11-777 Spring 2021 Class Project

Yun Cheng[*] **Yuxuan Liu**[*] **Tiffany Ma**[*] **Erin Zhang**[*]
{yuncheng, yuxuanli, tma1, xiaoyuz1}@andrew.cmu.edu

Abstract

Template for 11-777 Reports using the ACL 2021 Style File

1 Introduction and Problem Definition

[*] Everyone Contributed Equally – Alphabetical order

2 Related Work and Background

Action segmentation The goal of action segmentation is to temporarily localize action segments and classify the category of the action in each segment in an untrimmed input video. The application of action segmentation can be found in various fields such as robotics (Pardowitz et al., 2009) and behavior analysis (Shao et al., 2012). Action segmentation is closely related but different from action recognition and action detection. Action recognition identifies one action in a trimmed video, and action detection usually outputs a sparse set of actions. By contrast, action segmentation is a more complex task that considers a longer range of temporal relations between sequential activities for more fine-grained action recognition in an untrimmed video.

Early works Traditional approaches generally fall into three categories: sliding window approaches, segmental models, and recurrent networks (Huang et al., 2020). One of the earliest attempts is to detect action segments with temporal windows of different scales and non-maximum suppression (Rohrbach et al., 2012). However, this method is limited by the tradeoff between larger window size and computational costs. Others use segmental models like spatiotemporal CNNs with the semi-Markov model for tracking object relationships, action transitions, and environment change (Lea et al., 2016; Fathi and Rehg, 2013). With each action conditioned on the previous one, these methods are good at capturing local dependencies in consecutive visual patterns rather than long-range temporal relations. Other hybrid approaches include representing frames using Fisher Vectors with HMMs and GRUs for temporal modeling (Kuehne et al., 2016; Richard et al., 2017), which have their main drawback of efficiency. Another line of research focuses on temporal convolutional networks (TCNs) that perform fine-grained action segmentation using temporal convolutions (Lea et al., 2017). The method is extended to a multi-stage architecture with a set of dilated temporal convolutions in each stage. It is proven to be able to avoid temporal pooling and better capture long-range dependencies (Farha and Gall, 2019).

Graph convolution networks Recently, existing models are further improved by the introduction of graph convolution networks (GCNs). Built on top of action segmentation models, the graph-based module models the temporal relations between initial segmentation results with temporal proximity. It refines the pre-computed action segments by performing segment boundary regression and segment classification (Huang et al., 2020). The latest extension to this approach constructs multi-level dilated temporal graphs for temporal reasoning at different timescales (Wang et al., 2020). The limitations of the graph-based module exist in its dependence on the initial backbone output. For example, it suffers from low efficiency on large graphs if the initial segmentation is heavily fragmented. However, while abundant works have been done in unimodal action segmentation, we observe that almost none of the existing work attempts at multimodal approaches. Since textual data is one of the most common and accessible annotations of video data, we are interested in incorporating texts as a complimentary domain for better segmentation results.

Text alignment Identifying the relationship between two or more modalities is one of the core challenges in Multimodal settings (Baltrušaitis et al., 2019). An example of the unsupervised approaches to text alignment is to first perform temporal clustering individually on the video input and the text input, then use the two clusters to provide complementary information to one another (Alayrac et al., 2016). For instance, differences in two video segments can provide a temporal cue to a breaking point within the narrative script. Contextual information is used to assist the alignment of textual scripts and video frames (Shi et al., 2020). It is built by firstly collecting a mean pooling of each modality within K units. The mean representation of two modalities is then combined through a transformer model and concatenated to the embedding of the individual. Our task concerns video and scripts in the cooking domain. In one of the similar experiments, the text script is parsed into action-object (i.e. verb-noun) classes, and the video frames are aligned to the text script by matching the tokens of action-objects to those present in the frames (Malmaud et al., 2015).

Text-image matching To build better representation for the graph nodes, we want to use the narrations to attend to regions in the frame that are closely associated with the action that is conducted. In this way, different relevant objects in different frames can be used to distinguish the segment boundaries.

To use attention methods, we first need to provide a set of image features. To better extract object features in the images, Faster R-CNN (Ren et al., 2016) firstly generates Region Of Interests (ROIs) with high objectness, and it then uses intermediate convolution feature maps to classify the region and regress bounding boxes. Fully Convolutional One-Stage Object Detection (FCOS) (Tian et al., 2019) belongs to a family of anchor-less methods. Instead of regressing bounding boxes using the anchors as references, it regresses four values, l, t, r, b that represent the distance from a location in the image to the four sides of the bounding boxes. Moreover, it uses CNN feature maps from different levels to perform bounding box regression at various scales to capture objects with different sizes. It has been shown that anchor-less detectors perform better than anchor-based detectors on seen and unseen test sets, and FCOS can identify objects involved in the action (Yoon et al., 2020).

Given a set of image features, encoding regions in the image, and a set of word features extracted from the sentence, Stacked Cross Attention (Lee et al., 2018) determines the similarity between image-sentence pair by inferring how important a region is to the sentence, and it can also reversely infer how important a sentence is to the image. An additional position feature is concatenated for the object with the visual feature extracted by ResNet (Wang et al., 2019). The image is divided into blocks, and embedding vectors representing the positions of the blocks are combined with weights determined by overlap between the block and the visual feature. The addition is motivated by the fact that the positions of objects in the image are related to the semantics of the image. This intuition aligns with our task since we expect that the relative positions of objects are associated with the action during cooking.

3 Task Setup and Data (1 page)

The main task is to segment egocentric (first-person) cooking videos from EPIC-KITCHENS dataset into action-object pairs. Given a video clip in the form of a sequence of frames, we want to identify the type of actions as well as their start and end time in the given video.

3.1 Dataset

We use the largest egocentric (first-person) dataset EPIC-KITCHENS-100, which features 100 hours, 700 variable-length videos with 90K actions of 37 participants (Damen et al., 2020). Compared to YouTube-based datasets such as HowTo100M (Miech et al., 2019), EPIC-KITCHENS contains activities that are non-scripted and thus capture more natural settings such as parallel tasking . The egocentric view provides a unique perspective on people-object interactions, attention, and intention. Meanwhile, it also imposes extra challenges compared to third-person datasets like YouCook2 (Zhou et al., 2018). One of the challenges is that certain actions, such as eating and drinking, cannot be directly observed due to the limited field of view. Other challenges include unseen participants, unseen cooking actions, frame noises from different sources (i.e. background and lighting), long videos with many action instances, fragmentation of segments resulted from interleaving actions in multi-tasking, and weaker temporal correlations in objects interfering the correlations in actions.

3.2 Task formulation

There are two input modalities: video frames of egocentric cooking scenes and narrations describing the action in the scenes. The narrations are transcribed from the audio in the form of imperative phrases: verb-noun with optional propositional phrase. The goal is to predict a verb class as well as a noun class for each frame to identify the action in the segments. Afterwards, we combine the two classes into a tuple as the final output class label.

Formally, the visual input consists of a sequence of M RGB frames in temporal order, denoted as $F = (\mathbf{f}_i)_{i=1}^{M}$. The RGB frames are sampled from untrimmed videos at a rate of 50 frames per second. The textual input is a sequence of N audio-transcribed narrations in temporal order, denoted as $C = (\mathbf{c}_i)_{i=1}^{N}$. Our goal is to infer the action-object class label for each frame. The ground truth is given by $Y = (\mathbf{y}_i)_{i=1}^{M}$. Each $\mathbf{y}_i \in \{0,1\}^K \times \{0,1\}^L$ is a tuple of one-hot vectors encoding the true verb and noun class, where K is the number of verb classes and L is the number of noun classes.

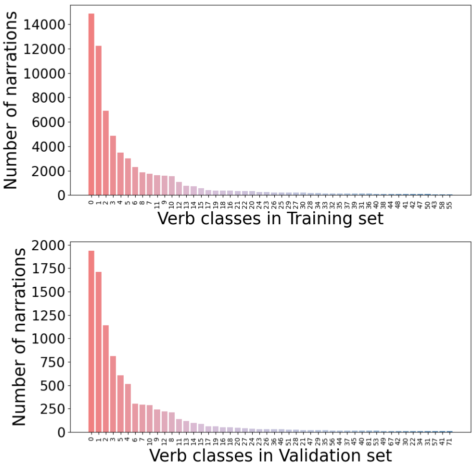


Figure 1: Frequency distribution of 50 most frequent verb class in training and validation set

3.3 Dataset Statistics

3.3.1 Text Analysis

Narrations in EPIC-KITCHENS-100 are mainly imperative phrases in the form of verb-noun with optional propositional phrase (e.g. *put down plate*, *put container on top of counter*). Each annotation includes start/stop timestamps and frames, action verbs and object nouns, which are extracted from the corresponding narration. Verbs and nouns are further classified into classes based on their semantic meaning. For example, *grab* and *get* belong to the same verb class. There are a total of 97 verb classes and 300 noun classes in the training and validation set.

We define the frequency of a verb/noun class as the number of narrations that contain a verb/noun from that class. Both verb and noun classes have a heavy tailed distribution with tail classes ($\leq$ 1/15 of the maximum frequency) accounting for 13.02% and 11.67% total verbs and 5.38% and 1.85% total nouns in the training and validation set respectively (Figure 1). Such a distribution indicates the intrinsic complexity and entropy of the text data. The training and validation set have similar composition: in the validation set, there are no unseen verb class and only four unseen noun classes, accounting for 0.03% of all narrations. Narration timestamps are relatively complete: only 0.17% in training and 0.72% in validation are missing. On the other

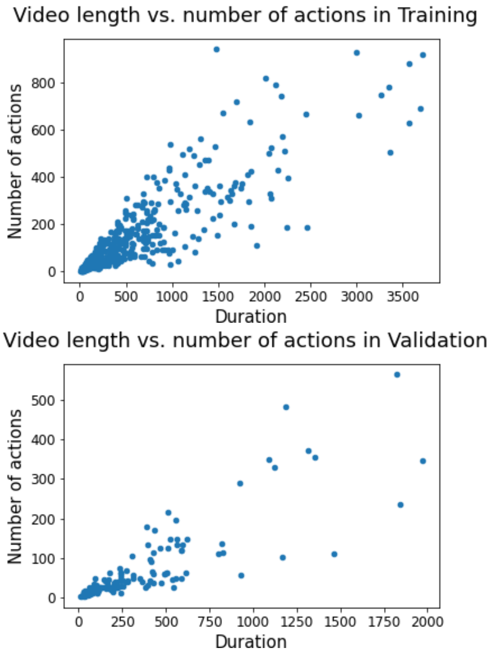


Figure 2: Number of action in each video against video length (in seconds)

hand, since there were no constraints on the recording duration, we observe a great variability across videos. Average sentence length of training and validation set is 15.1 and 14.8 with standard deviation of 6.3 and 6.0 words, respectively. Average number of actions per video is 135.8 (training) and 70.1 (validation) with standard deviation of 167.7 and 93.2. More distribution statistics can be found in Appendix A Table 2.

A natural assumption of our task is that there is none or minimal overlapping between action segments, i.e. only one action in almost all time frames. We check that there are at most 4 narrations in parallel in training and 3 in validation; only 3832 (5.70%) and 617 (0.92%) pairs of consecutive actions overlap for more than 1 second. We also inspect the feature embeddings of the verb and noun classes. Using GloVe word vectors pre-trained on Twitter (200d vectors) (Pennington et al., 2014), we do not notice significant interclass or intraclass clustering effect (Appendix A Figure 5).

3.3.2 Video Frame Analysis

We extract 1920 × 1080 RGB frames from the videos at a sampling rate of 50 FPS. Each frame is identified by participant id, video id, and a start/end frame number. More than half of the total 700 videos in the dataset have less than 25,000 frames.

Videos in EPIC-KITCHENS-100 have varied length with the longest video of 3708 seconds and shortest video of 10 seconds. 85.7 % of the videos are shorter than 1000 seconds and 66.0 % are less than 500 seconds (Appendix A Table 2, Figure 6). We also see that the number of narrations grows roughly linearly with video length (Figure 2).

We compile all training and validation samples of a given verb class and compute the average number of frames for this class (Appendix Figure 9). The top-10 verb classes with the most number of frames include actions like *grate, wait, prepare, knead, stir*, and *cut*; while those with the least number of frames contain actions like *bend, turn-off, turn-on, take, close*. It seems that actions involved during cooking take longer than those related to intermediate preparatory steps, and the average length of the action aligns with how people would respond if asked about which action would take longer. For most verb classes, the average number of frames in each class are roughly the same in both training and validation set, except a few where the validation sets have more frames. We also count the total number of frames for each verb class, summed over all training and validation samples in the class. We notice that such frequency corresponds to the trend of verb-class frequency in the annotations (Appendix A Figure 10). This indicates that within the dataset, the frequency of the verb class correlates to the amount of visual information in the dataset.

The dataset also provides bounding-box annotations for each frame, where it only distinguishes between two categories: hands and objects. Only active objects are annotated, so the number of object bounding-boxes in a frame approximates the number of objects that the person interacts with. We compute the average number of hand bounding-boxes appearing in a frame of each verb class. Class with less than 1.5 hand bounding-boxes include actions like *take, put-on, open, pull-down, walk*, and these correspond roughly to human impression on how many hands are needed for performing the action. We also compute the average number of objects bounding boxes in a frame of a given verb class. Verb classes with less than 1.8 object bounding boxes include actions like *open, close, shake, check, fold*, and *drink* (Appendix A Figure 7, 8). The average numbers of hand and object bounding boxes for the training and validation sets are mostly equal, despite the validation set misses a few verb classes. Full details can be found in Appendix A.

3.4 Metrics

We measure three metrics: frame-wise accuracy, segmental edit distance and segmental F1 score.

Frame-wise metrics is most commonly used in segmentation and include accuracy, precision, and recall. However, this group of metrics tends to be influenced more by actions with long duration than by those with short duration (Wang et al., 2020). Another problem is that it does not penalize for over-segmentation errors in the model (Wang et al., 2020; Lea et al., 2017). Therefore, we also consider segmental edit distance, which is useful because it reflects out-of-order and over-segmentation errors (Lea et al., 2017).

(Lea et al., 2017) also introduces segmental F1 score, which not only penalizes for over-segmentation but also avoids penalizing for minor temporal shifts between the prediction and the ground truth. It also has the advantage of depending on the number of actions instead of their duration. For each predicted action segment, we calculate its IoU with respect to the corresponding ground truth. If the score is above a threshold τ, then the prediction is considered as a true positive (TP) otherwise a false positive (FP). Over-segmentation is addressed since if more than one correct segments lie within a single true action, only one is labelled as TP and all others are FP. Here we consider overlapping thresholds of 10%, 25% and 50%, denoted by F1@$\{10, 25, 50\}$.

4 Models (2 pages)

4.1 Baselines

We have four baseline models: both EDTCN (Lea et al., 2017) and MS-TCN++ (Li et al., 2020) use temporal convolution networks to capture long-range dependencies. Richard et al. (2017) proposed a hybrid usage of GRU-based RNN and HMM to refine action alignment. DTGRM (Wang et al., 2020) uses multi-level dilated temporal graphs with an auxiliary self-supervised task to help correct wrong temporal relation in videos.

FC We implement a vanilla 2-layer fully connected neural network that performs frame-wise classification on the input video frames. The inputs are features of dimension 1024 extracted using pretrained I3D (Carreira and Zisserman, 2017). We use this simple model to show the task complexity and the improvement of other baselines in both accuracy and fragmentation issues.

EDTCN and Dilated TCN EDTCN (Lea et al., 2017) is the first work to present temporal convolution network (TCN), which aims to capture not only segmental features but also long-range patterns using a hierarchy of temporal convolution filters. It emphasizes the concept of receptive fields, which means a fixed-length input that the prediction output corresponds to. The encoder in the Encode-Decoder TCN (EDTCN) consists of layers of temporal convolutions, non-linear activation, and max pooling; the decoder has a similar structure, except that pooling is replaced with upsampling. Dilated TCN shares a similar structure with MS-TCN: a series of blocks, each with layers of dilated convolutions with exponentially growing dilation rates, and a residual connection between layer input and convolution signal.

MS-TCN++ MS-TCN (Farha and Gall, 2019) is a multi-stage architecture using TCN. The first layer of a single-stage TCN (SS-TCN) adjusts inputs dimension, followed by several dilated 1D temporal convolution layers with dilation factor doubled at each layer. All layers have ReLU activation with the residual connection. MS-TCN stacks four SS-TCNs so that each takes initial prediction probabilities from the previous stage and refines it. The overall architecture is trained with the cross entropy classification loss and a truncated mean squared error over the frame-wise log probabilities that penalizes over-segmentation. Extended from MS-TCN, MS-TCN++ (Li et al., 2020) decouples the prediction phase and the refinement phase and enables parameter sharing in the latter. Furthermore, it replaces the dilated layer with a dual dilated layer that combines two convolutions with different dilation factors in the first stage to addressing the limited receptive field. The SS-TCN in MS-TCN++ has 11 layers, each of which has 64 filters, kernel size 3 with a dropout rate of 0.5. The learning rate is set to be 0.0005 in training.

Weakly-supervised approach (RNN+HMM) Richard et al. (2017) tackles the action alignment task by iteratively refining the coarsely proposed segmentation. Given a set of video frames and an ordered action sequence, the model assigns an action segment index to each frame. The initial proposal for boundaries is constructed by equally dividing the frames upon actions in the action sequence in the sequence's original order. The model uses a hybrid component of both a GRU-based RNN network and an HMM network to train and realign the proposal. To create more fine-grained proposals, the authors propose to further split each action into a sequence of subactions. The model attempts to create more fine-grained segmentation by modeling the probability distribution of the subactions within an action using the HMM network. To alleviate the computational need for recurrent neural networks used for processing videos, the authors propose to split the video frames into overlapping chunks. For instance, a single input would be a video frame at time t and 20 frames before that. By splitting the input into smaller chunks, the model can take advantage of parallelism and utilize batch training.

DTGRM Wang et al. (2020) proposed DTGRM which refines a predicted result given by the backbone model (e.g. I3D) iteratively. The model stacks K dilated graph convolution layers to perform temporal reasoning across long timescales, where each layer updates the hidden representation of every input frame. To reduce over-segmentation error, an additional self-supervised task is introduced to simulate over-segmentation error by randomly exchanging part of input frames. Both the original and exchanged frame sequences are fed into the model as input, with the output being action class likelihood for two frame sequences as well as exchange likelihood for each frame. Since the model was trained on datasets with relatively shorter videos compared to EPIC-KITCHENS, we plan to trim the videos into overlapping clips of length 15 minutes with fixed fps for consistency.

4.2 Proposed Approach

Experiments on baseline models have demonstrated that one major issue with the pre-existing action segmentation methods like MSTCN on EPIC-KITCHENS dataset is to correctly classify the actions of each segment since EPIC-KITCHENS has much more diverse action classes than other datasets (Fathi et al., 2011; Stein and McKenna, 2013; Kuehne et al., 2014) that the baselines have evaluated on. Therefore, our proposed method utilizes MSTCN as a backbone model assisted by region of interest visual feature extraction and improves classification with an extra video-text matching component.

4.2.1 Region of Interest Visual Feature Extraction

As discussed earlier, one potential problem with the EPIC-KITCHENS dataset is the imbalance in the representation of visual and textual data. While the visual data contains both rich spatial and temporal information, the short verb-noun phrases provide very limited context to build an equally-representative textual space. To account for this imbalance, we assist the video-text embedding by providing visual inputs that aligns with the verb-noun annotations: we extract specific regions in the visual frames that represent actions and/or objects.

The SlowFast network (Feichtenhofer et al., 2019) consists of two streams of feature extractions: the Fast pathway focuses on extracting temporal information across a set of densely sampled frames, while the Slow pathway focuses on representing spatial semantics with high channel capacity and low temporal rate. Moreover, we want to incorporate region of interest (RoI) proposals into the feature extraction procedure such that we extract features of specific objects and actions and discard additional context such as background since textual information lacks additional context. Instead of passing in the full-resolution frame into the SlowFast network, we pass in sub-parts of the frame as proposed by Region Of Interest (RoI) models, thereby extracting object-specific or action-specific visual features.

4.2.2 Backbone Model

We use the original implementation of MSTCN in Farha and Gall (2019) as the backbone model since experiments on baselines show that it performs relatively well on finding action boundaries. The inputs to the backbone model are the RoI visual features extracted from the SlowFast network. Given the feature vectors $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ of a video, the model outputs an initial segmentation $(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_M)$ where M is the number of frames and $\hat{\mathbf{y}}_i$ is the action class label of the predicted verb of frame i.

4.2.3 Video-Text Matching

Since misclassification is one of the prominent issue in the baseline experiments, our proposed solution utilizes an enriched, pretrained video-text embedding to improve the labeling. We first extracted frame-level and video-level features similarly as in Miech et al. (2019). 2D features are extracted with the ImageNet pre-trained Resnet-152 (He et al., 2016) at the rate of about 1 FPS, and 3D features are extracted with the Kinetics (Carreira and Zisserman, 2017) pre-trained ResNeXt-101 16-frames model (Hara et al., 2018) to obtain about 0.78 feature per second. Denote the 2D features as $(\mathbf{x}_1^{2D}, \dots, \mathbf{x}_{M_{2D}}^{2D})$ and the 3D features as $(\mathbf{x}_1^{3D}, \dots, \mathbf{x}_{M_{3D}}^{3D})$ where $\mathbf{x}_i^{2D}, \mathbf{x}_i^{3D} \in \mathbb{R}^{2048}$. Given an initial segmentation result produced by the backbone model $(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_M)$, we first find the set of frame indexes corresponding to segment i as $(s_i^{2D})_{i=1}^t$ and $(s_i^{3D})_{i=1}^t$ respectively, where t is the number of segments, then we aggregate the features of one segment using temporal maxpooling and concatenate 2D and 3D features to form a single 4096-dimensional feature vector

$$
\begin{aligned}
\mathbf{v}^{2D} &= maxpool(\{\mathbf{x}_j^{2D}\}_{j \in s_i^{2D}}) \\
\mathbf{v}^{3D} &= maxpool(\{\mathbf{x}_j^{3D}\}_{j \in s_i^{3D}}) \\
\mathbf{v}_i &= concat(\mathbf{v}^{2D}, \mathbf{v}^{3D})
\end{aligned}
$$

Similar to Miech et al. (2019), we also use the GoogleNews pre-trained word2vec embedding model to obtain a word embedding $\mathbf{c}_i$ for each verb i (96 c_i if using actions for each action category, 977 c_i if differentiating among actions within the same action category). We then transform $\mathbf{v}_i, \mathbf{c}_i$ using the learned projection function finetuned on EPIC-KITCHENS $f : \mathbb{R}^{2048} \rightarrow \mathbb{R}^d, g : \mathbb{R}^{2048} \rightarrow \mathbb{R}^d$ where d is the dimension of the common video-text embedding space. Finally, we perform video-text matching between a segment $\mathbf{v}_i$ and every verb

$\mathbf{c}_i$ by computing the cosine similarity score as

$$s(\mathbf{v}_i, \mathbf{c}_j) = \frac{\langle f(\mathbf{v}_i), g(\mathbf{c}_j)\rangle}{\|f(\mathbf{v}_i)\|_2 \|g(\mathbf{c}_j)\|_2}$$

which is high when the action $\mathbf{c}_j$ is likely to take place in the segment represented by $\mathbf{v}_i$.

4.2.4 Improve Video-Text Matching with Cross-Modal Attention

The above describes a dual encoder model that independently maps text and video to a joint embedding. It has the advantage in scalability as it can results in efficient evaluation during test time. However, as Miech et al. (2021) points out, it has limited accuracy since the simple dot product is unlikely to capture the complex vision-text interactions. Analogous to how human perform video-text retrieval, one solution is to roughly select a few promising candidates then do fine-grained search for the best candidate by paying more *attention* to visual details. Therefore, we adapt the *Fast* and *Slow* models of Miech et al. (2021) in which the *fast* dual encoder quickly eliminates candidates with low relevance while the *slow* cross-attention model improves retrieval performance with grounding. Given an input segment $\mathbf{v}_i$, we perform retrieval by searching for an action class $\mathbf{c}_j$ such that segment $\mathbf{v}_i$ is most likely to decode action class $\mathbf{c}_j$. Specifically, given segment and action class pair $(\mathbf{v}_i, \mathbf{c}_j)$, we compute their similarity by

$$h(\mathbf{v}_i, \mathbf{c}_j) = \log(p(\mathbf{c}_j|\phi(\mathbf{v}_i); \theta))$$

where $\phi(\mathbf{v}_i)$ is extracted feature of segment $\mathbf{v}_i$ and θ is the parameters of the transformer model. To combine results from dual encoder model and cross-attention model, given input segment $\mathbf{v}_i$ and action class set $\mathcal{C}$ containing K action classes. we first obtain a subset of m action classes $\mathcal{C}_m$ (where $m \ll K$) that have the highest score according to the fast dual encoder model. We then retrieve the final top ranked action class by re-ranking the candidates using the cross attention model:

$$\mathbf{y}_i^* = \text{argmax}_{\mathbf{c}_j \in \mathcal{C}_m} h(\mathbf{v}_i, \mathbf{c}_j) + \beta s(\mathbf{v}_i, \mathbf{c}_j)$$

where β is a positive hyper-parameter that weights the output scores of the two models. We output $(\hat{\mathbf{y}}_{i,c}^*)$ as the classification probability of frame i as action c based on the similarity score and $(\mathbf{y}_i^*)_{i \in s_i^{3D}}$ as new labels for segment $i, i \in [t]$.

4.2.5 Novelty and Challenges

While most existing action segmentation methods work purely with video input, our approach is the first attempt to action segmentation in the multimodal setting. In particular, we exploits the semantic meaning of the text annotations to improve performance. Moreover, most action segmentation methods evaluate on smaller and simpler datasets, while EPIC-KITCHENS that we attempt is comparably much larger and more complex.

Meanwhile, existing methods aim to learn better temporal relationships among frames that are close together or far away from each other. For example, MSTCN uses dilated convoltuion and MSTCN++ improves with two branches of dilated convolutions each with a different dilation factor; DTGRM contains layers of graphs with nodes spanning neighborhoods of different sizes. These models, by surveying frames across time and learning temporal relationships, aim to differentiate between frames from the same action versus those from different actions. The intuition behind our method, however, is that an action verb, such as *take*, represents a very generic idea that could correspond to a large number of video segments with different contexts that are confounding factors: the specific objects that *take* happens on, other irrelevant objects in the scene, the background of the kitchen, or the way a *take* action takes place. Therefore, if the video-text matching model can make all video segments of *take* clustering around the word *take* in the joint embedding, then it will eliminate the distractions from intraclass variation within the category *take*.

Without extensive training and fine-tuning, when evaluated on recall metrics, R@$\{1, 5, 10\}$, the projection model gives comparable result as in the original pretrained HowTo100M model on image-text retrieval of the MSR-VTT dataset. However, the retrieval happens between videos and texts of the same batch; if we project all action verbs to the embedding space and rank which verb is the closest to a given video segment, the result is less desirable. Therefore, learning a good joint embedding space will be the biggest challenge of our method, especially given that we pre-tested with segments extracted from a video with ground truth start and end frame. Furthermore, the matching results are likely to be largely dependent on the initial segmentation. If the output from MSTCN is too noisy, which is very likely at the beginning stage of training, the video-text matching model might end up mapping all ambiguous segments, which contain a mix of parts from different actions, to a space that is equidistant away from all word embeddings of the action verbs. A potential solution to this issue may be to train MSTCN for a few epochs for a more stabilized and credible segmentation before adding in the video-text matching model to better guide classification.

We will also attempt to use the more meaningful, full annotation phrase in the form of (verb, noun) pair in the video-text matching. We expect this to create further challenge, one of which is the joint embedding may focus on clustering based on the objects rather than the actions. For example, if we have four segments corresponding to *take apple*, *take banana*, *put-down apple*, *put-down banana*, despite ideally we want *take apple* and *take banana* to be closer, it is possible that *take apple* and *put-down apple* segments are closer since the manipulated objects *apple* and *banana* are more "visible" in the video. In that way, the video-image matching model will not help with classifying actions. The non-descriptive nature of the annotations makes it difficult to learn a joint embedding space that separates segments of different actions.

4.2.6 Loss Function

Backbone We use a combination of cross-entropy classification loss

$$\mathcal{L}_c = \frac{1}{M}\sum_{t=1}^{M} -\log(\hat{\mathbf{y}}^*_{t,c})$$

and truncated mean squared smoothing loss that aims to reduce over-segmentation errors as in (Farha and Gall, 2019)

$$\Delta_{t,c} = |\log \hat{\mathbf{y}}^*_{t,c} - \log \hat{\mathbf{y}}^*_{t-1,c}|$$
$$\tilde{\Delta}_{t,c} = \begin{cases} \Delta_{t,c} & \text{if } \Delta_{t,c} \leq \tau \\ \tau & \text{otherwise} \end{cases}$$
$$\mathcal{L}_s = \frac{1}{MK}\sum_{t,c} \tilde{\Delta}^2_{t,c}$$

where M is the number of frames, K is the number of action classes, $\hat{\mathbf{y}}^*_{t,c}$ is the output probability of action class c of frame t. We use $\tau = 4, \lambda = 0.15$ as in the original experiment. The final loss function is given as the sum of loss at each stage of temporal convolution

$$\mathcal{L}_{stage} = \mathcal{L}_c + \lambda\mathcal{L}_s$$
$$\mathcal{L} = \sum_{stage} \mathcal{L}_{stage}$$

Video-Text Matching The joint embedding is trained separately using the max-margin ranking loss as in (Miech et al., 2019). The loss is given by

$$\sum_{i\in\mathcal{B}}\sum_{j\in N(i)} \max(0, \delta + s_{i,j} - s_{i,i}) + \max(0, \delta + s_{j,i} - s_{i,i})$$

where $\mathcal{B}$ is a mini-batch sample of segments-verb training pairs, s is the similarity score matrix of all training pairs, $N(i)$ denotes the set of negative pairs for pair i and δ is the margin. We fix $\delta = 0.1$ as in the original experiment.

	Train					Test				
Methods	Acc	Edit	F1@$\{10, 25, 50\}$			Acc	Edit	F1@$\{10, 25, 50\}$		
FC	44.00	26.71	12.42	22.64	19.40	34.90	18.58	17.47	13.66	8.04
MS-TCN (Farha and Gall, 2019)	43.52					38.65				
DTGRM (Wang et al., 2020)	52.58					37.71				
Proposed Method										

Table 1: Results of baseline models

5 Results (1 page)

The columns above are just examples that should be expanded to include all metrics and baselines.

6 Analysis (2 pages)

In this section, we will analyze the baseline models. In particular, since the baselines have not been evaluated on EPIC-KITCHENS previously, we conduct several ablation experiments to access the complexity of our dataset compared to other benchmark datasets used in the original papers of these models.

6.1 Analysis of Baselines

Datasets EPIC-KITCHENS is the largest egocentric dataset. Compared to other egocentric datasets such as GTEA (Fathi et al., 2011) and cooking datasets such as 50Salads (Stein and McKenna, 2013) and Breakfast (Kuehne et al., 2014), EPIC-KITCHENS contains much longer videos and richer verb classes. In addition, while 50Salads mostly contains consecutive actions, actions in EPIC-KITCHENS are usually separated by background frames where no action is being performed. These background frames comprise a significant portion of all frames (about 32%) and affect the performace of baseline models, especially the simpler ones.

FC A 2-layer fully connected network is trained with batch size 16 for 50 epochs. The input frames are down-sampled to 1.25FPS. Since the classification is performed frame-wise and considers no temporal relations, the result is highly fragmental. We further note that the model tends to overfit at an early stage. The poor generalizability is indicated by the relatively low Edit score and Figure 3.

MSTCN MSTCN demonstrates its effectiveness in segmenting out the most frequent label classes. The top 5 most frequent labels in the training set in EPIC-KITCHENS are the background, *wash*, *take*, *put*, and *cut* class. We observe that the model assigns one of the most frequent verb classes when it struggles to label the action classes. The result implies that the model is able to memorize the label frequency. One potential way to alleviate this situation is to normalize the frequency through the positional weights supplied to each class label.

EPIC-KITCHENS is more complex than the benchmark datasets in many aspects, such as longer video durations and thus more actions involved. We accounted for this increase in complexity by using 15-layer single-stage TCNs rather than the 10-layer ones which are claimed to achieve optimal performance in the original experiment (Farha and Gall, 2019). Our experiment shows that there is an improvement in when using a more complex model.

DTGRM The DTGRM model builds off MSTCN by adding an additional fine-tuning component that refines segmentation around the boundaries. Similar to MSTCN, DTGRM is able to output reasonable segmentation results. 3 shows that one of its improvements from MSTCN is its capability in clearly segmenting out smaller segments, which proves the effectiveness of the additional fine-tuning component even on a more complex dataset like EPIC-KITCHENS. However, we also notice that DTGRM tend to over-segment on videos with fewer segments.

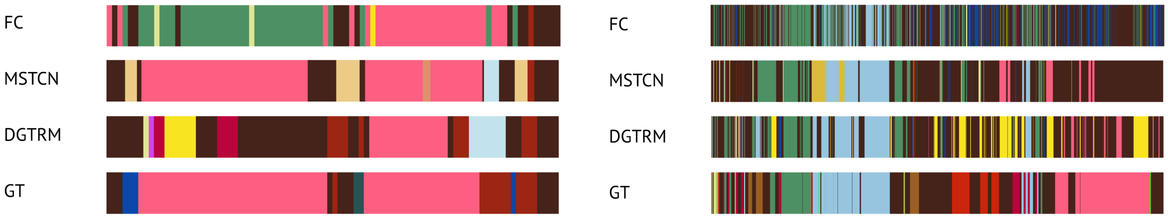
DTGRM also suffers from the label class imbalance problem. Similar to MSTCN, it labels majority of the segments as the most frequent vocabulary classes in the training set. With the weighted loss, DTGRM is able to predict a wider variety of labels. However, the classification accuracy is not as high as in the original experiment (Wang et al., 2020). This is reasonable given the richer vocabulary classes available in EPIC-KITCHENS.

Due to limitations on hardware, we are not able to expand the number of layers in the DTGRM model. To accommodate this, we sampled the feature inputs for every 10 frames of input to decrease input size. The more complex, sub-sampled DTGRM improved the over-segmentation issue in the original DTGRM by avoiding overly short segmentation under a sub-sampled setting.

Comparison Between Models The FC model is able to quickly gain performance at the start of training, but the performance also saturates early on. This indicates that the dataset and task requires a more complex model to learn. We observed that both the performance of MSTCN and DTGRM suffered from unbalanced verb class labels. This is a problem introduced by the EPIC-KITCHEN dataset, increasing the difficulty of the action segmentation task.

DTGRM has shown better performance in segmenting short action instance. However, on long action instances, DTGRM does not perform as well as MSTCN as the predictions of DTGRM tends to be heavily over-segmented.

Metrics We experimented with two variants of loss function: cross-entropy loss with and without weighting. Since background frames comprise a sizable portion, one phenomenon we observe dur-



(a) P26_39

(b) P16_04

Figure 3: Qualitative results of two videos across all models.

ing training is that models tend to classify most of the frames as background, at least in the first few iterations. Down-weighting background classes can mitigate this issue, but inconsistency still exists between the loss function and metrics we used, since classifying frames as the most common verb class (e.g. background, *wash*) can quickly increase frame-wise accuracy until some threshold (usually the percentage of these common verb class). Edit score is a better reflection of the fragmentation issue. Models like MSTCN that incorporate temporal information tend to have higher Edit score than simpler models such as FC.

6.2 Ablations and Their Implications

6.2.1 Video length

Variation of video length for EPIC-KITCHENS dataset is much higher than that of other datasets. The maximum video length is more than 1 hour while average is only 9 minute. To determine how video length affects performance of models, we divided the whole dataset into long videos that are longer than 10 minutes and short videos that are shorter than 10 minutes.

6.2.2 Vocabulary size

EPIC-KITCHENS contains a total of 97 verb classes, which is much more than that of 50Salads and Breakfast, which contain 17 and 48 verb classes respectively. To see if richer verb class will affect the performance of the baseline models that are originally proposed for 50Salads and Breakfast, we select a subset from 97 verb classes containing 54 verb classes with higher correlation to cooking. The remaining verb classes are labeled as background.

6.3 Qualitative Analysis and Examples

We choose two videos, "P26_39" and "P16_04", to evaluate the models qualitatively. The ground truth segmentation of "P26_39" (3a) consists of segments of long-duration, whereas "P16_04" consists of a large number of very short segments. We can see that it is very challenging for FC to get the class label correctly, as it predicts the most common verb "wash" (olive-green) instead of the ground truth "mix" (hot pink). For videos with a few number of long segments, DGTRM tends to over-segment, possibly because of its more complicated architecture compared to the other models. However, the tendency to over-segment, or being very sensitive to visual changes in frames, helps it to handle videos with very short segments interleaved together, as in 3b. The advantage of larger models like MSTCN and DGTRM over FC is clear from 3: both MSTCN and DGTRM have learned to predict the ground truth class and give more accurate segmentations. The over-segmentation issue of FC, which is much more severe than DGTRM, can be seen in 3b.

References

J. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien. 2016. Unsupervised learning from narrated instruction videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4575–4583.

T. Baltrušaitis, C. Ahuja, and L. Morency. 2019. Multimodal machine learning: A survey and taxonomy. volume 41, pages 423–443.

J. Carreira and A. Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2020. Rescaling egocentric vision. *CoRR*, abs/2006.13256.

Y. A. Farha and J. Gall. 2019. Ms-tcn: Multi-stage temporal convolutional network for action segmentation.

In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3570–3579.

A. Fathi and J. M. Rehg. 2013. Modeling actions through state changes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2586.

A. Fathi, X. Ren, and J. M. Rehg. 2011. Learning to recognize objects in egocentric activities. In *CVPR 2011*, pages 3281–3288.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. Slowfast networks for video recognition.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. 2018. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6546–6555.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Y. Huang, Y. Sugano, and Y. Sato. 2020. Improving action segmentation via graph-based temporal reasoning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14021–14031.

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

H. Kuehne, J. Gall, and T. Serre. 2016. An end-to-end generative framework for video segmentation and recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8.

C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. 2017. Temporal convolutional networks for action segmentation and detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1003–1012.

Colin Lea, Austin Reiter, Rene Vidal, and Gregory D. Hager. 2016. Segmental spatiotemporal cnns for fine-grained action segmentation.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked cross attention for image-text matching.

S. J. Li, Y. AbuFarha, Y. Liu, M. M. Cheng, and J. Gall. 2020. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nicholas Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. What's cookin'? interpreting cooking videos using text, speech and vision. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 143–152, Denver, Colorado. Association for Computational Linguistics.

Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2021. Thinking fast and slow: Efficient text-to-visual retrieval with transformers.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *ICCV*.

M. Pardowitz, Robert Haschke, Jochen Steil, and H. Ritter. 2009. Gestalt-based action segmentation for robot task learning. pages 347 – 352.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks.

A. Richard, H. Kuehne, and J. Gall. 2017. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1273–1282.

M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. 2012. A database for fine grained activity detection of cooking activities. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1194–1201.

Ling Shao, Ling Ji, Yan Liu, and Jianguo Zhang. 2012. Human action segmentation and recognition via motion and shape analysis. *Pattern Recognition Letters*, 33(4):438–445.

Botian Shi, Lei Ji, Zhendong Niu, Nan Duan, Ming Zhou, and Xilin Chen. 2020. Learning semantic concepts and temporal alignment for narrated video procedural captioning. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 4355–4363, New York, NY, USA. Association for Computing Machinery.

Sebastian Stein and Stephen J. McKenna. 2013. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, page 729–738, New York, NY, USA. Association for Computing Machinery.

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. 2019. Fcos: Fully convolutional one-stage object detection.

Dong Wang, Di Hu, Xingjian Li, and Dejing Dou. 2020. Temporal relational modeling with self-supervision for action segmentation.

Yaxiong Wang, Hao Yang, Xueming Qian, Lin Ma, Jing Lu, Biao Li, and Xin Fan. 2019. Position focused attention network for image-text matching.

Jihun Yoon, Seungbum Hong, Sanha Jeong, and Min-Kook Choi. 2020. Semi-supervised object detection with sparsely annotated dataset.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pages 7590–7598.

Appendix A Data Analysis

In this section, we present the full details of our data analysis.

	Training				Validation			
	Max.	Min.	Avg.	Std.	Max.	Min.	Avg.	Std.
Verb class frequency	14848	73	1314	2829	1937	71	191	398
Noun class frequency	3617	178	724	655	430	25	108	92
Sentence length	77	3	15.1	6.3	71	3	14.8	6.0
Actions per video	940	1	136	168	564	3	70	93
Frames per verb class	2129212	20165	225170	408408	407425	2702	42016	76950
Video length	3708	10	543	645	1969	11	344	377

Table 2: Statistics of EPIC-KITCHENS-100 training and validation set

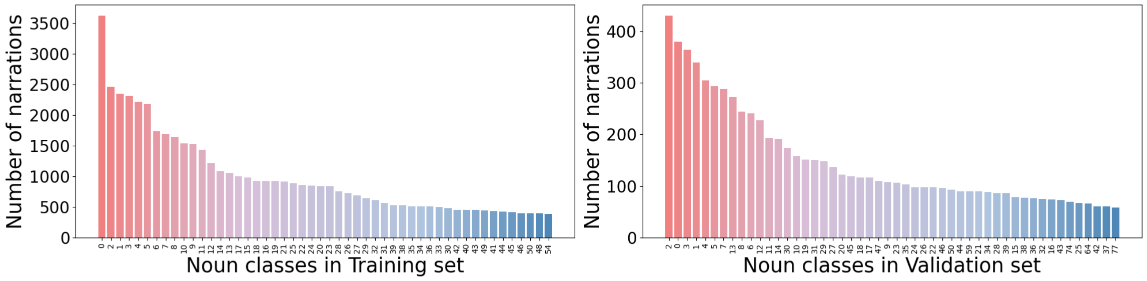


Figure 4: Frequency distribution of 50 most frequent noun classes in training and validation set

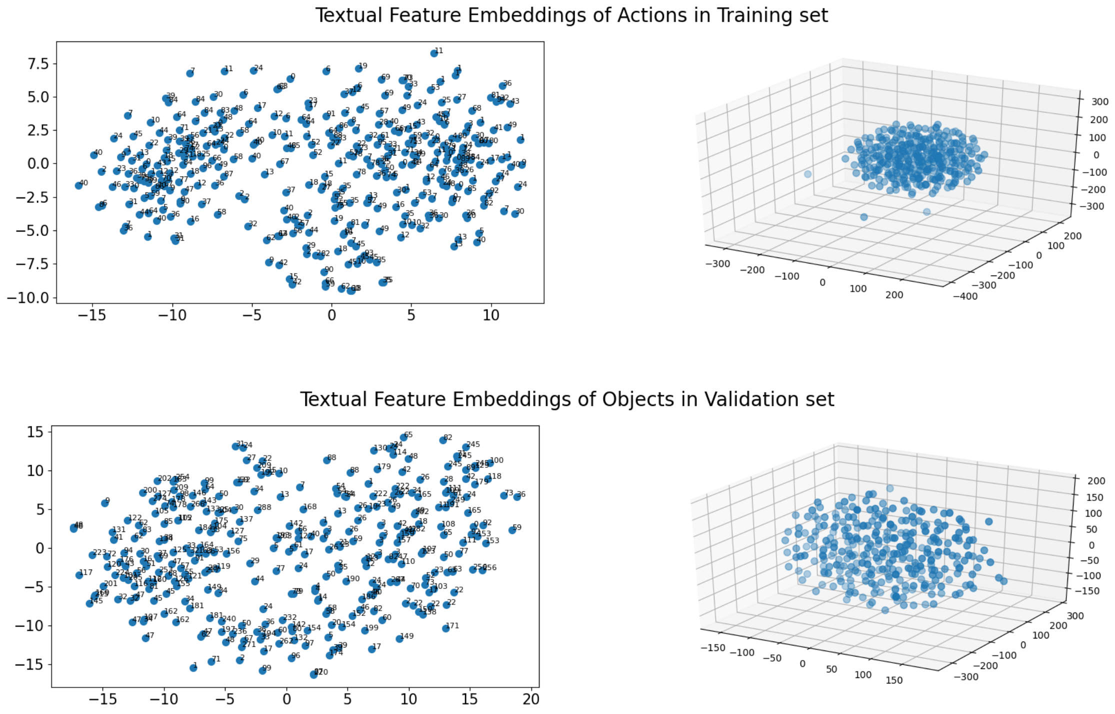


Figure 5: Example of visualizing feature embeddings of verb and noun classes in 2D and 3D space

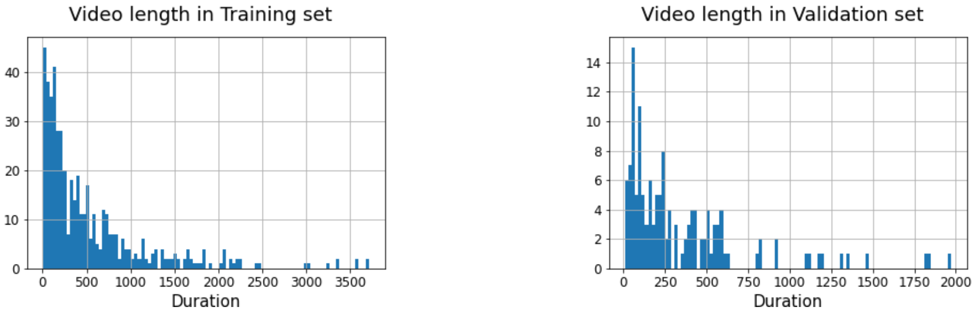


Figure 6: Distribution of video length (in seconds)

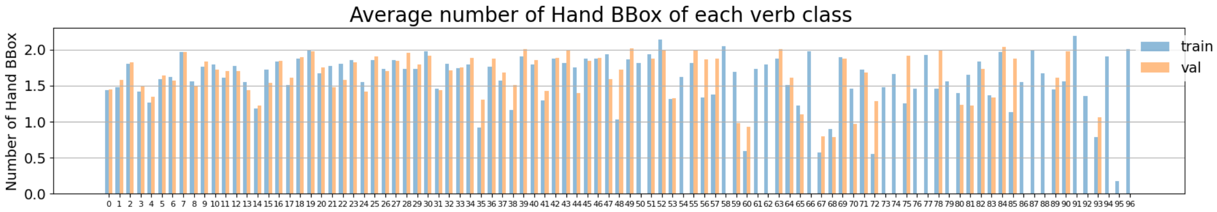


Figure 7: Average number of hand bounding-boxes in each frame of given verb class

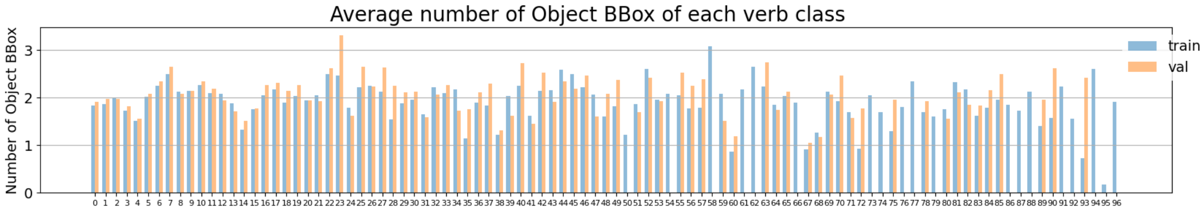


Figure 8: Average number of object bounding-boxes in each frame of given verb class

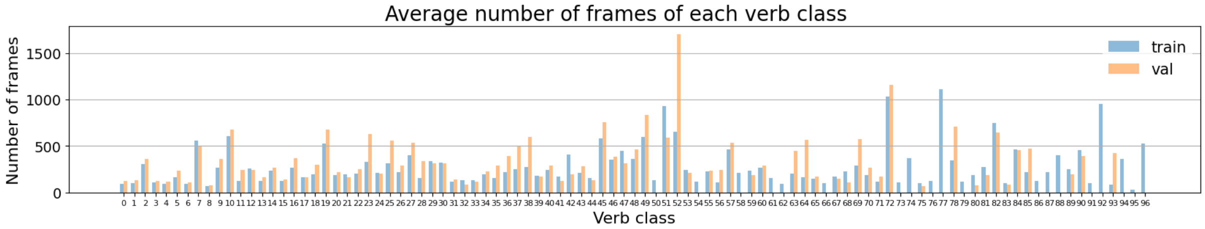


Figure 9: Average number of frames in a narration of a given verb class in training and validation set

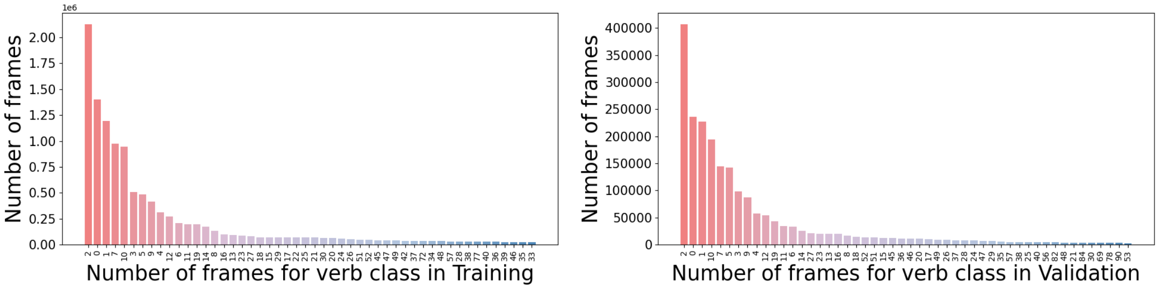


Figure 10: Distribution of number of frames in each video