Learning to Visually Connect Actions and their Effects—Appendix

Paritosh Parmar Eric Peh Basura Fernando
Centre for Frontier AI Research, Agency for Science, Technology and Research, Singapore
https://github.com/LUNAProject22/CATE

1. Dataset samples with and without object detections

For easier viewing, we have provided them in the accompanying **Video**.

2. Further details on Baseline Methods for Action Selection

In this section, we provide further details on the various baseline approaches that we design and evaluate. Actions are represented as video clips, while states by very short clips. Unless specified otherwise, we use a backbone model ($\mathcal{E}$) pretrained on a large-scale action recognition dataset to extract features for actions and states. Subsequently, various modules mentioned in the following operate on these features.

2.1. Naive model

Approach is illustrated in Figure 1. As a simple baseline, in naive matching the average of the initial and desired or the final state features are matched with the action features using cosine similarity. During inference, the action option with the highest similarity score is selected as the correct answer. Note that, this model involves no training; pretrained action recognition model is used as the state and action feature extractor.

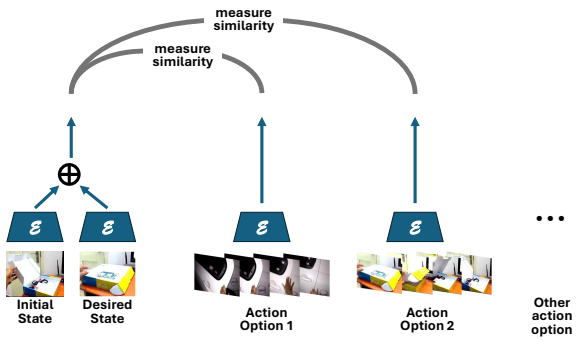


Figure 1. **Naive model.** This model involves no training, but only inference. $\bigoplus$ represents averaging operation.

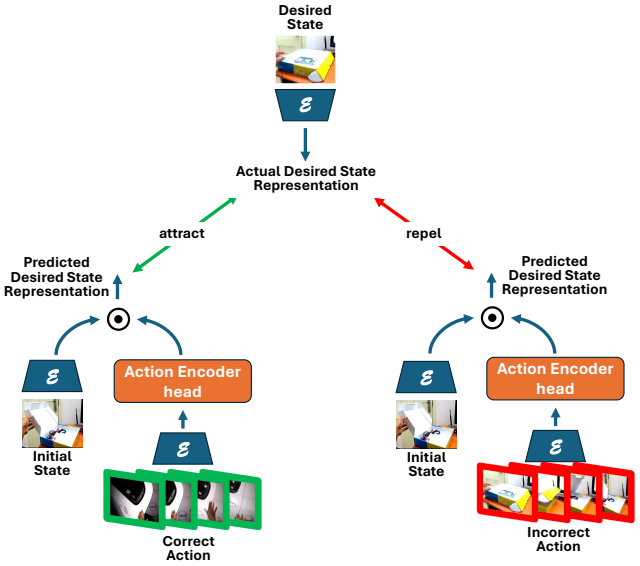


Figure 2. **Actions as Transformations model.** $\odot$ represents Hadamard product. Here, we have shown only one incorrect option; but in practice, during training, we consider all the incorrect options.

2.2. Treating Actions as Transformations

The approach (in training mode) is shown in Figure 2. Actions can be viewed as transformations [9] which when applied to the initial state yields the desired or the final state. While the original approach uses fixed transformations [9], our adaptation facilitates conditional transformations computed on the fly. We develop a model in which a transformation vector is computed from an action clip using a dense network. This transformation vector (A) when multiplied with the initial state vector (I) yields the final state vector (F). During training, we use contrastive learning [1] to train the parameters of action encoder: Cosine similarity between predicted and actual desired state representations is maximized for the correct action and minimized for the incorrect action. During inference, the action option predicting the final state with the highest similarity to the given final state feature is selected as the correct answer. We term

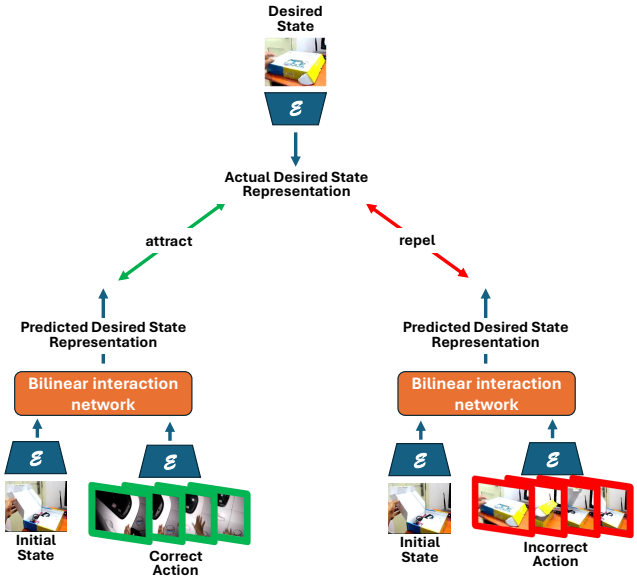


Figure 3. **MoRISA model.**

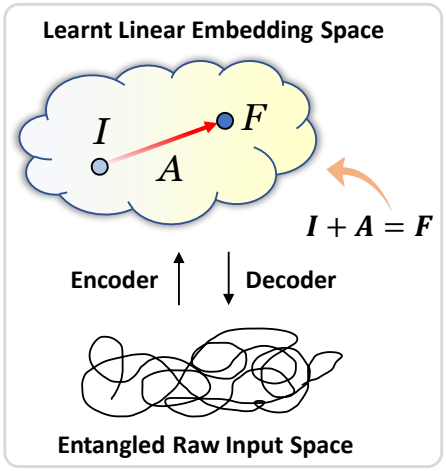
these models as Actions as transformations (AT).

2.3. Modeling rich interactions between initial state & action (MoRISA)

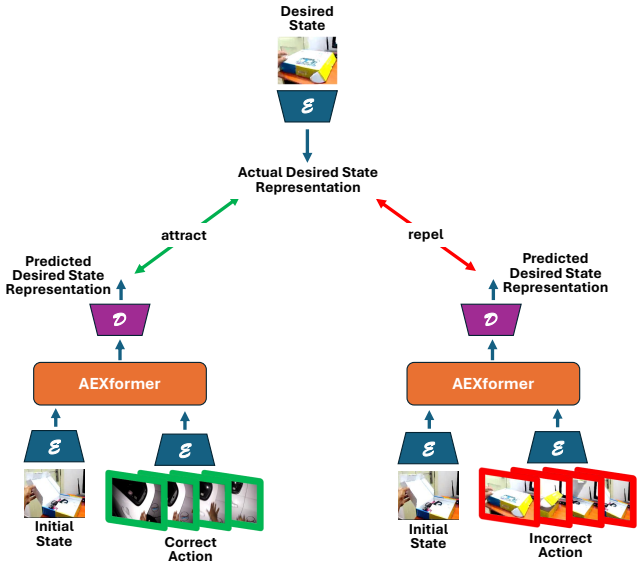
Approach (in training mode) is shown in Figure 3. Here a bilinear model [4] is leveraged to capture rich interactions between the initial state and the action to generate the final state in feature space. Essentially, initial state (I) and action representations (A) are fed into a bilinear model, which outputs a final state representation (F). The parameters of the model are trained using contrastive learning as before and the inference is done similarly to AT model above.

2.4. LinSAES: Learning Linear State-Action Embedding Space

Concept of this approach is shown in Figure 4a; and the framework (in training mode) is shown in Figure 4b. In this framework, actions and initial states undergo transformation into a linear embedding state-action representation space through a Transformer encoder (we term it as AEXformer in Figure 4b). The hypothesis suggests that within this *linear*, *disentangled* space, *adding* an action vector (A) to the initial state representation (I) enables a transition to the final state in the representation space (F) (illustrated in conceptual diagram in Figure 4a). The Transformer encoder processes a sequence of initial state and action representation vectors, and the resulting output is decoded by a linear decoder ($\mathcal{D}$) to produce the final state feature vector. During training, all model parameters are optimized end-to-end with the objective of aligning the resultant final state representation with the ground truth final state representation. Optimization and inference process remains the same as AT



(a) **LinSAES Concept.**



(b) **LinSAES model.**

Figure 4. **LinSAES concept (a) and model (b).**

model. We call this model LinSAES.

2.5. Connecting via Swapping

We will start by presenting method overview, followed by details. Approach (in training mode) is illustrated in Figure 5.

Method overview: all model parameters (encoders (α, β) and decoders ($\mathcal{D}$)) are trained end-to-end by enforcing the following. 1) **Transformation Encoding.** The initial and final states are transformed into an initial state and a corresponding state-transformation code through an encoder. This transformation code captures the changes or effects between the initial and final states. 2) **Reconstruction using Decoder.** Given the initial state and the state-transformation

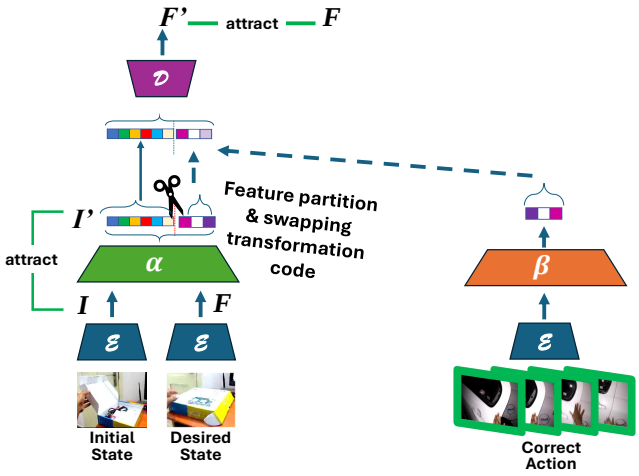


Figure 5. **Connecting via Swapping.** Here, we have not shown using of incorrect action for better explanation and avoiding confusion. However, in practice, we leverage incorrect actions.

code, a decoder reconstructs the final state. This decoder essentially learns to generate the final state based on the initial state and the transformation code. 3) **Obtaining Transformation Code from Cross-sample Actions.** Additionally, a state-transformation code can be obtained from the action (but from another sample) occurring between the initial and final states. This is done through another encoder that distills the transformation code from the action. Note that, during training the final state is reconstructed using both transformation codes—obtained from states and action. By obtaining equivalent state-transformation codes both from the states directly and from the actions, the model learns to connect actions to their effects. The decoder then utilizes this transformation code to reconstruct the final state from the initial state, effectively capturing the impact of actions on the state transition in the video data.

Method details: To explicitly isolate a n-dimensional "transformation "code", we pass the initial and final states (each represented by m-dimensional features) through an encoder α, which outputs a $(m+n)$-dimensional vector. We enforce the condition that the first m elements of this vector represent the initial state and the remaining n elements represent the transformation "code". Now, using another encoder (β), we distill a n-dimensional transformation code from an action option. We swap the transformation code obtained using states with that obtained from the action option. A decoder ($\mathcal{D}$) then reconstructs the final state from the extracted initial state and swaps the transformation code. All encoders and decoders are trained by enforcing that the extracted initial state be similar to the ground truth initial state and the reconstructed final state be similar to the ground truth final state in case the transformation code was coming from the correct action option (minimize the similarity if the transformation code was from incorrect action option). This approach is inspired by [5]. However, while their approach disentangles human pose and appearance, our approach extends it to disentangle state-transformation for linking actions and their effects.

2.6. CLIP

For this baseline, we adopt the previously discussed naive approach subsection 2.1. In this case, we use the CLIP [6] representations for states and actions. Further, to obtain a clip-level representation for action-clips, we average the frame-level representations. Inference process remains as the Naive approach.

2.7. VideoChat2

VideoChat2 [3] is powerful video understanding foundational model. It is trained on 20 challenging video understanding simultaneously. We evaluated various ways to adopt VideoChat2 for our problem with the objective of maximizing the performance on Action Selection. We obtained best results with the following approach. First, we provide the initial and the final states to the model alongwith the four action classes in language format, ask to identify the action taking place. Second, we ask the model to identify the action class (from the four choices in language format—these are nothing but the class names of the four action options) taking place in the correct action-clip. Then, we match the classes obtained from states and action-clips. If there is a match and the predicted class is correct, we consider that the model got the answer, else, we consider the answer of the model as wrong.

2.8. VideoMAE

Mask autoencoding (MAE) might seem similar to Action Selection, but there is a fundamental difference between them. MAE randomly masks patches without any explicit attention to initial and final states, whereas Action Selection particularly involves connecting actions and desired state changes. Nonetheless, VideoMAE has achieved strong results in video understanding. Therefore, for completeness, we consider VideoMAE as one of the baselines. By design, VideoMAE cannot be directly used for our task, so we adopt the previously discussed Naive approach, but using VideoMAE as the backbone for Action Selection task.

3. Introduction to Action Quality Assessment (AQA) and how it is connected to Effect-Affinity Assessment

Definition of AQA. AQA is a fine-grained action analysis task, where the models try to assess how well an action was

performed. Judging during Olympics diving can be a classic example of AQA. AQA requires paying attention to very fine-grained details of action. For example, it involves analyzing how tight was the athlete's form during somersaults, how close the athlete's feet were during twisting motion in air, was the athlete under-/over-rotated during entry into the water, how high the athlete jumped during take-off, *etc*. Taking these into consideration, a score is given out to indicate how good the athletes' performance was or whether the rules of the competition were followed.

Effect-Affinity Assessment and AQA. In our formulation, Effect-Affinity Assessment involves discerning between very nearby effects, and determining how far is an effect-frame from the action applied on an initial state. To solve this task, the model learns to pay attention to fine-grained details like what maneuvers the athlete applied/executed, and how that would result in what series of poses, or how the athlete's position would evolve as a result of their maneuvers. These details have an overlap with the elements of interest in AQA. Therefore, we hypothesize that representations learnt in solving our SSL CATE Effect-Affinity Assessment should transfer well to AQA task.

4. Action-Effect Joint Attention Visualization

We use the best performing Analogical-Reasoning model for *attention visualization*, employing a modified GradCAM [7] to generate joint attention heatmaps over states & actions. Specifically, we backpropagate the dot-product of state-change vector & the action vector through the initial, final, action branches into visual input space.

5. Further Action Selection performance analysis

1) We believe the Analogical model performed potentially because explicitly computing the state change and comparing with the action options might be more beneficial.

2) We have also provided the classwise accuracies from two of the best performing models: 1) Analogical Reasoning model; 2) LinSAES (learning linear state-action embedding space) in the Figure 6. In both cases, we observed a similar trend that the action classes on which the models were most accurately connected actions and effects were simple and single-step movements. Action classes where the models struggled the most were composite and complex involving multiple steps or sub-actions. These composite actions may require the model to understand not only individual actions but also their sequential dependencies and temporal relationships. The increased complexity can make it more challenging for the model to accurately predict the correct sequence of actions to achieve the desired outcome. The easiest classes to connect were derived from SSv2 dataset [2] representing common tasks or actions one might encounter in daily life or simple mechanical tasks. These actions focus on the manipulation and movement of objects in various ways, often on flat surfaces. Most difficult to connect actions were from COIN dataset [8], often associated with particular activities such as cooking, construction, or assembly. These actions involve more specific and varied tasks such as cutting, filling, inserting, and cooking, indicating a wider scope of activities.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1

[2] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017. 4

[3] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206, 2024. 3

[4] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015. 2

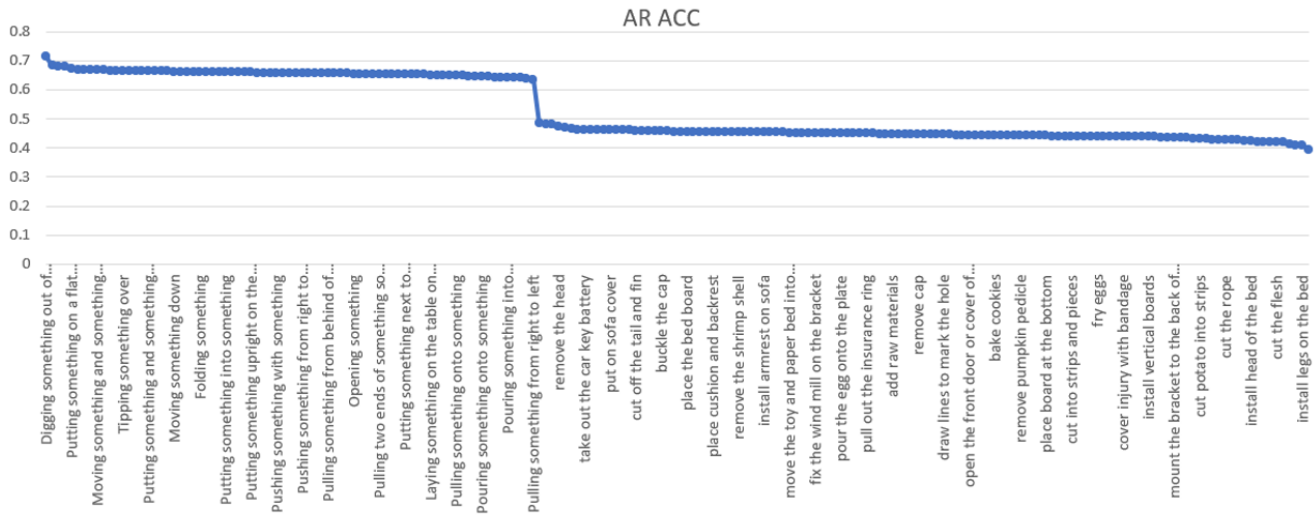
[5] Paritosh Parmar, Amol Gharat, and Helge Rhodin. Domain knowledge-informed self-supervised representations for workout form assessment. In *European Conference on Computer Vision*, pages 105–123. Springer, 2022. 3

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 3

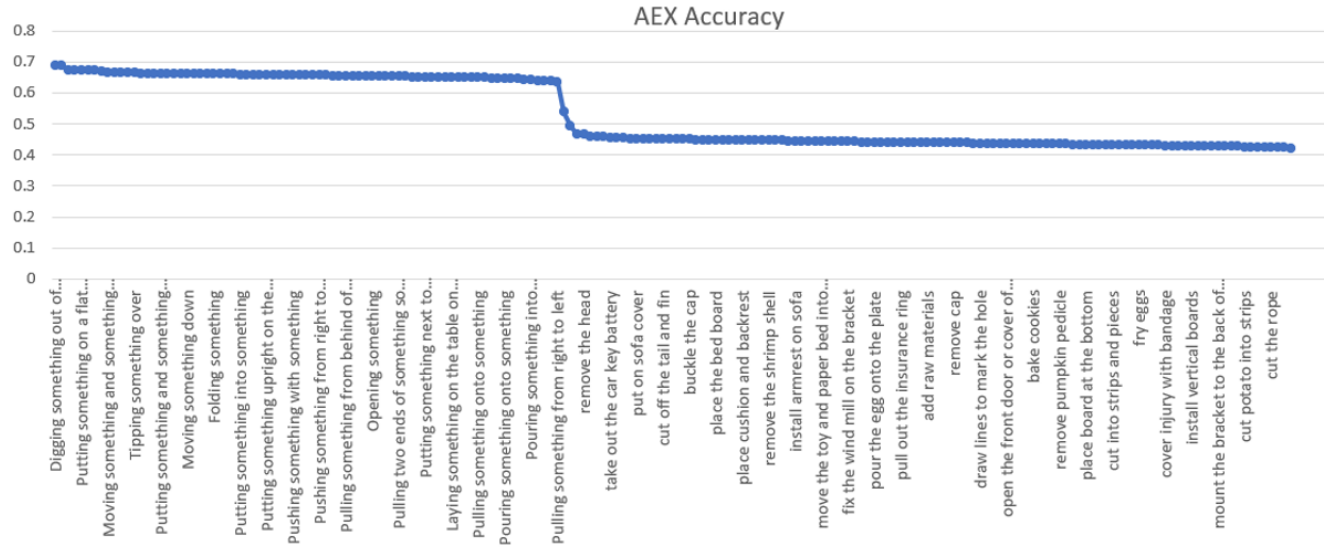
[7] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: visual explanations from deep networks via gradient-based localization. *International journal of computer vision*, 128:336–359, 2020. 4

[8] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019. 4

[9] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. Actions˜ transformations. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2658–2667, 2016. 1



Top 10		Bottom 10	
Digging something out of something	0.715749	install horizontal boards	0.425296
Pulling something onto something	0.683761	cut the wide side	0.421955
Lifting something up completely without letting it drop down	0.680169	add milk	0.421891
Pushing something onto something	0.679648	remove old lamp fixed cover	0.42132
Putting number of something onto something	0.670618	cut the rope	0.421243
Pouring something out of something	0.670539	close cover	0.419395
Moving something across a surface until it falls down	0.670443	draw an outline	0.412429
Bending something until it breaks	0.668829	cut the head	0.410232
Dropping something in front of something	0.667867	place the bed board	0.410019
Putting something similar to other things that are already on the table	0.667492	install fan bracket	0.394089



Top 10		Bottom 10	
Digging something out of something	0.6863	install head of the bed	0.4204
Lifting a surface with something on it until it starts sliding down	0.6863	install fans and lights	0.4202
Burying something in something	0.6738	drill with an electric drill	0.4198
Pulling two ends of something so that it gets stretched	0.6735	draw an outline	0.4174
Putting something on a flat surface without letting it roll	0.6725	cut the flesh	0.4151
Bending something until it breaks	0.6709	add milk	0.4129
Pushing something onto something	0.6709	place mattress on the bed	0.4109
Putting something something and something on the table	0.6696	cut the wide side	0.4031
Moving something and something closer to each other	0.6664	install legs on the bed	0.3968
Lifting up one end of something without letting it drop down	0.6645	install fan frame	0.3921

Figure 6. **Classwise accuracies from Analogical Reasoning (AR) LinSAES (AEX) models.** Easiest and most difficult classes are listed below.