Efficient Temporal Action Localization via Vision-Language Modeling

RQ: ***How well do current CLIP-based TAL methods perform and generalise in a limited compute power and data setting?***

Yunhan Wang ✉ y.wang-128@student.tudelft.nl

Attila Lengyel
Ombretta Strafforello
Robert-Jan Bruintjes
Jan van Gemert

Computer Vision Lab,
Delft University of Technology

1. Background

Temporal action localization (TAL) aims to localize the start and end time of actions in untrimmed videos and classify the action types. A video clip can include more than one action instance and background frames without actions.

Vision-language models: using contrastive learning to pair images with natural language signals, CLIP [1] has shown that paired image-caption data can be leveraged to learn powerful visual representations for zero-shot recognition. CLIP was extended for TAL by modelling temporal information such as the STALE [2] and Eff-Prompt [3] models. STALE was selected to evaluate due to its state-of-the-art performance.

4. Discussion

- The CLIP-based STALE model demonstrates adequate generalizability with limited training data amount in the close/open-set scenarios.
- STALE has a considerable gap in mAP performance compared with recent transformed-based TAL models such as ActionFormer [5].
- During model inference, STALE can utilize GPU better when the input video length is small. We recommend setting the input video length to smaller than or equal to 200 for fast inference and higher computational efficiency.
- 8-bit model quantization can remarkably reduce the time consumption of forward pass on STALE with a single CPU.

5. Reference

[1] Learning transferable visual models from natural language supervision. ICML 2021.

[2] Zero-shot temporal action detection via vision language prompting. ECCV 2022.

[3] Prompting visual-language models for efficient video understanding. ECCV 2022.

[4] UntrimmedNets for weakly supervised action recognition and detection. CVPR 2017.

[5] ActionFormer: localizing moments of actions with transformers. ECCV 2022

2.1 Method: Model

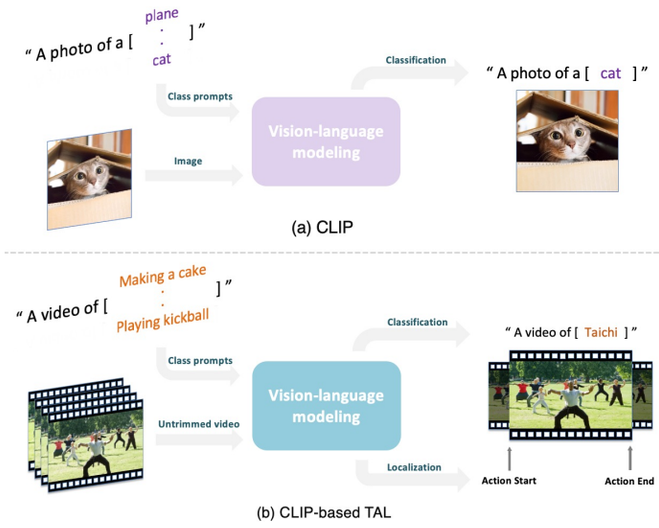

Fig 1. Illustration of the CLIP model (a) and a one-stage pipeline for CLIP-based TAL (b). The classes are treated as textual prompts and are trained to pair with vision instances. In (b), the outputs are the action class and action start/end time for each action instance in each untrimmed video.

2.2 Method: Data efficiency

Fig 2. Illustration of closed/open-set scenarios. Data efficiency experiments consists of both scenarios with limited training data (p). Subsets are sampled uniformly without replacement. Experiments are performed on the ActivityNet dataset.

2.3 Method: Compute efficiency

(1) Examining compute efficiency: record time consumption, memory consumption, GPU utilization percentage, MACs during model inference by varying input video length.

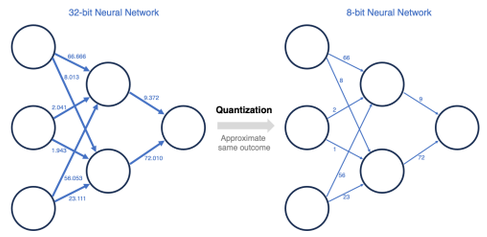

Fig 3. Illustration of quantizing a model from 32-bit to 8-bit to reduce floating computation. We quantize all STALE's linear and convolutional layers.

(2) Examining quantization effect: record time consumption of different model precisions during model forward pass on CPU/GPU by varying input video length.

3.1 Results: Data efficiency

Class Split	Enhanced Score	Avg Training Data Amount	ActivityNet v1.3 Average-mAP
75% Seen 25% Unseen	with	6575	21.7
		3304	21.7
	w/o	6575	8.4
		3304	8.5

Table 1. Comparison of open-set mAP results of STALE with/without the use of score enhancement and trained on 100% or 50% of data. Score enhancement: adopt classification from UntrimmedNet [4] for uncertain predictions.

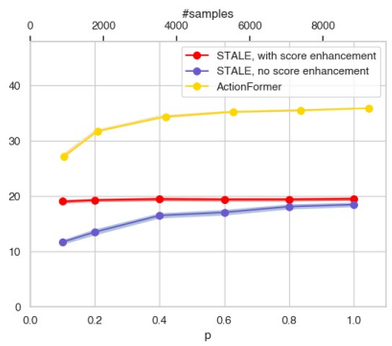

Fig 4. Closed-set average-mAP comparison with different p training data amounts between STALE with/without score enhancement and ActionFormer [5]. Without score enhancement, STALE trained with 7000 samples can even produce comparable results to STALE trained with 8840 samples.

3.2 Results: Compute efficiency

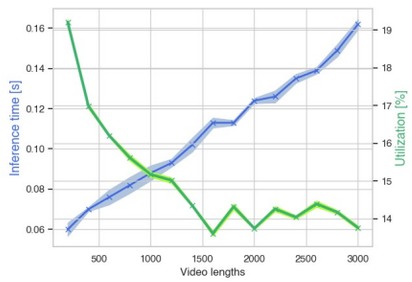

Fig 5. Inference time and GPU utilization with varying input video lengths. GPU utilization gradually drops as we increase video lengths and converge to around 14% since a video length of 1400.

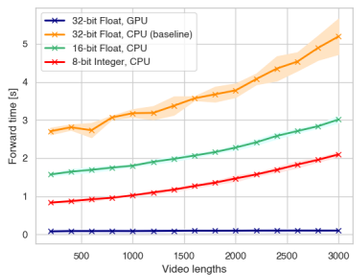

Fig 6. Effects of quantization on model forward pass time across different precisions on a CPU/GPU. The forward time on GPU has a linear increasing tendency from 0.080s at length 200 to 0.101s at length 3000. Quantizing STALE into 8-bit precision can drastically reduce forward time on the CPU compared to the original 32-bit.