Pixel-level Correspondence for Self-Supervised Learning from Video

Yash Sharma† 1 Yi Zhu 2 Chris Russell 2 Thomas Brox 2 3

Abstract

While self-supervised learning has enabled effective representation learning in the absence of labels, for vision, video remains a relatively un-tapped source of supervision. To address this, we propose Pixel-level Correspondence (PiCo), a method for dense contrastive learning from video. By tracking points with optical flow, we obtain a correspondence map which can be used to match local features at different points in time. We validate PiCo on standard benchmarks, outperforming self-supervised baselines on multiple dense prediction tasks, without compromising performance on image classification.

1. Introduction

Deep learning methods have yielded dramatic improvements in a plethora of domains by extracting useful representations from raw data (Bengio et al., 2013; LeCun et al., 2015), albeit assuming the availability of ample supervision. Recent advancements in self-supervised learning (Mikolov et al., 2013; Devlin et al., 2018; Chen et al., 2020a; He et al., 2021) have enabled effective representation learning without curated, labeled datasets (Goyal et al., 2021).

Self-supervised learning obtains supervisory signals from the data itself through the careful construction of prediction tasks which do not rely on manual annotation, yet encourage the model to extract useful features. Specifically, the task of predicting whether a pair, or a set, of examples are views of the “same” image, or “different” images, underlies the recent success of contrastive methods for learning representations of visual data (Wu et al., 2018; Van den Oord et al., 2018; Henaff, 2020; Hjelm et al., 2018; Bachman et al., 2019; He et al., 2020; Chen et al., 2020a).

In contrastive learning, view selection crucially influences the quality of the resulting representations (Tian et al., 2020; Zimmermann et al., 2021; Von Kügelgen et al., 2021). Existing approaches (He et al., 2020; Chen et al., 2020a;b) have constructed views via hand-crafted data augmentations, e.g. cropping sub-regions of the images. Cropping yields views that depict object parts, and thereby induces a learning signal for invariance to occluded objects (Purushwalkam & Gupta, 2020). With that said, augmentations are inherently limited; given a single image, simulating variation in object size, shape, or viewpoint can be difficult. Notably, such variation is ubiquitous in video (see Figure 1). The promise of temporal variation for representation learning has encouraged ample investigation in the context of self-supervision (Misra et al., 2016; Wei et al., 2018; Wang et al., 2019; Vondrick et al., 2018; Isola et al., 2015; Wiskott & Sejnowski, 2002; Klindt et al., 2020; Agrawal et al., 2015; Weis et al., 2021; Lachapelle et al., 2021).

How can we leverage video for learning self-supervised representations of images? While existing work has proposed a multitude of strategies (Wang & Gupta, 2015; Wang et al., 2017; Tschannen et al., 2020; Purushwalkam & Gupta, 2020; Gordon et al., 2020; Romijnders et al., 2021; Xiong et al., 2021; Wu & Wang, 2021; Chen et al., 2021), nearly all exploit instance discrimination methods (Dosovitskiy et al., 2014; Kolesnikov et al., 2019; He et al., 2020; Chen et al., 2020b) designed for global representation learning, or learning encodings at the image-level. However, recent work (Pinheiro et al., 2020; Wang et al., 2021; Xie et al., 2021)
We thus propose PitCo, a method for dense representation learning from video. Existing work proposed for static images has relied upon aforementioned geometric transformations, e.g., crops, to introduce variation. We demonstrate that temporal variation can also be utilized by tracking points using off-the-shelf optical flow estimators. We find that across a number of downstream tasks, PitCo outperforms existing work restricted to static frames, as well as existing work applied to video assuming static pixel correspondence.

2. Background

As our contribution enables dense representation learning to exploit the natural transformations inherent to video, we will focus on extending a method which learns representations through pixel-level contrastive learning, VADeR (Pinheiro et al., 2020). Thus, we will give a short description of the learning method before proceeding with our contribution, see (Pinheiro et al., 2020) for further details.

Let us represent a pixel \( u \) in image \( x \in I \subset \mathbb{R}^{3 \times h \times w} \) by the tuple \( (x, u) \). Let \( f \) be an encoder-decoder convolutional network that produces a \( d \)-dimensional embedding for every pixel in the image, i.e. \( f : (x, u) \mapsto z \in \mathbb{R}^d \). VADeR’s objective is to learn an embedding function that encodes \((x, u)\) into a representation that is invariant w.r.t. any view \( v_1, v_2 \in V_u \) containing the pixel \( u \). This is achieved through contrastive learning (Gutmann & Hyvärinen, 2010; 2012; Van den Oord et al., 2018), where the objective optimized in practice is to distinguish between views of the same pixel and views of different pixels.

\[
\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_{(v_1, v_2) \sim V_u} \left[ \log \frac{\exp\{\text{sim}(f(v_1, u), f(v_2, u))\}}{\sum_{j=1}^{K} \exp\{\text{sim}(f(v_1, u), f(v_j', u'))\}} \right]
\]

(1)

where \( K - 1 \) is the number of negative pixels, and the positive pair in the numerator is included in the denominator summation, i.e. \( (v'_K, u'_K) = (v_2, u) \). For implementation, the design details for MoCo were followed (He et al., 2020).

For \( f \), the semantic segmentation branch of (Kirillov et al., 2019) was adopted. A feature pyramid network (FPN) (Lin et al., 2017) adds a top-down path to a ResNet-50 (He et al., 2016), generating a pyramid of features (from 1/32 to 1/4 resolution). By adding a number of upsampling blocks at each resolution of the pyramid, the pyramid representations are merged into a single dense output representation with dimension 128 and scale 1/4. The ResNet-50 is initialized with MoCo (He et al., 2020), and pretraining is performed on the ImageNet-1K (IN-1K) (Deng et al., 2009) train split.

3. Method

Here, we detail our procedure for constructing pixel correspondence maps from video for dense contrastive learning.

3.1. Data

For pretraining, we experiment with Kinetics400 (K400) (Kay et al., 2017) and YouTube-8M (YT8M) (Abu-El-Haija et al., 2016). The K400 training set consists of approximately 240,000 videos trimmed to 10 seconds from 400 human action categories. We sample frame sequences at 30 Hz (Kuang et al., 2021). For tractability, we construct a subset of YT8M (YT8M-S) which matches the dataset statistics of K400. Specifically, for 240,000 random videos, we sample 10-second snippets at 30 Hz from shots detected using an off-the-shelf network (Souček & Lokoč, 2020). Further details are provided in the appendix.

3.2. Trajectories

We first compute and store optical flow on K400 and YT8M-S. While in preliminary experiments we found alternatives (Ilg et al., 2017; Sun et al., 2018) to perform comparably, for the presented set of experiments, we use RAFT (Teed & Deng, 2020) trained on a mixed dataset (Contributors, 2021) consisting of FlyingChairs (Dosovitskiy et al., 2015), FlyingThings3D (Mayer et al., 2016), Sintel (Butler et al., 2012), KITTI-2015 (Menzel & Geiger, 2015; Geiger et al., 2013), and HD1K (Kondermann et al., 2016). The horizontal and vertical components of the flow were linearly rescaled to a [0, 255] range and compressed using JPEG (after decompression, the flow is rescaled back to its original range) (Simonyan & Zisserman, 2014).

With the precomputed flow, we track points in the video. For each video, we sample an initial set of 1000 points at random locations on random frames. As in (Sundaram et al., 2010), each point is tracked to the next frame using the flow field \( w = (u, v)^T \).

\[
(x_{t+1}, y_{t+1})^T = (x_t, y_t)^T + (u_t(x_t, y_t), v_t(x_t, y_t))^T
\]

(2)

Between pixels, the flow is inferred using bilinear interpolation. Tracking is stopped as soon as a point is occluded, which is detected by checking the consistency of the forward and backward flow. In a non-occlusion case, the backward flow vector should point in the inverse direction of the forward flow vector: \( u_t(x_t, y_t) = -\hat{u}_t(x_t + u_t, y_t + v_t) \) and \( v_t(x_t, y_t) = -\hat{v}_t(x_t + u_t, y_t + v_t) \), where \( \hat{w}_t = (\hat{u}_t, \hat{v}_t) \).
denotes the flow from frame $t + 1$ to frame $t$. We thus use the following threshold:

$$|w + \hat{w}|^2 < \gamma(|w|^2 + |\hat{w}|^2) + \delta$$  \hspace{1cm} (3)

3.3. Learning

Existing proposals for visual representation learning with contrastive methods from video typically sample random frames from a given shot for constructing views (Tschannen et al., 2020; Chen et al., 2021; Gordon et al., 2020). Given the endpoints for a set of trajectories in each video, we propose a frame selection strategy for maximizing temporal separation and trajectory density, anchor sampling. After sampling an anchor frame, for each trajectory active on said frame, we find the endpoint furthest from said frame. If we are to select $N$ frames for learning, we select the top-$N$ according to endpoint count. With this strategy, as we vary the threshold hyperparameters, the temporal separation between the selected frames varies accordingly.

3.4. Implementation Details

For both implementing the objective and initializing the encoder, we use MoCo-v2 (Chen et al., 2020b;a) instead of MoCo (He et al., 2020). Notably, we found the use of a nonlinear projection head to be critical for performance. As in existing work (Long et al., 2015; Wang et al., 2021; Bai et al., 2022), for dense contrastive learning, we replace the linear layers in the MoCo-v2 global projection head with 128-dimensional dense output representation; we found that reducing the number of parameters in the dense projection head decreased downstream performance. We also decided to freeze the initialized encoder, thereby maintaining the downstream image/scene classification performance of the image-level encoding.

Finally, note that for the experiments prior to ablation, $\gamma = 0$, $\delta = 4.0$, and, each iteration, no more than 65536 point pairs (from frame pairs selected from 256 videos) are used.

4. Experiments

We compare PtCo to a set of baselines across datasets & tasks. We provide a visual representation of the comparison in Figure 2. In Static (Frames), a single frame is sampled from each video for view construction, thus, as in (Pinheiro et al., 2020), the variation in corresponding pixels is solely due to the geometric transformations in the MoCo-v2 data augmentation pipeline, i.e. random crops and horizontal flips. In Static (Video), as in PtCo, we sample multiple frames from a given video, but unlike PtCo, the pixel correspondence map is static, i.e. the optical flow field is assumed to consist of zero vectors. By comparing to “Static (Video),” we can isolate the value point tracking is yielding downstream. Additional details regarding evaluation are provided in the appendix.

4.1. COCO Semantic Segmentation

It is common practice in self-supervised learning to assess the quality of frozen features with a linear probe (Goyal et al., 2019; Kolesnikov et al., 2019). Following (Pinheiro et al., 2020), the output of each model is processed by a $1 \times 1$ convolutional layer, $4 \times$ upsample, and softmax, where for MoCo-v2, the effective stride is reduced from 1/32 to 1/4 by replacing strided convolutions with dilated ones (Chen et al., 2017; Yu et al., 2017). The linear predictor weights are trained using cross entropy.

In Table 1, we observe a tangible improvement in leveraging point trajectories for dense contrastive learning. Interestingly, we find pretraining on K400 largely delivers improved performance relative to YT8M-S. In accordance with previous work (Gordon et al., 2020), we notice that a number of videos in YT8M are unnatural, e.g. “video games” or “cartoons”, which clearly yields a domain gap with “everyday scenes containing common objects in their natural context” (Lin et al., 2014).

4.2. Additional Tasks & Benchmarks

Tasks: In Table 2, we evaluate representations on COCO object detection and instance segmentation. For this, we use Mask R-CNN (He et al., 2017) with a frozen FPN backbone (Lin et al., 2017). While PtCo significantly outper-
Table 2. Mask R-CNN. K400 pretraining, COCO object detection & instance segmentation with FPN frozen.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>APbb 50</th>
<th>APbb 75</th>
<th>APmk 50</th>
<th>APmk 75</th>
<th>APbb 50</th>
<th>APbb 75</th>
<th>APmk 50</th>
<th>APmk 75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static (Frame)</td>
<td>K400</td>
<td>6.09</td>
<td>15.1</td>
<td>3.73</td>
<td>14.8</td>
<td>6.09</td>
<td>15.1</td>
<td>3.73</td>
<td>14.8</td>
</tr>
<tr>
<td>Static (Video)</td>
<td>K400</td>
<td>7.92</td>
<td>19.7</td>
<td>4.60</td>
<td>18.7</td>
<td>7.92</td>
<td>19.7</td>
<td>4.60</td>
<td>18.7</td>
</tr>
<tr>
<td>PtCo</td>
<td>K400</td>
<td>10.8</td>
<td>24.3</td>
<td>7.94</td>
<td>23.1</td>
<td>10.8</td>
<td>24.3</td>
<td>7.94</td>
<td>23.1</td>
</tr>
</tbody>
</table>

Table 3. Additional Benchmarks: K400 pretraining, linear probing frozen model.

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC</th>
<th>CS</th>
<th>NYU-d v2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static (Frame)</td>
<td>31.0</td>
<td>34.0</td>
<td>1.000</td>
</tr>
<tr>
<td>Static (Video)</td>
<td>34.7</td>
<td>28.7</td>
<td>0.958</td>
</tr>
<tr>
<td>PtCo</td>
<td>35.6</td>
<td>35.1</td>
<td>0.950</td>
</tr>
</tbody>
</table>

Figure 3. γ: COCO linear probing varying the tracking threshold parameters γ and δ. YT-8M-S pretraining, w/o anchor sampling.

Figure 4. δ: COCO linear probing with finer-grained variation in δ. γ = 0, w/o anchor sampling.

5. Discussion

Limitations While we observe improved performance over our baseline methods, overall performance remains worse than supervised approaches, and there is substantial room for improvement. Specifically, our decoder-only training on video, when compared to the reported scores of encoder-decoder training on IN-1K in a similar experimental setting (Pinheiro et al., 2020), underperforms. In future work, we suggest (i) addressing the domain gap between the video datasets used for pretraining and the image datasets used for benchmarking (Tang et al., 2012; Kalogeiton et al., 2016; Kae & Song, 2020) and (ii) considering alternatives to our strategy of freezing the encoder for maintaining classification performance whilst improving dense prediction.

Conclusion We present PtCo, an approach to dense contrastive learning on video. We show the benefit in constructing pixel correspondence maps over time on a number of tasks and datasets. Our work serves as a first step towards leveraging the temporal variation inherent to video for dense prediction tasks, and in that vein, we encourage further exploration along the aforementioned direction.

4.3. Ablations

Tracking Threshold: In Figure 3, we evaluate the impact of varying γ and δ. While we do consistently observe improved performance with increased δ (up to a point, see Figure 4), the same cannot be said for γ. Given the computational cost in adjusting the trajectories w.r.t. γ, we were limited in our ablation, and encourage further exploration on the effect of this parameter.

Anchor Sampling: In Figure 5, we isolate the effect of anchor sampling on the downstream performance. We can see that as we increase δ, thereby using longer trajectories for pretraining, the gap between anchor sampling and randomly sampling frames narrows. As δ increases, the likelihood that a point pair will exist between a random pair of frames also increases, while the very same likelihood is invariant to δ when using anchor sampling.
Acknowledgements

We thank Peter Gehler, Andrii Zadaianchuk, and Max Horn for compute infrastructure support. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. Y.S. thanks the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for support.

References


A. Additional Details

A.1. Training

YT8M-S Given the size of YT-8M, the authors decided to release frame-level features of the videos instead of the videos themselves (Abu-El-Haija et al., 2016). For our purposes, we extracted YT-8M URLs\(^1\), and downloaded a sampled subset at the scale of K-400. We note that a number of the YT-8M URLs are no longer accessible. We used TransNetV2 (Souček & Lokoč, 2020) off-the-shelf as a high-performing deep learning approach for shot boundary detection.

Tracking The most notable difference with (Sundaram et al., 2010) corresponds to starting point sampling. In (Sundaram et al., 2010), a grid is instantiated on the first frame, and points are re-instantiated as trajectories are stopped. In contrast, we sampled starting points uniformly in space and time, to ensure the same trajectory computation is applicable to variable \(\gamma\) and \(\delta\). For storing the trajectories, in particular the consecutive norm differences between forward and backward flow vectors, we used half-precision. Finally, note that the RHS of Equation (3) is dependent on the flow vectors through the \(\gamma\) term, thus tuning \(\gamma\) requires extra computation relative to solely tuning \(\delta\).

Training In order to compute the loss, we must map point pairs to feature indices. For this, we simply scale the point indices by \(1/4\), given the dense output representation is at \(1/4\) resolution.

A.2. Evaluation

For each configuration, we use the default FPN config provided in Detectron2\(^2\) as a basis.

A.2.1. COCO SEMANTIC SEGMENTATION

Dataset: Following (Kirillov et al., 2019), semantic annotations are converted from panoptic annotations for the 2017 challenge images, where all “things” are assigned the same semantic label, while each “stuff” category is assigned a unique semantic label.

MoCo-v2: As in (Pinheiro et al., 2020), the dilated resnet architecture is used (Chen et al., 2017; Yu et al., 2017). For each stage where the stride is decreased from 2 to 1, the dilation factor is multiplicatively scaled by 2. With that, the output resolution of the RN-50 is \(1/4\), and can thereby be evaluated using the same linear prediction protocol as used for the encoder-decoder architectures.

Configuration: For data augmentation, we perform random absolute crops of size \(672 \times 672\) after resizing using the default parameters, followed by a random flip.

A.2.2. COCO INSTANCE SEGMENTATION & OBJECT DETECTION

Configuration: Only discrepancy with the default configuration is freezing the FPN. Thus, in contrast to the semantic segmentation & depth prediction evaluation, where solely a linear predictor is learned, the learned modules here are the proposal generator & ROI heads.

A.2.3. VOC & CITYSCAPES SEMANTIC SEGMENTATION

VOC Configuration: The minimum size after resizing was decreased to 480, and an absolute crop size of \(512 \times 512\) was specified. Number of gradient steps was decreased to 40000, with milestone steps decreased to 25000 and 35000. Note that training was performed on the “train_aug” dataset.

Cityscapes Configuration: The minimum size after resizing was decreased to 512, and the maximum size was increased to 2048. Crops of size \(512 \times 1024\) were performed. Batch size was increased from 16 to 32, base learning rate was decreased from 0.02 to 0.01, and the number of gradient steps was decreased to 65000, with milestone steps decreased to 40000 and 55000.

\(^1\)used following repository.
\(^2\)see here
A.2.4. NYU-DEPTH V2 DEPTH PREDICTION

**Dataset:** Downloaded from the ViRB framework release (Kotar et al., 2021).

**Configuration:** Given the variable resolution, both input examples and labels were resized to $224 \times 224$ prior to training and testing. For augmentation, only random flips were employed.