Continual Learning Improves Zero-Shot Action Recognition

Supplementary Material

Shreyank N Gowda[1], Davide Moltisanti[2], and Laura Sevilla-Lara[3]

[1] University of Nottingham, UK
[2] University of Bath, UK
[3] University of Edinburgh, UK
Shreyank.Narayanagowda@nottingham.ac.uk, dm2460@bath.ac.uk,
l.sevilla@ed.ac.uk

1 Implementation Details

The framework for X-CLIP and X-Florence [7] consists of three main components: a cross-frame communication transformer, a multi-frame integration transformer, and a text encoder. We offer three CLIP variants:

- X-CLIP-B/32, with 12 layers (L_c), 12 attention heads (N_h), embedding dimension of size 768 (d), and patch size 32(p);
- X-CLIP-B/16, with 12 layers (L_c), 12 attention heads (N_h), embedding dimension of size 768, and patch size 16(p);
- X-CLIP-L/14, with 24 layers (L_c), 12 attention heads (N_h), embedding dimension of size 1024 (d), and patch size 14 (p).

All models use a 1-layer multi-frame integration transformer. For X-CLIP the text encoder is the same as in CLIP [8], while for X-Florence it is the same as in Florence [11]. For X-Florence, the cross-frame communication transformer is replaced with CoSwin-H visual encoder, and a 4-layer multi-frame integration transformer is added. Both X-CLIP and X-Florence utilize a video-specific prompting mechanism set to 2 blocks. We add 2 fully connected layers at the end of the video encoder to convert this to a classification model. The first layer is of size 4096 and the second is equivalent to the dimensions of the semantic vector (768). All hyperparameters are set as in X-Florence [7] to ensure fair comparison.

The CVAE has an encoder that consists of three fully connected layers. It takes in input a vector of size equivalent to the output of the text encoder. We use a latent dimension of size 512 and use a three-layer decoder for reconstruction.

Our feature generation network follows earlier work [6, 10]. The generator consists of three fully connected layers, where the output layer has dimension matching the video features dimension. The decoder has also three fully connected layers, but its output size corresponds to the class-embedding size. The discriminator has two fully connected layers and outputs a single value. All hidden layers in all networks have size 4096.

Table 1: Generalized Zero-Shot results, where 'u', 's' and 'H' correspond to average unseen accuracy, average seen accuracy and the harmonic mean of the two. All the reported results are on the same splits.

Model	HMDB51			UCF-101		
	u	s	H	u	s	H
WGAN [10]	23.1	55.1	32.5	20.6	73.9	32.2
OD [6]	25.9	55.8	35.4	25.3	74.1	37.7
OD + SPOT [1]	26.7	54.1	35.7	28.3	74.1	40.9
CLASTER [3]	43.7	53.3	48.0	40.8	69.3	51.3
SDR [2]	50.1	57.5	53.5	47.3	81.2	59.7
GIL (Ours)	**52.8**	**57.8**	**55.1**	**68.2**	**89.8**	**77.5**

2 Detailed Generalized Zero-Shot Action Recognition Results

In order to better analyze performance of the model on GZSL, we report the average seen and unseen accuracies along with their harmonic mean. The results on the UCF101 [9] and HMDB51 [5] datasets are reported in Table 1. Results are averaged from 10 runs, where for each run we create a different random train/test split (which is the same for all models). We note that in this more challenging setting GIL performs significantly better than previous state-of-the-art (especially on UCF-101), highlighting that the model trained with GIL is able to better retain knowledge and generalize to unseen classes. We do not report on Truze [4] as we have no overlap between the train and test classes.

3 How Much Does Sampling Percentage of Data Affect Model Performance?

We choose to sample 10% of data from the seen classes at each iteration. Here we consider a few other settings sampling different amounts of data, namely: 1%, 5%, 10%, 20%, 50% and 100% of data at once. Table 2 shows the results. We do not see a notable change in performance when increasing from 1% to 10%, however beyond 10% the performance starts dropping. The higher the percentage of data we sample, the faster training is. These results suggest the generalization ability of the model are affected when it is trained "too fast", and that a gradual and slow introduction of new data is accordingly beneficial.

4 Using the Text Encoder of X-Florence Directly

Instead of using any semantic embedding, we could potentially leverage the text encoder from X-Florence directly. We try this and report these results in Table 3.

Table 2: Evaluating the impact of sampling percentage of data on zero-shot performance.

% of Data	HMDB51	UCF101
1	53.7 ± 1.3	79.1 ± 1.6
5	**53.9 ± 1.1**	79.2 ± 1.5
10	**53.9 ± 1.4**	**79.4 ± 1.4**
20	53.2 ± 1.5	78.6 ± 1.0
50	49.7 ± 2.8	75.2 ± 2.6
100	48.6 ± 3.7	73.1 ± 3.4

Table 3: Using the text encoder from the base model as semantic embedding, compared to the other semantic embeddings we evaluate in this work.

Semantic Embedding	HMDB-51	UCF-101
X-Florence Text Encoder	47.2 ± 4.1	71.6 ± 3.8
Word2Vec	48.9 ± 4.1	73.9 ± 4.1
Sen2Vec	50.8 ± 3.1	76.7 ± 2.9
ER	51.9 ± 1.5	77.9 ± 1.3
Stories	**53.9 ± 1.4**	**79.4 ± 1.4**

We see that even using a simple Word2Vec embedding does better than using the text encoder to produce semantic embeddings.

5 How Much Does the Number of Generated Synthetic Samples Affect Model Performance?

We generate synthetic samples from already seen classes in order to train our classifier with synthetic features and refresh its memory. All experiments in the main paper were conducted ensuring that the distribution of the newly sampled classes and the synthetic features were similar, i.e., we generate a number of synthetic features roughly equal to the number of new real samples. We also experimented with generating fewer samples (20% to 70% of the average number of samples in the new classes) and more samples (150% to 200%). We report these results in Table 4, where 100% corresponds to what we do for all results in the main paper. We see that either over generating or under generating leads to poorer results, which suggests that keeping the number of synthetic and real features balanced is beneficial to the model.

6 Freezing the Feature Generator

As mentioned in the paper we freeze the feature generator after it is trained on the pre-training dataset. We do this both to save computation resources and

Table 4: Using different percentages of synthetic data (relative to the size of the new classes) to train the classifier.

Percentage of synthetic samples	HMDB-51	UCF-101
20	48.4 ± 2.5	74.2 ± 1.6
50	50.5 ± 1.9	76.1 ± 1.8
70	52.9 ± 1.8	78.6 ± 1.5
100	**53.9 ± 1.4**	**79.4 ± 1.4**
120	53.6 ± 1.7	77.9 ± 1.8
150	52.7 ± 1.6	77.2 ± 1.6
200	49.7 ± 1.3	76.4 ± 1.5

Table 5: Comparing results obtained fine-tuning and freezing the feature generator.

Fine-tune stage	HMDB-51	UCF-101
Fine-tune on seen classes	49.5 ± 1.9	75.2 ± 1.6
Incremental fine-tune	53.1 ± 1.2	77.9 ± 1.3
Frozen	**53.9 ± 1.4**	**79.4 ± 1.4**

because in practice freezing this models leads to better results. We show this comparing results obtained fine-tuning the feature generator on the seen classes (one single fine-tuning) and fine-tuning the feature generator in the incremental learning stage together with the other models. Results are reported in Table 5, where we see that indeed freezing the feature generator gives better results. We speculate this is the case because fine-tuning the generator makes the overall framework more difficult to optimize, i.e., it is easier to optimize the video model with features generated from a stable generator.

7 Experiments with CL in ZSL setup

All our experiments are in the ZSL (or generalized ZSL) setting, where test classes are disjoint with the training set at all times. We also evaluate here our model after each fine-tuning stage, i.e., after each time we introduce the 10% of new classes. As expected, performance grows steadily as we introduce more data as seen in Fig 1. This also justifies the idea of slowly introducing data as opposed to directly fine-tune using all data.

References

1. S. N. Gowda. Synthetic sample selection for generalized zero-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023. 2
2. S. N. Gowda and L. Sevilla-Lara. Telling stories for common sense zero-shot action recognition. *arXiv preprint arXiv:2309.17327*, 2023. 2

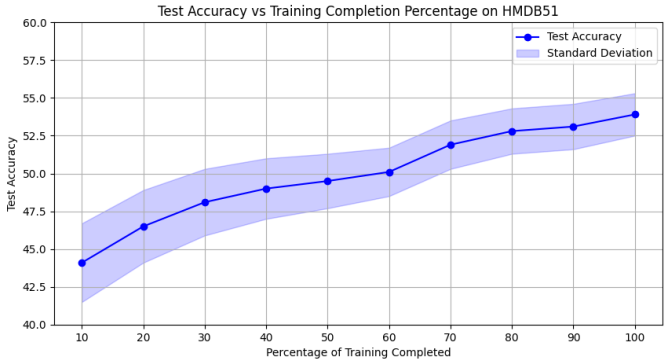

Fig. 1: Test accuracy versus training completion percentage.

3. S. N. Gowda, L. Sevilla-Lara, F. Keller, and M. Rohrbach. Claster: clustering with reinforcement learning for zero-shot action recognition. In *European Conference on Computer Vision*, 2022. 2
4. S. N. Gowda, L. Sevilla-Lara, K. Kim, F. Keller, and M. Rohrbach. A new split for evaluating true zero-shot action recognition. *arXiv preprint arXiv:2107.13029*, 2021. 2
5. H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, 2011. 2
6. D. Mandal, S. Narayan, S. K. Dwivedi, V. Gupta, S. Ahmed, F. S. Khan, and L. Shao. Out-of-distribution detection for generalized zero-shot action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2
7. B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling. Expanding language-image pretrained models for general video recognition. In *European Conference on Computer Vision*, 2022. 1
8. A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 2021. 1
9. K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR*, 2012. 2
10. Y. Xian, T. Lorenz, B. Schiele, and Z. Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. 1, 2
11. L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021. 1