HaLP: Hallucinating Latent Positives for Skeleton-based Self-Supervised Learning of Actions
Supplemental Material

Anshul Shah[1] Aniket Roy[1*] Ketul Shah[1*] Shlok Mishra[2]
David Jacobs[2,3] Anoop Cherian[4] Rama Chellappa[1]
[1]Johns Hopkins University [2]University of Maryland, College Park [3]Meta [4]MERL
{ashah95, aroy28, kshah33, rchella4}@jhu.edu {shlokm, dwj}@umd.edu cherian@merl.com

In this Supplementary material, we provide additional empirical studies, analyses, and details. We summarily list below the key sections.

1. Results on semi-supervised learning (Sec. A)
2. Additional ablation studies and analyses (Sec. B)
3. Results with multi-modal ensembles (Sec. C)
4. HaLP for Graph Representation Learning (Sec. D)
5. HaLP with AimCLR (Sec. E)
6. Implementation details (Sec. F)

A. Semi-supervised learning

In this experiment, we attach an MLP to the pre-trained backbone and finetune the MLP with labels on x% of the training data. We follow the same setting as prior work and report results at 1%, 5%, 10%, and 20% data. In Table 1, we see that HaLP outperforms the single modality baseline and prior state-of-the-art with comparable training showing that our representations can be adapted to the target task with little training data. HaLP with multi-modal training leads to results competitive with state-of-the-art.

B. Ablation studies

B.1. Different filtering variants:

As we discuss in the main paper, after the generation step we apply our **Rank Filter** (Main paper, Eq. (2) and Sec. 3.4) to filter out good positives for use with the loss. In this section, we experiment with alternative filtering approaches which can be used once the positives are generated. These are as follows:

variant-1: ${z_q}^\top z_i^H \geq {z_q}^\top z_k$
Generated positives should be closer to the query features (z_q) than the key features (z_k).

variant-2: ${z_k}^\top z_i^H \geq P_{\text{sel}}^\top z_k$
Generated positive should be closer to the key than the selected prototype.

In Table 2, we empirically compare the performance of these choices. We see that while all of our filters lead to improvements over the baseline, our proposed rank filter has the best performance. For these experiments, we generate 100 positives per anchor. We also report the number of retained positives after the filtering step. We see that since the rank filter has a stricter rule than variant-2, more positives get filtered out. The variant-1 is quite strict and retains 16 positives on average after the filtering step. These might be easier positives which might explain smaller improvements over the baseline.

B.2. Effect of changing μ:

In Table 3, we vary the value of μ in the loss function (Main paper, Eq. (8)) and observe its effect on the linear evaluation performance on the NTU60-xsub split. It is to be noted that as mentioned in the main paper, we use $\mu = 0$ for the first 200 epochs. We empirically notice that this led to better performance (Appendix B.4). First, we note that all of our approaches outperform the baseline (which has $\mu = 0$ throughout). We observe that the performance steadily improves as μ is increased. For ease of experimentation, we set $\mu = 1$ for all of our experiments. [1]

B.3. How many positives to generate:

This work proposes an efficient solution to hallucinate positives in the latent space. We can easily scale up the number of positives generated using our approach. In Table 4, we tabulate the performance by varying the number of positives. We see that using 100 positives has the best

*Aniket Roy and Ketul Shah contributed equally

[1]Note that one could also set μ/τ as a constant to be tuned but we refrain from doing so for making comparisons to the original MoCo loss easier.

Table 1. Results on NTU-60 in semi-supervised learning setup. Following CMD [6] we select a fraction of data and use that labelled data to fine-tune the network. We achieve consistent performance improvement over the other baselines for single-modality training. Using HaLP with CMD, gives improvements on NTU-60 x-sub and competetive performance on NTU-60 x-view. Results for AimCLR [3] were obtained using official models.

Method	NTU-60							
	x-view				x-sub			
	(1%)	(5%)	(10%)	(20%)	(1%)	(5%)	(10%)	(20%)
Additional training modalities or encoders								
ISC [8]	38.1	65.7	72.5	78.2	35.7	59.6	65.9	70.8
CrosSCLR-B [4]	49.8	70.6	77.0	81.9	48.6	67.7	72.4	76.1
CMD [4]	**53.0**	**75.3**	80.2	84.3	50.6	71.0	75.4	78.7
HaLP+CMD	**53.0**	**75.3**	**80.4**	**84.6**	**52.6**	**71.4**	**76.0**	**79.2**
Training using only joint								
LongT GAN [10]	-	-	-	-	35.2	-	62.0	-
MS^{2}L [5]	-	-	-	-	33.1	-	65.1	-
ASSL [7]	-	63.6	69.8	74.7	-	57.3	64.3	68.0
AimCLR [3]†	47.2	-	74.6	-	45.7	-	71.4	-
HaLP-Baseline	45.4	69.0	75.2	81.0	42.6	64.3	70.0	74.7
HaLP	**48.7**	**71.5**	**77.1**	**82.4**	**46.6**	**66.9**	**72.6**	**76.1**

Table 2. Different filtering approaches. We empirically evaluate the efficacy of various filtering approaches (defined in Appendix B.1). We see that while each of the approach leads to an improvement over the baseline, our rank filter outperforms the other proposed alternatives.

Method	G_{filtered} (/100)	NTU-60 x-sub
Baseline	-	78.0
Rank filter	91	79.7
variant-1	16	79.5
variant-2	100	79.6

Table 3. Effect of μ on linear evaluation performance. Recall that μ weighs the contribution of the HaLP loss compared to the standard contrastive loss. We see that increasing μ improves performance which saturates at higher values.

μ	NTU-60 x-sub
0 (baseline)	78.0
0.5	78.8
1.0	79.7
1.5	79.9
2.0	80.0

performance. As discussed in Sec. 3, our approximation could generate positives that might not satisfy the constraint in Eq. (2). Thus we pass all the generated positives through the `PosFilter()` before applying the loss. When using

Table 4. Effect of # positives on linear evaluation performance. We empirically see that generating 100 positives shows the best performance.

# Positives	NTU-60 x-sub
0 (baseline)	77.96
1	79.59
5	79.63
10	79.50
50	79.56
100	**79.71**
150	79.43

100 positives, we observe that $\sim 91\%$ positives are retained after the filtering step. We notice that other values of # positives also outperform the baseline, which shows the efficacy of our approach.

B.4. Changing when the HaLP loss is used:

For all experiments in the main paper, we set $\mu = 0$ for the first 200 epochs and use a constant $\mu = 1$ for the rest. In Table 5, we vary at which epoch μ is set to 1. We see that setting $\mu = 1$ at 200 epochs leads to the best results. Note that the generation of positives is a function of the anchor and the prototypes. We hypothesize that using generated positives too early might be sub-optimal due to noisy early representations while using it too late might not give the

Table 5. In this expeirment, we vary when to start generating new positives. We notice that setting $\mu = 1$ after 200 epochs gives the maximum improvement in performance. Note that all variants outperform the baseline.

$\mu = 1$ at X epochs	NTU-60 x-sub
50	79.6
100	79.6
200	**79.7**
300	78.7
400	78.3
$\mu = 0$ (baseline)	78.0

Table 6. Effect of varying the number of Queue elements used to extract prototypes

# Prototypes (N) $\rightarrow$	256	512	1024	2048	3072
NTU-60 x-sub	**79.7**	79.5	79.4	79.5	79.7

model enough training signal to significantly improve the representations.

B.5. Effect of topK

Note that following the prior works [6, 8], we use a memory queue of size 16384. Using the entire queue to obtain prototypes has two issues. First, this can increase the time to cluster. Secondly, note that the queue is obtained by appending the current keys in a first-in-first-out fashion. The use of very old (stale) keys can lead to sub-optimal representations since the model might have changed significantly. Thus we proposed to use only the most recent elements of the queue to obtain prototypes. In Table 6, we observe that using 256 most recent queue elements has the best performance.

B.6. Analyses of the generated positives

Evolution of similarities: In Fig. 1, we plot the similarity of z_k and z_i^H to the query z_q for the last 200 epochs of training. In Fig. 2, we plot the corresponding similarities to the queue (negatives). We see that the curves for real and hallucinated positives start with a larger gap but closely follow each other for the later part of training. We also notice that the mean similarities of positives are slightly higher for the real positives compared to the hallucinated ones. This verifies that our approach generates hard positives which lead to improved training. As expected, the anchor and positives are pulled closer together while the anchor and negatives are pushed farther away as training progresses.

Similarity of positives to prototypes: In Fig. 3, we plot the similarities of various positives to the prototypes. First, $\text{sim}(z_k, P)$ (blue curve) represents the similarity of the original key to each of the prototypes. The key idea in our work is to generate positives that share the same closest prototype

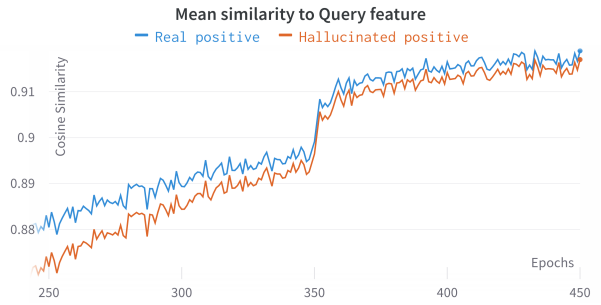

Figure 1. Mean similarity to query (z_q)

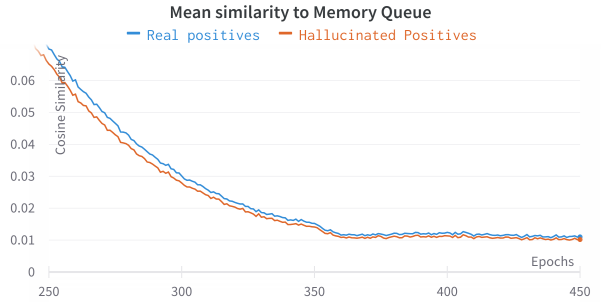

Figure 2. Mean similarity to Queue (negatives)

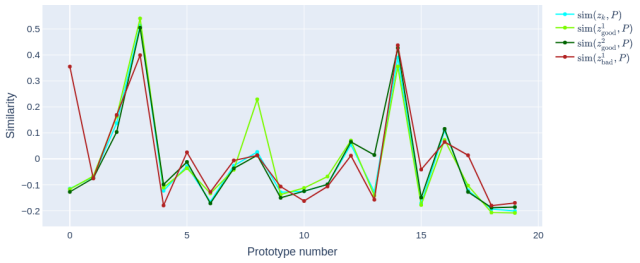

Figure 3. Similarity to prototype. We plot the similarity of the z_k and three generated positives to the prototypes. Two of the positives (green curves) share the same closest prototype as z_k (blue curve). These are the good positives which are used to compute the loss while the third positive (red curve) has a different closest prototype and is filtered out by the rank filter.

as the key z_k. We observe that the generated positives, ie positive 1 and positive 2 (green curves) satisfy the constraint above and add diversity to the generated positives while staying close to the original. We call these *good positives*. Different from these, a *bad positive* (red curve) significantly alters the similarities to each prototype and has a different closest prototype. This generated positive will get filtered out using our rank filter (Main paper, Sec. 3.4).

B.7. HaLP w/o memory bank

Most recent approaches (CMD, AimCLR, CrossSCLR, ISC) using contrastive learning use a memory queue due to

Table 7. Comparison with ensemble training. In this experiment, linear evaluation uses a late ensemble of various modalties. We see that HaLP+CMD outperforms the multi-modal CMD [6] approach.

Method	NTU-60		NTU-120		PKU-II
	x-sub	x-view	x-sub	x-set	x-sub
3s-CrosSCLR [4]	77.8	83.4	67.9	66.7	21.2
3s-AimCLR [3]	78.9	83.8	68.2	68.8	39.5
3s-CrosSCLR-B [4]	82.1	89.2	71.6	73.4	51.0
3s-CMD [6]	84.1	**90.9**	74.7	76.1	52.6
3s-HaLP	**85.1**	**91.0**	**75.5**	**76.8**	**53.4**

its effectiveness. We can use HaLP without a memory queue by clustering the key features from the current batch or use the key features directly as cluster centers. We implement a Baseline w/o momentum using the former and evaluate the effectiveness of HaLP loss over it. We see that HaLP is beneficial (kNN NTU60-xsub: 58.9 $\rightarrow$ 64.2). Note that our approach also helps GraphCL (Appendix D) which does not use momentum queue either.

C. Multi-modal ensemble

Following prior work, we perform test-time ensemble of the 3 modalities : joint, motion, and bone. In Table 7, we see that our approach leads to improvements over the baseline.

D. HaLP for Graph Representation Learning

As a future work, we want to apply our work to behavior recognition in infants for early diagnosis of Autism Spectrum Disorder. We are particularly interested in skeleton-based action recognition since skeletons are grounded on people in videos, are less sensitive to scene and object biases and have minimal privacy concerns. These advantages of skeleton data make them beneficial for our eventual target task. Self-supervised learning on skeletons has received less attention due to difficulties in crafting geometrically consistent data augmentations. In Table 8, we apply our approach over GraphCL and see improvements over the baseline showing the general nature of our approach.

Table 8. HaLP applied to Graph Representation Learning. We see that our approach of hallucinating positives also helps learn better encoders for general graphs. We use our plug-and-play module over GraphCL [9]

	NCI-1	PROTEINS	DD	MUTAG
GraphCL	77.87 ± 0.41	74.39 ± 0.45	78.62 ± 0.40	86.80 ± 1.3
+HaLP	$\mathbf{78.88 \pm 0.41}$	$\mathbf{74.65 \pm 0.70}$	$\mathbf{79.20 \pm 0.60}$	$\mathbf{89.35 \pm 1.2}$

E. HaLP with AimCLR [3]

To show the plug-and-play nature of our approach, we add our module to AimCLR [3], a recent Skeleton-SSL pipeline. We observe that adding HaLP is beneficial. The performance improves from 74.34 $\rightarrow$ **75.24** on NTU-60 x-sub showing the efficacy of our approach.

F. Implementation details

We use the same pre-training protocol as ISC [8] and CMD [6]. We use `Geomstats` [1] for k-Means clustering on the hypersphere with tolerance of 1E-3 and initial step size of 1.0. We use $\lambda = 0.8$ for all experiments. Wandb [2] was used for experiment tracking. Following are the key implementation details:

- Epochs : 450 and 1000 epochs for NTU and PKU respectively.
- Learning rate : 0.01, drop to 0.001 at 350 and 800 epochs for NTU and PKU respectively.
- Memory queue size : 16384
- Batch size : 64
- Number of positives generated : 100
- Number of prototypes : 20 for NTU-60 and PKU and 40 for NTU-120.
- We use $\mu = 0$ from 0-200 epochs. For NTU-60 and PKU, μ is set to 1 at 200 epochs, while it is set to 2 for NTU-120.
- Most recent elements used for clustering : 256
- Prototypes are updated every 5 steps.

References

[1] Geomstats https://geomstats.github.io/. 4

[2] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 4

[3] Tianyu Guo, Hong Liu, Zhan Chen, Mengyuan Liu, Tao Wang, and Runwei Ding. Contrastive learning from extremely augmented skeleton sequences for self-supervised action recognition. In *AAAI*, 2022. 2, 4

[4] Linguo Li, Minsi Wang, Bingbing Ni, Hang Wang, Jiancheng Yang, and Wenjun Zhang. 3d human action representation learning via cross-view consistency pursuit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4741–4750, 2021. 2, 4

[5] Lilang Lin, Sijie Song, Wenhan Yang, and Jiaying Liu. Ms2l: Multi-task self-supervised learning for skeleton based action recognition. *Proceedings of the 28th ACM International Conference on Multimedia*, 2020. 2

[6] Yunyao Mao, Wengang Zhou, Zhenbo Lu, Jiajun Deng, and Houqiang Li. Cmd: Self-supervised 3d action representation learning with cross-modal mutual distillation. *arXiv preprint arXiv:2208.12448*, 2022. 2, 3, 4

[7] Chenyang Si, Xuecheng Nie, Wei Wang, Liang Wang, Tieniu Tan, and Jiashi Feng. Adversarial self-supervised learning for semi-supervised 3d action recognition. *ArXiv*, abs/2007.05934, 2020. 2

[8] Fida Mohammad Thoker, Hazel Doughty, and Cees GM Snoek. Skeleton-contrastive 3d action representation learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1655–1663, 2021. 2, 3, 4

[9] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020. 4

[10] Nenggan Zheng, Jun Wen, Risheng Liu, Liangqu Long, Jianhua Dai, and Zhefeng Gong. Unsupervised representation learning with long-term dynamics for skeleton based action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2