Unsupervised Representation Learning by Sorting Sequences

Hsin-Ying Lee[1] Jia-Bin Huang[2] Maneesh Singh[3] Ming-Hsuan Yang[1]
[1]University of California, Merced [2]Virginia Tech [3]Verisk Analytics
http://vllab1.ucmerced.edu/~hylee/OPN/

Abstract

We present an unsupervised representation learning approach using videos without semantic labels. We leverage the temporal coherence as a supervisory signal by formulating representation learning as a sequence sorting task. We take temporally shuffled frames (i.e., in non-chronological order) as inputs and train a convolutional neural network to sort the shuffled sequences. Similar to comparison-based sorting algorithms, we propose to extract features from all frame pairs and aggregate them to predict the correct order. As sorting shuffled image sequence requires an understanding of the statistical temporal structure of images, training with such a proxy task allows us to learn rich and generalizable visual representation. We validate the effectiveness of the learned representation using our method as pre-training on high-level recognition problems. The experimental results show that our method compares favorably against state-of-the-art methods on action recognition, image classification, and object detection tasks.

1. Introduction

In recent years, Convolutional Neural Networks (CNNs) [18] have demonstrated the state-of-the-art performance in visual recognition tasks. The success of CNNs is primarily driven by millions of manually annotated data such as the ImageNet [5]. However, this substantially limits the scalability to new problem domains because manual annotations are often expensive and in some cases scarce (e.g., labeling medical images requires expertise). In contrast, a vast amount of *free* unlabeled images and videos are readily available. It is of great interest to explore strategies for representation learning by leveraging unlabeled data.

A new unsupervised learning paradigm has recently emerged as *self-supervised learning* [32, 42]. Within the context of deep neural networks, the key idea is to leverage the inherent structure of raw images and formulate a discriminative or reconstruction loss function to train the network. Examples include predicting the relative patch positions [7], reconstructing missing pixel values conditioned on the known surrounding [30], or predicting one subset of the data channels from another (e.g., predicting color channels from a gray image) [20, 43, 44]. Compared to image data, videos potentially provide much richer information as they not only consist of large amounts of image samples but also provide scene dynamics. Recent approaches explore unlabeled video data to learn feature representation through egomotion [2, 14], order verification [25, 9], tracking [42], and future frame prediction [24]. While these surrogate tasks do not directly use semantic labels, they provide effective supervisory signals as solving these tasks requires the semantic understanding of the visual data.

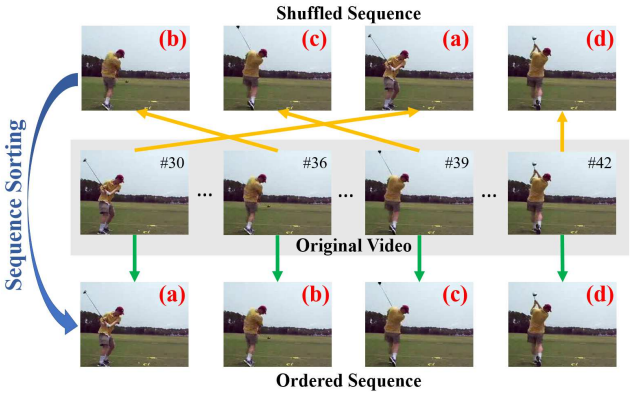


Figure 1: **Representations learning by sequence sorting.** Appearance variations and temporal coherence in videos offer rich supervisory signals for representation learning. Given a tuple of randomly shuffled frames (top row) sampled from an input video (middle row), we formulate the sequence sorting task as revealing the underlying chronology order of the sampled frames (bottom row). While no semantic labels are involved, solving this task requires high-level understanding of the temporal dynamics of videos. In this paper, we exploit the statistical temporal structure of images as our source of supervision.

In this paper, we propose a surrogate task for self-supervised learning using a large collection of unlabeled videos. Given a tuple of randomly shuffled frames, we train a neural network to sort the images into chronological or-

Figure 2: **Example tuples.** The examples shown here are automatically extracted tuples. Can you sort these shuffled frames? The answer is yes. By reasoning the relative poses using the knowledge of "how a person moves", we can predict the chronological order of the frames.

der, as shown in Figure 1. The *sequence sorting* problem provides strong supervisory signals as the network needs to reason and understand the statistical temporal structure of image sequences. We show several examples of shuffled frames in Figure 2. Our key observation is that we often unconsciously compare all pairs of frames to reason the chronological order of the sequence (as in comparison-based sorting methods). In light of this, we propose an Order Prediction Network (OPN) architecture. Instead of extracting features from all frames in a tuple simultaneously, our network first computes features from all the pairwise frames and fuses them for order prediction.

We conduct extensive experimental validation to demonstrate the effectiveness of using sequence sorting for representation learning. When used as a pre-training module, our method outperforms state-of-the-art approaches on the UCF-101 [37] and HMDB-51 [19] action benchmark datasets. While our model learns features from human action videos, we also demonstrate the generalizability for generic object classification and detection tasks, and show competitive performance on the PASCAL VOC 2007 dataset [8] when compared with the state-of-the-arts.

We make the following contributions in this work:

1) We introduce sequence sorting as a self-supervised representation learning approach using unlabeled videos. While feature learning based on sequence order has been exploited recently [25, 9, 15, 4], our sorting formulation is much richer than the binary verification counterparts [25].

2) We propose an Order Prediction Network architecture to solve the sequence sorting task by pairwise feature extraction. Quantitative results show that the proposed architecture provides significant performance improvement over the straightforward implementation.

3) We show that the learned representation can serve as a pre-trained model. Using less than 30,000 videos for unsupervised training, our model performs favorably against existing methods in action recognition benchmark datasets, and achieve competitive performance in classification and detection on the PASCAL VOC 2007 dataset.

2. Related Work

Unsupervised learning from static images. While CNNs have shown dominant performance in high-level recognition problems such as classification and detection, training a deep network often requires millions of manually labeled images. The inherent limitation from the fully supervised training paradigm highlights the importance of unsupervised learning to leverage vast amounts of unlabeled data. Unsupervised learning has been extensively studied over the past decades. Before the resurgence of CNNs, hand-craft features such as SIFT and HOG have been used to discover semantic classes using clustering [33, 36], or mining discriminative mid-level features [35, 6, 39]. With deep learning techniques, rich visual representations can be learned and extracted directly from images. A large body of literature focuses on reconstruction-based learning. Inspired from the original single-layer auto-encoders [28], several variants have been developed, including stack layer-by-layer restricted Boltzmann machines (RBMs), and auto encoders [3, 11, 21].

Another line of unsupervised learning is known as self-supervised learning. These methods define a supervisory signal for learning using the structure of the raw visual data. The spatial context in an image provides a rich source of supervision. Various existing approaches leverage spatial context for self-supervision, including predicting the relative patch positions [7], solving jigsaw puzzles [27], and inpainting missing regions based on their surrounding [30]. Another type of cue is through cross-channel prediction, e.g., image colorization [20, 43] and split-brain auto-encoders [44]. In addition to using only individual images, several recent directions have been explored by grouping visual entities using co-occurrence in space and time [13], using graph-based constraints [22], and cross-modal supervision from sounds [29]. Our work is similar to context-based approaches [7, 27, 30]. Instead of using *spatial* context of images, in this work we investigate the use of *temporal* context in videos.

Unsupervised learning from videos. The explosive increase of easily available videos on the web, like YouTube, presents an opportunity as well as several challenges for learning visual representations from unlabeled videos. Compared to images, videos provide the advantage of having an additional time dimension. Videos provide examples of appearance variations of objects over time. We can broadly categorize the unsupervised learning methods using

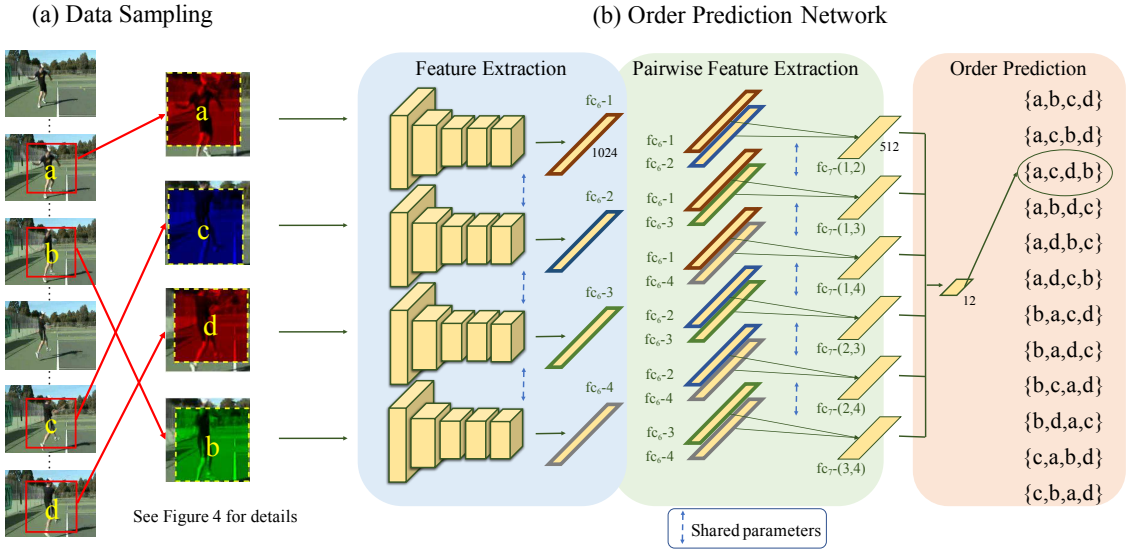


Figure 3: **Overview of our approach.** Our training strategy consists of two main steps. (a) *Data sampling* (Section 3.1). We sample candidate tuples from an input video based on motion magnitude. We then apply spatial jittering and channel splitting on selected patches to guide the network to focus on the semantics of the images rather than fixating on low-level features. Finally, we randomly shuffle the sampled patches to form an input tuple for training the CNN. (b) *Order Prediction Network* (Section 3.2). The proposed Order Prediction Network consists of three main components: (1) feature extraction, (2) pairwise feature extraction, and (3) order prediction. Features for each frame (fc_6) are encoded by convolutional layers. The pairwise feature extraction stage then extracts features from every pair of frames. We then have a final layer that takes these extracted features to predict order. Note that while we describe the architecture in three separate stages for the sake of presentation, the network training is end-to-end without stage-wise optimization.

videos into two groups. The first group focuses on frame reconstruction tasks, e.g., future frame prediction [38], frame interpolation [23], and video generation [40]. The second group learns feature representation by leveraging appearance variations presented in videos. Examples include enforcing the temporal smoothness of representation throughout a video [26, 15], applying tracking to capture appearance variation of moving objects [42], learning transformation in ego-motion videos [14, 2], verifying the order of input sequence [25, 9], and the transformation between color and optical flow [31].

The work most related to our method is that of [25, 9]. Similar to Misra et al. [25], our method makes use of the temporal order of frames as the source of supervision. However, instead of verifying correct/incorrect temporal order (i.e., binary classification), our supervisory signals are much richer: our network needs to predict $n!/2$ combinations for each n-tuple of frames. The proposed Order Prediction Network architecture also differs from the simple concatenation in [25]. The Order Prediction Network first extracts pairwise features and subsequently fuse the information for final predictions. Our quantitative results demonstrate performance improvement using the proposed design. Fernando et al. [9] exploit a similar notion of order verification to learn video representation. However, their approach takes as input a stack of frame differences and does not learn image representations. In contrast, our model can be used for *both* video understanding (e.g., action recognition) as well as image understanding (e.g., classification and detection) problems (as we show in Section 4).

3. Feature Learning by Sequence Sorting

Our goal is to capitalize the large quantity of *unlabeled* videos for feature learning. We propose to use sequence sorting as a surrogate task for training a CNN. Our hypothesis is that successfully solving the sequence sorting task will allow the CNN to learn useful visual representation to recover the temporal coherence of video by observing how objects move in the scene.

Specifically, we use up to four randomly shuffled frames sampled from a video as our input. Similar to the jigsaw puzzle problem in the spatial domain [27], we formulate the sequence sorting problem as a multi-class classification task. For each tuple of four frames, there are $4! = 24$ possible permutations. However, as some actions are both coherent forward and backward (e.g., opening/closing a door), we group both forward and backward permutations into the same class (e.g., $24/2$ classes for four frames). This forward-backward grouping is conceptually similar to the commonly used horizontal flipping for images. In the fol-

lowing, we describe two important factors in our approach: (1) training data sampling (Section 3.1) and (2) network architecture (Section 3.2).

3.1. Training data sampling

Preparing training data is crucial for self-supervised representation learning. In the proposed sequence sorting task, we need to balance the level of difficulty. On the one hand, sampling tuples from static regions produces nearly impossible tasks for the network to sort the shuffled sequence. On the other hand, we need to avoid the network picking up low-level cues to achieve the task. We describe three main strategies to generate our training data in this section.

Motion-aware tuple selection. We use the magnitude of optical flow to select frames with large motion regions similar to [25]. In addition to using optical flow magnitude for frame selection, we further select spatial patches with large motion. Specifically, for video frames in the range $[t_{min}, t_{max}]$, we use sliding windows to mine frame tuple $\{t_a, t_b, t_c, t_c\}$ with large motion, as illustrated in Figure 4(a).

Spatial jittering. As the previously selected tuples are extracted from the same spatial location, simple frame alignment could potentially be used to sort the sequence. We apply spatial jittering for each extracted patch to avoid the trivial cases (see Figure 4(b)).

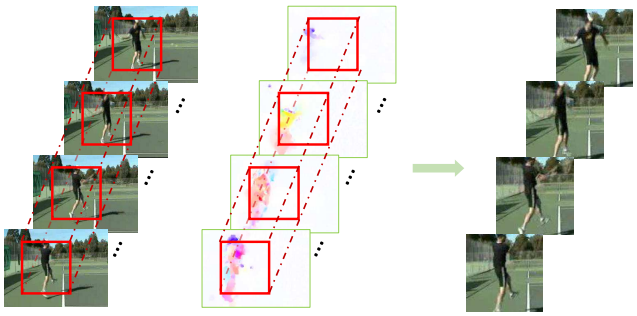
Channel splitting. To avoid the network from learning low-level features without semantic understanding, we apply channel splitting on the selected patches, as shown Figure 4(c). For each frame in a tuple, we randomly choose one channel and duplicate the values to other two channels. The effect is similar to using a grayscale image (as done in [42]). However, the use of channel splitting imposes additional challenges for the network compared with using grayscale images because grayscale images are generated from a fixed linear combination of the three color channels. We validate all design choices in Section 4.3.

3.2. Order Prediction Network

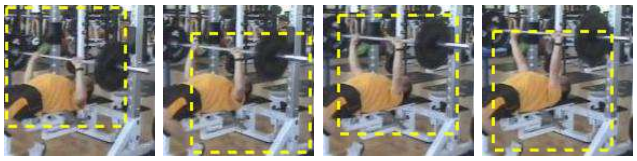
The proposed OPN has three main components: (1) frame feature extraction, (2) pairwise feature extraction, (3) order prediction. Figure 3 shows the architecture in the case of 4-tuple.

Frame feature extraction. Features for each frame ($fc6$) are encoded by convolutional layers. We use a Siamese architecture where all the branches share the same parameters.

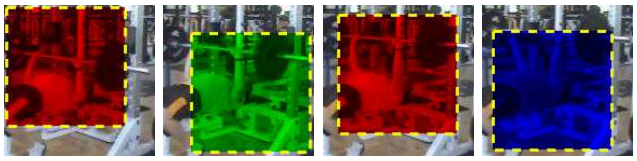
Pairwise feature extraction. A straightforward architecture design for solving the order prediction problem is to concatenate either $fc6$ or $fc7$ features for the frames and use the concatenation as the representation of the input tuple. However, such "taking one glimpse at all frames" approach may not capture the concept of *ordering* well.



(a) Motion-aware tuple selection



(b) Spatial jittering



(c) Channel splitting

Figure 4: **Data sampling strategies.** (a) We use a sliding windows approach on the optical flow fields to extract patches tuple with large motion magnitude. (b)(c) To prevent the network from extracting low-level features for sorting, we apply random spatial jittering and channel splitting as our data augmentation. Quantitative results show that these data augmentation strategies lead to improved performance.

Therefore, inspired from comparison-based sorting algorithms, we propose to perform pairwise feature extractions on extracted features. Specifically, we take the $fc6$ features from every pair of frames for extractions. For example, in Figure 3, the layer7-$(1,2)$ provides information of the relationship of the first and second frames.

Order prediction. The final order prediction is then based on the concatenation of all pairwise feature extractions after one fully connected layer and softmax function.

3.3. Implementation details

We implement our method and conduct all experiments using the Caffe toolbox [16]. We use the CaffeNet [16], a slight modification of AlexNet [18], as our architecture for convolutional layers. For the sake of efficiency, our network takes 80×80 patches as inputs. It dramatically reduces the number of parameters and training time. Our network has only 5.8M parameters up to $fc7$, compared to the 58.2M parameters used in *AlexNet*. As the architecture using feature

concatenation have 9M parameters, the performance gain of OPN does not come from the number of parameters.

We use stochastic gradient descent with a momentum of 0.9 and a dropout rate of 0.5 on fully connected layers. We also use batch normalization [12] on all layers. We extract 280k tuples from the UCF-101 dataset as our training data. To train the network, we set the batch size as 128 and the initial learning rate as 10^{-2}. We reduce the learning rate by a factor of 10 at 130k and 350k iterations, with a total of 200k iterations. The entire training process takes about 40 hours on one Titan X GPU. All the pre-trained models and the source code are available in the project page. [1]

4. Experiments

In this section, we validate the effectiveness of the learned representation. First, we treat our method as an unsupervised pre-training approach to initialize models for action recognition (Section 4.1), image classification, and object detection (Section 4.2). Second, we conduct an ablation study to quantify the contributions from individual components of our approach (Section 4.3). Third, we visualize the low-level filters and high-level activations (Section 4.4). Below we describe the variants of our model:

- **binary:** Order verification similar to [25].
- **3-tuple:** Takes a tuple of 3 frames as input and predicts $3!/2 = 3$ classes.
- **4-tuple:** Take a tuple of 4 frames as input and predicts $4!/2 = 12$ classes.
- **Concat:** Prediction order from the concatenation of $fc6$ features after two fully connected layers.

4.1. Action recognition

We use our approach as a pre-training method on the action recognition datasets. We compare our model with Misra et al. [25] and Fernando et al. [9] which learn features by verifying the order correctness, Purushwalkam et al. [31] which views optical flows features as transformation between RGB features, and Vondrick et al. [40] which applies GAN to generate videos.

Datasets. We use the three splits of the UCF-101 [37] and HMDB-51 [19] action recognition datasets to evaluate the performance of our unsupervised pre-trained network. The UCF-101 dataset consists of 101 action categories with about 9.5k videos for training and 3.5k videos for testing. The HMDB-51 dataset consists of 51 action categories with about 3.4k videos for training and 1.4k videos for testing. We evaluate the classification accuracy on both datasets.

Results. After training with *unlabeled* videos from UCF-101, we fine-tune the model using the labeled videos. Table 1 and Table 2 shows the results on the UCF-101 and HMDB-51 datasets, respectively. Overall, the quantitative results show that more difficult tasks provide stronger semantic supervisory signals and guide the network to learn more meaningful features. The OPN obtains 57.3% accuracy compared to 52.1% of from Vondrick et al. [40] on the UCF-101 dataset. To compare with [31], we also train

[1] http://vllab1.ucmerced.edu/~hylee/OPN/

Table 1: **Mean classification accuracy over the three splits of the UCF-101 dataset**. *Purushwalkam et al. [31] use videos from the UCF-101 (split 1), HMDB-51 (split 1), and ACT datasets for training. †Vondrick et al. [40] use C3D as their architecture. They use videos downloaded from Flickr.

Initialization	CaffeNet	VGG-M-2048
random	47.8	51.1
ImageNet	67.7	70.8
Misra et al. [25]	50.2	-
Purushwalkam et al. [31]*	-	55.4
Vondrick et al. [40]†	52.1	-
binary	51.6	56.8
3-tuple Concat	52.8	57.0
3-tuple OPN	53.2	58.3
4-tuple Concat	55.2	59.0
4-tuple OPN	**56.3**	**59.8**

Table 2: **Mean classification accuracy over the three splits of the HMDB-51 dataset**. Methods with "(UCF)" are pre-trained on the UCF-101 dataset. *Purushwalkam et al. [31] uses videos from the UCF-101 (split 1), HMDB-51 (split 1), and ACT datasets for training.

Initialization	CaffeNet	VGG-M-2048
random	16.3	18.3
Imagenet	28.0	35.3
Misra et al. [25]	18.1	-
Purushwalkam et al. [31]*	-	23.6
binary	20.9	21.0
3-tuple OPN	21.3	21.5
4-tuple OPN	21.6	21.9
Misra et al. [25] (UCF)	15.2	-
4-tuple OPN (UCF)	**22.1**	**23.8**

Table 3: **Comparison with O3N** [9]. The baseline is not the same because O3N uses stacks of frame differences (15 channels) as inputs. To use a similar setting, we take single frame difference (Diff) as inputs and initialize the weights with models trained on RGB and Diff features.

Method	unsupervised	supervised	UCF	HMDB
O3N [9]	Stack of Diff	Stack of Diff	60.3	32.5
OPN	RGB	Diff	**71.8**	36.7
OPN	Diff	Diff	71.4	**37.5**

Table 4: **Results of the Pascal VOC2007 classification and detection datasets.**

Method	Pretraining time	Source	Supervision	Classification	Detection
Krizhevsky et al. [18]	3 days	ImageNet	labeled classes	78.2	56.8
Doerch et al. [7]	4 weeks	ImageNet	context	55.3	46.6
Pathak et al. [30]	14 hours	ImagetNet+StreetView	context	56.5	44.5
Norrozi et al. [27]	2.5 days	ImageNet	context	**68.6**	**51.8**
Zhang et al. [44]	-	ImageNet	reconstruction	67.1	46.7
Wang and Gupta (color) [42]	1 weeks	100k videos, VOC2012	motion	58.4	44.0
Wang and Gupta (grayscale) [42]	1 weeks	100k videos, VOC2012	motion	62.8	**47.4**
Agrawal et al. [2]	-	KITTI, SF	motion	52.9	41.8
Misra et al. [25]	-	$<$ 10k videos	motion	54.3	39.9
Ours (OPN)	$<$ 3 days	$<$ 30k videos	motion	**63.8**	46.9

our model using VGG-M-2048 [34]. Note that the method in [31] uses the UCF-101, HMDB-51 and ACT datasets to train their model (about 20k videos). In contrast, our OPN uses videos from the UCF-101 training set and outperforms [31] by 5.1%. While using 3k videos from the HMDB-51 dataset for both unsupervised and supervised training, OPN performs slightly worse than [31].

We also compare with a recent method for video representation learning [9]. It is difficult to have a fair comparison because they use *stacks of frame differences* (15 channels) as inputs rather than RGB images. To use a similar setting, we take single frame difference $Diff(t) = RGB(t+1) - RGB(t)$ as inputs to train our model. We initialize the network with models trained on *RGB* and $Diff$ features. As shown in Table 3, our method compares favorably against [9] by more than 10% gain on the UCF-101 dataset and 5% on the HMDB-51 dataset. The performance of initializing with the model trained on *RGB* features is similar to with model trained on frame difference. The results demonstrate the generalizability of our model.

We also evaluate the transferability of our learned features. We initialize the weights with the model trained on the UCF-101 training set (without using any labels). Table 2 shows the results. Our method achieves 22.5% compared to 15.2% of [25] under the same setting. We achieve slightly higher performance when there is no domain gap (i.e., using training videos from the HMDB-51 dataset). The results suggest that our method is not heavily data dependent and is capable of learning generalizable representations.

4.2. PASCAL VOC 2007 classification and detection

To evaluate the generalization ability, we use our model as pre-trained weights for classification and detection tasks. We compare our approach with recent image-based and video-based unsupervised learning methods.

Dataset. The PASCAL VOC 2007 [8] dataset has 20 object classes and contains 5,011 images for training and 4,952 images for testing. We train our model using the UCF-101, HMDB-51, and ACT [41] datasets. For both tasks we use the same fine-tuning strategy described in Krähenbühl et al. [17] without the rescaling method. We use the *CaffeNet* architecture and the Fast-RCNN [10] pipeline for the detection task. We evaluate all algorithms using the mean average precision (mAP) [8]. Since our fully connected layers are different from the standard network, we copy only the weights of the convolutional layers and initialize the fully connected layers from a Gaussian distribution with mean 0 and standard deviation 0.005. For a fair comparison with existing work, we train and test our models without using batch normalization layers.

Results. Table 4 lists the summary of methods using static images and method using videos. While our performance is competitive, methods trained with ImageNet performs better than that using videos. We attribute this gap to the fact that the training images are object-centric while our training videos are human-centric (and thus may not contain diverse appearance variations of generic objects). Among the methods using videos, our method shows competitive performance to [42]. However, our method requires considerably less training time and less number of training videos.

4.3. Ablation analysis

We evaluate the effect of various design choices on the split 1 of the UCF-101 dataset. We first perform unsupervised pre-training using the videos from the training set. The learned weights are then used as the initialization for the supervised action recognition problem.

Motion. We select our training tuples according to the magnitude of optical flow. To demonstrate the necessity of this step, we compare it with randomly selecting frames from a video. We also use the optical flow direction as a further restriction. Specifically, the motion in the selected interval must remain in the same direction. Table 5 shows the results of how these tuple selection methods affect the final performance. Using random selection degrades the performance because the training data contain many similar patches that are impossible to be sorted (e.g., static re-

Table 5: **Comparison of different sampling strategies.** *Motion* uses the magnitude of optical flow for patches selection. *Direction* further restricts the monotonicity of optical flow direction in selected tuples. The results show that *Direction* oversimplifies the problems and thus degrades the performance.

Strategy	Action Recognition (%)
Random	47.2
Motion	**57.3**
Motion+Direction	52.6

Table 6: **Comparison of using different patch sizes.** Using 80×80 patches has advantages in all aspects.

Patch size	#Parameters	Training time	Action Recognition (%)
80	**5.8M**	**1x**	**57.3**
120	7.1M	1.4×	55.4
224	14.2M	2.2×	51.9

gions). We also observe that adding the direction constraint does not help. The direction constraint eliminates many tuples with shape deformation (e.g., pitching contains motions in reverse direction). The network thus is unable to learn meaningful high-level features.

Patch size. We experiment with different patch sizes for training the network. Due to the structure of fully connected layers, the patch size selection significantly affects the number of parameters and the training time. Table 6 shows the comparison among using patch size 80×80, 120×120, and the entire image. The results show that using 80×80 patches has an advantage in terms of the number of parameters, training time, and most importantly, the performance. One potential reason for the poor performance of using larger patches might be the insufficient amount of video training data.

Spatial jittering. Analogous to the random gap used in the context prediction task [7] and puzzle-solving task [27], we apply spatial jittering to frames in a tuple to prevent the network from learning low-level statistics. In practice, we apply random shift of $[-5, 5]$ pixels to bounding boxes in both horizontal and vertical directions. Table 7 shows the applying spatial jittering does further help the network to learn better features.

Channel splitting. To further prevent the network from learning trivial features, we reduce the visual clues from color. The most intuitive way is to use the grayscale image. However, grayscale images are generated from a fixed linear combination of the three color channels. To mitigate the effect of color, we randomly choose one representative

Table 7: **Effect of spatial jittering.** For both 3-tuple and 4-tuple cases, OPNs with spatial jittering perform better.

Method	Spatial jittering	Action Recognition (%)
3-tuple OPN		55.8
3-tuple OPN	✓	56.1
4-tuple OPN		56.5
4-tuple OPN	✓	57.3

Table 8: **Effect of pairwise feature extraction on order prediction and action recognition.** The results demonstrate the performance correlation between two tasks, and show that OPN facilitates the feature learning.

Method	Order Prediction (%)	Action Recognition (%)
3-tuple Concat	59	53.4
3-tuple OPN	**63**	**54.1**
4-tuple Concat	38	56.1
4-tuple OPN	**41**	**57.3**

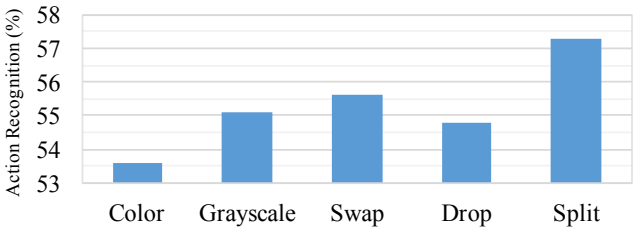
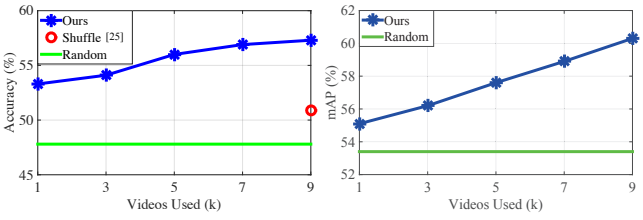


Figure 5: **Effect of different strategies on color channels.** The proposed channel splitting outperforms other strategies.

channel for every frame in a tuple, called *channel splitting (Split)*. We also explore the other two strategies: *Swap* randomly swaps two channels, and *Drop* randomly drops one or two channels. Figure 5 shows the gains of using the proposed channel splitting over other alternative strategies.

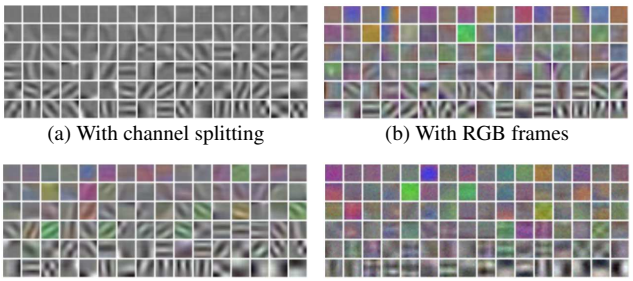
Pairwise feature extraction. We show the effect of the pairwise feature extraction stage as well as the performance correlation between the sequence sorting task and action recognition. We evaluate the order prediction task on a held-out validation set from the automatically sampled data. Table 8 shows the results. For both 3-tuple and 4-tuple, models with the pairwise feature extraction perform better than models with simple concatenation on both order prediction and action recognition tasks. The improvement of the pairwise feature extraction over concatenation is larger on 4-tuple than on 3-tuple due to the increased level of difficulty for the order prediction task.

Number of training videos. We demonstrate the scalability and potential of our method by comparing the performance of using a different amount of videos for unsuper-



(a) Action recognition (b) Classification

Figure 6: **Performance comparison using a different amount of videos**. The results show a steady performance improvement when training with more videos. We also show that the unsupervised pre-training offers significant advantages over random initialization.



(a) With channel splitting (b) With RGB frames

(c) With channel splitting, fine-tuned on UCF-101 (d) Random initialization, trained on UCF-101

Figure 7: **Visualization of Conv1 filters.** Filters in (a)(b) are trained on the UCF-101 dataset in an unsupervised manner. Filters in (c) are fine-tuned from filters in (a) on the UCF-101 dataset with supervision, while filters in (d) are trained from scratch. Note that those "color patch" filters are usually not desirable because they tend to make the further fine-tuning stuck at a bad initialization.

vised pre-training. Figure 6 shows the results on the UCF-101 and the PASCAL VOC 2007 datasets. On the UCF-101 dataset, our approach outperforms [25] using only 1k videos for pre-training. For the classification task on the PASCAL VOC 2007 dataset, the performance consistently improves with the number of training videos. Training with large-scale and diverse videos [1] is a promising future direction.

4.4. Visualization

We demonstrate the quality of the learned features by visualizing low-level first layer filter (conv1) as well as high-level activations (pool5).

Figure 7 shows the visualization of the learned filters in conv1. Figure 7(a) and (b) show that although using all color channels enable the network to learn some color filters, there are many "color patch" filters (see the first two rows in Figure 7(b)). These filters lack generalizability and easily make further fine-tuning stuck at a bad initialization. Comparing Figure 7(c) and (d), the filters are sharper and of more varieties when initialized by our method.

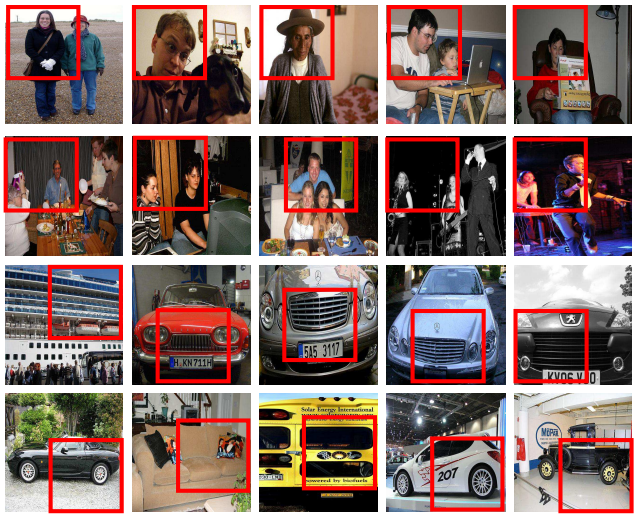


Figure 8: **Activation of Pool5 units.** Each row lists the top 5 patches that activate a specific unit from the VOC dataset. While we train the network on the UCF-101 dataset without using any manual annotations, the pool5 feature activations correspond to human head (1st and 2nd rows) and object parts (3rd and 4th rows).

Figure 8 shows the top 5 activations of several pool5 units on the Pascal VOC 2007 dataset. Although our model is trained on the UCF-101 dataset which focuses on action classes, it captures some meaningful regions without fine-tuning. For example, the first two rows are human-related and the the third and fourth rows capture the front of cars and wheel-like object, respectively.

5. Conclusions

In this paper, we present an unsupervised representation method through solving the sequence sorting problem (sorting a shuffled sequence into a chronological order). We propose an Order Prediction Network architecture to facilitate the training. Using our approach as pre-training, we demonstrate improved performance over state-of-the-art methods on the UCF-101 and HMDB-51 datasets. We also show the competitive generalization ability on classification and detection tasks. While promising results have been shown, there is still a performance gap between the unsupervised pre-training and the supervised pre-training methods. We believe that modeling the long-term evolution in videos (e.g., combining with a recurrent neural network) is a promising future direction.

Acknowledgements

This work is supported in part by the NSF CAREER Grant #1149783, gifts from Verisk, Adobe and NVIDIA.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 8

[2] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015. 1, 3, 6

[3] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. In *NIPS*, 2007. 2

[4] B. Brattoli, U. Büchler, A.-S. Wahl, M. E. Schwab, and B. Ommer. Lstm self-supervision for detailed behavior analysis. 2

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[6] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013. 2

[7] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 1, 2, 6, 7

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 2, 6

[9] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017. 1, 2, 3, 5, 6

[10] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 6

[11] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[13] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Learning visual groups from co-occurrences in space and time. In *ICLR, Workshop*, 2016. 2

[14] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015. 1, 3

[15] D. Jayaraman and K. Grauman. Slow and steady feature analysis: Higher order temporal coherence in video. In *CVPR*, 2016. 2, 3

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014. 4

[17] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2015. 6

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 4, 6

[19] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 2, 5

[20] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016. 1, 2

[21] Q. V. Le. Building high-level features using large scale unsupervised learning. In *ICML*, 2012. 2

[22] D. Li, W.-C. Hung, J.-B. Huang, S. Wang, N. Ahuja, and M.-H. Yang. Unsupervised visual representation learning by graph-based consistent constraints. In *ECCV*, 2016. 2

[23] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *ECCV*, 2016. 3

[24] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*, 2017. 1

[25] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 8

[26] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *ICML*, 2009. 3

[27] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2, 3, 6, 7

[28] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997. 2

[29] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016. 2

[30] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 1, 2, 6

[31] S. Purushwalkam and A. Gupta. Pose from action: Unsupervised learning of pose features based on motion. In *ECCV, Workshop*, 2016. 3, 5, 6

[32] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007. 1

[33] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006. 2

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 6

[35] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012. 2

[36] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *ICCV*, 2005. 2

[37] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 5

[38] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015. 3

[39] J. Sun and J. Ponce. Learning discriminative part detectors for image classification and cosegmentation. In *ICCV*, 2013. 2

[40] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, 2016. 3, 5

[41] X. Wang, A. Farhadi, and A. Gupta. Actions ˜ transformations. In *CVPR*, 2016. 6

[42] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *CVPR*, 2015. 1, 3, 4, 6

[43] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 1, 2

[44] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 1, 2, 6