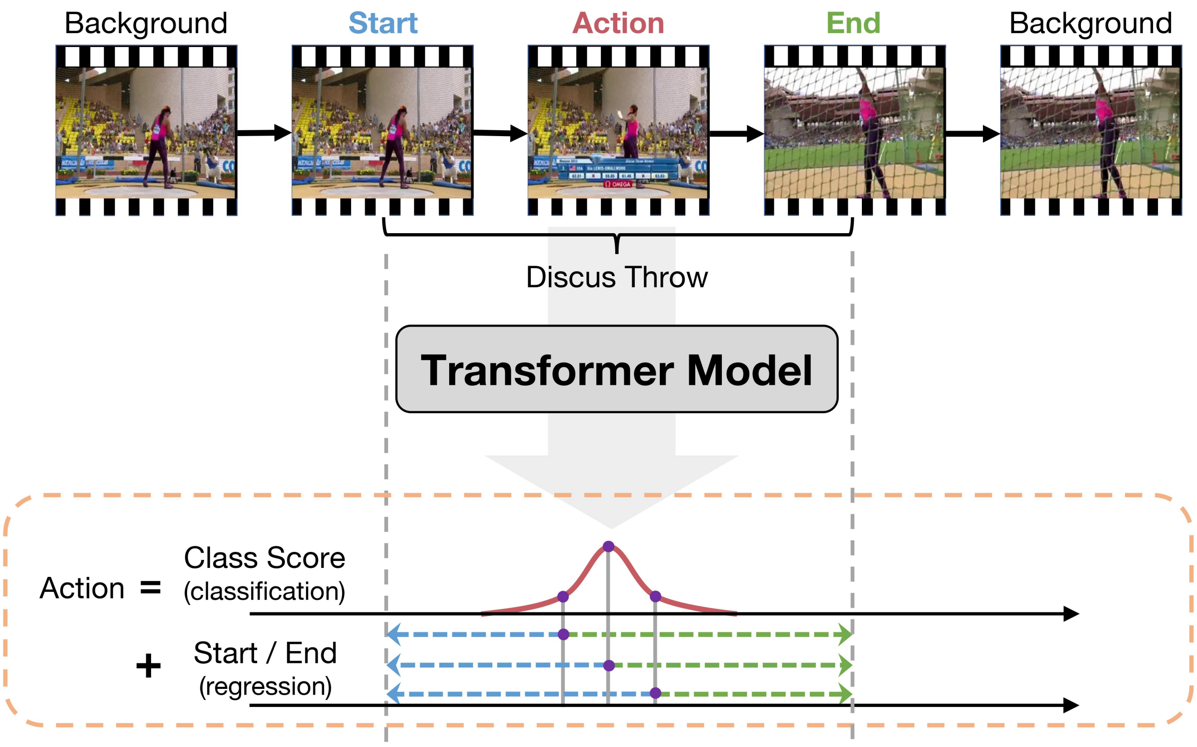


Figure 3.14: Query-based architecture. [66]

3.2.6. Datasets Used

Choosing what dataset to use for training your model depends on the task. Datasets designed for action recognition, such as Kinetics [28], are usually simpler because the videos are trimmed and contain only one main action per video.

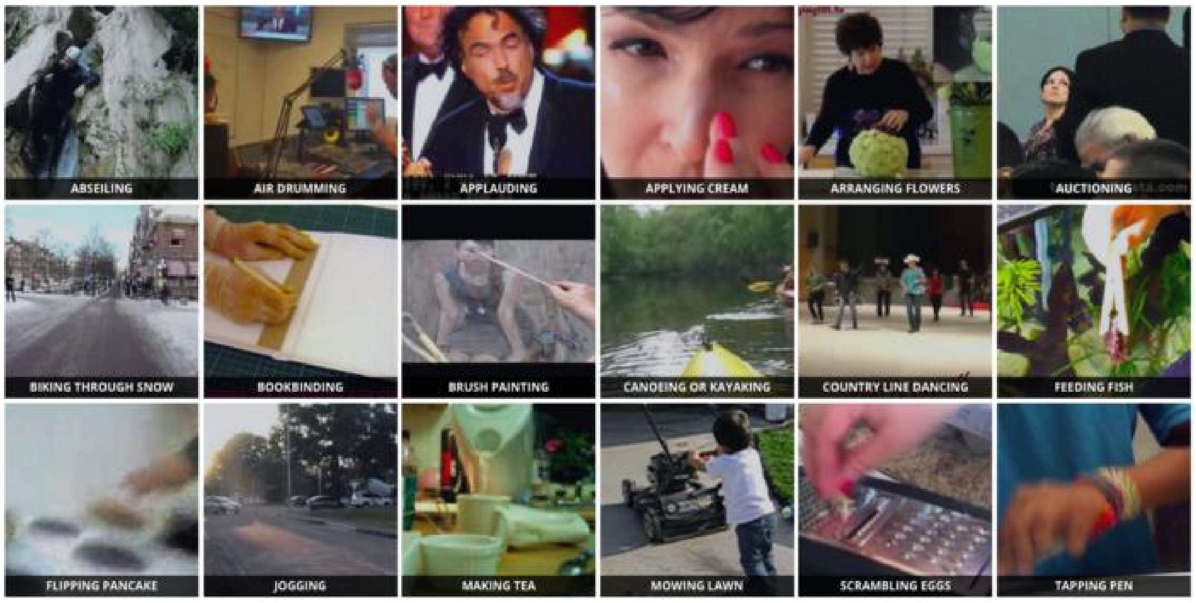


Figure 3.15: Kinetics, a dataset of high-quality annotated videos of human actions. The videos were extracted from YouTube and covered hundreds of human actions, depending on the version. In this dataset, videos last around 10 seconds and are labelled with a single action class. [28]

On the other hand, datasets for temporal action localization are more practical. These datasets contain untrimmed, longer videos, with multiple actions in a video. Moreover, some datasets for the same task are more complex than others, meaning that actions might be shorter, more subtle and require a deeper understanding of the video. This makes the task harder for the model. For example, the EPIC-KITCHENS-100 [12] dataset, Figure 3.16, is more complex than THUMOS14 for temporal action localization [61], Figure 3.17.

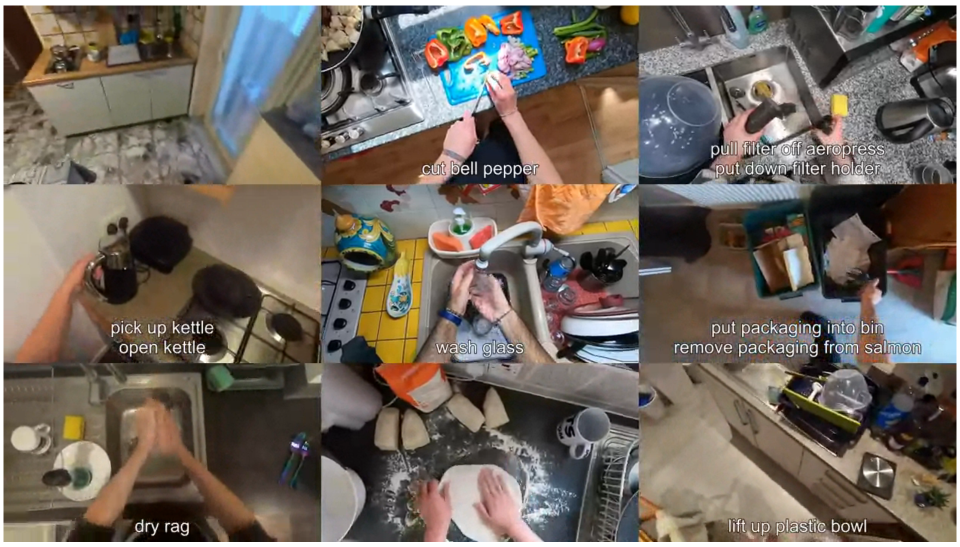


Figure 3.16: EPIC-KITCHENS-100 is a first-person (egocentric) dataset of videos capturing all daily activities in the kitchen. It contains multiple actions per video from 97 verb and 300 noun classes. [12]

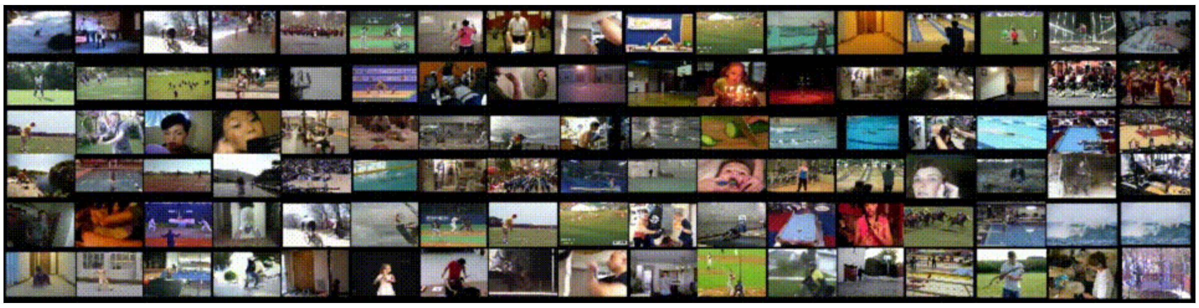


Figure 3.17: THUMOS14 is an action recognition dataset in temporally untrimmed videos. The dataset is used for both action recognition and temporal action localization. It is a simpler temporal action localization dataset with 20 action classes. [61]

3.3. Video Annotation Tools

Even though unsupervised and few-shot learning have progressed in the deep learning community, most state-of-the-art approaches are still fully supervised. So, enough labelled or annotated data must be provided for training and evaluating a neural network model. Annotation tools are designed to facilitate this process of creating labelled datasets for various tasks, such as image classification, object detection, semantic segmentation, and video annotation. These tools provide user-friendly interfaces and functionalities that speed up the annotation process, making it more efficient and accurate.

Video annotation tools allow users to annotate objects and actions across frames and time segments. In our paper, we care about annotating actions across time segments. This section presents two types of video annotation tools, linear and non-linear. Moreover, we explain two significant dimensionality reduction techniques, t-SNE and HSNE, necessary for understanding how some annotation tools work.

Most popular open-source video annotation tools are characterized by their robustness and versatility. These tools are primarily designed to provide a reliable and consistent annotation experience across various types of video data, sometimes in a collaborative environment, rather than focusing on accelerating the annotation process itself. One of the most widely used open-source video annotation tools is VIA (VGG Image Annotator) [16], developed by the Visual Geometry Group at the University of Oxford.

VIA is a standalone, web-based application that supports annotations for images, audio, and video files. VIA's strength lies in its flexibility, allowing users to define and customize their annotation attributes and metadata fields. This versatility makes VIA suitable for various annotation tasks across domains, such as object detection, instance segmentation and event annotations. VIA's interface can be seen in Figure 3.18.

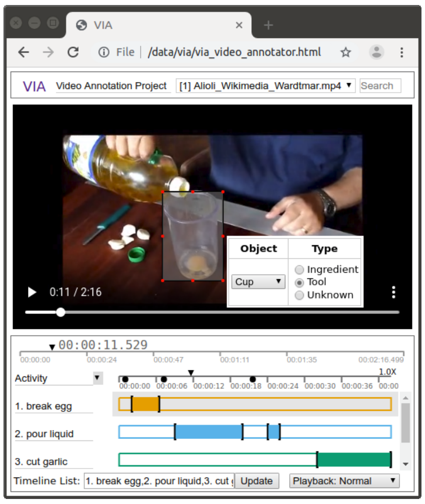

VIA
File | /data/via/via_video_annotator.html
VIA Video Annotation Project [1] Alioli_Wikimedia_Wardtmar.mp4 Search
Object
Type
Cup
Ingredient
Tool
Unknown
0:11 / 2:16
00:00:11.529
Activity
1.0X
1. break egg
2. pour liquid
3. cut garlic
Timeline List: 1. break egg,2. pour liquid,3. cut
Update
Playback: Normal

Figure 3.18: VIA annotation tool. [16]

Another popular open-source annotation tool is CVAT (Computer Vision Annotation Tool) [47], developed by Intel. While CVAT is versatile and supports various annotation formats for images, videos, and point clouds, it was not primarily designed for event annotation tasks. Therefore, annotating events with temporal boundaries is not supported in their tool.

NOVA (the NOVA tool) [4] introduces semi-automation and explainability to the annotation process. This tool aims to reduce the manual effort required by providing automated suggestions and explanations for the annotations, by implementing the pipeline from Figure 3.19. However, this approach does not solve the ”cold-start” problem, which refers to the need for large amounts of labelled data for models to begin performing well. The performance of NOVA is not reported in the paper [4], and we expect the gain in annotation speed to be negligible for most tasks.

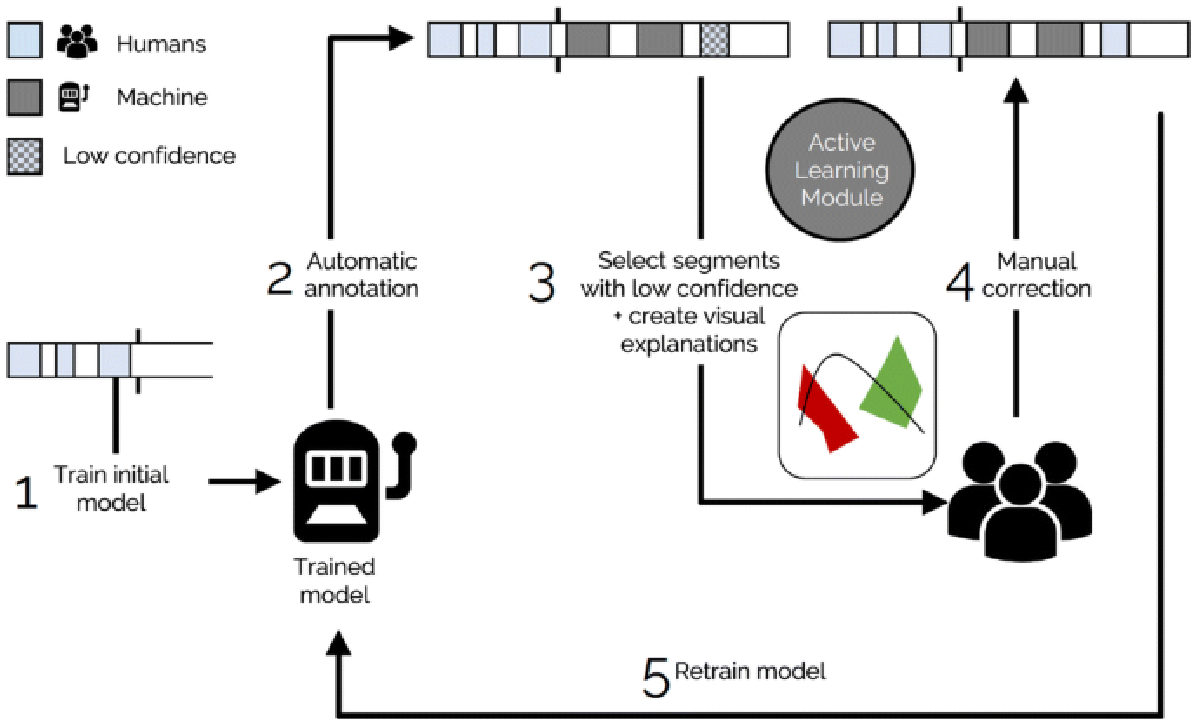


Figure 3.19: The NOVA tool’s pipeline. [4]

Another tool, FEVA [49], attempts to address the steep learning curve associated with annotation software by providing a more user-friendly interface for human annotators. While this approach may improve the usability of the annotation process, it still follows a linear annotation workflow, where the time required for annotation scales linearly with the video length. This means that as the video duration increases, the annotation time grows proportionally, limiting the overall efficiency of the annotation process.

To solve the linearity of the annotation process, t-EVA [42] uses t-SNE (t-Distributed Stochastic Neighbor Embedding) to project high-dimensional video features into a 2D embedding space. This allows annotators to visualize and interact with the video content intuitively, enabling efficient selection and labelling of relevant segments. However, t-EVA suffers from the overflow problem, which arises when the number of data points (video features) exceeds the available space within the 2D embedding. As more features are projected onto the limited 2D canvas, the visualization becomes cluttered and dense, making it increasingly difficult for annotators to distinguish and select specific regions accurately. This overflow problem can severely impact the effectiveness and usability of the annotation process, especially when dealing with large-scale video datasets or long video sequences with numerous events or actions to annotate, as illustrated in Figure 3.20.

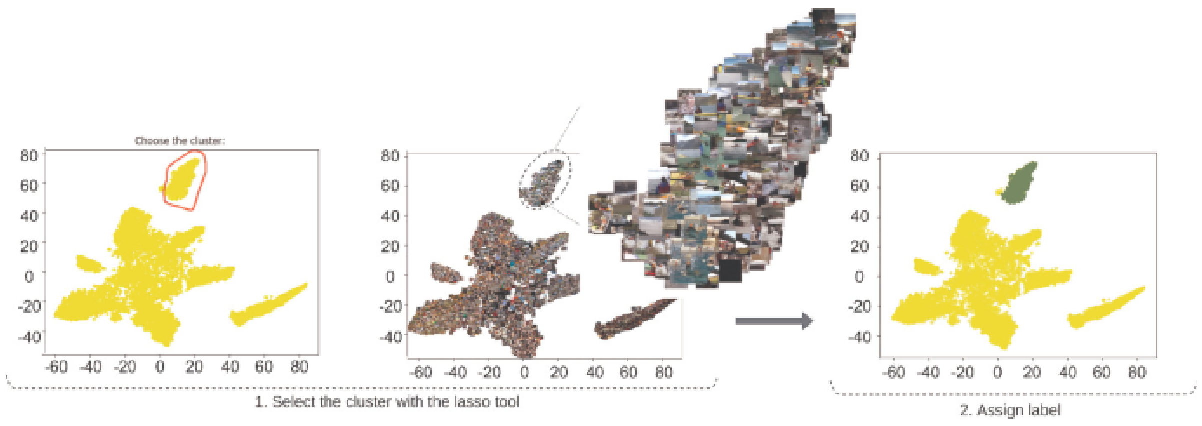


Figure 3.20: Illustration of the t-EVA pipeline. [42]

3.3.1. Dimensionality Reduction

Dimensionality reduction techniques are used in many applications, including video annotation tools. t-SNE is a nonlinear dimensionality reduction technique for high-dimensional data visualisation and exploration. t-SNE works by preserving local relationships between data points, making it useful for exploring similarities and differences in large datasets like those found in genomics [29], image processing [56], or natural language processing [8]. However, t-SNE also has limitations. It can fail when dealing with large datasets due to its computational complexity, and it may produce misleading visualizations if hyperparameters are not tuned. The full explanation of how t-SNE works can be found in the original paper [56].

3.3.2. t-SNE

The main steps of t-SNE are:

1. Compute pairwise similarities between all high-dimensional points using a Gaussian kernel.

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/(2\sigma_i^2))}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/(2\sigma_i^2))} \tag{3.2}$$

   Where:

   - $p_{j|i}$ is the conditional probability that x_i would pick x_j as its neighbor
   - x_i and x_j are high-dimensional data points
   - σ_i is the variance of the Gaussian centred on point i
   - $\|x_i - x_j\|^2$ is the squared Euclidean distance between x_i and x_j

2. Transform pairwise similarities into joint probabilities by normalizing the similarities for each data point.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{3.3}$$

   Where:

   - p_{ij} is the symmetrized joint probability
   - n is the total number of data points

3. Define a similar set of joint probabilities in the low-dimensional space and optimize the positions of low-dimensional points.

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{3.4}$$

   Where:

   - q_{ij} is the joint probability in the low-dimensional space

- y_i and y_j are low-dimensional representations of x_i and x_j

Optimize by minimizing the Kullback-Leibler divergence:

$$C = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) \tag{3.5}$$

Where:

- C is the cost function to be minimized

Update low-dimensional points using gradient descent:

$$y_i^{(t+1)} = y_i^{(t)} - \eta \frac{\partial C}{\partial y_i} \tag{3.6}$$

Where:

- $y_i^{(t)}$ is the position of point i at iteration t
- η is the learning rate
- $\frac{\partial C}{\partial y_i}$ is the gradient of the cost function with respect to y_i

4. Visualize the data by plotting the low-dimensional embedding. This step involves plotting the final values of y_i for all points, typically in a 2D or 3D space, as in Figure 3.21.

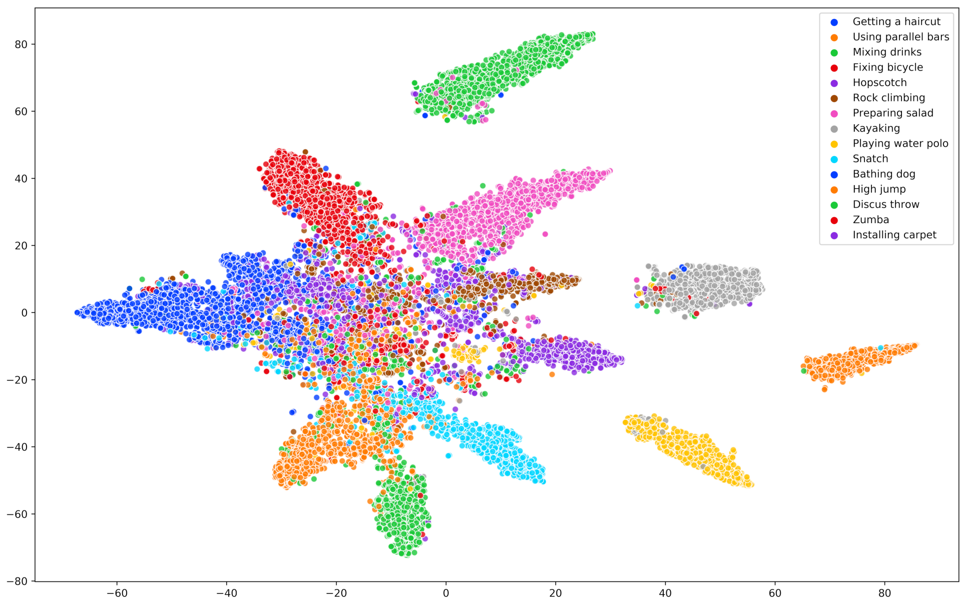


Figure 3.21: t-SNE Example with data from the ActivityNet dataset. [42]

3.3.3. Hierarchical SNE (HSNE)

HSNE has been successfully applied in various domains, including cell analysis in biology [57] and large-scale image collections [41], demonstrating its effectiveness in handling and visualizing massive high-dimensional datasets. The main steps of HSNE are:

1. The Euclidean distances between the high-dimensional data points are computed. The distances are used to calculate each point's k-nearest neighbourhood (KNN) and create a KNN graph.

2. The KNN graph is used to select the landmarks or points in the next scale.
3. For each landmark, an area of influence over the points in the previous scale is computed.
4. Overlaps in the areas of influence are used to create similarities between the points at the new scale; Steps 3 and 4 are repeated to create landmarks for each scale, as illustrated in Figure 3.22.

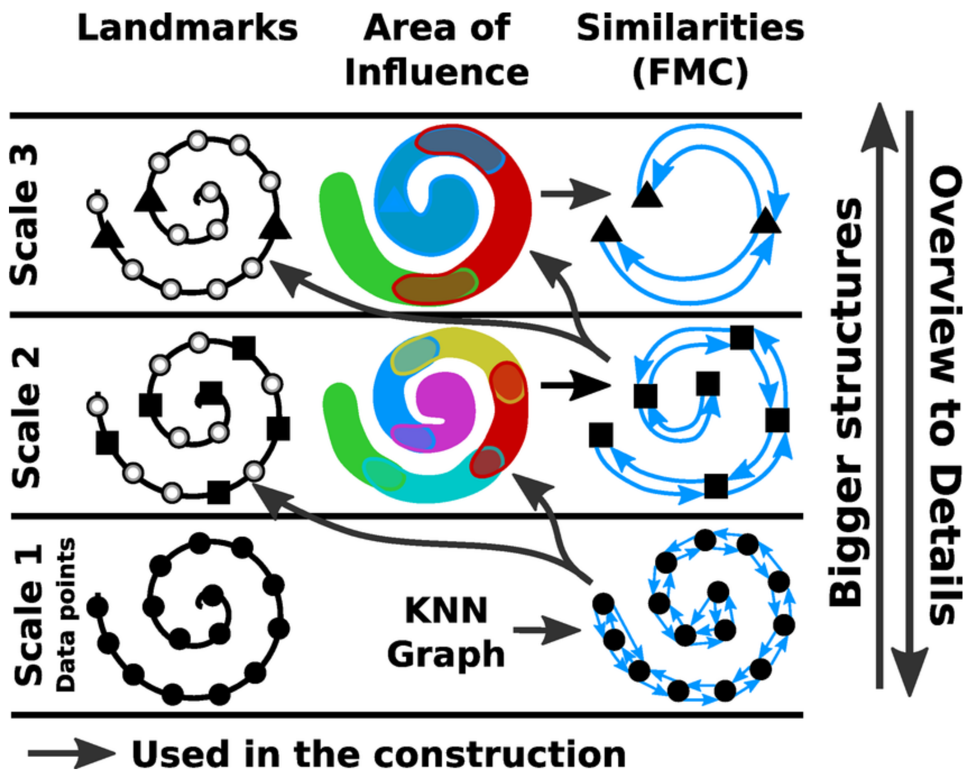


Figure 3.22: The HSNE pipeline. [41]

HSNE offers several advantages over t-SNE:

- Scalability: HSNE can handle much larger datasets than t-SNE, as it does not need to process all data points simultaneously.
- Multi-scale visualization: The hierarchical structure allows data exploration at multiple scales.

However, HSNE also has some limitations:

- Increased complexity: The hierarchical structure adds complexity to the algorithm and its implementation.
- Potential loss of fine-grained detail: using landmark points may result in some loss of detail at higher levels of the hierarchy.

Figure 3.23 shows an example of HSNE applied to a large dataset, illustrating its multi-scale nature.

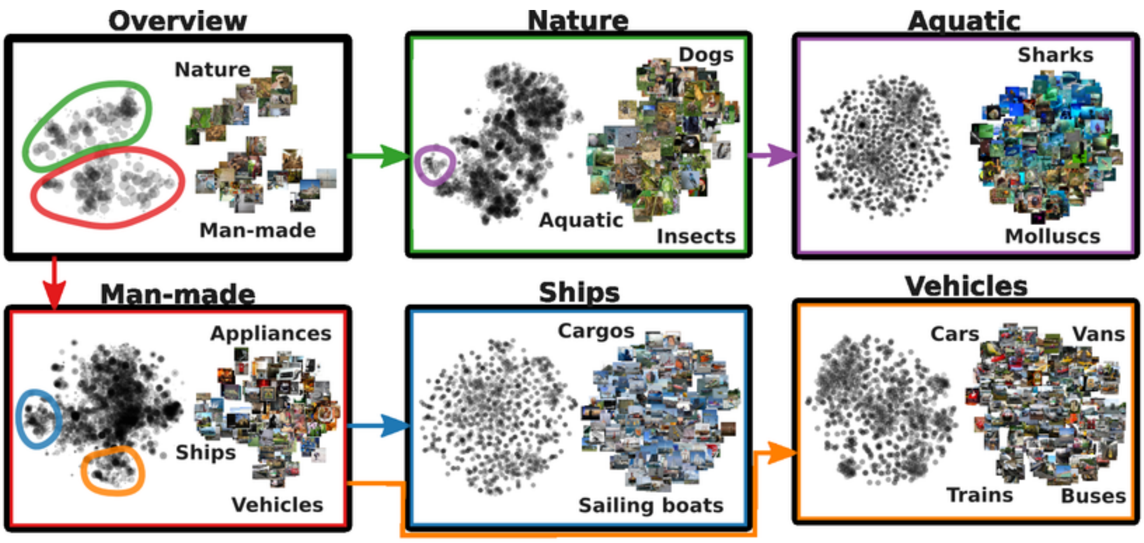


Figure 3.23: Example of HSNE analysis. [41]

References

[1] Mahamad Alam. “Codes in MATLAB for training artificial neural network using particle swarm optimization”. In: *Research Gate* (2016), pp. 1–16.

[2] Yali Amit, Pedro Felzenszwalb, and Ross Girshick. “Object detection”. In: *Computer Vision: A Reference Guide*. Springer, 2021, pp. 875–883.

[3] Sören Anderson. “On optimal dimension reduction for sensor array signal processing”. In: *Signal Processing* 30.2 (1993), pp. 245–256.

[4] Tobias Baur et al. “eXplainable cooperative machine learning with NOVA”. In: *KI-Künstliche Intelligenz* 34 (2020), pp. 143–164.

[5] Saifullahi Aminu Bello et al. “Deep learning on 3D point clouds”. In: *Remote Sensing* 12.11 (2020), p. 1729.

[6] Fabian Caba Heilbron et al. “Activitynet: A large-scale video benchmark for human activity understanding”. In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2015, pp. 961–970.

[7] Joao Carreira and Andrew Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[8] David M Chan et al. “t-SNE-CUDA: GPU-Accelerated t-SNE and its Applications to Modern Data”. In: *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE. 2018, pp. 330–338.

[9] Leiyu Chen et al. “Review of image classification algorithms based on convolutional neural networks”. In: *Remote Sensing* 13.22 (2021), p. 4712.

[10] Yaosen Chen et al. “Boundary graph convolutional network for temporal action detection”. In: *Image and Vision Computing* 109 (2021), p. 104144.

[11] Philip A Chou, Maxim Koroteev, and Maja Krivokuća. “A volumetric approach to point cloud compression—Part I: Attribute compression”. In: *IEEE Transactions on Image Processing* 29 (2019), pp. 2203–2216.

[12] Dima Damen et al. “Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100”. In: *International Journal of Computer Vision* (2022), pp. 1–23.

[13] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[14] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[15] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[16] Abhishek Dutta and Andrew Zisserman. “The VIA annotation software for images, audio and video”. In: *Proceedings of the 27th ACM international conference on multimedia*. 2019, pp. 2276–2279.

[17] Christoph Feichtenhofer et al. “Slowfast networks for video recognition”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6202–6211.

[18] Jiyang Gao et al. “Turn tap: Temporal unit regression network for temporal action proposals”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3628–3636.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[20] Raghav Goyal et al. “The” something something” video database for learning and evaluating visual common sense”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5842–5850.

[21] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. “A survey on instance segmentation: state of the art”. In: *International journal of multimedia information retrieval* 9.3 (2020), pp. 171–189.

[22] Shijie Hao, Yuan Zhou, and Yanrong Guo. “A brief survey on semantic segmentation with deep learning”. In: *Neurocomputing* 406 (2020), pp. 302–321.

[23] He-Yen Hsieh, Ding-Jie Chen, and Tyng-Luh Liu. “Contextual proposal network for action localization”. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2022, pp. 2129–2138.

[24] Kai Hu et al. “Overview of temporal action detection based on deep learning”. In: *Artificial Intelligence Review* 57.2 (2024), p. 26.

[25] De-An Huang et al. “What makes a video a video: Analyzing temporal information in video understanding models and datasets”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7366–7375.

[26] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[27] Hueihan Jhuang et al. “Towards understanding action recognition”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 3192–3199.

[28] Will Kay et al. “The kinetics human action video dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

[29] Dmitry Kobak and Philipp Berens. “The art of using t-SNE for single-cell transcriptomics”. In: *Nature communications* 10.1 (2019), p. 5416.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[31] Sampo Kuutti et al. “A survey of deep learning applications to autonomous vehicle control”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.2 (2020), pp. 712–733.

[32] Gal Lavee, Ehud Rivlin, and Michael Rudzsky. “Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39.5 (2009), pp. 489–504.

[33] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[34] Chuming Lin et al. “Learning salient boundary feature for anchor-free temporal action localization”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 3320–3329.

[35] Tianwei Lin et al. “Bmn: Boundary-matching network for temporal action proposal generation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3889–3898.

[36] Xiaolong Liu et al. “End-to-end temporal action detection with transformer”. In: *IEEE Transactions on Image Processing* 31 (2022), pp. 5427–5441.

[37] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[38] Zhihan Lv et al. “Deep learning for intelligent human–computer interaction”. In: *Applied Sciences* 12.22 (2022), p. 11457.

[39] Mishaim Malik et al. “Automatic speech recognition: a survey”. In: *Multimedia Tools and Applications* 80 (2021), pp. 9411–9457.

[40] Ranyu Ning, Can Zhang, and Yuexian Zou. “Srf-net: Selective receptive field network for anchor-free temporal action detection”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2460–2464.

[41] Nicola Pezzotti et al. “Hierarchical stochastic neighbor embedding”. In: *Computer Graphics Forum*. Vol. 35. 3. Wiley Online Library. 2016, pp. 21–30.

[42] Soroosh Poorgholi, Osman Semih Kayhan, and Jan C van Gemert. “t-eva: Time-efficient t-sne video annotation”. In: *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV*. Springer. 2021, pp. 153–169.

[43] Zhuwei Qin et al. “How convolutional neural network see the world-A survey of convolutional neural network visualization methods”. In: *arXiv preprint arXiv:1804.11191* (2018).

[44] Zhiwu Qing et al. “Temporal context aggregation network for temporal action proposal refinement”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 485–494.

[45] Luca Rossetto et al. “Interactive video retrieval in the age of deep learning–detailed evaluation of VBS 2019”. In: *IEEE Transactions on Multimedia* 23 (2020), pp. 243–256.

[46] Julian Schrittwieser et al. “Mastering atari, go, chess and shogi by planning with a learned model”. In: *Nature* 588.7839 (2020), pp. 604–609.

[47] Boris Sekachev et al. *opencv/cvat: v1.1.0*. Version v1.1.0. Aug. 2020. DOI: `10.5281/zenodo.4009388`. URL: `https://doi.org/10.5281/zenodo.4009388`.

[48] Zheng Shou, Dongang Wang, and Shih-Fu Chang. “Temporal action localization in untrimmed videos via multi-stage cnns”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1049–1058.

[49] Snehesh Shrestha et al. “FEVA: Fast Event Video Annotation Tool”. In: *arXiv preprint* (2023).

[50] GSDMA Sreenu and Saleem Durai. “Intelligent video surveillance: a review through deep learning techniques for crowd analysis”. In: *Journal of Big Data* 6.1 (2019), pp. 1–27.

[51] Matteo Stefanini et al. “From show to tell: A survey on deep learning-based image captioning”. In: *IEEE transactions on pattern analysis and machine intelligence* 45.1 (2022), pp. 539–559.

[52] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[53] Jing Tan et al. “Relaxed transformer decoders for direct action proposal generation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 13526–13535.

[54] Du Tran et al. “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[55] Elahe Vahdani and Yingli Tian. “Deep learning-based action detection in untrimmed videos: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2022), pp. 4302–4320.

[56] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[57] Vincent Van Unen et al. “Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types”. In: *Nature communications* 8.1 (2017), p. 1740.

[58] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[59] Athanasios Voulodimos et al. “Deep learning for computer vision: A brief review”. In: *Computational intelligence and neuroscience* 2018.1 (2018), p. 7068349.

[60] Dongyang Wang, Junli Su, and Hongbin Yu. “Feature extraction and analysis of natural language processing for deep learning English language”. In: *IEEE Access* 8 (2020), pp. 46335–46345.

[61] Limin Wang, Yu Qiao, Xiaoou Tang, et al. “Action recognition and detection by combining motion and appearance features”. In: *THUMOS14 Action Recognition Challenge* 1.2 (2014), p. 2.

[62] Qiang Wang et al. “Rcl: Recurrent continuous localization for temporal action detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 13566–13575.

[63] Huifen Xia and Yongzhao Zhan. “A survey on temporal action localization”. In: *IEEE Access* 8 (2020), pp. 70477–70487.

[64] Youzi Xiao et al. “A review of object detection based on deep learning”. In: *Multimedia Tools and Applications* 79 (2020), pp. 23729–23791.

[65] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into imaging* 9 (2018), pp. 611–629.

[66] Chen-Lin Zhang, Jianxin Wu, and Yin Li. “Actionformer: Localizing moments of actions with transformers”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 492–510.

[67] Jianguang Zhang et al. “Semi-supervised image-to-video adaptation for video action recognition”. In: *IEEE transactions on cybernetics* 47.4 (2016), pp. 960–973.

[68] Ce Zheng et al. “Deep learning-based human pose estimation: A survey”. In: *ACM Computing Surveys* 56.1 (2023), pp. 1–37.