Supplementary Material for
Temporal Alignment Networks for Long-term Video

Tengda Han[1] Weidi Xie[1,2] Andrew Zisserman[1]
[1]Visual Geometry Group, University of Oxford [2]Shanghai Jiao Tong University
https://www.robots.ox.ac.uk/~vgg/research/tan/

Contents

1. Automatic Data Curation for HowTo100M

In this section, we describe the procedure for automatically curate HowTo100M dataset, that targets three problems: Incorrect Language Translation, Incorrect Linebreaks and Incorrect Sentence Partition.

Incorrect Language Translation refers to the cases that the YouTube ASR system fails to detect the language and treats speech as English by default, thus generating incorrect text for non-English languages (as shown in Figure 1-a, the original language of speech is Thai). In order to detect such cases, we use open-sourced language detection library [13] which is a language classifier trained from Wikipedia that supports 53 languages. The language detection library takes a phrase or sentence as input, and returns a probability normalized over 53 languages. Specifically, for each HTM video, we randomly sample 5 subtitles, and average their probability of being English language, denoted as $\bar{p}_{\text{en}}$, Then we discard the videos with $\bar{p}_{\text{en}} < 0.9$, which accounts for about 3% of the entire dataset.

Incorrect Linebreaks refers to the cases where repetitive sentences exist in consecutive subtitles, as shown in Figure 1-b. We detect the linebreaks, and remove the duplicated sentence segments. In total, this operation corrects the captions for 68% of the whole dataset.

Incorrect Sentence Partition refers to the cases that the YouTube ASR system recognises unpunctuated sentences and wraps sentences to fit the width of the window, which generates incompleted sentence fractions. To restore completed sentences, we first remove all the linebreaks and combine the sentence fractions into an unpunctuated paragraph, then use an off-the-shelf BERT-based model to restore the punctuation [10] (the model is also trained without manual labelling), where the completed sentences are cut out at the full stop. We update the start and end timestamps for the completed sentence by interpolating from the word timestamps.

Overall, the three automatic data curation steps filter out incorrect subtitles and improve the quality of subtitles. Note that all the modules used in the curation are trained without human labelling, thus can be easily scalable.

2. Details of HTM-Align Dataset

In this section, we describe the annotation process, and show some examples from our **HTM-Align** dataset.

2.1. Annotation Details

In this paper, we use the open-sourced VIA [1] for annotation. A screenshot of the annotation process is shown in Figure 2. Given the video and its subtitles with start-end timestamps, the annotator mainly performs two tasks: (1) determine if a subtitle sentence is alignable with the video (i.e. visually related), (2) adjust the start-end timestamps to cover the visual content that is described by the "alignable" sentence. Overall, the completed annotation contains two types of subtitles. For alignable subtitles, the

(a) ZJcle-JvC_Y, 01:52 --> 02:32

```
"slow coty's pazham goku heil economy"
'hatta bernie medicine mohamed'
'design ah travel car'
'participant come home man hello come'
```

(b) Ua07eqNaLOA, 01:11 --> 01:25

```
"and very little bit of salt and that is\nabout it let's begin the process in a",
"about it let's begin the process in a\nbowl let's add some capsicum and the DC",
"bowl let's add some capsicum and the DC\nDec tomatoes these hearings the tomatoes",
"Dec tomatoes these hearings the tomatoes\nis very very important otherwise your"
```

Figure 1: Two problematic subtitle samples in HowTo100M dataset. Subtitles within a short temporal window are shown. **(a)** The presenter speaks in Thai, but the text is translated to English by ASR, resulting in nonsense text. **(b)** The linebreak "\n" appears in every sentence segment, resulting in repeated text segments.

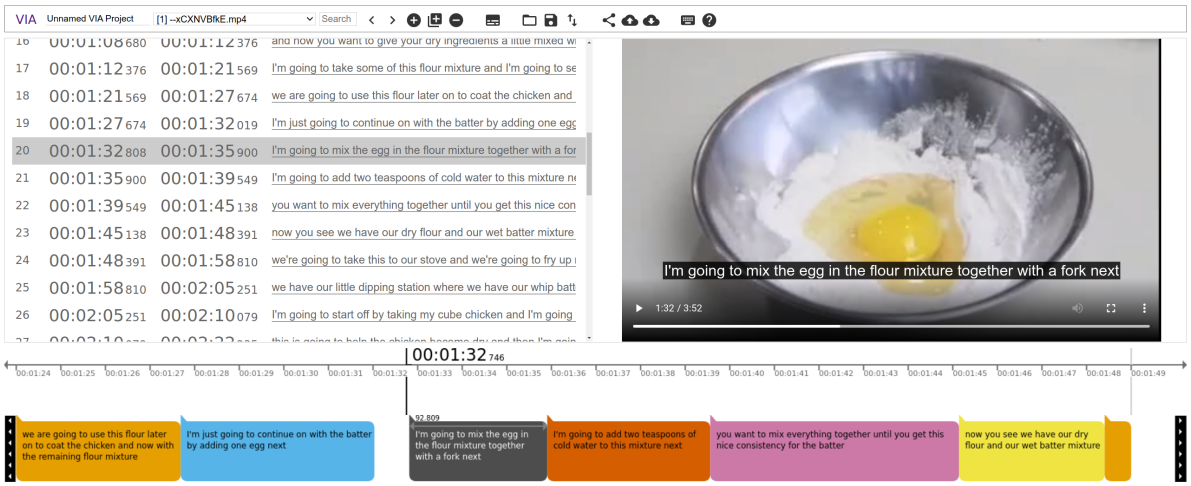

Figure 2: A screenshot of the annotation process for the HTM-Align dataset. Two tasks are performed during annotation. The annotator (1) verifies the alignability of a subtitle sentence, and if alignable, (2) adjusts the start-end timestamp to align subtitle sentence with the visual scene.

sentence with its aligned start-end timestamp is provided, for non-alignable subtitles, the sentence with its original start-end timestamp is simply inherited.

2.2. Examples

Here, we show two example videos with the aligned annotations from **HTM-Align** dataset in Figure 3. More example videos are included in the submitted zip file. We show video on the top, the alignment annotations in the middle, and the TAN outputs at the bottom. Note that the audio is copied from YouTube files only for presentation purpose, in order to denote the raw timestamps of the YouTube subtitles. Our model does not take audio input.

3. Implementation Details

3.1. Architectural Details

Visual Backbone. As introduced in paper Section 4.2, we adopt a pre-trained S3D [16] (pre-trained with MIL-NCE [7]) as the video backbone. Specifically, we decode the video with 16fps, feed S3D with 16-frame non-overlapping temporal window, and take the feature vector (1024D) just before the final fully-connected layer of S3D. The original video is firstly resized to have shorter size equal to 256 pixels, then the frames are center-cropped with 224×224 resolution before feeding into the S3D. The 1024D feature vectors are projected to 512D with a fully-connected layer before feeding into the transformer.

Language Backbone. As mentioned in the main paper Section 4.2, we adopt a Bag-of-word (BoW) model based on Word2Vec embeddings. Specifically, following [7], we use the Google-News self-supervised pre-trained Word2Vec embeddings (300D) from [9]. For each sentence, we take a maximum of 32 words and the word embedding is independently passed to two fully-connected layers and projected to 16×512D, and a max-pooling is applied to get a sentence embedding (512D).

Transformer Modules. As shown in the main paper Figure 2, the TAN is consists of a *Multimodal Transformer*, while the auxiliary dual encoder contains a *Video Transformer*. For both of them, we use the pre-norm multi-layer transformer encoder implemented in [11]. As for the depth of transformer, we use a 6-layer transformer by default, and we also ablate different depth (3-layer) in main paper Section 5.2. The transformer uses 8-head attention mechanism, takes hidden vectors with 512D, and uses 2048D feed-forward dimensions. For the video features, both transformers share a learnable position encoding to inject temporal information. For the language features, we do not use po-

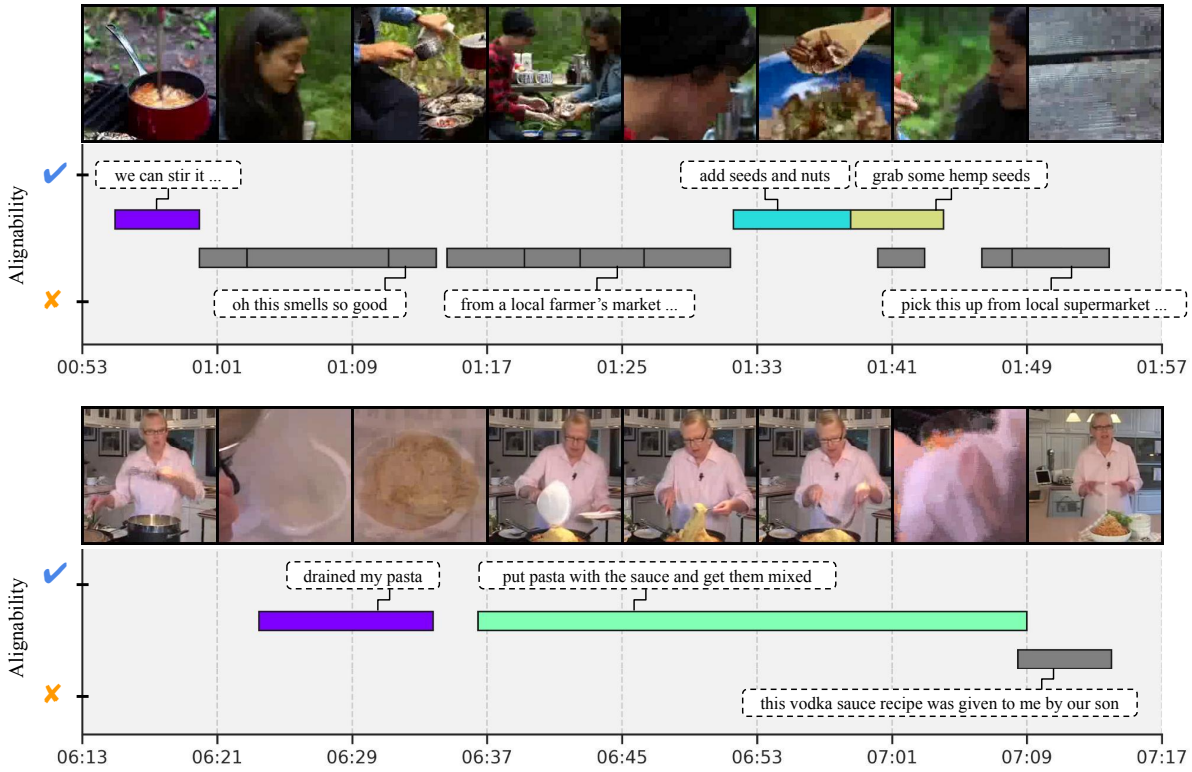

Figure 3: Example videos from **Aligned-HTM** dataset. We visualize two 64-second video segments. The horizontal axis denotes the original video timestamp. ✓ indicates the narrations with positive alignability, ✗ refers to the texts that are visually NOT alignable, the width of the colored rectangle indicates the start-end timestamps of the text, which are all provided in the dataset. For some of the non-alignable sentences we only show the rectangle but omit the text for clarity.

sitional encoding, as we found it tends to lead trivial solutions, where the model learns the temporal alignment *only* by positional encoding.

3.2. Details for Downstream Datasets

Breakfast-Action [5] includes 1.7k videos from a third-person-view for the breakfast preparation activities in kitchen. There are in total 48 different actions, plus a background action. Here, action classes are only short phrases like 'fry egg' and 'pour milk', and each video contains 6 action instances on average.

YouCook2 [17] contains 2k long videos covering 89 cooking recipes. The key procedures of the recipe are localized and annotated with sentences. On average, each video has 8 annotated segments, totalling 14K action segments. We mainly report the text-video retrieval performance on about 3.5k validation segments.

UCF101 [14] contains 13k short (4-14 seconds) video clips spanning 101 human actions.

HMDB51 [6] contains 7k short (1-6 seconds) video clips spanning 51 human actions.

K400 [4] stands for Kinetics-400 dataset, which contains about 300K 10-second video clips for 400 human actions.

3.3. Details for Downstream Tasks

Temporal Alignment on Breakfast-Action. We evaluate this alignment task using TAN (without the auxiliary dual encoder), as explained in main paper Section 6. We first apply the same data pre-processing as in pre-training on HTM-370K, i.e. decoding video with 16fps and extract S3D features with center-cropped 16-frame input, resulting in 1 feature per second without temporal overlap. For the language encoding, we use the same procedure and extract single vector (512D) for each action class like 'crack egg'. Breakfast-Action dataset provides a background action class, which usually appears at the start and end of the video. Note that we only take the non-background video segments (i.e. the middle part of video) to evaluate the temporal alignment task. For a single video, the visual features with T time steps and a list of K text features are fed into the transformer module, resulting with the alignment matrix $\hat{\mathbb{A}} \in \mathbb{R}^{K \times T}$. In practice, T can be longer than 1 minute. To fit our positional encoding that is only trained for 64 seconds, we crop visual feature with a 64-second (64 time

stamps) sliding window, and stitch multiple $\mathbb{R}^{K\times 64}$ alignment matrices as the final $\hat{\mathbb{A}}$. The alignment matrix $\hat{\mathbb{A}}$ is then passed to Dynamic Time Warping (DTW [12]) to get the action boundaries.

Text-based Video Retrieval on YouCook2. The validation split of YouCook2 contains about 3.5K paired video segments and action descriptions. We evaluate this retrieval task with the auxiliary dual encoder, as explained in main paper Section 6. For visual encoding, we first use the same data pre-processing as in pre-training on HTM-370K, *i.e.* extracting S3D video features with 1 feature per second. Following the prior works [3, 7], we pre-crop the original long video to obtain short video segments based on the annotated start-end timestamps for each annotated sentence. Each video segment is fed into the video transformer of the *dual encoder* to get the visual output ($\hat{v}_d \in \mathbb{R}^{t\times 512}$) where t denotes the duration of the segment, then a temporal average pooling is applied to get a 512D vector, representing each video segments. For textual encoding, we extract the sentence feature (a single 512D vector) for each action description, such as 'chop lettuce and place it in a bowl'. Finally, each sentence embedding retrieves segment-wise video features by cosine similarity and the metric is calculated following [7].

4. End-to-end Representation Learning with auto-aligned HTM

4.1. Details for End-to-end Training

We conduct the end-to-end trainings on on 25% of all the original HowTo100M raw videos, First, we use the model-H from the paper Table 1 to compute the text-video alignment. For each text, we store the timestamp with the highest similarity as well as the alignability score. For data sampling, we drop the texts with the lowest 50% alignability score, and load the auto-aligned text-video pair for training. To introduce more hard negatives for Info-NCE loss, we randomly sample 2 text-video pairs from each long-video. For each video clip, we follow [7] and use 16 frames with 5 fps. In this way, we fit a batch size of 32 text-video pairs (come from 16 source videos). We load the official S3D-word2vec backbone from [7], and finetune the backbone on our auto-aligned HowTo100M subset for only 40 epochs – equivalent to 10 epochs on full-set HowTo100M in terms of the number of iterations. We use an AdamW optimizer with 10^{-4} learning rate and a cosine decay learning rate schedule for training.

4.2. Details for Linear Probe Evaluation

We keep the S3D backbone frozen, and only train a linear layer on the 1024-D visual feature with a cross-entropy loss. We use AdamW optimizer with 10^{-2} initial learning rate with cosine decay learning rate schedule and 10^{-3} weight decay to train the linear layer. We both UCF101 and HMDB51, we use the split-1. The network takes input as 16 frames video clips with 5 fps. The linear layer was trained for 200 epochs on UCF101, 100 epochs on HMDB51, 35 epochs on K400 respectively. For inference, we take the sliding windows along the time axis and apply spatial ten-crop strategy (4 corner crops and 1 center crop, with/without flipping), and average the probability. As shown in the paper Table 4, after finetuning on the automatically-aligned HowTo100M subset, the linear probe performances on all three datasets are clearly improved. Note that, such procedure for alignment and end-to-end finetuning can be iteratively conducted to learn stronger video representations. We leave these for future works.

5. Qualitative Results

In this section, we show two qualitative examples for temporal alignment, and there are more detailed qualitative results in video format in our *submitted zip file*.

Two examples from HTM-Align are shown in Figure 4. The first row shows the *manually-aligned* timestamps of each sentence, where some sentences are annotated as 'not alignable'. Note that in example (a) the ground-truth textual-visual alignment does not follow the monotonic temporal order, this could happen in natural instructional videos but DTW-type approaches cannot handle [2]. This ordering issue is also discussed in the main paper Section 1. The second row shows the $K \times T$ heat-map from MIL-NCE [7] backbone features. The third row shows the alignment matrix $\hat{\mathbb{A}}$ from our Temporal Alignment Network. For the clarity of visualization, we normalise the similarity scores over the time axis with a softmax operation.

The qualitative results show that our method gives a much cleaner temporal alignment than MIL-NCE (row-3 vs row-2), and our temporal alignment is close to human judgement (row-3 vs row-1). The TAN also predicts an accurate alignability for each sentence that roughly matches human judgement. In detail, the binary prediction is obtained by simply thresholding the probability with 0.5 threshold. Our TAN is trained on the noisy HowTo100M as same as MIL-NCE, but is able to learn a good temporal alignment by the proposed denoising mechanism, showing the effectiveness of our method.

6. Limitations and Ethical Concerns

Our work has the following limitations: (1) The current model can only process videos of about 1-minute long at a time, and is limited by the memory of existing GPUs hardware if we intend to align longer video sequence or train the system end-to-end beyond this duration; (2) Our alignment method cannot handle repetitive text or repetitive visual content in a video (for example in the zipped video

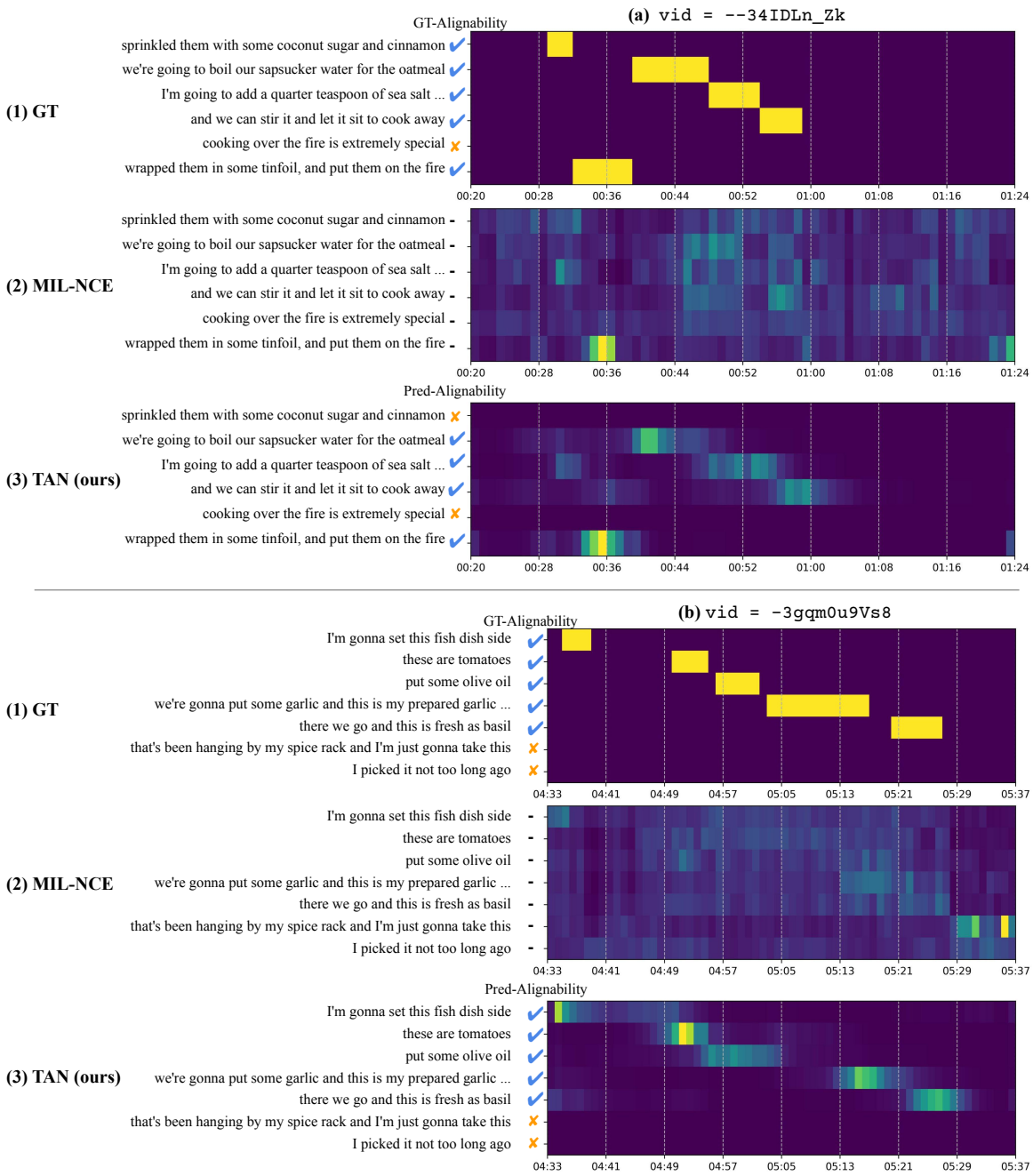

Figure 4: Textual-visual heat-maps from MIL-NCE [7] and our TAN, computed on two examples from HTM-Align dataset. The raw sentence is shown on the y-axis. **(1)** The first row shows the ground-truth textual-visual alignment by manual annotation, which is a $K \times T$ *binary* map where the lighter region means the text is aligned to certain video segments. The alignability of each text is also annotated. **(2)** The second row shows the $K \times T$ textual-visual similarity map from the MIL-NCE backbone features. Note that MIL-NCE does not classify the alignability of each sentence. **(3)** The third row shows the $K \times T$ textual-visual similarity map ($\hat{\mathbb{A}}$) from our TAN. Our model also predict the alignability of each sentence, which is shown on the figure.

`tan-x2--0XlcSx1lNs.mp4`, the action 'stir fry' occurs multiple times and is not aligned well), as one sentence can potentially be aligned to multiple video segments and vice versa. Note, this is a general problem in the tasks that require visual-language correspondence, for example, in retrieval task [15]. In practice, we find our proposed model does not suffer from this limitation though, as repetitions are uncommon in natural instructional videos.

For ethical concerns, we are aware that we use a public instructional video dataset [8] that uploaders shared on

YouTube, which might have gender, age, geographical or cultural bias, also inevitably human faces appear in the dataset.

References

[1] Abhishek Dutta, Ankush Gupta, and Andrew Zissermann. VGG image annotator (VIA). http://www.robots.ox.ac.uk/~vgg/software/via/, 2016. 1

[2] Nikita Dvornik, Isma Hadji, Konstantinos G. Derpanis, Animesh Garg, and Allan D. Jepson. Drop-dtw: Aligning common signal between sequences while dropping outliers. In *NeurIPS*, 2021. 4

[3] Jianfeng Gao Jianwei Yang, Yonatan Bisk. Taco: Token-aware cascade contrastive learning for video-text alignment. In *Proc. ICCV*, 2021. 4

[4] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 3

[5] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proc. CVPR*, 2014. 3

[6] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *Proc. ICCV*, pages 2556–2563, 2011. 3

[7] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Proc. CVPR*, 2020. 2, 4, 5

[8] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *Proc. ICCV*, 2019. 5

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *NIPS*, 2013. 2

[10] Daulet Nurmanbetov. Bert-restore-punctuation model from huggingface. https://huggingface.co/felflare/bert-restore-punctuation, 2021. 1

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 2

[12] H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1978. 4

[13] Nakatani Shuyo. Language detection library. http://code.google.com/p/language-detection/, 2010. 1

[14] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 3

[15] Michael Wray, Hazel Doughty, and Dima Damen. On semantic similarity in video retrieval. In *Proc. CVPR*, 2021. 5

[16] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *Proc. ECCV*, 2018. 2

[17] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018. 3