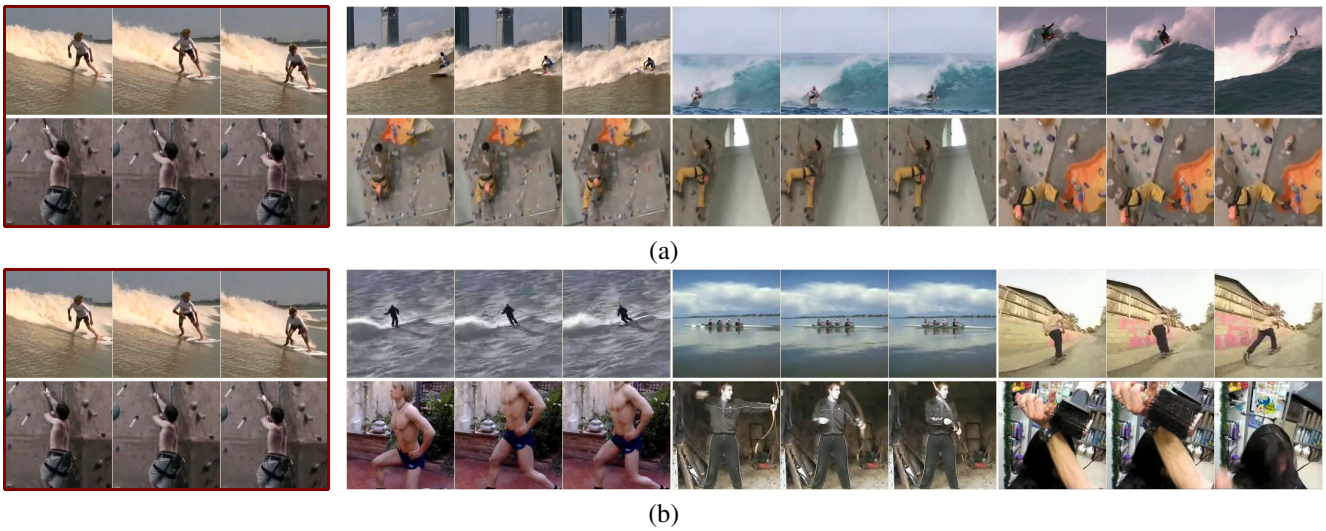
Video Representation Learning by Dense Predictive Coding

Tengda Han Weidi Xie Andrew Zisserman
Visual Geometry Group, Department of Engineering Science, University of Oxford
{htd, weidi, az}@robots.ox.ac.uk



(a)

(b)

Figure 1: Nearest Neighbour (NN) video clip retrieval on UCF101. Each row contains four video clips, a query clip and the top three retrievals using clip embeddings. To get the embedding, each video is passed to a 3D-ResNet18, average pooled to a single vector, and cosine similarity is used for retrieval. (a) Embeddings obtained by Dense Predictive Coding (DPC); (b) Embeddings obtained by using the inflated ImageNet pretrained weights. The DPC captures the semantics of the human action, rather than the scene appearance or layout as captured by the ImageNet trained embeddings. In the DPC retrievals the actual appearances of frames can vary dramatically, *e.g.* in the change in camera viewpoint for the climbing case.

Abstract

The objective of this paper is self-supervised learning of spatio-temporal embeddings from video, suitable for human action recognition.

We make three contributions: First, we introduce the Dense Predictive Coding (DPC) framework for self-supervised representation learning on videos. This learns a dense encoding of spatio-temporal blocks by recurrently predicting future representations; Second, we propose a curriculum training scheme to predict further into the future with progressively less temporal context. This encourages the model to only encode slowly varying spatial-temporal signals, therefore leading to semantic representations; Third, we evaluate the approach by first training the DPC model on the Kinetics-400 dataset with self-supervised learning, and then finetuning the representation on a downstream task, i.e. *action recognition. With single stream (RGB only), DPC pretrained representations achieve state-of-the-art self-supervised performance on both UCF101 (*75.7% *top1 acc) and HMDB51 (*35.7% *top1 acc), outperforming all previous learning methods by a significant margin, and approaching the performance of a baseline pre-trained on ImageNet. The code is available at* https://github.com/TengdaHan/DPC.

1. Introduction

Videos are very appealing as a data source for self-supervision: there is almost an infinite supply available (from Youtube etc.); image level proxy losses can be used at the frame level; and, there are plenty of additional proxy losses that can be employed from the temporal information. One of the most natural, and consequently one of the first video proxy losses, is to predict future frames in the videos based on frames in the past. This has ample scope for exploration by varying the extent of the past knowledge (the temporal aggregation window used for the prediction) and also the temporal distance into the future for the predicted frames. However, future frame prediction does have a serious disadvantage – that the future is not deterministic – so methods may have to consider multiple hypotheses with multiple instance losses, or other distributions and losses over their predictions.

Previous approaches to future frame prediction in

video [23, 24, 36, 42, 41] can roughly be divided into two types: those that predict a reconstruction of the actual frames [23, 24, 36, 42]; and those that only predict the latent representation (the embedding) of the frames [41]. If our goal of self-supervision is only to learn a representation that allows generalization for downstream discriminative tasks, *e.g*. action recognition in video, then it may not be necessary to waste model capacity on resolving the stochasticity of frame appearance in detail, *e.g*. appearance changes due to shadows, illumination changes, camera motion, etc. Approaches that only predict the frame embedding, such as Vondrick *et al*. [41], avoid this potentially unnecessary task of detailed reconstruction, and use a mixture model to resolve the uncertainty in future prediction. Although not applied to videos (but rather to speech signals and images), the Contrastive Predictive Coding (CPC) model of Oord *et al*. [40] also learns embeddings, in their case by using a multi-way classification over temporal audio frames (or image patches), rather than the regression loss of [41].

In this paper we propose a new idea for learning spatio-temporal video embeddings, that we term "Dense Predictive Coding" (DPC). The model is designed to predict the future representations based on the recent past [47]. It is inspired by the CPC [40] framework, and more generally by previous research on learning word embeddings [25, 26, 28]. DPC is also trained by using a variant of noise contrastive estimation [9], therefore, in practice, the model has never been optimized to predict the exact future, it is only asked to solve a multiple choice question, *i.e*. pick the correct future states from lots of distractors. In order to succeed in this task, the model only needs to learn the shared semantics of the multiple possible future states, and this common/shared representation is the kind of invariance required in many of the vision tasks, *e.g*. action recognition in videos. In other words, the optimization objective will actually benefit from the fact that the future is not deterministic, and map the representation of all possible future states to a space that their embeddings are close. Concurrent work [2] applies similar method on reinforcement learning.

The contributions of this paper are three-fold: First, we introduce Dense Predictive Coding (DPC) framework for self-supervised representation learning on videos, we task the model to predict the future embedding of the spatio-temporal blocks recurrently (as used in N-gram prediction). The model is trained to pick the "correct" future states from a pool of distractors, therefore treated as a multi-way classification problem. Second, we propose a curriculum training scheme that enables the model to gradually predict further in the future (up to 2 seconds) with progressively less temporal context, leading more challenging training samples, and preventing the model from using shortcuts such as optical flow; Third, we evaluate the approach by first training the DPC model on the Kinetics-400 [16] dataset using self-supervised learning, and then fine-tuning on action recognition benchmarks. Our DPC model achieves state-of-the-art self-supervised performance on both UCF101 (75.7% top1 acc) and HMDB51 (35.7% top1 acc), outperforming all previous single-stream (RGB only) self-supervised learning methods by a significant margin.

2. Related Work

Self-supervised learning from images. In recent years, methods for self-supervised learning on images have achieved an impressive performance in learning high-level image representations. Inspired by the variants of Word2vec [3, 25, 26] that rely on predicting words from their context, Doersch *et al*. [5] proposed the pretext task of predicting the relative location of image patches. This work spawned a line of work in context-based self-supervised visual representation learning methods, *e.g*. in [29]. In contrast to the context-based idea, another set of pretext tasks include carefully designed image-level classification, such as rotation [8] or pseudo-labels from clustering [4]. Another class of pre-text tasks is for dense predictions, *e.g*. image inpainting [32], image colorization [48], and motion segmentation prediction [31]. Other methods instead enforce structural constraints on the representation space [30].

Self-supervised learning from videos. Other than the predictive tasks reviewed in the introduction, another class of proxy tasks is based on temporal sequence ordering of the frames [27, 7, 46]. [12, 14, 44] use the temporal coherence as a proxy loss. Other approaches use egomotion [1, 13] to enforce equivariance in feature space [13]. In contrast, [15] predicts the transformation applied to a spatio-temporal block. In [17], the authors propose to use a 3D puzzle as the proxy loss. Recently [43, 21, 45], leveraged the natural temporal coherency of color in videos, to train a network for tracking and correspondence related tasks.

Action recognition with two-stream architectures. Recently, the two-stream architecture [33] has been a foundation for many competitive methods. The authors show that optical flow is a powerful representation that improves action recognition dramatically. Other modalities like audio signal can also benefits visual representation learning [19]. While in this paper, we deliberately avoid using any information from optical flow or audio, and aim to probe the upperbound of self-supervised learning with *only* RGB streams. We leave it as a future work to explore how much boost optical flow branch and audio branch can bring to our self-supervised learning architecture.

3. Dense Predictive Coding (DPC)

In this section, we describe the learning framework, details of the architecture, and the curriculum training that gradually learns to predict further into the future with progressively less temporal context.

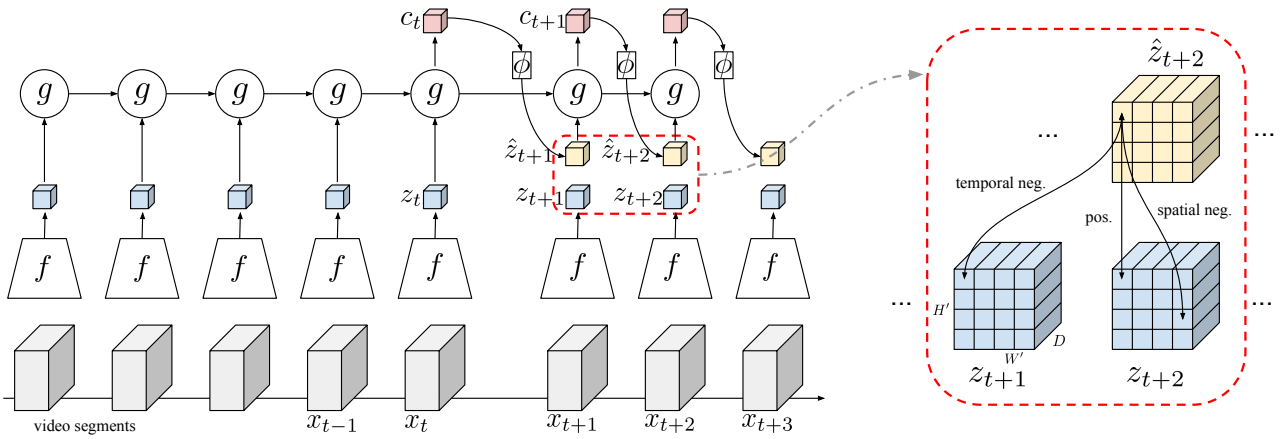


Figure 2: A diagram of **Dense Predictive Coding** method. The left part is the pipeline of the DPC, which is explained in Sec. 3.1. The right part (in the dashed rectangle) is an illustration of the Pred-GT pair construction for contrastive loss, which is explained in Sec. 3.2.

3.1. Learning Framework

The goal of DPC is to predict a slowly varying semantic representation based on the recent past, *e.g.* we construct a prediction task that observes about 2.5 seconds of the video and predict the embedding for the future 1.5 seconds, as illustrated in Figure 2. A video clip is partitioned into multiple non-overlapping blocks $x_1, x_2, \ldots, x_n$, with each block containing an equal number of frames. First, a non-linear encoder function $f(.)$ maps each input video block x_t to its latent representation z_t, then an aggregation function $g(.)$ temporally aggregates t consecutive latent representations into a context representation c_t:

$$z_t = f(x_t) \tag{1}$$

$$c_t = g(z_1, z_2, ..., z_t) \tag{2}$$

where x_t has dimension $\mathbb{R}^{T\times H\times W\times C}$, and z_t is a feature map with dimension $\mathbb{R}^{1\times H'\times W'\times D}$, organized as time $\times$ height $\times$ width $\times$ channels. [1]

The intuition behind the predictive task is that if one can infer future semantics from c_t, then the context representation c_t and the latent representations $z_1, z_2, ..., z_t$ must have encoded strong semantics of the input video clip. Thus, we introduce a predictive function $\phi(.)$ to predict the future. In detail, $\phi(.)$ takes the context representation as the input and predicts the future clip representation:

$$\hat{z}_{t+1} = \phi(c_t) = \phi\big(g(z_1, z_2, \ldots, z_t)\big) \tag{3}$$

$$\hat{z}_{t+2} = \phi(c_{t+1}) = \phi\big(g(z_1, z_2, \ldots, z_t, \hat{z}_{t+1})\big) \tag{4}$$

where c_t denotes the context representation from time step 1 to t, and $\hat{z}_{t+1}$ denotes the predicted latent representation of the time step $t+1$. In the spirit of Seq2seq [37], representations are predicted in a sequential manner. We predict q steps in the future, at each time step t, the model consumes the previously generated embedding ($\hat{z}_{t-1}$) as input when generating the next ($\hat{z}_t$), further enforcing the prediction to be conditioned on all previous observations and predictions, and therefore encourages an N-gram like video representation.

[1] In our initial experiments, $x_t \in \mathbb{R}^{5\times 128\times 128\times 3}$, $z_t \in \mathbb{R}^{1\times 4\times 4\times 256}$

3.2. Contrastive Loss

Noise Contrastive Estimation (NCE) [9] constructs a binary classification task: a classifier is fed with real samples and noise samples, and the objective is to distinguish them. A variant of NCE [28, 40] classifies one real sample among many noise samples. Similar to [28, 40], we use a loss based on NCE for the predictive task. NCE over feature embeddings encourages the predicted representation $\hat{z}$ to be close to the ground truth representation z, but not so strictly that it has to resolve the low-level stochasticity.

In the forward pass, the ground truth representation z and the predicted representation $\hat{z}$ are computed. The representation for the i-th time step is denoted as z_i and $\hat{z}_i$, which have the same dimensions. Note that, instead of pooling into a feature vector, both z_i and $\hat{z}_i$ are kept as feature maps ($z_i, \hat{z}_i \in \mathbb{R}^{H'\times W'\times D}$), which maintains the spatial layout representation. We denote the feature vector in each spatial location of the feature map as $z_{i,k} \in \mathbb{R}^D$ and $\hat{z}_{i,k} \in \mathbb{R}^D$ where i denotes the temporal index and k is the spatial index $k \in \{(1,1),(1,2),\ldots,(H,W)\}$. The similarity of the predicted and ground-truth pair (Pred-GT pair) is computed by the dot product $\hat{z}_{i,k}^\top z_{j,m}$. The objective is to optimize:

$$\mathcal{L} = -\sum_{i,k}\left[\log\frac{\exp(\hat{z}_{i,k}^\top \cdot z_{i,k})}{\sum_{j,m}\exp(\hat{z}_{i,k}^\top \cdot z_{j,m})}\right] \tag{5}$$

In essense, this is simply a cross-entropy loss (negative log-likelihood) that distinguishes the positive Pred-GT pair out of all other negative pairs. For a predicted feature vector $\hat{z}_{i,k}$, the only positive pair is $(\hat{z}_{i,k}, z_{i,k})$, i.e. the pre-

dicted and ground-truth features at the same time step and same spatial location. All the other pairs $(\hat{z}_{i,k}, z_{j,m})$ where $(i,k) \neq (j,m)$, are negative pairs. The loss encourages the positive pair to have a higher similarity than any negative pairs. If the network is trained in a mini-batch consisting of B video clips and each of the B clips is from distinct video, more negative pairs can be obtained.

To discriminate the different types of negative pairs, given a Pred-GT pair $(\hat{z}_{i,k}, z_{j,m})$, we define the terminology as follows:

Easy negatives: is the Pred-GT pair that is formed from two distinct videos. These pairs are naturally easy because they usually have distinct color distributions and thus predicted feature and ground-truth feature have low similarity.

Spatial negatives: is the Pred-GT pair that is formed from the same video but at a different spatial position in the feature map, *i.e.* $k \neq m$, while i, j can be any index.

Temporal negatives (hard negatives): is the Pred-GT pair that comes from the same video and same spatial position, but from different time steps, *i.e.* $k = m, i \neq j$. They are the hardest pair to classify because their score will be very close to the positive pairs.

Overall, we use a similar idea to the Multi-batch training [38]. If the mini-batch has batch size B, the feature map has spatial dimension $H' \times W'$ and the task is to classify one of q time steps, the number of each classes follows:

$$\begin{aligned} &\text{Pos} : \text{N}_{temporal} : \text{N}_{spatial} : \text{N}_{easy} \\ =&1 : (q-1) : (H'W'-1)q : (B-1)H'W'q \end{aligned}$$

Curriculum learning strategy. A curriculum learning strategy is designed by progressively increasing the number of prediction steps of the model (Sec. 4.1.4). For instance, the training process can start by predicting only 2 steps (about 1 second), *i.e.* only computing $\hat{z}_{t+1}$ and $\hat{z}_{t+2}$, and the Pred-GT pairs are constructed between $\{z_{t+1}, z_{t+2}\}$ and $\{\hat{z}_{t+1}, \hat{z}_{t+2}\}$. After the network has learnt this simple task, it can be trained to predict 3 steps (about 1.5 seconds), *e.g.* computing $\hat{z}_{t+1}$, $\hat{z}_{t+2}$ and $\hat{z}_{t+3}$ and construct Pred-GT pairs accordingly. Importantly, curriculum learning introduces more hard negatives throughout the training process, and forces the model to gradually learn to predict further in the future with progressively less temporal context. Meanwhile, the model is gradually trained to grasp the uncertain nature in its prediction.

3.3. Avoiding Shortcuts and Learning Semantics

Empirical experience in self-supervised learning indicates that if the proxy task is well-designed and requires semantic understanding, a more difficult learning task usually leads to a better-quality representation [22]. However, ConvNets are notoriously known for learning shortcuts for tackling tasks [5, 29, 46]. In our training, we employ a number of mechanisms to avoid potential shortcuts, as detailed next.

Disrupting optical flow. A trivial solution of our predictive task is that $f(.)$, $g(.)$ and $\phi(.)$ together learn to capture low-level optical flow information and perform feature extrapolation as the prediction. To force the model to learn high-level semantics, a critical operation is frame-wise augmentation, *i.e.* random augmentation for each individual frame in the video blocks, such as frame-wise color jittering including random brightness, contrast, saturation, hue and random greyscale during training. Furthermore, the curriculum of predicting further into the future, *i.e.* predicting the semantics for the next a few seconds, also ensures that optical flow alone will not be able to solve this prediction task.

Temporal receptive field. The temporal receptive field (RF) of $f(.)$ is limited by cutting the input video clip into non-overlapping blocks before feeding it into $f(.)$. Thus, the effective temporal RF of each feature map z_i is strictly restricted to be within each video block. This avoids the network being able to discriminate positive and hard-negative by recognizing relative temporal position.

Spatial receptive field. Due to the depth of CNN, each feature vector $\hat{z}_{i,k}$ in the final predicted feature map $\hat{z}_i$ has a large spatial RF that (almost) covers the entire input spatial dimension. This creates a shortcut to discriminate positive and spatial negative by using padding patterns. One can limit the spatial RF by cutting input frames into patches [40, 17]. However this brings some drawbacks: First, the self-supervised pre-trained network will have limited receptive field (RF), so the representation may not generalize well for downstream tasks where a large RF is required. Second, limiting spatial RF in videos makes the context feature too weak. The context feature has a spatio-temporal RF that covers a thin cube in the video flow. Neglecting context is also not ideal for understanding video semantics and brings ambiguity to the predictive task. Considering this trade-off, our method does not restrict the spatial RF.

Batch normalization. Common practice uses Batch Normalization [11] (BN) in deep CNN architecture. The BN layer may provide shortcuts that the network acknowledges the statistical distribution of the mini-batch, which benefits the classification. In [40], the authors demonstrate BN results in network cheating, and the ResNet trained with BN does not generalize to the downstream image classification task. In our method, we find the effect of BN shortcut is very limited. The self-supervised training gives similar accuracy using either BN or Instance Normalization [39] (IN). For downstream tasks like classification, a network with BN gives 5%-10% accuracy gain comparing with a network with IN. It is hard to train a deep CNN without normalization for either self-supervised training or supervised training. Overall, we use BN in our encoder function $f(.)$.

3.4. Network Architecture

We choose to use a 3D-ResNet similar to [10] as the encoder $f(.)$. Following the convention of [6] there are four residual blocks in ResNet architecture, namely res_2, res_3, res_4 and res_5, and only expand the convolutional kernels in res_4 and res_5 to be 3D ones. For experiment analysis, we used 3D-ResNet18, denoted as R-18 below.

To train a strong encoder $f(.)$, a weak aggregation function $g(.)$ is preferable. Specifically, a one-layer Convolutional Gated Recurrent Unit (ConvGRU) with kernel size $(1, 1)$ is used, which shares the weights amongst all spatial positions in the feature map. This design allows the aggregation function to propagate features in the temporal axis. A dropout [35] with $p = 0.1$ is used when computing hidden state in each time step. A shallow two-layer perceptron is used as the predictive function $\phi(.)$.

3.5. Self-Supervised Training

For data pre-processing, we use 30 fps videos with a uniform temporal downsampling by factor 3, *i.e*. take one frame from every 3 frames. These consecutive frames are grouped into 8 video blocks where each block consists of 5 frames. Frames are sampled in a consecutive way with consistent temporal stride to preserve the temporal regularity, because random temporal stride introduces uncertainties to the predictive task especially when the network needs to distinguish the difference among different time steps. Specifically, each video block spans over 0.5s and the entire 8 segments span over 4s in the raw video. The predictive task is initially designed to observe the first 5 blocks and predict the remaining 3 blocks (denoted as '5pred3' afterwards), which is observing 2.5 seconds to predict the following 1.5 seconds. We also experiment with different predictive configuration like 4pred4 in Sec. 4.1.4.

For data augmentation, we apply random crop, random horizontal flip, random grey, and color jittering. Note that the random crop and random horizontal flip are applied for the entire clip in a consistent way. Random grey and color jittering are applied in a frame-wise manner to prevent the network from learning low-level flow information as mentioned above (in Sec. 3.3), *e.g*. each video block may contain both colored and grey-scale image with different contrast. All models are trained end-to-end using Adam [18] optimizer with an initial learning rate 10^{-3} and weight decay 10^{-5}. Learning rate is decayed to 10^{-4} when validation loss plateaus. A batchsize of 64 samples per GPU is used, and our experiments use 4 GPUs.

4. Experiments and Analysis

In the following sections we present controlled experiments, and aim to investigate four aspects: *First*, an ablation study on the DPC model to show the function of different design choices, *e.g*. sequential prediction, dense prediction. *Second*, the benefits of training on a larger, and more diverse dataset. *Third*, the correlation between performance on self-supervised learning and performance on the downstream supervised learning task. *Fourth*, the variation in the learnt representations when predicting further into the future.

Datasets. The DPC is a general self-supervised learning framework for any video types, but we focus here on human action videos *e.g*. UCF101 [34], HMDB51 [20] and Kinetics-400 [16] datasets. UCF101 contains 13K videos spanning over 101 human action classes. HMDB51 contains 7K videos from 51 human action classes. Kinetics-400 (K400) is a big video dataset containing 306K video clips for 400 human action classes.

Evaluation methodology. The self-supervised model is trained either on UCF101 or K400. The representation is evaluated by its performance on a downstream task, *i.e*. action classification on UCF101 and HMDB51. For all the experiments below: we report top1 accuracy for self-supervised learning in the middle column of all tables; and report the top1 accuracy for supervised learning for action classification on UCF101 in the rightmost column. In self-supervised learning, the top1 accuracy refers to how often the multi-way classifier picks the right Pred-GT pair, *i.e*. this is not related with any action classes. While for supervised learning, the top1 accuracy indicates the action classification accuracy on UCF101. Note, we report the first training/testing splits of UCF101 and HMDB51 in all the experiments, apart from the comparison with the state of the art in Table 4 where we report the average accuracy over three splits.

Action classifier. During supervised learning, 5 video blocks are passed as input (the same as for self-supervised training, *i.e*. each block is of $\mathbb{R}^{5\times128\times128\times3}$), and encoded as a sequence of feature maps with the encoding function $f(.)$ (a 3D-ResNet). As with the self-supervised architecture, the aggregation function $g(.)$ (a ConvGRU) aggregates the feature maps over time and produces a context feature. The context feature is further passed through a spatial pooling layer followed by a fully-connected layer and a multi-way softmax for action classification. The classifier is trained using the Adam [18] optimizer with an initial learning rate 10^{-3} and weight decay 10^{-3}. Learning rate is decayed twice to 10^{-4} and 10^{-5} Note that the *entire* network is trained end-to-end. The details of the architecture are given in Appendix A.

During inference, video clips from the validation set are densely sampled from an input video and cut into blocks ($\mathbb{R}^{5\times128\times128\times3}$) with half-length overlapping. Augmentations are removed and only center crop is used. The softmax probabilities are averaged to give the final classification result.

4.1. Performance Analysis

4.1.1 Ablation Study on Architecture

In this section, we present an ablation study by gradually removing components from the DPC model (see Table 1). For efficiency, all the self-supervised learning experiments refer to the 5pred3 setting, *i.e*. 5 video blocks (2.5 second) are used as input to predict the future 3 steps (1.5 second).

Network	Self-Sup. (UCF)			Sup. (UCF)
	setting	method	top1 acc	top1 acc
R-18	-	- (rand. init.)	-	46.5
R-18	5pred3	DPC	**53.6**	**60.6**
R-18	5pred3	remove Seq.	51.3	56.9
R-18	5pred3	remove Map	36.5	44.9

Table 1: Ablation study of DPC. *remove Seq* means removing the sequential prediction mechanism in DPC, and replacing by parallel prediction. *remove Map* means removing the dense feature map design in DPC, and use a feature vector instead. Self-supervised tasks are trained on UCF101 using 5pred3 setting. Representation learned from each self-supervised task is evaluated by training a supervised action classifier on UCF101.

Compared with the baseline model trained with random initialization and fully supervised learning, our DPC model pre-trained with self-supervised learning has a significant boost (top1 acc: 46.5% vs. 60.6%). When removing the sequential prediction, *i.e*. all 3 future steps are predicted in parallel with three different fully-connected layers, the accuracy for both self-supervised learning and supervised learning start to drop. Lastly, we further replace the dense feature map by the average-pooled feature vector, *i.e*. it becomes a CPC-like model, we are not able to train this model either on self-supervised learning task or supervised learning. This demonstrates that *dense* predictive coding is essential to our success, and sequential prediction also helps to boost the model performance.

4.1.2 Benefits of Large Datasets

In this section, we investigate the benefits of pre-training on a large-scale dataset (UCF101 vs. K400), we keep the 5pred3 setting and evaluate the effectiveness for downstream task on UCF101. Results are shown in Table 2.

Network	Self-Sup.			Sup. (UCF)
	setting	dataset	top1 acc	top1 acc
R-18	5pred3	UCF101	53.6	60.6
R-18	5pred3	K400	**61.1**	**65.9**

Table 2: Results of DPC on UCF101 and K400 respectively. Both experiments use 5pred3 setting. Representations are evaluated by training a supervised action classifier on UCF101 (right column).

Training the model on K400 increases the self-supervised accuracy to 61.1%, and supervised accuracy from 60.6% to 65.9%, suggesting the model has captured more regularities than a smaller dataset like UCF101. It is clear that DPC will benefit from large-scale video dataset (infinite supply available), which naturally provides more diverse negative Pred-GT pairs.

4.1.3 Self-Supervised vs. Classification Accuracy

In this section, we investigate the correlation between the accuracy of self-supervised learning and downstream supervised learning. While training DPC (5pred3 task on K400), we evaluate the representation at different training stages (number of epochs) on the downstream task (on UCF101). The results are shown in Figure 3.

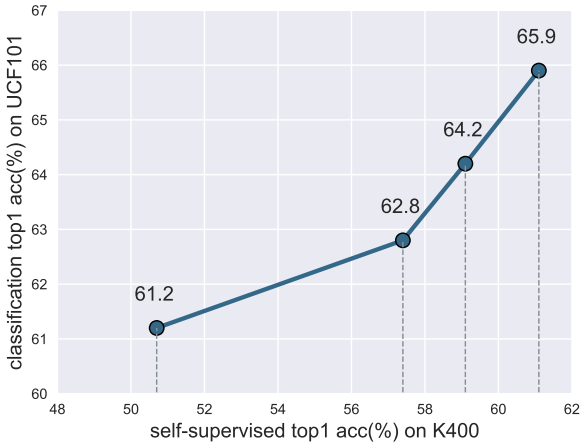


Figure 3: Relation between self-supervised accuracy and classification accuracy. Self-supervised model (DPC) is trained on K400 and the weights at epoch $\{13, 48, 81, 109\}$ are saved, which achieve $\{50.7\%, 57.4\%, 59.1\%, 61.1\%\}$ self-supervised accuracy respectively. The checkpoints are evaluated by finetuning on UCF101.

It can be seen that a higher accuracy in self-supervised task always leads to a higher accuracy in downstream classification. The result indicates that DPC has actually learnt visual representations that are not only specific to self-supervised task, but are also generic enough to be beneficial for the downstream task.

4.1.4 Benefits of Predicting Further into the Future

Due to the increase of uncertainty, predicting further into the future in video sequences gets more difficult, therefore more abstract (semantic) understanding is required. We hypothesize that if we can train the model to predict further, the learnt representation should be even better. In this section, we employ curriculum learning to gradually train the model to predict further with progressively less temporal context, *i.e*. from 5pred3 to 4pred4 (4 video blocks as input and predict the future 4 steps).

The result shows that the 4pred4 setting gives a substantially lower accuracy on the self-supervised learning than

Network	Self-Sup. (K400)			Sup. (UCF)
	setting	curr.	top1 acc	top1 acc
R-18	5pred3	✗	61.1	65.9
R-18	4pred4	✗	48.3	64.9
R-18	5pred3+4pred4	✓	**50.8**	**68.2**

Table 3: Results of DPC with different prediction steps. All models are trained on K400 with *same* number of 320k iterations. Note that for 5pred3 and 4pred4, the model is trained from scratch. '5pred3+4pred4' denotes that curriculum learning strategy, *i.e*. initialized with the pre-trained weights from 5pred3 task. The representation is evaluated by training an action classifier on UCF101 (right column).

5pred3. This is actually not surprising, as 4pred4 naturally introduces 33% more hard negative pairs than predicting future 3 steps, making the self-supervised learning more difficult (explained in Section 3.2).

Interestingly, despite a lower accuracy on self-supervised learning task, when comparing with 5pred3, curriculum learning on 4pred4 provides 2.3% performance boost on the downstream supervised task (top1 acc: 68.2% vs. 65.9%). The experiment also shows that curriculum learning is effective as it achieves higher performance than training 4pred4 task from scratch (top1 acc: 68.2% vs. 64.9%). Similar effect is also observed in [19].

4.1.5 Summary

Through the experiments above, we have demonstrated the keys to the success of DPC. *First*, it is critical to do dense predictive coding, *i.e*. predicting both temporal and spatial representation in the future blocks, and sequential prediction enables a further boost in the quality of the learnt representation. *Second*, a large-scale dataset helps to improve the self-supervised learning, as it naturally contains more world patterns and provides more diverse negative sample pairs. *Third*, the representation learnt from DPC is generic, as a higher accuracy in the self-supervised task also yield a higher accuracy in the downstream classification task. *Fourth*, predicting further into the future is also beneficial, as the model is forced to encode the high-level semantic representations, and ignore the low-level information.

5. Comparison with State-of-the-art Methods

The results are given in Table 4, four phenomena can be observed: *First*, when self-supervised training with only UCF101, our DPC (60.6%) outperforms all previous methods under similar settings. Note that OPN [22] performs worse when input resolution increases, which indicates a simple self-supervised task like order prediction may not capture the rich semantics from videos. *Second*, when using Kinetics-400 for self-supervised pre-training, our DPC (68.2%) outperforms all the previous methods by a large margin. Note that, in the work [15, 17], the authors use a full-scale 3D-ResNet18 architecture (33.6M parameters), *i.e*. all convolutions are 3D, however our modified 3D-ResNet18 has fewer parameters (only the last 2 blocks are 3D convolutions). The authors of [17] obtain 65.8% accuracy by combing the rotation classification [15] with their Space-Time Cubic Puzzles method, essentially multi-task learning. When only considering their Space-Time Cubic Puzzles method, they obtain 63.9% top1 accuracy. On HMDB51, our method also outperforms the previous state of the art result by 0.8% (34.5% vs. 33.7%). *Third*, when applying on larger input resolution (224×224) and using model with more capacity (3D-ResNet34), our DPC clearly dominate all self-supervised learning methods (75.7% on UCF101 and 35.7% on HMDB51), further demonstrating that DPC is able to take advantage from networks with more capacity and today's large-scale datasets. *Fourth*, ImageNet pretrained weights have been a golden baseline for action recognition [33], our self-supervised DPC is the first model that surpasses the performance of models (VGG-M) pre-trained with ImageNet (75.7% vs. 73.0% on UCF101).

5.1. Visualization

We visualize the Nearest Neighbour (NN) of the video segments in the spatio-temporal feature space in Figure 4 and Figure 1. In detail, one video segment is randomly sampled from each video, then the spatio-temporal feature $z_i = f(x_i)$ is extracted and pooled into a vector. Then the feature vector is used to compute the cosine similarity score. In all figures, Figure 4a includes the video clips retrieved using our DPC model from self-supervised learning, note that the network does not receive any class label information during training. In comparison, Figure 4b uses the inflated ImageNet pre-trained weights.

It can be seen, that the ImageNet model is able to encode the scene semantics, *e.g*. human faces, crowds, but does not capture any semantics about the human actions. In contrast, our DPC model has actually learnt the video semantics *without* using any manual annotation, for instance, despite the background change in running, DPC can still correctly retrieve the video block.

5.2. Discussion

Why should the DPC model succeed in learning a representation suitable for action recognition, given the problem of a non-deterministic future? There are three reasons: First, the use of the softmax function and multi-way classification loss enables multi-modal, skewed, peaked or long tailed distributions; the model can therefore handle the task of predicting the non-deterministic future. Second, by avoiding the shortcuts, the model has been prevented from learning simple smooth extrapolation of the embeddings; it is forced to learn semantic embeddings to succeed in its learning task. Third, in essense, DPC is trained by predicting future representations, and use them as a "query" to pick

Self-Supervised Method (RGB stream only)			Supervised Accuracy (top1 acc)	
Method	Architecture (#param)	Dataset	UCF101	HMDB51
Random Initialization	3D-ResNet18 (14.2M)	-	46.5	17.1
ImageNet Pretrained [33]	VGG-M-2048 (25.4M)	-	73.0	40.5
Shuffle & Learn [27] (227×227)	CaffeNet (58.3M)	UCF101/HMDB51	50.2	18.1
OPN [22] (80×80)	VGG-M-2048 (8.6M)	UCF101/HMDB51	59.8	23.8
OPN [22] (120×120)	VGG-M-2048 (11.2M)	UCF101/HMDB51	55.4	-
OPN [22] (224×224)	VGG-M-2048 (25.4M)	UCF101/HMDB51	51.9	-
Ours (128×128)	3D-ResNet18 (14.2M)	UCF101	**60.6**	-
3D-RotNet [15] (112×112)	3D-ResNet18-full (33.6M)	Kinetics-400	62.9	33.7
3D-ST-Puzzle [17] (224×224)	3D-ResNet18-full (33.6M)	Kinetics-400	63.9 (65.8*)	33.7*
Ours (128×128)	3D-ResNet18 (14.2M)	Kinetics-400	**68.2**	**34.5**
Ours (224×224)	3D-ResNet34 (32.6M)	Kinetics-400	**75.7**	**35.7**

Table 4: Comparison with other self-supervised methods, results are reported as an average over three training-testing splits. Note that, previous works [15, 17] use full-scale 3D-ResNet18, *i.e.* all convolutions are 3D, and the input sizes for different models have been shown. *indicates the results from the multi-task self-supervised learning, *i.e.* Rotation + 3D Puzzle.

(a)

(b)

Figure 4: More examples of video retrieval with nearest neighbour (same setting as Figure 1). Figure 4a is the NN retrieval with DPC pre-trained $f(.)$ on UCF101 (performance reported in Sec. 4.1.2). Figure 4b is the NN retrieval with ImageNet inflated $f(.)$. Retrieval is performed on UCF101 validation set.

the correct "key" from lots of distractors. In order to succeed in this task, the model has to learn the shared semantics of the multiple possible future states, as this is the only way to always solve the multiple choice problem, no matter what future state appears along with the distractors. This common/shared representation is the invariance we are wishing for, *i.e.* higher level semantics. In other words, the representation of all these possible future states will be mapped to a space that their embeddings are close.

6. Conclusion

In this paper, we have introduced the Dense Predictive Coding (DPC) framework for self-supervised representation learning on videos, and outperformed the previous state-of-the-art by a large margin on the downstream tasks of action classification on UCF101 and HMDB51. As for future work, one straightforward extension of this idea is to employ different methods for aggregating the temporal information – instead of using a ConvGRU for temporal aggregation ($g(.)$ in the paper), other methods like masked CNN and attention based methods are also promising. In addition, empirical evidence shows that optical flow is able to boost the performance for action recognition significantly; it will be interesting to explore how optical flow can be trained jointly with DPC with self-supervised learning to further enhance the representation quality.

Acknowledgements

Funding for this research is provided by the Oxford-Google DeepMind Graduate Scholarship, and by the EPSRC Programme Grant Seebibyte EP/M013774/1.

References

[1] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *ICCV*, 2015. 2

[2] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R. Devon Hjelm. Unsupervised state representation learning in atari. In *NIPS*, 2019. 2

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. In *JMLR*, 2003. 2

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 2

[5] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *CVPR*, 2015. 2, 4

[6] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *arXiv preprint arXiv:1812.03982*, 2018. 5

[7] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017. 2

[8] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2

[9] Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010. 2, 3

[10] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018. 5

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[12] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. In *ICLR*, 2015. 2

[13] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015. 2

[14] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *CVPR*, 2016. 2

[15] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning by video geometric transformations. *arXiv preprint arXiv:1811.11387*, 2018. 2, 7, 8

[16] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2, 5

[17] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, 2019. 2, 4, 7, 8

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[19] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *NIPS*, 2018. 2, 7

[20] Hilde Kuehne, Huei-han Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 5

[21] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. In *BMVC*, 2019. 2

[22] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequence. In *ICCV*, 2017. 4, 7, 8

[23] William Lotter, Gabriel Kreiman, and David D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*, 2017. 2

[24] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 2

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *NIPS*, 2013. 2

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 2

[27] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016. 2, 8

[28] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*, 2013. 2, 3

[29] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2, 4

[30] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *ICCV*, 2017. 2

[31] Deepak Pathak, Ross B. Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 2

[32] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[33] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 7, 8

[34] 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. 5

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *JMLR*, 2014. 5

[36] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015. 2

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 3

[38] Oren Tadmor, Yonatan Wexler, Tal Rosenwein, Shai Shalev-Shwartz, and Amnon Shashua. Learning a metric embedding for face recognition using the multibatch method. In *NIPS*, 2016. 4

[39] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 4

[40] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2, 3, 4

[41] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabelled video. In *CVPR*, 2016. 2

[42] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NIPS*, 2016. 2

[43] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018. 2

[44] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2

[45] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 2

[46] Donglai Wei, Joseph Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In *CVPR*, 2018. 2, 4

[47] Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. In *Neural Computation*, 2002. 2

[48] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *ECCV*, 2016. 2

Appendix

A. Architectures in detail

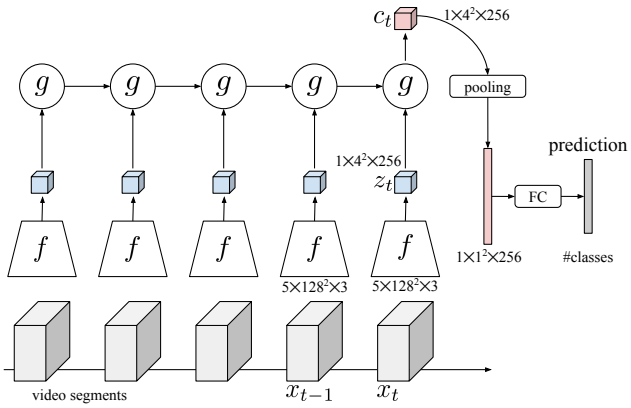


Figure 5: The action classifier structure used to evaluate the representation.

We use tables to display CNN structures. The dimension of convolutional kernels are denoted by {temporal × spatial^2, channel size}. The strides are denoted by {temporal stride, spatial stride^2}. The 'output sizes' column displays the dimension of feature map after the operation (except the dimension of input data in the first row), where $\{t \times d^2 \times C\}$ denotes {temporal size × spatial size^2 × channel size}, and T denotes the number of video blocks. In the following tables we take 3D-ResNet18 backbone with 128×128 input resolution as an example.

Structure of the action classifier. Table 5 gives the details of the action classifier which is used to evaluate the learned representation. Figure 5 is a diagram of the action classifier structure. For an input video with 30 fps, first a temporal stride 3 is applied, *i.e.* every 3rd frame is taken, resulting in 10 fps. Then $T \times 5$ consecutive frames are sampled and truncated into T video blocks, *i.e.* each video block has a size $5 \times 128^2 \times 3$, and we take $T = 5$ for the action classifier.

The action classifier is built with $f(.)$ and $g(.)$. The encoder function $f(.)$ takes 5 video blocks, each block contains 5 video frames ($5 \times (5 \times 128^2 \times 3)$) as input, spatio-temporal features (z) are extracted from the 5 video blocks with shared encoder ($f(.)$). Then the aggregation function $g(.)$ (ConvGRU) aggregates the 5 spatio-temporal feature maps into one spatio-temporal feature map, which is referred to as the context c in the paper. The context c is then pooled into a feature vector followed by a fully-connected layer.

Structure of the DPC. The DPC is built from $f(.)$ and $g(.)$ with an additional prediction mechanism, which is described in Table 6. Here we use 5pred3 setting for an example, where $f(.)$ takes 5 video blocks and extracts 5 spatio-temporal feature maps, then $g(.)$ aggregates feature maps into context c. The prediction function $\phi(.)$ is a two-layer perceptron, which takes the context c as input and produces a predicted feature $\hat{z}$ as output. The contrastive loss is computed using z and $\hat{z}$ as described in the paper Sec. 3.2.

module	specification	output sizes $T \times t \times d^2 \times C$
input data	-	$5 \times (5 \times 128^2 \times 3)$
$f(.)$	see Table 7	$5 \times (1 \times 4^2 \times 256)$ (z)
$g(.)$	see Table 8	$1 \times 1 \times 4^2 \times 256$ (c)
pool	1×4^2 stride $1, 1^2$	$1 \times 1 \times 1^2 \times 256$
final fc	1-layer FC	$1 \times 1 \times 1^2 \times$ # classes
	compute cross-entropy loss	

Table 5: The structure of the linear classifier.

module	specification	output sizes $T \times t \times d^2 \times C$
input data	-	$5 \times (5 \times 128^2 \times 3)$
$f(.)$	see Table 7	$5 \times (1 \times 4^2 \times 256)$ (z)
$g(.)$	see Table 8	$1 \times 1 \times 4^2 \times 256$ (c)
$\phi(.)$	2-layer FC	$1 \times 1 \times 4^2 \times 256$ ($\hat{z}$)
	compute loss using z and $\hat{z}$	

Table 6: The structure of DPC model.

Structure of $f(.)$. The detailed structure of the encoder function $f(.)$ is shown in Table 7. Note that $f(.)$ takes input video blocks independently, so the number of video block T is omitted in the table.

stage	specification	output sizes $t \times d^2 \times C$
input data	-	$5 \times 128^2 \times 3$
conv$_1$	$1 \times 7^2, 64$ stride $1, 2^2$	$5 \times 64^2 \times 64$
pool$_1$	$1 \times 3^2, 64$ stride $1, 2^2$	$5 \times 32^2 \times 64$
res$_2$	$\begin{bmatrix} 1 \times 3^2, 64 \\ 1 \times 3^2, 64 \end{bmatrix} \times 2$	$5 \times 32^2 \times 64$
res$_3$	$\begin{bmatrix} 1 \times 3^2, 128 \\ 1 \times 3^2, 128 \end{bmatrix} \times 2$	$5 \times 16^2 \times 128$
res$_4$	$\begin{bmatrix} 3 \times 3^2, 256 \\ 3 \times 3^2, 256 \end{bmatrix} \times 2$	$3 \times 8^2 \times 256$
res$_5$	$\begin{bmatrix} 3 \times 3^2, 256 \\ 3 \times 3^2, 256 \end{bmatrix} \times 2$	$2 \times 4^2 \times 256$
pool$_2$	$2 \times 1^2, 256$ stride $1, 1^2$	$1 \times 4^2 \times 256$

Table 7: The structure of the encoding function $f(.)$ with 3D-ResNet18 backbone.

Structure of $g(.)$. The structure of the temporal aggregation function $g(.)$ is shown in Table 8. It aggregates the fea-

ture maps over the past T time steps. Note that in the case of sequential prediction, T increments by 1 after each prediction step. Table 8 shows the case where $g(.)$ aggregates the feature maps over the past 5 steps.

stage	specification	output sizes $T \times t \times d^2 \times C$
input data	-	$5 \times 1 \times 4^2 \times 256$
ConvGRU	$[1^2, 256] \times 1$ layer	$1 \times 1 \times 4^2 \times 256$

Table 8: The structure of aggregation function $g(.)$.

B. t-SNE clustering of DPC context representation

This section shows the t-SNE clustering of the context representation on UCF101 extracted by $f(.)$ and $g(.)$ (Figure 6). In detail, 5 consecutive video blocks are sampled from each video in the validation set, then the feature maps $\{z_1, ..., z_5\}$ are extracted from each video block and aggregated into context representation c_5 and then pooled into vectors. We use t-SNE to visualize the context vectors in 2D. For clarity, only 10 action classes (out of 101 classes from UCF101) are displayed. The upper-left figure visualizes the context features extracted by randomly initialized $f(.)$ and $g(.)$. The following 3 figures show the context features extracted by $f(.)$ and $g(.)$ after $\{13, 48, 109\}$ epochs of DPC training on K400, without any finetuning on UCF101.

It can be seen that as the DPC training proceeds the intra-class distance is reduced (compared to the random initialization) and also the inter-class distance is increased, *i.e.* the self-supervised DPC method is clustering the feature vectors into action classes.

C. Cosine distance histogram of DPC context representation

This section shows the cosine distance of the context representation on UCF101 extracted by DPC pre-trained $f(.)$ and $g(.)$ (Figure 7). We use the same setting as Figure 6 and extract one context representation for each video and pool into vector. Then we compute the cosine distance of each pair of context vectors across the entire UCF101 validation set. The cosine distance is summarized by histogram, where 'positive' means two source videos are from the same action class and 'negative' means two source videos are from different action classes. For clarity, 17 out of 101 action classes are evenly sampled from UCF101 and visualized. Note that there is no finetunning in this stage, *i.e.* the network doesn't see any action labels.

It can be seen that for all action classes, the context representations from the same action class have higher cosine similarity, *i.e.* DPC can cluster actions without knowing action labels.

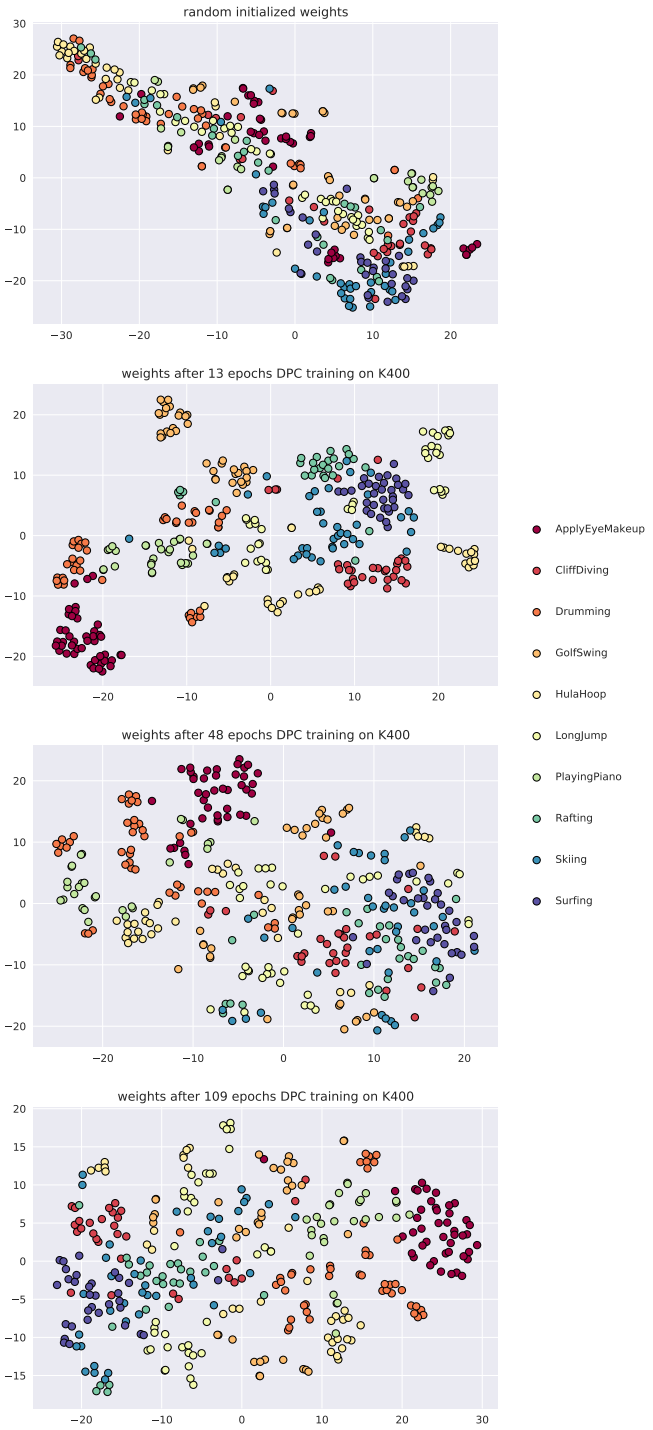


Figure 6: t-SNE visualization of the context representations on UCF101 validation set extracted by different $f(.)$ and $g(.)$ after $\{0$ (random init.), $13, 48, 109\}$ epochs DPC training on K400.

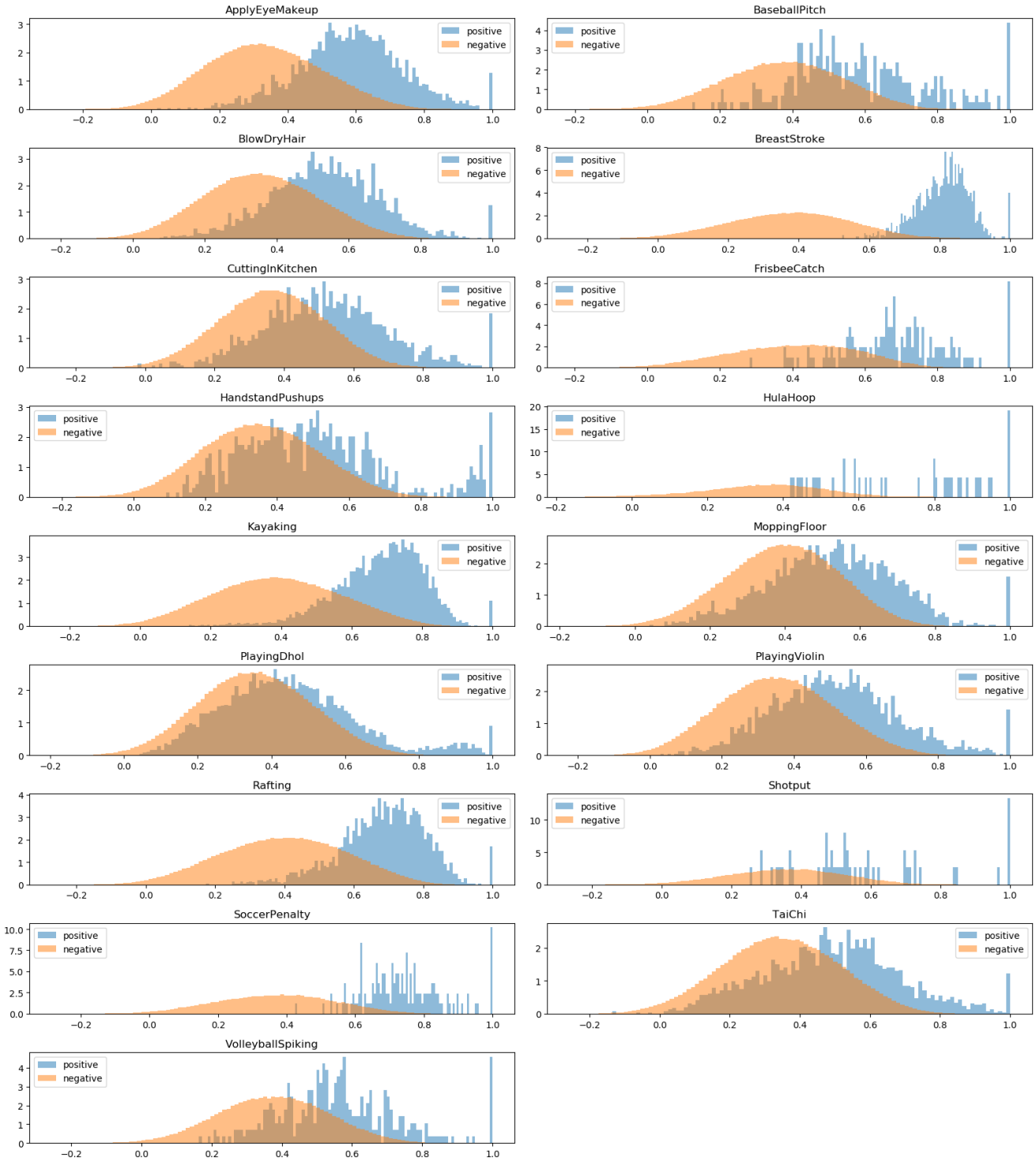


Figure 7: Histogram of the cosine distance of the context representations extracted from UCF101 validation set by DPC weights. 'Positive' and 'negative' refer to the video pairs that are from the same or different action classes. DPC is trained on K400 without any finetuning on UCF101.