Back to the Future: Cycle Encoding Prediction for Self-supervised Video Representation Learning

Xinyu Yang
xinyu.yang@bristol.ac.uk
Majid Mirmehdi
m.mirmehdi@bristol.ac.uk
Tilo Burghardt
tilo@cs.bris.ac.uk

Department of Computer Science
Faculty of Engineering
University of Bristol
Bristol, UK

Abstract

We show that learning video feature spaces in which temporal cycles are maximally predictable benefits action classification. In particular, we propose a novel learning approach, Cycle Encoding Prediction (CEP), that is able to effectively represent high-level spatio-temporal structure of unlabelled video content. CEP builds a latent space wherein the concept of closed forward-backward, as well as backward-forward, temporal loops is approximately preserved. As a self-supervision signal, CEP leverages the bi-directional temporal coherence of entire video snippets and applies loss functions that encourage both temporal cycle closure and contrastive feature separation. Architecturally, the underpinning network architecture utilises a single feature encoder for all input videos, adding two predictive modules that learn temporal forward and backward transitions. We apply our framework for pretext training of networks for action recognition and report significantly improved results for the standard datasets UCF101 and HMDB51.

1 Introduction

Videos constitute highly structured objects which offer rich and intrinsically correlated information that is often suitable as a self-supervision signal for unsupervised representation learning. Yet, defining which exact aspects of the video should be exploited for effective learning of semantically relevant embeddings, and how the resulting latent spaces are to be constructed, structured, and constrained, is a topic of active research [2, 3, 5, 7, 8, 9, 43].

In this paper, we take a closer look at exploiting the bidirectional temporal structure of video to support the learning of semantically relevant high-level spatio-temporal feature spaces. In particular, as illustrated in Fig. 1, we show that learning feature spaces, in which temporal cycles (going back and forth in time) become maximally predictable, can benefit action classification. Intuitively, any latent space that is suitable for classifying actions should carry a transitive structure which encodes the distinctive 'playing out' of actions. Walking along a path of forward predictions followed by an equal number of backward predictions (a bi-directional temporal cycle) should cancel out and provide an exploitable self-supervision criterion for pre-training. Note, this does not model multi-hypothesis predictions [38, 40].

Following this idea, we propose a novel pretext learning approach, named Cycle Encoding Prediction (CEP), which operates on unlabelled video and is driven by bi-directional temporal cycle closure. It aims at constructing a latent embedding wherein sequences of

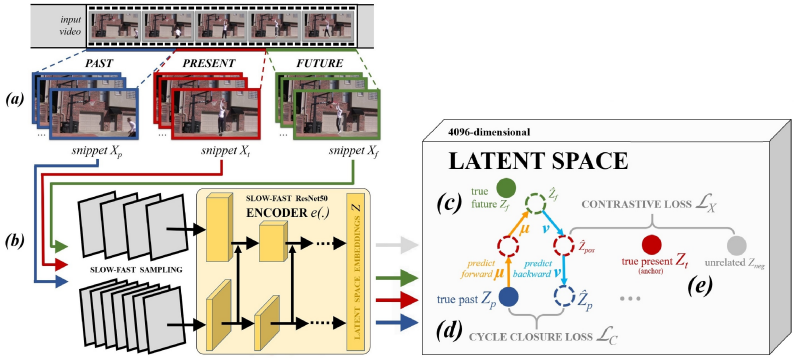

Figure 1: **Overview of Self-Supervised CEP Approach.** We represent videos in a feature space wherein bi-directional temporal cycles are maximally predictable. ***(a)*** The video data is sampled into snippets where each snippet X_t has a past X_p and future X_f neighbour. ***(b)*** In pretext learning, a latent space $Z_t = e(X_t)$ is generated via an encoder e (such as 3DResNet or SlowFast). ***(c)*** In this space, we encourage that learnt temporal forward μ and backward υ predictions form closed cycles (example cycle from past Z_p to some future $\hat{Z}_f$ and back to some past $\hat{Z}_p$ shown). ***(d)*** Embedding e and predictors μ and υ are co-trained in self-supervised mode, minimising a loss $\mathcal{L}_C$ that favours cycle closure. ***(e)*** Further, we apply a contrastive loss $\mathcal{L}_X$ in concert to encourage feature separation.

temporal forward-backward and backward-forward predictions form approximately closed cycles. Meanwhile, similar video states are, as per contrastive learning, still mapped to nearby locations. Thus, to generate this space, CEP enforces loss functions that deliver both bi-directional temporal cycle closure and contrastive feature separation. To cope with GPU memory restrictions, we use a memory bank to generate negative features for our contrastive learning. Further, to avoid trivial solutions that are non-semantic-bearing, we introduce Synchronized Temporal-Group Normalization as well as exploit clues from optical flow. The proposed CEP approach is fundamentally different from future-frame prediction works such as DPC [15], TCC [6], and TimeCycle [44]. For example, DPC [15] is a *uni-directional* predictive contrastive model that aggregates temporal embeddings recurrently to make predictions, and in TimeCycle [44], patch features from a sequence of frames, generated by tracking forward and backward to learn the affinity matrix between nearby frames. In contrast, CEP jointly learns high level temporal cycle structure via a closure loss and the prediction feature space itself via a contrastive loss (see Fig. 1).

We note that the fundamental differences between our work and the main body of work on contrastive cycle consistency, such as [13, 15, 20, 44], are that we employ temporal cycle closure as a direct loss objective in a *latent semantic space of video snippets* – thus, CEP is not bound to single frame association, appearance-similarity or tracking tasks. Instead, CEP provides temporally guided, *semantic* self-supervision that encodes entire video segments and their temporal relationships, e.g. to further action recognition.

Our key contributions are: (i) we present a novel temporal cycle-exploiting minimisation objective for self-supervised pretext learning that provides superior generalisation and representational ability in latent space, (ii) we introduce the Cycle Encoding Prediction approach that implements this minimisation objective in combination with contrastive learning and by adding key components, such as a memory bank and ways to avoid trivial solutions, (iii) we quantitatively compare different self-supervised video representation learning architectures, while different design choices are also investigated to consider the contributions of each component, (iv) we achieve competitive or better than the state-of-the-art results (depending on the pretext training dataset, architecture, etc.) performance on standard datasets UCF101 and HMDB51. Our source code is available at https://github.com/youshyee/CEP.

2 Related Work

Spatial Representation Learning – Single image frames already provide a wealth of intrinsic spatial and colour channel relationships which form structure and can be utilised for unsupervised learning of the content-specific structure. Tasks such as colourising grey-scale images [24, 49], image inpainting [33], predicting relative image patch positions [5], or image jigsaw puzzles [21, 32] leverage this information. However, whilst single frames can capture the essence of many high-level semantic concepts, such as actions [36] (and self-supervision may be able to learn this information [42]), their expressiveness is limited with regard to many motion-dependent [29] and fine-grained [28] action recognition tasks.

Spatial-Temporal Representation Learning – Videos contain rich spatio-temporal information that is naturally suitable for unsupervised learning. Many existing works act on unlabelled videos for spatio-temporal representation learning, e.g. [6, 10, 11, 15, 17, 22, 25, 26, 30, 42, 44, 48]. Temporal coherence and dynamics of video were exploited for self-supervised learning early in [42]. Positive samples from the tracker, together with the negatives that were sampled randomly, were deployed in a contrastive learning framework with a pre-designed Siamese-triplet network for self-supervised learning. Alternatively, one may leverage the temporal continuity of video snippets to design the proxy task [25, 30, 47]. Shuffle & Learn [30] discriminates the non-chronologically ordered frames from a video to enforce the learning of temporal relations among frames. Similarly, OPN [25] designs the learning task by reordering the shuffled frames from a video which enforces the learning of temporal continuity. In VCOP [47], by relying on the temporal coherence within snippets from a shuffled video, predicting the order becomes the proxy task for self-supervised learning. Rather than sorting whole frames, ST-puzzle [22] focuses on individual non-overlapped spatial patches to introduce a proxy task to re-order permuted 3D spatio-temporal snippets.

Since the introduction of GANs [12], their application towards constructive video generation and prediction have received a great deal of interest. In [41], video frames are partitioned into several spatial regions by different appearance and motion patterns. Exploiting the statistical relationship within the patterns for a known frame can then be used to derive relationships in unknown frames by a dual-stream network. DPC [15] observes an earlier period of a video with a recurrent network for global context extraction. Then, the network makes a prediction for the context of a later period of the video using the Noise Contrastive Estimation (NCE) loss as the learning constraint. Mem-DPC [16] extends DPC by appending a memory mechanism that queries a similar pattern from memory when making predictions.

More recent works leverage speed information from videos to form a self-supervised learning task, e.g. [1, 34, 48]. PRP [48] considers the semantic similarity and temporal structure difference of the same video at different playback rates, introducing multi-task learning objectives that enforce the model to enrich the detail temporally as well as to regress the pre-defined playback rate. SpeedNet [1] seeks to learn the appearance and motion from unlabelled videos by predicting the playback rate for the clip. RSPNet [34] extends this to a relative speed perception task by applying speed augmentation strategy to make the network consider appearance related content. CVRL [35], as the video extension of SimCLR [4], performs contrastive learning by augmenting video clips spatially and temporally.

Despite substantial progress in self-supervised learning by prediction, existing methods have not yet fully exploited the representational potential of temporal predictions within video. In this paper, we address this problem and describe the learning of video feature spaces where cycle-consistent bi-directional temporal predictions are optimised in tandem with contrastive feature separation.

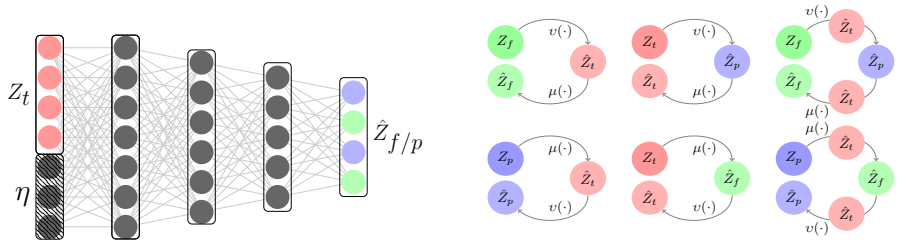

Figure 2: ***(left)* Structure of Predictors μ and υ.** An embedding Z_t concatenates with white noise η as the input. The two predictors are implemented with 4-layers MLP with ReLu activation after each layer. Note that future prediction function and past prediction function share this same architecture, but they do not share parameters in training. ***(right)* Elementary Cycles in Embedding Space.** The current, future, and past states are represented in pink, green, and purple respectively. The depicted 6 most fundamental cycles are considered for cycle consistency. Note that apart from these elementary cycles, the concept could be extended and longer loops are possible, but they require increasing computational resource and a related analysis would be beyond the scope of this paper.

3 Cycle Encoding Prediction

We would like to utilise unlabelled video and its property of temporal cycle closure to pretext learn a latent space for content encoding. To do this, we exploit the bi-directional temporal coherence of video as a self-supervision signal and enforce loss functions that encourage temporal cycle closure, as well as contrastive feature separation.

Framework – CEP operates on video snippets $X_t \in \mathbb{R}^{T\times H\times W\times C}$ harvested from the video stream (as illustrated in Fig. 1), where $T \times H \times W \times C$ are, time, height, width, and channel respectively. Given a particular snippet X_t representing the current state, there are two adjacent neighbouring snippets: the past snippet $X_p = X_{t-1}$, and the future snippet $X_f = X_{t+1}$. We consider snippets being encoded within a latent space via a non-linear network $e(\cdot)$, a function which we would like to learn and which should preserve temporal cycle closure and contrastive feature separation. Formally, $e(\cdot)$ encodes input video snippets X_t as locations Z_t in latent space, *i.e.* $Z_t = e(X_t)$, where the latent space Z_t is a feature of the form $\mathbb{R}^D$.

On the most fundamental level, we note that if the future state $Z_f = e(X_f)$ or past state $Z_p = e(X_p)$ can be predicted by current Z_t, then the latent space representation must have preserved elementary aspects of the temporal structure of input video clips. To implement this concept, we introduce two predictive functions $\mu(\cdot,\eta)$ and $\upsilon(\cdot,\eta)$ to predict future and past states, respectively. We add white noise η in both predictive functions to increase the stochasticity. As exemplified in Fig. 2 *(left)*, $\mu(\cdot)$ can take current state Z_t and past state Z_p, and predict the future by the current and the current by the past, such that

$$\hat{Z}_f = \mu(Z_t,\eta) = \mu(e(X_t),\eta) \quad \text{and} \quad \hat{Z}_t = \mu(Z_p,\eta) = \mu(e(X_p),\eta). \tag{1}$$

Similarly, the elementary function $\upsilon(\cdot)$ allows for the following direct predictions, *i.e.*,

$$\hat{Z}_p = \upsilon(Z_t,\eta) = \upsilon(e(X_t),\eta) \quad \text{and} \quad \hat{Z}_t = \upsilon(Z_f,\eta) = \upsilon(e(X_f),\eta). \tag{2}$$

Note that we clearly differentiate between latent space locations Z_t arising directly from the embedding function and $\hat{Z}_t$ arising from predictions solely within latent space.

Consistency of Predicted Cycles – We observe that the predictive functions can be applied to embeddings recursively, as depicted in Fig. 2 *(right)*. Such application allows for the generation of entire cycles of predictions knowing that the future operator $\mu(\cdot)$ is the reverse of the past operator $\upsilon(\cdot)$, *i.e.* $\mu(\cdot) = \upsilon^{-1}(\cdot)$, or in full detail,

$$Z_t \approx \upsilon^{(n)}\big(\mu^{(n)}(Z_t,\eta),\eta\big) \quad \text{and} \quad Z_t \approx \mu^{(n)}\big(\upsilon^{(n)}(Z_t,\eta),\eta\big)\,, \tag{3}$$

where $\mu^{(n)}(\cdot)$ denotes the future predictive function $\mu(\cdot)$ is applied n times.

In contrast to building a loss function directly by measuring the success of predictions (which would require comparison with at least another transfer $e(\cdot)$ into the embedding space), we will instead construct a loss solely *within* the embedding space that encourages closure of cycles. Conceptually, this allows for far greater representational flexibility and generalisation in the latent space whilst still enforcing high-level temporal consistency.

Fig. 2 *(right)* depicts all six basic possible predictive cycles amongst current, future, and past embeddings. Thus, our target is to find a space wherein cycle predictions can be made such as to minimise the distance between the start and end of all closed cycles. Thus, we arrive at the CEP minimisation objective

$$\begin{aligned}\mathcal{L}_C =& ||Z_t - \upsilon(\mu(Z_t,\eta),\eta)||_2 + ||Z_t - \mu(\upsilon(Z_t,\eta),\eta)||_2 \\ &+ \sum_{n=1}^{2} ||Z_p - \upsilon^{(n)}(\mu^{(n)}(Z_p,\eta),\eta)||_2 + \sum_{n=1}^{2} ||Z_f - \mu^{(n)}(\upsilon^{(n)}(Z_f,\eta),\eta)||_2 \,.\end{aligned} \tag{4}$$

Essentially, this loss sums over the distance errors occurring across the six different basic cycles. Based on this loss, the set of parameters θ, that is the union over θ_e for the encoder network together with θ_μ and θ_υ, are updated by optimising $\arg\min_\theta \mathcal{L}_C$. To compute the parameters of this in practice, the feature encoder and the predictors are co-optimised and evolve together. This process essentially computes an embedding function $e(\cdot)$ that produces a space where closed cycle prediction is maximally achieved. In tandem with this process, both $\mu(\cdot)$ and $\upsilon(\cdot)$ are constructed as actual implementations of cycle prediction.

Contrastive Loss – In addition to encouraging cycle consistency, we want the elements Z_t of the latent space be organised in such a way that distance separates unrelated features, whilst keeping related ones close. In order to implement this, we introduce a second loss that enforces contrastive feature separation after performing our predictive tasks. For this, we adopt InfoNCE [14]. In the forward pass, the ground truth representation Z_t and the predicted representation $\hat{Z}_t$ are computed as shown in Fig. 1. Z_t is then treated as the anchor, whilst any $\hat{Z}_t$ is treated as a positive sample. Negative samples $\mathcal{I}$ are taken from mini-batch and memory bank. InfoNCE demands the similarity between the ground truth and positive to be higher than the one between ground truth and negatives. If the similarity is measured by a bi-linear function, a normalised probability P_{pos} can be represented as

$$P_{pos} = \frac{\exp(S_p/\xi)}{\exp(S_p/\xi) + \sum_{i^* \in \mathcal{I}} \exp(S_{i^*}/\xi)} \,, \tag{5}$$

where $S_p \in \mathbb{R}^1$ denotes the similarity result between the positive pair after the bi-linear operation $S_p = Z_t \cdot \hat{Z}_t^T$, S_{i^*} represents the results for negative pairs, respectively, and ξ is the temperature hyper-parameter[45]. Based on this normalised positive pair distribution P_{pos}, given the distribution $\mathcal{T}$, the optimisation task for InfoNCE can be stated as

$$\arg\min_\theta \mathcal{L}_X = -\mathbb{E}_{p \in \mathcal{T}} \left(\log P_{pos} \right). \tag{6}$$

Memory Bank – Contrastive learning benefits from large mini-batch sizes [19], but given limited GPU memory, it is hard to accommodate both model size and batch size within hardware constraints. Inspired by [19], we increase the cardinality of possible negative samples $\mathcal{I}$ by giving up back-propagation of the gradient for negatives. Instead, we use a proxy encoder $e'(\cdot)$ to generate negative features. The encoder $e'(\cdot)$ is updated based on the encoder $e(\cdot)$, but with momentum, $\theta_{e'} v \leftarrow m\theta_{e'} + (1-m)\theta_e$, given $\theta_e \leftarrow \theta_e + \lambda \nabla \mathcal{L}$, where λ is the learning rate for the encoder $e(\cdot)$ and $m \in [0,1)$ is the momentum coefficient.

As depicted in Fig. 3 *(left)*, $\mathcal{I}$ samples are picked from memory queue $\mathcal{Q}$ at each iteration. Negative features are then generated by proxy encoder $e'(\cdot)$, i.e. $Z_{i^*} = e'(X_{i^*}) \mid i^* \in \mathcal{I} \sim \mathcal{Q}$.

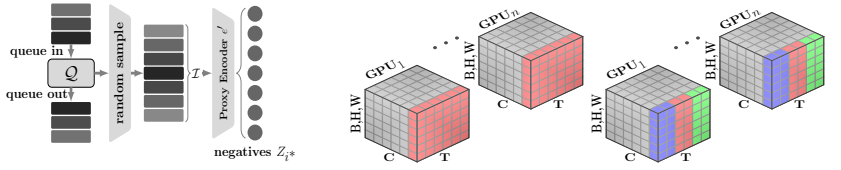

Figure 3: ***(left)* Memory Bank.** During each training iteration, $\mathcal{I}$ samples are picked from a memory queue $\mathcal{Q}$ to stand-in as source material for negative generation. Following that, the negative features are generated by proxy encoder $e'(\cdot)$.***(middle+right)* Sync-BN vs. Sync-TGN.** In contrast to Sync-BN shown in the middle, Sync-TGN shown on the right organises the temporal dimension into several chunks (purple, pink and green) representing past, present, future snippets in CEP and computes mean and stardard-deviation along the (**B**atch,**H**eight,**W**idth) axes.

Batch Normalization Leaks. Our encoder $e(.)$ contains Synchronized Batch Normalization (Sync-BN) layers. However, intra-temporal communication (*e.g.* Sync-BN) among frame snippets may leak information, such that predictors can learn shortcut solutions. To resolve this, we introduce Synchronized Temporal-Group Normalization (Sync-TGN) as shown in Fig. 3 *(right)* and ablated in Section 6 to dilute any leaking signal.

Identity Solution. We note that, theoretically, the identity for both μ and υ leads to a valid solution for $\mathcal{L}_C$ (see Eq. 4). Yet, practically we find that both the use of white noise η (see Fig. 2 *(left)*) and co-optimisation by $\mathcal{L}_X$ disperse mappings in latent space and avoid collapse to an identity cycle. The latter was never observed in any of our experiments.

Optical Flow Obfuscation. Another possible trivial solution of CEP is the generation of temporal predictions purely from low-level optical flow rather than semantic information. As a counter-measure, we obfuscate optical flow by applying an augmentation strategy to the video snippets, *e.g.* random horizontal flipping, colour jittering, and random cropping. Note that this preserves semantic temporal coherence. Our direct ablation in Section 6 shows the effectiveness of this approach.

4 Experiments

We now detail the datasets and environments used for learning, and evaluate the effect of CEP pre-training on the downstream task of action recognition. We provide detailed ablation studies to support our design decisions.

Datasets – For the initial pretraining dataset, **Kinetics** [3] is used in its unlabelled form. It contains 400 action classes, with at least 400 clips per action. Each video lasts around 10s and is taken from YouTube. The actions are human-focused and range from human-object interactions, such as playing instruments, to human-human interactions, such as shaking hands. **UCF101** [37] is selected as the primary downstream action recognition task. It contains 13K YouTube videos of different action categories and has a large variety of different camera motions, object appearances, poses and scales, as well as background clutter. The popular **HMDB51** [23] is our secondary evaluation dataset with 51 action categories covering a wide range of activities from facial actions to body movements. HMDB51 is often considered more difficult than UCF101 as many classes within it are quite similar.

Architecture – The non-linear encoding function $e(\cdot)$ is implemented using popular spatio-temporal representation backbones (with modified last FC layer yielding an output feature size of 2048, *e.g.* SlowFast [9], (2+1)D-ResNet [39], 3D-ResNet [18], and S3D-G [46], allowing us to both compare directly with other methods and to compare these architectures. The two predictors υ and μ are implemented with 4-layers MLP (see Fig. 2).

Method	Year	Self-Supervision Training			Evaluation	
		Dataset	Resolution	Architecture	UCF101	HMDB51
01 | Random Init.		–	128×128	SlowFast	56.2	23.1
02 | CEP		Kinetics-400	128×128	SlowFast	**68.5**	**34.7**
03 | VCOP [40]	2019	UCF101	224×224	(2+1)D-ResNet	72.4	30.9
04 | PRP† [48]	2020	UCF101	224×224	(2+1)D-ResNet	72.7	35.9
05 | CEP		UCF101	224×224	(2+1)D-ResNet	**75.5**	36.3
06 | PRP + CEP		UCF101	224×224	(2+1)D-ResNet	73.8	**37.1**
07 | DPC [15]	2019	Kinetics-400	224×224	3D-ResNet34	75.7	35.7
08 | Mem-DPC [16]	2020	Kinetics-400	224×224	3D-ResNet34	**78.1**	**41.2**
09 | CEP		Kinetics-400	224×224	3D-ResNet34	76.4	36.5
10 | DPC+CEP		Kinetics-400	224×224	3D-ResNet34	**78.1**	38.4
11 | 3D-Puzzle [24]	2019	Kinetics-400	224×224	3D-ResNet18	65.8	33.7
12 | RSPNet [5]	2021	Kinetics-400	224×224	3D-ResNet18	74.3	**41.8**
13 | CEP		Kinetics-400	224×224	3D-ResNet18	**75.9**	36.6
14 | RSPNet [5]	2021	Kinetics-400	224×224	(2+1)D-ResNet	**81.1**	**44.6**
15 | TCGL [27]	2021	Kinetics-400	224×224	(2+1)D-ResNet	77.6	39.7
16 | CEP		Kinetics-400	224×224	(2+1)D-ResNet	76.7	37.6
17 | CoCLR(RGB) [17]	2020	Kinetics-400	128×128	S3D-G	87.9	54.6
18 | SpeedNet [1]	2020	Kinetics-400	224×224	S3D-G	81.1	48.8
19 | RSPNet† [5]	2021	Kinetics-400	224×224	S3D-G	88.3	59.0
20 | CEP		Kinetics-400	224×224	S3D-G	84.1	46.3
21 | RSPNet + CEP		Kinetics-400	224×224	S3D-G	**90.1**	**59.5**
22 | CVRL [35]	2020	Kinetics-400	224×224	R3D-50($\times$4)	92.2	67.9
23 | ρBYOL [11]	2021	Kinetics-400	224×224	R3D-50($\times$4)	**95.5**	**73.6**

Table 1: **Performance Comparison with State-of-the-Art.** The top-1 accuracy rate is reported on downstream action recognition tasks for UCF101 and HMDB51 in RGB data in split-1 val split. All methods are end-to-end, fine-tuned on the target evaluation set after self-supervised pre-training. † represents our re-implementation experiments under our directly comparable hardware settings.

Pre-processing – Video snippets X_t of 16 frames each are formed by sampling the original 30fps video at a rate of 6 fps. Frames are selected with a consistent stride to preserve the regularity of temporal dynamics. We sample 3 snippets without overlap to form associated triples. For augmentation, we apply snippet-wise random horizontal flip and random colour jittering. Finally, snippets are rescaled and centre-cropped to 224×224 resolution.

Training – We implemented CEP in Pytorch using distributed training on 8 GPUs, each with a mini-batch size of 3 with input tensor sizes being $\mathbb{R}^{3\times3\times16\times224\times224}$. To produce predictor extensions, we use Gaussian noise with $\eta \in \mathbb{R}^{1024}$. Similar to [31], we bootstrap the groundtruth representation Z_t from encoder $e(.)$ with its weights shifted to those from 5000 iterations before, resulting in faster convergence.

For cycle consistency, we run all 6 possible cycles (see Fig. 2 *(right)*) and use the overall loss $\mathcal{L}_\mathcal{C}$ for back-propagation. For contrastive learning, the negative features are sampled from the memory bank with the proxy encoder $e'(\cdot)$ using the momentum coefficient $m = 0.9$ and InfoNCE temperature $\xi = 1$. We sample 16 easy negative features from the memory bank, and use 4 hard negative features in the mini-batch where easy negatives are from different videos, whilst the hard negatives are from the same video. The ratio between easy and hard samples is 4:1 so as to make the learning task neither too hard to learn nor too easy to overfit. All the models are trained end-to-end using SGD as optimiser with an initial learning rate of 10^{-2}, momentum 0.9 and weight decay 10^{-4}. The system requires to minimise the combined objective $\mathcal{L} = \lambda\mathcal{L}_C + \mathcal{L}_X$, where λ is set to 0.1 intuitively. During inference, video snippets from the validation set are sampled using the same strategy mentioned above, but all the augmentations are removed maintaining a scaled centre-crop.

Compute Requirements – Quantifying CEP's computational footprint with different

backbone architectures shows the approach has a very light parameter impact yet significant FLOPS cost. For the S3D-G backbone, the CEP concept increases parameters from 11.04M to 13.16M and FLOPS from 34.3G to 85.6G. For the R(2+1)D backbone, CEP increases parameters from 15.02M to 17.15M and FLOPS from 171.8G to 429.5G. For 3D-ResNet-34, CEP increases parameters from 64.17M to 66.29M and FLOPS from 100.58G to 261.06G. For example, using 3D-ResNet18 for CEP training (see Table 1, row 13) takes approx. 1 week for 50 epochs with 8 NVIDIA P100 GPUs running on Lenovo nx360 nodes with 2.4 GHz Intel E5-2680 v4 (Broadwell) CPUs and 128 GB of RAM using the Kinetics-400 dataset as data source with batch size 24.

5 Results

We compare CEP's performance by looking at different state-of-the-art self-supervised methods, fine-tuning on the action classification tasks of UCF101 and HMDB51. Our results are reported based on split-1 validation accuracy.

Baselines – At the lower resolution of 128×128 (rows 01–02 in Table 1), we note the positive effect of CEP pre-training against a random initialisation with a significant boost of 12.3% on UCF101 and 11.6% on HMDB51. Next, training only on the UCF101 dataset at 224×224 resolution and on a (2+1)D-ResNet (*i.e.* rows 03–05), our CEP approach outperforms other comparable methods, such as VCOP [47] and PRP [48], reaching 75.5% and 36.3% accuracy for UCF101 and HMDB51, respectively. Methods 07–10 show a direct comparison with DPC [15]. Given the same 3D-ResNet34 backbone, CEP exceeds DPC [15] at 76.4% and 36.5% on UCF101 and HMDB51, respectively. When we integrate the CEP concept into DPC (to get DPC+CEP), a further improvement leading to 78.1% and 38.4% can be observed on those two datasets, respectively. UCF101 MemDPC [16] results are on-par with DPC+CEP which uses fewer parameters and 10% fewer FLOPS. On rows 11–20 in Table 1, depending on the backbone network, CEP exhibits a mixed performance compared to other recent networks. In particular, with a 3DResNet18 backbone CEP outperforms 3D-Puzzle [22] by a significant margin on both datasets, while also outperforming RSPNet [54] on UCF101. On the (2+1)D-ResNet backbone, CEP on its own achieves competitive performance in comparison to TCGL [27]. RSPNet on the other hand outperforms both CEP and TCGL in this category. When fuelled by a S3D-G backbone, RSPNet dominates CEP, yet CEP exceeds Google's SpeedNet [1] on UCF101, while being competitive on HMDB51.

Add-on Efficacy of CEP – However, we find that adding CEP to leading architectures (rows 06, 10, 21) while keeping their main concepts unchanged consistently outperforms their original benchmarks supporting general efficacy of our concept. Specifically, to use CEP as an add-on we introduce forward and backward prediction modules on top of feature embeddings of other architectures. We then contrastively learn the predictive embeddings and enforce cycle consistency by introducing our $\mathcal{L}_X$ and $\mathcal{L}_C$ losses, respectively, to the overall loss. For PRP with a (2+1)D-ResNet backbone, integrating the CEP concept can introduce a 1.1% and 1.2% performance boost for UCF101 and HMDB51, respectively (*i.e.* rows 04 vs. 06). For DPC with a 3D-ResNet34 backbone, DPC+CEP increases the baseline by 2.4% and 2.7%, respectively (*i.e.* rows 07 vs. 10). The state-of-the-art RSPNet can also benefit from CEP with an improved benchmark for RSPNet+CEP of 90.1% on UCF101 and 59.5% on HMDB51 (*i.e.* rows 19 vs. 21). For the architectures tested so far this produces a leading performance (rows 1–21). We conclude that CEP is consistently and demonstrably effective as an add-on technique across the architectures tested.

Massively Large Setups – CVRL [35] on row 22 and ρBYOL [10] on row 23 show

(a)

Pre-Training Loss Functions	Eval. (UCF) Top1 acc
random init.	56.2
$\mathcal{L}_C$ only, (μ *and* υ)	61.1
$\mathcal{L}_X$ only, ($\mu = \upsilon$)	63.7
$\mathcal{L}_X$ only, (μ *and* υ)	64.6
$\mathcal{L}_X$+$\mathcal{L}_C$, (μ *and* υ)	**68.5**

(b)

Optical Flow Obfuscation	Eval. (UCF) Top1 Acc
✗	70.3
✓	**75.4**

(c)

Memory Bank Strategies M-Bank	Strategy	m	Eval. (UCF) Top1 Acc
✗	-	-	71.2
✓	static	-	72.8
✓	dynamic	0.1	73.3
✓	dynamic	0.9	**75.4**

(d)

Normalisation Strategies	Eval. (UCF) Top1 Acc
Sync-BN	71.8
Sync-TGN	**75.4**

(e)

Temporal Receptive Field	Eval. (UCF) Top1 acc
1.5s	73.7
3s	**75.4**

(f)

Feature Dimensionality temp. pres.	output size	Eval. (UCF) Top1 Acc
✓	2×2048	73.3
✗	2048	71.2
✗	4096	**75.4**

Table 2: **Detailed Ablation Studies.** Verification of key component effectiveness via ablation. All studies pre-train on Kinetics-400 with a SlowFast Resnet50 encoder and evaluate performance on UCF101. All experiments apart from *(a)* use a resolution of 224×224 for experiments. Note that all design choices demonstrably impact positively on performance. In particular, as shown in ***(a)***, the use of independent forward and backward predictors μ *and* υ has value and improves performance, as does the addition of each of the proposed core losses $\mathcal{L}_X$ and $\mathcal{L}_C$. For instance, adding cycle consistency $\mathcal{L}_C$ to an otherwise fixed setup is effective adding 3.9% to accuracy.

impressive performance on UCF101 and HMDB51. Yet, these works are not directly apple-to-apple comparable since setups run at massively larger scale. Particularly, huge batch sizes (CVRL: 1024 vs. CEP: 24), very large backbone networks (CVRL: 31.7M vs. rhoBYOL: 31.8M vs. CEP: 13.06M parameters) and more than an order of magnitude further pre-train epochs (CVRL: 800 vs. rhoBYOL: 800 vs. CEP: 50 epochs) are used. We are unable to offer direct comparisons to these massive experiments as we are limited by computational resources. Nevertheless, our results showed that the core CEP concept effectively promotes semantic learning, is valuable and widely applicable as an add-on, where it enhanced the performance of all X+CEP architectures tested.

6 Ablation Experiments

In this section, we present a detailed ablation study in order to quantify the impact of the CEP concept and validate the effectiveness of the components in the basic setup. In particular, we will show that cycle consistency $\mathcal{L}_C$ improves learning as it encourages an inverse relationship between otherwise independent forward and backward predictors $\mu(\cdot)$ and $\upsilon(\cdot)$.

Efficacy of both Contrastive and Cycle Loss – For experiments that quantify the importance of contrastive loss $\mathcal{L}_X$ and cycle loss $\mathcal{L}_C$ during training, we use a fixed SlowFast setup with a ResNet50 as the encoder. The input is rescaled and randomly cropped to a resolution of 128×128. The models are pre-trained on the Kinetics-400 dataset as before, and results are reported after fine-tuning on UCF101 dataset for 30 epochs. As shown in Table 2 *(a)*, both $\mathcal{L}_X$ and $\mathcal{L}_C$ contribute significantly to CEP's performance, which drops considerably when removing any one of the losses. The addition of cycle consistency to an otherwise fixed system improves accuracy significantly by 3.9%.

Consistency of $\mathcal{L}_C$ Efficacy – To show coherent efficacy of the cycle consistency concept throughout the training process, we compare the full CEP setup against one that does not use $\mathcal{L}_C$ at different points of the pre-training process – the same comparison as shown after full pre-training in the last two lines of Table 2 *(a)*. Table 3 shows the results confirming that throughout pre-training the use of $\mathcal{L}_C$ is beneficial and able to enhance action recognition.

Obfuscation of Inter-clip Optical Flow – We will now ablate further components of the basic setup beyond the core concepts. As discussed earlier and quantified in Table 2 *(b)*, the augmentation-based obfuscation of inter-clip optical flow does indeed improve learning and boosts accuracy significantly by 5.1% when tested at 224×224 resolution.

Pre-Training Loss Functions	179K-step	358K-step	537k-step
No Cycle Consistency used: $\mathcal{L}_X$ only	61.0	64.2	64.6
Cycle Consistency used: $\mathcal{L}_X+\mathcal{L}_C$	**63.1**	**64.8**	**68.5**

Table 3: **Consistency of $\mathcal{L}_C$ Efficacy throughout Pre-Training.** UCF101 Top 1 accuracy performance when pre-training for different numbers of steps with and without cycle consistency $\mathcal{L}_C$, fixing all other settings. The results confirm that the use of cycle consistency consistently benefits performance.

Memory Bank Strategies – As shown in Table 2 *(c)*, when tested again at 224×224 resolution learning benefits from a large batch size as demonstrated recently for other contrastive learning in He *et al*. [19]. Feature memory banks can provide this property at controllable cost. We compare two memory bank strategies: (i) static negative features are sampled from the memory bank, and (ii) negatives are generated by a proxy encoder $e'(\cdot)$ dynamically. These models take 4 negatives from the mini-batch and sample 16 negatives from the memory bank per step. We compare these settings against a model without a memory bank that just takes 4 negatives per step from the mini-batch. We conclude that a dynamic memory bank with a large momentum factor m can clearly provide benefits to CEP learning.

Synchronized Temporal-Group Normalization – Next, we explore the effect of our Sync-TGN in preventing the CEP spiralling into a trivial solution. For the BN experiment, we use the same batch size (3 per GPU) for both sync-BN and sync-TGN, and we have 8 GPUs in both scenarios. Table 2 *(d)* depicts the effectiveness of Sync-TGN over Sync-BN showing an increase of 3.6% in self-supervised learning accuracy, supporting our hypothesis that information leak can be mitigated to some extent.

Temporal Receptive Field – To investigate the effect of the temporal receptive field (RF), we sparsely sample 16 frames as the input. These 16 frames cover either 1.5s or 3s of the input video. As Table 2 *(e)* shows, the input clip with the larger temporal RF increases the learning performance by 1.7%, which suggests that the model benefits from longer exposure to scene dynamics for learning of semantics.

Feature Space Dimensionality – Finally, Table 2 *(f)* summarises our evaluations on the influence of changing the dimensionality of the embedding space. Unsurprisingly, a larger representational space benefits performance. Note that for noise η, we use half of the embedding's feature size, *e.g.* if $Z_t \in \mathbb{R}^{2048}$ then $\eta \in \mathbb{R}^{1024}$. Splitting the feature space artificially into sub-spaces to preserve temporal separation was found to be ineffective.

7 Conclusion

In this paper, we showed that action classification can benefit from learning feature spaces of video snippets in which temporal cycles are maximally predictable. We introduced Cycle Encoding Prediction which can effectively encode video in this latent space using the concepts of closed temporal cycles and contrastive feature separation. CEP can be seen as a temporal regularisation approach used to guide the construction of latent semantic video spaces. It unlocks the related self-supervision signal and is demonstrably effective when used for pretext learning. Ablation studies support the effectiveness of the core loss functions and system components. We reported results across different backbones for the standard datasets UCF101 and HMDB51.

We showed that the CEP concept introduces an effective and elegant self-supervision criterion for video that can serve as an additional guide for pretext training. As an add-on concept it was shown to enhance the performance of many existing architectures.

References

[1] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos. In CVPR, 2020.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. In TPAMI, pages 1798–1828, 2013.

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In CVPR, pages 6299–6308, 2017.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In ICML, pages 1597–1607. PMLR, 2020.

[5] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In ICCV, pages 1422–1430, 2015.

[6] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In CVPR, pages 1801–1810, 2019.

[7] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In CVPR, pages 203–213, 2020.

[8] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In CVPR, pages 1933–1941, 2016.

[9] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In ICCV, pages 6202–6211, 2019.

[10] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In CVPR, pages 3299–3309, 2021.

[11] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In CVPR, pages 3636–3645, 2017.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, NeurIPS, pages 2672–2680. Curran Associates, Inc., 2014.

[13] Daniel Gordon, Kiana Ehsani, Dieter Fox, and Ali Farhadi. Watching the world go by: Representation learning from unlabeled videos. arXiv preprint arXiv:2003.07990, 2020.

[14] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pages 297–304, 2010.

[15] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In ICCVW, 2019.

[16] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In ECCV, 2020.

[17] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. NeurIPS, 33:5679–5690, 2020.

[18] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In ICCVW, pages 3154–3160, 2017.

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In CVPR, pages 9729–9738, 2020.

[20] Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. In NeurIPS, 2020.

[21] Dahun Kim, Donghyeon Cho, Donggeun Yoo, and In So Kweon. Learning image representations by completing damaged jigsaw puzzles. In WACV, pages 793–802. IEEE, 2018.

[22] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In AAAI, volume 33, pages 8545–8552, 2019.

[23] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In ICCV, 2011.

[24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In CVPR, pages 6874–6883, 2017.

[25] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In ICCV, pages 667–676, 2017.

[26] Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-task self-supervised learning for temporal correspondence. In NeurIPS, 2019.

[27] Yang Liu, Keze Wang, Haoyuan Lan, and Liang Lin. Temporal contrastive graph for self-supervised video representation learning. arXiv preprint arXiv:2101.00820, 2021.

[28] Khoi-Nguyen C Mac, Dhiraj Joshi, Raymond A Yeh, Jinjun Xiong, Rogerio S Feris, and Minh N Do. Learning motion in feature space: Locally-consistent deformable convolution networks for fine-grained action detection. In ICCV, pages 6282–6291, 2019.

[29] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In CVPR, pages 2891–2900, 2017.

[30] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In ECCV, pages 527–544. Springer, 2016.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518 (7540):529–533, 2015.

[32] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In ECCV, pages 69–84. Springer, 2016.

[33] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In CVPR, pages 2536–2544, 2016.

[34] Chen Peihao, He Deng, Huang Dongliang, Long Xiang, Zeng Runhao, Wen Shilei, Tan Mingkui, and Chuang Gan. Rspnet: Relative speed perception for unsupervised video representation learning. In AAAI, 2021.

[35] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. In CVPR, pages 6964–6974, 2021.

[36] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In NeurIPS, pages 568–576, 2014.

[37] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402, 2012.

[38] Dídac Surís, Ruoshi Liu, and Carl Vondrick. Learning the predictability of the future. In CVPR, pages 12607–12617, 2021.

[39] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In CVPR, pages 6450–6459, 2018.

[40] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In TPAMI, pages 98–106, 2016.

[41] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Yunhui Liu, and Wei Liu. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In CVPR, pages 4006–4015, 2019.

[42] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In ICCV, pages 2794–2802, 2015.

[43] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In CVPR, pages 7794–7803, 2018.

[44] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In CVPR, pages 2566–2576, 2019.

[45] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In CVPR, pages 3733–3742, 2018.

[46] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In ECCV, pages 305–321, 2018.

[47] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In CVPR, 2019.

[48] Yuan Yao, Chang Liu, Dezhao Luo, Yu Zhou, and Qixiang Ye. Video playback rate perception for self-supervised spatio-temporal representation learning. In CVPR, pages 6548–6557, 2020.

[49] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In ECCV, pages 649–666. Springer, 2016.