# Unsupervised Visual Representation Learning by Tracking Patches in Video

Guangting Wang[1] Yizhou Zhou[1] Chong Luo[2] Wenxuan Xie[2] Wenjun Zeng[2] Zhiwei Xiong[1]
University of Science and Technology of China[1] Microsoft Research Asia[2]
{flylight, zyz0205}@mail.ustc.edu.cn {cluo, wenxie, wezeng}@microsoft.com zwxiong@ustc.edu.cn

## Abstract

*Inspired by the fact that human eyes continue to develop tracking ability in early and middle childhood, we propose to use tracking as a proxy task for a computer vision system to learn the visual representations. Modelled on the Catch game played by the children, we design a Catch-the-Patch (CtP) game for a 3D-CNN model to learn visual representations that would help with video-related tasks. In the proposed pretraining framework, we cut an image patch from a given video and let it scale and move according to a pre-set trajectory. The proxy task is to estimate the position and size of the image patch in a sequence of video frames, given only the target bounding box in the first frame. We discover that using multiple image patches simultaneously brings clear benefits. We further increase the difficulty of the game by randomly making patches invisible. Extensive experiments on mainstream benchmarks demonstrate the superior performance of CtP against other video pretraining methods. In addition, CtP-pretrained features are less sensitive to domain gaps than those trained by a supervised action recognition task. When both trained on Kinetics-400, we are pleasantly surprised to find that CtP-pretrained representation achieves much higher action classification accuracy than its fully supervised counterpart on Something-Something dataset.*

## 1. Introduction

During the development of artificial intelligence, we can always take inspiration from the way human brain learns, and computer vision is no exception. For instance, the insight behind building the ImageNet dataset was "to give the algorithms the kind of training data that a child was given through experiences in both quantity and quality."[1] In this work, we intend to address the visual representation learning problem in computer vision, so we look for clues from what developing eyes learn to do in childhood. Our intuition is that once a computer vision system learns what developing eyes are capable of, the visual features it extracts should contain the most important information needed by downstream vision tasks.

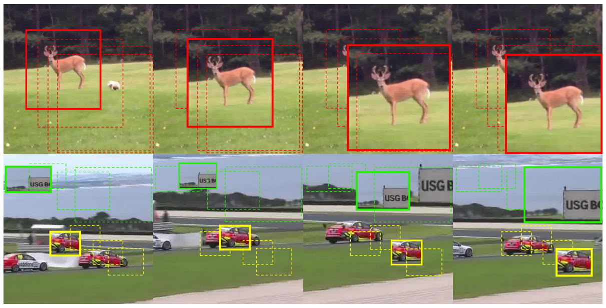

Figure 1: Illustration of the Catch-the-Patch game we designed to train a computer vision system. We randomly crop one or multiple patches from a video clip, let them scale and move in a smooth way, and then train the neural network to predict the positions and sizes of the patches in each frame.

It is not surprising that the ability to track, or to follow a moving target, caught our attention. It is not only an important capability of human eyes, but it has also been regarded as an important technology in computer vision and the basis of video analysis. In this work, however, we do not treat tracking as an ultimate task. Instead, we want to use it as a proxy task for a computer vision system to learn feature representations of visual signals. Here, the visual signals need to be videos, or moving pictures, instead of static images. Ideally, the learning process does not require human annotation, or should be self-supervised. Only in this way can we make full use of the large amount of video data on the Internet. This falls into an active area of research called self-supervised video representation learning, which aims to learn video understanding models [37, 51, 38, 50] without access to human annotation.

The research progress in this area lags far behind a closely related area called self-supervised image representation learning, where several ground-breaking works [14, 3] emerged in recent years. A possible reason is that videos are much larger in size and more redundant in its original

[1] Fei-fei Li's TED talk "How we teach computers to understand pictures," 2005.

representation than images. It is therefore more critical to design an efficient proxy task which could guide the neural network to acquire the core capability or to distill the most important information. Some existing proxy tasks propose to estimate the orientation of video frames [20], to predict the spatial-temporal order [22, 44], or to estimate the playback speed of the input video clip [1, 47, 4]. These tasks may fail to capture the fine-grained information as they only care about the coarse global attributes.

Our work focuses on helping the network develop the ability to follow a moving target. The proxy task used to pretrain the network is inspired by the training of human vision system. It is well-known that the *Catch* game can help children develop their visual tracking abilities. For computers, we want to design a similar game. It would be ideal if we could throw all kinds of realistic objects with various appearance into the videos, but it is hard to implement. So, we step back and cut a patch from the existing video and let it change and move in the way we have pre-set it. The pretraining objective is to predict the location and size of this patch in all input frames given only the patch information in the initial frame. We call this game *Catch-the-Patch* (CtP).

Although the concept of the game is simple, it is not an easy task to design the details for the best pretraining results. We find that throwing more than one image patch, changing and moving in different patterns, in a video at the same time brings clear benefits. In addition, making image patches invisible, or disappear, from time to time can further exercise the network's ability to associate adjacent frames. We call this masked region model (MRM). We train an R3D network [38] and an R(2+1)D network [38] using our invented CtP game and apply CtP-pretrained video representation to two downstream tasks, namely action recognition and video clip retrieval.

Experimental results show that CtP significantly outperforms existing proxy tasks in video representation learning. On UCF-101 dataset, our CtP-pretrained R3D model [38] achieves 86.2% top-1 classification accuracy. It outperforms the most advanced method TempTrans [19] by 6% absolute gains. Furthermore, for datasets like Something-something-V1 which require more temporal relationship mining, CtP-pretraining leads to a 48.3% top-1 accuracy. Surprisingly, it even surpasses the fully supervised counterpart (44.1%) by a notable margin. To summarize, the contributions of this work are three-fold:

- Inspired by the Catch game which helps children develop their eyes, we design a Catch-the-Patch game for neural networks to learn visual features from videos.
- We scientifically design the details of the game, including using more than one patches for training and introducing the MRM. These designs have been carefully validated by ablations studies.
- We carry out comprehensive evaluation of the proposed method. CtP pretraining not only achieves state-of-the-art results for standard downstream tasks, but also closes the performance gap between unsupervised and supervised video representation learning.

## 2. Related Work

Our work is about learning visual representations from videos, so we first review a group of most related work called unsupervised video representation learning in Section 2.1. Image representation learning is not involved here, as they have a very different problem setting from ours. In our proposal, tracking is used as a proxy task, but it is different from the object tracking task that computer vision researchers are familiar with. Therefore, we spend some paragraphs in Section 2.2 to discuss the connections and differences. Last, strictly speaking, the training data we used are synthetic. Can synthetic data help us achieve efficient training? We tend to have a positive answer after reviewing some related papers in Section 2.3.

### 2.1. Unsupervised video representation learning

Research works in this area fall into one of the two categories: transformation-based methods and contrastive-learning-based methods.

The central idea of transformation-based methods is to construct some transformations so that video representation models can be trained to recognize those transformations. Typical transformations include image rotation [20], spatial shuffling [22], temporal shuffling [44], and speed change [1, 47, 4]. Some approaches also leverage multiple transformations to improve performance. For example, VCP [27] uses rotation degree and shuffling order as supervision signals. TempTrans [19] integrates a set of temporal transformations including speed change and random shuffling.

The other major category is contrastive learning [11, 12, 6, 40, 13, 45, 46, 36], which has been proven effective in many other domains like image [14] and speech [29] pretraining. In general, contrastive learning aims at discriminating positive and negative pairs. The definition of "positive" and "negative" pairs varies in different methods. For instance, VideoPace [40] adopts the speed attribute as the condition to assign positive and negative labels. DPC and its follow-up work [11, 12] introduces a future prediction module. The predicted future features and corresponding ground-truth future features are considered positive, while the rests are negatives. In addition to the label assigning, there is also related work that constructs training pairs between two different modalities, such as RGB-flow pair [13] and audio-visual pair [28].

There is no conclusion yet as to which category is better over the other, but our work falls into transformation-based methods. A major characteristic that differentiate our work from the other works in the same category is that the transformation is applied to local regions instead of the entire frame or clip. This design guides the neural network to learn region-level temporal correspondence, which we believe is the basic information for most downstream tasks.

### 2.2. Visual tracking

Visual tracking is one of the fundamental research tasks in computer vision. There is a large body of research work that addresses both single object tracking and multiple object tracking. However, these methods are beyond the scope of this paper, as we are not trying to solve the tracking problem. Instead, we are using tracking as a proxy task to learn video representations. It should be mentioned that it is considered legitimate to leverage some non-data-driven tracking methods [17] to provide pseudo ground truth of visual tracking in self-supervised learning.

There are some related works [39, 43, 41, 25, 24, 42] which use tracking as a proxy task to learn image representations. These works train a two-dimensional (2D) backbone network to extract features from a single frame. Tracking is performed between consequent frames. Since representation learning prefers an unsupervised approach to a supervised one, these works also avoid from accessing human annotations. Vondrick et al. [39] assume that the color information of a region is temporally stable, so they propose to estimate the corresponding positions based on the coherency of colors in the video. Wang et al. [43] and UDT [41] introduce a cycle-consistency constraint. After a few steps of forward tracking and then the same number of steps of backward tracking, the predicted location of the target should be close to the starting point.

Our work is different from these image representation learning methods. Again, there is no conclusion yet whether visual representation should be learned from images or videos, but these two camps have very different problem settings and evaluation processes. The most notable difference is that video representation learning takes a video clip, or a sequence of frames, as input and trains a 3D backbone.

### 2.3. Training with synthetic data

To meet the high demand for training data from machine learning algorithms, people produce synthetic data through various approaches, including game engines [31, 9], 3D models [8], and generative models[33]. It has been proven that synthetic data play an important role in model pretraining [9, 8]. The way we throw image patches into a video to create synthetic training data is similar to the idea behind Flying Chairs dataset [8]. This work overlays a chair, which is generated by a 3D model with pre-set movement, on a real image. Our work overlays a cropped image patch on a real video, since 3D models are more expensive and are not able to cover all types of targets.

Our work is also related to a category of work that treats synthetic data as a regularization term. Typical work includes MixUp [49], CutOut [7] and CutMix [48]. The key idea of these works is to add some constraints over the input data and labels by human priors. For example, MixUp assumes that if an image is mixed with another one, the new ground-truth label should also be a weighted combination of two original labels. Our pretraining method can also be viewed as a regularization term to the vanilla video representation model, which forces the model to encode the foreground objects' movements.

## 3. Catch-the-Patch Learning Framework

*Catch*, or *playing catch*, is one of the most basic children's games. The participants throw a ball, a beanbag, or a frisbee back and forth to each other. In early and middle childhood, this game helps human vision system develop the tracking ability. Now, we design the game *Catch-the-Patch* for computers to develop the tracking ability. This is used as a proxy task for neural networks to learn video representations. In this section, we first provide a framework overview in Section 3.1. Then, the design details are illustrated in Section 3.2. Last but not least, we discuss how to acquire proper self-supervision signals to facilitate effective learning in Section 3.3.

### 3.1. Framework Overview

In computer vision, tracking is to locate a specified target in a video clip given the bounding box of the target in the initial frame. Target locations are usually represented by upright rectangular bounding boxes. Target locations in a sequence of frames form a tracking trajectory. In this work, we use $\boldsymbol{B}$ to denote a ground-truth trajectory:

$$\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_T],$$

where $T$ is the total number of video frames and $\boldsymbol{b}_i$ denotes the target bounding box in the $i$-th frame.

The goal of our work is to train a general video representation model $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. The model $f_{\boldsymbol{\theta}}$ receives a video clip $\mathbf{x}$ as input and extracts the spatial-temporal features $\mathbf{v}$, which can be formulated as $\mathbf{v} = f_{\boldsymbol{\theta}}(\mathbf{x})$

In order to enable this representation to encode the information of object trajectory, we introduce a dedicated prediction head that estimates the tracking trajectory based on the extracted video representation:

$$\hat{\boldsymbol{B}} = h_{\boldsymbol{\phi}}(\mathbf{v}, \boldsymbol{b}_1),$$

where $h$ is the prediction head and $\boldsymbol{\phi}$ is the learnable parameters. Conceptually, the function $h_{\boldsymbol{\phi}}$ takes a bounding box on the starting frame $\boldsymbol{b}_1$ as query and predicts the entire corresponding trajectory $\hat{\boldsymbol{B}} = [\hat{\boldsymbol{b}}_i]_{i=1}^{T}$. Under this formulation, we can naturally apply the ground-truth trajectory $\boldsymbol{B}$ to

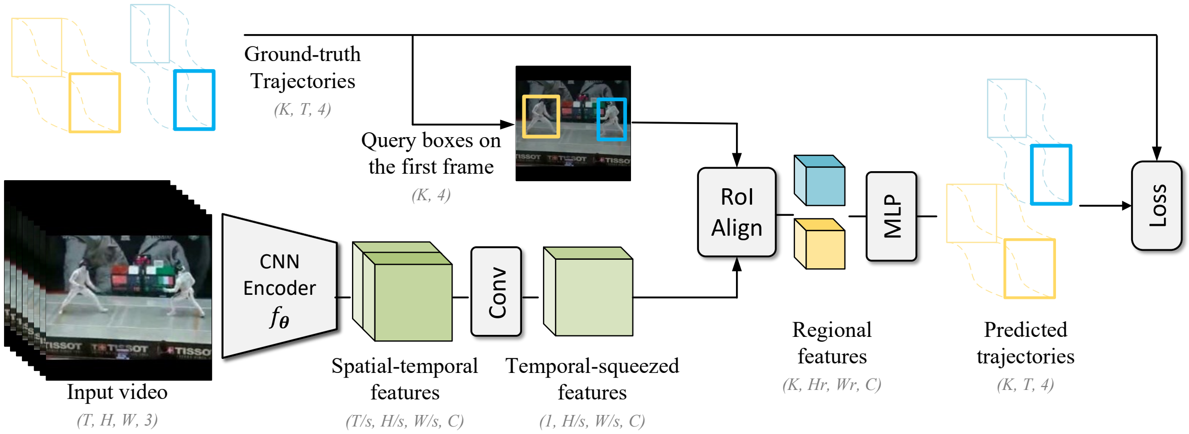

Figure 2: Illustration of the proposed Catch-the-Patch learning framework. The input videos are synthetic ones with overlaid image patches. The CNN encoder $f_{\boldsymbol{\theta}}$ is the 3D model we intend to pretrain. Given the initial locations of the image patches, the rest of the network is expected to predict the entire trajectories based on the extracted spatial-temporal features from $f_{\boldsymbol{\theta}}$.

supervise the training of video representation model $f_{\boldsymbol{\theta}}$. Assume that we have $M$ video clips $\{\mathbf{x}_i\}_{i=1}^{M}$ in the dataset and each video clip has $K$ ground-truth trajectories $\{\boldsymbol{B}_i^{(j)}\}_{j=1}^{K}$. The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are jointly optimized under the loss function:

$$\mathcal{L} = \frac{1}{MK}\sum_{i=1}^{M}\sum_{j=1}^{K} d(\boldsymbol{B}_i^{(j)}, \hat{\boldsymbol{B}}_i^{(j)}),$$

where $d$ is a predefined distance metric that measures how far the predicted trajectory is from the ground-truth.

### 3.2. Design details

The proposed framework is composed of three components: a video representation model $f_{\boldsymbol{\theta}}$, a prediction head $h_{\boldsymbol{\phi}}$ and a distance metric $d$. In this section, we will instantiate each component.

The detailed architecture is illustrated in Fig. 2. Generally speaking, the video representation model $f_{\boldsymbol{\theta}}$ can be a typical convolution neural network (CNN) encoder designed for video analysis tasks, such as C3D [37], R3D [38] or TSM [26]. The CNN encoder contains some temporal modules that establish relationships among video frames. The receptive field of the encoded spatial-temporal features is wide enough to cover the entire video clip, which makes it possible to explore the temporal correspondences for any queries.

By design, our prediction head $h_{\boldsymbol{\phi}}$ receives two inputs: the spatial-temporal features of the input video clip and the bounding box query on the starting frame. We adopt the RoI Align operation [15] to associate these two inputs. RoI Align can crop the features in the given bounding boxes and encode them into a fixed size tensor. Before this operation, we use a 3D convolution layer to squeeze the temporal dimension of the spatial-temporal features. The convolution layer has a spatial kernel size of $1 \times 1$, and the temporal kernel size is the same as the temporal dimension size of the input features. This layer is motivated by the bottleneck design in ResNet [16]. We compress the temporal dimension and try to recover the entire trajectory from it.

After extracting regional features for each query bounding box, a two-layer multilayer perceptron (MLP) network is adopted to produce a vector of size $T \times 4$, where $T$ is the number of input frames. This vector encodes the relative deformation between the predicted trajectory and the query bounding box. Formally, we represent a bounding box $\boldsymbol{b}$ by a quadruple $(x, y, w, h)$, where $(x, y)$ is the center coordinates and $(w, h)$ is the spatial dimensions. The predicted corresponding box in the $i$-th frame $\hat{\boldsymbol{b}}_i$ can be written as:

$$\begin{aligned} \hat{x}_i &= x_1 + \sigma_x \hat{t}_{i,1} & \hat{y}_i &= y_1 + \sigma_y \hat{t}_{i,2} \\ \hat{w}_i &= w_1 \exp\left(\sigma_w \hat{t}_{i,2}\right) & \hat{h}_i &= h_1 \exp\left(\sigma_h \hat{t}_{i,3}\right) \end{aligned}$$

where $(x_1, y_1, w_1, h_1)$ is the query bounding box in the starting frame, $\hat{\boldsymbol{t}}_i$ is the estimated targets for the $i$-th frame and $\boldsymbol{\sigma}$ is a set of constant scaling factors. In this work, $(\sigma_x, \sigma_h, \sigma_w, \sigma_h)$ are set to $(0.8, 0.8, 0.04, 0.04)$.

Following the common practice in object detection [30], the distance function $d$ is defined in linear space for the center coordinates and log space for the spatial dimensions. Given the ground-truth bounding box $(x_i, y_i, w_i, h_i)$, we use Smooth-L1 function $L$ to calculate the distances:

$$\begin{aligned} d(x_i, \hat{x}_i) &= L(\frac{x_i - \hat{x}_i}{\sigma_x}) & d(y_i, \hat{y}_i) &= L(\frac{y_i - \hat{y}_i}{\sigma_y}) \\ d(w_i, \hat{w}_i) &= L(\frac{1}{\sigma_w}\log\frac{w_i}{\hat{w}_i}) & d(h_i, \hat{h}_i) &= L(\frac{1}{\sigma_h}\log\frac{h_i}{\hat{h}_i}) \end{aligned}$$

Compared with the CNN encoder $f_{\boldsymbol{\theta}}$, the prediction head $h_{\boldsymbol{\phi}}$ is light-weight. For instance, an R3D-18 CNN encoder accounts for more than 82% of the parameters in the entire framework. Hence, the power of visual tracking mainly lies in the CNN encoder. After training, the learned encoder can be applied to various downstream tasks such as video recognition, tagging, and retrieval.

### 3.3. Synthetic data generation

Ideally, visual representation should be learned from real objects and real trajectories. However, it is impossible to annotate the trajectories of countless objects in a huge number of videos. This is the reason why we step back and resort to synthetic data sets.

In order to create realistic training data, we design a three-step process for data generation. First, we randomly generate a pseudo trajectory that simulates the object movement. To ensure smoothness, we first determine the bounding boxes in some key frames. The trajectory positions in the rest frames are linearly interpolated between two neighboring key frames. Second, we randomly select one bounding box from the pseudo trajectory and copy the image patch from the video frame. Finally, the copied image patch is scaled and overlaid on all original video frames according to the pseudo trajectory. When we track the copied image patch, the pseudo trajectory provides the ground-truth. We repeat this process multiple times so that each training video clip will have multiple targets and corresponding ground-truth trajectories. These targets may or may not overlap. Later, we will show through experiments that this design significantly improves the pretraining quality.

To strength the awareness of temporal relationships, we further introduce a masked region model (MRM), which is inspired by the masked language model in BERT [5]. When constructing synthetic videos, the simulated patch will be randomly masked out in some frames with a probability of 0.2. Although the masked patch is invisible in these frames, the model is still compelled to predict the virtual locations. It encourages the pretrained model to exploit the temporal context information in successive frames.

## 4. A Deep Dive into the Proxy Task

CtP framework has adopted synthetic data for the training of proxy task. A natural concern is whether it has really learned how to track in real videos. To figure it out, we conduct an experiment to compare CtP-pretrained model with two baselines. One is a randomly initialized model, and the other is a 3D model inflated from a 2D model pretrained on ImageNet [2]. The latter represents a model that has learned visual representations from images. We do not compare our model with standard trackers due to the big difference in problem setting. For instance, standard trackers perform tracking frame by frame, by comparing the visual features generated from 2D models. The tracking target is scaled to a fairly large resolution, and the search region is adjusted per frame according to the tracking result in the previous frame. In our case, we evaluate a 3D model by directly providing 16 sequential frames. The tracking results in these 16 frames are obtained in a single forward pass.

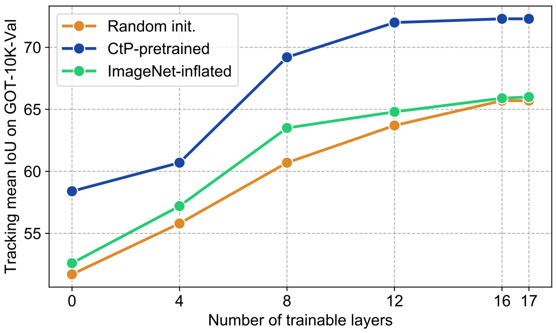

Figure 3: Tracking performance evaluation of a 17-layer R3D model under different training strategies.

We use one recently proposed large-scale tracking dataset, called GOT-10k [18], to conduct the evaluation. Specifically, we fine-tune CtP-pretrained model and two reference models using the GOT-10k training set and then evaluate them on the validation set. In the fine-tuning process, we experiment with multiple settings that freeze a different portion of parameters in the pretrained model. The evaluation metric is the mean interaction-over-union (mIoU) value between the predicted trajectory and the ground-truth. Fig. 3 shows the results. CtP-pretrained model achieves significant gain over the reference models in all the fine-tuning configurations. This set of experiments verifies that our model does learn features that were beneficial to object tracking through the designed CtP game. At the very least, it provides an excellent initialization for tasks that care about object motion.

## 5. Experiments

### 5.1. Implementation details

**Model** We adopt the standard R3D-18 and R(2+1)D-18 [38] as the video representation models, following the common practice in previous research [44, 27, 47, 19]. After the CNN encoder, the RoI Align operation produces several regional features with a spatial size of $5 \times 5$. The channel size of the MLP prediction head is 512. When transferring to the downstream tasks, we only adopt the pretrained weights of the CNN encoder and the prediction head is dropped.

**Pretraining data** Most of the evaluated models are pretrained on UCF-101 [34] (shorted as “UCF”) [2] and Kinetics-400 datasets [21] (shorted as “K400”). During pretraining, the temporal length of input videos is 16, with a frame interval ranging from 1 to 5. For each video clip, we generate three independent ground-truth trajectories. The

[2] We use training split 1 from the official splits of UCF-101 dataset.

spatial resolution of the input clip is $112 \times 112$, and the size of the cropped patch is uniformly sampled from $16 \times 16$ to $64 \times 64$. The maximum speed of the trajectory is limited to 3 pixels per frame, and the scale ratio of the bounding boxes in two successive frames should fall in $[e^{-0.025}, e^{0.025}]$.

**Optimization** The pretraining process lasts for 300 epochs on UCF and 90 epochs on K400. We adopt a standard stochastic gradient descent (SGD) algorithm to optimize the training objective function. The initial learning rate is $0.01$, which is decayed by a factor of $0.1$ at 100th and 200th epoch (30th and 60th for K400), respectively. The optimizer momentum is $0.9$ and the weight decay is $10^{-4}$. During training, the total batch size is 32. Using 8 NVIDIA V100 GPU cards, training takes about 4 hours to finish on UCF and 2 days on K400.

**Evaluation** We evaluate the learned video representations in two downstream tasks: action recognition and video clip retrieval. For action recognition, we append a one-layer linear classifier after the CNN encoder. The entire model is then fine-tuned on the target dataset for 150 epochs. In our experiments, we perform evaluation on UCF, HMDB-51 [23] (shorted as "H51"). For video clip retrieval, we follow the same implementation as in VCOP [44]. In both tasks, the performance is evaluated by top-$k$ accuracy. If the class of a test video appears in its $k$ most confident predictions (or its $k$ nearest training clips for retrieval task), it is considered to be a correct classification (or retrieval).

## 5.2. Sources of ground-truths

We have used synthetic training data in this work. The advantage is that the target trajectory is pre-set and therefore the ground-truth is always accurate and precise. The disadvantage, however, is that the pictures are not real. Apparently, there is another option to use real pictures but less precise ground-truth. In this section, we discuss and compare some alternatives to obtain pseudo ground-truth.

**Teacher tracker.** One option is to generate pseudo labels by a teacher tracker. Although there have emerged many advanced trackers in the deep learning era, we choose one of the best non-data-driven trackers, named KCF [17], in this experiment. KCF relies on handcrafted features, so it fits into the setting of unsupervised learning.

**Cycle consistency.** Cycle consistency constraint is adopted by some tracking-based unsupervised image representation learning methods [43, 25]. We can also make use of this constraint to generate ground-truths. A simple implementation is to use the inverse trajectory of the backward tracking as the ground-truth of forward tracking. However, using this strategy alone may lead to a trivial solution of static trajectory. Therefore, we use a mixed solution where a quarter of the training labels are sourced from the synthetic data, and the rest is from backward tracking.

**Real annotations.** There exist some annotated datasets for visual object tracking. We find that GOT-10k dataset [18] is fairly large with a variety of object classes. The ground-truth labels are considered accurate, but one disadvantage is that there is only one annotated object in each video. Besides, the movement of non-object is not annotated throughout the dataset.

Table 1: Ablation analysis of different ways to acquire ground-truths. In this experiment, we adopt R3D-18 models transferred to the action recognition task.

| Pretraining | Source of | Top-1 Acc. (%) | |
|---|---|---|---|
| dataset | ground-truths | UCF | H51 |
| None | - | 65.0 | 30.9 |
| UCF-101 | Teacher tracker | 70.6 | 44.4 |
| UCF-101 | Cycle consistency | 81.1 | 53.3 |
| UCF-101 | Pre-set (×3) | 83.9 | 53.6 |
| GOT-10k | Real annotation | 77.0 | 51.4 |
| GOT-10k | Pre-set (×1) | 80.4 | 49.4 |
| GOT-10k | Pre-set (×3) | **85.9** | **55.7** |

We train R3D-18 models using these different sources of ground-truths. The learned video representations are transferred to the action recognition task on UCF and H51 datasets, as specified in the evaluation protocol 5.1.

Table 1 presents the results. It is not surprising that all pretrained models achieve better performance than a randomly initialized model. When pretrained on UCF, using pre-set trajectories achieves significantly better results than the other two choices. It suggests that accurate ground-truths might be more crucial than realistic pictures for the tracking proxy task.

We also compare between using synthetic data with pre-set trajectories and real data with human annotations. Since there is only one annotated object in each video, we also tested a setting that only uses one image patch per video, which is denoted by "Pre-set ×1". We are surprised to find that even with this setting, there is no disadvantage in using synthetic data. The good performance may attribute to the fact that both the image patches and their trajectories can change in every training epoch. Furthermore, if we use more pre-set trajectories per video (3 as default), the performance is even better.

## 5.3. Ablation analysis

We analyze several design choices in the Catch-the-Patch framework. For time efficiency, models evaluated in this subsection are trained on a subset (about 25%) of K400. The pretrained models are transferred to the action recognition task on UCF [34] and H51 [23].

**Number of pre-set trajectories.** For each video clip, we generate different numbers of pre-set trajectories. The

Table 2: Ablation analysis of our proposed CtP. The model is pretrained on a subset of K400 and transferred to the action recognition task.

(a) Num of trajectories.

| Num of trajs | Top-1 Acc. (%) |  |
|---|---|---|
|  | UCF | H51 |
| 1 | 81.7 | 49.1 |
| 2 | 83.0 | 51.9 |
| 3 | 82.8 | 53.2 |
| 4 | 82.2 | 54.1 |

(b) Masked region model.

| Backbone | MRM | Top-1 Acc. (%) |  |
|---|---|---|---|
|  |  | UCF | H51 |
| R3D |  | 82.8 | 53.2 |
| R3D | ✓ | **84.0** | **55.3** |
| R(2+1)D |  | 85.1 | 55.9 |
| R(2+1)D | ✓ | **87.2** | **57.8** |

Table 3: Comparison with baseline pretraining approaches. We report the top-1 accuracy of transferred video representation models on UCF-101, HMDB-51 and Something-Something-V1 (SS) datasets.

| Backbone | Pretraining |  | Top-1 Acc. (%) |  |  |
|---|---|---|---|---|---|
|  | Method | Dataset | UCF | H51 | SS |
| R3D | None | None | 65.0 | 30.9 | 39.2 |
| R3D | Supervised | ImageNet | 79.5 | 40.0 | 42.9 |
| R3D | Supervised | K400 | **91.6** | **60.5** | 43.3 |
| R3D | CtP | K400 | 86.2 | 57.0 | **44.2** |
| R(2+1)D | None | None | 67.0 | 29.5 | 40.6 |
| R(2+1)D | Supervised | K400 | **92.7** | **64.5** | 43.9 |
| R(2+1)D | CtP | K400 | 88.4 | 61.7 | **48.3** |

ablation results are presented in Table 2 (a). As the number of trajectories increases from 1 to 2, the top-1 accuracy of the transferred model is significantly improved. We believe that, under single-trajectory supervision, the model tends to learn only a global motion. Using multiple trajectories simulates a situation where each region can have its own motion state. The extracted features should contain enough information for the tracking head to simultaneously capture the motion and distinguish between different trajectories. When the number of trajectories exceeds 3, the performance starts to saturate. Therefore, we use 3 trajectories in the rest of our experiments.

**Masked region model.** When generating synthetic videos, some overlaid patches are masked out with a random probability. To predict the virtual positions of masked patches, the model needs to exploit the temporal context information. In Table 2 (b), we present the experimental results of training with and without MRM. It clearly shows that MRM helps to improve the video representation learning in both R3D and R(2+1)D backbones.

## 5.4. Comparison with baseline approaches

We compare our pretraining method with two baseline approaches: random initialization and supervised pretraining on Kinetics or ImageNet. After supervised pretraining on ImageNet, we inflate the 2D convolutional kernels to 3D as in I3D [2]. The results of the action recognition task are presented in Table 3.

Table 4: Comparison with state-of-the-art video representation learning approaches. The downstream task is action recognition on UCF-101 and HMDB-51 datasets. The column "Arch." denote the input spatial resolution and the encoder architecture. The mark † means that the results are produced by our re-implementation.

| Method | Dataset | Arch. | Top-1 Acc. (%) |  |
|---|---|---|---|---|
|  |  |  | UCF | H51 |
| DPC [11] | K400 | R-2D3D | 75.7 | 35.7 |
| CBT [35] | K600 | S3D | 79.5 | 44.6 |
| MemDPC [12] | K400 | R-2D3D | 78.1 | 41.2 |
| SpeedNet [1] | K400 | S3D-G | 81.1 | 48.8 |
| CEP [46] | K400 | SlowFast | 77.0 | 36.8 |
| CoCLR [13] | K400 | S3D | 87.9 | 54.6 |
| VCP [27] | UCF | R3D | 66.0 | 31.5 |
| PRP [47] | UCF | R3D | 66.5 | 29.7 |
| TempTrans [19] | UCF | R3D | 77.3 | 47.5 |
| Ours | UCF | R3D | **83.9** | **53.6** |
| TempTrans [19] | K400 | R3D | 79.3 | 49.8 |
| MoCo †[14] | K400 | R3D | 77.0 | 43.4 |
| VCOP †[44] | K400 | R3D | 73.3 | 41.4 |
| 3DRotNet †[20] | K400 | R3D | 77.5 | 41.4 |
| MemDPC †[12] | K400 | R3D | 75.3 | 41.2 |
| SpeedNet †[1] | K400 | R3D | 83.5 | 50.6 |
| Ours | K400 | R3D | **86.2** | **57.0** |
| Pace [40] | UCF | R(2+1)D | 75.9 | 35.9 |
| VCOP [44] | UCF | R(2+1)D | 72.4 | 30.9 |
| VCP [27] | UCF | R(2+1)D | 66.3 | 32.2 |
| PRP [47] | UCF | R(2+1)D | 72.1 | 35.0 |
| TempTrans [19] | UCF | R(2+1)D | 81.6 | 46.4 |
| Ours | UCF | R(2+1)D | **86.2** | **57.1** |
| Pace [40] | K400 | R(2+1)D | 77.1 | 36.6 |
| Ours | K400 | R(2+1)D | **88.4** | **61.7** |

CtP Pretraining achieves much higher accuracy than random initialization and ImageNet pretraining on all three datasets. On UCF-101, CtP-pretrained R3D-18 model gets an absolute gain of 20.4% over the random baseline and 5.9 % over the ImageNet baseline. Unsurprisingly, the model pretrained with action recognition labels on K400 achieves the highest accuracy on UCF and H51. It is encouraging that the performance gap between our model and this K400-supervised model is not large. Interestingly, when both models are evaluated on the Something-Something [10] dataset, our model achieves a better performance. This may due to the fact that accurate classification of the fine-grained actions in SS relies on the quality of local features,

Table 5: Comparison with state-of-the-art video representation learning approaches in video clip retrieval task. In this experiment, we use R3D-18 as the CNN encoder.

| Method | Dataset | UCF Acc. (%) | | H51 Acc. (%) | |
|---|---|---|---|---|---|
| | | Top-1 | Top-5 | Top-1 | Top-5 |
| VCOP [44] | UCF | 14.1 | 30.3 | 7.6 | 22.9 |
| VCP [27] | UCF | 18.6 | 33.6 | 7.6 | 24.4 |
| PRP [47] | UCF | 22.8 | 38.5 | 8.2 | 25.8 |
| Pace [40] | UCF | 19.9 | 36.2 | 8.2 | 24.2 |
| Ours | UCF | **23.4** | **40.9** | **11.4** | **30.2** |
| SpeedNet [1] | K400 | 13.0 | 28.1 | - | - |
| TempTrans [19] | K400 | 26.1 | **48.5** | - | - |
| Ours | K400 | **29.0** | 47.3 | **11.8** | **30.1** |

which is the advantage of our method.

### 5.5. Comparison with state-of-the-art approaches

Following common practices, we compare our method with the state-of-the-art (SOTA) approaches by transferring the learned representations to two downstream tasks.

**Action recognition.** The evaluation results are presented in Table 4. It should be noted that the finetuning settings, including input resolution, training epochs, and data augmentations, can dramatically affect the final accuracy. Unfortunately, there is no standard setting exists. In order to present an apple-to-apple comparison, we have tried our best to integrate the existing open-sourced work with our finetuning pipeline (marked as † in Table 4).

Overall, the CtP learning framework significantly outperforms existing approaches under the same training configurations. For example, when an R3D-18 encoder is pretrained on K400 dataset, CtP improves the very recent approach SpeedNet [1] by an absolute gain of 2.7 % on UCF-101. Meanwhile, we also benchmark MoCo [14], a representative method designed for the image representation learning, on the action recognition task. Experimental results demonstrate that it cannot work well for video. Compared with other methods trained with different architectures or resolutions, our method achieves a vastly higher accuracy of 88.4 % on UCF-101 and 61.7 % on HMDB-51.

**Video clip retrieval.** We use the exact same evaluation protocol as in VCOP [44] and report the retrieval accuracy on UCF-101 and HMDB-51 datasets. The results in Table 5 clearly shows that our pretraining method achieves superior performances on both datasets.

### 5.6. Data efficiency

We plot the data efficiency curve in Fig. 4. Two models are trained with different percentages of labeled data on UCF-101. One is initialized by CtP-pretrained representations while the other is trained from scratch. The advantage of CtP-pretraining is more significant when there are fewer number of labeled data. Notably, under the help of pretraining, with only 20% of the labeled data, we can achieve a similar performance as a randomly initialized classifier trained on the entire labeled dataset.

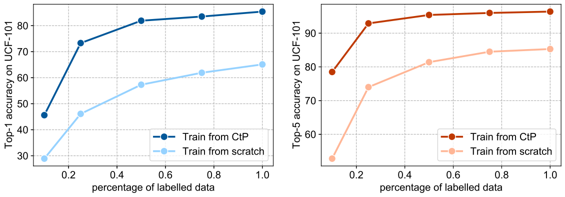

Figure 4: Data efficiency of representations. The pretraining is conducted on K400 dataset.

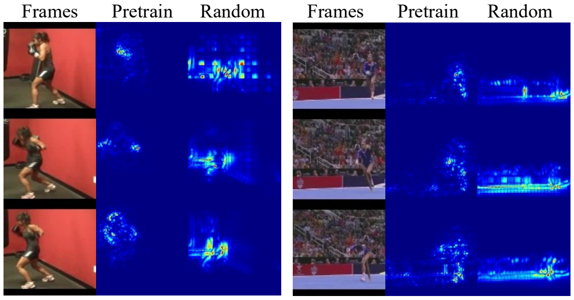

Figure 5: Visualization of important pixels by GradCAM

### 5.7. Visualization

We use guided GradCAM [32] to highlight the important pixels that contribute to the final classification decision. The classifier is pretrained and then finetuned on UCF-101. We also visualize the results of the classifier trained from scratch. Fig. 5 shows that pretrained classifier successfully captures the salient regions, while the results of the random baseline are noisy, especially for complex scenes. This further verifies that our pretraining enhances the network's ability to follow moving targets.

## 6. Conclusion

In this paper, we have proposed *Catch-the-Patch* learning framework which uses tracking as a proxy task to learn video feature extraction. It is an unsupervised framework without access to any human annotations. Comprehensive experiments have proved the rationality and effectiveness of our approach. In the future, we plan to explore semi-supervised approaches and design a bootstrap process to create more realistic training data. We also plan to train the model using the sheer amount of video data on the Internet. After all, that is what unsupervised learning is all about.

## Acknowledgement

The authors would like to express their appreciation to Xiaoyi Zhang for the inspiring discussions.

## References

[1] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos. In *CVPR*, pages 9922–9931, 2020. 2, 7, 8

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017. 5, 7

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 1

[4] Hyeon Cho, Taehoon Kim, Hyung Jin Chang, and Wonjun Hwang. Self-supervised spatio-temporal representation learning using variable playback speed prediction. *arXiv preprint arXiv:2003.02692*, 2020. 2

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 5

[6] R Devon Hjelm and Philip Bachman. Representation learning with video deep infomax. *arXiv preprint arXiv:2007.13278*, 2020. 2

[7] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 3

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 3

[9] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. 3

[10] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 5, 2017. 7

[11] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *ICCV Workshop*, 2019. 2, 7

[12] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. *arXiv preprint arXiv:2008.01065*, 2020. 2, 7

[13] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. *arXiv preprint arXiv:2010.09709*, 2020. 2, 7

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020. 1, 2, 7, 8

[15] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. *T-PAMI*, 42(2):386–397, 2020. 4

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 4

[17] Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *T-PAMI*, 37(3):583–596, 2015. 3, 6

[18] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*, 2019. 5, 6

[19] Simon Jenni, Givi Meishvili, and Paolo Favaro. Video representation learning by recognizing temporal transformations. *arXiv preprint arXiv:2007.10730*, 2020. 2, 5, 7, 8

[20] Longlong Jing, Xiaodong Yang, Jingen Liu, and Yingli Tian. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv preprint arXiv:1811.11387*, 2018. 2, 7

[21] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 5

[22] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, volume 33, pages 8545–8552, 2019. 2

[23] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011. 6

[24] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. *arXiv preprint arXiv:1905.00875*, 2019. 3

[25] Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-task self-supervised learning for temporal correspondence. In *Advances in Neural Information Processing Systems*, pages 318–328, 2019. 3, 6

[26] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, pages 7082–7092, 2019. 4

[27] Dezhao Luo, Chang Liu, Yu Zhou, Dongbao Yang, Can Ma, Qixiang Ye, and Weiping Wang. Video cloze procedure for self-supervised spatio-temporal learning. In *AAAI*, 2020. 2, 5, 7, 8

[28] Shuang Ma, Zhaoyang Zeng, Daniel McDuff, and Yale Song. Learning audio-visual representations with active contrastive coding. *arXiv preprint arXiv:2009.09805*, 2020. 2

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

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *T-PAMI*, 39(6):1137–1149, 2017. 4

[31] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016. 3

[32] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017. 8

[33] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017. 3

[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, 6

[35] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*, 3(5), 2019. 7

[36] Li Tao, Xueting Wang, and Toshihiko Yamasaki. Self-supervised video representation using pretext-contrastive learning. *arXiv preprint arXiv:2010.15464*, 2020. 2

[37] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015. 1, 4

[38] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459, 2018. 1, 2, 4, 5

[39] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 391–408, 2018. 3

[40] Jiangliu Wang, Jianbo Jiao, and Yun-Hui Liu. Self-supervised video representation learning by pace prediction. *arXiv preprint arXiv:2008.05861*, 2020. 2, 7, 8

[41] Ning Wang, Yibing Song, Chao Ma, Wengang Zhou, Wei Liu, and Houqiang Li. Unsupervised deep tracking. In *CVPR*, pages 1308–1317, 2019. 3

[42] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015. 3

[43] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, pages 2566–2576, 2019. 3, 6

[44] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*, pages 10334–10343, 2019. 2, 5, 6, 7, 8

[45] Ceyuan Yang, Yinghao Xu, Bo Dai, and Bolei Zhou. Video representation learning with visual tempo consistency. *arXiv preprint arXiv:2006.15489*, 2020. 2

[46] Xinyu Yang, Majid Mirmehdi, and Tilo Burghardt. Back to the future: Cycle encoding prediction for self-supervised contrastive video representation learning. *arXiv preprint arXiv:2010.07217*, 2020. 2, 7

[47] Yuan Yao, Chang Liu, Dezhao Luo, Yu Zhou, and Qixiang Ye. Video playback rate perception for self-supervised spatio-temporal representation learning. In *CVPR*, pages 6548–6557, 2020. 2, 5, 7, 8

[48] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019. 3

[49] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 3

[50] Yizhou Zhou, Xiaoyan Sun, Chong Luo, Zheng-Jun Zha, and Wenjun Zeng. Spatiotemporal fusion in 3d cnns: A probabilistic view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9829–9838, 2020. 1

[51] Yizhou Zhou, Xiaoyan Sun, Zheng-Jun Zha, and Wenjun Zeng. Mict: Mixed 3d/2d convolutional tube for human action recognition. In *CVPR*, pages 449–458, 2018. 1