Supplementary Material for "Holistic Interaction Transformer Network for Action Detection"

Gueter Josmy Faure[1] Min-Hung Chen[2] Shang-Hong Lai[12]
[1]National Tsing Hua University, Taiwan [2]Microsoft AI R&D Center, Taiwan
josmyfaure@gapp.nthu.edu.tw vitec6@gmail.com lai@cs.nthu.edu.tw

In the supplementary material, we aim to provide deeper technical details about our framework, and show more qualitative results.

1. More Technical Details

1.1. Temporal Interaction (TI): Memory retrieval

As stated in the paper, we use a memory span $S = 30$ and the total number of cached frame features passed to the TI module is $2S + 1$ (features of the current frame and 30 frames from each side of the current frame). Inspired by [4] and [5], we keep a cached memory pool $\mathcal{M}$ with temporal person features, and at the end of each iteration, we update the cached person features. This dynamic approach does not increase the computational cost as the number of frame increases. It also allows us to train the model end-to-end since we don't need to pre-train and freeze memory features as is the case with [2] and [1].

1.2. How we chose QKV for the interaction modules

By definition of Cross Attention, the query should stem from the main features and the key and value from the helping features. In our RGB stream, we aim to model interaction between the target actor and the supporting actors (when available) and the context (object and hands). Therefore, the target person is constitutes our main feature. The other features serving as key and and value help us find better representations for the person features (query). Specifically, the hands interaction sub-network outputs better person features after looking at how the target person interacts with the hand features. *Theoretically, the features that will be passed to the final classifier should be the query*.

1.3. Intra-Modality Aggregation

The intuition for the intra-modality aggregation components z_r and z_p is to filter and pass only the best of the previously computed features (including the previous block's output and all other interaction blocks) to the next interaction module. We experimented with channel-wise concatenation followed by convolution, feature average, and weighted sum to achieve this. The first is more computationally expensive (Convolution on a very high dimensional vector); the second is straightforward but assumes every feature is equally important. Therefore, we chose to use the weighted sum approach. The only feature fusion mechanism we found in the literature is channel-wise concatenation followed by PCA. However, it suffers from the same shortcomings as the concatenation + convolution approach. Many initialization schemes came to mind, especially Kaiming and Xavier's initializations. However, they do not work well for this task because they would magnify or reduce the magnitude of our input instead of letting the model learn as it sees fit. We ruled out random initialization since we want every feature to be equally weighted initially. Learning from scratch usually requires small weights, therefore we tried $-k \times ln(10)$ with k taking values $2, 5, 7$, then settled for $k = 5$ (initial weights ≈ -11.5).

1.4. Implementation details for AVA and Multi-Sports

We train AVA with two different activation functions for the classifier. For pose actions, we use a softmax classifier since there can only be one pose for each person. AVA is designed such that one person can have up to three of both person interaction and object interaction labels. Therefore, we use a sigmoid activation for person-interaction-related classes and another sigmoid for object-interaction-related ones. The network is trained for 110k iterations, with the first 2k iterations serving as linear warmup. We use a base learning rate of 0.0004, decreasing by a factor of 10 at iterations 70k and 90k. We use the SGD optimizer and a batch size of 16 to train the model on 8 GPUs. At inference, we predict action labels for human bounding boxes detected by the Faster-RCNN [3] instance detector with a confidence threshold of 0.8. We follow the same configurations as AVA to train MultiSports, except for the activation functions. MultiSports does not have overlapping actions;

therefore, we only use softmax activation for the classifier.

2. Extended Qualitative Results

All of the qualitative results presented here are based on frames extracted from the J-HMDB dataset. *Ours* refers to our proposed **HIT** network, and AIA is an interaction modeling framework presented in [5]. For a fair comparison, we implement AIA on top of the same backbone we use for our model. With Figure 1, we want to illustrate how our interaction framework compares to AIA [5] in terms of action sequence detection. As the subject is climbing the stairs, her action becomes more an more apparent to our model and the confidence score given to the action sharply increases. However, AIA's confidence score stays stagnant. Such a result highlights the strength of our temporal interaction modeling approach.

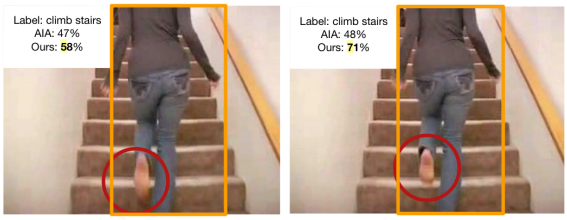


Figure 1: **Temporal Interaction.** Even though the encircled part is the only noticeable difference between frames t(left) and $t+1$(right), our model upgrades the confidence score, as it progresses through the video.

In Figure 2a, AIA misclassifies "brush hair" as "wave". Looking at the whole video, any human would detect that the subject is brushing their hair. However, it's not clear for that particular frame. Our model, with strong temporal support and atomic interaction modeling, is able to correctly detect this action. Figure 2b is challenging since the "throw" action that is happening here is competing with the actions "stand". Still, our model gives the bulk of the confidence score to the correct action label. Figure 2c is blurry, just like many of the frames that fall within the "jump" class. Pose features comes in handy for such actions. We can also see from Figure 3 that for that particular action class, our method significantly outperforms AIA. Figures 2d to 2f corroborate the arguments that pose interaction (2d, 2e) and hand interaction (2f) are essential for accurate atomic action detection.

Figure 3 is a detailed per-class comparison table between our method and AIA. Our method significantly outperforms AIA on challenging hand-related classes such as "throw", "catch" and "wave". It also registers strong performance against classes with fast movements ("jump", "run"). Overall, our HIT network outperforms AIA on every class, providing a significant upgrade on current interaction frameworks used in the literature.

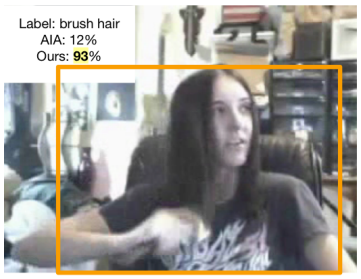
2.1. Failure cases

Most of the failure cases for our method fall into one of three categories. The first one is similar-looking classes. Classes such as "throw" and "catch" usually share the same pose signature and are visually identical. In Figures 4a and 4b, we see how close the class "throw" can be to the class "catch" and "kick ball", respectively. Looking at these frames, any human could misclassify these actions. For these kinds of classes, memory modeling could help. However, it is not guaranteed to work every time. To clear the "throw"-"catch" confusion, the question the memory features has to answer is the following: is the object coming from or heading to the person's hands? The second common failure has to do with incorrectly labeled classes. For instance, a person standing up is not a person who stands. Someone might kneel while standing up, as shown in Figure 4c. However, for the J-HMDB dataset, the label is the same throughout the video. Therefore, the act of kneeling while standing up is considered "stand" and should be classified as such even though it is visually wrong. The third category most methods struggle with is occlusion (see Figure 4d). Playing golf and swinging a baseball are almost identical, with the key difference being the object the actor is using. Is it a baseball bat or a golf club? In this case, however, the object is occluded. Therefore, the model finds it difficult to differentiate between "golf" and "swing baseball."

How should we then go about solving these issues? In our case, we try and aggregate as much information as possible. However, having so much information is costly. The best answer to these problems would be better temporal support, but that would raise another question: how do we define "better temporal support"?. While some might advocate for more extended temporal support, it would increase the computation overhead while not necessarily translating into higher detection accuracy. Some actions need long temporal support, some need very little, and others need none; therefore, deciding how much memory to keep around is challenging. And if we keep a longer memory span, the need to compress the feature would be more pressing, and most existing compression methods are lossy.

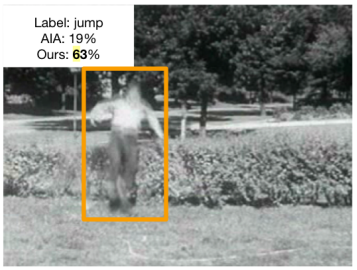
3. Complexity analysis

We perform a mini-experiment to determine the FPS of our model with different settings. We could not compare with other frameworks since they do not report these numbers. The backbone has a FPS of around 16, and paired with AIA [5], it is reduced to 13.66. Our model is marginally more complex than AIA due to the added features. However, it can be justified by the considerable improvements in accuracy.



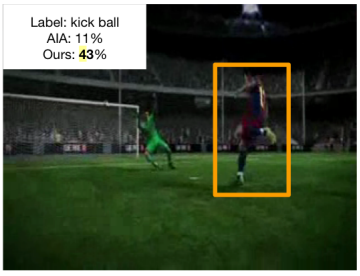
(a) AIA miclassifies the action in this frame as "wave".

(b) Frame with the action class "Throw" where the object being thrown is undetected.



(c) Blurry frame, typical of the action "jump". The second most challenging class in J-HMDB.

(d) A blurry frame but with clear pose signature.



(e) Frame from a video game. AIA evenly distributes the score between many different classes.

(f) "catch" is strongly hand-related.

Figure 2: **More Qualitative results.** Comparison with AIA

Model	frame@0.5	FPS
Backbone	58.85	**15.97**
Backbone + AIA[5]	77.25	13.66
Backbone + Ours	**83.81**	12.37

Table 1: **Performance and FPS comparison between the video backbone, AIA and our framework.**

References

[1] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[2] Junting Pan, Siyu Chen, Mike Zheng Shou, Yu Liu, Jing Shao, and Hongsheng Li. Actor-context-actor relation network for spatio-temporal action localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 464–474, 2021.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[4] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[5] Jiajun Tang, Jin Xia, Xinzhi Mu, Bo Pang, and Cewu Lu. Asynchronous interaction aggregation for action detection. In *European Conference on Computer Vision*, pages 71–87. Springer, 2020.

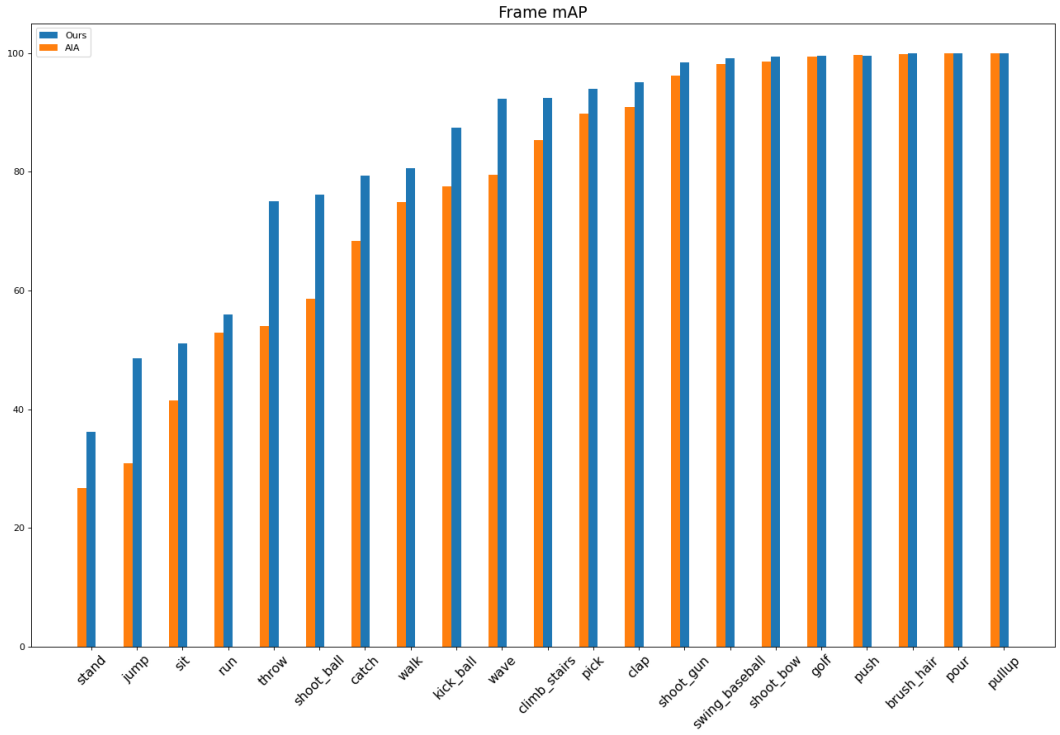
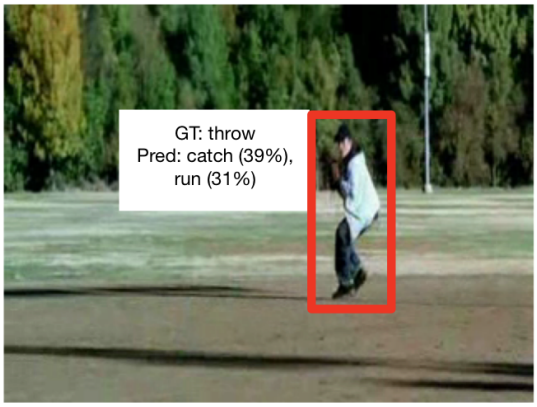
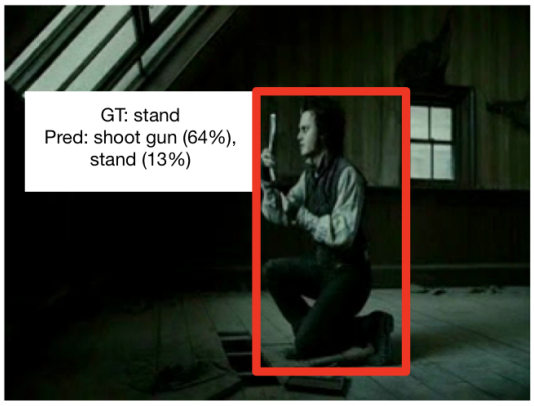


Figure 3: **Per-class frame mAP comparison with AIA.**



(a) Confusion between the classes “throw” and “catch”.

(b) Confusion between the classes “throw” and “kick ball”.



(c) The ground-truth is incorrect in this case.

(d) Due to the partial occlusion, “swing baseball” and “golf” look almost identical.

Figure 4: **Failure cases.** Most failure cases are due to similar-looking classes (a and b), incorrectly labeled classes (c), and partial occlusion (d).