HAVANA: Hierarchical Stochastic Neighbor Embedding for Accelerated Video ANnotAtions

Alexandru Bobe(✉) and Jan C. van Gemert

Delft University of Technology, Delft, The Netherlands
bobe.alexandru.olanda@gmail.com

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 h 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.

Keywords: Video Understanding · Annotation Tool · Feature Extraction · Dimensionality Reduction

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 [31], medical procedures [22], and autonomous driving [48] are delayed due to the lack of high-quality data.

Even in established domains like surveillance [41] and sports [44], 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 [17].

While recent years have seen remarkable advances in video understanding, the models for these tasks are still data-hungry [21,35]. Tasks like action recognition

A. Del Bue et al. (Eds.): ECCV 2024 Workshops, LNCS 15634, pp. 134–150, 2025.
https://doi.org/10.1007/978-3-031-92591-7_9

[25], temporal localization [29], and anticipation [12] 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 Fig. 1. Unfortunately, traditional annotation tools [8,38] force human experts to label iteratively each video from start to finish, making the process unscalable and time-consuming.

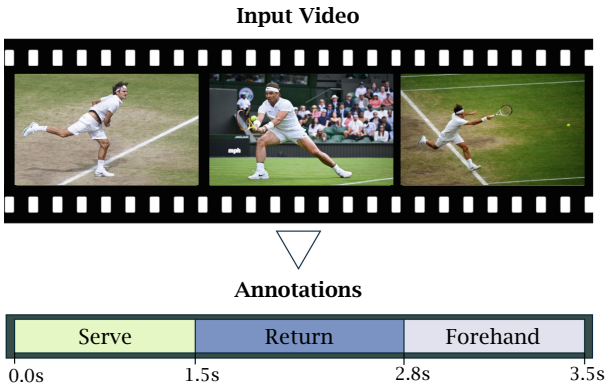

Fig. 1. An example of temporal video annotation. The goal of this task is to create text annotations which identify the actions happening at each moment in the input video. These annotations can be further used to train new models or by domain experts for analysis.

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) [33]. 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 [19,26,42]. Originally, the video understanding domain focused on foundational tasks like determining if an event has occurred and extracting an event summary [26]. Later, the field evolved into more intricate tasks, including captioning videos with descriptions [1], answering questions about videos [45] and anticipating the progression of the videos [10,42]. The field advancement in the complexity of tasks brought the need for more expressive models with increased levels of video interpretation [4,49] and sufficiently large datasets [40]. 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.

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 [7,18,46]. The most popular deep-learning techniques for TAL can be classified depending on the design method into anchor-based methods [37], boundary-based methods [28], and query-based methods [18,29]. Query-based methods are the most recent and naturally perform best when trained with large enough datasets [46,51]. However, large annotated datasets are not available for some real-world specific actions, like tennis videos [16] or network data [13].

Temporal Video Annotations. Temporal video annotation, also called event annotation [38], 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 [6,20]. 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 [3,8,14,23,47]. For example, VIA [8] 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 [34].

The progress in video understanding offers the opportunity for automating parts of the video annotation process. NOVA [15] 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 [15] and we expect the gain in annotation speed to be neglectable for most tasks. FEVA [38] 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 [34] 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 [33]. 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 [51] uses different visual features extracted with various backbones depending on the dataset. For the ActivityNet 1.3 dataset, [11] ActionFormer uses visual features from the R(2+1)D-34 model [43]. Moreover, for the EPIC Kitchens 100 dataset [6] 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 [36]. Two of the most popular linear methods are PCA [2] and LDA [50]. 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) [30] 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.

Moreover, the performance of t-SNE degrades quickly in terms of speed and visualization quality with larger datasets [33]. 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 [24].

A technique called Hierarchical Stochastic Neighbor Embedding (HSNE) [33] addresses both the time performance issues and the overflowing problem. On the MNIST dataset [27], HSNE performs more than ten times faster than t-SNE alone [33]. 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 [8]. 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 320×180 takes approximately 172 800 values. The increased data size when using plain

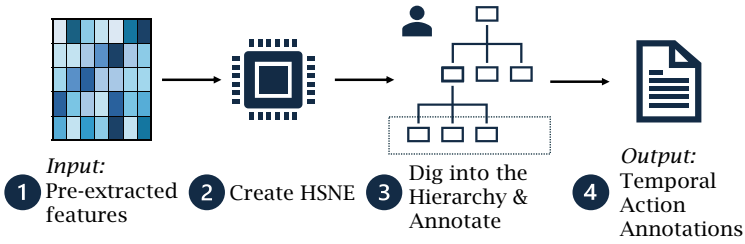

Fig. 2. The figure depicts an overview of our annotation pipeline. (1) The model-agnostic pre-extracted features are used as input in the analysis. (2) The HSNE (Hierarchical Stochastic Neighbor Embedding) analysis is created. (3) Human-in-the-loop approach for digging into the hierarchy and annotating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

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 [5], R(2+1)D [43] and SlowFast [9]. 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 [30]. 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 [30], whereas for how t-SNE works in detail inside HSNE, we refer to this paper [33]. 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 [33].

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

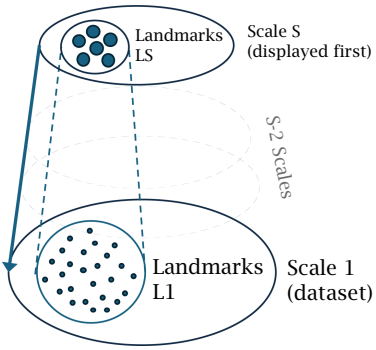

Fig. 3. A visualisation of how the landmarks and scales work in HSNE. The area of influence of the landmarks in Scale S can be seen in Scale 1. The number of intermediate scales varies depending on the user-specified S parameter.

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 [33]. 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.

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

4.1 Datasets and Features Used

We used features extracted from three datasets throughout the experiments: a synthetically generated dataset, Thumos14 [20] and Epic-Kitchens-100 [6].

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* to be 2:1. 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 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 [51], extracted from the THUMOS14 test set. The features were extracted from two-stream I3D

models [5] pre-trained on Kinetics [5], utilizing 16-frame clips at 30 fps and a stride of 4 frames. This configuration gave a single feature vector every 0.1333 s. The total amount of videos used for extraction was 213, which amounted to roughly 12 h of videos. The ratio between *Background* and *Actions* in the videos corresponding to the features was 2:1.

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 [51], 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 s of videos. The total amount of videos used for extraction was 138, which amounted to more than 13 h of videos. The ratio between *Background* and *Actions* in the videos corresponding to the features was 1:2. The ratio shows how action-dense the dataset is compared to THUMOS14.

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 [8].

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 [39] from the Scikit-Learn library [32].

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.

Results in Fig. 4 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 h of videos.

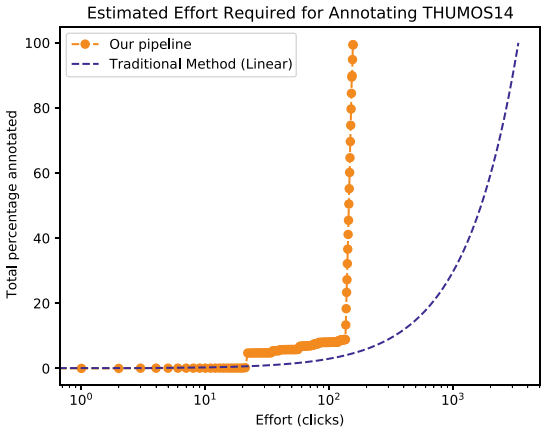

Fig. 4. Estimated effort in clicks needed to annotate the test set of THUMOS14 using our pipeline and a traditional linear method. The results show a more than 10 times improvement when using our method.

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?

When using the perfect features the different classes and the background are perfectly separated from the last scale. 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 throughout the scales to annotate.

Figure 5 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.

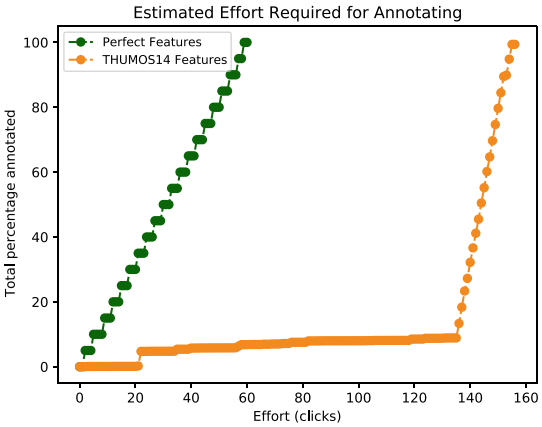

Fig. 5. An estimation of effort required to annotate the test set of THUMOS14. Perfect features represent the synthetic features, created as explained in Sect. 4.1. The THUMOS14 features correspond to the test set features of the THUMOS14 dataset. The plot shows a 50% possible improvement in terms of effort achieved by increasing the feature quality.

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 6 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.

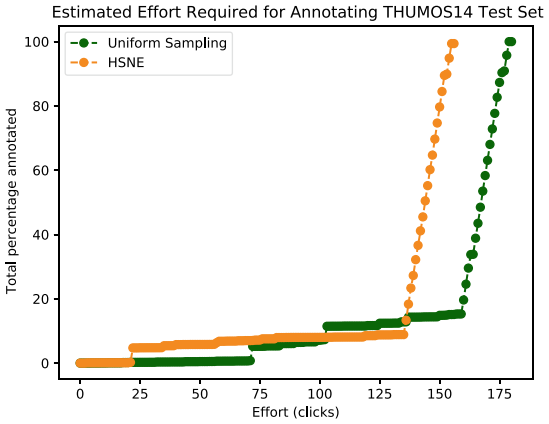

Fig. 6. The estimated effort required to annotate the THUMOS14 test set using extracted features. *Uniform sampling* represents the baseline landmark selection strategy, where each point at *Scale 3* maps to 125 points at *Scale 1*. The plot shows a 13% improvement when using the HSNE landmark selection strategy.

4.5 Exp 4: 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 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 drilling was performed using KMeans clustering, from the Scikit-Learn library [32]. 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 7 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 Fig. 8. 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.

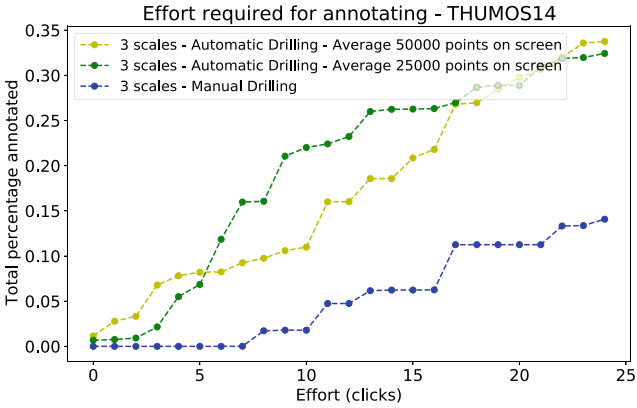

Fig. 7. The total percentage we could annotate with 25 clicks when using manual and automatic drilling on THUMOS14. It is visible that automatic drilling can save effort.

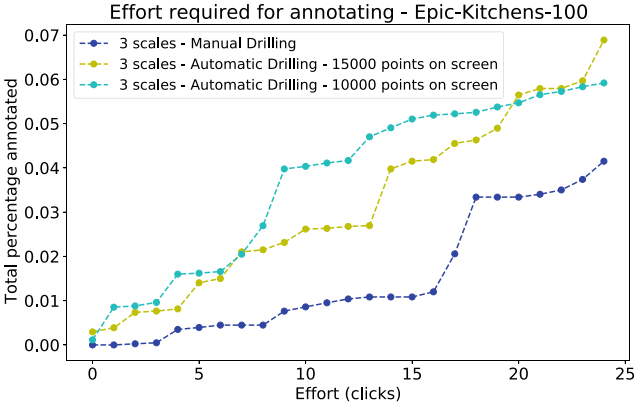

Fig. 8. The total percentage we could annotate with 25 clicks when using manual and automatic drilling on Epic-Kitchens-100. It is visible that automatic drilling can save effort.

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. Abdar, M., et al.: A review of deep learning for video captioning (2023)
2. Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdisc. Rev. Comput. Stat. **2**(4), 433–459 (2010)
3. Aubert, O., Prié, Y., Schmitt, D.: Advene as a tailorable hypervideo authoring tool: a case study. In: Proceedings of the 2012 ACM Symposium on Document Engineering, pp. 79–82 (2012)
4. Borges, P., Conci, N., Cavallaro, A.: Video-based human behavior understanding: a survey. IEEE Trans. Circuits Syst. Video Technol. **23**(11), 1993–2008 (2013)
5. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6299–6308 (2017)
6. Damen, D., et al.: Rescaling egocentric vision: collection, pipeline and challenges for epic-kitchens-100. Int. J. Comput. Vis. 1–23 (2022)
7. Ding, G., Sener, F., Yao, A.: Temporal action segmentation: an analysis of modern techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
8. Dutta, A., Zisserman, A.: The via annotation software for images, audio and video. In: Proceedings of the 27th ACM International Conference on Multimedia, pp. 2276–2279 (2019)
9. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6202–6211 (2019)
10. Furnari, A., Farinella, G.M.: Rolling-unrolling lstms for action anticipation from first-person video. IEEE Trans. Pattern Anal. Mach. Intell. **43**(11), 4021–4036 (2020)
11. Ghanem, B., et al.: The activitynet large-scale activity recognition challenge 2018 summary. arXiv preprint arXiv:1808.03766 (2018)
12. Girdhar, R., Grauman, K.: Anticipative video transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13505–13515 (2021)

13. Guerra, J.L., Catania, C., Veas, E.: Datasets are not enough: challenges in labeling network traffic. Comput. Secur. **120**, 102810 (2022)
14. Halter, G., Ballester-Ripoll, R., Flueckiger, B., Pajarola, R.: Vian: a visual annotation tool for film analysis. In: Computer Graphics Forum, vol. 38, pp. 119–129. Wiley Online Library (2019)
15. Heimerl, A., Baur, T., Lingenfelser, F., Wagner, J., André, E.: Nova-a tool for explainable cooperative machine learning. In: 2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII), pp. 109–115. IEEE (2019)
16. Host, K., Ivašić-Kos, M.: An overview of human action recognition in sports based on computer vision. Heliyon **8**(6) (2022)
17. Hovad, E., Hougaard-Jensen, T., Clemmensen, L.K.H.: Classification of tennis actions using deep learning. arXiv preprint arXiv:2402.02545 (2024)
18. Hu, K., Shen, C., Wang, T., Xu, K., Xia, Q., Xia, M., Cai, C.: Overview of temporal action detection based on deep learning. Artif. Intell. Rev. **57**(2), 26 (2024)
19. Huang, D.A., 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, pp. 7366–7375 (2018)
20. Idrees, H., et al.: The thumos challenge on action recognition for videos "in the wild". Comput. Vis. Image Understand. **155**, 1–23 (2017)
21. Kaushal, V., Iyer, R., Kothawade, S., Mahadev, R., Doctor, K., Ramakrishnan, G.: 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), pp. 1289–1299. IEEE (2019)
22. Khalid, S., Goldenberg, M., Grantcharov, T., Taati, B., Rudzicz, F.: Evaluation of deep learning models for identifying surgical actions and measuring performance. JAMA Netw. Open **3**(3), e201664–e201664 (2020)
23. Kipp, M.: Anvil: A universal video research tool. Handbook of corpus phonology, pp. 420–436 (2014)
24. Kobak, D., Linderman, G.C.: Umap does not preserve global structure any better than t-sne when using the same initialization. BioRxiv, pp. 2019–12 (2019)
25. Kong, Y., Fu, Y.: Human action recognition and prediction: a survey. Int. J. Comput. Vision **130**(5), 1366–1401 (2022)
26. Lavee, G., Rivlin, E., Rudzsky, M.: Understanding video events: a survey of methods for automatic interpretation of semantic occurrences in video. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **39**(5), 489–504 (2009)
27. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
28. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: Bmn: boundary-matching network for temporal action proposal generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3889–3898 (2019)
29. Liu, X., et al.: End-to-end temporal action detection with transformer. IEEE Trans. Image Process. **31**, 5427–5441 (2022)
30. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**(11) (2008)
31. Moniruzzaman, Md., Islam, S., Bennamoun, M., Lavery, P.: Deep learning on underwater marine object detection: a survey. In: Blanc-Talon, J., Penne, R., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2017. LNCS, vol. 10617, pp. 150–160. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70353-4_13
32. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)

33. Pezzotti, N., Höllt, T., Lelieveldt, B., Eisemann, E., Vilanova, A.: Hierarchical stochastic neighbor embedding. In: Computer Graphics Forum, vol. 35, pp. 21–30. Wiley Online Library (2016)
34. Poorgholi, S., Kayhan, O.S., van Gemert, J.C.: t-EVA: time-efficient t-SNE video annotation. In: Del Bimbo, A., et al. (eds.) ICPR 2021. LNCS, vol. 12664, pp. 153–169. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68799-1_12
35. Rangasamy, K., As'ari, M.A., Rahmad, N.A., Ghazali, N.F., Ismail, S.: Deep learning in sport video analysis: a review. TELKOMNIKA (Telecommun. Comput. Electron. Control) **18**(4), 1926–1933 (2020)
36. Reddy, G.T., et al.: Analysis of dimensionality reduction techniques on big data. Ieee Access **8**, 54776–54788 (2020)
37. Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1049–1058 (2016)
38. Shrestha, S., Sentosatio, W., Peng, H., Fermuller, C., Aloimonos, Y.: Feva: fast event video annotation tool. arXiv preprint arXiv:2301.00482 (2023)
39. Sibson, R.: Slink: an optimally efficient algorithm for the single-link cluster method. Comput. J. **16**(1), 30–34 (1973)
40. Strafforello, O., Schutte, K., Van Gemert, J.: Are current long-term video understanding datasets long-term? In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2967–2976 (2023)
41. Subudhi, B.N., Rout, D.K., Ghosh, A.: Big data analytics for video surveillance. Multimedia Tools Appl. **78**(18), 26129–26162 (2019). https://doi.org/10.1007/s11042-019-07793-w
42. Tang, Y., et al.: Video understanding with large language models: a survey. arXiv preprint arXiv:2312.17432 (2023)
43. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6450–6459 (2018)
44. Vinyes Mora, S., Knottenbelt, W.J.: Deep learning for domain-specific action recognition in tennis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 114–122 (2017)
45. Wang, J., et al.: Chatvideo: a tracklet-centric multimodal and versatile video understanding system. arXiv preprint arXiv:2304.14407 (2023)
46. Warchocki, J., et al.: Benchmarking data efficiency and computational efficiency of temporal action localization models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3008–3016 (2023)
47. Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., Sloetjes, H.: Elan: a professional framework for multimodality research. In: 5th International Conference on Language Resources and Evaluation (LREC 2006), pp. 1556–1559 (2006)
48. Wong, K., Gu, Y., Kamijo, S.: Mapping for autonomous driving: opportunities and challenges. IEEE Intell. Transp. Syst. Mag. **13**(1), 91–106 (2020)
49. Wu, C.Y., Krahenbuhl, P.: Towards long-form video understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1884–1894 (2021)
50. Ye, J., Janardan, R., Li, Q.: Two-dimensional linear discriminant analysis. In: Advances in Neural Information Processing Systems, vol. 17 (2004)
51. Zhang, C.L., Wu, J., Li, Y.: Actionformer: localizing moments of actions with transformers. In: European Conference on Computer Vision, pp. 492–510. Springer (2022)