Supplementary material for: Semantically Guided Representation Learning For Action Anticipation

Anxhelo Diko[1], Danilo Avola[⋆1], Bardh Prenkaj[⋆2], Federico Fontana[1], and Luigi Cinque[1]

[1] Sapienza University of Rome, Computer Science Department
`{diko,avola,fontana.f,cinque}@si.uniroma1.it`
[2] Technical University of Munich, Chair of Responsible Data Science
`bardh.prenkaj@tum.de`

We supplement the main paper (hereafter, MP) by adding details on S-GEAR's architecture (see Sec. A) and extend our original experimentations (see Sec. B).

A S-GEAR's Details

The main paper gives a general overview of S-GEAR architecture, which comprises a visual encoder, the temporal content aggregation module, the prototype attention module, and the causal transformer decoder. While the visual encoder [2, 9] – a standard ViT – and the temporal decoder [4, 9, 11]– causal transformer based on masked-attention – are well-known architectures in the literature, we specifically tailor the two intermediate modules to serve our purposes. We provide the details of such blocks in Sec. A.1 and A.2. In Sec. A.3, give details regarding the visual prototype initialization. Finally, in Sec. A.4, we give details regarding the training strategy used to train S-GEAR with pre-extracted features.

⋆ Equal second author contribution.

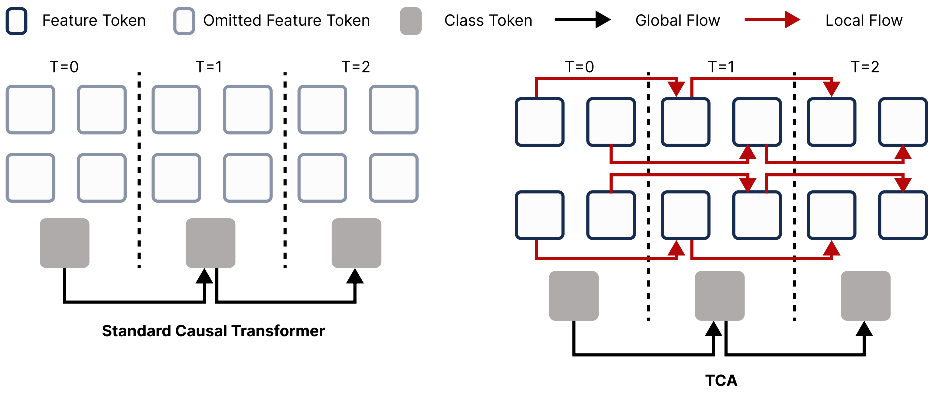


Fig. 1: Standard causal transformer flow vs. TCA flow.

A.1 Temporal Context Aggregation

Sequential models like LSTMs and causal transformers excel at handling temporal frame sequences. However, relying on class tokens, they prioritize global information [4, 11] and neglect spatial cues. To illustrate the difference between the standard causal transformer and our proposed *Temporal Context Aggregator* (TCA), we provide the reader with Fig. 1. Here, the left-hand side shows the workflow of a standard causal transformer applied on a sequence of ViT frame features composed of *local* feature tokens and a *global* class token. In this scenario, the causal transformer omits the local information and only propagates the global information in time to create causal representations [4,11]. Because local tokens encode specific scene details within different regions, not propagating their information hinders the model from understanding scene dynamics (e.g., how an object's location changes as a particular action progresses). Therefore, we design the TCA φ block. TCA extends the information flow by propagating global and local tokens across time (see Fig. 1 right), building causal representations considering scene dynamics at a finer spatial scale.

TCA builds on the attention mechanism and processes intermediate features I_t. Thus, I_t undergoes linear processing to generate the query (Q_t), key (K_t), and value (V_t) vector representations. Afterward, as shown in Fig. 2 (left), the TCA uniquely aggregates keys and values from past frames to subsequent ones before computing the attention matrix. This approach enables the queries of each frame to access a rich set of keys and values infused with comprehensive spatiotemporal information about past contexts, enabling better temporal dependency [10]. Specifically, the K_t and V_t vectors are augmented as in Eqns. 1 and 2, respectively:

$$\hat{K}_t = \begin{cases} K_t & \text{if } t = 0 \\ \delta(K_t, \alpha_{t-1} \cdot \hat{K}_{t-1}) & \text{otherwise} \end{cases}, \tag{1}$$

$$\hat{V}_t = \begin{cases} V_t & \text{if } t = 0 \\ \delta(V_t, \alpha_{t-1} \cdot \hat{V}_{t-1}) & \text{otherwise} \end{cases}, \tag{2}$$

where $\hat{K}_t$ and $\hat{V}_t$ are the augmented keys and queries of frame f_t, α_{t-1} is a learnable weight parameter that balances the quantity of the information transmitted from past observations, and δ is a permutation invariant aggregation function. Note that we tried different functions for δ (e.g., cumulative-max), but through empirical analyses, we chose summation.

Once these vectors are obtained, the computation proceeds with the standard self-attention operations. Precisely, Q_t and $\hat{K}_t$ compute the attention scores through scaled matrix multiplication, then normalized into $[0, 1]$ weights through a softmax function. Afterward, the weights aggregate information between the augmented feature tokens of $\hat{V}_t$ and create causal representations $\overline{I}_t$. Formally, the procedure can be defined as follows:

$$\overline{I}_t = \text{softmax}\left(\frac{Q_t \hat{K}_t^\top}{\sqrt{d}}\right) \hat{V}_t. \tag{3}$$

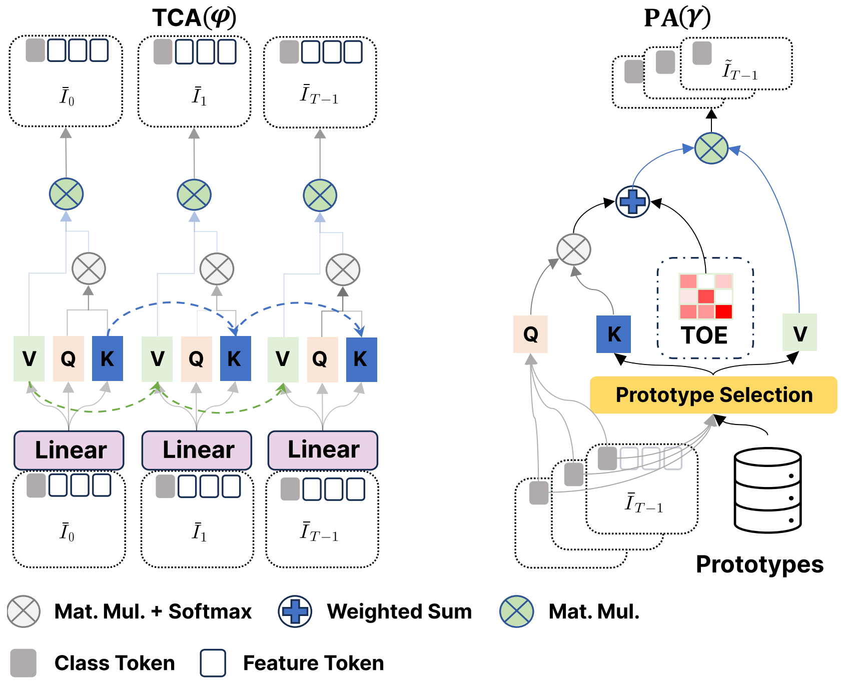


Fig. 2: The TCA block (left) generates detailed and sequential representations by aggregating past observation values V and keys K into current ones. It follows a standard attention mechanism to capture temporal dependencies. The PA block (right) performs cross-modality attention to aggregate selected prototypes from visual queries and uses a TOE weight matrix to encode temporal awareness. Note that the red gradient in TOE represents the magnitude of learnable weights.

Now, the class token representations $\overline{I}_t^0$ of $\overline{I}_t$ encodes the global representation of frame t enhanced by detailed contextual information of the past frames.

A.2 Prototype Attention

Recall that S-GEAR encodes semantic relationships between actions. To help the encoding of such relationships, we integrate a *Prototype Attention* (PA) block γ detailed in Fig. 2 (right). The PA module helps the network learn meaningful representations by incorporating semantic information from the visual prototypes. PA has two stages: **(1)** selecting the prototypes and **(2)** modeling the relationship between features and prototypes.

Similar to TCA, we build PA upon the attention mechanism, giving in input both the class tokens from the intermediate encodings generated by ViT – i.e., $I^0 = \{I_0^0, I_1^0, \ldots, I_{T-1}^0\}$ and the visual prototypes ρ_v. We rely on the relative similarities between actions to address **(1)**. Specifically, we begin by calculating the cosine similarity between each I_t^0 and the visual prototypes, obtaining the relative representation vector $\mathbf{r}^{I_t^0}$ of frame t as in Eq. 4:

$$\mathbf{r}^{I_t^0} = cos(I_t^0, \rho_v). \tag{4}$$

Then, we select the top k most similar prototypes for each feature vector for the remaining calculations. However, for simplicity, let us assume we select the most similar prototype for each feature vector. After acquiring the estimated prototypes, PA addresses **(2)** by modeling their relationship with the feature encodings using the attention mechanism. In this case, the set of prototypes represents both the key (K) and value (V) vectors. Conversely, the query (Q) vector is derived from I^0. The first step of the relationship modeling is the computation of the attention scores W_a through a scaled matrix multiplication between Q and K as in Eq. 5:

$$W_a = \frac{QK^\top}{d}. \tag{5}$$

Following the standard attention procedure, the next step in the attention process should be normalizing and applying W_a to V and having the output features. However, the selected prototypes do not have temporal continuity like the sequential frames and contradict the temporal causality built from TCA when the fusion occurs (see Sec. 3.1 in MP). Inspired by [5], we introduce a *Temporal Order Encoding* (TOE) weight vector shaped as a Toeplitz matrix to model the temporal order between elements of V. We provide the reader with an example to illustrate Toeplitz matrices and their unique structure. Here, we show a 5-element TOE as a 3×3 Toeplitz matrix Δ as in Eq. 6:

$$\Delta = \begin{pmatrix} w_0 & w_1 & w_2 \\ w_3 & w_0 & w_1 \\ w_4 & w_3 & w_0 \end{pmatrix}, \tag{6}$$

where w_i represents the i^{th} weight from the TOE for $i \in \{0, 1, ..., 4\}$. Notice that a single weight represents each diagonal. Additionally, with a 3×3 matrix, we can model the temporal relationships of a sequence of three elements. Hence, for a sequence of T elements like V, we need a $T \times T$ Toeplitz matrix built from a $(2T-1)$-element TOE. Generalizing, the $T \times T$ Δ functionality allows PA to model the relative temporal position or order between elements of V when aggregating features. To apply Δ to V, we first sum Δ with the normalized W_a and then perform a matrix multiplication between the resulting matrix and V as in Eq. 7:

$$\tilde{I} = (\beta(\text{softmax}(W_a)) + (1-\beta)\Delta)V, \tag{7}$$

where β is a learnable scaling factor that balances the sum between W_a and Δ. $\tilde{I} \in \mathbb{R}^{T \times d}$ now represents the selected visual prototypes fused with the current context and with encoded relative temporal awareness.

A.3 Prototype Initialization

We initialize our visual prototypes ρ_v using action samples generated by the proposed architecture (detailed in Sec. 3.1 in MP). First, we train the network (PA

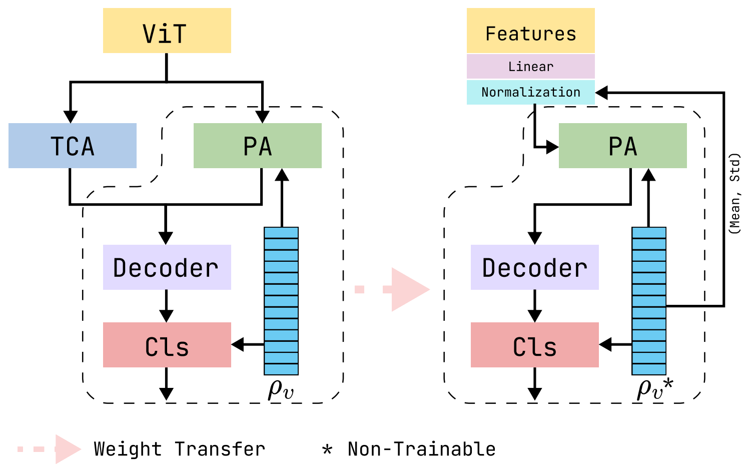


Fig. 3: Training S-GEAR with pre-extracted features.

omitted) for action recognition on EK100 for the egocentric tasks and 50S for the exocentric tasks. Then, without fine-tuning, we extract multiple representations for each action class present in a dataset. The average of these representations becomes the "typical action sample" used to initialize the corresponding class prototype. We also experimented with other initialization methods like random values and using the language prototypes ρ_ℓ as initial values. However, such methods did not provide significant improvements. Thus, the results we report in MP rely on the first initialization approach.

A.4 Training S-GEAR With Pre-extracted Features

S-GEAR's modular design allows it to work with different visual backbones, although it's primarily designed for end-to-end training with a ViT architecture [2]. To ensure a fair comparison with the SOTA [3, 4, 11], we also train our network using pre-extracted features from TSN, Faster R-CNN (FRCNN) and irCSN backbones provided by Furnari et al. [3] and Girdhar et al. [4], respectively.

Our training process (see Fig. 3) is built upon aligning the pre-extracted feature distribution with ρ_v learned from S-GEAR in its end-to-end training with ViT. These prototypes already capture the desired structure of the latent space. By aligning with them, we simplify training when we lack the variability introduced by typical video preprocessing techniques (i.e., random cropping or flipping). To this end, given pre-extracted visual features χ_t $\forall t \in [0, T-1]$, we first apply a linear transformation and then normalize them using the mean μ_{ρ_v} and standard deviation σ_{ρ_v} from the learned prototypes as in Eq. 8:

$$I_t = \frac{\text{lin}(\chi_t) - \mu_{\rho_v}}{\sigma_{\rho_v}}. \tag{8}$$

Table 1: Weights associated with individual loss terms to train S-GEAR for each task.

Dataset	$\mathcal{L}_{Sem}$	$\mathcal{L}_{Reg}$	$\mathcal{L}_{Cls}$	$\mathcal{L}_{Past}$	$\mathcal{L}_{Feat}$
EK100	4.0	1.0	1.0	1.0	1.0
EK55	2.0	1.0	1.0	1.0	1.0
EG	2.0	1.0	1.0	0.1	1.0
50S	1.0	0.1	1.0	0.1	1.0

Hence, leveraging the pre-trained weights of PA, causal decoder, and classification head from the end-to-end training, we fine-tune S-GEAR to adapt to the new visual features I_t. During this process, ρ_v remains unchanged.

B Experiments Extension

Here, we provide additional details regarding our experiments. In Sec. B.1, we provide details regarding the weights used for each loss function to train S-GEAR for each dataset. In Sec. B.2, we provide the results supporting the ablation study on the anticipation time τ_a from Sec. 4.6 in the MP. In Sec. B.3, we give details regarding the ensemble setup of our models used to obtain the multimodal results on EK55/100 from the MP. In Sec. B.4, we try different combinations of backbones and modalities on EK100's validation set. Finally, in Sec. B.5, we extend Sec. 4.7 from the MP and compare the semanticity learned from ρ_v and that encoded from ρ_ℓ.

B.1 Composed Loss Weights

Here, we provide the specific weights for each loss term introduced in Sec. 3.3 of the MP. Table 1 details these weights. Notice that we combine $\mathcal{L}_{Sem}$ and $\mathcal{L}_{Reg}$ due to their similar purpose in MP. However, they are weighted individually during optimization to account for their different magnitudes. Importantly, we use the same weights for both EK100 and EK55 datasets, regardless of whether training end-to-end with ViT features or using pre-extracted features. The sole exception is $\mathcal{L}_{Sem}$ because visual prototypes remain frozen in the pre-extracted feature training scenario (see Sec. A.4).

B.2 Ablation on τ_a

Here, we report the results of experiments in Sec. 4.6 of MP. Specifically, the results correspond to Fig. 5 of MP exploring Top-5 Acc. on EK55 and EG for τ_a spanning from 0.25s to 2.0s.

Table 2: Results on the EK55 validation set regarding Top-5 Accuracy at different Anticipation Time-Steps averaged across the three official splits. All results in this table are obtained using only RGB modality.

Model	Top-5 Accuracy % at different τ_a (s)							
	2.0	1.75	1.5	1.25	1.0	0.75	0.5	0.25
FN [1]	23.47	24.07	24.68	25.66	26.27	26.87	27.88	28.96
RL [7]	25.95	26.49	27.15	28.48	29.61	30.81	31.86	32.84
RU-LSTM [3]	25.44	26.89	28.32	29.42	30.83	32.00	33.31	34.47
SRL [8]	25.82	27.21	28.52	29.81	31.68	33.11	34.75	36.89
S-GEAR (ours)	**28.57**	**29.95**	**31.34**	**32.87**	**34.48**	**34.92**	**36.49**	**37.49**

Table 3: Results on the EGTEA Gaze+ validation set regarding Top-5 Accuracy at different Anticipation Time-Steps averaged across the three official splits.

Model	Top-5 Accuracy % at different τ_a (s)							
	2.0	1.75	1.5	1.25	1.0	0.75	0.5	0.25
FN [1]	54.06	54.94	56.75	58.34	60.12	62.03	63.96	66.45
RL [7]	55.18	56.31	58.22	60.35	62.56	64.65	67.35	70.42
RU-LSTM [3]	56.82	59.13	61.42	63.53	66.40	68.41	71.84	74.25
SRL [8]	59.69	61.79	64.93	66.45	70.67	73.49	78.02	82.61
HRO [6]	**60.12**	**62.32**	**65.53**	67.18	71.46	**74.05**	79.24	83.92
S-GEAR (ours)	58.85	61.52	64.99	**70.56**	**71.93**	73.71	**79.26**	**85.41**

B.3 Multimodal Ensemble

For EK100's validation set, we evaluate three S-GEAR versions with varying backbone combinations. Firstly, S-GEAR uses late fusion[3] of ViT-based RGB S-GEAR (weight: 2.5) and FRCNN object features (weight: 0.5). S-GEAR-2B adds ViT↓-based RGB S-GEAR (weight: 1.5) to the previous fusion. Finally, S-GEAR-4B combines all RGB S-GEAR variants (ViT, ViT↓, TSN, irCSN with weights 2.5:1.5:1:1) and FRCNN object features (weight: 0.5). The same weight combinations apply for the EK100 test set. On the EK55 validation set, we late fuse ViT, irCSN, and TSN-based RGB S-GEAR (all weighted 1.5), along with TSN flow features (weight: 1) and FRCNN object features (weight: 1). Note that except for the ViT backbones, all other RGB and modality features are pre-extracted as provided in [3, 4].

B.4 Fine-grained Multimodal Ablation

We present results from combining different backbones and modalities on the EK100 validation set (see Table 4). We use late fusion with the following backbone weights: ViT (weight: 2.5), ViT↓ (weight: 1.5), TSN (weight: 1.0), irCSN

[3] Weighted Combination of predictions from different models and modalities.

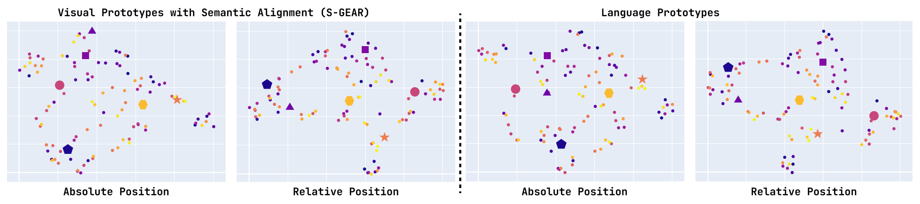
Table 4: Ablation of different RGB backbones and modalities on EK100 validation set. ViT↓ represents ViT with an input size of 224×224. All backbone and modalities are combined through late fusion.

RGB Backbones	Modalities	Verb	Noun	Action
ViT	RGB + Obj	29.5	37.8	18.9
ViT	RGB + Flow	30.2	35.4	18.4
ViT	RGB + Obj + Flow	29.6	37.4	18.9
ViT + ViT↓	RGB + Obj	30.5	**38.4**	<u>19.6</u>
ViT + ViT↓	RGB + Flow	**30.8**	36.2	19.5
ViT + ViT↓	RGB + Obj + Flow	30.2	36.9	19.6
ViT + TSN	RGB + Obj	30.4	38.1	19.5
ViT + TSN	RGB + Flow	<u>30.8</u>	36.6	19.2
ViT + TSN	RGB + Obj + Flow	29.7	37.8	19.4
ViT + irCSN	RGB + Obj	30.5	<u>38.2</u>	19.6
ViT + irCSN	RGB + Flow	29.5	37.0	19.1
ViT + irCSN	RGB + Obj + Flow	29.4	37.6	19.6
ViT + ViT↓ + irCSN + TSN	RGB + Obj	30.2	37.0	**19.9**
ViT + ViT↓ + irCSN + TSN	RGB + Flow	30.4	36.6	19.5
ViT + ViT↓ + irCSN + TSN	RGB + Obj + Flow	29.9	37.3	19.6

(weight: 1.0), and equal weights (weight: 0.5) for Object and Flow features. We report results for various combinations, ranging from models using a single backbone to those combining up to four. We can notice that combining RGB and object modalities provides the best overall results regarding Top-5 Recall, which motivates our choice to exclude flow from the final ensemble model.

B.5 Closer Look at Semanticity

In Sec. 4.7 of MP, we graphically show that the latent space topology defined from S-GEAR's learned prototypes is similar to the one defined from language prototypes but not perfectly aligned due to the influence of visual cues. Such resemblance in latent space geometry suggests that S-GEAR can semantically reason regarding action associations and consider scene information simultaneously. To dive deeper into this aspect, here we consider a group of randomly selected reference actions – i.e., *Pour Oil*, *Put Pan*, *Take Sponge*, *Compress Sandwich*, *Cut Tomato*, and *Move Around Bacon* – and analyze the prototypes found in their proximity. Graphically, we show these comparisons in Fig. 4. The grey boxes report the top-5 most similar actions for each reference action (top bold string). Green actions (text on the left in each grey box) represent the most similar actions in the language space. Black actions (right side of the grey boxes) highlight alignment between vision and language, whereas red actions state that there is a mismatch between the two. Each action contains the cosine similarity between it and the reference one (see values inside the brackets). Notice that



★ Pour Oil	
Put Oil Container (0.61)	Put Oil Container (0.61)
Open Oil Container (0.59)	Open Oil Container (0.59)
Pour Seasoning (0.54)	Pour Seasoning (0.56)
Close Oil Container (0.53)	Close Oil Container (0.53)
Take Oil Container (0.5)	Pour Condiment (0.52)

■ Put Pan	
Take Pan (0.71)	Move Around Pan (0.72)
Put Cooking Utensil (0.67)	Take Pan (0.69)
Wash Pan (0.66)	Put Cooking Utensil (0.67)
Put Pot (0.64)	Take Cooking Utensil (0.65)
Move Around Pan (0.6)	Put Pot (0.6)

● Take Sponge	
Put Sponge(0.8)	Squeeze Washing Liquid (0.74)
Wash Hand (0.42)	Put Sponge (0.68)
Wash Pan (0.39)	Wash Bowl (0.42)
Take Bowl (0.38)	Wash Pan (0.41)
Wash Bowl (0.38)	Was Hand (0.41)

⬢ Compress Sadwich	
Spread Condiment (0.43)	Spread Condiment (0.45)
Mix Eggs (0.38)	Take Bread (0.4)
Put Condiment (0.37)	Put Chees (0.39)
Put Bread (0.36)	Put Bread (0.37)
Open Bread (0.35)	Open Bread (0.37)

⬟ Cut Tomato	
Put Tomato (0.85)	Take Tomato (0.86)
Take Tomato (0.83)	Put Tomato (0.83)
Put Tom. in Cont. (0.68)	Cut Onion (0.69)
Take Tom. from Cont. (0.65)	Take Tom. from Cont. (0.66)
Cut Onion (0.54)	Put Tom. in Cont. (0.56)

▲ Move Around Bacon	
Move Around Pan (0.5)	Spread Condiment (0.61)
Move Around Patty (0.48)	Move Around Patty (0.49)
Move Around Pot (0.46)	Pull Apart Onion (0.47)
Move Arount Eat. Ut. (0.46)	Move Around Pan (0.47)
Spread Condiment (0.4)	Move Around Eat. Ut. (0.43)

Fig. 4: (best viewed in color) Fine-grained semanticity comparison.

actions are associated with the same activities in both spaces regarding action classes and the magnitude of their cosine similarity. Nevertheless, in 5/6 reported cases, we have at least one divergence between the actions and the reference one. This phenomenon re-emphasizes the divergence of visual prototypes influenced by visual cues and the co-occurrences of actions in videos. Interestingly, even divergent actions mostly have reasonable connections (i.e., they are likely to co-occur) with the reference action. This underscores S-GEAR's prototypes' ability to keep their semantic composure and account for the co-occurrence of actions influenced by the observed action segments[4]. Notice that in the case of *Take Sponge* as a reference action, the divergence between S-GEAR and language prototypes is the action *Squeeze Washing Liquid*, which is probably the most likely action to co-occur with *Take Sponge* in a kitchen scenario. Additionally, it shows that with the proposed method, a perfect alignment between two spaces cannot be obtained as long as the task is bounded to a given dataset. However, in such cases, as we showed in Sec. 4.6 of MP (5th setup from Table 3), relying completely on global semantics (i.e., language prototypes) is less beneficial for action anticipation than merging it with the visual cues (S-GEAR) due to the importance of motion and scene composition in suggesting possible future actions.

References

1. De Geest, R., Tuytelaars, T.: Modeling temporal structure with lstm for online action detection. In: Proc. of the IEEE/CVF Winter Conf. on Appl. of Comput. Vis. pp. 1549–1557 (2018). `https://doi.org/10.1109/WACV.2018.00173`
2. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.:

[4] Notice that the observed segments come from the EG dataset

An image is worth 16x16 words: Transformers for image recognition at scale. In: Int. Conf. on Learning Representations. pp. 1–21 (2021)

3. Furnari, A., Farinella, G.M.: Rolling-unrolling lstms for action anticipation from first-person video. IEEE Trans. on Pattern Anal. and Mach. Intell. **43**(11), 4021–4036 (2021). https://doi.org/10.1109/TPAMI.2020.2992889
4. Girdhar, R., Grauman, K.: Anticipative video transformer. In: Proc. of the IEEE/CVF Int. Conf. on Comput. Vis. pp. 13505–13515 (2021)
5. Huang, F., Lu, K., Yuxi, C., Qin, Z., Fang, Y., Tian, G., Li, G.: Encoding recurrence into transformers. In: The Eleventh Int. Conf. on Learning Representations. pp. 1–13 (2022)
6. Liu, T., Lam, K.M.: A hybrid egocentric activity anticipation framework via memory-augmented recurrent and one-shot representation forecasting. In: Proc. of the IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. pp. 13904–13913 (2022)
7. Ma, S., Sigal, L., Sclaroff, S.: Learning activity progression in lstms for activity detection and early detection. In: Proc. of the IEEE/CVF Conf. on Comput. Vis. and pattern Recognit. pp. 1942–1950 (2016)
8. Qi, Z., Wang, S., Su, C., Su, L., Huang, Q., Tian, Q.: Self-regulated learning for egocentric video activity anticipation. IEEE Trans. on Pattern Anal. and Mach. Intell. **45**(6), 6715–6730 (2023). https://doi.org/10.1109/TPAMI.2021.3059923
9. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Adv. Neural. Inf. Process. Syst. vol. 30, pp. 1–11 (2017)
10. Wu, C.Y., Li, Y., Mangalam, K., Fan, H., Xiong, B., Malik, J., Feichtenhofer, C.: Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In: Proc. of the IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. pp. 13587–13597 (2022)
11. Xu, X., Li, Y.L., Lu, C.: Learning to anticipate future with dynamic context removal. In: Proc. of the IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. pp. 12724–12734 (2022). https://doi.org/10.1109/CVPR52688.2022.01240