Time-Efficient Video Annotation

with t-SNE

TIME-EFFICIENT VIDEO ANNOTATION

WITH T-SNE

by

Soroosh POORGHOLI

to obtain the degree of Master of Science
at The Delft University of Technology,
to be defended publicly on Monday August 31st, 2020 at 15:00

Student number:	4823001	
Project duration:	November, 2019 – August, 2020	
Thesis committee:	Dr. J. C. van Gemert,	TU Delft, supervisor
	Dr. M. Loog,	TU Delft, Computer Vision Lab
	Dr. T. Hölt,	TU Delft, Graphics and Visualization Group

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

I hope this research can help someone somewhere.

Soroosh Poorgholi

CONTENTS

PREFACE

The current report on *Time-Efficient Video Annotation with t-SNE* is the result of my master graduation project at Delft University of Technology during the academic year 2018-2020. The research has been done under the supervision of Dr. J.C. van Gemert and the daily supervision of Osman S. Kayhan at the TU Delft computer vision lab. I did my bachelor's in electrical engineering and got familiar with artificial intelligence and deep learning during the last year of my bachelor studies. However, understanding how machines can learn like humans and the world of possibilities in front of it was a magical moment for me and made me change my mind and move toward computer science and deep learning in my masters. I faced so many challenges during the adaptation from electrical engineering to computer science in such a short amount of time during my masters; however, I enjoyed the challenging moments I believed in my path and the possibilities of AI to help people.

Furthermore, I would like to thank Jan and Osman for their guidance and support along the way and the opportunity to do my research with them on a real-world problem. I would also like to thank Dr. Marco Loog and Dr. Thomas Höllt for showing interest in my work and participating in my defense committee.

In the end, I do not know how to express my gratitude toward my family for their unconditional love, emotional, moral, and financial support. To my mom, if you are reading this, yes, I finally finished it. To my girlfriend and best friend Dodo, it would have been much more challenging to go through all the challenges without you. Thank you for being there for me and helping me through the difficult times.

Soroosh Poorgholi
The Netherlands,
August 2020

1

Research Paper

Time-Efficient Video Annotation with t-SNE

Soroosh Poorgholi*
s.poorgholi74@gmail.com
Delft University of Technology

Supervisor: Osman Semih Kayhan
Computer Vision Lab
Delft University of Technology

Supervisor: Jan C. van Gemert
Computer Vision Lab
Delft University of Technology

***Abstract*—Video understanding has received more attention in the past few years due to the availability of several large-scale video datasets and improvement in the computational power of computers. However, annotating large-scale video datasets are cost-intensive due to their complexity. In this work, we propose a time-efficient video annotation method using spatio-temporal feature similarity and t-SNE dimensionality reduction to make the annotation process more efficient. Placing the same actions from different videos near each other in the two-dimensional space based on feature similarity helps the oracle to group label the video clips. We evaluate the performance of our method on two subsets of the ActivityNet (v1.3) dataset. We show that our method can outperform conventional video labeling tools time-wise while maintaining a reasonable test accuracy on video classification task compared to the ground-truth labels. To further evaluate the generalization of our method, we test our performance on Sports-1M and Breakfast datasets.**

I. INTRODUCTION

Availability of large-scale video datasets [1]–[3] and increase in computational power of GPUs, has made video understanding in different tasks such as action recognition [4]–[6], object tracking [7]–[9] anomaly detection [10]–[12] an attractive topic of research. Various supervised methods, [5], [6], [13], have improved video classification and temporal localization accuracy on large-scale video datasets such as ActivityNet (v1.3) [1]; however, labeling videos on such a large-scale dataset, requires a lot of human work. Therefore other methods try to train the networks for tasks such as video action recognition in a semi-supervised [14]–[16] manner without having the full labels. To decrease the dependency on the quality and amount of annotated data, [17], [18] investigated pre-training features with internet videos with noisy labels in a weakly supervised manner. Later the model is fine-tuned on the target dataset in a supervised manner. Self-supervised methods have also been proposed [19], where the frame order in unlabeled videos can help the learning process. However, these methods are unable to achieve higher accuracy on video classification tasks than supervised models on large-scale video datasets such as Kinetics [2].

Fully-supervised models require much annotated data that is unavailable as videos are naturally unlabeled, and annotating them is labor-intensive. Large scale datasets [1], [2], [20] use crowdsourcing strategies like *Amazon Mechanical Turk* (AMT) to annotate the videos. [2] uses majority voting between multiple AMT workers to accept annotation of a single video. Using such methods is not efficient for video annotation on a large scale as it requires a lot of time and money. Other tools have been proposed [21] to make the annotation process easier by providing open-source software such as *MuViLab* that enables the oracle to annotate multiple parts of a video at the same time. However, none of these methods exploit the structure of the video data.

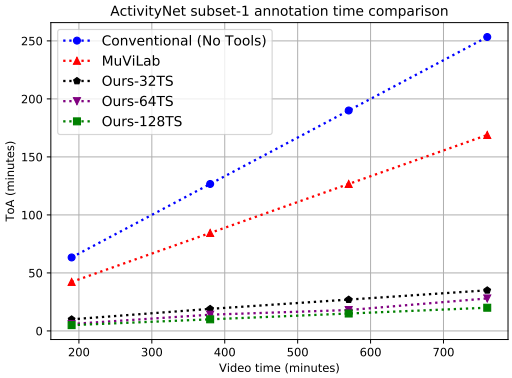


Fig. 1: Comparison of time of annotation (ToA) using different tools versus video time for the ActivityNet [1] subset-1. Our annotation method outperforms the conventional (no specific tools) annotation and MuViLab [21] in annotation time. With a window size of 128 time-steps (128-TS), our method can annotate 769 minutes of video in 21 minutes while MuViLab can do the same in 170 minutes, and conventional annotation can do in 255 minutes. The MuViLab and conventional annotation numbers are extrapolated.

In this work, we introduce a smart annotation tool that can help the oracle group label the videos based on their latent space feature similarity in two-dimensional space. Transferring the high dimensional features to two dimensions using t-SNE, gives the annotator an easy view to group label the videos both temporal labels and classification labels. The annotation speed depends on the quality of the extracted features and how well they are placed together in the t-SNE plot. The better the classes are separated in the t-SNE plot, group labeling is faster for the oracle.

We evaluated our method on two subsets of ActivityNet (v1.3 datasets) [1]. We further evaluate our annotation method

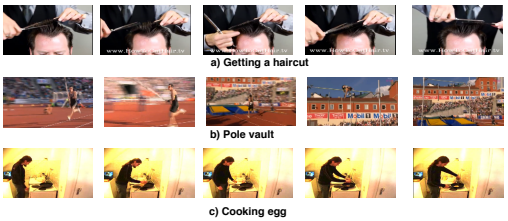


Fig. 2: Example videos from short-range (a) and mid-range (b) actions of ActivityNet (v1.3) [1] and complex action from the Breakfast dataset (c). Short-range actions can be detected using a single frames. Mid-range actions need more frames compared to short-range actions to unfold and complex actions need more time than mid-range action to happen since they contain multiple sub-actions [23]

on 15 random classes of Sports-1M dataset [3]. Finally, to test the generalization of our method, we conduct experiments on a subset of Breakfast [22] dataset, where the videos contain complex actions. An example of complex action compared to short and mid-range actions can be seen in Figure 2. Consecutive frames of a short-range action show a repetitive visual pattern. Therefore an action such as the one in Figure 2-a can be classified even with a few frames as *Getting a haircut*. However, it can be seen in Figure 2-b, the mid-range action needs more time (a few seconds) to unfold, and the action can not be categorized just by looking at the first few frames because it might be High jump or Low jump in this case. The video in Figure 2-c is an instructional video of making eggs from the Breakfast dataset. 2-c needs more time compared to *a* and *b* to unfold since there is a sequence of sub-actions in the video such as *putting oil in the pan, cracking egg*.

We show that our annotation method outperforms conventional annotation techniques (with no specific tools) and a open-source software called MuViLab [21] in time of annotation (ToA) by a large margin on the ActivityNet dataset while still being able to keep the test accuracy on video classification task within a 5% range of the ground truth. Conventional annotation refers to oracle watching the videos and annotating the temporal boundaries of the actions in the video without any specific tool. *MuViLab* is a more advanced open-source tool that extracts short clips from each video and plays them simultaneously in a grid-like figure beside each other. Oracle can annotate the video by selecting multiple short clips at the same time and assigning the specific class.

II. Related Work

Video Understanding has been under lots of development in the past few years. In the past the focus was on the use of specific hand-designed features such as HOG3D [24] SIFT-3D [25], optical flow [26] and iDT [27]. Among these methods, iDT and optical flow have been considered state of the art in hand-designed features and are still being used in combination with CNNs in different architectures such as two-stream networks [28]. Later many attempts have been made to use 2D CNNs and extract features from video frames and combine them with different temporal integration functions [29], [30]. [31] gives an analysis of how to integrate temporal information of video frames with different fusion methods ("*late fusion*", "*slow fusion*", etc.) by using temporal convolutions on top of the spatial convolutions.

Introduction of 3D convolution [5], [32] in CNNs which extends the 2D CNNs in temporal dimension showed promising results in the task of action recognition in large-scale video datasets. At the time of writing this paper using 3D CNNs in different variations such as single stream and multiple-stream are among the state of the art in the task of video understanding [23], [33]–[38]. While some of these methods use computationally expensive features such as optical flow and iDT, other works has also been done to make these approaches faster in training and inference such as [23], [39].

Our method uses extracted features from a 3D Resnet-34 [40] because it is faster compared to two-stream networks with hand-designed features, and it shows promising results.

Dimensionality Reduction Dimensionality reduction (DR) has been an essential tool for high-dimensional data analysis. Linear DR method such as PCA is easier to understand since the lower-dimension representation is a linear combination of the high-dimensional axes. Non-linear methods, on the other hand, are not so easy to explain but are more useful to capture a more complex high-dimensional pattern [41]. In general non-linear DR tries to maintain the local structure of the data in the transition from high-dimension to low-dimension and tends to ignore larger distances between the features [42]. t-Distributed Stochastic Neighbor Embedding (t-SNE) introduced by [43] is a non-linear DR technique which is used more for the visualization purposes. However, t-SNE has raised some concerns regarding the reliability and interpret-ability of the results. [44] mentions some of the disadvantages of using t-SNE such as 1) reliability of the results on the hyper-parameter choice 2) The fact that the cluster size might not mean anything 3) Distance between clusters might be deceiving. However, the data used in [44] is artificial data in which the distribution is known before applying t-SNE. [45] proves that t-SNE is able to distinct well-separable clusters in low-dimensional space. Moreover, some works have been proposed for more effective use of t-SNE. [42] proposes an interactive tool to support interactive exploration and visualization of high-dimensional data. More recent work introduced UMAP algorithm for dimensionality reduction [46]. UMAP minimises the cross-entropy between two fuzzy topological representation to decrease the execution time compared to t-SNE. However, t-SNE shows better results in reducing feature dimensions while keeping the local homogeneity for our dataset which is reported in the ablation study. Moreover, it has been well studied and the complexity has been reduced further to linear using [47]. Therefore, t-SNE is used as the primary method to visualize and inspect high-dimensional features in two dimensions in this work.

Data Annotation is essential for supervised models. Differ-

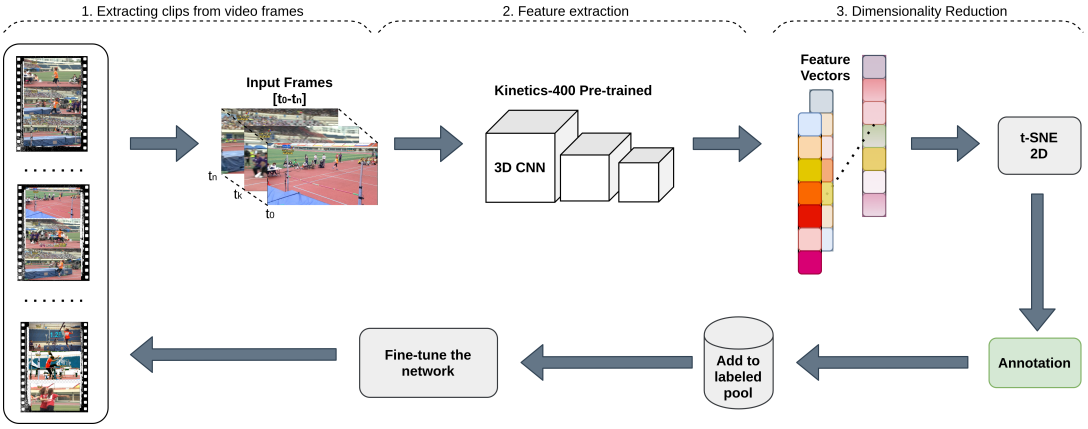


Fig. 3: Proposed pipeline for the smart annotation tool: 1) Video clips are extracted from *n* consecutive frames [t_0-t_n] (time-steps) of the video. 2) Spatio-temporal features are extracted from the last layer of a 3D ConvNet before the classifier layer. 3) High dimensional features are projected to two dimensions using t-SNE, and are plotted on a scatter plot. 4) Oracle annotates the clips represented in the scatter plot using a lasso tool. 5) The newly annotated data is added to the labeled pool. 6) The network is fine-tuned for a certain number of epochs. This cycle is repeated until all the videos are labeled.

ent tools have been proposed for making an easy annotation tool for videos and images, but they usually do not exploit the structure of the data, which is especially useful in videos [21], [48], [49]. Some works [50]–[53] have been done to make the process of image annotation easier. [50] offers a real-time framework for annotating internet images, and [51] uses multi-instances learning to learn the classes and image attributes together; however, none of these methods use a deep representation of data. In more recent works [52] uses *Deep Multiple Instance Learning* to automatically annotate images and [53] uses semi-supervised t-SNE and feature space visualization in lower dimension to provide an interactive annotation environment for images. [54] proposed a general framework for annotating images and videos. However, to best of our knowledge, our method is the first video annotation platform that can exploit the structure of video data to increase the annotation speed.

III. Method

A. Pipeline

In Figure 3, we propose a pipeline for incremental labeling with t-SNE based on feature similarity. In the beginning, a certain number of videos are randomly selected from the unlabeled pool, and the spatio-temporal features are extracted using a 3D ConvNet. Afterward, The features embedding are transferred to a two-dimensional space using t-SNE and visualized on a 2D plot. The oracle has two subplots for annotation: (i) A scatter plot in which the oracle can use a lasso tool to group videos (ii) A scatter plot with the middle frame of each clip in which the oracle can move and zoom with the cursor on the plot and observe where to annotate. After annotating the first set of videos, the video clips are moved to the labeled pool, and the 3D network is fine-tuned for a certain number of epochs on the newly labeled videos. We continue this process until all the videos are labeled.

B. 3D Convolutional Neural Network

To annotate the videos as fast as possible, we avoid calculating features such as iDT and Optical Flow, which are computationally intensive. Instead, we use 3D ConvNets with convolution kernels along the spatial and temporal dimensions to extract features from the videos. Then, we split each video v into k shorter clips $v_i = [clip_1, ..., clip_k]$ by sampling every n non-overlapping frames $clip_i = [frame_1, ..., frame_n]$. Sampling in multiple time-steps enables us to capture different lengths of actions in the dataset. Later each clip c_i is fed into the 3D ConvNet, for feature extraction. The features are extracted from the last *average pooling* layer before the classification layer.

C. Temporal Feature Enhancement

Using single-stream 3D ConvNets, we can extract short and mid-range actions from datasets such as the ActivityNet (v1.3) and Sports-1M; however, 3D ConvNets fail to model complex action of such in Breakfast dataset [23]. The reason is that complex actions contain multiple sub-actions and need a longer duration than short and mid-range actions to happen. Moreover the temporal ordering between the sub-actions diminishes after extracting the features from a 3D ConvNet which can be a helpful cue for the sub-activity level annotation. Therefore, to obtain better temporal coherency through learning the order of sub-actions in the video, we use the method proposed by [55] to train a two-layer multi-layer

perceptron (MLP) to learn the temporal ordering of the sub-actions in complex action. In principle, similar sub-actions can happen in the same order in different videos (e.g. crack the egg before frying the egg). Therefore, we can train a two-layer MLP on the videos belonging to a single complex action to predict the relative time of the sub-actions in the video. We call this method temporal enhancement of the features extracted from the 3D ConvNet. The training loss of the MLP can be seen in Equation 1, which is the mean squared error ($Loss_{mlp}$) of the predicted time (t_{pred}) and the actual time (t_{actual}) of each of the *N* inputs. No sub-action label is used for training, and the MLP only needs to be trained once using video level labels and extracted features as inputs. Later the features are extracted from the second layer of the MLP.

$$Loss_{mlp} = \frac{1}{N}\sum_{t=1}^{N}(t_{pred} - t_{actual})^2 \tag{1}$$

D. *Dimensionality Reduction*

The output of the extracted features is a matrix *X* with dimensions $n \times dim$. The parameter *dim* is the feature vector size from the 3D ConvNet, which depends on the architecture and *n* is the total number of clips generated by the videos.

In t-SNE, the pair-wise distance between each data-point $[(x_i, x_j) \mid 1 \leqslant i, j \leqslant n]$ is transferred to a probability distribution p_{ij} to represent similarity [6].

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

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{3}$$

σ_i is the variance of a Gaussian distribution that is centered around x_i and its value is found iteratively through trial and error using a binary search [43]. The pair-wise similarity in the high-dimensional space is calculated with Equation 2 and 3. Each pair of low-dimensional data-point $[(y_i, y_j) \mid 1 \leqslant i, j \leqslant n]$ is also modeled as a t-Distribution student with one degree of freedom called q_{ij} and is calculated using the Equation 4.

$$q_{ij} = \frac{(1+ \parallel y_i - y_j \parallel^2)^{-1}}{\sum_{k \neq l}(1+ \parallel y_k - y_l \parallel^2)^{-1}} \tag{4}$$

To find a proper joint probability distribution *Q* in low dimensions that represents the high-dimensional joint probability distribution *P*, a cost function *C* has been defined in Equation 5 using Kullback-Leibler (KL) divergence between the probability distributions. The cost function is later optimized with gradient descend in a certain number of iterations.

$$KL(P_i \parallel Q_i) = \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{5}$$

This is the original t-SNE algorithm and has an $O(N^2)$ complexity where *N* is the number of data-points. To increase the speed of calculation of low-dimensional representation different methods has been proposed in [47], [56], [57]. In this paper, we use the Barnes-Hut optimized version, which reduces the complexity of $O(NlogN)$ where *N* is the number of data-points.

E. *How to Annotate?*

Figure 4 shows an overview of the annotation tool, which works in the following way. First, the oracle sees the scatter plot with all points with the same color representing the unlabeled pool (Figure 4 left) and the corresponding middle frame of each clip in the video (Figure 4 middle). Second, by using the lasso tool, the oracle can draw a lasso around the scatter plots based on the visual similarity and inspection of the video frames. Third, labels are assigned by the oracle, and the network is fine-tuned for a certain number of epochs. Fourth, the same process repeats until all the videos are annotated.

IV. EXPERIMENTS

In this section, we first explain the benchmark dataset and evaluation metrics. Further, we empirically show how our method can speed up annotation for the ActivityNet dataset while keeping the video classification accuracy in a close range to the ground truth. We also compare our results with other annotation tools, such as MuViLab [21]. Further, we qualitatively show how our method can help to annotate the Sports1-M [3] and Breakfast datasets [22].

A. *Datasets*

ActivityNet (v1.3) is an untrimmed video dataset with a wide range of human activities [1]. It comprises of 203 classes with an average of 137 untrimmed videos per class in about 849 hours of video. We use two subsets of the ActivityNet dataset. The first subset comprises 10 random classes, namely *Preparing salad, Kayaking, Fixing bicycle, Mixing drinks, Bathing dog, Getting a haircut, Snatch, Installing carpet, Hopscotch, Zumba* consisting of 607 videos with 407 training videos and 200 testing videos. The second subset adds another 5 handpicked classes, which are *Playing water polo, High jump, Discus throw, Rock climbing, Using parallel bars*, and they are visually close to some of the 10 random classes to make the classification task harder. The second subset comprises 950 videos with 639 videos in training and 311 videos in the test set.

Sports-1M is a large-scale public video dataset with 1.1 million YouTube videos of 487 fine-grained sports classes [3]. We choose a subset of 15 random classes of the Sports-1M dataset namely *boxing, kyūdō, rings (gymnastics), yoga, judo, skiing, dachshund racing, snooker, drag racing, olympic weightlifting, motocross, team handball, hockey, paintball, beach soccer* with 702 videos in total. The dataset provides video level annotation for the entire untrimmed video; however, the temporal boundaries of the actions in the video are not identified. Approximately 5% of the videos contain more than one label.

Breakfast is a dataset for human cooking activities from multiple cameras with multiple view-points [22]. It includes 1712 videos with an average length of 2.3 minutes per video.

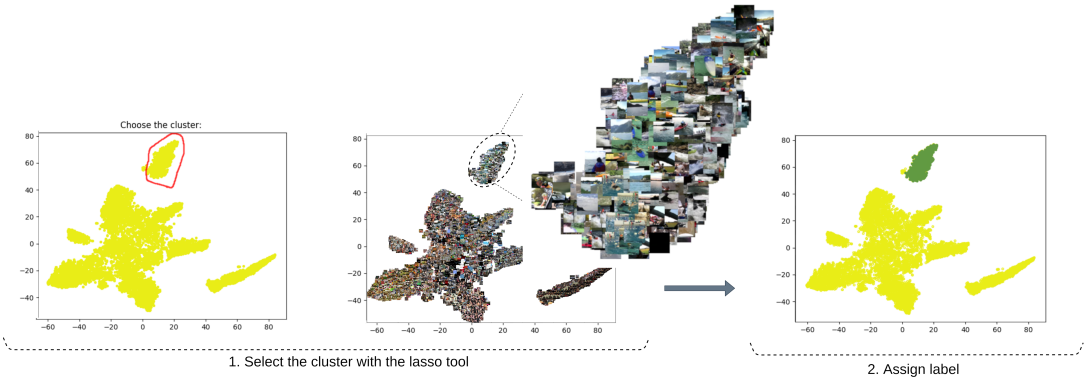


Fig. 4: A minimal representation of the annotation tool. 1) The oracle can see the scatter plot (left) and the corresponding frames from the videos (middle) in separate figures. 2) Based on the inspection of the figures, the oracle can detect different clusters of an action class (kayaking) and use the lasso tool to select the cluster. 3) In the end, the oracle assigns a label, and based on the assigned class name, the selected points in the scatter plot change color.

It contains 10 breakfast preparation classes from 18 different kitchens. Each video represents only one activity class, but it contains sub-actions such as *"put oil in pan", "cracking egg", "frying egg", "putting egg in dish"* under "*fried egg*". The creators of the dataset have provided temporal boundary annotation for each sub-action.

B. *Evaluation Metrics*

To evaluate our method on ActivityNet subsets, we report the time of annotation *(ToA)* as a metric to measure how fast the oracle can annotate a certain number of videos. ToA for conventional annotation and MuViLab on ActivityNet subset-1 is extrapolated since annotating 13 hours of video using these methods was not feasible. We also report video classification accuracy in the form of mean average precision (mAP) for the ActivityNet subsets to measure the quality of annotation when the network is fine-tuned with our annotation versus when fine-tuned with the ground truth annotation. The mAP (%) is used instead of a confusion matrix because some videos of ActivityNet contain more than one action [1].

For the Sports-1M [3] and the Breakfast [22] dataset, we perform a qualitative analysis of the t-SNE projections. Following [55], we also report the accuracy of our method using mean over frames (MoF) for the Breakfast dataset. Mean over frame measures on average how much the temporal boundaries acquired by our annotation matches of those in the ground truth annotation.

To motivate our design choices beyond qualitative results, we introduce a realistic annotation emulation metric to estimate the quality of t-SNE projections on a global and local level. To report how well the t-SNE projection can separate the classes at a global level, we use a measure of cluster homogeneity and completeness. Homogeneity measures if the points in a cluster only belong to one class and completeness measures if all points from one class are grouped in the same cluster. In an ideal t-SNE projection, all the points in each cluster belong to one class (homogeneity=1.0), and all the points from a class are in the same cluster (completeness=1.0), which makes the annotation process much faster. For clustering, K-Means clustering with K being the number of classes is used. We use K-Means because it is fast and has a low number of parameters to select from.

Furthermore, following [53] to emulate the oracle's annotation speed, we also use a measure of local homogeneity using K-nearest neighbors (KNN) with *K=4*. KNN can estimate the lower-dimension local homogeneity between the features in the scatter plot. The higher the KNN accuracy, the higher the local homogeneity and better grouping, which means the oracle can annotate the points faster.

C. *Implementation Details*

Feature Extraction. We use the ResNet-34 3D architecture [40] pre-trained on Kinetics-400 as a feature extractor for all the experiments owing to their good performance and usage of RGB frames only. Following [40], each frame is resized spatially to 112×112 pixels from the original resolution. Each video is transferred to clips by sampling every 32 consecutive frames. The feature extractor in every forward pass takes a clip in the form of a 5D tensor as an input. Each dimension of the input tensor represents the batch size, input color channels, number of frames, spatial height, and width, respectively. Namely, an input tensor for a clip sampled at 32 frames can be shown as (1, 3, 32, 112, 112). The features are extracted after the final 3D average pooling with an (8, 4, 4) kernel before the classifier layer (fully connected layers and softmax). The dimensions of the feature vectors are $k\times512$ with k being the total number of clips and later reduced to $k\times2$ using t-SNE.

t-SNE. For dimensionality reduction, a Barnes-Hut implementation of t-SNE with two components are used from the

scikit-learn library [58]. The perplexity is set to 30, and the early exaggeration parameter is 12, with a learning rate of 200. The cost function is optimized for 2500 iteration.

Training. After annotating each set of videos, the network is fine-tuned for a certain number of epochs. For training the ResNet-34 3D implementation from [40] is used. The sample duration is 32 frames for each clip, and the input batch size is 32 as well. The 3D ConvNet takes a 5D tensor as an input. The parameters are the same as in the feature extractor. Stochastic gradient descends (SGD) is used as the optimizer with a learning rate of 0.1, weight decay of 1e-3, and momentum of 0.9.

Temporal Enhancement. For temporal enhancement of the features, we use the ResNet-34 features of a single complex action to train a two-layer MLP to learn the sub-action coherency. For optimization, Adam optimizer, with a learning rate of 1e-3, is used. After training the network for 100 epochs on features belonging to one complex action, we extract the features from the second layer of the MLP.

D. Results on ActivityNet

ActivityNet Subset-1. First, we put all the 407 videos in the unlabeled pool. Then we divide the videos into four different sets of unlabeled videos. The clips are generated with 32 consecutive frames, and the features are extracted using the Resnet-34 3D. Later the videos are annotated, and after annotation of every set of unlabeled videos, the network is fine-tuned for 20 epochs with the labeled videos. The process continues until the network reaches 100 epochs. The videos are annotated incrementally and Table I, shows that the annotation time drops after every iteration of annotation and fine-tuning. Because of incremental labeling and fine-tuning, the network learns to extract better features from the videos, which can be better grouped in the t-SNE plot. It is also expected that the oracle spends more time annotating the first few unlabeled set as the network is not yet fine-tuned, and the quality of annotation at the early stage has a significant impact on the next iterations of extracted features.

Epoch	0	20	40	60	80	100
ToA (seconds)	600	552	516	450	240	180

TABLE I: Oracle's time of annotation (ToA) is shown on subset 1 of ActivityNet (v1.3) dataset with 10 classes containing 407 videos (13 hours). At every iteration from 0 to 60, 102 new videos are annotated, and the network is fine-tuned for 20 epochs. From epoch 60 to 100, no new video is added, and the network is fin-tuned on the existing label videos. It can be seen with incremental annotation and fine-tuning the annotation time in the later epochs drops.

Figure 5 shows the video classification accuracy on the test set. It can be seen in Figure 5, annotating the videos with our method can achieve a classification accuracy of 67.2%, which is comparable to the ground truth (blue) accuracy of 69.7%.

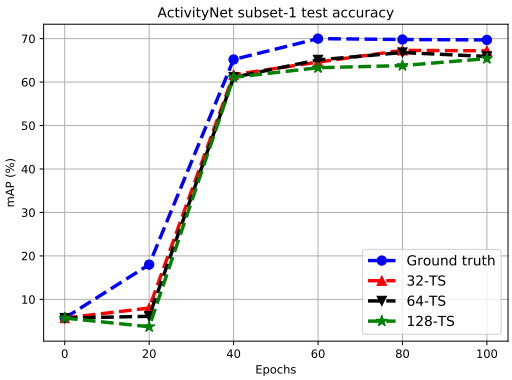


Fig. 5: Comparison of video classification accuracy in form of mAP (%) between fine-tuning the 3D ConvNet on ground truth label versus fine-tuning with our annotation acquired using different time-steps (TS). Fine-tuning the 3D ConvNet on the annotation generated by our method, can achieve comparable video classification accuracy to the ground truth (5% range).

Annotation Speed. In the following we compare our annotation speed with *conventional method* and an open source tool called *MuViLab* [21] on the ActivityNet subset-1 dataset.

Table II shows the result of the comparison between the conventional labeling and MuViLab versus our method in different time-steps. Our method outperforms both techniques on ActivityNet subset-1 in annotation speed by a large margin (4 to 6 times faster). Because most annotation tools do not take advantage of the temporal aspect of the videos at the annotation stage, but our method exploits the spatio-temporal features to place similar actions near each other in the 2D t-SNE plot for the oracle to group-label.

Table III shows, our method can achieve almost the same test accuracy on video classification task when trained on the ground-truth label, but much faster.

	Conventional	**MuViLab**	**Our-32**	**Our-64**	**Our-128**
Time Gain	3 x	4.5 x	18 x	24 x	36 x

TABLE II: Comparison of time gain when annotating with different methods on a subset-1 of ActivityNet containing 769 minutes of video. Our method with 128 time-steps (Ours-128) outperforms conventional and MuViLab [21] methods with labeling 769 minutes of video in 21 minutes.

Increasing Annotation Speed. We further investigate increasing the time-steps in each clip to increase the annotation speed. One way to increase the annotation speed is by putting more videos on the screen for the oracle to annotate. However, since ActivityNet videos on average have 30 frames per second (FPS), every 32 time-steps that we sample represent almost 1 second ($\sim \frac{32}{30}$) of video. Putting all of the 407 videos

(13 hours) overflows the screen with the frames and makes the annotation harder for the oracle. One way to prevent overflowing the figures with thousands of frames is to increase the time-steps for sampling frames for each clip to the point that the network can still preserve the temporal coherency between the clips. This way we can show all of the videos on the 2D plot in less number of points.

The experiments indicate that it is possible to increase the clip time-steps to 128 frames without losing much of the temporal coherency between the clips of the videos. 128 time-steps are almost equal to 4 seconds ($\sim \frac{128}{30}$) of video, which is almost the average length of a mid-range action of ActivityNet.

Figure 5 shows the test accuracy of Resnet-34 3D fine-tuned with ground truth labels, fine-tuned with labels provided by the oracle using our method with three different time-steps. Moreover, Table III shows that incremental annotation and fine-tuning with 32 time-steps (32-TS) gives a very close mAP to the ground truth accuracy. Using 128 time-steps (128-TS) reduces test accuracy while increasing the annotation speed. The decrease in accuracy compared to the 32-TS version is expected since the annotation is more prone to noise when time-step is increased to 128 frames. With 128-TS for each clip, every point in the scatter plot represents 4 seconds of the video while it represents 1 second in the 32-TS version. Which means miss-labeling points in the 128 version comes with more significant consequences in the fine-tuning process. However, Table III shows using 128-TS compared to the 32-TS, it increases the annotation speed twice while the mAP decreases less than 2%.

Method	GT	32-TS	64-TS	128-TS
mAP	69.7 %	67.2 %	65.9 %	65.4 %
ToA (minutes)	-	42	31	21

TABLE III: Comparison of video classification accuracy (mAP) and ToA (time of annotation) on ActivityNet subset-1. This subset contains 407 videos in about 13 hours of video. Our method in 32 time-steps (32-TS) and 128-TS (128-TS) achieves comparable test accuracy to the ground truth accuracy and requires a much shorter time to annotate.

E. Generalization

To further demonstrate the generalization of our method, we conduct the same annotation experiment on a more challenging subset of ActivityNet (v1.3) with 15 classes. Moreover, we evaluate our method on a subset of Sports-1M [3] with 15 random classes. We also test our method in a different setting; we try to annotate the Breakfast dataset [22] which includes complex actions on a sub-activity level and investigate the effect of temporal enhancement module on Breakfast dataset.

ActivityNet (v1.3) Subset-2. Subset 2 of ActivityNet (v1.3) contains 637 training videos and 311 test videos. The first iteration of features is extracted from the 637 training videos and is annotated in 15 minutes by the oracle. After 20 epochs of fine-tuning the ResNet-34 3D, the new features are extracted, and the labels are fine-tuned again by the oracle.

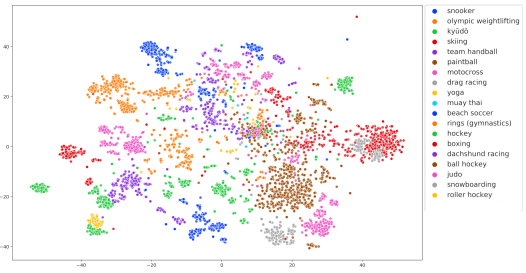


Fig. 6: t-SNE projection of extracted features from 200 videos from the Sports-1M [3] dataset with ground truth labels as colors. 200 videos are from 15 random classes; however, some videos contain more than one activity class. The 4-NN accuracy, which emulates the quality of the projection through measuring local homogeneity, is 92.3%, indicating such a figure is annotate-able by the oracle.

After this stage, the network is fine-tuned for 80 epochs. After fine-tuning for 100 epochs, our method reaches a test accuracy of 66.4%, while the training with ground-truth labels achieves an accuracy of 68.3% on the video classification task.

To emulate the annotation speed in form of local homogeneity, we report KNN accuracy. The 4-NN accuracy of the final features is 92.4%, which shows the quality of the extracted features is enough for the oracle to annotate. This validates our method on the ActivityNet subset-2 and shows our method is not over-fitting on subset-1 of ActivityNet and can still perform well with more fine-grained classes.

Sports-1M. We further validate our method on a subset of Sports-1M [3] dataset with 15 random classes. We randomly sample 200 videos (860 minutes) from the total 702 videos available in the 15 classes. The features are extracted from 200 videos, and ground truth labels of the two-dimensional features can be seen in Figure 6. The 4-NN accuracy is 92.3%, which shows the features can be annotated based on similarity. Using our method, we were able to annotate 860 minutes of video in 28 minutes giving us a time gain of 30.7.

Breakfast. Different subsets of Breakfast are used to evaluate the generalization of our method on complex actions. The first subset is 36 minutes of videos of 3 classes (*cereals, fried egg, and sandwich*) in 5 different kitchens. Figure 8 shows that our method groups similar kitchens together instead of grouping similar actions. The reason is that the Breakfast dataset is a *in-the-wild* dataset in where most of the kitchens have different viewpoints, and the 3D features alone can not separate the actions in the different videos. Therefore, our method can not be used with the same setting as ActivityNet and Sports-1M for sub-action annotation with group labeling in the Breakfast dataset. However, our method can be used for video level annotation, as can be seen in Figure 8 that different

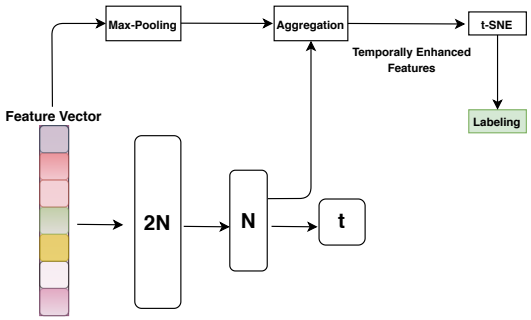


Fig. 7: Temporal feature enhancement. The 3D ConvNet features of a single complex activity *(e.g., frying egg)* are used to train a two-layer MLP to learn the temporal coherency between the sub-actions. After training the MLP, new features are extracted from the second layer of the MLP and aggregated in the form of summation with the max-pooled version of the 3D ConvNet features. Aggregating the features adds temporal coherency to the t-SNE projection of a single video.

actions in the same kitchen can be separated.

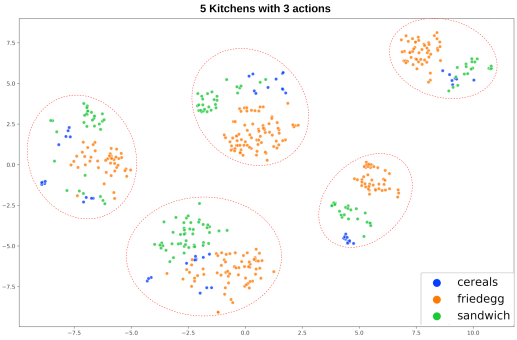


Fig. 8: 2D t-SNE projection of high dimensional features of 3 complex actions in 5 different kitchens with different camera viewpoint. Our method can not place the same actions from different videos near each other and instead place the same kitchens together due to the complexity of the dataset.

Since group labeling, multiple videos containing multiple sub-action in different kitchens is not feasible at a sub-activity level using our method, in a more realistic setting, we aim to annotate the sub-actions in a single video instead of multiple videos. Besides, to exploit the temporal coherency of the sub-actions, we temporally enhance the features using an MLP-based method we adapt from [55]. The MLP is trained on the extracted features of one complex activity (*fried egg*) to learn the ordering of the sub-actions (*putting oil in the pan, cracking egg*) in complex actions. Finally, the extracted features from the second layer of the MLP are aggregated in the form of summation with the 3D ConvNet features. The goal of feature aggregation is to increase the temporal coherency between the ResNet-34 3D features vectors.

It can be seen in Figure 7 that the new features are extracted from the second layer of the MLP. The dimensions of the MLP feature vector is $n \times 128$. Where n is the number of feature vectors from the 3D ConvNet. To aggregate the MLP features with the ResNet features, the ResNet-34 3D features are reduced in spatial dimension using a max pooling kernel and later added to the MLP feature vectors.

Figure 10 shows the effect of using and MLP to temporally enhance the features. The color-coding in the scatter plot represents the relative time of the clips in the video. Lighter points in the scatter plots represent early clips and the darker colors represent later clips in the video. It can be seen in Figure 10-a, there is no temporal ordering of sub-actions in the scatter plot. After temporally enhancing the features using the MLP, Figure 10-b shows that the temporal ordering of clips appears in the scatter plot.

To quantitatively evaluate the effect of the temporal enhancement, in an experiment, a video of *fried egg*, 196 seconds long, is annotated with four different methods using conventional annotation, MuViLab, our method with ResNet-34 3D features (RN-34) and our method with the temporally enhanced features. The maximum annotation budget for annotating the*fried egg* video is 3 minutes. It can be seen in Table IV, temporal enhancement increases the annotation quality in the form of mean over frame (MoF) percentage compared to using only 3D ResNet-34 features. The reason is that the new features are temporally enhanced based of the temporal coherency between the sub-actions in the video. However, our method can not excel MuViLab and conventional annotation techniques for such a dataset in annotation speed and accuracy. The reason is that our method's power is in grouping the same actions from multiple videos, which can not be done on the Breakfast dataset due to the difficulty of the dataset and multiple viewpoints. The qualitative results of using our annotation method compared to ground truth annotation can be seen in Figure 9.

	Conventional	**MuViLab**	**RN-34**	**Time Enhanced**
ToA (minutes)	2.19	2	3	3
MoF	96%	95.2%	52%	63.4%

TABLE IV: Time of annotation (ToA) and mean over frame (MoF) comparison between different annotation methods on one video of the Breakfast dataset. Temporally enhancing the ResNet-34 3D (RN-34) features can increase the mean over frames (MoF) with the same annotation budget (3 minutes) but can not achieve a comparable result with conventional annotation and MuViLab. Conventional annotation and MuViLab can achieve higher MoF in shorter time of annotation.

V. Ablation Study

In this section, we conduct an ablation study to motivate our design choices in the following aspects: (i) dimensionality

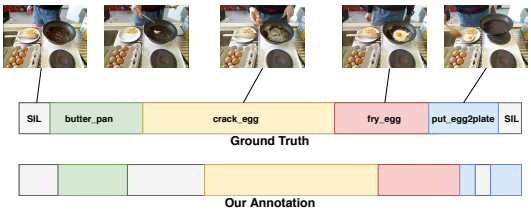


Fig. 9: Qualitative comparison of annotation quality on a video of *fried egg* from the Breakfast dataset [22]. Our annotation method using temporally enhanced features can label the video so that the temporal ordering of sub-actions is still intact.

	4-NN	Exec Time
t-SNE	94%	160
UMAP	83.3%	40

TABLE V: Comparison of quality of projection and execution time (Exec Time) of t-SNE versus UMAP through measuring 4-NN accuracy to estimate local homogeneity. UMAP has a faster execution time than BH-t-SNE but t-SNE can give a better projection with higher local homogeneity which can speed up the annotation process.

reduction method,(ii) t-SNE parameter selection and (iii) 2D versus 3D backbone for feature extraction.

A. *Dimensionality Reduction*

We investigate using PCA as a linear dimensionality method and t-SNE as a non-linear dimensionality method for visualizing the high-dimensional features in two dimensions. We use the extracted feature from the ActivityNet subset-1 with 407 videos. Figure 11-b shows qualitatively that PCA is not able to group similar features and separate unalike features from the videos in the transition to a lower dimension, making the annotation more difficult. However, Figure 11-a, shows that t-SNE projection can maintain the local structure of each class while separating the features from different classes. To report the quality of projection in quantitative measures, we use KNN with K=4. The 4-NN classification accuracy in Figure 11 for the t-SNE projection is 80.6% and for the PCA projection is 58.2%. Therefore, PCA, a linear dimensionality method, cannot reduce the feature dimension while placing similar classes near each other.

We further compare our result with UMAP [46] versus Barnes-Hut t-SNE [56] for ActivityNet subset-1. It can be seen in Table V, UMAP offers a faster computation time than Barnes-Hut t-SNE which is not an issue since faster implementations of t-SNE are available such as [47], [59].

B. *t-SNE Parameters*

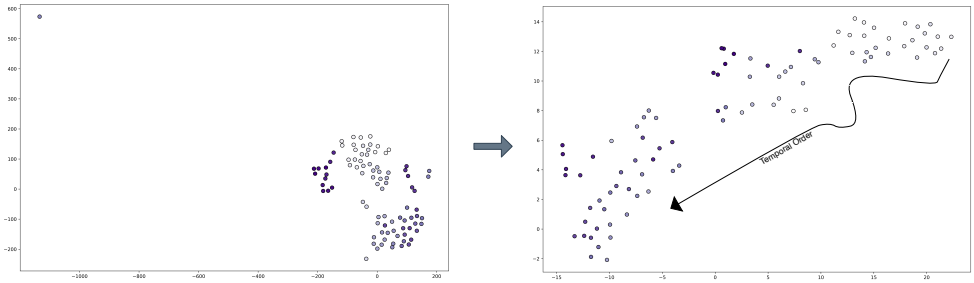
We investigate using different perplexity parameters for the t-SNE projection. [43] recommend using perplexity parameter between [5-50], however larger and denser datasets requires relatively higher perplexity. With low perplexity, the local structure of data in each video dominates the action grouping from multiple video [44], but our goal is to group multiple actions from different videos. To emulate the t-SNE projection quality for the annotation, we report homogeneity and completeness scores with different perplexities in Table VI. As Table VI shows, perplexity 30 shows the highest homogeneity and completeness scores meaning that t-SNE projection with perplexity 30 can separate the classes in a better way than projecting with the other perplexity parameters. Therefore, using t-SNE with perplexity 30 makes the group labeling process easier for the oracle.

	px-5	px-15	px-30	px-50	px-100	px-120
Homogeneity	44.7%	58.7%	62.5%	61.3%	61.7%	61.5%
Completeness	42.5%	56.1%	60%	58.5%	59%	58.8%

TABLE VI: Comparison of homogeneity and completeness scores as a measure to emulate the quality of t-SNE projection in a global-level. Higher homogeneity means all the points in a cluster belongs to the same class. Higher completeness means all the points belonging to a class are in the same cluster. Perplexity 30 gives the highest homogeneity and completeness score and is used for the t-SNE projection.

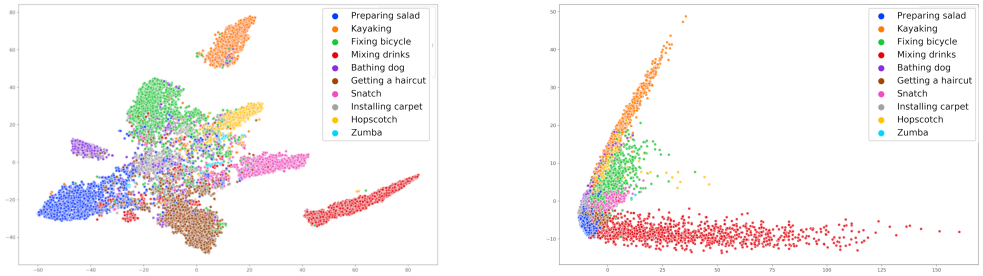
C. *2D-3D Comparison*

We investigate replacing the 3D ConvNet with a 2D CNN to compare the quality of the feature embedding. For 3D ConvNet, ResNet-34 3D pre-trained on Kinetics [2] and for the 2D CNN ResNet-50 pre-trained on Kinetics [2] are used. The reason we chose Resnet-50 instead of Resnet-34 for the 2D CNN was that the Kinetics pre-trained weights was only available for ResNet-50. To conduct the experiment, we sample every 32 consecutive frames (time-steps) as a clip in the 3D ConvNet, and for the 2D CNN, we choose one frame for every 32 frames to represent that specific window. The experiment is done on the subset-1 of the Activity-Net dataset with 10 classes. It can be seen in Figure 12 that we start the experiments with 32 time-steps. Later we gradually increase the time-steps to the point that either we lose temporal information or we run out of data-points (whichever comes first). With 32 time-steps, Figure 12, we can see the 2D CNN can capture the same action in different videos but can not place them together as well as the 3D ConvNet. Therefore the colors representing the classes are better gathered in a close proximity in the 3D ConvNet which makes the annotation process faster than the 2D CNN projection. Moreover, Figure 12 shows, by increasing the time-steps for frame sampling, the 2D CNN, even with deeper architecture, 50 compared to 34, starts losing the temporal coherency between the data-points as it only focuses on the spatial information between the frames. Focusing only on spatial information can still work in lower time-steps (32-TS) because the frames from the same action contain similar spatial information. However, using spatial information alone becomes problematic in higher time-steps as



a) ResNet-34 3D Features b) Temporally Enhanced Features

Fig. 10: Color-coded t-SNE projection of a complex action video from the Breakfast dataset [22]. Lighter colors belong to the early clip in the video, and darker colors belong to the videos later in the video. a) Shows the t-SNE projection of the ResNet34 3D features, which shows the temporal coherency between the sub-actions is lost. b) Shows the t-SNE projection of the temporally enhance features. It can be seen in b, the MLP can add the temporal coherency to the features in low dimensional space which make annotating the complex action in a sub-activity level easier.



a) t-SNE projection b) PCA projection

Fig. 11: Visual comparison of the projection quality of high-dimensional features to two dimensions using t-SNE (a) and PCA (b). Linear DR methods such as PCA are unable to maintain the structure of the high-dimensional data in two dimensions.

increasing the time-steps reduces the spatial similarity between the frames.

Moreover, to evaluate our findings quantitatively, we use KNN accuracy as a quantitative emulation for the quality of features for annotation. Table VII shows that increasing the number of frames in the clips, degrades the 4-NN accuracy. Therefore, the local homogeneity decreases more drastically in 2D CNNs compared to 3D CNNs, which makes annotation more difficult for the oracle. In other words, the 2D CNN alone can not maintain the temporal structure of the data in higher time-steps. Therefore, in our method, 3D features are extracted to be used for group labeling.

	2D CNN	3D CNN
32-TS	93.1 %	100 %
64-TS	89.3 %	97.6 %
128-TS	74.6 %	95.2 %

TABLE VII: Comparison of 4-NN accuracy of extracted features from a 2D CNN (ResNet-50) and a 3D ConvNet (ResNet-34 3D) on subset-1 of ActivityNet [1]. Increasing time-steps cause the 2D CNN to lose the spatial similarity between the frames and fail to group them in the t-SNE plot while the 3D ConvNet can still group similar actions even in higher time-steps.

VI. Conclusion

In this work, we introduced a smart annotation tool that can help the oracle group label the videos based on their latent space feature similarity in two-dimensional space. The method can be useful in annotating large-scale video datasets, especially if the annotation budget and time are limited. Our

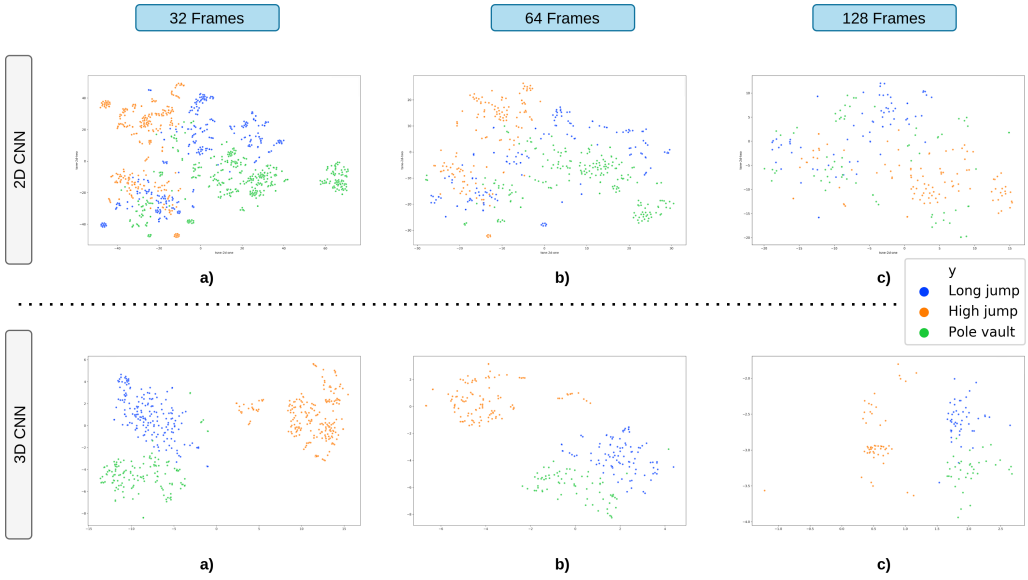


Fig. 12: Comparison of t-SNE projection of extracted features from a 2D CNN versus a 3D ConvNet for videos from 3 action classes of ActivityNet dataset [1]. Increasing the time-steps for sampling clips from the videos causes the 2D CNN to lose the clips' spatial information. However, the features from the 3D ConvNet can maintain the coherency between the clips as they take time into account in the convolution kernels.

method can outperform conventional annotation and MuViLab [21] time-wise in order of magnitude. However, we showed that our technique could not achieve the same result on a complex action dataset such as Breakfast [22]. We speculate that the reason is the number of multiple challenges in the Breakfast dataset. These challenges include multiple viewpoints, temporal variation in the sub-actions, and spatial similarity of sub-actions, which makes using feature similarity for action gathering a challenging task. In a more straightforward problem to annotate one video, we adapted a temporal enhancement method from [55] to increase the temporal coherency between the features from the sub-actions. The temporal feature enhancement helps the annotation speed compared to using ResNet-34 3D features only; however, it can not outperform the time and quality of annotation in MuViLab and conventional annotation methods. For future work, we expect using a viewpoint invariant network in combination with a way of capturing the local and global dynamics of the actions, e.g. *Multi-scale Temporal Convolutions*, can help group labeling complex action datasets at a sub-activity level.

References

[1] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–970, 2015.

[2] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *CoRR*, vol. abs/1705.06950, 2017.

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[4] K. Liu, W. Liu, C. Gan, M. Tan, and H. Ma, "T-c3d: Temporal convolutional 3d network for real-time action recognition," in *AAAI*, 2018.

[5] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3D: generic features for video analysis," *CoRR*, vol. abs/1412.0767, 2014.

[6] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," *CoRR*, vol. abs/1608.00859, 2016.

[7] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "A simple baseline for multi-object tracking," 2020.

[8] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," *CoRR*, vol. abs/1812.05050, 2018.

[9] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," *CoRR*, vol. abs/1808.06048, 2018.

[10] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," *CoRR*, vol. abs/1801.04264, 2018.

[11] T. N. Nguyen and J. Meunier, "Anomaly detection in video sequence with appearance-motion correspondence," 2019.

[12] T. N. Nguyen and J. Meunier, "Anomaly detection in video sequence with appearance-motion correspondence," 2019.

[13] A. Diba, M. Fayyaz, V. Sharma, A. H. Karami, M. M. Arzani, R. Yousefzadeh, and L. V. Gool, "Temporal 3d convnets: New architecture and transfer learning for video classification," *CoRR*, vol. abs/1711.08200, 2017.

[14] U. Ahsan, C. Sun, and I. A. Essa, "Discrimnet: Semi-supervised action recognition from videos using generative adversarial networks," *CoRR*, vol. abs/1801.07230, 2018.

[15] M. Zeng, T. Yu, X. Wang, L. T. Nguyen, O. J. Mengshoel, and I. Lane, "Semi-supervised convolutional neural networks for human activity recognition," *CoRR*, vol. abs/1801.07827, 2018.

[16] U. Ahsan, C. Sun, and I. A. Essa, "Discrimnet: Semi-supervised action recognition from videos using generative adversarial networks," *CoRR*, vol. abs/1801.07230, 2018.

[17] R. Girdhar, D. Tran, L. Torresani, and D. Ramanan, "Distinit: Learning video representations without a single labeled video," *CoRR*, vol. abs/1901.09244, 2019.

[18] D. Ghadiyaram, M. Feiszli, D. Tran, X. Yan, H. Wang, and D. Mahajan, "Large-scale weakly-supervised pre-training for video action recognition," *CoRR*, vol. abs/1905.00561, 2019.

[19] B. Fernando, H. Bilen, E. Gavves, and S. Gould, "Self-supervised video representation learning with odd-one-out networks," *CoRR*, vol. abs/1611.06646, 2016.

[20] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The EPIC-KITCHENS dataset," *CoRR*, vol. abs/1804.02748, 2018.

[21] L. D. Alessandro Masullo, "Muvilab." https://github.com/ale152/muvilab, 2019.

[22] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 780–787, 2014.

[23] N. Hussein, E. Gavves, and A. W. M. Smeulders, "Timeception for complex action recognition," *CoRR*, vol. abs/1812.01289, 2018.

[24] A. Kläser, M. Marszalek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *BMVC*, 2008.

[25] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, (New York, NY, USA), p. 357–360, Association for Computing Machinery, 2007.

[26] L. Sevilla-Lara, Y. Liao, F. Güney, V. Jampani, A. Geiger, and M. J. Black, "On the integration of optical flow and action recognition," *CoRR*, vol. abs/1712.08416, 2017.

[27] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *2013 IEEE International Conference on Computer Vision*, pp. 3551–3558, 2013.

[28] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *CoRR*, vol. abs/1406.2199, 2014.

[29] Z. Xu, Y. Yang, and A. G. Hauptmann, "A discriminative CNN video representation for event detection," *CoRR*, vol. abs/1411.4006, 2014.

[30] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," in *Proceedings of (CVPR) Computer Vision and Pattern Recognition*, pp. 3165 – 3174, July 2017.

[31] J. Carreira, V. Patraucean, L. Mazaré, A. Zisserman, and S. Osindero, "Massively parallel video networks," *CoRR*, vol. abs/1806.03863, 2018.

[32] Z. Shou, D. Wang, and S. Chang, "Action temporal localization in untrimmed videos via multi-stage cnns," *CoRR*, vol. abs/1601.02129, 2016.

[33] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," *CoRR*, vol. abs/1705.07750, 2017.

[34] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[35] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, "Video action transformer network," *CoRR*, vol. abs/1812.02707, 2018.

[36] B. Martinez, D. Modolo, Y. Xiong, and J. Tighe, "Action recognition with spatial-temporal discriminative filter banks," 2019.

[37] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," *CoRR*, vol. abs/1711.10305, 2017.

[38] D. Tran, H. Wang, L. Torresani, and M. Feiszli, "Video classification with channel-separated convolutional networks," *CoRR*, vol. abs/1904.02811, 2019.

[39] J. Carreira, V. Patraucean, L. Mazaré, A. Zisserman, and S. Osindero, "Massively parallel video networks," *CoRR*, vol. abs/1806.03863, 2018.

[40] K. Hara, H. Kataoka, and Y. Satoh, "Learning spatio-temporal features with 3d residual networks for action recognition," *CoRR*, vol. abs/1708.07632, 2017.

[41] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Springer, 2007.

[42] A. Chatzimparmpas, R. M. Martins, and A. Kerren, "t-visne: Interactive assessment and interpretation of t-sne projections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, p. 2696–2714, Aug 2020.

[43] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. nov, pp. 2579–2605, 2008. Pagination: 27.

[44] M. Wattenberg, F. Viégas, and I. Johnson, "How to use t-sne effectively," 2016.

[45] G. C. Linderman and S. Steinerberger, "Clustering with t-sne, provably," *CoRR*, vol. abs/1706.02582, 2017.

[46] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2018.

[47] N. Pezzotti, J. Thijssen, A. Mordvintsev, T. Höllt, B. v. Lew, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "Gpgpu linear complexity t-sne optimization," *IEEE Transactions on Visualization and Computer Graphics (Proceedings of VAST 2019)*, vol. 26, no. 1, 2020.

[48] o. c. darrenl, "labelimg." https://github.com/tzutalin/labelImg, 2017.

[49] o. c. Amit K Gupta, "imglab." https://github.com/NaturalIntelligence/imglab, 2017.

[50] J. Li and J. Z. Wang, "Real-time computerized annotation of pictures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 985–1002, 2008.

[51] Gang Wang and D. Forsyth, "Joint learning of visual attributes, object classes and visual saliency," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 537–544, 2009.

[52] J. Wu, Y. Yu, C. Huang, and K. Yu, "Deep multiple instance learning for image classification and auto-annotation," June 2015.

[53] F. P. S. Luus, N. Khan, and I. Akhalwaya, "Active learning with tensorboard projector," *CoRR*, vol. abs/1901.00675, 2019.

[54] A. Dutta and A. Zisserman, "The via annotation software for images, audio and video," in *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, (New York, NY, USA), p. 2276–2279, Association for Computing Machinery, 2019.

[55] A. Kukleva, H. Kuehne, F. Sener, and J. Gall, "Unsupervised learning of action classes with continuous temporal embedding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*, 2019.

[56] L. van der Maaten, "Barnes-hut-sne," 2013.

[57] L. van der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 93, pp. 3221–3245, 2014.

[58] 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*, vol. 12, pp. 2825–2830, 2011.

[59] D. M. Chan, R. Rao, F. Huang, and J. F. Canny, "t-sne-cuda: Gpu-accelerated t-sne and its applications to modern data," 2018.

2

INTRODUCTION

With the introduction of large-scale video dataset such as [1, 2], and increase in the computational power of GPUs video understanding became an attractive topic of research in the past few years. Different supervised methods, such as C3D [3], T3D [4], TSN [5], have improved the accuracy on the video understanding tasks such as video classification, temporal segmentation, etc. However, all the mentioned models are fully-supervised, and supervised models require much annotated data that is unavailable as videos are naturally unlabeled, and annotating them takes much effort.

Some works have been done to decrease the dependency on the quality and amount of annotated data. [6, 7] investigated pre-training features with large-scale internet videos with noisy labels in a weakly supervised manner. Later the model has been fine-tuned on the target dataset in a supervised manner. [8] proposes a self-supervised method where the frame order in unlabeled videos can help the learning process. Other methods tried to train the networks for action recognition in a semi-supervised [9] [10] manner without having the full labels. [11] proposed UntrimmedNets using video-level annotation with attention mechanisms instead of temporal labels for the task of temporal action recognition. However, these methods were unable to achieve higher accuracy compared to supervised models.

The main ways that the research community annotate such datasets are 1- Outsourcing it to the annotation companies 2- Annotating the videos using conventional methods (with no specific tools) 3- Annotating the video dataset using available software such as MuViLab [12]. The first method usually requires proper funding. (Charades [13] authors spent 1 USD per video for annotation). The second and third methods are time consuming since it is not efficient to label hundreds of hours of video data manually. Most of these methods do not take advantage of the structure and similarity of the data.
In this work, we introduce a framework that can speed up the video annotation up to 36 times faster than the mentioned annotation tools.

REFERENCES

[1] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, *Activitynet: A large-scale video benchmark for human activity understanding,* in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015) pp. 961–970.

[2] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, *The kinetics human action video dataset,* CoRR **abs/1705.06950** (2017), arXiv:1705.06950 .

[3] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, *C3D: generic features for video analysis,* CoRR **abs/1412.0767** (2014), arXiv:1412.0767 .

[4] A. Diba, M. Fayyaz, V. Sharma, A. H. Karami, M. M. Arzani, R. Yousefzadeh, and L. V. Gool, *Temporal 3d convnets: New architecture and transfer learning for video classification,* CoRR **abs/1711.08200** (2017), arXiv:1711.08200 .

[5] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, *Temporal segment networks: Towards good practices for deep action recognition,* CoRR **abs/1608.00859** (2016), arXiv:1608.00859 .

[6] R. Girdhar, D. Tran, L. Torresani, and D. Ramanan, *Distinit: Learning video representations without a single labeled video,* CoRR **abs/1901.09244** (2019), arXiv:1901.09244 .

[7] D. Ghadiyaram, M. Feiszli, D. Tran, X. Yan, H. Wang, and D. Mahajan, *Large-scale weakly-supervised pre-training for video action recognition,* CoRR **abs/1905.00561** (2019), arXiv:1905.00561 .

[8] B. Fernando, H. Bilen, E. Gavves, and S. Gould, *Self-supervised video representation learning with odd-one-out networks,* CoRR **abs/1611.06646** (2016), arXiv:1611.06646 .

[9] U. Ahsan, C. Sun, and I. A. Essa, *Discrimnet: Semi-supervised action recognition from videos using generative adversarial networks,* CoRR **abs/1801.07230** (2018), arXiv:1801.07230 .

[10] M. Zeng, T. Yu, X. Wang, L. T. Nguyen, O. J. Mengshoel, and I. Lane, *Semi-supervised convolutional neural networks for human activity recognition,* CoRR **abs/1801.07827** (2018), arXiv:1801.07827 .

[11] L. Wang, Y. Xiong, D. Lin, and L. V. Gool, *Untrimmednets for weakly supervised action recognition and detection,* CoRR **abs/1703.03329** (2017), arXiv:1703.03329 .

[12] L. D. Alessandro Masullo, *Muvilab,* `https://github.com/ale152/muvilab` (2019).

[13] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, *Hollywood in homes: Crowdsourcing data collection for activity understanding,* CoRR **abs/1604.01753** (2016), arXiv:1604.01753 .

3

BACKGROUND ON DEEP LEARNING

Deep learning is a subset of machine learning that has been used in a wide variety of tasks such as object detection [1], activity recognition [2], person re-identification [3], etc. Feed forward neural networks or multi-layer perceptrons (MLP) are an essential ports of the deep learning models. If we consider the feed forward model as a module, the goal of this module is to approximate a function f^* that can produce an output $y^* = f^*(x)$ where y^* should be close to the actual output y [4]. A MLP can learn such a function that can map an input x to a category $y = f(x;\theta)$ to come up with the closest approximation to the reality $y = y^*$. θ is a set of learn-able parameters for each network based on the network architecture. Feed-forward networks are the basis of the deep learning architectures. Most of the other models such as convolutions neural networks (CNN), recurrent neural networks (RNN), etc. are a variation of feed-forward networks which are widely used in image recognition and natural language processing (NLP) tasks respectively.

In the following sections Neural Networks, CNNs, and tips and tricks for a better approximation function are explained in more detail.

3.1. NEURAL NETWORKS

3.1.1. SINGLE PERCEPTRON

Neural networks(NN) are called *neural* because they were inspired by neuroscience [4]. A single-layer NN is called a perceptron, which can be seen in Figure 3.1. The input vector is shown with $X = x_0, ..., x_n$, which are multiplied by the weight vector $W = w_0, ..., w_n$ in a forward pass. The value of weighted inputs is later added to a bias term and is fed into an activation function *f*. An activation function is an essential tool to a perceptron since it adds non-linearity to the summation of inputs, which enables a perceptron to learn complicated functions. Some of the most used activation functions are Sigmoid and Relu 3.2. An overview of different activation functions are given in [5].

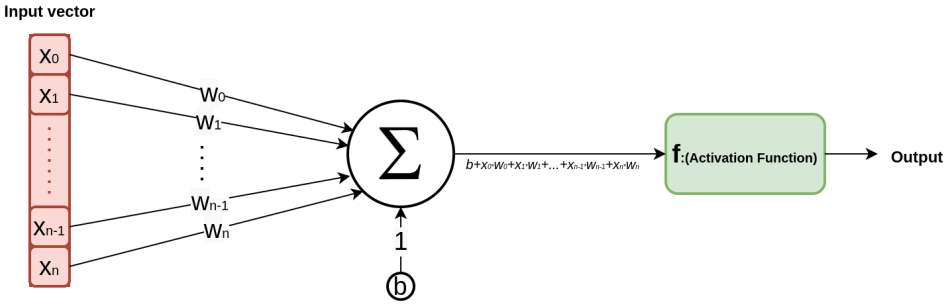


Figure 3.1: Single perceptron that takes an input vector, calculates the weighted sum and feeds it into an activation function to add non-linearity.

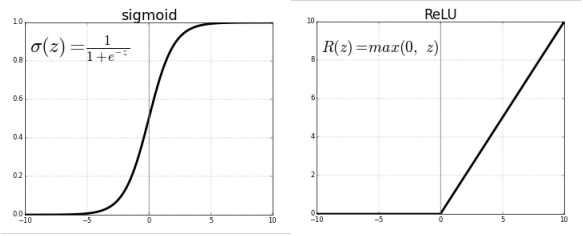


Figure 3.2: Two most used activation functions.

3.1.2. MULTI-LAYER PERCEPTRON

One can increase the complexity of the NN by adding more perceptrons to a single layer (making the network wider) or adding more layers (making the network deeper) to make a multi-layer perceptron (MLP). MLPs are also called feed-forward *networks* because they usually are a combination of multiple approximation functions $y^* = f^n(f^{n-1}(...f^1(x)))$ that form a network [4] where f^1 is called the first layer and f^n is called the n-th layer of the neural network. n indicates the depth of the model, which is a hyper-parameter that needs to be carefully selected. During the training, learn-able parameters in the layers are pushed so that the network output y* matches the desired output y. Figure 3.3 shows a MLP with 1 hidden layer with 4 neurons. The output y^* is calculated as follows:

$$\vec{h} = w_1 * \vec{x} + b_1 \tag{3.1}$$

$$\vec{y*} = w_2 * \vec{h} + b_2 \tag{3.2}$$

3.1.3. TRAINING

The process of training for an MLP refers to finding the best learn-able parameters for the weights and biases. These parameters are usually randomly initialized at the beginning and later updated during the training process to minimize a loss function. Different loss

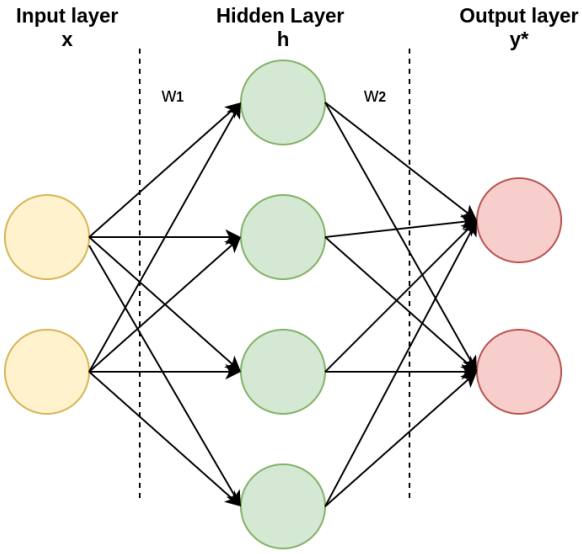


Figure 3.3: Fully connected artificial neural network (ANN) with one hidden layer.

functions have been used in different settings [6]. One of the most used functions is mean squared error (MSE), which is shown in Equation 3.3.

$$MSE = \frac{1}{n}\sum_{i=1}^{N}(y_i - y_i^*)^2 \tag{3.3}$$

The loss is defined based on the difference between the target output and the NN output. Methods such as gradient descents are used to solve this optimization problem by finding the best parameters to reduce the loss function. In deeper architectures to update the weights in the earlier layers *Backpropagation* is used. The name refers to the fact that the calculation of the gradients start from the output layer and proceeds backward to the first layer. Some parts of each gradient calculation is used from one layer to another. Therefore, backpropagation is a more efficient approach rather than calculating the gradients of each layer separately [7].

3.2. CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNN) are a specific type of NNs that use more for processing data with a grid structure such as image (2D grid) or video (3D grid). As the name indicates, CNNs use convolution operations in at least one of their hidden layers [4]. As can be seen in Figure 3.4, deep CNNs are usually a cascade of multiple convulsions and pooling layers and non-linearity. Convolution filters in different sizes slide over the image to extract feature maps, and later pooling layers are used to disregard the non-useful information and reduce the spatial information to what is important for classifying an image. Moreover, activation functions such as RELU are used to add non-linearity to the structure of the network.

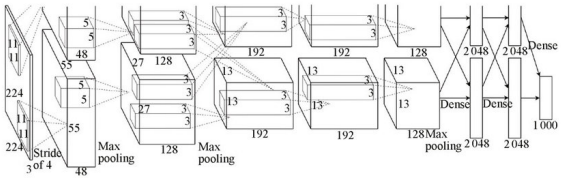


Figure 3.4: Architecture of AlexNet [8], a breakthrough for ImageNet [9] classification

Later in this section, various parts of a CNN and the training process are discussed in more detail.

3.2.1. CONVOLUTIONS

Convolutions are used to extract useful information from the input data. Different filters can be chosen to apply the convolution operating on it based on the need. For example, in image processing, some filters are designed to extract the edges from the images. The weights of the filters in CNN are learn-able parameters which are randomly initialized at the beginning and later updated during the training process to minimize a loss function.

CNNs have some advantages over MLPs, which make them a better choice for certain applications like image and video processing. CNNs are translation invariant, which means changing the location of an object in an image does not affect the output after applying the convolution layer. Therefore, one filter can be used to detect a certain object in an image regardless of the object's position. In deep learning applications, convolutions are multiplications and additions. As can be seen in the Figure 3.5 the 3*3 kernel with weights [[-1, 0, 1],[-2, 0, 2], [-1, 0, 1]] slides on the input to generate the feature map on the right side of Figure 3.5. In the first layers of the CNN, the filters usually try to learn general structures such as edges and lines, etc. Deeper in the network, the filters learn complicated structures.

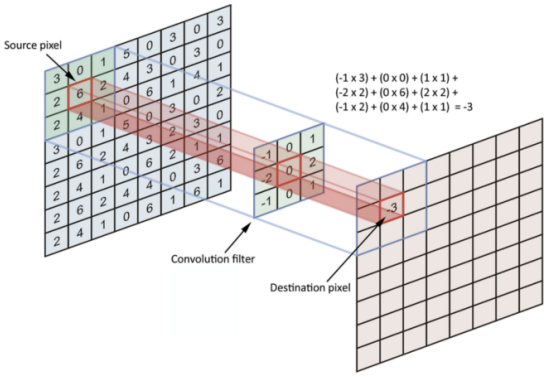


Figure 3.5: Demonstration of applying a two-dimensional convolution filter [10]

3.2.2. POOLING LAYER

After feeding the input to the convolution layers, we get the linear activation maps. The linear activation is later fed into the non-linear activation functions such as RELU. Further, the output of the activation function is fed to a layer called pooling. Different pooling operations, *Max Pooling, Average Pooling*, is used to decrease the spatial resolution of the input and to capture the collective information in a neighborhood of the input. Similar to the convolution operation, pooling is done using a sliding window on top of the feature map. For a pooling operation, the stride of the window determines how much the spatial resolution will be reduced. Max Pooling and Average Pooling outputs the maximum and average value of the k*k window, respectively. Figure 3.6 shows a summary of all mentioned layers.

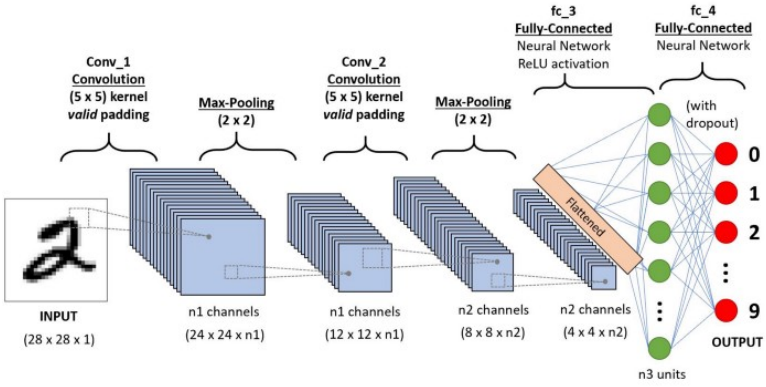


Figure 3.6: Example of feeding an input image to a convolutional neural network [10].

Some network architectures are fully convolutions [11]. Some networks, such as [8] used the convolutions network as a feature extractor and connected a fully connected layer at the end for the classification of the features. A function such as *Softmax* can be used on the output of the MLP to produce class probabilities for the input.

3.2.3. REGULARIZATION

The CNN can be trained in the same way the MLP training was explained in Chapter 2 and it should generalize well on the unseen data. Otherwise, a model that performs wells on the training data but fails to perform well on the validation data is called to be over-fitting on the training data. Many parameters can cause over-fitting of the NN, such as choosing the wrong hyper-parameters or small size of training data. More complex networks with deeper architectures have high representational capabilities and need more training data for a successful approximation function. Therefore if providing more training data for the model is not possible, it makes sense to reduce the complexity of the network. Regularization techniques add a penalization factor that forces the model to learn a more generalize-able representation of the training data. Some of the different regularisation methods are mentioned in the following sections.

EARLY STOPPING

Early stopping refers to the process of stopping the training before the network gets close to overfitting. This can be done by comparing the accuracy of the training set and the validation set every few epochs.

DROPOUT

As can be seen in Figure 3.7, the main idea behind dropout is that at every iteration of training with a probability p, a certain amount of the neurons are ignored (dropped) [12]. Dropping random neurons force the surviving ones to learn a mapping from the input to the target output using a smaller architecture.

More regularisation methods have been discussed in more detail in [4].

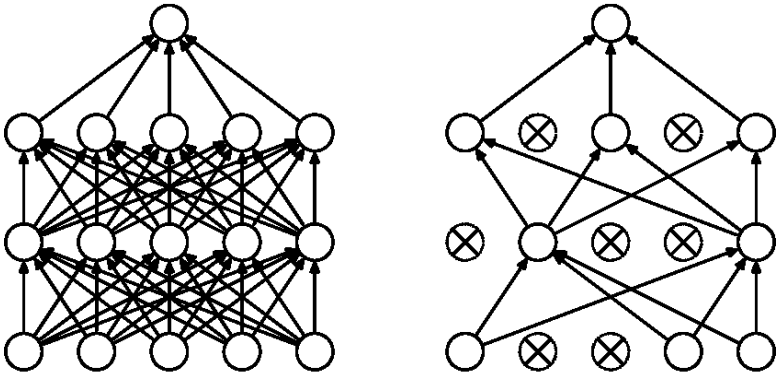


Figure 3.7: Left: Fully connect NN architecture Right: NN with dropout some random neurons [12]

REFERENCES

[1] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement,* CoRR **abs/1804.02767** (2018), arXiv:1804.02767 .

[2] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, *C3D: generic features for video analysis,* CoRR **abs/1412.0767** (2014), arXiv:1412.0767 .

[3] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi, *Deep learning for person re-identification: A survey and outlook,* (2020), arXiv:2001.04193 [cs.CV] .

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) http://www.deeplearningbook.org.

[5] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, *Activation functions: Comparison of trends in practice and research for deep learning,* CoRR **abs/1811.03378** (2018), arXiv:1811.03378 .

[6] K. Janocha and W. M. Czarnecki, *On loss functions for deep neural networks in classification,* CoRR **abs/1702.05659** (2017), arXiv:1702.05659 .

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation applied to handwritten zip code recognition,* Neural Computation **1**, 541 (1989).

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks,* Commun. ACM **60**, 84–90 (2017).

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database,* in *CVPR09* (2009).

[10] *Simple introduction to convolutional neural networks,* https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac, date: 08.08.2020.

[11] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation,* CoRR **abs/1411.4038** (2014), arXiv:1411.4038 .

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting,* J. Mach. Learn. Res. **15**, 1929–1958 (2014).

4

VIDEO UNDERSTANDING

The goal of this chapter is to introduce the reader with 3D convolution and ReNet3D feature extractor.

4.1. INTRODUCTION

Video understanding has been under development in the past few years. With large GPUs and enough data, video understanding has been able to achieve good results in applications such as [1] and group activity recognition [2]. In the past the focus was on the use of specific hand-designed features such as HOG3D [3] SIFT-3D [4], optical flow [5] and iDT [6]. Among these methods, iDT and optical flow have been considered state of the art in hand-designed featured and are still being used in combination with CNNs in different architectures such as two-stream networks [7]. Later many attempts have been made to use 2D CNNs and extract features from video frames and combine the extracted features with some temporal integration function [8, 9]. [10] gives an analysis of how to integrate temporal information of video frames with different fusion methods (e.g. "late fusion", "slow fusion", etc.) by using temporal convolutions on top of the spatial convolutions.

With the introduction of 3D convolution [11, 12] in CNNs which extends the 2D CNNs in temporal dimension showed promising results in the task of action recognition in large-scale video datasets. At the time of writing this paper using 3D CNNs in different variations such as single stream and multiple-stream are state of the art in the task of video understanding [13–19]. However some of these methods use computationally expensive features such as optical flow and iDT. Some other works has also been done to make these approaches fast in training and inference such as [16, 20].

4.2. DATASETS

Introduction of large scale video datasets such as ActivityNet [21], Kinetics 400 [22], Breakfast [23] and many other datasets for different tasks with variation in length of action, type of activity, etc. enabled a great acceleration in research for video understanding tasks. Two datasets that have been used in our research paper are ActivityNet and Breakfast dataset.

ActivityNet is an untrimmed video dataset with a wide range of complex human activities. It comprises 203 classes with an average of 137 untrimmed videos per class in about 849 hours of video. ActivityNet is used as a benchmark dataset for different video understanding tasks such as untrimmed video classification, trimmed activity classification, and activity detection. In our research paper, untrimmed video classification accuracy has been used as a metric to compare the classification result obtained with our method vs. the ground truth labels.

Breakfast is a dataset for human cooking activities from multiple cameras and viewpoints. It contains 1712 videos with an average length of 2.3 minutes per video. It contains 12 breakfast preparation activities. Each video represents only one activity class, but it contains sub-actions such as "Getting the bowl" under "Making cereal." Temporal annotation for each sub-action has been provided. In this work, we qualitatively show how our method can help to label this dataset and how temporal feature enhancement can help the process.

4.3. METHODS

In this section, we focus more on the task of activity recognition in videos. Action recognition is the process of detecting specific actions from consecutive frames in a video where there is also some background (non-action) parts in the video. In our research paper 3D convolutions are used from a C3D network [12] to extract Spatio-temporal features from videos. A more in-depth analysis of C3D and Resnet-3D is given in the following section.

4.3.1. C3D

3D convolution extends 2D convolutions among the video's temporal aspect, enabling the network to model temporal information better than 2D CNNs. Figure 4.1 shows the difference between a 2D convolution vs a 3D convolution. 2D convolution while applied on a single or multiple frames output and image and therefore lost the temporal information of the video [12]. Their results also show that using the C3D with iDT features and a linear classifier yields the highest classification results.

Later other researches such as [24] extended the work from the original C3D paper [12] by using the Resnet architecture and replacing the 2D convolution and pooling layers with their respective 3D version. 3D residual blocks are based on the 2D version from [25] that bypasses the input from one layer to the next. Table 4.2 shows the proposed architecture of the 3D ResNet.

The input clips' size is 16*3*112*112, meaning that after reducing the resolution to

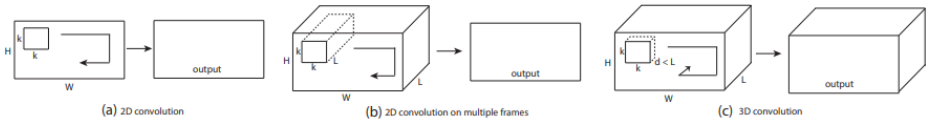


Figure 4.1: Comparison of different convolution kernels [12]

Layer Name	Architecture	
	18-layer	34-layer
conv1	$7 \times 7 \times 7, 64$, stride 1 (T), 2 (XY)	
conv2_x	$3 \times 3 \times 3$ max pool, stride 2 $\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 3$
average pool, 400-d fc, softmax		

Figure 4.2: 3D ResNet architecture [12]

112*112, 16 consecutive frames are used as an input clip to the 3D network. After training, the ResNet-34 3D shows better accuracy than the original C3D on the video classification task on the Kinetics dataset.

The acceptable accuracy of video classification and the ease of feature extraction compared to two-stream methods caused us to use Resnet-34 3D as the feature extractor for our research.

4.3.2. I3D

Two-Stream Inflated 3D ConvNet (I3D) [26] unlike C3D [12] use two-stream networks to achieve a higher accuracy in action recognition. I3D uses the structure of well-studied 2D CNN by inflating the 2D pooling and kernels to their 3D version by adding an extra temporal dimension. They also show that 3D models can be pre-trained on 2D datasets like ImageNet by repeatedly copying a single frame and making a video sequence out of it. As can be seen in Figure 4.3 I3D uses an ImageNet pre-trained 3D ConvNet and a 3D ConvNet for the optical flow features, which are trained separately and average their prediction in the testing phase.

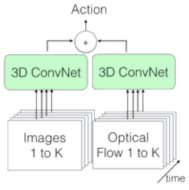


Figure 4.3: Two-stream 3D ConvNet [26]

4.3.3. T-C3D

T-C3D [27] claims that C3D [24] is not capable to model long-term features in the videos where actions take more time to unfold (High jump, Long jump, etc.) and extends the work done on the 3D ConvNets with adding a temporal encoding method to capture the features across the video.

As can be seen in Figure 4.4 after transforming the video into smaller clips and extracting the short-term temporal motion. Later the Spatio-temporal features are fed into the temporal encoding module to model the long-range motion across the entire video. The temporal encoding module aggregates the features with functions such as (average pooling, max pooling, etc.) computed from different clips of the video.

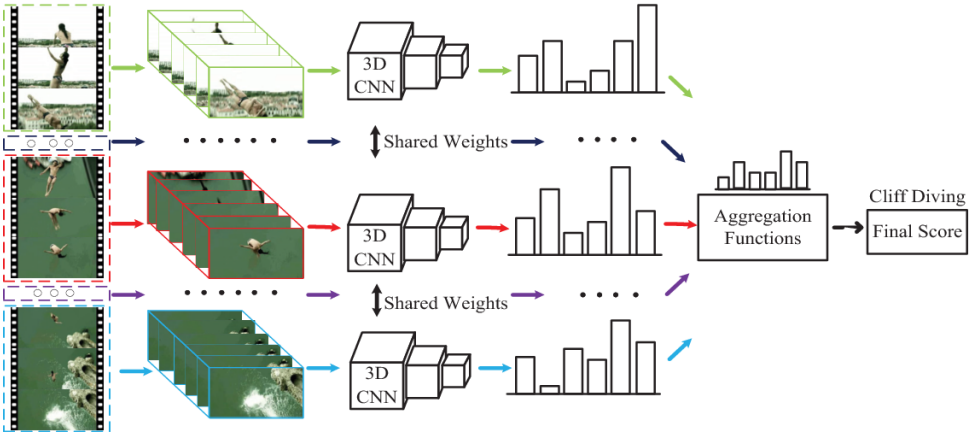


Figure 4.4: Encoding temporal information for long-range motion across the video with feature aggregation [27]

REFERENCES

[1] Q. Rao and J. Frtunikj, *Deep learning for self-driving cars: Chances and challenges,* in *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)* (2018) pp. 35–38.

[2] K. Gavrilyuk, R. Sanford, M. Javan, and C. G. M. Snoek, *Actor-transformers for group activity recognition,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[3] A. Kläser, M. Marszalek, and C. Schmid, *A spatio-temporal descriptor based on 3d-gradients,* in *BMVC* (2008).

[4] P. Scovanner, S. Ali, and M. Shah, *A 3-dimensional sift descriptor and its application to action recognition,* in *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07 (Association for Computing Machinery, New York, NY, USA, 2007) p. 357–360.

[5] L. Sevilla-Lara, Y. Liao, F. Güney, V. Jampani, A. Geiger, and M. J. Black, *On the integration of optical flow and action recognition,* CoRR **abs/1712.08416** (2017), arXiv:1712.08416 .

[6] H. Wang and C. Schmid, *Action recognition with improved trajectories,* in *2013 IEEE International Conference on Computer Vision* (2013) pp. 3551–3558.

[7] K. Simonyan and A. Zisserman, *Two-stream convolutional networks for action recognition in videos,* CoRR **abs/1406.2199** (2014), arXiv:1406.2199 .

[8] Z. Xu, Y. Yang, and A. G. Hauptmann, *A discriminative CNN video representation for event detection,* CoRR **abs/1411.4006** (2014), arXiv:1411.4006 .

[9] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, *Actionvlad: Learning spatio-temporal aggregation for action classification,* in *Proceedings of (CVPR) Computer Vision and Pattern Recognition* (2017) pp. 3165 – 3174.

[10] J. Carreira, V. Patraucean, L. Mazaré, A. Zisserman, and S. Osindero, *Massively parallel video networks,* CoRR **abs/1806.03863** (2018), arXiv:1806.03863 .

[11] Z. Shou, D. Wang, and S. Chang, *Action temporal localization in untrimmed videos via multi-stage cnns,* CoRR **abs/1601.02129** (2016), arXiv:1601.02129 .

[12] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, *C3D: generic features for video analysis,* CoRR **abs/1412.0767** (2014), arXiv:1412.0767 .

[13] J. Carreira and A. Zisserman, *Quo vadis, action recognition? A new model and the kinetics dataset,* CoRR **abs/1705.07750** (2017), arXiv:1705.07750 .

[14] C. Feichtenhofer, H. Fan, J. Malik, and K. He, *Slowfast networks for video recognition,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2019).

[15] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, *Video action transformer network,* CoRR **abs/1812.02707** (2018), arXiv:1812.02707 .

[16] N. Hussein, E. Gavves, and A. W. M. Smeulders, *Timeception for complex action recognition,* CoRR **abs/1812.01289** (2018), arXiv:1812.01289 .

[17] B. Martinez, D. Modolo, Y. Xiong, and J. Tighe, *Action recognition with spatial-temporal discriminative filter banks,* (2019), arXiv:1908.07625 [cs.CV] .

[18] Z. Qiu, T. Yao, and T. Mei, *Learning spatio-temporal representation with pseudo-3d residual networks,* CoRR **abs/1711.10305** (2017), arXiv:1711.10305 .

[19] D. Tran, H. Wang, L. Torresani, and M. Feiszli, *Video classification with channel-separated convolutional networks,* CoRR **abs/1904.02811** (2019), arXiv:1904.02811 .

[20] J. Carreira, V. Patraucean, L. Mazaré, A. Zisserman, and S. Osindero, *Massively parallel video networks,* CoRR **abs/1806.03863** (2018), arXiv:1806.03863 .

[21] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, *Activitynet: A large-scale video benchmark for human activity understanding,* in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015) pp. 961–970.

[22] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, *The kinetics human action video dataset,* CoRR **abs/1705.06950** (2017), arXiv:1705.06950 .

[23] H. Kuehne, A. Arslan, and T. Serre, *The language of actions: Recovering the syntax and semantics of goal-directed human activities,* in *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014) pp. 780–787.

[24] K. Hara, H. Kataoka, and Y. Satoh, *Learning spatio-temporal features with 3d residual networks for action recognition,* CoRR **abs/1708.07632** (2017), arXiv:1708.07632 .

[25] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition,* CoRR **abs/1512.03385** (2015), arXiv:1512.03385 .

[26] J. Carreira and A. Zisserman, *Quo vadis, action recognition? A new model and the kinetics dataset,* CoRR **abs/1705.07750** (2017), arXiv:1705.07750 .

[27] K. Liu, W. Liu, C. Gan, M. Tan, and H. Ma, *T-c3d: Temporal convolutional 3d network for real-time action recognition,* in *AAAI* (2018).

5

DIMENSIONALITY REDUCTION

This chapter aims to give a general overview of the most used dimensionality reduction methods.

5.1. INTRODUCTION

Dimensionality reduction (DR) has been an essential tool for high-dimensional data analysis. Linear DR method such as PCA is easier to understand since the lower-dimension representation is a linear combination of the high-dimensional axes. Non-linear methods, on the other hand, are not so easy to explain but are more useful to capture a more complex high-dimensional pattern [1]. In general non-linear DR tries to minimize the local structure of the data from high-dimension to low dimension and tends to ignore more considerable distances between data points [2].

5.2. T-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) introduced by [3] is a non-linear DR technique which is used more for the visualization purposes as it raised some concerns regarding the reliability and interpretability of the results. [4] mentions some of the disadvantages of using t-SNE such as 1) Reliability of the results on the hyper-parameter choice 2) The fact that the cluster size might not mean anything 3) Distance between clusters might be deceiving. However, the data used in [4] is artificial data in which the distribution is known before applying t-SNE. Some works have been done to avoid these issues and use t-SNE more effectively. [2] proposes an interactive tool to support interactive exploration and visualization of high-dimensional data. In our research paper, a Barnes-Hut variation [5] of t-SNE has been used to reduce the algorithm complexity from $O(N^2)$ to O(N log N) where N is the number of data-points. t-SNE has shown satisfactory results in reducing feature dimensions while keeping the local homogeneity; therefore, it is used as the primary method to visualize high-dimensional data in this work.

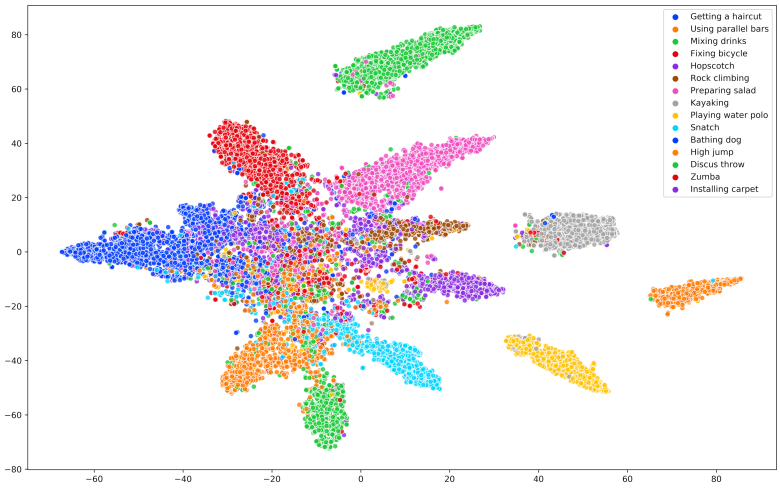


Figure 5.1: t-SNE projection of extracted features from 407 videos of a ActivityNet subset using extracted features from ResNet-34 3D

In t-SNE, the pair-wise distance between the data-points is transferred to a probability distribution to represent similarity.

$$p_{ij} = \frac{exp(-\parallel x_i - x_j \parallel^2)/2\sigma^2)}{\sum_{k \neq i} exp(-\parallel x_k - x_i \parallel^2)/2\sigma^2)} \tag{5.1}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{5.2}$$

The pair-wise similarity in the high dimensional space is calculated with Equation 5.1 and 5.2. Each low-dimensional points are also modeled as a probability distribution called q_ij and is calculated using the Equation 5.3

$$q_{ij} = \frac{(1+\parallel y_i - y_j \parallel^2)^{-1}}{\sum_{k \neq L}(1+\parallel y_k - y_l \parallel^2)^{-1}} \tag{5.3}$$

To find the proper low-dimensional(Q) representation of the high-dimensional representation (P), a cost function (C) has been defined in Equation 5.4 using Kullback-Leibler (KL) divergence between the probability distributions. The cost function is later optimized with gradient descend in a certain number of iterations.

$$KL(P_i \parallel Q_i) = \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{5.4}$$

REFERENCES

[1] V. M. Lee, John A., *Nonlinear Dimensionality Reduction* (Springer, 2007).

[2] A. Chatzimparmpas, R. M. Martins, and A. Kerren, *t-visne: Interactive assessment and interpretation of t-sne projections,* IEEE Transactions on Visualization and Computer Graphics **26**, 2696–2714 (2020).

[3] L. van der Maaten and G. Hinton, *Visualizing high-dimensional data using t-sne,* Journal of Machine Learning Research **9**, 2579 (2008), pagination: 27.

[4] M. Wattenberg, F. Viégas, and I. Johnson, *How to use t-sne effectively,* (2016).

[5] L. van der Maaten, *Barnes-hut-sne,* (2013), arXiv:1301.3342 [cs.LG] .

6

DATA ANNOTATION

This chapter aims to familiarize the reader with some of the most used data annotation methods.

6.1. INTRODUCTION

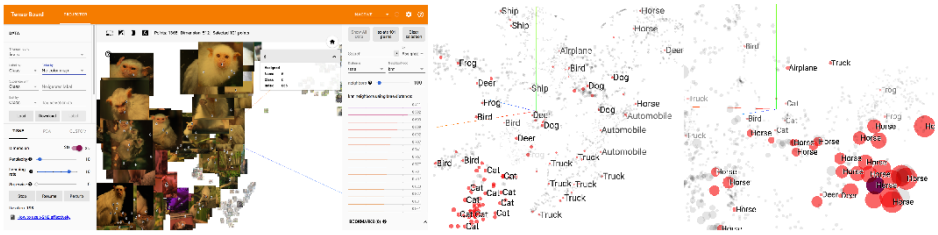
Data to machine learning is like oxygen to human beings; without sufficient and high-quality data, machine learning models can not achieve the proper accuracy; however, annotating is a labour intensive task for the oracle. Different tools have been proposed for making an easy annotation tool for videos and images. However, they usually do not exploit the structure of the data, which is especially useful in videos. In the past decades, some works have been done to make the process of image annotation easier. [1] offers a real-time framework for annotating internet images and [2] uses multi-instances learning to learn the classes and image attributes together; however, none of these methods use a deep representation of data.

In more recent works [3] uses Deep Multiple Instance Learning to annotate images automatically, and [4] uses semi-supervised t-SNE and feature-space visualization in lower dimension to provide an interactive annotation environment. Some works, such as [5] proposed a general framework for annotating images and videos. However, to our understanding, no video annotation platform has been proposed which exploits the data structure.

6.2. ANNOTATION TOOLS

6.2.1. IMAGE ANNOTATION

There are many image annotation tools available that can output the annotation based on different dataset styles, such as PascalVOC or YOLO [6]. However, only a few tools exploit the feature similarity between the images to reduce the work done by the oracle. [4] proposes a framework on top of the Tensorboard Projector based on the feature similarity between images for active learning purposes.



(a) TensorBoard Projector (b) Cat sample (c) Horse sample

Figure 6.1: a) Tensorboard Projector b)3D t-SNE projection of CIFAR-10) [4]

VIDEO ANNOTATION

The conventional way of labeling video data that is still being done on some of the research is playing the video by oracle in a typical video player, and the oracle can pause the video at any time and label the temporal boundaries of the action. This is usually done in the form of crowdsourcing the job through Amazon Turk or other platforms. Such methods are frustrating; however, some open-source tools make the annotation process easier with small tricks.

MuViLab is an open-source video annotation tool that can be used for temporal and class label annotation of videos faster than the conventional method. As can be seen in Figure 6.2, MuViLab shows the entire video in the form of smaller clips near each other in one figure so that it is more manageable for the oracle to see the similarity between the clips and make the annotation easier. However, even such software does not exploit the video's temporal aspect to its fullest, which is what we propose in our research paper.

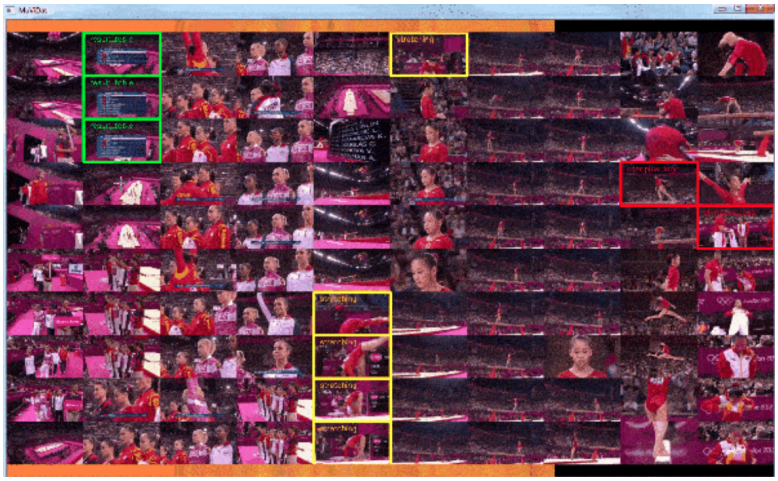


Figure 6.2: MuViLab [7] for video annotation

REFERENCES

[1] J. Li and J. Z. Wang, *Real-time computerized annotation of pictures,* IEEE Transactions on Pattern Analysis and Machine Intelligence **30**, 985 (2008).

[2] Gang Wang and D. Forsyth, *Joint learning of visual attributes, object classes and visual saliency,* in *2009 IEEE 12th International Conference on Computer Vision* (2009) pp. 537–544.

[3] J. Wu, Y. Yu, C. Huang, and K. Yu, *Deep multiple instance learning for image classification and auto-annotation, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2015).

[4] F. P. S. Luus, N. Khan, and I. Akhalwaya, *Active learning with tensorboard projector,* CoRR **abs/1901.00675** (2019), arXiv:1901.00675 .

[5] A. Dutta and A. Zisserman, *The via annotation software for images, audio and video,* in *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19 (Association for Computing Machinery, New York, NY, USA, 2019) p. 2276–2279.

[6] a Multiple contributors, *labelimg,* https://github.com/tzutalin/labelImg (2017).

[7] L. D. Alessandro Masullo, *Muvilab,* https://github.com/ale152/muvilab (2019).