

Continual Learning Improves Zero-Shot Action Recognition

Shreyank N Gowda¹, Davide Moltisanti², and Laura Sevilla-Lara³

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

Abstract. Zero-shot action recognition requires a strong ability to generalize from pre-training and seen classes to novel unseen classes. Similarly, continual learning aims to develop models that can generalize effectively and learn new tasks without forgetting the ones previously learned. The generalization goals of zero-shot and continual learning are closely aligned, however techniques from continual learning have not been applied to zero-shot action recognition. In this paper, we propose a novel method based on continual learning to address zero-shot action recognition. This model, which we call *Generative Iterative Learning* (GIL) uses a memory of synthesized features of past classes, and combines these synthetic features with real ones from novel classes. The memory is used to train a classification model, ensuring a balanced exposure to both old and new classes. Experiments demonstrate that *GIL* improves generalization in unseen classes, achieving a new state-of-the-art in zero-shot recognition across multiple benchmarks. Importantly, *GIL* also boosts performance in the more challenging generalized zero-shot setting, where models need to retain knowledge about classes seen before fine-tuning.

1 Introduction

Traditional supervised deep learning has shown prowess in diverse applications [21, 24, 35, 67], but its strong dependence on large amounts of labeled data [5, 9, 38] has become its Achilles' heel, especially when encountering novel data categories. Zero-shot learning (ZSL) opens up new possibilities, focusing on classifying data from novel categories that are unseen during training. Continual learning (CL) is another area of research that addresses the same fundamental challenges involved in novel data, though in a distinct way: learning new tasks (or data) without forgetting previous knowledge. Indeed, ZSL and CL share the objective of building representations with high generalization abilities. In ZSL, the goal is to generalize from pre-train and seen classes to unseen classes, while in CL the goal is to generalize from a task to the next one without losing performance in the first one. Previous work has applied CL to improve ZSL in the image domain [14, 26, 36, 43, 64], however, to the best of our knowledge, no previous work has leveraged CL for zero-shot action recognition in video.

In this work, we thus propose the first method for zero-shot action recognition based on a continual learning paradigm. We name our method **GIL: Generative Iterative Learning**. GIL consists of three stages: during initialization we build a Replay Memory storing prototypical video representations for all classes in a pre-training dataset. A Generative Adversarial Network (GAN) is also trained to produce features resembling those stored in the memory. In the incremental learning stage we then gradually fine-tune the video model on a fine-tuning dataset, mixing features generated with the GAN from the memory (old knowledge) with real features obtained for the new classes (new knowledge). The incremental learning stage is alternated with the update stage, where we refresh the memory adding prototypical representations from the new classes. This cycle ensures that the model learns new classes while not forgetting the old ones, which is the basis of continual learning. GIL has a strong generalization ability and improves recognition both in the standard zero-shot setting and the more challenging generalized zero-shot setting (GZSL), where the model is tested also on classes seen during pre-training. This is demonstrated on three standard ZSL/GZSL action recognition benchmarks (UCF-101 [55], HMDB-51 [34], Kinetics-600 [5,6]), where we improve by a wide margin of up to 20%. We publish our code at <https://sites.google.com/view/gil-accv>.

2 Related Work

Zero-Shot Learning in Action Recognition. Previous research focuses on establishing a shared embedding space between video features and semantic labels [60, 61]. Others explore error-correcting codes [49], pairwise class relationships [11], and interclass dynamics [12]. Other approaches include leveraging out-of-distribution detectors [39] and graph neural networks [13]. Recently, clustering of joint visual-semantic features was proposed [19]. This cluster-based technique and others, like ReST [37] and JigSawNet [48], emphasize the joint modeling of visual and textual features, albeit with varying approaches to bridge the visual and text spaces. Image-based foundational models have also shown to be excellent zero-shot learners in video. These models have been ported to videos with multi-modal prompting [58], self-regulating prompts [32] or by using cross-frame attention mechanism that explicitly exchanges information across frames [44]. Enhancing semantic embeddings from simple word2vec [41] to more elaborate definitions [6] has shown great promise for ZSL. For instance, action “Stories” [18] enhance the semantic space by breaking down actions into descriptive steps. We use semantic embeddings to build semantic class representations. We employ Stories [18] due to its good performance, and show that our method improves over the baseline model regardless of the semantic embedding we choose. We also evaluate the more challenging generalized zero-shot learning (GZSL) setting. Recent work [7] uses a CVAE to disentangle visual features into semantic-consistent and unrelated parts, enhancing generalization. [23] combines feature generation with contrastive embeddings to create a more discriminative space for both real and synthetic sample. [8] employs adversarial training with aug-

mented samples to maintain semantic consistency and robustness. Our method differs from these by incorporating continual learning with a replay memory to combine synthesized features of past classes with real features of new ones. This updates knowledge incrementally, unlike the other methods that focus on static feature manipulation [7, 23] or adversarial training [8].

Continual Learning. Traditional deep learning models suffer from catastrophic forgetting [40]. Continual learning aims at training models that can learn from a continuous stream of data without forgetting previously acquired knowledge [47]. Various methods have been proposed to tackle this. Some have explored the concept of elastic weight consolidation [33] to regularize the changes in network weights. Others have utilized memory augmentation strategies, like the use of external memory modules [22]. Progressive neural networks [50] expand the network architecture to leverage prior knowledge via lateral connections to previously learned features to alleviate catastrophic forgetting. There are also methods that leverage meta-learning principles, allowing the model to learn how to learn across tasks [10]. Recent work has focused on distilling knowledge from one part of the model to another [28], and generating instance level prompts [30]. Another promising avenue is the use of generative replay [52], where generated samples from previous tasks are mixed with new data to reduce forgetting. In this work we use a feature generator network to build a replay memory [52]. Instead of generating instance-like features, we generate class-like prototypical representations, which enhances efficiency. In class-incremental learning efficiency has also been investigated managing memory. [65, 66] suggest that focusing on different network layers, particularly the deeper ones, can improve memory efficiency and performance. This aligns with our approach, where we fine-tune the last two layers to leverage adaptability while maintaining core model stability.

Continual Learning and Zero-Shot Learning. As discussed before, CL and ZSL share the goal of learning generalizable representations. CL has been explored for ZSL in image tasks [14, 53, 64]. Most of these works convert the problem of standard zero-shot learning into that of a continual learning problem dubbed continual zero-shot learning (CZSL). CZSL mimics human lifelong learning by continuously incorporating new classes from the unseen world, evaluating the model on both seen and unseen categories. To tackle this, prior work has proposed class normalization [53], semantic guided random walks [64], experience replay with dark knowledge distillation [15] and meta-learned attributes [56]. Most similar to our work is the use of generative replay for CZSL [14] that uses attribute information to generate synthetic features. In contrast, we store class prototype and associated distribution as noise and generate synthetic data using this. Further, CL has not been used for zero-shot action recognition in videos, and catastrophic forgetting has not been evaluated in this context. We propose the first study of this problem in zero-shot action recognition and propose the first method for zero-shot action recognition based on continual learning.

3 Methodology

3.1 Zero-Shot Problem Formulation

Let P and S be respectively a pre-training and a fine-tuning dataset for seen classes. Each element in P and S is a tuple (x, y, a) . Here, x denotes the spatio-temporal features of a video; y is the class label from the set of seen classes Y_S ; and a is the semantic representation of class y . This representation can be manually annotated or derived automatically, e.g., with Stories [18]. Let U be the collection of pairs (u, a) , where u is a class from the set of unseen classes Y_U , and a indicates its corresponding semantic representation. The seen Y_S and unseen classes Y_U have no common elements. In the ZSL setting, given a video input the objective is to determine a class label from the unseen class set, represented as $f_{ZSL} : X \rightarrow Y_U$. In the Generalized Zero-Shot (GZSL) setting, given a video input, the goal is to identify a class label from the combined set of seen and unseen classes, expressed as $f_{GZSL} : X \rightarrow Y_S \cup Y_U$.

3.2 Preliminaries

We now describe the two main building blocks that our model is based on: feature generation from ZSL and replay memory from CL.

Feature Generation. The standard feature generation approach [39, 42, 54, 59] learns to generate visual features for unseen classes using a GAN [16]. Given these synthetic features, a classifier that takes in visual features is trained to predict unseen classes. More concretely, the GAN consists of a feature generator network (\mathcal{F}), a discriminator (G) and a projection network (H). Given the visual input x_i and its class label y_i , the generator takes in the semantic embedding a_i for the class label y_i and some noise z , and produces a synthetic visual feature vector \hat{x}_i . The projection network maps the visual feature x_i to an approximation of the semantic embedding \hat{a}_i . The discriminator’s job is to distinguish the synthesized features from the real features. All these networks are trained jointly, where the generator and the discriminator compete in a mini-max game. The optimization function to train these networks is as follows:

$$\begin{aligned} \mathcal{L}_D = & \mathbb{E}_{(x,a) \sim p_{(x_S \times a_S)}} [G(x, H(x))] \\ & - \mathbb{E}_{z \sim p_z} \mathbb{E}_{a \sim p_a} [(G(\mathcal{F}(a, z), a))] \\ & - \alpha \mathbb{E}_{z \sim p_z} \mathbb{E}_{a \sim p_a} \left[(\|\nabla_{\hat{x}} G(\mathcal{F}(a, z))\|_2 - 1)^2 \right], \end{aligned} \quad (1)$$

where $p_{(x_S \times a_S)}$ is the joint empirical distribution of visual and semantic descriptors of the seen classes, p_a is the empirical distribution of seen classes semantic embeddings, p_z is the noise distribution and α is a penalty coefficient.

There are two additional losses, used to improve the quality of the generated features. These are the classification regularized loss \mathcal{L}_{CLS} [59] and the mutual information loss \mathcal{L}_{MI} [2]. Putting all losses together, the objective function that we minimize to train the vanilla pipeline is:

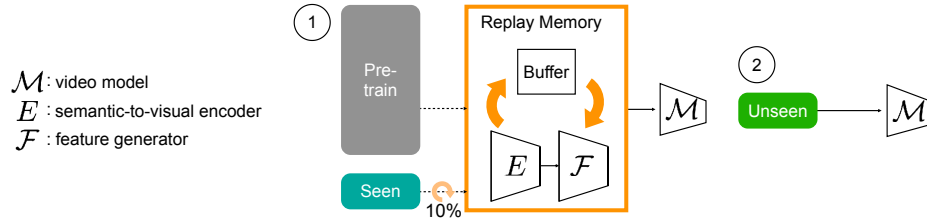


Fig. 1: GIL at a glance. The main idea is to use a Replay Memory [52] from Continual Learning to fine-tune a video model \mathcal{M} . The Replay Memory consists of a buffer which contains classes from the pre-training and fine-tuning datasets. Samples in the buffer are generated with a semantic-to-visual encoder E and a feature generator \mathcal{F} .

$$\min_{\mathcal{F}} \min_H \max_G \mathcal{L}_{\mathcal{D}} + \lambda_1 \mathcal{L}_{CLS}(\mathcal{F}) + \lambda_2 \mathcal{L}_{MI}(\mathcal{F}). \quad (2)$$

Once these networks are trained on the seen classes, the generator is used to synthesize visual features for the unseen classes.

Continual Learning. Formally, given a sequence of tasks T_1, T_2, \dots, T_n , where each task T_i consists of a dataset $D_i = (x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \dots$, the objective of CL is to perform well at T_i without losing accuracy on any of the tasks T_j , where $1 < j < i$. For ZSL and GZSL, we can consider the pre-training set P to be T_1 , each set of seen classes that are gradually added to the training set to be $T_{2, \dots, N-1}$ and the unseen classes to be T_N . Replay Memory [52] is a powerful technique in CL to mitigate catastrophic forgetting. The idea is to leverage generative models to generate or “replay” samples from past tasks while learning new ones, preserving the knowledge from earlier tasks. This is achieved by incorporating a replay buffer B , which stores a subset of samples from previous tasks. A generator \mathcal{F} is trained to generate samples that are representative of those in B . For a new task T_i , the model is trained on the current task data D_i as well as on the replayed samples generated by \mathcal{F} , which helps generalization ability and alleviates catastrophic forgetting.

3.3 Our Approach: Generative Iterative Learning (GIL)

A broad overview of our architecture can be seen in Fig. 1. The main idea is to use a Replay Memory to fine-tune a video model \mathcal{M} which we will use for zero-shot prediction. The Replay Memory ensures that \mathcal{M} does not lose its generalization abilities. The memory consists of a buffer, which contains data from the pre-training dataset as well as the seen classes from the fine-tuning dataset, which is incorporated progressively in small batches. The buffer starts off as real data and progressively includes synthetic data, generated by a conditional variational auto-encoder E and a feature generation network \mathcal{F} . The model is trained iteratively, alternating the fine-tuning stage with a memory update stage until all classes in the fine-tuning dataset are seen. We next describe each component

in detail, and later describe how we train and test the framework.

Foundation Model. Our approach is model-agnostic. We choose a foundation model \mathcal{M} to encode the videos, so that they will live in a meaningful semantic space, which is essential in ZSL. We use X-Florence [44] as our initial pre-trained model due to its good performance and will show results obtained with other backbones as well. X-Florence is a video model built on top of the image foundational model Florence [63]. X-Florence consists of three main components: a video encoder, a text encoder and a video-specific prompt generator that are pre-trained together (more details in [44]). We only use the video encoder. We adapt \mathcal{M} by adding two fully-connected layers at the end that generate a video representation in the same space as the textual representation, which enables testing with a nearest neighbor search. We keep the backbone of \mathcal{M} fixed at all times and only train these additional layers when fine-tuning \mathcal{M} .

Class Prototype and Noise. Models based on Replay Memory use a buffer that stores a subset of samples from each task. However, a random subset of samples does not necessarily reflect the true distribution of a class. Instead, we use class prototypes, which are the average feature representation (μ) of each sample (x_i) in a class (C). We also use the distance between a class prototype and each sample (the standard deviation $\sigma(C)$) to better approximate the class distribution.

Replay Memory. The Replay Memory [52] consists of a buffer and a Conditional Variational Autoencoder (CVAE) [54], which acts as a semantic-to-visual encoder. For each class seen in training we archive its prototype and noise in the buffer. The CVAE receives a semantic embedding and is trained to output a latent variable that is equivalent to the concatenation of the prototype $\mu(C)$ and noise $\sigma(C)$, for all classes in the buffer. We denote the CVAE as the model E . Finally, we use an off-the-shelf feature generator model [59] \mathcal{F} that has been successful for GZSL in video. Given a class prototype and data noise, \mathcal{F} is trained to create features of that class that resemble those of the model \mathcal{M} .

3.4 Training

Our training pipeline can be seen in Figure 2. GIL consists of three phases: an Initialization Phase that establishes the foundation model, an Incremental Phase that iteratively introduces new class data, and an Update Phase ensuring the model remains rooted in its previous knowledge while learning new classes.

Initialization Stage. The foundation model \mathcal{M} starts off as X-Florence [44] pre-trained on Kinetics¹. We freeze \mathcal{M} and use it to generate visual features for all instances belonging to each class i in the pre-training dataset. For each

¹ The model is pre-trained on Kinetics400 for the evaluation on Kinetics220, and on Kinetics605 for the evaluation on UCF and HMDB.

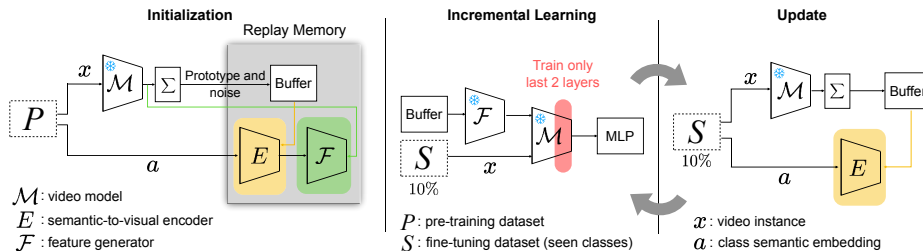


Fig. 2: Overview of GIL. The initialization stage involves booting the replay memory, i.e., storing pre-training class prototypes and noise obtained averaging video features encoded by \mathcal{M} . E is then trained to produce prototypes given a semantic embedding, and \mathcal{F} is trained to generate visual features from the output of E . In the incremental learning stage we fine-tune \mathcal{M} with a mix of synthetic and real features. Synthetic features are generated from the memory buffer, while real features are obtained with the backbone of \mathcal{M} . Real features are added gradually sampling a subset of new classes at the time. In the update stage we add prototypes and noise for the new classes to the buffer and fine-tune E with this new data.

class i we then calculate its prototype and noise, as described in Sec. 3.3. This is denoted in Fig. 2 as the module Σ . We store prototypes and noise μ_i, σ_i of all classes in the Replay Memory buffer. We then train the CVAE (E) to predict a prototype and noise ($\hat{\mu}_i, \hat{\sigma}_i$) given a semantic embedding (Stories [18] in our case). After this, we train the feature generation network \mathcal{F} to generate features for a class given its prototype and noise. These features should be similar to those encoded by \mathcal{M} . \mathcal{F} and E are trained from scratch. We use only \mathcal{F} after the initial training of the feature generator network and hence drop \mathcal{G} and \mathcal{H} from the notation and figures for clarity. \mathcal{F} is optimized as described in Eq 2. E is optimized using the mean squared error. After this stage, \mathcal{F} is permanently fixed.

Incremental Learning Stage. This stage is the core of our approach and is our contribution to incorporate CL to boost generalization in ZSL. In this stage we fine-tune the last two layers of \mathcal{M} both with synthetic features generated from \mathcal{F} as well as with real features from novel classes. The real features are generated using the backbone of \mathcal{M} , sampling videos belonging to a small subset of new classes at every iteration, which we set to be 10% of the total number of classes in the fine-tuning dataset. The synthetic features are generated using all the prototypes in the memory buffer. In the first iteration, the synthetic features will be only from the pre-training classes. In a subsequent iteration, they will be both from the pre-training classes and the new classes sampled until then. The gradual incorporation of new classes helps maintain the generalization of \mathcal{M} . More formally, at iteration t we generate features \hat{f}_j^k for each class k in the buffer seen during previous stages, which we denote with C_{t-1} :

$$\hat{f}_j^k = \mathcal{F}(\mu_k, \sigma_k), \forall k \in C_{t-1}, \forall j \in 1 \dots J \quad (3)$$

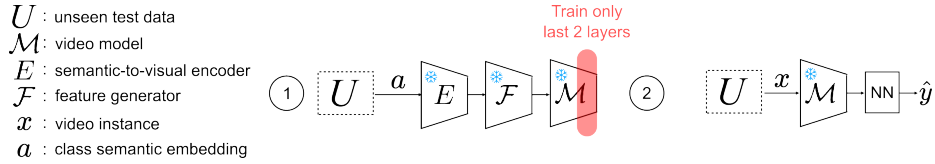


Fig. 3: Testing pipeline. (1) We project unseen classes fine-tuning \mathcal{M} with synthetic features. This is done by first feeding E a class semantic embedding. E then outputs a class prototype and noise, which are fed to \mathcal{F} to generate the visual features we use to fine-tune \mathcal{M} . (2) After fine-tuning, given a test video instance we perform a nearest neighbor (NN) search to predict class \hat{y} .

where J is the number of generated features for class k , which we choose to be in the range of the average number of samples in the new set of classes. Real features f_j^k are computed for the new classes in C_t using the backbone of \mathcal{M} :

$$f_j^k = \mathcal{M}(x_j), \forall k \in C_t, \forall j \in |C_t(k)| \quad (4)$$

where x_j is a video from a new class k and $|C_t(k)|$ is the number of samples in the class. \mathcal{M} is fine-tuned on both \hat{f}_j^k and f_j^k to predict their class labels. We attach a multi-layer perceptron classifier to fine-tune \mathcal{M} , using the standard cross-entropy loss. This classifier is used only to fine-tune \mathcal{M} , i.e., we use the output of \mathcal{M} with a nearest neighbor search for prediction during inference, not the output of the classifier (more details on testing later).

Update Stage. Once the model has been trained on the new set of classes C_t , an update process ensures a smooth transition to the next iteration. The new class prototypes and their noise structure are first archived into the Memory Buffer. E is then fine-tuned to generate prototypes and noise for the new classes, while \mathcal{F} remains unchanged. The iterative process of incremental learning and memory update continues until all training (seen) classes are incorporated in the buffer.

3.5 Testing

After training the model on all seen classes, we feed the semantic embeddings of the unseen classes U to E . This gives us class prototypes and noise, which we feed to \mathcal{F} . This in turn generates unseen class features that are used to fine-tune \mathcal{M} with the additional classifier we employ during the incremental learning stage. We fine-tune \mathcal{M} only on the unseen classes in U in the ZSL setting and on both seen and unseen classes $U \cup S$ in GZSL. This is depicted in Figure 3. We highlight that we do not use any test video instance for fine-tuning. We only use class semantic embeddings (to generate visual features), which is common practice in ZSL [3, 19, 39]. At test time, a video sample x_i from an unseen class is fed to \mathcal{M} to produce an embedding. We assign x_i the class of the nearest neighbor (K -NN with $K = 1$) in the embedding space.

4 Experimental Analysis

In this section we first describe the datasets we use in our experiments. We then measure catastrophic forgetting in the standard zero-shot setting, which shows the loss of generalization that happens during the fine-tuning process. We also compare GIL to the current state-of-the-art in the ZSL and GZSL settings. We observe that GIL improves by a wide margin, with up to 20% improvement with respect to previously published work. We finally present several ablation studies. Implementation details can be found in the supplementary material.

Datasets and Evaluation Metric. We conduct experiments on the HMDB-51 [34] and UCF-101 [55] datasets, which are standard choices in zero-shot action recognition. These datasets comprise 6,766 and 13,320 videos and encompass 51 and 101 classes, respectively. We follow the widely adopted 50/50 splits introduced by Xu et al. [60], where 50% of the classes are designated as seen, and the other 50% are designated as unseen. All models are pre-trained on the non-overlapping classes [3] of Kinetics-400 [5]. Since we pre-train our model to avoid overlap to these datasets, the overlap issues that zero-shot action recognition may suffer from [20] is not a concern. We also investigate the Kinetics-600 split [5] proposed in ER [6]. In this configuration, the 200 classes from Kinetics-600 [4] are treated as unseen classes for a model trained on the Kinetics-400 dataset. Unless otherwise stated, following previous work [18,49,68] we report the average top-1 accuracy and standard deviation from 10 independent runs.

4.1 Measuring Catastrophic Forgetting in Zero-Shot Learning

We first show the problem of catastrophic forgetting from a quantifiable point of view. We consider four backbones: X-CLIP-B/16 [44], X-CLIP-B/32 [44], X-Florence [44] and Vita-CLIP-B/16 [58]. We pre-train the models on Kinetics and fine-tune them on UCF-101 with and without GIL. When not using GIL we follow a standard zero-shot approach based on a feature generation network [59], i.e., we remove the replay memory and the incremental learning cycle during training, while the video backbone and the feature generator remain the same (more details in the supplementary material). We also evaluate the models on the pre-training dataset (Kinetics), as reported in Table 1. While we see a large improvement on the zero-shot setting with GIL (up to 6.2%), we see an even larger improvement when testing on the pre-training dataset (up to 24.7%). We make the following observations from these results. First, there is a clear problem of catastrophic forgetting where the model overfits to the seen classes after fine-tuning (performance without GIL on Kinetics is poor). Second, CL mitigates this problem (performance with GIL is good on Kinetics). Third, features generated with GIL generalize better and do not overfit to the seen classes, as GIL also improves ZSL performance on UCF-101.

Table 1: Evaluating catastrophic forgetting in ZSL. Results indicate top-1 accuracy from a single run. Models were pre-trained on Kinetics-400, fine-tuned on UCF-101, and tested on both UCF-101 and Kinetics. We compare a standard zero-shot approach based on a feature generation network [59] to our GIL approach.

Model		GIL Zero-shot (UCF-101)	Pre-training (Kinetics)
X-Clip B/16 [44]	×	72.0	50.2
X-Clip B/16 [44]	✓	77.3	71.8
X-Clip B/32 [44]	×	70.8	49.5
X-Clip B/32 [44]	✓	76.1	71.1
X-Florence [44]	×	73.2	53.7
X-Florence [44]	✓	79.4	78.5
Vita-CLIP [58]	×	75.0	51.8
Vita-CLIP [58]	✓	77.9	73.3

4.2 Comparison to State-Of-The-Art

Zero-shot setting. We compare to recent state-of-the-art (SOTA) models on the ZSL setting where the evaluation is done only on the unseen classes. Recent SOTA use vision-language training, however we also list earlier vision-only methods for completeness. Results on UCF-101 and HMDB-51 are reported in Table 2, while Table 4 reports results obtained on Kinetics200 setting [6]. We note that our method attains new state-of-the-art. We also test popular image-based zero-shot CL frameworks [14, 15] using the same backbone and semantic embeddings and see GIL improves by 8-10%. In Table 3 we also compare to recent SOTA on the GZSL setting, where the evaluation is done on both seen and unseen classes. This is a less explored but much more challenging scenario. We see improvements of up to 19.7% in this setting. These results show that GIL attains new state-of-the-art. In ZSL we see that GIL learns features that generalize better and boost zero-shot performance, while in GZSL we note that the model also better retains pre-training knowledge thanks to the CL paradigm.

4.3 Ablation Study

Effect of Replay Memory. In Table 5 we explore different design choices for the replay memory. First, we remove the memory altogether. Second, we store some samples for each class chosen randomly instead of storing the class prototype and standard deviation. Third, we replace the standard deviation with a few more random samples (shown in the table as Class Prototype + Random Samples). All options improve over the baseline of not using a memory replay. We also observe that the choice of class representation has a strong effect on the results, and using the proposed class prototype and data noise is the most effective of all. Storing the class distribution as noise also significantly lowers the standard deviation in the reported top-1 accuracy, i.e., results obtained with this choice of replay memory are more robust as they vary less.

Table 2: Comparing GIL to SOTA in the zero-shot setting on HMDB-51 and UCF-101.

Vision-only Methods	Backbone	HMDB-51	UCF-101
OD [39] (<i>CVPR'19</i>)	I3D	30.2 ± 2.7	26.9 ± 2.8
OD + SPOT [17] (<i>CVPRW'23</i>)	I3D	34.4 ± 2.2	40.9 ± 2.6
GGM [42] (<i>WACV'19</i>)	C3D	20.7 ± 3.1	20.3 ± 1.9
E2E [3] (<i>CVPR'20</i>)	C3D	32.7	48.0
ER-ZSAR [6] (<i>ICCV'21</i>)	TSM	35.3 ± 4.6	51.8 ± 2.9
CLASTER [19] [6] (<i>ECCV'22</i>)	I3D	42.6 ± 2.6	52.7 ± 2.2
JigSawNet [48] (<i>ECCV'22</i>)	R(2+1)D	39.3 ± 3.9	56.8 ± 2.8
RGSCl [51] (<i>KBS'24</i>)	I3D	34.1 ± 2.9	40.2 ± 3.8
SupVFD [45] (<i>ExSys'24</i>)	I3D	31.9 ± 3.2	39.6 ± 2.9
Vision-Language Methods	Backbone	HMDB-51	UCF-101
A5 [29] (<i>ECCV'22</i>)	CLIP-B/16	44.3 ± 2.2	69.3 ± 4.2
X-CLIP-B/16 [44] (<i>ECCV'22</i>)	CLIP-B/16	44.6 ± 5.2	72.0 ± 2.3
Vita-CLIP-B/16 [58] (<i>CVPR'23</i>)	CLIP-B/16	48.6 ± 0.6	75.0 ± 0.6
EPK-CLIP [62] (<i>ExSys'24</i>)	CLIP-B/16	48.7 ± 0.7	75.3 ± 0.9
VicTR [31] (<i>CVPR'24</i>)	CLIP-B/16	51.0 ± 1.3	72.4 ± 0.3
M ² -CLIP [57] (<i>AAAI'24</i>)	CLIP-B/16	47.1 ± 0.4	78.7 ± 1.2
GIL (Ours)	CLIP-B/16	52.7 ± 1.1	77.9 ± 1.4
Tf-gczsl [15]	Florence	45.4 ± 3.5	68.6 ± 2.6
GRCZSL [14]	Florence	45.9 ± 3.1	69.8 ± 3.9
X-Florence [44] (<i>ECCV'22</i>)	Florence	48.4 ± 4.9	73.2 ± 4.2
SDR [18] (<i>Arxiv'23</i>)	Florence	52.7 ± 3.4	75.5 ± 3.2
GIL (Ours)	Florence	53.9 ± 1.4	79.4 ± 1.4

Table 5: Comparing different choices of replay memory in the zero-shot setting. No Mem: No Replay Memory, Rand: Random Sampling, Proto + Rand: Class Prototype + Random Sampling, Proto + Noise: Class Prototype + Data Noise.

Method	HMDB-51	UCF-101
No Mem	48.7 ± 4.4	73.5 ± 3.5
Rand	49.4 ± 4.1	75.1 ± 2.9
Proto + Rand	50.6 ± 3.8	76.6 ± 2.7
Proto + Noise	53.9 ± 1.4	79.4 ± 1.4

Table 6: Comparing different backbones in the zero-shot setting. CLIP [44]: X-CLIP-B/16 (*ECCV'22*), X-F [44]: X-Florence [44] (*ECCV'22*), Vita [58]: Vita-CLIP-B/16 [58] (*CVPR'23*)

Backbone	HMDB-51	UCF-101
CLIP	44.6 ± 5.2	72.0 ± 2.3
CLIP + GIL	48.9 ± 1.9	77.3 ± 1.5
X-F	48.4 ± 4.9	73.2 ± 4.2
X-F + GIL	53.9 ± 1.4	79.4 ± 1.4
Vita	48.6 ± 0.6	75.0 ± 0.6
Vita + GIL	52.7 ± 1.1	77.9 ± 1.4

Choice of Backbone. GIL is model-agnostic. We chose X-Florence [44] due to its strong performance on zero shot learning, and explore now the effect of using other backbones. Results can be seen in Table 6. We note that all backbones improve when trained with GIL, which highlights its generalizability. Since we do not make any specific assumptions on the backbone architecture, we believe

Table 3: Comparing GIL to SOTA in the generalized zero-shot setting on HMDB-51 and UCF-101. Reported values show the harmonic mean of the seen and unseen accuracies over 10 runs, $\pm std$. More details in the supplementary.

Vision-only Methods	Backbone	HMDB-51	UCF-101
WGAN [59] (<i>CVPR'18</i>)	I3D	32.7 ± 3.4	44.4 ± 3.0
OD [39] (<i>CVPR'19</i>)	I3D	36.1 ± 2.2	49.4 ± 2.4
GGM [42] (<i>WACV'19</i>)	C3D	20.1 ± 2.1	23.7 ± 1.2
CLASTER [19] [6] (<i>ECCV'22</i>)	I3D	50.8 ± 2.8	52.8 ± 2.1
RGSL [51] (<i>KBS'24</i>)	I3D	37.6 ± 2.4	50.0 ± 3.0
SupVFD [45] (<i>ExSys'24</i>)	I3D	37.0 ± 2.2	50.2 ± 2.8
FR-VAEGAN [27] (<i>Access'24</i>)	I3D	38.6 ± 0.1	54.7 ± 0.2
Vision-Language Methods	Backbone	HMDB-51	UCF-101
SDR [18] (<i>ArXiv'23</i>)	Florence	53.5 ± 3.3	57.8 ± 4.1
GIL (Ours)	Florence	55.1 ± 1.9	77.5 ± 1.9

GIL is a general framework that can be plugged to any model.

Using Real Data from Kinetics. During the incremental learning stage we use both synthetic and real features from the *fine-tuning* dataset, but only use synthetic features from the *pre-training* dataset (generated from the memory). A natural question to ask is thus why not use real data from the pre-training dataset as well during the incremental learning stage. We explore this question and show results with the zero-shot setting in Table 7. Here we vary how data from the *pre-training* dataset is obtained, however for the *fine-tuning* dataset we keep using a mix of synthetic and real data. We consider three scenarios. First, we sample only real data during the incremental stage, i.e., we do not use the feature-generating network and instead use real features obtained with the video backbone. In the second scenario we use a combination of real and synthesized data (with a 50:50 ratio). The last scenario is GIL’s default setting, using purely synthetic data. The baseline in this experiment is the original X-Florence model. Regardless of the type of data we use for the pre-training dataset, the replay memory improves over the baseline significantly, and using purely synthetic data gives the best performance. We hypothesize this happens because synthetic features tend to form a more compact representation, which is beneficial to fine-tune the video model. In Figure 4 we show this, plotting t-SNE projections [25] of real and synthetic features for a random set of 10 classes in HMDB-51². Indeed, synthetic features form more compact clusters, whereas real features are more scattered. This is not surprising as synthetic features are generated from class prototypes while real features are obtained from video instances.

Choice of Semantic Embedding. We use Stories [18] as our semantic embedding. Here we compare with other encodings: word2vec [41], sen2vec [46]

² We show 10 classes for clarity, but plots were identical with several random samples.

Table 4: Comparing GIL to SOTA in the zero-shot setting on Kinetics-200. Following ER [6], seen classes are those in Kinetics-400, while the new 200 classes in Kinetics-600 are unseen. Results report average top-1/5 accuracy $\pm std$ from 10 runs.

Vision-only Methods	Backbone	Top-1	Top-5
SJE [1] (<i>CVPR'15</i>)	TSM	22.3 \pm 0.6	48.2 \pm 0.4
ER-ZSAR [6] (<i>ICCV'21</i>)	TSM	42.1 \pm 1.4	73.1 \pm 0.3
JigSawNet [48] (<i>ECCV'22</i>)	R(2+1)D	45.9 \pm 1.6	78.8 \pm 1.0
SDR-I3D [18] (<i>ArXiv'23</i>)	I3D	50.8 \pm 1.9	82.9 \pm 1.3
Vision-Language Methods	Backbone	Top-1	Top-5
X-CLIP-B/16 [44] (<i>ECCV'22</i>)	CLIP B/16	65.2 \pm 0.4	86.1 \pm 0.8
Vita-CLIP-B/16 [58] (<i>CVPR'23</i>)	CLIP B/16	67.4 \pm 0.5	86.9 \pm 0.6
GIL (Ours)	CLIP B/16	70.9 \pm 1.3	90.7 \pm 1.2
SDR [18] (<i>ArXiv'23</i>)	Florence	55.1 \pm 2.2	86.1 \pm 3.1
X-Florence [44] (<i>ECCV'22</i>)	Florence	68.8 \pm 0.9	88.4 \pm 0.6
GIL (Ours)	Florence	72.5 \pm 1.6	93.1 \pm 1.1

and ER [6]. We report results from this experiment with the zero-shot setting in Table 8. We note that GIL improves over the baseline (the original X-Florence model) regardless of the semantic embedding we use. This further highlights the generalizability of our approach. Results with Word2Vec improve only marginally. We believe this is due to the limited semantic space word2vec is able to encode in comparison to the others.

Fine-tuning the Feature Generator. We keep the feature generator fixed after initialization. We do this for two reasons. Firstly, training a feature generator is expensive, and we wish to keep computational cost low. More importantly, we did not find a discernible difference in performance when fine-tuning the feature generator. More results on this in the Supplemental Material.

Table 7: Comparing types of data from the pre-training dataset for the incremental stage.

Type of Data	HMDB-51	UCF-101
Baseline [44]	48.4 \pm 4.9	73.2 \pm 4.2
Real Only	51.8 \pm 2.4	76.5 \pm 2.2
Real + Synthetic	52.7 \pm 1.9	78.5 \pm 2.4
Synthetic Only	53.9 \pm 1.4	79.4 \pm 1.4

Table 8: Comparing semantic embeddings for the incremental stage.

SE	HMDB-51	UCF-101
Baseline [44]	48.4 \pm 4.9	73.2 \pm 4.2
Word2Vec [41]	48.9 \pm 4.1	73.9 \pm 4.1
Sen2Vec [46]	50.8 \pm 3.1	76.7 \pm 2.9
ER [6]	51.9 \pm 1.5	77.9 \pm 1.3
Stories [18]	53.9 \pm 1.4	79.4 \pm 1.4

5 Conclusion

This paper introduces Generative Iterative Learning (GIL), a method that combines a generative approach for zero-shot learning within a continual learning paradigm. This work is the first to use CL to improve zero-shot action recognition

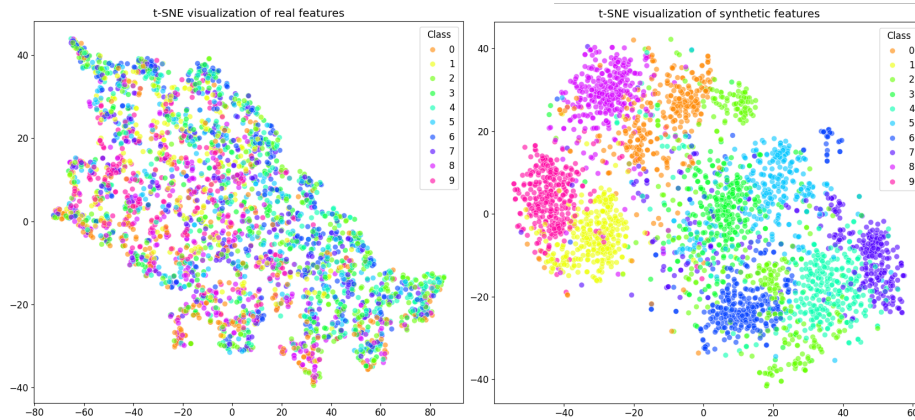


Fig. 4: Comparing real (left) vs real (right) features using t-SNE projections. We show the embeddings obtained from 10 random classes in HMDB-51. Synthetic features appear more compact than real features, which is beneficial to train the video model. We believe this is the reason why using exclusively synthetic features for the pre-training classes works better than using real features or a mix of the two.

in video. The core innovations of GIL lie in its replay memory and incremental learning. The memory stores class prototypes that help synthesize features from both past classes and new, real examples. The continual updating of this memory enables the retraining of the classification model in a balanced manner, ensuring equal exposure to old and new information. Experiments show that GIL significantly boosts the model’s generalization capabilities in unseen classes. Results demonstrate we achieve a new state-of-the-art in zero-shot action recognition on three standard benchmarks, irrespective of the video backbone we use.

Limitations. One of the possible limitations of GIL is that relying on synthesized features could introduce biases or inaccuracies in feature representation, potentially affecting the model’s performance on highly varied or dynamic datasets. Further, evaluating on datasets with viewpoints that differ from those seen in training may not benefit from the proposed approach.

Future Work. While this work has shown great promise in the integration of a continual learning framework for action recognition in the zero-shot setting, the idea could potentially be tested on few-shot and semi-supervised settings as the problem of catastrophic forgetting would still exist. Further, GIL could also be applied for other video tasks such as video localization.

References

1. Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B.: Evaluation of output embeddings for fine-grained image classification. In: Proceedings of the IEEE conference

- on computer vision and pattern recognition (2015) [13](#)
2. Belghazi, M.I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., Hjelm, D.: Mutual information neural estimation. In: International conference on machine learning (2018) [4](#)
 3. Brattoli, B., Tighe, J., Zhdanov, F., Perona, P., Chalupka, K.: Rethinking zero-shot video classification: End-to-end training for realistic applications. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020) [8](#), [9](#), [11](#)
 4. Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., Zisserman, A.: A short note about kinetics-600. arXiv preprint arXiv:1808.01340 (2018) [9](#)
 5. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017) [1](#), [2](#), [9](#)
 6. Chen, S., Huang, D.: Elaborative rehearsal for zero-shot action recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) [2](#), [9](#), [10](#), [11](#), [12](#), [13](#)
 7. Chen, Z., Luo, Y., Qiu, R., Wang, S., Huang, Z., Li, J., Zhang, Z.: Semantics disentangling for generalized zero-shot learning. In: Proceedings of the IEEE/CVF international conference on computer vision (2021) [2](#), [3](#)
 8. Chen, Z., Zhang, P., Li, J., Wang, S., Huang, Z.: Zero-shot learning by harnessing adversarial samples. In: Proceedings of the 31st ACM International Conference on Multimedia (2023) [2](#), [3](#)
 9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition (2009) [1](#)
 10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. Proceedings of the 34th international conference on machine learning (2017) [3](#)
 11. Gan, C., Lin, M., Yang, Y., De Melo, G., Hauptmann, A.G.: Concepts not alone: Exploring pairwise relationships for zero-shot video activity recognition. In: Thirtieth AAAI conference on artificial intelligence (2016) [2](#)
 12. Gan, C., Lin, M., Yang, Y., Zhuang, Y., Hauptmann, A.G.: Exploring semantic inter-class relationships (sir) for zero-shot action recognition. In: Proceedings of the National Conference on Artificial Intelligence (2015) [2](#)
 13. Gao, J., Zhang, T., Xu, C.: I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence (2019) [2](#)
 14. Gautam, C., Parameswaran, S., Mishra, A., Sundaram, S.: Generative replay-based continual zero-shot learning. arXiv preprint arXiv:2101.08894 (2021) [1](#), [3](#), [10](#), [11](#)
 15. Gautam, C., Parameswaran, S., Mishra, A., Sundaram, S.: Tf-gczsl: Task-free generalized continual zero-shot learning. Neural Networks (2022) [3](#), [10](#), [11](#)
 16. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems (2014) [4](#)
 17. Gowda, S.N.: Synthetic sample selection for generalized zero-shot learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2023) [11](#)
 18. Gowda, S.N., Sevilla-Lara, L.: Telling stories for common sense zero-shot action recognition. arXiv preprint arXiv:2309.17327 (2023) [2](#), [4](#), [7](#), [9](#), [11](#), [12](#), [13](#)

19. Gowda, S.N., Sevilla-Lara, L., Keller, F., Rohrbach, M.: Cluster: clustering with reinforcement learning for zero-shot action recognition. In: European Conference on Computer Vision (2022) [2](#), [8](#), [11](#), [12](#)
20. Gowda, S.N., Sevilla-Lara, L., Kim, K., Keller, F., Rohrbach, M.: A new split for evaluating true zero-shot action recognition. arXiv preprint arXiv:2107.13029 (2021) [9](#)
21. Gowda, S.N., Yuan, C.: Colornet: Investigating the importance of color spaces for image classification. In: Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part IV 14. pp. 581–596. Springer (2019) [1](#)
22. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al.: Hybrid computing using a neural network with dynamic external memory. *Nature* (2016) [3](#)
23. Han, Z., Fu, Z., Chen, S., Yang, J.: Contrastive embedding for generalized zero-shot learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2021) [2](#), [3](#)
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016) [1](#)
25. Hinton, G., van der Maaten, L.: Visualizing data using t-sne *journal of machine learning research* (2008) [12](#)
26. Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D., Yan, R.: Overcoming catastrophic forgetting for continual learning via model adaptation. In: International conference on learning representations (2018) [1](#)
27. Huang, K., Mckeever, S., Miralles-Pechuán, L.: Generalised zero-shot learning for action recognition fusing text and image gans. *IEEE Access* (2024) [12](#)
28. Hung, C.Y., Tu, C.H., Wu, C.E., Chen, C.H., Chan, Y.M., Chen, C.S.: Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems* (2019) [3](#)
29. Ju, C., Han, T., Zheng, K., Zhang, Y., Xie, W.: Prompting visual-language models for efficient video understanding. In: European Conference on Computer Vision (2022) [11](#)
30. Jung, D., Han, D., Bang, J., Song, H.: Generating instance-level prompts for rehearsal-free continual learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2023) [3](#)
31. Kahatapitiya, K., Arnab, A., Nagrani, A., Ryoo, M.S.: Victr: Video-conditioned text representations for activity recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024) [11](#)
32. Khattak, M.U., Wasim, S.T., Naseer, M., Khan, S., Yang, M.H., Khan, F.S.: Self-regulating prompts: Foundational model adaptation without forgetting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2023) [2](#)
33. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* (2017) [3](#)
34. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: Hmdb: a large video database for human motion recognition. In: 2011 International Conference on Computer Vision (2011) [2](#), [9](#)

35. Li, K., Wang, Y., Zhang, J., Gao, P., Song, G., Liu, Y., Li, H., Qiao, Y.: Uniformer: Unifying convolution and self-attention for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023) [1](#)
36. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In: *International Conference on Machine Learning* (2019) [1](#)
37. Lin, C.C., Lin, K., Wang, L., Liu, Z., Li, L.: Cross-modal representation learning for zero-shot action recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022) [2](#)
38. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: *European Conference on Computer Vision* (2014) [1](#)
39. Mandal, D., Narayan, S., Dwivedi, S.K., Gupta, V., Ahmed, S., Khan, F.S., Shao, L.: Out-of-distribution detection for generalized zero-shot action recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019) [2](#), [4](#), [8](#), [11](#), [12](#)
40. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation* (1989) [3](#)
41. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems* (2013) [2](#), [12](#), [13](#)
42. Mishra, A., Verma, V.K., Reddy, M.S.K., Arulkumar, S., Rai, P., Mittal, A.: A generative approach to zero-shot and few-shot action recognition. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018) [4](#), [11](#), [12](#)
43. Nguyen, C.V., Achille, A., Lam, M., Hassner, T., Mahadevan, V., Soatto, S.: Toward understanding catastrophic forgetting in continual learning. *arXiv preprint arXiv:1908.01091* (2019) [1](#)
44. Ni, B., Peng, H., Chen, M., Zhang, S., Meng, G., Fu, J., Xiang, S., Ling, H.: Expanding language-image pretrained models for general video recognition. In: *European Conference on Computer Vision* (2022) [2](#), [6](#), [9](#), [10](#), [11](#), [13](#)
45. Niu, C., Shang, J., Zhou, Z., Yang, J.: Superclass-aware visual feature disentangling for generalized zero-shot learning. *Expert Systems with Applications* (2024) [11](#), [12](#)
46. Pagliardini, M., Gupta, P., Jaggi, M.: Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507* (2017) [12](#), [13](#)
47. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. *Neural Networks* (2019) [3](#)
48. Qian, Y., Yu, L., Liu, W., Hauptmann, A.G.: Rethinking zero-shot action recognition: Learning from latent atomic actions. In: *European Conference on Computer Vision* (2022) [2](#), [11](#), [13](#)
49. Qin, J., Liu, L., Shao, L., Shen, F., Ni, B., Chen, J., Wang, Y.: Zero-shot action recognition with error-correcting output codes. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) [2](#), [9](#)
50. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016) [3](#)
51. Shang, J., Niu, C., Tao, X., Zhou, Z., Yang, J.: Generalized zero-shot action recognition through reservation-based gate and semantic-enhanced contrastive learning. *Knowledge-Based Systems* (2024) [11](#), [12](#)

52. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. *Advances in neural information processing systems* (2017) [3](#), [5](#), [6](#)
53. Skorokhodov, I., Elhoseiny, M.: Class normalization for (continual)? generalized zero-shot learning. In: *International Conference on Learning Representations* (2020) [3](#)
54. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems* (2015) [4](#), [6](#)
55. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR* (2012) [2](#), [9](#)
56. Verma, V., Mehta, N., Liang, K.J., Mishra, A., Carin, L.: Meta-learned attribute self-interaction network for continual and generalized zero-shot learning. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2024) [3](#)
57. Wang, M., Xing, J., Jiang, B., Chen, J., Mei, J., Zuo, X., Dai, G., Wang, J., Liu, Y.: A multimodal, multi-task adapting framework for video action recognition. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2024) [11](#)
58. Wasim, S.T., Naseer, M., Khan, S., Khan, F.S., Shah, M.: Vita-clip: Video and text adaptive clip via multimodal prompting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023) [2](#), [9](#), [10](#), [11](#), [13](#)
59. Xian, Y., Lorenz, T., Schiele, B., Akata, Z.: Feature generating networks for zero-shot learning. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018) [4](#), [6](#), [9](#), [10](#), [12](#)
60. Xu, X., Hospedales, T., Gong, S.: Transductive zero-shot action recognition by word-vector embedding. *International Journal of Computer Vision* (2017) [2](#), [9](#)
61. Xu, X., Hospedales, T.M., Gong, S.: Multi-task zero-shot action recognition with prioritised data augmentation. In: *European Conference on Computer Vision* (2016) [2](#)
62. Yang, Z., An, G., Zheng, Z., Cao, S., Wang, F.: Epk-clip: External and priori knowledge clip for action recognition. *Expert Systems with Applications* (2024) [11](#)
63. Yuan, L., Chen, D., Chen, Y.L., Codella, N., Dai, X., Gao, J., Hu, H., Huang, X., Li, B., Li, C., et al.: Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432* (2021) [6](#)
64. Zhang, W., Janson, P., Yi, K., Skorokhodov, I., Elhoseiny, M.: Continual zero-shot learning through semantically guided generative random walks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023) [1](#), [3](#)
65. Zhou, D.W., Wang, Q.W., Qi, Z.H., Ye, H.J., Zhan, D.C., Liu, Z.: Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024) [3](#)
66. Zhou, D.W., Wang, Q.W., Ye, H.J., Zhan, D.C.: A model or 603 exemplars: Towards memory-efficient class-incremental learning. In: *ICLR* (2023) [3](#)
67. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. *ICLR* (2021) [1](#)
68. Zhu, Y., Long, Y., Guan, Y., Newsam, S., Shao, L.: Towards universal representation for unseen action recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) [9](#)