DOCTORAL THESIS

# Learning and Reasoning with Visual Correspondence in Time

Xiaolong Wang
*The Robotics Institute*
*Carnegie Mellon University*


**Tech Report Number:** CMU-RI-TR-19-75

Submitted in partial fulfillment
of the requirements of the degree of
Doctor of Philosophy in Robotics

September 2019

Doctoral Committee:

Abhinav Gupta, *Carnegie Mellon University* (Chair)
Martial Hebert, *Carnegie Mellon University*
Deva Ramanan, *Carnegie Mellon University*
Alexei A. Efros, *University of California, Berkeley*
Rob Fergus, *New York University*

# Abstract

There is a famous tale in computer vision: Once, a graduate student asked the famous computer vision scientist Takeo Kanade: "What are the three most important problems in computer vision?" Takeo replied: "Correspondence, correspondence, correspondence!" Indeed, even for the most commonly applied Convolutional Neural Networks (ConvNets), they are internally learning representations that lead to correspondence across objects or object parts. The way these networks learn is via human annotations on millions of static images. For example, humans will label images as dog, car, etc. However, this is not how we humans learn. The visual system of an infant develops in a dynamic and continuous environment without using semantics until much later in life.

In this thesis, I will argue that we need to go beyond images and exploit the massive amount of correspondence in videos. In videos, we have millions of pixels linked to each other by time. I will discuss how to learn correspondence from continuous observations in videos without any human supervision. Once the correspondence is given, it can be utilized as supervision in training the ConvNets, eliminating the need for manual labels. Besides supervision, capturing long-range correspondence is also the key to video understanding as well as interaction reasoning. The effectiveness of these ideas will be demonstrated on tasks including object recognition, tracking, action recognition, affordance and physical property estimation.

# Acknowledgments

I would like to thank my advisor Abhinav Gupta, for all his efforts over the past five years. He motivates me to set high standard for my research, gives inspirational advice in both research and career paths, and helps me in improving presentation and writing skills. His mentorship has been invaluable, and I have enjoyed the good work we have done and our time together.

Thank you to my thesis committee - Martial Hebert, Deva Ramanan, Alyosha Efros and Rob Fergus for their time, kind advice and feedback for my thesis. You have taught me a lot and guided me on improving the quality of my research.

I am also very lucky to work with a few incredible minds during my internships in my Ph.D. studies. I thank all my mentors, Ali Farhadi, Kaiming He, and Alyosha Efros. My first internship experience is with Ali, who has taught me to spend most of my time on defining the right problem before working on it; My second internship mentor Kaiming has motivated me to work on high impact research and taught me the golden rule: "Less is more"; Alyosha was my host during my visit in Berkeley, he not only influenced me on research, but also taught me to look beyond the techniques and think about the fundamentals and the philosophy behind the method.

I have been fortunate to work with many hardworking and smart collaborators. I would like to thank David Fouhey, Abhinav Shrivastava, Ali Farhadi, Gunnar Sigurdsson, Gul Varol, Ivan Laptev, Rohit Girdhar, Yuan Yuan, Xiaodan Liang, Kaiming He, Ross Girshick, Wei Yang, Yufei Ye, Tian Ye, James Davidson, Roozbeh Mottaghi, Xueting Li, Sifei Liu, Ming-Hsuan Yang, Kihwan Kim, Jan Kautz, Allan Jabri and Alyosha Efros. I have learned a lot from you and it is a great experience working with you.

Besides working on projects, I have also received many supports from my labmates. I would like to thank David Fouhey, Carl Doersch, Abhinav Shrivastava, Xinlei Chen, Jacob Walker, Ishan Misra, Olga Russakovsky, Yin Li, Gunnar Sigurdsson, Nadine Chang, Rohit Girdhar, Aayush Bansal, Shubham Tulsiani, Saurabh Gupta, Kenneth Marino, Lerrel Pinto, Adithya Murali, Senthil Purushwalkam, Sam Powers, Dhiraj Gandhi, Tian Ye, Yufei Ye, Nilesh Kulkarni, Tao Chen, Wenxuan Zhou, Victoria Dean, and Helen Jiang, Gaurav Pathak, Pratyusha Sharma. Thank you all for discussing with me on interesting ideas, improving my paper writing and presentation. I would especially thank you for helping me improve my talks in thesis proposal and job presentation.

Thank you to Suzanne Lyons Muth, Christine Downey, and Lynnetta Miller for all the help and support over my 5 years in CMU.

Thanks to my advisors and mentors before my Ph.D. studies: Liang Lin for getting me started on computer vision and his guidance on my first research topic; Xiangji Chen for taking me in the programming world.

Finally, I would like to thank my family for their support. I want to thank my parents for their unconditional love and countless sacrifices over so many years. I especially want to thank my wife Sifei for her love, support, encouragement and inspirations on many research ideas.

# Contents

# Chapter 1

# Introduction

In recent years, the field of computer vision has been completely transformed by the success of deep neural networks. A key ingredient behind this success is human annotations on millions of static images. Unfortunately, this key ingredient also turns out to be the biggest bottleneck: the number of labels is limited by the high cost of human labor. As computer vision works towards more difficult and structured AI tasks, it becomes more challenging for humans to provide training supervision. Given this situation, we ask the question: is there any information in the data we have not fully utilized to guide learning?

To solve this problem, we need to go beyond images and exploit the structure in videos across space and time. The key is that our visual world is continuous and smoothly-varying over time. This leads to the spatio-temporal stability in videos, which is thought to play an important role in the development of biological vision. For example, Wood [354] performed studies on newborn chicks in different controlled environments, and observed that when the chicks were raised in a visual environment where the object is not temporally smooth, their object recognition abilities were severely impaired.

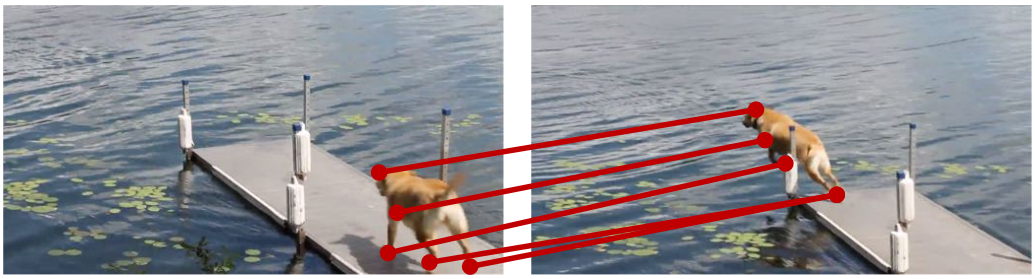


Figure 1.1: Visual correspondence between pixels in time.

How do we utilize this stability and continuity in the visual world to help us in learning visual representations? In this thesis, we focus on the visual correspondence in time. There is unlimited amount of correspondence in videos: Since the visual world is continuous in the video, there is inherent correspondence between observations in different time steps. These observations can be represented by pixels (Figure 1.1), objects or scenes. Correspondence is the glue that connects disparate visual percepts into persistent entities. In fact, finding the

correspondence is one of the first capabilities that develops in infants[1].

Inspired by these observations, we have explored correspondence for learning visual representations in three directions in this thesis: (i) I will first discuss how to learn correspondence and utilize correspondence as the supervisory signal in training deep networks, which eliminates the need for human annotations; (ii) Besides using correspondence for supervision, we can explicitly model the correspondence in videos via relationship reasoning for human action recognition; (iii) Going beyond recognition, correspondence is also the key for understanding common sense including the affordance of the scene and the physical properties of objects. I will explain these three directions in the following sections.

## 1.1 Self-supervised Learning

To break the limitations of human supervision, we have been working on utilizing the correspondence as the supervisory signals to train deep networks. We will first introduce how to learn correspondence automatically from the videos without human supervision [347]. Once we have the correspondence, we can perform visual tracking to obtain pairs of object patches with different views and perform similarity learning for deep representations [341, 345].

In Chapter 2, we developed a self-supervised framework for learning deep spatial features that enable finding dense correspondence between frames separated far in time (i.e., long-range flow). Note that this is extremely difficult for human to label, because annotating which pixel in one frame corresponds to a pixel in another frame is labor intensive. Our model learns to track spatial features back and forth through time, relying on the inherent cycle-consistency of events in time for self-supervision (i.e., consistency of the starting query and ending result). The acquired representations can be applied across a wide range of visual correspondence tasks (e.g., tracking segmentation and pose) *without* any further training on the target dataset. This is a fundamental component in this thesis.

Once we learn to track and find correspondence in videos, we can use the correspondence as the supervisory signal to train deep networks. In Chapter 3, we download 100K YouTube videos and obtain 4 million tracks of object patches from them. With these visual tracks, we train an AlexNet architecture network with a ranking loss function: two patches in the same track should have similar deep feature representations given that they might be the same object with different views or deformations. Surprisingly, even without any labels, we observe that semantics actually emerge after training with this ranking objective. This is one of the *first* works showing that we can train a ConvNet with a standard architecture in a self-supervised manner. Moving ahead, we have also worked on combining the tracking signal with another self-supervised signal in Chapter 4. Our results show that the self-supervised representation can not only be transferred to object detection tasks, but also perform better than supervised learning (with human annotations) in 3D understanding. These observations suggest the great potential of the generalization ability of self-supervised representations.

[1] http://www.aoa.org/patients-and-public/good-vision-throughout-life/childrens-vision/infant-vision-birth-to-24-months-of-age

## 1.2 Video Understanding

Most current approaches for visual tasks feed all the raw data into a simple deep network and attempt to learn everything (including the structure) implicitly. While this has brought us many successes, this thesis has shown that explicit reasoning through correspondence in videos [335, 338, 344] leads to improved performance in video understanding.

In neural networks, both convolutional and recurrent operations can only process one local neighborhood at a time. In order to overcome this, we have introduced non-local operations as neural network modules in Chapter 5, which capture the long-range dependencies between corresponding and correlated patterns in videos. Specifically, this non-local operation updates the feature at a position as a weighted sum of the related or similar features at all positions across space and time. By embedding the non-local operations in neural networks (namely, Non-local Neural Networks), the model achieved state-of-the-art performance in the task of human action recognition. Motivated by this work, in Chapter 6, we have also built a method that uses a space-time region graph, which takes object regions as nodes and connects them across space and time. By reasoning with graph neural networks on object level concepts in an explicit way, it not only leads to better performance but also makes the model more explainable.

In Chapter 7, we argue that the essence of an action lies in the changes it brings to the environment. For example, the action of "kicking a ball" is defined by the state changes of the ball caused by the player. Thus, we model action as the transformation from the state of environment before the action (precondition) to the state after the action occurred (effect). In the experiments, we show state-of-the-art performance on action recognition tasks. More interestingly, the model can also be applied to visual prediction task: Given the precondition state and the transformation (action), the model can retrieve the corresponding video of the effect state which is visually consistent with the precondition state.

## 1.3 Interaction Reasoning

One of the long-term goals of computer vision is to integrate it with robotics. This requires us to learn a intellectual system which not only is able to perceive the world, but also interact with the world. In this thesis, I will introduce my efforts on understanding and reasoning with visual interaction. Specifically, I have focused on designing neural networks to understand scene affordance [337] and the physical properties of the objects [377], with visual correspondence as the key component.

In Chapter 8, we study the scene affordance. Affordance is first proposed by James J. Gibson in late seventies, where he describes affordances as opportunities for interactions in a scene or environment. We design a system to watch TV sitcoms and find correspondence between the frames of the same scene but across different videos. We perform human pose estimation and collect all the human poses appeared in the same scene together. By collecting this data, we can train a network to understand the interaction between the human poses and the scene.

Besides observing the interaction between humans and scenes. We also propose to understand the physical properties of the objects by watching the interaction between two different objects in Chapter 9. We train a predictive neural network model from the given interaction videos. By creating a bottleneck in the network, we show that we can learn disentangled representations for different physical properties. The key of training this predictive model is: instead of directly predicting the pixels in the future frame, we predict the correspondence (represented by optical flow) between the current frames and the future frame.

**Following is the relevant publication list for each chapter.**

1. Chapter 2 - Learning Correspondence from the Cycle-consistency of Time [347] (CVPR 2019)
2. Chapter 3 - Unsupervised Learning of Visual Representations using Videos [341] (ICCV 2015)
3. Chapter 4 - Transitive Invariance for Self-supervised Visual Representation Learning [345] (ICCV 2017)
4. Chapter 5 - Non-local Neural Networks [338] (CVPR 2018)
5. Chapter 6 - Videos as Space-Time Region Graphs [344] (ECCV 2018)
6. Chapter 7 - Actions $\sim$ Transformations [335] (CVPR 2016)
7. Chapter 8 - Binge Watching: Scaling Affordance Learning from Sitcoms [337] (CVPR 2017)
8. Chapter 9 - Interpretable Intuitive Physics Model [377] (ECCV 2018)

# Part I

# Self-supervised Learning with Correspondence

# Chapter 2

# Learning Correspondence from the Cycle-consistency of Time

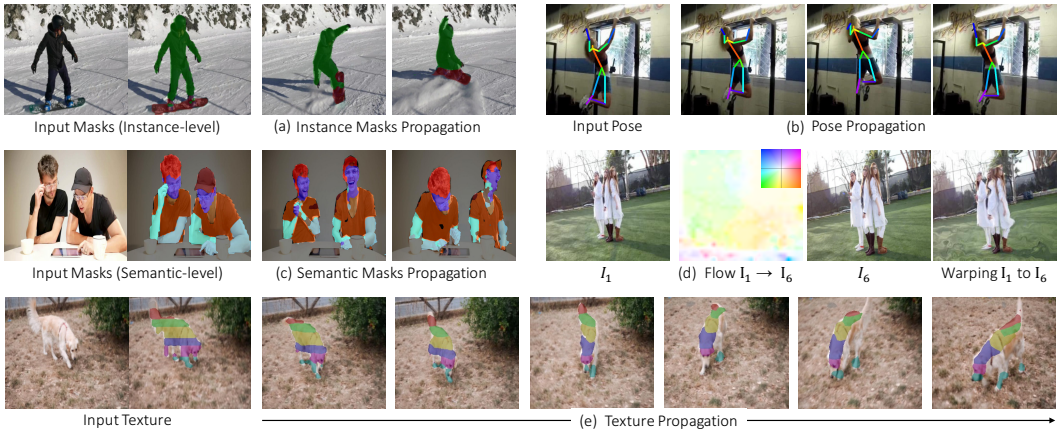



Figure 2.1: We propose to learn a representation for visual correspondence from raw video. Without any fine-tuning, the acquired representation generalizes to various tasks involving visual correspondence, allowing for propagation of: (a) Multiple Instance Masks; (b) Pose; (c) Semantic Masks; (d) Long-Range Optical Flow; (e) Texture.

It is an oft-told story that when a young graduate student asked Takeo Kanade what are the three most important problems in computer vision, Kanade replied: "Correspondence, correspondence, correspondence!" Indeed, most fundamental vision problems, from optical flow and tracking to action recognition and 3D reconstruction, require some notion of visual correspondence. Correspondence is the glue that links disparate visual percepts into persistent entities and underlies visual reasoning in space and time.

Learning representations for visual correspondence, from pixel-wise to object-level, has been widely explored, primarily with supervised learning approaches requiring large amounts of labelled data. For learning low-level correspondence, such as optical flow, synthetic computer graphics data is often used as supervision [65, 133, 247, 299], limiting

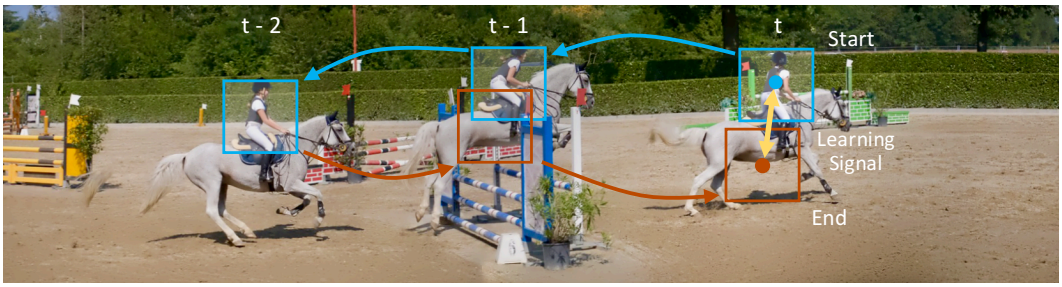


Figure 2.2: **A Cycle in Time.** Given a video, tracking along the sequence formed by a cycle in time can be self-supervised: the target is simply the beginning of the cycle. The yellow arrow between the start and end represents the differentiable learning signal.

generalization to real scenes. On the other hand, approaches for learning higher-level semantic correspondence rely on human annotations [117, 312, 334], which becomes prohibitively expensive at large scale. In this work, our aim is to learn representations that support reasoning at various levels of visual correspondence (Figure 2.1) from scratch and without human supervision.

A fertile source of free supervision is video. Because the world does not change abruptly, there is inherent visual correspondence between observations adjacent in time. The problem is how to find these correspondences and turn them into a learning signal. In a largely static world observed by a stationary camera, such as a webcam trained on the Eiffel Tower, correspondence is straightforward because nothing moves and capturing visual invariance (to weather, lighting) amounts to supervised metric learning. In the dynamic world, however, change in appearance is confounded by movement in space. Finding correspondence becomes more difficult because capturing visual invariance now requires learning to track, but tracking relies on a model of visual invariance. This chapter proposes to learn to do both simultaneously, in a self-supervised manner.

The key idea is that we can obtain unlimited supervision for correspondence by tracking backward and then forward (i.e. along a cycle in time) and using the *inconsistency* between the start and end points as the loss function (Figure 2.2). We perform tracking by template-matching in a learned deep feature space. To minimize the loss – i.e. to be *cycle-consistent* – the model must learn a feature representation that supports identifying correspondences across frames. As these features improve, the ability to track improves, inching the model toward cycle-consistency. Learning to chain correspondences in such a feature space should thus yield a visual similarity metric tolerant of local transformations in time, which can then be used at test-time as a stand-alone distance metric for correspondence.

While conceptually simple, implementing objectives based on cycle-consistency can be challenging. Without additional constraints, learning can take shortcuts, making correspondences cycle-consistent but wrong [395]. In our case, a track that never moves is inherently cycle-consistent. We avoid this by forcing the tracker to re-localize the next patch in each successive frame. Furthermore, cycle-consistency may not be achievable due to sudden changes in object pose or occlusions; *skip*-cycles can allow for cycle-consistency by skipping frames, as in Figure 2.3 (right). Finally, correspondence may be poor early in training, and shorter cycles may ease learning, as in Figure 2.3 (left). Thus, we simultaneously learn from many kinds of cycles to induce a natural curriculum and provide better training data.

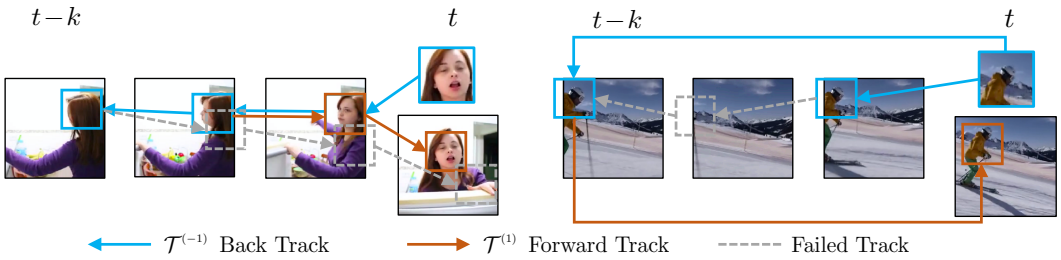



Figure 2.3: **Multiple Cycles and Skip Cycles.** Cycle-consistency may not be achievable due to sudden changes in object pose or occlusions. Our solution is to optimize multiple cycles of different lengths simultaneously. This allows learning from shorter cycles when the full cycle is too difficult (left). This also allows cycles that skip frames, which can deal with momentary occlusions (right).

The proposed formulation can be used with any differentiable tracking operation, providing a general framework for learning representations for visual correspondence from raw video. Because the method does not rely on human annotation, it can learn from the near infinite video data available online. We demonstrate the usefulness of the learned features for tasks at various levels of visual correspondence, ranging from pose, keypoint, and segmentation propagation (of objects and parts) to optical flow.

## 2.1 Background

**Temporal Continuity in Visual Learning.** Temporal structure serves as a useful signal for learning because the visual world is continuous and smoothly-varying. Spatio-temporal stability is thought to play a crucial role in the development of invariant representations in biological vision [196,354,355,378]. For example, Wood [354] showed that for newborn chicks raised in a visual world that was not temporally smooth, object recognition abilities were severely impaired. Computational approaches for unsupervised learning have sought to leverage this continuity, such as continuous transformation learning [74,322], "slow" feature learning [141,353,403] and information maximization between neighbouring patches in time [313]. Our work can be seen as slow feature learning with fixation, learned end-to-end without supervision.

**Self-supervised Representation Learning from Video.** Learning representations from video using time as supervision has been extensively studied, both as future prediction task [95,215, 219,290] as well as motion estimation [2,141,199,209,309]. Our approach is most related to the methods of Wang et al. [342,346] and Pathak et al. [234], which use off-the-shelf tools for tracking and optical flow respectively, to provide supervisory signal for training. However, representations learned in this way are inherently limited by the power of these off-the-shelf tools as well as their failure modes. We address this issue by learning the representation and the tracker jointly, and find the two learning problems to be complementary. Our work is also inspired by the innovative approach of Vondrick et al [319] where video colorization is used as a pretext self-supervised task for learning to track. While the idea is very intriguing, in Section 2.3 we find that colorization is a weaker source of supervision for correspondence than cycle-consistency, potentially due to the abundance of constant-color regions in natural scenes.

**Tracking.** Classic approaches to tracking treat it as a matching problem, where the goal is to find a given object/patch in the next frame (see [66] for overview), and the key challenge is to track reliably over extended time periods [1, 151, 270, 356]. Starting with the seminal work of Ramanan et al. [246], researchers largely turned to "tracking as repeated recognition", where trained object detectors are applied to each frame independently [5, 117, 152, 194, 312, 334, 362]. Our work harks back to the classic tracking-by-matching methods in treating it as a correspondence problem, but uses learning to obtain a robust representation that is able to model wide range of appearance changes.

**Optical Flow.** Correspondence at the pixel level – mapping where each pixel goes in the next frame – is the optical flow estimation problem. Since the energy minimization framework of Horn and Schunck [128] and coarse-to-fine image warping by Lucas and Kanade [213], much progress has been made in optical flow estimation [20, 65, 133, 223, 247, 298, 299]. However, these methods still struggle to scale to long-range correspondence in dynamic scenes with partial observability. These issues have driven researchers to study methods for estimating long-range optical flow [19, 193, 251, 252, 257, 265]. For example, Brox and Malik [19] introduced a descriptor that matches region hierarchies and provides dense and subpixel-level estimation of flow. Our work can be viewed as enabling mid-level optical flow estimation.

**Mid-level Correspondence.** Given our focus on finding correspondence at the patch level, our method is also related to the classic SIFT Flow [206] algorithm and other methods for finding mid-level correspondences between regions across different scenes [107, 156, 394]. More recently, researchers have studied modeling correspondence in deep feature space [110, 157, 182, 254, 255, 310]. In particular, our work draws from Rocco et al. [254, 255], who propose a differentiable soft inlier score for evaluating quality of alignment between spatial features and provides a loss for learning semantic correspondences. Most of these methods rely on learning from simulated or large-scale labeled datasets such as ImageNet, or smaller custom human-annotated data with narrow scope. We address the challenge of learning representations of correspondence without human annotations.

**Forward-Backward and Cycle Consistency.** Our work is influenced by the classic idea of forward-backward consistency in tracking [1, 151, 270, 356], which has long been used as an evaluation metric for tracking [151] as well as a measure of uncertainty [1]. Recent work on optical flow estimation [140, 217, 222, 315, 349] also utilizes forward-backward consistency as an optimization goal. For example, Meister et al. [222] combines one-step forward and backward consistency check with pixel reconstruction loss for learning optical flows. Compared to pixel reconstruction, modeling correspondence in feature space allows us to follow and learn from longer cycles. Forward-backward consistency is a specific case of cycle-consistency, which has been widely applied as a learning objective for 3D shape matching [132], image alignment [394, 395, 397], depth estimation [92, 379, 393], and image-to-image translation [7, 399]. For example Zhou et al. [395] used 3D CAD models to render two synthetic views for pairs of training images and construct a correspondence flow 4-cycle. To the best of our knowledge, our work is the first to employ cycle-consistency across multiple steps in time.

## 2.2 Approach

An overview of the training procedure is presented in Figure 2.4(a). The goal is to learn a feature space $\phi$ by tracking a patch $p_t$ extracted from image $I_t$ backwards and then forwards

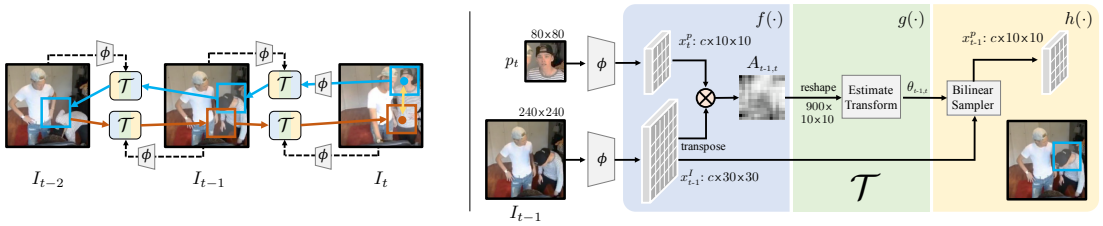



Figure 2.4: **Method Overview**. (a) On the left: During training, the model learns a feature space encoded by $\phi$ to perform tracking using tracker $\mathcal{T}$. By tracking backward and then forward, we can use cycle-consistency to supervise learning of $\phi$. Note that only the initial patch $p_t$ is explicitly encoded by $\phi$; other patch features along the cycle are obtained by localizing image features. (b) On the right: We show one step of tracking back in time from $t$ to $t-1$. Given input image features $x_{t-1}^I$ and query patch features $x_t^p$, $\mathcal{T}$ localizes the patch $x_{t-1}^p$ in $x_{t-1}^I$. This operation is performed iteratively to track along the cycle in (a).

in time, while minimizing the cycle-consistency loss $l_\theta$ (yellow arrow). Learning $\phi$ relies on a simple tracking operation $\mathcal{T}$, which takes as inputs the features of a current patch and a target image, and returns the image feature region with maximum similarity. Our implementation of $\mathcal{T}$ is shown in Figure 2.4(b): without information of where the patch came from, $\mathcal{T}$ must match features encoded by $\phi$ to localize the next patch. As shown in Figure 2.4(a), $\mathcal{T}$ can be iteratively applied backwards and then forwards through time to track along an arbitrarily long cycle. The cycle-consistency loss $l_\theta$ is the euclidean distance between the spatial coordinates of initial patch $p_t$ and the patch found at the end of the cycle in $I_t$. In order to minimize $l_\theta$, the model must learn a feature space $\phi$ that allows for robustly measuring visual similarity between patches along the cycle.

Note that $\mathcal{T}$ is *only used in training* and is deliberately designed to be weak, so as to place the burden of representation on $\phi$. At test time, the learned $\phi$ is used directly for computing correspondences. In the following, we first formalize cycle-consistent tracking loss functions and then describe our architecture for mid-level correspondence.

### 2.2.1 Cycle-Consistency Losses

We describe a formulation of cycle-consistent tracking and use it to succinctly express loss functions based on temporal cycle-consistency.

#### Recurrent Tracking Formulation

Consider as inputs a sequence of video frames $I_{t-k:t}$ and a patch $p_t$ taken from $I_t$. These pixel inputs are mapped to a feature space by an encoder $\phi$, such that $x_{t-k:t}^I = \phi(I_{t-k:t})$ and $x_t^p = \phi(p_t)$.

Let $\mathcal{T}$ be a differentiable operation $x_s^I \times x_t^p \mapsto x_s^p$, where $s$ and $t$ represent time steps. The role of $\mathcal{T}$ is to localize the patch features $x_s^p$ in image features $x_s^I$ that are most similar to $x_t^p$. We can apply $\mathcal{T}$ iteratively in a forward manner $i$ times from $t-i$ to $t-1$:

$$\mathcal{T}^{(i)}(x_{t-i}^I, x^p) = \mathcal{T}(x_{t-1}^I, \mathcal{T}(x_{t-2}^I, ...\mathcal{T}(x_{t-i}^I, x^p)))$$

By convention, the tracker $\mathcal{T}$ can be applied backwards $i$ times from time $t-1$ to $t-i$:

$$\mathcal{T}^{(-i)}(x_{t-1}^I, x^p) = \mathcal{T}(x_{t-i}^I, \mathcal{T}(x_{t-i+1}^I, ...\mathcal{T}(x_{t-1}^I, x^p)))$$

#### Learning Objectives

The following learning objectives rely on a measure of agreement $l_\theta(x_t^p, \hat{x}_t^p)$ between the initial patch and re-localized patch (defined in Section 2.2.2).

**Tracking**: The cycle-consistent loss $\mathcal{L}_{long}^i$ is defined as

$$\mathcal{L}_{long}^i = l_\theta(x_t^p, \mathcal{T}^{(i)}(x_{t-i+1}^I, \mathcal{T}^{(-i)}(x_{t-1}^I, x_t^p))).$$

The tracker attempts to follow features backward and then forward $i$ steps in time to re-arrive to the initial query, as depicted in Figure 2.4(a).

**Skip Cycle:** In addition to cycles through consecutive frames, we also allow skipping through time. We define the loss on a two-step skip-cycle as $\mathcal{L}_{skip}^i$:

$$\mathcal{L}_{skip}^i = l_\theta(x_t^p, \mathcal{T}(x_t^I, \mathcal{T}(x_{t-i}^I, x_t^p))).$$

This attempts longer-range matching by skipping to the frame $i$ steps away.

**Feature Similarity:** We explicitly require the query patch $x_t^p$ and localized patch $\mathcal{T}(x_{t-i}^I, x_t^p)$ to be similar in feature space. This loss amounts to the negative Frobenius inner product between spatial feature tensors:

$$\mathcal{L}_{sim}^i = -\langle x_t^p, \mathcal{T}(x_{t-i}^I, x_t^p) \rangle$$

In principle, this loss can further be formulated as the inlier loss from [255]. The overall learning objective sums over the $k$ possible cycles, with weight $\lambda = 0.1$:

$$\mathcal{L} = \sum_{i=1}^{k} \mathcal{L}_{sim}^i + \lambda \mathcal{L}_{skip}^i + \lambda \mathcal{L}_{long}^i.$$

### 2.2.2 Architecture for Mid-level Correspondence

The learning objective thus described can be used to train arbitrary differentiable tracking models. In practice, the architecture of the encoder determines the type of correspondence captured by the acquired representation. In this work, we are interested in a model for mid-level temporal correspondence. Accordingly, we choose the representation to be a mid-level deep feature map, coarser than pixel space but with sufficient spatial resolution to support tasks that require localization. An overview is provided in Figure 2.4(b).

#### Spatial Feature Encoder $\phi$

We compute spatial features with a ResNet-50 architecture [114] without $\text{res}_5$ (the final 3 residual blocks). We reduce the spatial stride of $\text{res}_4$ for larger spatial outputs. Input frames are $240 \times 240$ pixels, randomly cropped from video frames re-scaled to have $\texttt{min}(H, W) = 256$. The size of the spatial feature of the frame is thus $30 \times 30$. Image patches are $80 \times 80$, randomly cropped from the full $240 \times 240$ frame, so that the feature is $10 \times 10$. We perform $l_2$ normalization on the channel dimension of spatial features to facilitate computing cosine similarity.

**Differentiable Tracker $\mathcal{T}$**

Given the representation from the encoder, we perform tracking with $\mathcal{T}$. As illustrated in Figure 2.4(b), the differentiable tracker is composed of three main components.

**Affinity function** $f$ provides a measure of similarity between coordinates of spatial features $x^I$ and $x^p$. We denote the affinity function as $f(x^I, x^p) := A$, such that $f : \mathbb{R}^{c\times 30\times 30} \times \mathbb{R}^{c\times 10\times 10} \rightarrow \mathbb{R}^{900\times 100}$.

A generic choice for computing the affinity is the dot product between embeddings, referred to in recent literature as attention [314,339] and more historically known as normalized cross-correlation [65,194]. With spatial grid $j$ in feature $x^I$ as $x^I(j)$ and the grid $i$ in $x^p$ as $x^p(i)$,

$$A(j,i) = \frac{\exp\left(x^I(j)^\intercal x^p(i)\right)}{\sum_j \exp\left(x^I(j)^\intercal x^p(i)\right)} \tag{2.1}$$

where the similarity $A(j,i)$ is normalized by the softmax over the spatial dimension of $x^I$, for each $x^p(i)$. Note that the affinity function is defined for any feature dimension.

**Localizer** $g$ takes affinity matrix $A$ as input and estimates localization parameters $\theta$ corresponding to the patch in feature $x^I$ which best matches $x^p$. $g$ is composed of two convolutional layers and one linear layer. We restrict $g$ to output 3 parameters for the bilinear sampling grid (i.e. simpler than [136]), corresponding to 2D translation and rotation: $g(A) := \theta$, where $g : \mathbb{R}^{900\times 100} \rightarrow \mathbb{R}^3$. The expressiveness of $g$ is intentionally limited so as to place the burden of representation on the encoder (see Appendix B).

**Bilinear Sampler** $h$ uses the image feature $x^I$ and $\theta$ predicted by $g$ to perform bilinear sampling to produce a new patch feature $h(x^I, \theta)$ which is in the same size as $x^p$, such that $h : \mathbb{R}^{c\times 30\times 30} \times \mathbb{R}^3 \rightarrow \mathbb{R}^{c\times 10\times 10}$.

**End-to-end Joint Training**

The composition of encoder $\phi$ and $\mathcal{T}$ forms a differentiable patch tracker, allowing for end-to-end training of $\phi$ and $\mathcal{T}$:

$$x^I, x^p = \phi(I), \phi(p)$$
$$\mathcal{T}(x^I, x^p) = h(x^I, g(f(x^I, x^p))).$$

**Alignment Objective** $l_\theta$ is applied in the cycle-consistent losses $\mathcal{L}^i_{long}$ and $\mathcal{L}^i_{skip}$, measuring the error in alignment between two patches. We follow the formulation introduced by [254]. Let $M(\theta_{x^p})$ correspond to the bilinear sampling grids used to form a patch feature $x^p$ from image feature $x^I$. Assuming $M(\theta_{x^p})$ contains $n$ sampling coordinates, the alignment objective is defined as:

$$l_\theta(x^p_*, \hat{x}^p_t) = \frac{1}{n}\sum_{i=1}^{n} ||M(\theta_{x^p_*})_i - M(\theta_{\hat{x}^p_t})_i||_2^2$$

## 2.3 Experiments

We report experimental results for a model trained on the VLOG dataset [70] from scratch; training on other large video datasets such as Kinetics gives similar results (see Appendix A.3). The trained representation is evaluated *without fine-tuning* on several challenging

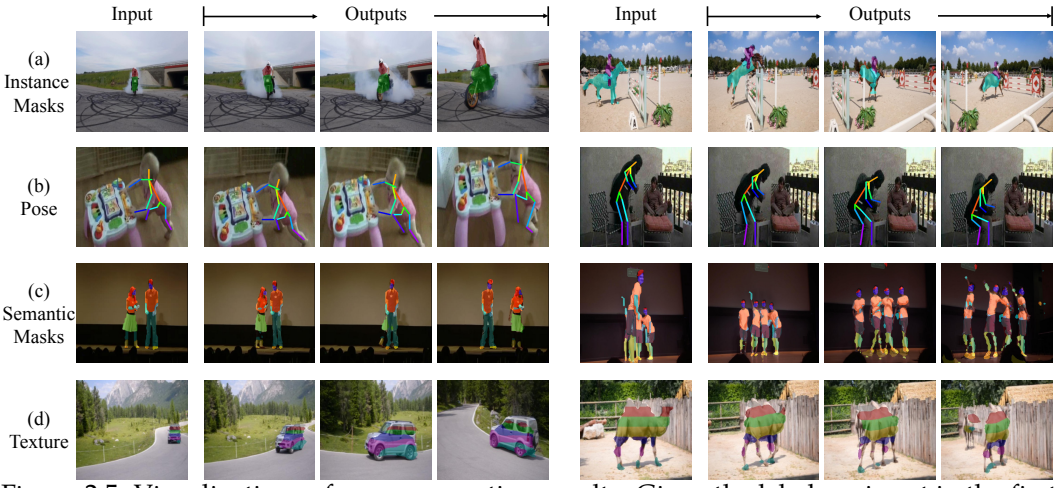


Figure 2.5: Visualizations of our propagation results. Given the labels as input in the first frame, our feature can propagate them to the rest of frames, without further fine-tuning. The labels include (a) instance masks in DAVIS-2017 [241], (b) pose keypoints in JHMDB [143], (c) semantic masks in VIP [392] and even (d) texture map.

video propagation tasks: DAVIS-2017 [241], JHMDB [143] and Video Instance-level Parsing (VIP) [392]. Through various experiments, we show that the acquired representation generalizes to a range of visual correspondence tasks (see Figure 2.5).

### 2.3.1 Common Setup and Baselines

**Training.** We train the model on the VLOG dataset [70] without using any annotations or pre-training. The VLOG dataset contains 114K videos and the total length of the videos is 344 hours. During training, we set the number of past frames as $k = 4$. We train on a 4-GPU machine with a mini-batch size of 32 clips (8 clips per GPU), for 30 epochs. The model is optimized with Adam [158] with a learning rate of $0.0002$ and momentum term $\beta_1 = 0.5, \beta_2 = 0.999$.

**Inference.** At test time, we use the trained encoder's representation to compute dense correspondences for video propagation. Given initial labels of the first frame, we propagate the labels to the rest of the frames in the video. Labels are given by specified targets for the first frame of each task, with instance segmentation masks for DAVIS-2017 [241], human pose keypoints JHMDB [143], and both instance-level and semantic-level masks for VIP [392]. The labels of each pixel are discretized to $C$ classes. For segmentation masks, $C$ is the number of instance or semantic labels. For keypoints, $C$ is the number of keypoints. We include a background class. We propagate the labels in the feature space. The labels in the first frame are one-hot vectors, while propagated labels are soft distributions.

**Propagation by k-NN.** Given a frame $I_t$ and a frame $I_{t-1}$ with labels, we compute their affinity in feature space: $A_{t-1,t} = f(\phi(I_{t-1}), \phi(I_t))$ (Eq. 2.1). We compute label $y_i$ of pixel $i$ in $I_t$ as

$$y_i = \sum_j A_{t-1,t}(j, i) y_j, \tag{2.2}$$

where $A_{t-1,t}(j,i)$ is the affinity between pixels $i$ in $I_t$ and $j$ in $I_{t-1}$. We propagate from the top-5 pixels with the greatest affinity $A_{t-1,t}(j,i)$ for each pixel $i$. Labels are propagated from $I_{t-1:t-K}$, as well as $I_1$, and averaged. Finally, we up-sample the label maps to image size. For segmentation, we use the `argmax` of the class distribution of each pixel. For keypoints, we choose the pixel with the maximum score for each keypoint type.

**Baselines.** We compare with the following baselines:

- **Identity**: Always copy the first frame labels.
- **Optical Flow** (FlowNet2 [133]): A state-of-the-art method for predicting optical flow with neural networks [133]. We adopt the open-source implementation which is trained with synthetic data in a supervised manner. For a target frame $I_t$, we compute the optical flow from frame $I_{t-1}$ to $I_t$ and warp the labels in $I_{t-1}$ to $I_t$.
- **SIFT Flow** [206]: For a target frame $I_t$, we compute the SIFT Flow between $I_t$ and its previous frames. We propagate the labels in $K$ frames before $I_t$ and the first frame via SIFT Flow warping. The propagation results are averaged to compute the labels for $I_t$.
- **Transitive Invariance** [346]: A self-supervised approach that combines multiple objectives: (i) visual tracking on raw video [342] and (ii) spatial context reasoning [47]. We use the open-sourced pre-trained VGG-16 [280] model and adopt our proposed inference procedure.
- **DeepCluster** [25]: A self-supervised approach which uses a K-means objective to iteratively update targets and learn a mapping from images to targets. It is trained on the ImageNet dataset without using annotations. We apply the trained model with VGG-16 and adopt the same inference procedure as our method.
- **Video Colorization** [319]: A self-supervised approach for label propagation. Trained on the Kinetics [154] dataset, it uses color propagation as self-supervision. The architecture is based on 3D ResNet-18. We report their results.
- **ImageNet Pre-training** [114]: The conventional setup for supervised training of ResNet-50 on ImageNet.
- **Fully-Supervised Methods**: We report fully-supervised methods for reference, which not only use ImageNet pre-training but also fine-tuning on the target dataset. Note that these methods do not always follow the inference procedure used with method, and labels of the first frame are not used for JHMDB and VIP at test time.

### 2.3.2 Instance Propagation on DAVIS-2017

We apply our model to video object segmentation on the DAVIS-2017 validation set [241]. Given the initial masks of the first frame, we propagate the masks to the rest of the frames. Note that there can be multiple instances in the first frame. We follow the standard metrics including the region similarity $\mathcal{J}$ (IoU) and the contour-based accuracy $\mathcal{F}$. We set $K = 7$, the number of reference frames in the past.

We show comparisons in Table 2.1. Comparing to the recent Video Colorization approach [319], our method is 7.3% in $\mathcal{J}$ and 6.7% in $\mathcal{F}$. Note that although we are only $4.4\%$ better than the DeepCluster baseline in $\mathcal{J}$, we are better in contour accuracy $\mathcal{F}$ by $6.2\%$. Thus, DeepCluster does not capture dense correspondence on the boundary as well.

For fair comparisons, we also implemented our method with a ResNet-18 encoder, which has less parameters compared to the VGG-16 in [25,346] and the 3D convolutional ResNet-18

| model | Supervised | $\mathcal{J}$(Mean) | $\mathcal{F}$(Mean) |
|---|---|---|---|
| Identity | | 22.1 | 23.6 |
| Random Weights (ResNet-50) | | 12.4 | 12.5 |
| Optical Flow (FlowNet2) [133] | | 26.7 | 25.2 |
| SIFT Flow [206] | | 33.0 | 35.0 |
| Transitive Inv. [346] | | 32.0 | 26.8 |
| DeepCluster [25] | | 37.5 | 33.2 |
| Video Colorization [319] | | 34.6 | 32.7 |
| Ours (ResNet-18) | | 40.1 | 38.3 |
| Ours (ResNet-50) | | **41.9** | **39.4** |
| ImageNet (ResNet-50) [114] | ✓ | 50.3 | 49.0 |
| Fully Supervised [24, 371] | ✓ | 55.1 | 62.1 |

Table 2.1: Evaluation on instance mask propagation on DAVIS-2017 [241]. We follow the standard metric on region similarity $\mathcal{J}$ and contour-based accuracy $\mathcal{F}$.

in [319]. We observe that results are only around $2\%$ worse than our model with ResNet-50, which is still better than the baselines.

While the ImageNet pre-trained network performs better than our method on this task, we argue it is easy for the ImageNet pre-trained network to recognize objects under large variation as it benefits from curated object-centric annotation. Though our model is only trained on indoor scenes without labels, it generalizes to outdoor scenes.

Although video segmentation is an important application, it does not necessarily show that the representation captures dense correspondence.

### 2.3.3 Pose Keypoint Propagation on JHMDB

To see whether our method is learning more spatially precise correspondence, we apply our model on the task of keypoint propagation on the split 1 validation set of JHMDB [143]. Given the first frame with $15$ labeled human keypoints, we propagate them through time. We follow the evaluation of the standard PCK metric [372], which measures the percentage of keypoints close to the ground truth in different thresholds of distance. We set the number of reference frames same as experiments in DAVIS-2017.

As shown in Table 2.2, our method outperforms all self-supervised baselines by a large margin. We observe that SIFT Flow actually performs better than other self-supervised learning methods in PCK@.1. Our method outperforms SIFT Flow by $8.7\%$ in PCK@.1 and $9.9\%$ in PCK@.2. Notably, our approach is only $0.7\%$ worse than ImageNet pre-trained features in PCK@.1 and performs better in PCK@.2.

### 2.3.4 Semantic and Instance Propagation on VIP

We apply our approach on the Video Instance-level Parsing (VIP) dataset [392], which is densely labeled with semantic masks for different human parts (e.g., hair, right arm, left arm, coat). It also has instance labels that differentiate humans. Most interestingly, the duration of a video ranges from 10 seconds to 120 seconds in the dataset, which is much longer than aforementioned datasets.

We test our method on the validation set of two tasks in this dataset: (i) The first task is to propagate the semantic human part labels from the first frame to the rest of the video, and

| model | Supervised | PCK@.1 | PCK@.2 |
|---|---|---|---|
| Identity | | 43.1 | 64.5 |
| Optical Flow (FlowNet2) [133] | | 45.2 | 62.9 |
| SIFT Flow [206] | | 49.0 | 68.6 |
| Transitive Inv. [346] | | 43.9 | 67.0 |
| DeepCluster [25] | | 43.2 | 66.9 |
| Video Colorization [319] | | 45.2 | 69.6 |
| Ours (ResNet-18) | | 57.3 | 78.1 |
| Ours (ResNet-50) | | **57.7** | **78.5** |
| ImageNet (ResNet-50) [114] | ✓ | 58.4 | 78.4 |
| Fully Supervised [285] | ✓ | 68.7 | 92.1 |

Table 2.2: Evaluation on pose propagation on JHMDB [143]. We report the PCK in different thresholds.

| model | Supervised | mIoU | $AP^r_{\text{vol}}$ |
|---|---|---|---|
| Identity | | 13.6 | 4.0 |
| Optical Flow (FlowNet2) [133] | | 16.1 | 8.3 |
| SIFT Flow [206] | | 21.3 | 10.5 |
| Transitive Inv. [346] | | 19.4 | 5.0 |
| DeepCluster [25] | | 21.8 | 8.1 |
| Ours (ResNet-50) | | **28.9** | **15.6** |
| ImageNet (ResNet-50) [114] | ✓ | 34.7 | 16.1 |
| Fully Supervised [392] | ✓ | 37.9 | 24.1 |

Table 2.3: Evaluation on propagating human part labels in Video Instance-level Parsing (VIP) dataset [392]. We measure *Semantic Propagation* with mIoU and *Part Instance Propagation* in $AP^r_{\text{vol}}$.

evaluate with the mean IoU metric; (ii) In the second task, the labels in the first frame are given with not only the semantic labels but also the instance identity. Thus, the model must differentiate the different arms of different human instances. We use the standard instance-level human parsing metric [197], mean Average Precision, for overlap thresholds varying from 0.1 to 0.9. Since part segments are relatively small (compared to objects in DAVIS-2017), we increase the input image size to $560 \times 560$ for inference, and use two reference frames, including the first frame.

**Semantic Propagation.** As shown with the mIoU metric in Table 2.3, our method again exceeds all self-supervised baselines by a large margin (a [319] model is currently not available). ImageNet pre-trained models have the advantage of semantic annotation and thus do not necessarily have to perform tracking. As shown in Figure 2.5(c), our method is able to handle occlusions and multiple instances.

**Part Instance Propagation.** This task is more challenging. We show the results in mean $AP^r_{\text{vol}}$ in Table 2.3. Our method performs close to the level of ImageNet pre-trained features. We show different radial thresholds for average precision ($AP^r_{\text{vol}}$) in Table 2.4. ImageNet pre-trained features performs better under smaller thresholds and worse under larger thresholds, suggesting that it has an advantage in finding coarse correspondence while our method is more capable of spatial precision.

| model | $AP^r_{\text{vol}}$ | IoU threshold 0.3 | 0.5 | 0.7 |
|---|---|---|---|---|
| Ours (ResNet-50) | 15.6 | 23.0 | **12.7** | **5.4** |
| ImageNet (ResNet-50) [114] | **16.1** | **24.2** | 11.9 | 4.8 |

Table 2.4: A more detailed analysis of different thresholds for *Part Instance Propagation* on the VIP dataset [392].

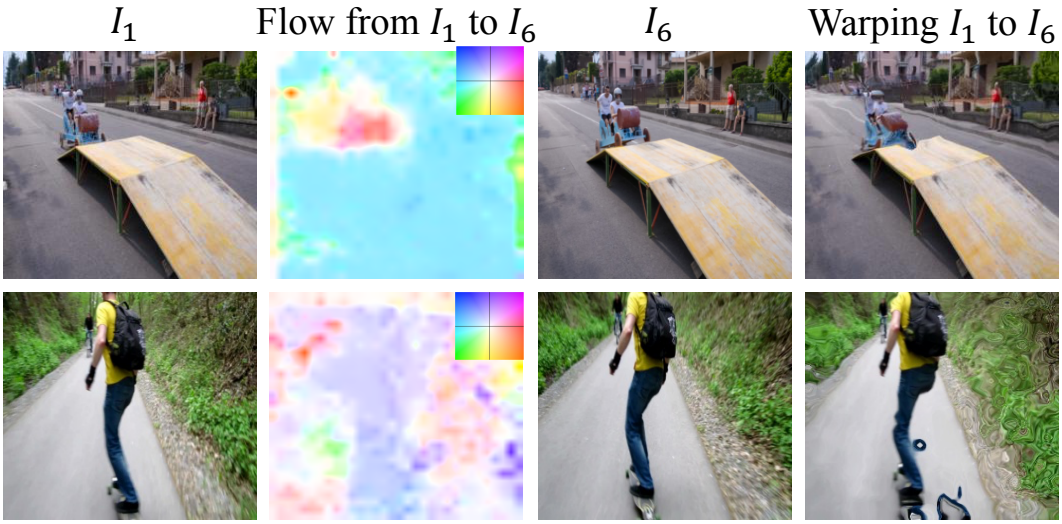



Figure 2.6: Given $I_1, I_6$ which have 5-frame gap, we compute the long-range flows between them with our representation. This flow can be used to warp $I_1$ to generate image similar to $I_6$.

### 2.3.5 Texture Propagation

The acquired representation allows for propagation of not only instance and semantic labels, but also textures. We visualize texture propagation in Figure 2.5 (d); these videos are samples from DAVIS-2017 [241]. We "paint" a texture of 6 colored stripes on an the object in the first frame and propagate it to the rest of the frames using our representation. We observe that the structure of the texture is well preserved in the following frames, demonstrating that the representation allows for finding precise correspondence smoothly though time. See the project page for video examples.

### 2.3.6 Video Frame Reconstructions

Though we do not optimize for pixel-level objectives at training time, we can evaluate how well our method performs on pixel-level reconstruction. Specifically, given two images $I_s$ and $I_t$ distant in time in a video, we compute coordinate-wise correspondences under the acquired representation and generate a flow field for pixel movement between $I_s$ and $I_t$. We then upsample the flow field to the same size as the image and warp it on Image $I_s$ to generate a new image $I'_t$ (as shown in Figure 2.6). We compare the L1 distance between $I'_t$ and $I_t$ in RGB space and report the reconstruction errors in Table 2.5.

| model | 5-F | 10-F |
|---|---|---|
| Identity | 82.0 | 97.7 |
| Optical Flow (FlowNet2) [133] | 62.4 | 90.3 |
| ImageNet (ResNet-50) [114] | 64.0 | 79.2 |
| Ours (ResNet-50) | **60.4** | **76.4** |

Table 2.5: We compute the long-range flow on two frames and warp the first one with the flow. We compare the warped frame with the second frame in L1 distance. The gaps are 5 or 10 frames.

For fair comparison, we perform this experiment on the DAVIS-2017 validation set, which none of the reported methods have seen. We experiment with two time gaps, 5 and 10 frames. For the smaller gap, FlowNet2 [133] performs reasonably well, whereas reconstruction degrades for larger gaps. In both cases, our method performs better than FlowNet2 and the ImageNet pre-trained network. This is encouraging: our method is not trained with pixel-level losses, yet out-performs methods trained with pixel-level tasks and human supervision.

## 2.4 Discussion

While in principle our method should keep improving with more data, in practice, learning seems to plateau after a moderate amount of training (i.e. 30 epochs). An important next step is thus how to better scale to larger, noisier data. A crucial component is improving robustness to occlusions and partial observability, for instance, by using a better search strategy for finding cycles at training time. Another issue is deciding *what* to track at training time. Picking patches at random can result in issues such as stationary background patches and tracking ambiguity – e.g. how should one track a patch containing two objects that eventually diverge? Jointly learning what to track may also give rise to unsupervised object detection. Finally, incorporating more context for tracking both at training and test time may be important for learning more expressive models of spatial-temporal correspondence.

We hope this work is a step toward learning from the abundance of visual correspondence inherent in raw video in a scalable and end-to-end manner. While our experiments show promising results at certain levels of correspondence, much work remains to cover the full spectrum.

# Chapter 3

# Self-supervised Learning with Visual Tracking

In the last chapter, we have explored a way to learn correspondence from the videos with self-supervised learning. In this chapter, we want to see if self-supervision can also lead to a general representation for many different vision applications, ranging from mid-level to high-level tasks.

What is a general visual representation and how can we learn it? At the start of this decade, most computer vision research focused on "what" and used hand-defined features such as SIFT [211] and HOG [39] as the underlying visual representation. Learning was often the last step where these low-level feature representations were mapped to semantic/3D/functional categories. However, the last three years have seen the resurgence of learning visual representations directly from pixels themselves using the deep learning and ConvNets [146, 171, 187]. At the heart of ConvNets is a completely supervised learning paradigm. Often millions of examples are first labeled using Mechanical Turk followed by data augmentation to create tens of millions of training instances. ConvNets are then trained using gradient descent and back propagation. But one question still remains: is strong-supervision necessary for training these ConvNets? Do we really need millions of semantically-labeled images to learn a good visual representation? It seems humans can learn visual representations using little or no semantic supervision but our current learning approaches still remain completely supervised.

In this chapter, we explore the alternative: how we can exploit the unlabeled visual data on the web to train ConvNets (e.g. AlexNet [171])? In the past, there have been several attempts at unsupervised learning using millions of static images [183, 289] or frames extracted from videos [225, 304, 404]. The most common architecture used is an auto-encoder which learns representations based on its ability to reconstruct the input images [15, 231, 248, 316]. While these approaches have been able to automatically learn V1-like filters given unlabeled data, they are still far away from supervised approaches on tasks such as object detection. So, what is the missing link? We argue that static images themselves might not have enough information to learn a good visual representation. But what about videos? Do they have enough information to learn visual representations? In fact, humans also learn their visual representations not from millions of static images but years of dynamic sensory inputs. Can

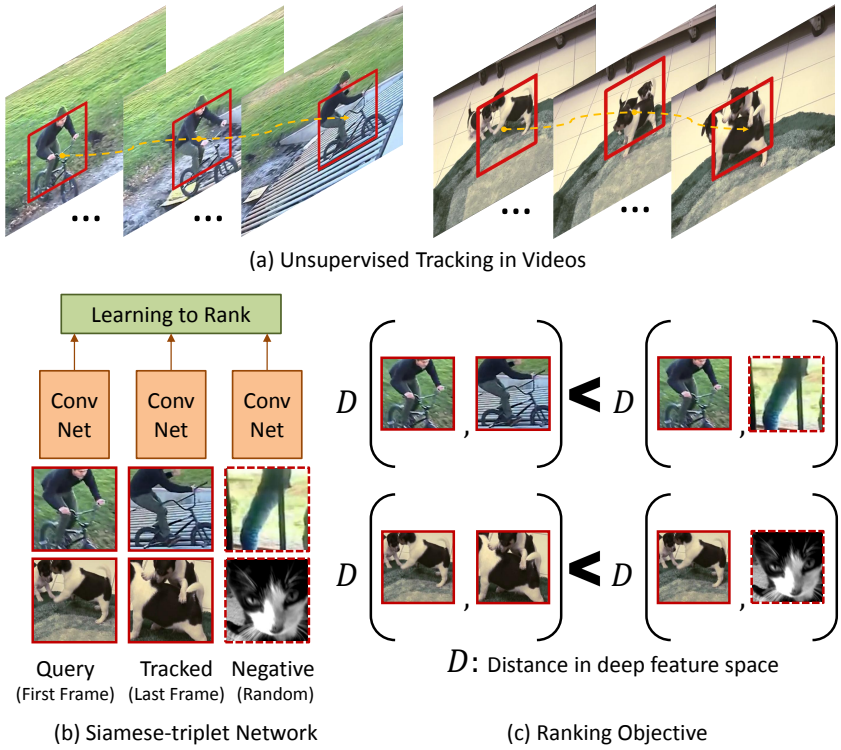



Figure 3.1: Overview of our approach. (a) Given unlabeled videos, we perform unsupervised tracking on the patches in them. (b) Triplets of patches including query patch in the initial frame of tracking, tracked patch in the last frame, and random patch from other videos are fed into our siamese-triplet network for training. (c) The learning objective: Distance between the query and tracked patch in feature space should be smaller than the distance between query and random patches.

we have similar learning capabilities for ConvNets?

We present a simple yet surprisingly powerful approach for unsupervised learning of ConvNets using hundreds and thousands of unlabeled videos from the web. Visual tracking is one of the first capabilities that develops in infants and often before semantic representations are learned[1]. Taking a leaf from this observation, we propose to exploit visual tracking for learning ConvNets in an unsupervised manner. Specifically, we track millions of "moving" patches in hundreds of thousands of videos. Our key idea is that two patches connected by a track should have similar visual representation in deep feature space since they probably belong to same object. We design a Siamese-triplet network with ranking loss function to train the ConvNet representation. This ranking loss function enforces that in the final deep feature space the first frame patch should be much closer to the tracked patch than any other randomly sampled patch. We demonstrate the strength of our learning algorithm using extensive experimental evaluation. Without using a single image

[1]http://www.aoa.org/patients-and-public/good-vision-throughout-life/childrens-vision/infant-vision-birth-to-24-months-of-age

from ImageNet [259], just using 100K unlabeled videos and VOC 2012 dataset, we train an ensemble of AlexNet networks that achieves $52\%$ mAP (no bounding box regression) . This performance is similar to its ImageNet-supervised counterpart, an ensemble which achieves a mAP of $54.4\%$. We also show that our network trained using unlabeled videos also achieves similar performance to its completely supervised counterpart on other tasks such as surface normal estimation. To the best of our knowledge, the results reported in this chapter come closest to standard supervised ConvNets approaches (which use millions of semantically-labeled images) among unsupervised approaches.

## 3.1 Background

Unsupervised learning of visual representations has a rich and diverse history starting from original auto-encoders work of Olhausen and Field [231] and early generative models. Most of the work in the area of unsupervised learning can be broadly divided into three categories. The first class of unsupervised learning algorithms focus on learning generative models with strong priors [122, 295, 351]. These algorithms essentially capture co-occurence statistics of features. The second class of algorithms use manually defined features such as SIFT or HOG and perform clustering over training data to discover semantic classes [262, 284]. Some of these recent algorithms also focus on learning mid-level representations rather than discovering semantic classes themselves [45, 46, 283].

The third class of algorithms and more related to our chapter is unsupervised learning of visual representations from the pixels themselves using deep learning approaches [14, 57, 120, 183, 189, 214, 269, 289, 302, 316]. Starting from the seminal work of Olhausen and Field [231], the goal is to learn visual representations which are (a) sparse and (b) reconstructive. Olhausen and Field [231] showed that using this criteria they can learn V1-like filters directly from the data. However, this work only focused on learning a single layer. This idea was extended by Hinton and Salakhutdinov [120] to train a deep belief network in an unsupervised manner via stacking layer-by-layer RBMs. Similar to this, Bengio et al. [15] investigated stacking of both RBMs and autoencoders. As a next step, Le et al. [183] scaled up the learning of multi-layer autoencoder on large-scale unlabeled data. They demonstrated that although the network is trained in an unsupervised manner, the neurons in high layers can still have high responses on semantic objects such as human heads and cat faces. Sermanet et al. [269] applied convolutional sparse coding to pre-train the model layer-by-layer in unsupervised manner. The model is then fine-tuned for the pedestrian detection task on the labeled datasets.

However, it is not clear if static images is the right way to learn visual representations. Therefore, researchers have started focusing on learning representations using videos [96, 185, 225, 291, 304, 404]. Early work such as [404] focused on inclusion of constraints via video to autoencoder framework. The most common constraint is the smoothing constraints which enforces learned representations to be temporally smooth. Similar to this, Goroshin et al. [96] proposed to learn auto-encoders based on the slowness prior. Other approaches such as Taylor et al. [304] trained convolutional gated RBMs to learn latent representations from pairs of successive images. This was extended in a recent work by Srivastava et al. [291] where they proposed to learn a LSTM model in an unsupervised manner. Given a few consecutive frames, their optimization goal for LSTM model includes reconstructing the given frames and predicting the future frames. Our work differs from this body of work

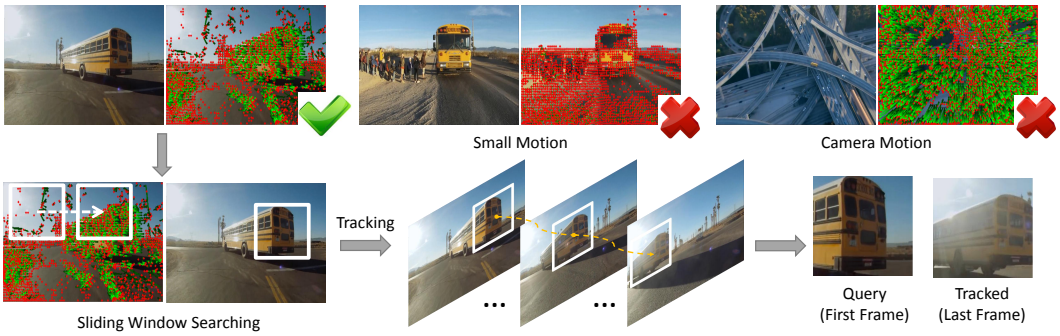



Figure 3.2: The process of patch mining in videos. Given the video about buses (the "bus" label are not utilized), we perform IDT on it. On the top three pairs of images, red points represents the SURF feature points, green represents the trajectories for the points. We reject the frames with small motions (middle pairs) as well as frames with large camera motion as the camera zoom in (right pairs). Given the selected frame, we perform sliding window on it to find the bounding box containing most of the moving SURF points. As illustrated in the bottom line, given the initial bounding box in the frame, we perform tracking along the video for 30 frames. The query patch in the first frame and tracked patch in the last frame are collected as one pair of training samples.

in two aspects: (a) We train our model with patches obtained from tracking; (b) Instead of training auto-encoders, we train a deep ConvNet which can be transferred to different challenging vision tasks.

Finally, our work is also related to metric learning via deep networks [33, 93, 105, 126, 207, 328]. For example, Chopra et al. [33] proposed to learn convolutional networks in a siamese architecture for face verification. Wang et al. [328] introduced a deep triplet ranking network to learn fine-grained image similarity. However, all these methods required labeled data. Our work is also related to [201], which used ConvNets pre-trained on ImageNet classification and detection dataset as initialization, and performed semi-supervised learning in videos to solve object detection in target domain. However, in our work, we propose an unsupervised approach instead of semi-supervised algorithm.

## 3.2 Overview

Our goal is to train convolutional neural networks using hundreds and thousands of unlabeled videos from the Internet. We follow the AlexNet architecture [171] to design our base network. However, since we do not have labels, it is not clear what should be the loss function and how we should optimize it. But in case of videos, we have another supervisory information: time. For example, we all know that the scene does not change drastically within a short time in a video and same object instances appear in multiple frames of the video. So, how do we exploit this information to train a ConvNet-based representation?

We sample millions of patches in these videos and track them over time. Since we are tracking these patches, we know that the first and last tracked frames correspond to the same instance of the moving object or object part. Therefore, any visual representation that we learn should keep these two data points close in the feature space. But just using

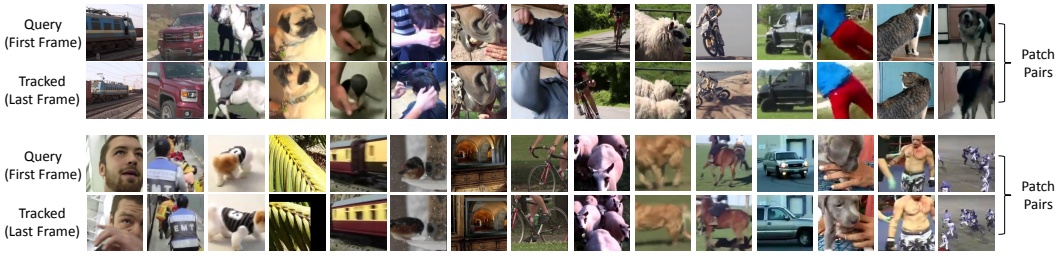



Figure 3.3: Examples of patch pairs we obtain via patch mining in the videos.

this constraint is not sufficient: all points can be mapped to a single point in feature space. Therefore, for training our ConvNet, we sample a third patch which creates a triplet. For training, we use a loss function [328] that enforces that the first two patches connected by tracking are closer in feature space than the first one and a random one.

But training a network with such triplets converges fast since the task is easy to overfit to. One way is to increase the number of training triplets. However, after initial convergence most triplets satisfy the loss function and therefore back-propagating gradients using such triplets is inefficient. Instead, analogous to hard-negative mining, we select the third patch from multiple patches that violates the constraint (loss is maximum). Selecting this patch leads to more meaningful gradients for faster learning.

## 3.3 Patch Mining in Videos

The first step in our learning procedure is to extract training instances from videos. In our case, every training instance for learning the deep network consists of three patches. The loss function enforces the pairs of patches connected by tracks to have more similar representations as compared to any other two randomly selected patches. But what do these patches that are tracked correspond to? Since our videos are unlabeled, the location and the extent of the objects in the frame are unknown. Therefore, instead of trying to extract patches corresponding to semantic objects, we focus on using motion information in the video to extract moving image patches. Note that these patches might contain objects or part of an object as shown in Figure 3.2. These patches are then tracked over time to obtain a second patch. The initial and tracked patches are then grouped together. The details are explained below.

Given a video, we want to extract patches of interest (patches with motion in our case) and track these patches to create training instances. One obvious way to find patches of interest is to compute optical flow and use the high magnitude flow regions. However, since YouTube videos are noisy with a lot of camera motion, it is hard to localize moving objects using simple optical flow magnitude vectors. Thus we follow a two-step approach: in the first step, we obtain SURF [11] interest points and use Improved Dense Trajectories (IDT) [323] to obtain motion of each SURF point. Note that since IDT applies a homography estimation (video stabilization) method, it reduces the problem caused by camera motion. Given the trajectories of SURF interest points, we classify these points as moving if the flow magnitude is more than 0.5 pixels. We also reject frames if (a) very few ($< 25\%$) SURF interest points are classified as moving because it might be just noise; (b) majority of SURF interest

points ($> 75\%$) are classified as moving as it corresponds to moving camera. Once we have extracted moving SURF interest points, in the second step, we find the best bounding box such that it contains most of the moving SURF points. The size of the bounding box is set as $h \times w$, and we perform sliding window with it in the frame. We take the bounding box which contains the most number of moving SURF interest points as the interest bounding box. In the experiment, we set $h = 227, w = 227$ in the frame with size $448 \times 600$.

**Tracking.** Given the initial bounding box, we perform tracking using the KCF tracker [118]. After tracking along 30 frames in the video, we obtain the second patch. This patch acts as the similar patch to the query patch in the triplet. Note that the KCF tracker does not use any supervised information except for the initial bounding box.

## 3.4 Learning Via Videos

In the previous section, we discussed how we can use tracking to generate pairs of patches where the first patch (query) is initialized based on motion and the second patch is obtained after tracking the query patch for 30 frames. We use this procedure to generate millions of such pairs (See Figure 3.3 for examples of pairs of patches mined). We now describe how we use these as training instances for our visual representation learning.

### Siamese Triplet Network

Our goal is to learn a feature space such that the query patch is closer to the tracked patch as compared to any other randomly sampled patch. To learn this feature space we design a Siamese-triplet network. A Siamese-triplet network consist of three base networks which share the same parameters (see Figure 3.4). For our experiments, we take the image with size $227 \times 227$ as input. The base network is based on the AlexNet architecture [171] for the convolutional layers. Then we stack two full connection layers on the pool5 outputs, whose neuron numbers are 4096 and 1024 respectively. Thus the final output of each single network is $1024$ dimensional feature space $f(\cdot)$. We define the loss function on this feature space.

### Ranking Loss Function

Given the set of patch pairs $\mathbb{S}$ sampled from the video, we propose to learn an image similarity model in the form of ConvNet. Specifically, given an image $X$ as an input for the network, we can obtain its feature in the final layer as $f(X)$. Then, we define the distance of two image patches $X_1, X_2$ based on the cosine distance in the feature space as,

$$D(X_1, X_2) = 1 - \frac{f(X_1) \cdot f(X_2)}{\|f(X_1)\| \|f(X_2)\|}. \tag{3.1}$$

We want to train a ConvNet to obtain feature representation $f(\cdot)$, so that the distance between query image patch and the tracked patch is small and the distance between query patch and other random patches is encouraged to be larger. Formally, given the patch set $\mathbb{S}$, where $X_i$ is the original query patch (first patch in tracked frames), $X_i^+$ is the tracked patch and $X_i^-$ is a random patch, we want to enforce $D(X_i, X_i^-) > D(X_i, X_i^+)$.

Given a triplet of image patches $X_i, X_i^+, X_i^-$ as input, where $X_i, X_i^+$ is a tracked pair and $X_i^-$ is obtained from a different video, the loss of our ranking model is defined by hinge

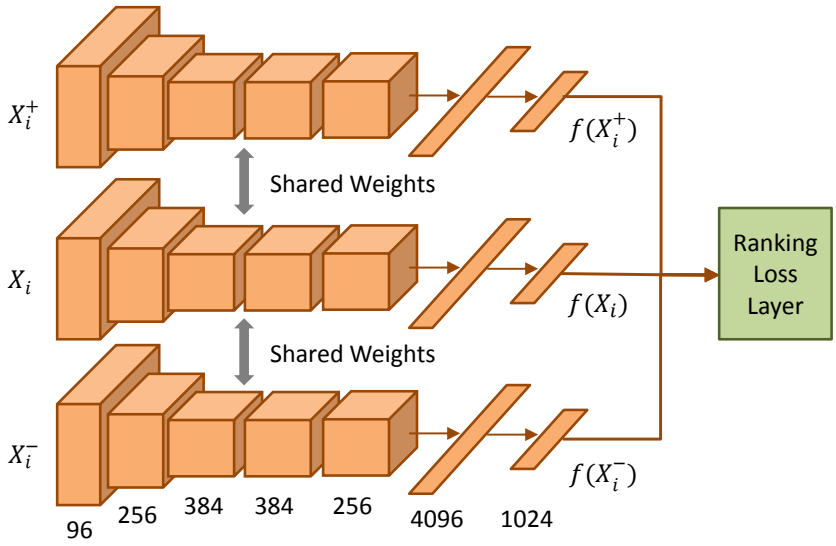



Figure 3.4: Siamese-triplet network. Each base network in the Siamese-triplet network share the same architecture and parameter weights. The architecture is rectified from AlexNet by using only two full connection layers. Given a triplet of training samples, we obtain their features from the last layer by forward propagation and compute the ranking loss.

loss as,

$$L(X_i, X_i^+, X_i^-) = \max\{0, D(X_i, X_i^+) - D(X_i, X_i^-) + M\}, \tag{3.2}$$

where $M$ represents the gap parameters between two distances. We set $M = 0.5$ in the experiment. Then our objective function for training can be represented as,

$$\min_W \frac{\lambda}{2} \parallel W \parallel_2^2 + \sum_{i=1}^{N} \max\{0, D(X_i, X_i^+) - D(X_i, X_i^-) + M\}, \tag{3.3}$$

where $W$ is the parameter weights of the network, i.e., parameters for function $f(\cdot)$. $N$ is the number of the triplets of samples. $\lambda$ is a constant representing weight decay, which is set to $\lambda = 0.0005$.

## Hard Negative Mining for Triplet Sampling

One non-trivial part for learning to rank is the process of selecting negative samples. Given a pair of similar images $X_i, X_i^+$, how can we select the patch $X_i^-$, which is a negative match to $X_i$, from the large pool of patches? Here we first select the negative patches randomly, and then find hard examples (in a process analogous to hard negative mining).

**Random Selection:** During learning, we perform mini-batch Stochastic Gradient Descent (SGD). For each $X_i, X_i^+$, we randomly sample $K$ negative matches in the same batch $\mathbb{B}$, thus we have $K$ sets of triplet of samples. For every triplet of samples, we calculate the gradients over three of them respectively and perform back propagation. Note that we shuffle all the

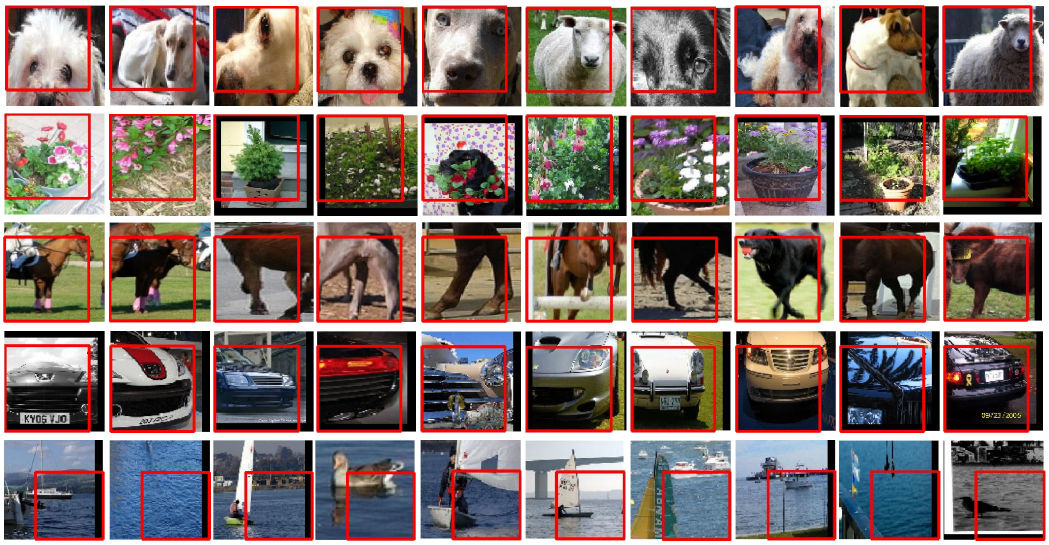


Figure 3.5: Top regions for the pool5 neurons for the base network trained in unsupervised manner. The receptive field for the pool5 neurons is $195 \times 195$ pixels. We use the red boxes to represent the receptive regions giving the neuron high responses. We show regions with top responses for 5 neurons in 5 rows.

images randomly after each epoch of training, thus the pair of patches $X_i, X_i^+$ can look at different negative matches each time.

**Hard Negative Mining:** While one can continue to sample random patches for creating the triplets, it is more efficient to search the negative patches smartly. After 10 epochs of training using negative data selected randomly, we want to make the problem harder to get more robust feature representations. Analogous to hard-negative mining procedure in SVM, where gradient descent learning is only performed on hard-negatives (not all possible negative), we search for the negative patch such that the loss is maximum and use that patch to compute and back propagate gradients.

Specifically, the sampling of negative matches is similar as random selection before, except that this time we select according to the loss(Eq. 3.2). For each pair $X_i, X_i^+$, we calculate the loss of all other negative matches in batch $\mathbb{B}$, and select the top $K$ ones with highest losses. We apply the loss on these $K$ negative matches as our final loss and calculate the gradients over them. Notice that since the feature of each sample is already computed after the forward propagation, we only need to calculate the loss over these features, thus the extra computation for hard negative mining is very small. For the experiments in this chapter, we use $K = 4$.

## Adapting for Supervised Tasks

Given the ConvNet learned by using unsupervised data, we want to transfer the learned representations to the tasks with supervised data. In our experiments, we apply our model to two different tasks including object detection and surface normal estimation. In both tasks

we take the base network from our Siamese-triplet network (which is based on AlexNet architecture) and adjust the full connection layers and outputs accordingly. We introduce two ways to fine-tune and transfer the information obtained from unsupervised data to supervised learning.

One straight forward approach is directly applying our ranking model as a pre-trained network for the target task. More specifically, we use the parameters of the convolutional layers in the base network of our triplet architecture as initialization for the target task. For the full connection layers, we initialize them randomly. This method of transferring feature representation is very similar to the approach applied in RCNN [83]. However, RCNN uses the network pre-trained with ImageNet Classification data. In our case, the unsupervised ranking task is quite different from object detection and surface normal estimation. Thus, we need to adapt the learning rate to the fine-tuning procedure introduced in RCNN. We start with the learning rate with $\epsilon = 0.01$ instead of $0.001$ and set the same learning rate for convolutional layers and full connection layers. This setting is crucial since we want the pre-trained features to be used as initialization of supervised learning, and adapting the features to the new task.

In this chapter, we explore one more approach to transfer/fine-tune the network. Specifically, we note that there might be more juice left in the millions of unsupervised training data (which could not be captured in the initial learning stage). Therefore, we use an iterative fine-tuning scheme. Given the initial unsupervised network, we first fine-tune using the PASCAL VOC data. Given the new fine-tuned network, we use this network to re-adapt to ranking triplet task. Here we again transfer convolutional parameters for re-adapting. Finally, this re-adapted network is fine-tuned on the VOC data yielding a better trained model. We show in the experiment that this circular approach gives improvement in performance. We also notice that after two iterations of this approach the network converges.

## Model Ensemble

We proposed an approach to learn ConvNets using unlabeled videos. However, there is absolutely no limit to generating training instances and pairs of tracked patches (YouTube has more than billions of videos). This opens up the possibility of training multiple ConvNets using different sets of data. Once we have trained these ConvNets, we append the fc7 features from each of these ConvNets to train the final SVM. Note that the ImageNet trained models also provide initial boost for adding more networks (See Table 3.1).

## Implementation Details

We apply mini-batch SGD in training. As the 3 networks share the same parameters, instead of inputting 3 samples to the triplet network each time, we perform the forward propagation for the whole batch by a single network and calculate the loss based on the output feature. Given a pair of patches $X_i, X_i^+$, we randomly select another patch $X_i^- \in \mathbb{B}$ which is extracted in a different video from $X_i, X_i^+$. Given their features from forward propagation $f(X_i), f(X_i^+), f(X_i^-)$, we can compute the loss according to Eq. 3.2.

For learning, we download 100K videos from YouTube using the URLs provided by [201]. By performing our patch mining method on the videos, we obtain 8 million image patches. We train three different networks separately using 1.5M, 1.5M and 5M training instances. Therefore, we report number based on these three networks. To train our siamese-triplet

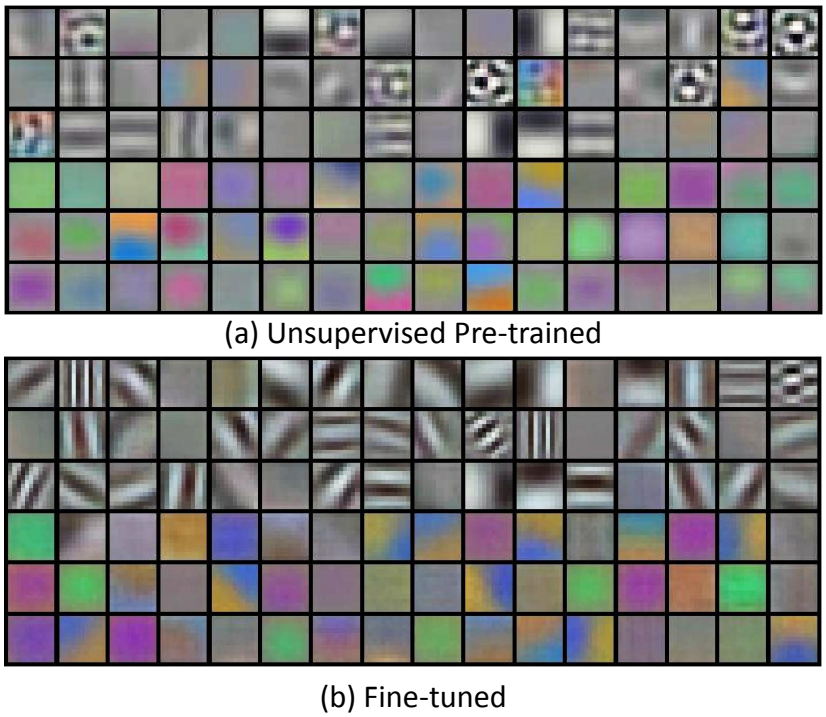



Figure 3.6: Conv1 filters visualization. (a) The filters of the first convolutional layer of the siamese-triplet network trained in unsupervised manner. (b) By fine-tuning the unsupervised pre-trained network on PASCAL VOC 2012, we obtain sharper filters.

networks, we set the batch size as $|\mathbb{B}| = 100$, the learning rate starting with $\epsilon_0 = 0.001$. For the dataset with 1.5M and 5M patches, we first trained our network with random negative samples with this learning rate for 150K iterations, and then we apply hard negative mining based on it. For training on 1.5M patches, we reduce the learning rate by a factor of 10 at every 80K iterations and train for 240K iterations. For training on 5M patches, we reduce the learning rate by a factor of 10 at every 120K iterations and train for 350K iterations.

## 3.5 Experiments

We demonstrate the quality of our learned visual representations with qualitative and quantitative experiments. Qualitatively, we show the convolutional filters learned in layer 1 (See Figure 3.6). Our learned filters are similar to V1 though not as strong. However, after fine-tuning on PASCAL VOC 2012, these filters become quite strong. We also show that the underlying representation is reasonable by showing what the neurons in Pool5 layers represent (See Figure 3.5). We use the red bounding boxes to represent the receptive field with top responses for five different neurons(one neuron each line). We can see the clusters represented by these neurons are quite reasonable and correspond to semantic parts of objects. For example, the first neuron represents animal heads, second represents potted plant, etc.

| VOC 2012 test | external | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| scratch | 0 | 66.1 | 58.1 | 32.7 | 23.0 | 21.8 | 54.5 | 56.4 | 50.8 | 21.6 | 42.2 | 31.8 | 49.2 | 49.8 | 61.6 | 52.1 | 25.1 | 52.6 | 31.3 | 50.0 | 49.1 | 44.0 |
| unsup + ft | 1.5M | 68.8 | 62.1 | 34.7 | 25.3 | 26.6 | 57.7 | 59.6 | 56.3 | 22.0 | 42.6 | 33.8 | 52.3 | 50.3 | 65.6 | 53.9 | 25.8 | 51.5 | 32.3 | 51.7 | 51.8 | 46.2 |
| unsup + ft | 5M | 69.0 | 64.0 | 37.1 | 23.6 | 24.6 | 58.7 | 58.9 | 59.6 | 22.3 | 46.0 | 35.1 | 53.3 | 53.7 | 66.9 | 54.1 | 25.4 | 52.9 | 31.2 | 51.9 | 51.8 | 47.0 |
| unsup + ft (2 ensemble) | 6.5M | 72.4 | 66.2 | 41.3 | 26.4 | 26.8 | 61.0 | 61.9 | 63.1 | 25.3 | 51.0 | 38.7 | 58.1 | 58.3 | 70.0 | 56.2 | 28.6 | 56.1 | 38.5 | 55.9 | 54.3 | 50.5 |
| unsup + ft (3 ensemble) | 8M | 73.5 | 67.8 | 43.5 | 28.9 | 27.9 | **62.3** | **62.6** | 64.9 | 27.3 | 51.5 | 41.6 | 59.1 | 60.0 | 71.8 | **58.3** | **29.7** | 56.1 | 39.1 | **58.6** | 55.6 | 52.0 |
| unsup + iterative ft | 5M | 67.7 | 64.0 | 41.3 | 25.3 | 27.3 | 58.8 | 60.3 | 60.2 | 24.3 | 46.7 | 34.4 | 53.6 | 53.8 | 68.2 | 55.7 | 26.4 | 51.1 | 34.3 | 53.4 | 52.3 | 48.0 |
| RCNN 70K | | 72.7 | 62.9 | 49.3 | 31.1 | 25.9 | 56.2 | 53.0 | 70.0 | 23.3 | 49.0 | 38.0 | 69.5 | 60.1 | 68.2 | 46.4 | 17.5 | 57.2 | 46.2 | 50.8 | 54.1 | 50.1 |
| RCNN 70K (2 ensemble) | | **75.3** | 68.3 | 53.1 | 35.2 | 27.7 | 59.6 | 54.7 | 73.4 | 26.5 | 53.0 | 42.2 | 73.1 | **66.1** | 71.0 | 48.5 | 21.7 | 59.2 | 50.8 | 55.2 | **58.0** | 53.6 |
| RCNN 70K (3 ensemble) | | 74.6 | **68.7** | **54.9** | **35.7** | 29.4 | 61.0 | 54.4 | **74.0** | **28.4** | **53.6** | **43.0** | **74.0** | **66.1** | **72.8** | 50.3 | 20.5 | **60.0** | **51.2** | 57.9 | **58.0** | **54.4** |
| RCNN 200K (big stepsize) | | 73.3 | 67.1 | 46.3 | 31.7 | **30.6** | 59.4 | 61.0 | 67.9 | 27.3 | 53.1 | 39.1 | 64.1 | 60.5 | 70.9 | 57.2 | 26.1 | 59.0 | 40.1 | 56.2 | 54.9 | 52.3 |

Table 3.1: mean Average Precision (mAP) on VOC 2012. The second column "external" represents the number of patches used to pre-train the model in the unsupervised manner.

For quantitative evaluations, we evaluate our approach by transferring the feature representation learned in unsupervised manner to the tasks with labeled data. We focus on two challenging problems: object detection and surface normal estimation.

## Object Detection

For object detection, we perform our experiments on PASCAL VOC 2012 dataset [59]. We follow the detection pipeline introduced in RCNN [83], which borrowed the ConvNets pre-trained on other datasets and fine-tuned on it using the VOC data. The fine-tuned ConvNet was then used to extract features followed by training SVMs for each object class. However, instead of using ImageNet pre-trained network as initialization in RCNN, we use our ConvNets trained in the unsupervised manner. Note that the network architecture is based on AlexNet. We fine-tune our network with the trainval set (11540 images) and train SVMs with the same images. Evaluation is performed in the standard test set (10991 images).

At the fine-tuning stage, we change the output to $21$ and initialize the convolutional layers with our unsupervised pre-trained network. To fine-tune the network, we start with learning rate as $\epsilon = 0.01$ and reduce the learning rate by a factor of 10 at every 80K iterations. The network is fine-tuned for 200K iterations. Note that for all the experiments, no bounding box regression is performed.

We compare our method with the model trained from scratch as well as using ImagNet pre-trained network. Notice that the results for VOC 2012 reported in RCNN [83] are obtained by only fine-tuning on the train set without using the val set. The mAP for VOC 2012 reported in [83] is $49.6\%$. For fair comparison, we fine-tuned the ImageNet pre-trained network with VOC 2012 trainval set. Moreover, as the step size of reducing learning rate in RCNN [83] is set to 20K and iterations for fine-tuning is 70K, we also try to enlarge the step size to 50K and fine-tune the network for 200K iterations. We report the results for both of these settings.

**Single Model.** We show the results in Table 3.1. As a baseline, we train the network from scratch on VOC 2012 dataset and obtain $44\%$ mAP. Using our unsupervised network pre-trained with 1.5M pair of patches and then fine-tuned on VOC 2012, we obtain mAP of $46.2\%$ (unsup+ft, external data = 1.5M). By looking into more data, using 5M patches in pre-training and then fine-tune, we can achieve $47\%$ mAP (unsup+ft, external data = 5M). These results indicate that our unsupervised network provides a significant boost as compared to the scratch network. More importantly, when more unlabeled data is applied, we can get better performance ( $3\%$ boost compared to training from scratch).

**Model Ensemble.** As looking at more external data in unsupervised pre-training gives

| | (Lower Better) | | (Higher Better) | | |
|---|---|---|---|---|---|
| | Mean | Median | 11.25° | 22.5° | 30° |
| scratch | 38.6 | 26.5 | 33.1 | 46.8 | 52.5 |
| unsup + ft | 35.2 | 23.2 | 34.9 | 49.4 | 55.8 |
| ImageNet + ft | 33.3 | 20.8 | 36.7 | 51.7 | 58.1 |
| UNFOLD [69] | 35.1 | 19.2 | 37.6 | 53.3 | 58.9 |
| Discr. [176] | 32.5 | 22.4 | 27.4 | 50.2 | 60.2 |
| 3DP (MW) [68] | 36.0 | 20.5 | 35.9 | 52.0 | 57.8 |

Table 3.2: Results on NYU v2 for per-pixel surface normal estimation, evaluated over valid pixels.

the boost in performance, we also try combining different models using different unlabeled data in pre-training. By ensembling two fine-tuned networks which are pre-trained using 1.5M and 5M patches, we obtained a boost of $3.5\%$ comparing to the single model, which is $50.5\%$(unsup+ft (2 ensemble)). By moving one step forward, we ensemble all three different networks pre-trained with different sets of data, whose size are 1.5M, 1.5M and 5M respectively. We get another boost of $1.5\%$ and reach $52\%$ mAP(unsup+ft (3 ensemble)).

**Baselines.** We also compare our approach with RCNN [83] which uses ImageNet pre-trained models. Following the procedure in [83], we obtain $50.1\%$ mAP (RCNN 70K) by setting the step size to 20K and fine-tuning for 70K iterations. To generate a model ensemble, three ConvNets are first trained on the ImageNet dataset separately, and then they are fine-tuned with the VOC 2012 dataset. The result of ensembling two of these networks is $53.6\%$ mAP (RCNN 70K (2 ensemble )), and ensembling three of these networks gives $0.8\%$ improvement, leading to $54.4\%$ mAP (RCNN 70K (3 ensemble )). For fair of comparison, we also fine-tuned the ImageNet pre-trained model with larger step size(50K) and more iterations(200K). The result is $52.3\%$ mAP (RCNN 200K (big stepsize)). Note that while ImageNet network shows diminishing returns with ensembling since the training data remains similar, in our case since every network in the ensemble looks at different sets of data, we get huge performance boosts.

**Exploring a better way to transfer learned representation.** Given our fine-tuned model using 5M patches in pre-training (unsup+ft, external = 5M), we use it to re-learn and re-adapt to the unsupervised triplet task. After that, the network is re-applied to fine-tune on VOC 2012. We repeat this iterative approach twice in our experiment and find it converges very quickly. The final result for this single model is $48\%$ mAP (unsup + iterative ft), which is $1\%$ better than the initial fine-tuned network.

## Surface Normal Estimation

To illustrate that our unsupervised representation can be generalized to different tasks, we adapt the unsupervised ConvNet to the task of surface normal estimation from a RGB image. In this task, we want to estimate the orientation of the pixels. We perform our experiments on the NYUv2 dataset [274], which includes 795 images for training and 654 images for testing. Each image is has corresponding depth information which can be used to generate groundtruth surface normals. For evaluation and generating the groundtruth, we adopt the protocols introduced in [68] which is used by different methods [68, 69, 176] on this task.

Figure 3.7: Surface normal estimation results on NYU dataset. For visualization, we use green to represent horizontal surface, blue representing facing right and red representing facing left, i.e., blue $\rightarrow$ X; green $\rightarrow$ Y; red $\rightarrow$ Z.

To apply deep learning to this task, we followed the same form of outputs and loss function as the coarse network mentioned in [336]. Specifically, we first learn a codebook by performing k-means on surface normals and generate 20 codewords. Each codeword represents one class and thus we transform the problem to 20-class classification for each pixel. Given a $227 \times 227$ image as input, our network generates surface normals for the whole scene. The output of our network is $20 \times 20$ pixels, each of which is represented by a distribution over 20 codewords. Thus the dimension of output is $20 \times 20 \times 20 = 8000$.

The network architecture for this task is also based on the AlexNet. To relieve over-fitting, we only stack two full connection layers with $4096$ and $8000$ neurons on the pool5 layer. During training, we initialize the network with the unsupervised pre-trained network. We use the same learning rate $1.0 \times 10^{-6}$ as mentioned in [336] and fine-tune the network with 10K iterations given the small number of training data. Note that unlike [336], we do not utilize any data from the videos in NYU dataset for training.

For comparisons, we also trained networks from scratch as well as using ImageNet pre-trained. We show our results in Table 3.2. Compared to the recent results which do not use external data [68, 69, 176], we show that we can get reasonable results even using this small amounts of training data. Our approach(unsup + ft) is generally $2 \sim 3\%$ better than network trained from scratch in 5 different metrics. We show a few qualitative results in Figure 3.7.

# Chapter 4

# Transitive Invariance for Self-supervised Learning

In Chapter 3, we have introduced a self-supervised method for learning deep representation which is invariant to object viewpoint and deformation changes. This type of visual invariance is actually a core issue in learning visual representations. Modern deep representations are capable of learning high-level invariance from large-scale data [259] , *e.g.*, viewpoint, pose, deformation, and semantics. These can also be transferred to complicated visual recognition tasks [84, 210].

In the scheme of supervised learning, human annotations that map a variety of examples into a single label provide supervision for learning invariant representations. For example, two horses with different illumination, poses, and breeds are invariantly annotated as a category of "horse". Such human knowledge on invariance is expected to be learned by capable deep neural networks [170, 188] through carefully annotated data. However, large-scale, high-quality annotations come at a cost of expensive human effort.

Unsupervised or "self-supervised" learning (*e.g.*, [48, 199, 234, 239, 343, 384, 385]) recently has attracted increasing interests because the "labels" are free to obtain. Unlike supervised learning that learns invariance from the semantic labels, the self-supervised learning scheme mines it from the nature of the data. We observe that most self-supervised approaches learn representations that are invariant to: (i) *inter-instance* variations, which reflects the commonality among different instances. For example, relative positions of patches [48] (see also Figure 4.3) or channels of colors [384, 385] can be predicted through the commonality shared by many object instances; (ii) *intra-instance* variations. Intra-instance invariance is learned from the pose, viewpoint, and illumination changes by tracking a single moving instance in videos (Chapter 3). However, either source of invariance can be as rich as that provided by human annotations on large-scale datasets like ImageNet.

Even after significant advances in the field of self-supervised learning, there is still a long way to go compared to supervised learning. What should be the next steps? It seems that an obvious way is to obtain multiple sources of invariance by combining multiple self-supervised tasks, *e.g.*, via multiple losses. Unfortunately, this naïve solution turns out to give little improvement (as we will show by experiments).

We argue that the trick lies not in the tasks but in the way of exploiting data. To leverage

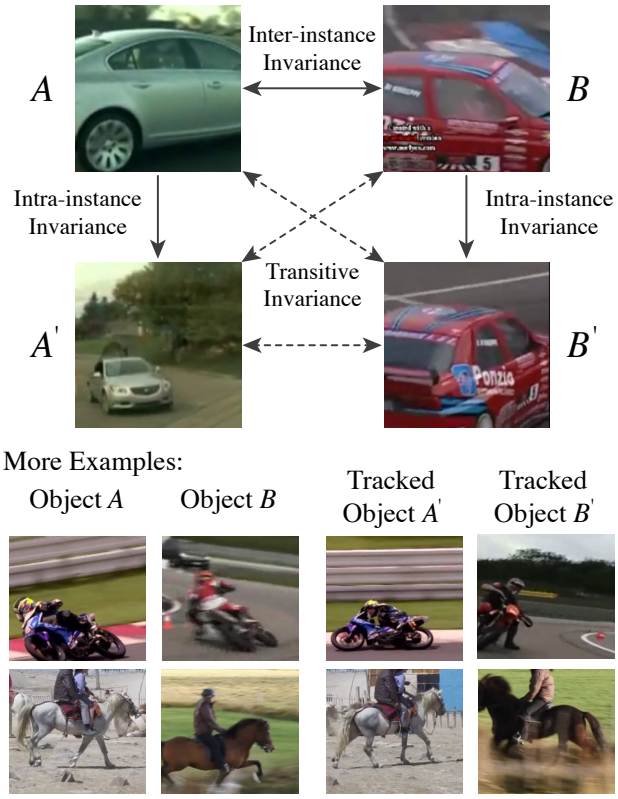



Figure 4.1: We propose to obtain rich invariance by applying simple transitive relations. In this example, two different cars $A$ and $B$ are linked by the features that are good for inter-instance invariance (*e.g.*, using [48]); and each car is linked to another view ($A'$ and $B'$) by visual tracking [343]. Then we can obtain new invariance from object pairs $\langle A, B' \rangle$, $\langle A', B \rangle$, and $\langle A', B' \rangle$ via transitivity. We show more examples in the bottom.

both intra-instance and inter-instance invariance, in this chapter we construct a huge affinity graph consisting of two types of edges (see Figure 4.1): the first type of edges relates "different instances of similar viewpoints/poses and potentially the same category", and the second type of edges relates "different viewpoints/poses of an identical instance". We instantiate the first type of edges by learning commonalities across instances via the approach of [48], and the second type by unsupervised tracking of objects in videos [343]. We set up simple transitive relations on this graph to infer more complex invariance from the data, which are then used to train a Triplet-Siamese network for learning visual representations.

Experiments show that our representations learned without any annotations can be well transferred to the object detection task. Specifically, we achieve $63.2\%$ mAP with VGG16 [281] when fine-tuning Fast R-CNN on VOC2007, against the ImageNet pre-training baseline of $67.3\%$. More importantly, we also report the first-ever result of un-/self-supervised pre-training models fine-tuned on the challenging COCO object detection dataset [205], achieving $23.5\%$ AP comparing against $24.4\%$ AP that is fine-tuned from an ImageNet pre-trained counterpart (both using VGG16). To our knowledge, this is the closest accuracy to the

ImageNet pre-training counterpart obtained on object detection tasks.

## 4.1 Background

Unsupervised learning of visual representations is a research area of particular interest. Approaches to unsupervised learning can be roughly categorized into two main streams: (i) generative models, and (ii) self-supervised learning. Earlier methods for generative models include Anto-Encoders [184, 190, 231, 317] and Restricted Boltzmann Machines (RBMs) [15, 58, 121, 303]. For example, Le *et al*. [184] trained a multi-layer auto-encoder on a large-scale dataset of YouTube videos: although no label is provided, some neurons in high-level layers can recognize cats and human faces. Recent generative models such as Generative Adversarial Networks [94] and Variational Auto-Encoders [160] are capable of generating more realistic images. The generated examples or the neural networks that learn to generate examples can be exploited to learn representations of data [51, 53].

Self-supervised learning is another popular stream for learning invariant features. Visual invariance can be captured by the same instance/scene taken in a sequence of video frames [2, 141, 199, 219, 234, 292, 320, 343]. For example, Wang and Gupta [343] leverage tracking of objects in videos to learn visual invariance within individual objects; Jayaraman and Grauman [141] train a Siamese network to model the ego-motion between two frames in a scene; Mathieu *et al*. [219] propose to learn representations by predicting future frames; Pathak *et al*. [234] train a network to segment the foreground objects where are acquired via motion cues. On the other hand, common characteristics of different object instances can also be mined from data [48, 384, 385]. For example, relative positions of image patches [48] may reflect feasible spatial layouts of objects; possible colors can be inferred [384, 385] if the networks can relate colors to object appearances. Rather than rely on temporal changes in video, these methods are able to exploit still images.

Our work is also closely related to mid-level patch clustering [45, 46, 283] and unsupervised discovery of semantic classes [262, 284] as we attempt to find reliable clusters in our affinity graph. In addition, the ranking function used in this chapter is related to deep metric learning with Siamese architectures [34, 93, 106, 127, 329].

**Analysis of the two types of invariance.** Our generic framework can be instantiated by any two self-supervised methods that can respectively learn inter-/intra-instance invariance. In this chapter we adopt Doersch *et al*.'s [48] context prediction method to build inter-instance invariance, and Wang and Gupta's [343] tracking method to build intra-instance invariance. We analyze their behaviors as follows.

The context prediction task in [48] randomly samples a patch (blue in Figure 4.3) and one of its eight neighbors (red), and trains the network to predict their relative position, defined as an 8-way classification problem. In the first two examples in Figure 4.3, the context prediction model is able to predict that the "leg" patch is below the "face" patch of the cat, indicating that the model has learned some commonality of spatial layout from the training data. However, the model would fail if the pose, viewpoint, or deformation of the object is changed drastically, *e.g*., in the third example of Figure 4.3 — unless the dataset is diversified and large enough to include gradually changing poses, it is hard for the models to learn that the changed pose can be of the same object type.

On the other hand, these changes can be more successfully captured by the visual tracking

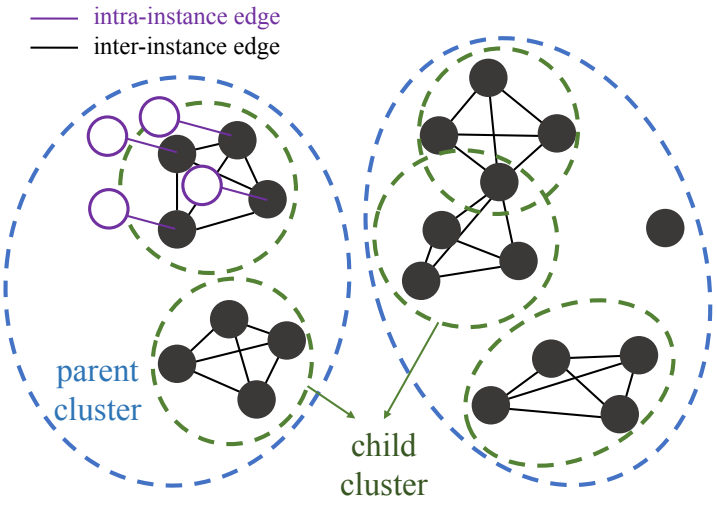



Figure 4.2: Illustrations for our graph construction. We first cluster the object nodes into coarser clusters (namely "parent" clusters) and then inside each cluster we perform nearest-neighbor search to obtain "child" clusters consisting of 4 samples. Samples in each child cluster are linked to each other with the "inter-instance" edges. We add new samples via visual tracking and link them to the original objects by "intra-instance" edges.

method presented in [343], *e.g.*, see $\langle A, A' \rangle$ and $\langle B, B' \rangle$ in Figure 4.1. But by tracking an identical instance we cannot associate different instances of the same semantics. Thus we expect the representations learned in [343] are weak in handling the variations between different objects in the same category.

## 4.2 Overview

Our goal is to learn visual representations which capture: (i) inter-instance invariance (*e.g.*, two instances of cats should have similar features), and (ii) intra-instance invariance (pose, viewpoint, deformation, illumination, and other variance of the same object instance). We have tried to formulate this as a multi-task (multi-loss) learning problem in our initial experiments (detailed in Table 4.2 and 4.3) and observed unsatisfactory performance. Instead of doing so, we propose to obtain a richer set of invariance by performing transitive reasoning on the data.

Our first step is to construct a graph that describes the affinity among image patches. A node in the graph denotes an image patch. We define two types of edges in the graph that relate image patches to each other. The first type of edges, called *inter-instance* edges, link two nodes which correspond to different object instances of similar visual appearance; the second type of edges, called *intra-instance* edges, link two nodes which correspond to an identical object captured at different time steps of a track. The solid arrows in Figure 4.1 illustrate these two types of edges.

Given the built graph, we want to *transit* the relations via the known edges and associate unconnected nodes that may provide under-explored invariance (Figure 4.1, dash arrows).

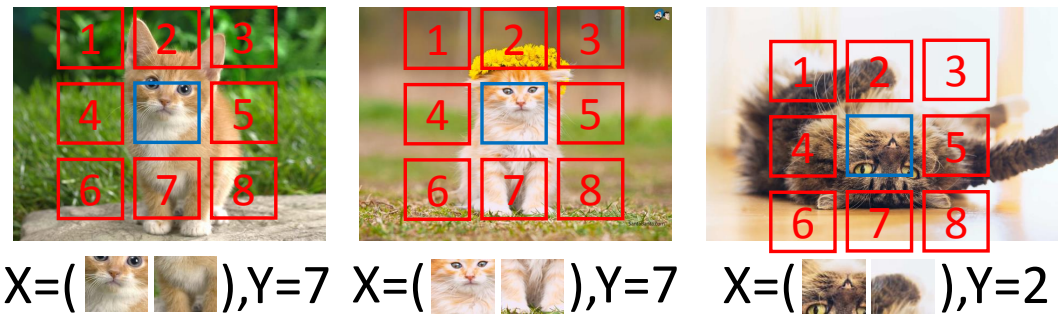



Figure 4.3: The *context prediction* task defined in [48]. Given two patches in an image, it learns to predict the relative position between them.

Specifically, as shown in Figure 4.1, if patches $\langle A, B\rangle$ are linked via an inter-instance edge and $\langle A, A'\rangle$ and $\langle B, B'\rangle$ respectively are linked via "intra-instance" edges, we hope to enrich the invariance by simple transitivity and relate three new pairs of: $\langle A', B'\rangle$, $\langle A, B'\rangle$, and $\langle A', B\rangle$ (Figure 4.1, dash arrows).

We train a Triplet-Siamese network that encourages similar visual representations between the invariant samples (*e.g.*, any pair consisting of $A, A', B, B'$) and at the same time discourages similar visual representations to a third distractor sample (*e.g.*, a random sample $C$ unconnected to $A, A', B, B'$). In all of our experiments, we apply VGG16 [281] as the backbone architecture for each branch of this Triplet-Siamese network. The visual representations learned by this backbone architecture are evaluated on other recognition tasks.

## 4.3 Graph Construction

We construct a graph with inter-instance and intra-instance edges. Firstly, we apply the method of [343] on a large set of 100K unlabeled videos (introduced in [343]) and mine millions of moving objects using motion cues (Sec. 4.3). We use the image patches of them to construct the nodes of the graph.

We instantiate inter-instance edges by the self-supervised method of [48] that learns context predictions on a large set of still images, which provide features to cluster the nodes and set up inter-instance edges (Sec. 4.3). On the other hand, we connect the image patches in the same visual track by intra-instance edges (Sec. 4.3).

### Mining Moving Objects

We follow the approach in [343] to find the moving objects in videos. As a brief introduction, this method first applies Improved Dense Trajectories (IDT) [324] on videos to extract SURF [12] feature points and their motion. The video frames are then pruned if there is too much motion (indicating camera motion) or too little motion (*e.g.*, noisy signals). For the remaining frames, it crop a 227×227 bounding box (from ∼600×400 images) which includes the most number of moving points as the foreground object. However, for computational efficiency, in this chapter we rescale the image patches to $96 \times 96$ after cropping and we use them as

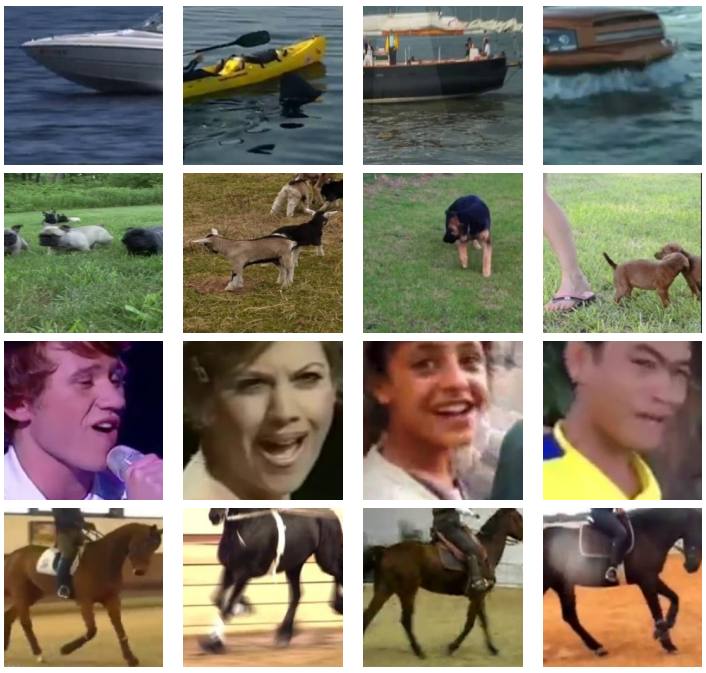


Figure 4.4: Some example clustering results. Each row shows the 4 examples in a child cluster (Sec. 4.3).

inputs for clustering and training.

## Inter-instance Edges via Clustering

Given the extracted image patches which act as nodes, we want to link them with extra inter-instance edges. We rely on the visual representations learned from [48] to do this. We connect the nodes representing image patches which are close in the feature space. In addition, motivated by the mid-level clustering approaches [45, 283], we want to obtain millions of object clusters with a small number of objects in each to maintain high "purity" of the clusters. We describe the implementation details of this step as follows.

We extract the `pool5` features of the VGG16 network trained as in [48]. Following [48], we use ImageNet *without* labels to train this network. Note that because we use a patch size of 96×96, the dimension of our `pool5` feature is 3×3×512=4608. The distance between samples is calculated by the cosine distance of these features. We want the object patches in each cluster to be close to each other in the feature space, and we care less about the differences between clusters. However, directly clustering millions of image patches into millions of small clusters (*e.g.*, by K-means) is time consuming. So we apply a hierarchical clustering approach (2-stage in this chapter) where we first group the images into a relatively small number of clusters, and then find groups of small number of examples inside each cluster via nearest-neighbor search.

Specifically, in the first stage of clustering, we apply K-means clustering with $K = 5000$

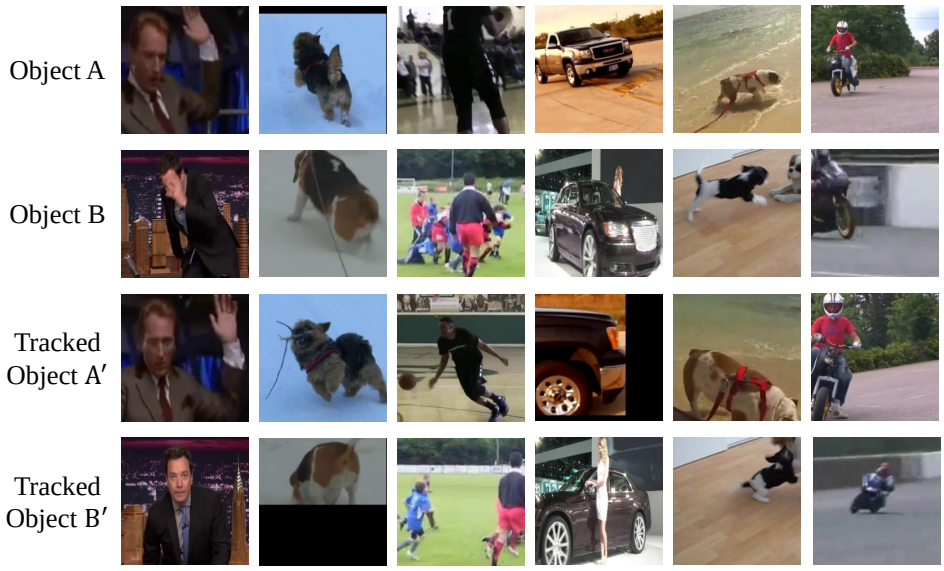



Figure 4.5: Examples used for training the network. Each column shows a set of image patches $\{A, B, A', B'\}$. Here, $A$ and $B$ is linked by an inter-instance edge, and $A'/B'$ is linked to $A/B$ via intra-instance edges.

on the image patches. We then remove the clusters with number of examples less than 100 (this reduces $K$ to 546 in our experiments on the image patches mined from the video dataset). We view these clusters as the "parent" clusters (blue circles in Figure 4.2). Then in the second stage of clustering, inside each parent cluster, we perform nearest-neighbor search for each sample and obtain its top 10 nearest neighbors in the feature space. We then find any group of samples with a group size of 4, inside which all the samples are each other's top-10 nearest neighbors. We call these small clusters with 4 samples "child" clusters (green circles in Figure 4.2). We then link these image patches with each other inside a child cluster via "inter-instance" edges. Note that different child clusters may overlap, *i.e.*, we allow the same sample to appear in different groups. However, in our experiments we find that most samples appear only in one group. We show some results of clustering in Figure 4.4.

### Intra-instance Edges via Tracking

To obtain rich variations of viewpoint and deformation changes of the same object instance, we apply visual tracking on the mined moving objects in the videos as in [343]. More specifically, given a moving object in the video, it applies KCF [119] to track the object for $N = 30$ frames and obtain another sample of the object in the end of the track. Note that the KCF tracker does not require any human supervision. We add these new objects as nodes to the graph and link the two samples in the same track with an intra-instance edge (purple in Figure 4.2).

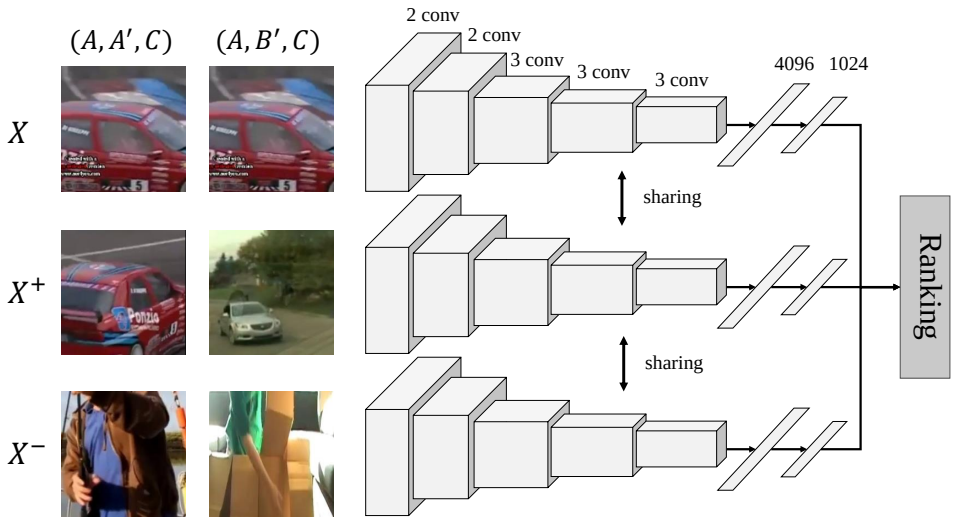



Figure 4.6: Our Triplet-Siamese network. We can feed in the network with different combinations of examples.

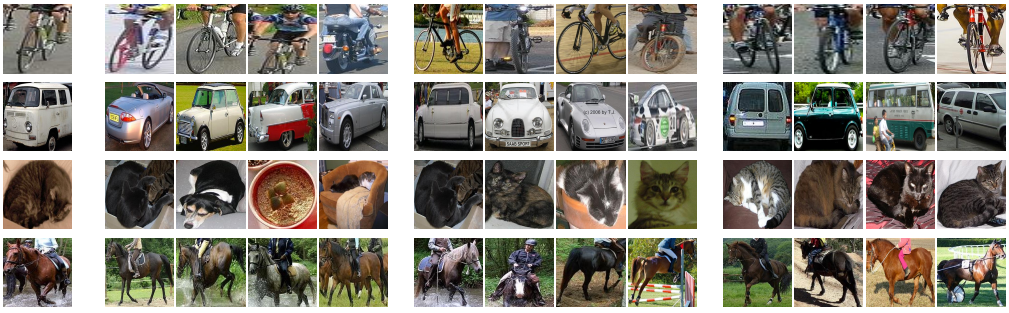
## 4.4 Learning with Transitions in the Graph

With the graph constructed, we want to link more image patches (see dotted links in Figure 4.1) which may be related via the transitivity of invariance. Objects subject to different levels of invariance can thus be related to each other. Specifically, if we have a set of nodes $\{A, B, A', B'\}$ where $\langle A, B\rangle$ are connected by an inter-instance edge and $\langle A, A'\rangle$ and $\langle B, B'\rangle$ are connected by an intra-instance edge, by assuming transitivity of invariance we expect the new pairs of $\langle A, B'\rangle$, $\langle A', B\rangle$, and $\langle A', B'\rangle$ to share similar high-level visual representations. Some examples are illustrated in Figure 4.1 and 4.5.

We train a deep neural network (VGG16) to generates similar visual representations if the image patches are linked by inter-instance/intra-instance edges or their transitivity (which we call a positive pair of samples). To avoid a trivial solution of identical representations, we also encourage the network to generate dissimilar representations if a node is expected to be unrelated. Specifically, we constrain the image patches from different "parent" clusters (which are more likely to have different categories) to have different representations (which we call a negative pair of samples). We design a Triplet-Siamese network with a ranking loss function [329, 343] such that the distance between related samples should be smaller than the distance of unrelated samples.

Our Triplet-Siamese network includes three towers of a ConvNet with shared weights (Figure 4.6). For each tower, we adopt the standard VGG16 architecture [281] to the convolutional layers, after which we add two fully-connected layers with 4096-d and 1024-d outputs. The Triplet-Siamese network accepts a triplet sample as its input: the first two image patches in the triplet are a positive pair, and the last two are a negative pair. We extract their 1024-d features and calculate the ranking loss as follows.

Given an arbitrary pair of image patches $A$ and $B$, we define their distance as: $D(A, B) = 1 - \frac{F(A)\cdot F(B)}{\|F(A)\|\|F(B)\|}$ where $F(\cdot)$ is the representation mapping of the network. With a triplet



Query (a) Context Prediction Network (b) Our Network (c) ImageNet Pre-trained Network

Figure 4.7: Nearest-neighbor search on the PASCAL VOC dataset. We extract three types of features: (a) context prediction network from [48], (b) network trained with our self-supervised method, and (c) the network pre-trained in the annotated ImageNet dataset. We show that our network can represent a greater variety (*e.g.*, viewpoints) of objects of the same category.

of $(X, X^+, X^-)$ where $(X, X^+)$ is a positive pair and $(X, X^-)$ is a negative pair as defined above, we minimize the ranking loss:

$$\mathcal{L}(X, X^+, X^-) = \max\{0, \mathcal{D}(X, X^+) - \mathcal{D}(X, X^-) + m\},$$

where $m$ is a margin set as $0.5$ in our experiments. Although we have only one objective function, we have different types of training examples. As illustrated in Figure 4.6, given the set of related samples $\{A, B, A', B'\}$ (see Figure 4.5) and a random distractor sample $C$ from another parent cluster, we can train the network to handle, *e.g.*, viewpoint invariance for the same instance via $\mathcal{L}(A, A', C)$ and invariance to different objects sharing the same semantics via $\mathcal{L}(A, B', C)$.

Besides exploring these relations, we have also tried to enforce the distance between different objects to be larger than the distance between two different viewpoints of the same object, *e.g.*, $\mathcal{D}(A, A') < \mathcal{D}(A, B')$. But we have not found this extra relation brings any improvement. Interestingly, we found that the representations learned by our method can in general satisfy $\mathcal{D}(A, A') < \mathcal{D}(A, B')$ after training.

## 4.5 Experiments

We perform extensive analysis on our self-supervised representations. We first evaluate our ConvNet as a feature extractor on different tasks *without* fine-tuning . We then show the results of transferring the representations to vision tasks including object detection and surface normal estimation with fine-tuning.

**Implementation Details.** To prepare the data for training, we download the 100K videos from YouTube using the URLs provided by [202, 343]. By mining the moving objects and tracking in the videos, we obtain ∼10 million image patches of objects. By applying the transitivity on the graph constructed, we obtain 7 million positive pairs of objects where each pair of objects are two different instances with different viewpoints. We also randomly sample 2 million object pairs connected by the intra-instance edges.

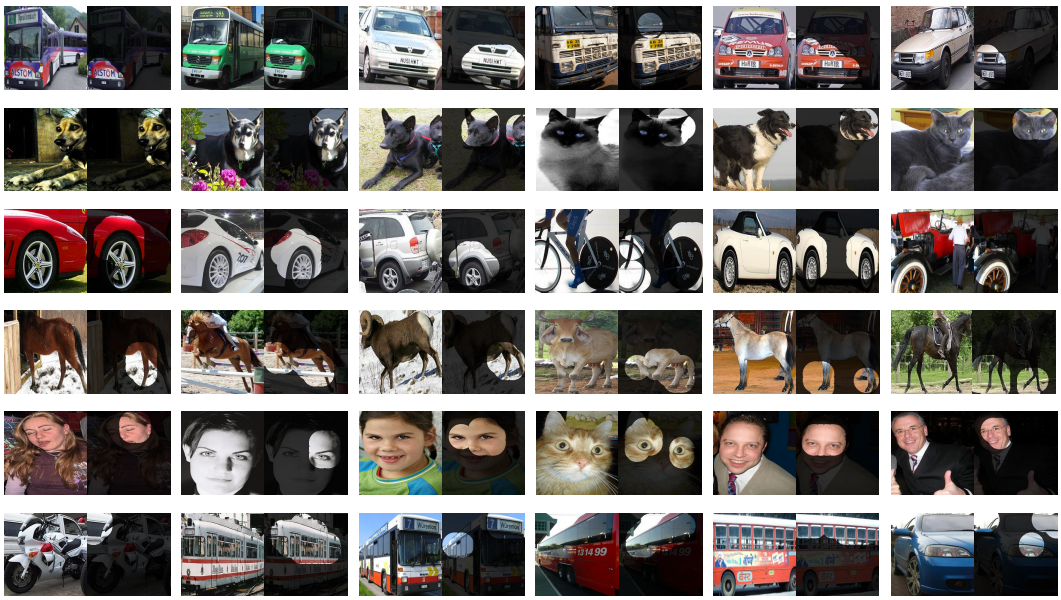


Figure 4.8: Top 6 responses for neurons in 6 different convolutional units of our network, visualized using [391].

We train our network with these 9 million pairs of images using a learning rate of 0.001 and a mini-batch size of 100. For each pair we sample the third distractor patch from a different "parent cluster" in the same mini-batch. We use the network pre-trained in [48] to initialize our convolutional layers and randomly initialized the fully connected layers. We train the network for 200K iterations with our method.

## Qualitative Results without Fine-tuning

We first perform nearest-neighbor search to show qualitative results. We adopt the pool5 feature of the VGG16 network for all methods without any fine-tuning (Figure 4.7). We do this experiment on the object instances cropped from the PASCAL VOC 2007 dataset [60] (`trainval`). As Figure 4.7 shows, given an query image on the left, the network pre-trained with the context prediction task [48] can retrieve objects of very similar viewpoints. On the other hand, our network shows more variations of objects and can often retrieve objects with the same class as the query. We also show the nearest-neighbor results using fully-supervised ImageNet pre-trained features as a comparison.

We also visualize the features using the visualization technique of [391]. For each convolutional unit in conv5_3, we retrieve the objects which give highest activation responses and highlight the receptive fields on the images. We visualize the top 6 images for 6 different convolutional units in Figure 4.8. We can see these convolutional units are corresponding to different semantic object parts (*e.g.*, fronts of cars or buses wheels, animal legs, eyes or faces).

| method | mAP | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | persn | plant | sheep | sofa | train | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| from scratch | 39.7 | 51.7 | 55.8 | 21.7 | 24.0 | 10.5 | 58.7 | 59.2 | 41.1 | 18.2 | 32.9 | 35.6 | 33.4 | 60.4 | 57.3 | 45.5 | 19.7 | 29.2 | 30.8 | 61.0 | 47.3 |
| Vid-Edge [199] | 44.2 | 54.4 | 58.2 | 39.6 | 30.8 | 12.5 | 58.7 | 61.9 | 51.0 | 22.0 | 41.4 | 47.4 | 41.5 | 63.2 | 58.4 | 47.5 | 17.2 | 27.6 | 45.4 | 59.8 | 45.4 |
| Context [48] | 61.5 | 70.8 | 72.1 | 54.7 | 49.7 | 31.0 | 72.3 | 76.9 | 70.8 | 44.6 | 61.1 | 59.8 | 67.0 | 74.6 | 72.5 | 68.3 | 29.4 | 58.5 | 66.9 | 75.1 | 54.3 |
| Tracking [343] | 60.2 | 65.7 | 73.2 | 55.4 | 46.4 | 30.9 | 74.0 | 76.9 | 67.8 | 40.9 | 58.0 | 60.9 | 65.0 | 74.1 | 71.6 | 67.1 | 31.5 | 55.0 | 61.8 | 73.9 | 53.8 |
| Ours | 63.2 | 68.4 | 74.6 | 57.1 | 49.6 | 34.1 | 73.5 | 76.9 | 73.2 | 45.8 | 63.3 | 66.3 | 68.6 | 74.9 | 74.2 | 69.5 | 31.9 | 57.4 | 70.3 | 75.9 | 59.3 |
| ImageNet | 67.3 | 74.4 | 78.0 | 65.9 | 54.4 | 39.7 | 76.4 | 78.6 | 82.5 | 48.6 | 73.3 | 67.2 | 78.4 | 77.3 | 75.7 | 72.2 | 32.2 | 65.8 | 66.8 | 75.2 | 62.4 |

Table 4.1: Object detection Average Precision (%) on the VOC 2007 test set using **Fast** R-CNN [81] (with selective search proposals [311]): comparisons among different self-supervised learning approaches.

| method | mAP | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | persn | plant | sheep | sofa | train | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ours | 63.2 | 68.4 | 74.6 | 57.1 | 49.6 | 34.1 | 73.5 | 76.9 | 73.2 | 45.8 | 63.3 | 66.3 | 68.6 | 74.9 | 74.2 | 69.5 | 31.9 | 57.4 | 70.3 | 75.9 | 59.3 |
| Multi-Task | 62.1 | 70.0 | 74.2 | 57.2 | 48.4 | 33.0 | 73.6 | 77.6 | 70.7 | 45.0 | 61.5 | 64.8 | 67.2 | 74.0 | 72.9 | 68.3 | 32.4 | 56.6 | 64.1 | 74.1 | 57.5 |
| Ours (15-frame) | 61.5 | 70.3 | 74.1 | 53.3 | 47.1 | 33.5 | 74.6 | 77.1 | 67.7 | 43.3 | 58.1 | 65.5 | 65.8 | 75.2 | 72.2 | 67.6 | 31.6 | 55.5 | 65.6 | 74.6 | 57.2 |
| Ours (HOG) | 60.4 | 65.8 | 73.4 | 54.7 | 47.7 | 30.2 | 75.6 | 77.1 | 67.6 | 42.0 | 58.8 | 63.2 | 65.3 | 74.1 | 72.0 | 67.2 | 29.9 | 54.4 | 62.1 | 72.9 | 53.9 |

Table 4.2: More ablative studies on object detection on the VOC 2007 test set using **Fast** R-CNN [81] (with selective search proposals [311]).

## Analysis on Object Detection

We evaluate how well our representations can be transferred to object detection by fine-tuning Fast R-CNN [81] on PASCAL VOC 2007 [60]. We use the standard `trainval` set for training and `test` set for testing with VGG16 as the base architecture. For the detection network, we initialize the weights of convolutional layers from our self-supervised network and randomly initialize the fully-connected layers using Gaussian noise with zero mean and $0.001$ standard deviation.

During fine-tuning Fast R-CNN, we use $0.00025$ as the starting learning rate. We reduce the learning rate by $1/10$ in every 50K iterations. We fine-tune the network for 150K iterations. Unlike standard Fast R-CNN where the first few convolutional layers of the ImageNet pre-trained network are fixed, we fine-tuned all layers on the PASCAL data as our model is pre-trained in a very different domain (*e.g.*, video patches).

We report the results in Table 4.1. If we train Fast R-CNN *from scratch* without any pre-training, we can only obtain $39.7\%$ mAP. With our self-supervised trained network as initialization, the detection mAP is increased to $63.2\%$ (with a $23.5$ points improvement). Our result compares competitively (4.1 points lower) to the counterpart using ImageNet pre-training ($67.3\%$ with VGG16).

As we incorporate the invariance captured from [343] and [48], we also evaluate the results using these two approaches individually (Table 4.1). By fine-tuning the context prediction network of [48], we can obtain $61.5\%$ mAP. To train the network of [343], we use exactly the same loss function and initialization as our approach except that there are only training examples of the same instance in the same visual track (*i.e.*, only the samples linked by intra-instance edges in our graph). Its results is $60.2\%$ mAP. Our result ($63.2\%$) is better than both methods. This comparison indicates the effectiveness of exploiting a greater variety of invariance in representation learning.

**Is multi-task learning sufficient?** An alternative way of obtaining both intra- and inter-instance invariance is to apply multi-task learning with the two losses of [48] and [343]. Next

| | All | >c1 | >c2 | >c3 | >c4 | >c5 |
|---|---|---|---|---|---|---|
| Context [48] | 62.6 | 61.1 | 60.9 | 57.0 | 49.7 | 38.1 |
| Tracking [343] | 62.2 | 61.5 | 62.2 | 61.4 | 58.9 | 39.5 |
| Multi-Task [48,343] | 62.4 | 63.2 | 63.5 | 62.9 | 58.7 | 27.6 |
| Ours | 65.0 | 64.5 | 63.6 | 60.4 | 55.7 | 43.1 |
| ImageNet | 70.9 | 71.1 | 71.1 | 70.2 | 70.3 | 64.3 |

Table 4.3: Object detection Average Precision (%) on the VOC 2007 test set using joint training **Faster** R-CNN [249].

we compare with this method.

For the task in [343], we use the same network architecture as our approach; for the task in [48], we follow their design of a Siamese network. We apply different fully connected layers for different tasks, but share the convolutional layers between these two tasks. Given a mini-batch of training samples, we perform ranking among these images as well as context prediction in each image simultaneously via two losses. The representations learned in this way, when fine-tuned with Fast R-CNN, obtain $62.1\%$ mAP ("Multi-task" in Table 4.2). Comparing to only using context prediction [48] ($61.5\%$), the multi-task learning only gives a marginal improvement ($0.6\%$). This result suggests that multi-task learning in this way is not sufficient; organizing and exploiting the relationships of data, as done by our method, is more effective for representation learning.

**How important is tracking?** To further understand how much visual tracking helps, we perform ablative analysis by making the visual tracks shorter: we track the moving objects for $15$ frames instead of by default $30$ frames. This is expected to reduce the viewpoint/pose/deformation variance contributed by tracking. Our model pre-trained in this way shows $61.5\%$ mAP ("15-frame" in Table 4.2) when fine-tuned for detection. This number is similar to that of using context prediction only (Table 4.1). This result is not surprising, because it does not add much new information for training. It suggests adding stronger viewpoint/pose/deformation invariance is important for learning better features for object detection.

**How important is clustering?** Furthermore, we want to understand how important it is to cluster images with features learned from still images [48]. We perform another ablative analysis by replacing the features of [48] with HOG [40] during clustering. The rest of the pipeline remains exactly the same. The final result is $60.4\%$ mAP ("HOG" in Table 4.2). This shows that if the features for clustering are not invariant enough to handle different object instances, the transitivity in the graph becomes less reliable.

## Object Detection with Faster R-CNN

Although Fast R-CNN [81] has been a popular testbed of un-/self-supervised features, it relies on Selective Search proposals [311] and thus is not fully end-to-end. We further evaluate the representations on object detection with the *end-to-end* Faster R-CNN [249] where the Region Proposal Network (RPN) may suffer from the features if they are low-quality.

**PASCAL VOC 2007 Results.** We fine-tune Faster R-CNN in 8 GPUs for 35K iterations

| | AP | $AP^{50}$ | $AP^{75}$ | $AP^{S}$ | $AP^{M}$ | $AP^{L}$ |
|---|---|---|---|---|---|---|
| from scratch | 20.5 | 40.1 | 19.0 | 5.6 | 22.5 | 32.7 |
| Context [48] | 22.7 | 43.5 | 21.2 | 6.6 | 24.9 | 36.5 |
| Tracking [343] | 22.6 | 42.8 | 21.6 | 6.3 | 25.0 | 36.2 |
| Multi-Task [48,343] | 22.0 | 42.3 | 21.1 | 6.6 | 24.5 | 35.0 |
| Ours | 23.5 | 44.4 | 22.6 | 7.1 | 25.9 | 37.3 |
| ImageNet (shorter) | 23.7 | 44.5 | 23.5 | 7.2 | 26.9 | 37.4 |
| ImageNet | 24.4 | 46.4 | 23.1 | 7.9 | 27.4 | 38.1 |

Table 4.4: Object detection Average Precision (%, COCO definitions) on COCO minival using joint training **Faster** R-CNN [249]. "(shorter)" indicates a shorter training time (fewer iterations, 61.25K) used by the codebase of [249].

with an initial learning rate of $0.00025$ which is reduced by $1/10$ after every 15K iterations. Table 4.3 shows the results of fine-tuning all layers ("All") and also ablative results on freezing different levels of convolutional layers (*e.g.*, the column >c3 represents freezing all the layers below and including conv3_x in VGG16 during fine-tuning). Our method gets even better results of $65.0\%$ by using Faster R-CNN, showing a larger gap compared to the counterparts of [48] ($62.6\%$) and [343] ($62.2\%$). Noteworthily, when freezing all the convolutional layers and only fine-tuning the fully-connected layers, our method ($43.1\%$) is much better than other competitors. And we again find that the multi-task alternative does not work well for Faster R-CNN.

**COCO Results.** We further report results on the challenging COCO detection dataset [205]. To the best of our knowledge this is the first work of this kind presented on COCO detection. We fine-tune Faster R-CNN in 8 GPUs for 120K iterations with an initial learning rate of $0.001$ which is reduced by $1/10$ after 80k iterations. This is trained on the COCO `trainval35k` split and evaluated on the `minival5k` split, introduced by [13].

We report the COCO results on Table 4.4. Faster R-CNN fine-tuned with our self-supervised network obtains $23.5\%$ AP using the COCO metric, which is very close ($<1\%$) to fine-tuning Faster R-CNN with the ImageNet pre-trained counterpart ($24.4\%$). Actually, if the fine-tuning of the ImageNet counterpart follows the "shorter" schedule in the public code (61.25K iterations in 8 GPUs, converted from 490K in 1 GPU)[1], the ImageNet supervised pre-training version has $23.7\%$ AP and is comparable with ours. This comparison also strengthens the significance of our result.

To the best of our knowledge, our model achieves the best performance reported to date on VOC 2007 and COCO using un-/self-supervised pre-training.

## Adapting to Surface Normal Estimation

To show the generalization ability of our self-supervised representations, we adopt the learned network to the surface normal estimation task. In this task, given a single RGB image as input, we train the network to predict the normal/orientation of the pixels. We evaluate our method on the NYUv2 RGBD dataset [275] dataset. We use the official split of 795 images for training and 654 images for testing. We follow the same protocols for

[1] https://github.com/rbgirshick/py-faster-rcnn

| | Mean | Median | 11.25° | 22.5° | 30° |
|---|---|---|---|---|---|
| | (lower is better) | | (higher is better) | | |
| from scratch | 31.3 | 25.3 | 24.2 | 45.6 | 56.8 |
| Context [48] | 29.0 | 21.6 | 28.8 | 51.5 | 61.9 |
| Tracking [343] | 27.8 | 21.8 | 27.4 | 51.1 | 62.5 |
| Ours | **26.0** | **18.0** | **33.9** | **57.6** | **67.5** |
| ImageNet | 27.8 | 21.2 | 29.0 | 52.3 | 63.4 |

Table 4.5: Results on NYU v2 for per-pixel surface normal estimation, evaluated over valid pixels.

generating surface normal ground truth and evaluations as [68, 69, 176].

To train the network for surface normal estimation, we apply the Fully Convolutional Network (FCN 32-s) proposed in [210] with the VGG16 network as base architecture. For the loss function, we follow the design in [336]. Specifically, instead of direct regression to obtain the normal, we use a codebook of 40 codewords to encode the 3-dimension normals. Each codeword represents one class thus we turn the problem into a 40-class classification for each pixel. We use the same hyperparameters as in [210] for training and the network is fine-tuned for same number of iterations (100K) for different initializations.

To initialize the FCN model with self-supervised nets, we copy the weights of the convolutional layers to the corresponding layers in FCN. For ImageNet pre-trained network, we follow [210] by converting the fully connected layers to convolutional layers and copy all the weights. For the model trained from scratch, we randomly initialize all the layers with "Xavier" initialization [91] .

Table 4.5 shows the results. We report mean and median error for all visible pixels (in degrees) and also the percentage of pixels with error less than 11.25, 22.5 and 30 degrees. Surprisingly, we obtain much better results with our self-supervised trained network than ImageNet pre-training in this task (3 to 4% better in most metrics). As a comparison, the network trained in [48, 343] are slightly worse than the ImageNet pre-trained network. These results suggest that our learned representations are competitive to ImageNet pre-training for high-level semantic tasks, but outperforms it on tasks such as surface normal estimation. This experiment suggests that different visual tasks may prefer different levels of visual invariance.

## 4.6 Discussion

In this chapter, we propose a self-supervised method for learning visual representations handling multiple invariations in data. Instead of doing multi-task learning, we propose to use multiple sources to organize and relate data points for training. Our learned representations achieve the best performance reported to date on VOC 2007 and COCO using un-/self-supervised pre-training. On the challenging COCO detection dataset, we present a surprisingly small gap of self-supervised pre-training to the powerful, prevalent ImageNet pre-training counterpart.

# Part II

# Video Understanding with Correspondence

# Chapter 5

# Non-local Neural Networks

In the previous chapters, we have introduced how can we use correspondence to provide a supervisory signal to train deep networks. Besides supervision, in this chapter, we will illustrate that correspondence is also the key in designing modern neural network architectures.

Capturing *long-range* dependencies is of central importance in deep neural networks. For sequential data (*e.g.*, in speech, language), *recurrent* operations [124, 258] are the dominant solution to long-range dependency modeling. For image data, long-distance dependencies are modeled by the large receptive fields formed by deep stacks of *convolutional* operations [73, 188].

Convolutional and recurrent operations both process a *local* neighborhood, either in space or time; thus long-range dependencies can only be captured when these operations are applied repeatedly, propagating signals progressively through the data. Repeating local operations has several limitations. First, it is computationally inefficient. Second, it causes optimization difficulties that need to be carefully addressed [114, 124]. Finally, these challenges make multi-hop dependency modeling, *e.g.*, when messages need to be delivered back and forth between distant positions, difficult.

In this chapter, we present *non-local* operations as an efficient, simple, and generic component for capturing long-range dependencies with deep neural networks. Our proposed non-local operation is a generalization of the classical non-local mean operation [21] in computer vision. Intuitively, a non-local operation computes the response at a position as a weighted sum of the features at *all positions* in the input feature maps (Figure 5.1). The set of positions can be in space, time, or spacetime, implying that our operations are applicable for image, sequence, and video problems.

There are several advantages of using non-local operations: (a) In contrast to the progressive behavior of recurrent and convolutional operations, non-local operations capture long-range dependencies directly by computing interactions between any two positions, regardless of their positional distance; (b) As we show in experiments, non-local operations are efficient and achieve their best results even with only a few layers (*e.g.*, 5); (c) Finally, our non-local operations maintain the variable input sizes and can be easily combined with other operations (*e.g.*, convolutions as we will use).

We showcase the effectiveness of non-local operations in the application of video classification. In videos, long-range interactions occur between distant pixels in space as well as

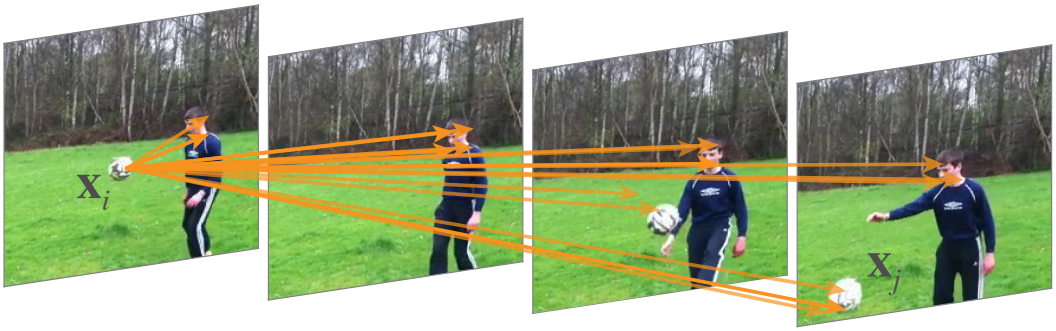



Figure 5.1: A spacetime ***non-local*** operation in our network trained for video classification. A position $\mathbf{x}_i$'s response is computed by the weighted average of the features of *all* positions $\mathbf{x}_j$ (only the highest weighted ones are shown here). In this example computed by our model, note how it relates the ball in the first frame to the ball in the last two frames. More examples are in Figure 5.3.

time. A single non-local block, which is our basic unit, can directly capture these spacetime dependencies in a feedforward fashion. With a few non-local blocks, our architecures called *non-local neural networks* are more accurate for video classification than 2D and 3D convolutional networks [306] (including the inflated variant [27]). In addition, non-local neural networks are more computationally economical than their 3D convolutional counterparts. Comprehensive ablation studies are presented on the Kinetics [154] and Charades [273] datasets. *Using RGB only and without any bells and whistles* (*e.g.*, optical flow, multi-scale testing), our method achieves results on par with or better than the latest competitions winners on both datasets.

To demonstrate the generality of non-local operations, we further present object detection/segmentation and pose estimation experiments on the COCO dataset [205]. On top of the strong Mask R-CNN baseline [112], our non-local blocks can increase accuracy on all three tasks at a small extra computational cost. Together with the evidence on videos, these image experiments show that non-local operations are generally useful and can become a basic building block in designing deep neural networks.

## 5.1 Background

**Non-local image processing.** Non-local means [21] is a classical filtering algorithm that computes a weighted mean of all pixels in an image. It allows distant pixels to contribute to the filtered response at a location based on patch appearance similarity. This non-local filtering idea was later developed into BM3D (block-matching 3D) [37], which performs filtering on a group of similar, but non-local, patches. BM3D is a solid image denoising baseline even compared with deep neural networks [22]. Block matching was used with neural networks for image denoising [23, 191]. Non-local matching is also the essence of successful texture synthesis [55], super-resolution [90], and inpainting [9] algorithms.

**Graphical models.** Long-range dependencies can be modeled by graphical models such as conditional random fields (CRF) [168, 177]. In the context of deep neural networks, a CRF can be exploited to post-process semantic segmentation predictions of a network [31]. The

iterative mean-field inference of CRF can be turned into a recurrent network and trained [28, 111, 208, 268, 388]. In contrast, our method is a simpler feedforward block for computing non-local filtering. Unlike these methods that were developed for segmentation, our general-purpose component is applied for classification and detection. These methods and ours are also related to a more abstract model called graph neural networks [267].

**Feedforward modeling for sequences.** Recently there emerged a trend of using feedforward (*i.e.*, non-recurrent) networks for modeling sequences in speech and language [78, 232, 367]. In these methods, long-term dependencies are captured by the large receptive fields contributed by very deep 1-D convolutions. These feedforward models are amenable to parallelized implementations and can be more efficient than widely used recurrent models.

**Self-attention.** Our work is related to the recent *self-attention* [314] method for machine translation. A self-attention module computes the response at a position in a sequence (*e.g.*, a sentence) by attending to all positions and taking their weighted average in an embedding space. As we will discuss in the next, self-attention can be viewed as a form of the non-local mean [21], and in this sense our work bridges self-attention for machine translation to the more general class of non-local filtering operations that are applicable to image and video problems in computer vision.

**Interaction networks.** *Interaction Networks* (IN) [10, 350] were proposed recently for modeling physical systems. They operate on graphs of objects involved in pairwise interactions. Hoshen [129] presented the more efficient Vertex Attention IN (VAIN) in the context of multi-agent predictive modeling. Another variant, named Relation Networks [266], computes a function on the feature embeddings at all pairs of positions in its input. Our method also processes all pairs, as we will explain ($f(\mathbf{x}_i, \mathbf{x}_j)$ in Eq.(5.1)). While our non-local networks are connected to these approaches, our experiments indicate that the *non-locality* of the model, which is orthogonal to the ideas of attention/interaction/relation (*e.g.*, a network can attend to a local region), is the key to their empirical success. Non-local modeling, a long-time crucial element of image processing (*e.g.*, [21, 55]), has been largely overlooked in recent neural networks for computer vision.

**Video classification architectures.** A natural solution to video classification is to combine the success of CNNs for images and RNNs for sequences [52, 381]. In contrast, feedforward models are achieved by 3D convolutions (C3D) [144, 306] in spacetime, and the 3D filters can be formed by "inflating" [27, 62] pre-trained 2D filters. In addition to end-to-end modeling on raw video inputs, it has been found that optical flow [279] and trajectories [326, 330] can be helpful. Both flow and trajectories are off-the-shelf modules that may find long-range, non-local dependency. A systematic comparison of video architectures can be found in [27].

## 5.2 Non-local Neural Networks

We first give a general definition of non-local operations and then we provide several specific instantiations of it.

## Formulation

Following the non-local mean operation [21], we define a generic non-local operation in deep neural networks as:

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j). \tag{5.1}$$

Here $i$ is the index of an output position (in space, time, or spacetime) whose response is to be computed and $j$ is the index that enumerates all possible positions. $\mathbf{x}$ is the input signal (image, sequence, video; often their features) and $\mathbf{y}$ is the output signal of the same size as $\mathbf{x}$. A pairwise function $f$ computes a scalar (representing relationship such as affinity) between $i$ and all $j$. The unary function $g$ computes a representation of the input signal at the position $j$. The response is normalized by a factor $\mathcal{C}(\mathbf{x})$.

The non-local behavior in Eq.(5.1) is due to the fact that all positions ($\forall j$) are considered in the operation. As a comparison, a convolutional operation sums up the weighted input in a *local* neighborhood (*e.g.*, $i-1 \leq j \leq i+1$ in a 1D case with kernel size 3), and a recurrent operation at time $i$ is often based only on the current and the latest time steps (*e.g.*, $j = i$ or $i-1$).

The non-local operation is also different from a fully-connected (*fc*) layer. Eq.(5.1) computes responses based on relationships between different locations, whereas *fc* uses learned weights. In other words, the relationship between $\mathbf{x}_j$ and $\mathbf{x}_i$ is not a function of the input data in *fc*, unlike in non-local layers. Furthermore, our formulation in Eq.(5.1) supports inputs of *variable* sizes, and maintains the corresponding size in the output. On the contrary, an *fc* layer requires a fixed-size input/output and loses positional correspondence (*e.g.*, that from $\mathbf{x}_i$ to $\mathbf{y}_i$ at the position $i$).

A non-local operation is a flexible building block and can be easily used together with convolutional/recurrent layers. It can be added into the earlier part of deep neural networks, unlike *fc* layers that are often used in the end. This allows us to build a richer hierarchy that combines both non-local and local information.

## Instantiations

Next we describe several versions of $f$ and $g$. Interestingly, we will show by experiments (Table 5.2a) that our non-local models are not sensitive to these choices, indicating that the generic non-local behavior is the main reason for the observed improvements.

For simplicity, we only consider $g$ in the form of a linear embedding: $g(\mathbf{x}_j) = W_g \mathbf{x}_j$, where $W_g$ is a weight matrix to be learned. This is implemented as, *e.g.*, 1×1 convolution in space or 1×1×1 convolution in spacetime.

Next we discuss choices for the pairwise function $f$.

**Gaussian.** Following the non-local mean [21] and bilateral filters [305], a natural choice of $f$ is the Gaussian function. In this chapter we consider:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\mathbf{x}_i^T \mathbf{x}_j}. \tag{5.2}$$

Here $\mathbf{x}_i^T \mathbf{x}_j$ is dot-product similarity. Euclidean distance as used in [21,305] is also applicable, but dot product is more implementation-friendly in modern deep learning platforms. The normalization factor is set as $\mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$.

**Embedded Gaussian.** A simple extension of the Gaussian function is to compute similarity in an embedding space. In this chapter we consider:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}. \tag{5.3}$$

Here $\theta(\mathbf{x}_i) = W_\theta \mathbf{x}_i$ and $\phi(\mathbf{x}_j) = W_\phi \mathbf{x}_j$ are two embeddings. As above, we set $\mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$.

We note that *the self-attention module [314] recently presented for machine translation is a special case of non-local operations in the embedded Gaussian version.* This can be seen from the fact that for a given $i$, $\frac{1}{\mathcal{C}(\mathbf{x})} f(\mathbf{x}_i, \mathbf{x}_j)$ becomes the *softmax* computation along the dimension $j$. So we have $\mathbf{y} = \textit{softmax}(\mathbf{x}^T W_\theta^T W_\phi \mathbf{x}) g(\mathbf{x})$, which is the self-attention form in [314]. As such, our work provides insight by relating this recent self-attention model to the classic computer vision method of non-local means [21], and extends the sequential self-attention network in [314] to a generic space/spacetime non-local network for image/video recognition in computer vision.

Despite the relation to [314], we show that the attentional behavior (due to softmax) is *not* essential in the applications we study. To show this, we describe two alternative versions of non-local operations next.

**Dot product.** $f$ can be defined as a dot-product similarity:

$$f(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j). \tag{5.4}$$

Here we adopt the embedded version. In this case, we set the normalization factor as $\mathcal{C}(\mathbf{x}) = N$, where $N$ is the number of positions in $\mathbf{x}$, rather than the sum of $f$, because it simplifies gradient computation. A normalization like this is necessary because the input can have variable size.

The main difference between the dot product and embedded Gaussian versions is the presence of softmax, which plays the role of an activation function.

**Concatenation.** Concatenation is used by the pairwise function in Relation Networks [266] for visual reasoning. We also evaluate a concatenation form of $f$:

$$f(\mathbf{x}_i, \mathbf{x}_j) = \text{ReLU}(\mathbf{w}_f^T [\theta(\mathbf{x}_i),\ \phi(\mathbf{x}_j)]). \tag{5.5}$$

Here $[\cdot, \cdot]$ denotes concatenation and $\mathbf{w}_f$ is a weight vector that projects the concatenated vector to a scalar. As above, we set $\mathcal{C}(\mathbf{x}) = N$. In this case, we adopt ReLU [228] in $f$.

The above several variants demonstrate the flexibility of our generic non-local operation. We believe alternative versions are possible and may improve results.

## Non-local Block

We wrap the non-local operation in Eq.(5.1) into a non-local block that can be incorporated into many existing architectures. We define a non-local block as:

$$\mathbf{z}_i = W_z \mathbf{y}_i + \mathbf{x}_i, \tag{5.6}$$

where $\mathbf{y}_i$ is given in Eq.(5.1) and "$+\mathbf{x}_i$" denotes a residual connection [114]. The residual connection allows us to insert a new non-local block into any pre-trained model, without

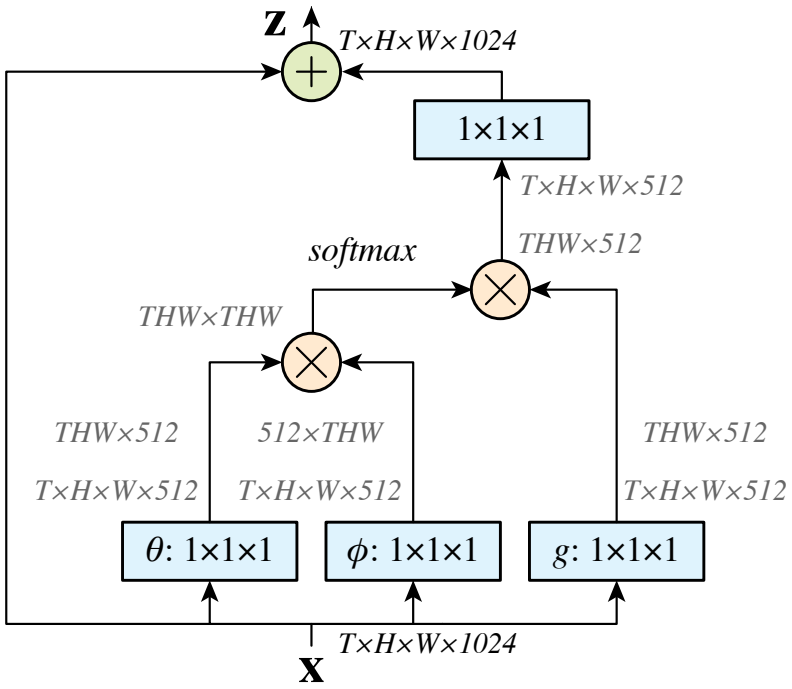



Figure 5.2: A spacetime **non-local block**. The feature maps are shown as the shape of their tensors, *e.g.*, $T\times H\times W\times 1024$ for 1024 channels (proper reshaping is performed when noted). "$\otimes$" denotes matrix multiplication, and "$\oplus$" denotes element-wise sum. The softmax operation is performed on each row. The blue boxes denote $1\times1\times1$ convolutions. Here we show the embedded Gaussian version, with a bottleneck of 512 channels. The vanilla Gaussian version can be done by removing $\theta$ and $\phi$, and the dot-product version can be done by replacing softmax with scaling by $1/N$.

breaking its initial behavior (*e.g.*, if $W_z$ is initialized as zero). An example non-local block is illustrated in Figure 5.2. The pairwise computation in Eq.(5.2), (5.3), or (5.4) can be simply done by matrix multiplication as shown in Figure 5.2; the concatenation version in (5.5) is straightforward.

The pairwise computation of a non-local block is lightweight when it is used in high-level, sub-sampled feature maps. For example, typical values in Figure 5.2 are $T = 4$, $H = W = 14$ or 7. The pairwise computation as done by matrix multiplication is comparable to a typical convolutional layer in standard networks. We further adopt the following implementations that make it more efficient.

**Implementation of Non-local Blocks.** We set the number of channels represented by $W_g$, $W_\theta$, and $W_\phi$ to be half of the number of channels in $\mathbf{x}$. This follows the bottleneck design of [114] and reduces the computation of a block by about a half. The weight matrix $W_z$ in Eq.(5.6) computes a position-wise embedding on $\mathbf{y}_i$, matching the number of channels to that of $\mathbf{x}$. See Figure 5.2.

A subsampling trick can be used to further reduce computation. We modify Eq.(5.1) as: $\mathbf{y}_i = \frac{1}{\mathcal{C}(\hat{\mathbf{x}})}\sum_{\forall j} f(\mathbf{x}_i, \hat{\mathbf{x}}_j)g(\hat{\mathbf{x}}_j)$, where $\hat{\mathbf{x}}$ is a subsampled version of $\mathbf{x}$ (*e.g.*, by pooling). We

| layer | | output size |
|---|---|---|
| $\text{conv}_1$ | 7×7, 64, stride 2, 2, 2 | 16×112×112 |
| $\text{pool}_1$ | 3×3×3 max, stride 2, 2, 2 | 8×56×56 |
| $\text{res}_2$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$ | 8×56×56 |
| $\text{pool}_2$ | 3×1×1 max, stride 2, 1, 1 | 4×56×56 |
| $\text{res}_3$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$ | 4×28×28 |
| $\text{res}_4$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$ | 4×14×14 |
| $\text{res}_5$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$ | 4×7×7 |
| global average pool, fc | | 1×1×1 |

Table 5.1: Our *baseline* ResNet-50 C2D model for video. The dimensions of 3D output maps and filter kernels are in T×H×W (2D kernels in H×W), with the number of channels following. The input is 32×224×224. Residual blocks are shown in brackets.

perform this in the spatial domain, which can reduce the amount of pairwise computation by 1/4. This trick does not alter the non-local behavior, but only makes the computation sparser. This can be done by adding a max pooling layer after $\phi$ and $g$ in Figure 5.2.

We use these efficient modifications for all non-local blocks studied In this chapter.

## 5.3 Video Classification Models

To understand the behavior of non-local networks, we conduct comprehensive ablation experiments on video classification tasks. First we describe our baseline network architectures for this task, and then extend them into 3D ConvNets [27, 306] and our proposed non-local nets.

**2D ConvNet baseline (C2D).** To isolate the temporal effects of our non-local nets *vs*. 3D ConvNets, we construct a simple 2D baseline architecture in which the temporal dimension is trivially addressed (*i.e.*, only by pooling).

Table 5.1 shows our C2D baseline under a ResNet-50 backbone. The input video clip has 32 frames each with 224×224 pixels. All convolutions in Table 5.1 are in essence 2D kernels that process the input frame-by-frame (implemented as $1\times k\times k$ kernels). This model can be directly initialized from the ResNet weights pre-trained on ImageNet. A ResNet-101 counterpart is built in the same way.

Figure 5.3: Examples of the behavior of a non-local block in $\text{res}_3$ computed by a 5-block non-local model trained on Kinetics. These examples are from held-out validation videos. The starting point of arrows represents one $\mathbf{x}_i$, and the ending points represent $\mathbf{x}_j$. The 20 highest weighted arrows for each $\mathbf{x}_i$ are visualized. The 4 frames are from a 32-frame input, shown with a stride of 8 frames. These visualizations show how the model finds related clues to support its prediction.

The only operation involving the temporal domain are the pooling layers. In other words, this baseline simply aggregates temporal information.

**Inflated 3D ConvNet (I3D).** As done in [27, 62], one can turn the C2D model in Table 5.1 into a 3D convolutional counterpart by "inflating" the kernels. For example, a 2D $k\times k$ kernel can be inflated as a 3D $t\times k\times k$ kernel that spans $t$ frames. This kernel can be initialized from 2D models (pre-trained on ImageNet): each of the $t$ planes in the $t\times k\times k$ kernel is initialized by the pre-trained $k\times k$ weights, rescaled by $1/t$. If a video consists of a single static frame repeated in time, this initialization produces the same results as the 2D pre-trained model run on a static frame.

We study two cases of inflations: we either inflate the 3×3 kernel in a residual block to 3×3×3 (similar to [27]), or the first 1×1 kernel in a residual block to 3×1×1 (similar to [62]). We denote these as $\text{I3D}_{3\times3\times3}$ and $\text{I3D}_{3\times1\times1}$. As 3D convolutions are computationally intensive, we only inflate one kernel for every 2 residual blocks; inflating more layers shows diminishing return. We inflate $\text{conv}_1$ to 5×7×7.

The authors of [27] have shown that I3D models are more accurate than their CNN+LSTM counterparts.

**Non-local network.** We insert non-local blocks into C2D or I3D to turn them into non-local nets. We investigate adding 1, 5, or 10 non-local blocks; the implementation details are

described in the next section in context.

### Implementation Details

**Training.** Our models are pre-trained on ImageNet [260]. Unless specified, we fine-tune our models using 32-frame input clips. These clips are formed by randomly cropping out 64 consecutive frames from the original full-length video and then dropping every other frame. The spatial size is 224×224 pixels, randomly cropped from a scaled video whose shorter side is randomly sampled in $[256, 320]$ pixels, following [282]. We train on an 8-GPU machine and each GPU has 8 clips in a mini-batch (so in total with a mini-batch size of 64 clips). We train our models for 400k iterations in total, starting with a learning rate of 0.01 and reducing it by a factor of 10 at every 150k iterations (see also Figure 5.4). We use a momentum of 0.9 and a weight decay of 0.0001. We adopt dropout [123] after the global pooling layer, with a dropout ratio of 0.5. We fine-tune our models with BatchNorm (BN) [134] enabled when it is applied. This is in contrast to common practice [114] of fine-tuning ResNets, where BN was frozen. We have found that enabling BN in our application reduces overfitting.

We adopt the method in [113] to initialize the weight layers introduced in the non-local blocks. We add a BN layer right after the last $1\times1\times1$ layer that represents $W_z$; we do not add BN to other layers in a non-local block. The scale parameter of this BN layer is initialized as zero, following [97]. This ensures that the initial state of the entire non-local block is an identity mapping, so it can be inserted into any pre-trained networks while maintaining its initial behavior.

**Inference.** Following [282] we perform spatially fully-convolutional inference on videos whose shorter side is rescaled to 256. For the temporal domain, in our practice we sample 10 clips evenly from a full-length video and compute the softmax scores on them individually. The final prediction is the averaged softmax scores of all clips.

## 5.4 Experiments on Video Classification

We perform comprehensive studies on the challenging Kinetics dataset [154]. We also report results on the Charades dataset [273] to show the generality of our models.

### Experiments on Kinetics

Kinetics [154] contains ~246k training videos and 20k validation videos. It is a classification task involving 400 human action categories. We train all models on the training set and test on the validation set.

Figure 5.4 shows the curves of the training procedure of a ResNet-50 C2D baseline *vs*. a non-local C2D with 5 blocks (more details in the following). Our non-local C2D model is consistently better than the C2D baseline *throughout the training procedure*, in both training and validation error.

Figure 5.1 and Figure 5.3 visualize several examples of the behavior of a non-local block computed by our models. Our network can learn to find meaningful relational clues regardless of the distance in space and time.

| model, R50 | top-1 | top-5 |
|---|---|---|
| C2D baseline | 71.8 | 89.7 |
| Gaussian | 72.5 | 90.2 |
| Gaussian, embed | 72.7 | **90.5** |
| dot-product | **72.9** | 90.3 |
| concatenation | 72.8 | **90.5** |

(a) **Instantiations**: 1 non-local block of different types is added into the C2D baseline. All entries are with ResNet-50.

| model, R50 | top-1 | top-5 |
|---|---|---|
| baseline | 71.8 | 89.7 |
| $res_2$ | 72.7 | 90.3 |
| $res_3$ | **72.9** | 90.4 |
| $res_4$ | 72.7 | **90.5** |
| $res_5$ | 72.3 | 90.1 |

(b) **Stages**: 1 non-local block is added into different stages. All entries are with ResNet-50.

| model | | top-1 | top-5 |
|---|---|---|---|
| R50 | baseline | 71.8 | 89.7 |
| | 1-block | 72.7 | 90.5 |
| | 5-block | 73.8 | 91.0 |
| | 10-block | **74.3** | **91.2** |
| R101 | baseline | 73.1 | 91.0 |
| | 1-block | 74.3 | 91.3 |
| | 5-block | **75.1** | **91.7** |
| | 10-block | **75.1** | 91.6 |

(c) **Deeper non-local models**: we compare 1, 5, and 10 non-local blocks added to the C2D baseline. We show ResNet-50 (top) and ResNet-101 (bottom) results.

| model | | top-1 | top-5 |
|---|---|---|---|
| R50 | baseline | 71.8 | 89.7 |
| | space-only | 72.9 | 90.8 |
| | time-only | 73.1 | 90.5 |
| | spacetime | **73.8** | **91.0** |
| R101 | baseline | 73.1 | 91.0 |
| | space-only | 74.4 | 91.3 |
| | time-only | 74.4 | 90.5 |
| | spacetime | **75.1** | **91.7** |

(d) **Space *vs.* time *vs.* spacetime**: we compare non-local operations applied along space, time, and spacetime dimensions respectively. 5 non-local blocks are used.

| model, R101 | params | FLOPs | top-1 | top-5 |
|---|---|---|---|---|
| C2D baseline | 1× | 1× | 73.1 | 91.0 |
| $I3D_{3\times3\times3}$ | 1.5× | 1.8× | 74.1 | 91.2 |
| $I3D_{3\times1\times1}$ | **1.2**× | 1.5× | 74.4 | 91.1 |
| NL C2D, 5-block | **1.2**× | **1.2**× | **75.1** | **91.7** |

(e) **Non-local *vs.* 3D Conv**: A 5-block non-local C2D *vs.* inflated 3D ConvNet (I3D) [27]. All entries are with ResNet-101. The numbers of parameters and FLOPs are relative to the C2D baseline (43.2M and 34.2B).

| | model | top-1 | top-5 |
|---|---|---|---|
| R50 | C2D baseline | 71.8 | 89.7 |
| | I3D | 73.3 | 90.7 |
| | NL I3D | **74.9** | **91.6** |
| R101 | C2D baseline | 73.1 | 91.0 |
| | I3D | 74.4 | 91.1 |
| | NL I3D | **76.0** | **92.1** |

(f) **Non-local 3D ConvNet**: 5 non-local blocks are added on top of our best I3D models. These results show that non-local operations are complementary to 3D convolutions.

| | model | top-1 | top-5 |
|---|---|---|---|
| R50 | C2D baseline | 73.8 | 91.2 |
| | I3D | 74.9 | 91.7 |
| | NL I3D | **76.5** | **92.6** |
| R101 | C2D baseline | 75.3 | 91.8 |
| | I3D | 76.4 | 92.7 |
| | NL I3D | **77.7** | **93.3** |

(g) **Longer clips**: we fine-tune and test the models in Table 5.2f on the 128-frame clips. The gains of our non-local operations are consistent.

Table 5.2: **Ablations** on Kinetics action classification. We show top-1 and top-5 classification accuracy (%).

Table 5.2 shows the ablation results, analyzed as follows:

**Instantiations.** Table 5.2a compares different types of a single non-local block added to the C2D baseline (right before the last residual block of $res_4$). Even adding one non-local block can lead to ~1% improvement over the baseline.

Interestingly, the embedded Gaussian, dot-product, and concatenation versions perform similarly, up to some random variations (72.7 to 72.9). As discussed in Sec. 5.2, the non-

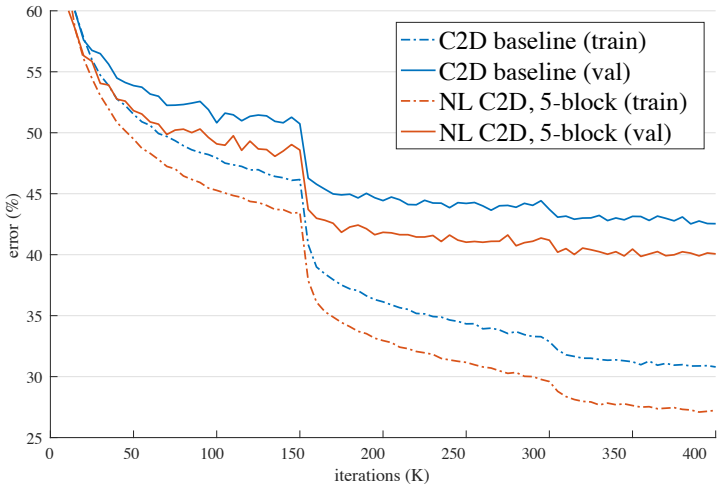



Figure 5.4: Curves of the training procedure on Kinetics for the ResNet-50 C2D baseline (blue) *vs*. non-local C2D with 5 blocks (red). We show the top-1 training error (dash) and validation error (solid). The validation error is computed in the same way as the training error (so it is 1-clip testing with the same random jittering at training time); the final results are in Table 5.2c (R50, 5-block).

local operations with Gaussian kernels become similar to the self-attention module [314]. However, our experiments show that the attentional (softmax) behavior of this module is *not* the key to the improvement in our applications; instead, it is more likely that the non-local behavior is essential, and it is insensitive to the instantiations.

In the rest of this chapter, we use the embedded Gaussian version by default. This version is easier to visualize as its softmax scores are in the range of $[0, 1]$.

**Which stage to add non-local blocks?** Table 5.2b compares a single non-local block added to different stages of ResNet. The block is added to right before the last residual block of a stage. The improvement of a non-local block on $\text{res}_2$, $\text{res}_3$, or $\text{res}_4$ is similar, and on $\text{res}_5$ is slightly smaller. One possible explanation is that $\text{res}_5$ has a small spatial size ($7\times7$) and it is insufficient to provide precise spatial information. More evidence of a non-local block exploiting spatial information will be investigated in Table 5.2d.

**Going deeper with non-local blocks.** Table 5.2c shows the results of more non-local blocks. We add 1 block (to $\text{res}_4$), 5 blocks (3 to $\text{res}_4$ and 2 to $\text{res}_3$, to every other residual block), and 10 blocks (to every residual block in $\text{res}_3$ and $\text{res}_4$) in ResNet-50; in ResNet-101 we add them to the corresponding residual blocks. Table 5.2c shows that more non-local blocks in general lead to better results. We argue that multiple non-local blocks can perform long-range multi-hop communication. Messages can be delivered back and forth between distant positions in spacetime, which is hard to do via local models.

It is noteworthy that the improvement of non-local blocks is *not* just because they add depth to the baseline model. To see this, we note that in Table 5.2c the non-local 5-block ResNet-50 model has 73.8 accuracy, higher than the deeper ResNet-101 baseline's 73.1. However, the 5-block ResNet-50 has only ~70% parameters and ~80% FLOPs of the ResNet-

| model | backbone | modality | top-1 | top-5 |
|---|---|---|---|---|
| I3D in [27] | Inception | RGB | 71.1† | 89.3† |
| 2-Stream I3D in [27] | Inception | RGB + flow | 74.2† | 91.3† |
| RGB baseline in [16] | Inception-ResNet-v2 | RGB | 73.0 | 90.9 |
| 3-stream late fusion [16] | Inception-ResNet-v2 | RGB + flow + audio | 74.9 | 91.6 |
| 3-stream LSTM [16] | Inception-ResNet-v2 | RGB + flow + audio | 77.1 | 93.2 |
| 3-stream SATT [16] | Inception-ResNet-v2 | RGB + flow + audio | 77.7 | 93.2 |
| NL I3D [ours] | ResNet-50 | RGB | 76.5 | 92.6 |
| | ResNet-101 | RGB | **77.7** | **93.3** |

Table 5.3: Comparisons with state-of-the-art results in **Kinetics**. Numbers with † were reported on the test set; otherwise on the validation set. We include the Kinetics 2017 competition winner's results [16], but their best results exploited audio signals (marked in gray) so were not vision-only solutions. †: individual top-1 or top-5 numbers are not available from the test server at the time of submitting this manuscript.

101 baseline, and is also *shallower*. This comparison shows that the improvement due to non-local blocks is complementary to going deeper in standard ways.

We have also tried to add standard residual blocks, instead of non-local blocks, to the baseline models. The accuracy is not increased. This again shows that the improvement of non-local blocks is not just because they add depth.

**Non-local in spacetime.** Our method can naturally handle spacetime signals. This is a nice property: related objects in a video can present at distant space and long-term time interval, and their dependency can be captured by our model.

In Table 5.2d we study the effect of non-local blocks applied along space, time, or spacetime. For example, in the space-only version, the non-local dependency only happens within the same frame: *i.e.*, in Eq.(5.1) it only sums over the index $j$ in the same frame of the index $i$. The time-only version can be set up similarly. Table 5.2d shows that both the space-only and time-only versions improve over the C2D baseline, but are inferior to the spacetime version.

**Non-local net *vs.* 3D ConvNet.** Table 5.2e compares our non-local C2D version with the inflated 3D ConvNets. Non-local operations and 3D convolutions can be seen as two ways of extending C2D to the temporal dimensions.

Table 5.2e also compares the number of parameters and FLOPs, relative to the baseline. Our non-local C2D model is more accurate than the I3D counterpart (*e.g.*, 75.1 *vs.* 74.4), while having a smaller number of FLOPs (1.2× *vs.* 1.5×). This comparison shows that our method can be more effective than 3D convolutions when used alone.

**Non-local 3D ConvNet.** Despite the above comparison, non-local operations and 3D convolutions can model different aspects of the problem: 3D convolutions can capture local dependency. Table 5.2f shows the results of inserting 5 non-local blocks into the $\text{I3D}_{3\times1\times1}$ models. These non-local I3D (NL I3D) models improve over their I3D counterparts (+1.6 point accuracy), showing that non-local operations and 3D convolutions are complementary.

**Longer sequences.** Finally we investigate the generality of our models on longer input videos. We use input clips consisting of 128 consecutive frames without subsampling. The sequences throughout all layers in the networks are thus 4× longer compared to the 32-frame counterparts. To fit this model into memory, we reduce the mini-batch size to 2 clips per GPU. As a result of using small mini-batches, we freeze all BN layers in this case. We initialize

| model | modality | *train/val* | *trainval/test* |
|---|---|---|---|
| 2-Stream [271] | RGB + flow | 18.6 | - |
| 2-Stream +LSTM [271] | RGB + flow | 17.8 | - |
| Asyn-TF [271] | RGB + flow | 22.4 | - |
| I3D [27] | RGB | 32.9 | 34.4 |
| I3D [ours] | RGB | 35.5 | 37.2 |
| NL I3D [ours] | RGB | **37.5** | **39.5** |

Table 5.4: Classification mAP (%) in the **Charades** dataset [273], on the *train/val* split and the *trainval/test* split. Our results are based on ResNet-101. Our NL I3D uses 5 non-local blocks.

this model from the corresponding models trained with 32-frame inputs. We fine-tune on 128-frame inputs using the same number of iterations as the 32-frame case (though the mini-batch size is now smaller), starting with a learning rate of 0.0025. Other implementation details are the same as before.

Table 5.2g shows the results of 128-frame clips. Comparing with the 32-frame counterparts in Table 5.2f, all models have better results on longer inputs. We also find that our NL I3D can maintain its gain over the I3D counterparts, showing that our models work well on longer sequences.

**Comparisons with state-of-the-art results.** Table 5.3 shows the results from the I3D authors [27] and from the Kinetics 2017 competition winner [16]. We note that these are comparisons of systems which can differ in many aspects. Nevertheless, our method surpasses all the existing RGB or RGB + flow based methods by a good margin. *Without using optical flow and without any bells and whistles*, our method is on par with the heavily engineered results of the 2017 competition winner.

### Experiments on Charades

Charades [273] is a video dataset with ~8k training, ~1.8k validation, and ~2k testing videos. It is a multi-label classification task with 157 action categories. We use a per-category sigmoid output to handle the multi-label property.

We initialize our models pre-trained on Kinetics (128-frame). The mini-batch size is set to 1 clip per GPU. We train our models for 200k iterations, starting from a learning rate of 0.00125 and reducing it by 10 every 75k iterations. We use a jittering strategy similar to that in Kinetics to determine the location of the 224×224 cropping window, but we rescale the video such that this cropping window outputs 288×288 pixels, on which we fine-tune our network. We test on a single scale of 320 pixels.

Table 5.4 shows the comparisons with the previous results on Charades. The result of [27] is the 2017 competition winner in Charades, which was also fine-tuned from models pre-trained in Kinetics. Our I3D baseline is higher than previous results. As a controlled comparison, our non-local net improves over our I3D baseline by 2.3% on the test set.

## 5.5 Extension: Experiments on COCO

We also investigate our models on static image recognition. We experiment on the Mask R-CNN baseline [112] for COCO [205] object detection/segmentation and human pose estimation (keypoint detection). The models are trained on COCO `train2017` (*i.e.*, `trainval35k`

| method | | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | $AP^{mask}$ | $AP^{mask}_{50}$ | $AP^{mask}_{75}$ |
|---|---|---|---|---|---|---|---|
| R50 | baseline | 38.0 | 59.6 | 41.0 | 34.6 | 56.4 | 36.5 |
| | +1 NL | **39.0** | **61.1** | **41.9** | **35.5** | **58.0** | **37.4** |
| R101 | baseline | 39.5 | 61.4 | 42.9 | 36.0 | 58.1 | 38.3 |
| | +1 NL | **40.8** | **63.1** | **44.5** | **37.1** | **59.9** | **39.2** |
| X152 | baseline | 44.1 | 66.4 | 48.4 | 39.7 | 63.2 | 42.2 |
| | +1 NL | **45.0** | **67.8** | **48.9** | **40.3** | **64.4** | **42.8** |

Table 5.5: Adding 1 non-local block to Mask R-CNN for COCO **object detection** and **instance segmentation**. The backbone is ResNet-50/101 or ResNeXt-152 [365], both with FPN [204].

| model | $AP^{kp}$ | $AP^{kp}_{50}$ | $AP^{kp}_{75}$ |
|---|---|---|---|
| R101 baseline | 65.1 | 86.8 | 70.4 |
| NL, +4 in head | 66.0 | 87.1 | 71.7 |
| NL, +4 in head, +1 in backbone | **66.5** | **87.3** | **72.8** |

Table 5.6: Adding non-local blocks to Mask R-CNN for COCO **keypoint detection**. The backbone is ResNet-101 with FPN [204].

in 2014) and tested on `val2017` (*i.e.*, `minival` in 2014).

**Object detection and instance segmentation.** We modify the Mask R-CNN backbone by adding one non-local block (right before the last residual block of $\text{res}_4$). All models are fine-tuned from ImageNet pre-training. We evaluate on a standard baseline of ResNet-50/101 and a high baseline of ResNeXt-152 (X152) [365]. Unlike the original paper [112] that adopted stage-wise training regarding RPN, we use an improved implementation with end-to-end joint training similar to [250], which leads to higher baselines than [112].

Table 5.5 shows the box and mask AP on COCO. We see that a single non-local block improves all R50/101 and X152 baselines, on all metrics involving detection and segmentation. $AP^{box}$ is increased by $\sim$1 point in all cases (*e.g.*, +1.3 point in R101). Our non-local block has a nice effect *complementary* to increasing the model capacity, even when the model is upgraded from R50/101 to X152. This comparison suggests that *non-local dependency has not been sufficiently captured by existing models despite increased depth/capacity*.

In addition, the above gain is at a very small cost. The single non-local block only adds $<$5% computation to the baseline model. We also have tried to use more non-local blocks to the backbone, but found diminishing return.

**Keypoint detection.** Next we evaluate non-local blocks in Mask R-CNN for keypoint detection. In [112], Mask R-CNN used a stack of 8 convolutional layers for predicting the keypoints as 1-hot masks. These layers are local operations and may overlook the dependency among keypoints across long distance. Motivated by this, we insert 4 non-local blocks into the keypoint head (after every 2 convolutional layers).

Table 5.6 shows the results on COCO. On a strong baseline of R101, adding 4 non-local blocks to the keypoint head leads to a $\sim$1 point increase of keypoint AP. If we add one extra non-local block to the backbone as done for object detection, we observe an in total 1.4 points increase of keypoint AP over the baseline. In particular, we see that the stricter criterion of $AP_{75}$ is boosted by 2.4 points, suggesting a stronger localization performance.

## 5.6 Discussion

We presented a new class of neural networks which capture long-range dependencies via non-local operations. Our non-local blocks can be combined with any existing architectures. We show the significance of non-local modeling for the tasks of video classification, object detection and segmentation, and pose estimation. On all tasks, a simple addition of non-local blocks provides solid improvement over baselines. We hope non-local layers will become an essential component of future network architectures.

# Chapter 6

# Videos as Space-Time Region Graphs

We have shown encouraging results in action recognition in videos with Non-local operator in the last chapter. However, the Non-local operator is applied in every pixel in the feature space (from low layers to higher layers), which makes it inefficient and redundant. Moreover, the non-local operator does not process any temporal ordering information. To solve these two problems, we propose to model the videos as space-time region graphs, and perform reasoning on top.

Consider a simple action such as "opening a book" as shown in Fig. 6.1. When we humans see the sequence of images, we can easily recognize the action category; yet our current vision systems (with hundreds of layers of 3D convolutions) struggle on this simple task. Why is that? What is missing in current video recognition frameworks?

Let's first take a closer look at the sequence shown in Fig. 6.1. How do humans recognize the action in the video corresponds to "opening a book"? We argue that there are two key ingredients to solving this problem: First, the shape of the book and how it changes over time (i.e., the object state changes from closed to open) is a crucial cue. Exploiting this cue requires temporally linking book regions across time and modeling actions as transformations. But just modeling temporal dynamics of objects is not sufficient. The state of objects change after interaction with human or other objects. Thus we also need to model human-object and object-object interactions as well for action recognition.

However, our current deep learning approaches fail to capture these two key ingredients. For example, the state-of-the-art approaches based on two-stream ConvNets [281, 333] are still learning to classify actions based on individual video frame or local motion vectors. Local motion clearly fails to model the dynamics of shape changes. To tackle this limitation, recent work has also focused on modeling long term temporal information with Recurrent Neural Networks [50, 195, 224, 381] and 3D Convolutions [27, 306, 308, 366]. However, all these frameworks focus on the features extracted from the whole scenes and fail to capture long-range temporal dependencies (transformations) or region-based relationships. In fact, most of the actions are classified based on the background information instead of capturing the key objects (e.g., the book in "opening a book") as observed in [272].

On the other hand, there have been several efforts to specifically model the human-

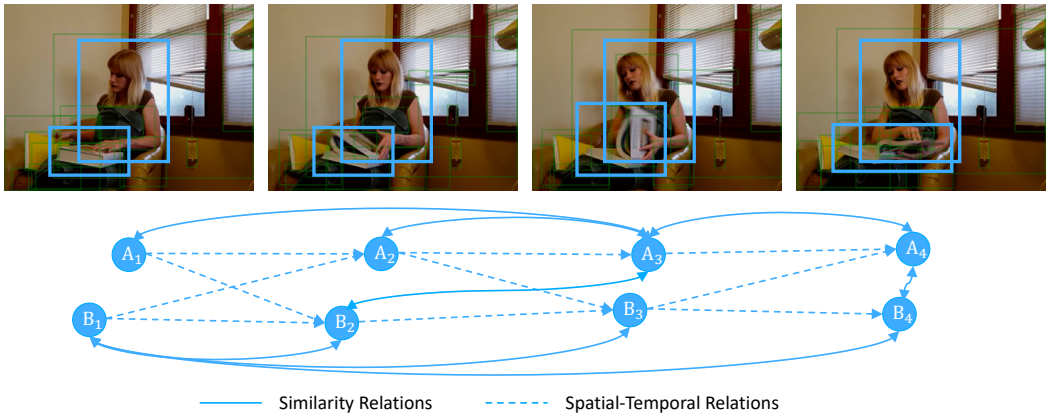



Figure 6.1: How do you recognize simple actions such as opening book? We argue action understanding requires appearance modeling but also capturing temporal dynamics (how shape of book changes) and functional relationships. We propose to represent videos as space-time region graphs followed by graph convolutions for inference.

object or object-object interactions [103, 374]. This direction have been recently revisited with ConvNets in an effort to improve object detection [88, 130, 376], visual relationship detection [212] and action recognition [87], etc. However, the relationship reasoning is still performed in static images failing to capture temporal dynamics of these interactions. Thus, it is very hard for these approaches to capture the changes of object states over time as well as the causes and effects of these changes.

In this chapter, we propose to perform long-range temporal modeling of human-object and object-object relationships via a graph-based reasoning framework. Unlike existing approaches which focus on local motion vectors, our model takes in a long range video sequence (e.g., more than 100 frames or 5 seconds). We represent the input video as a **space-time region graph** where each node in the graph represent region of interest in the video. Region nodes are connected by two types of edges: appearance-similarity and spatio-temporal proximity. Specifically, **(i) Similarity Relations**: regions which have similar appearance or semantically related are connected together. With similarity relations, we can model how the states of the same object change and the long range dependencies between any two objects in any frames. **(ii) Spatial-Temporal Relations**: objects which overlap in space and close in time are connected together via these edges. With spatial-temporal relations, we can capture the interactions between nearby objects as well as the temporal ordering of object state changes.

Given the graph representation, we perform reasoning on the graph and infer the action by applying the Graph Convolution Networks (GCNs) [162]. We conduct our experiments in the challenging Charades [273] and 20BN-Something-Something [98] datasets. Both datasets are extremely challenging as the actions cannot be easily inferred by the background of the scene and the 2D appearance of the objects or humans. Our model shows significant improvements over state-of-the-art results of action recognition. Especially in the Charades dataset, we obtain $4.4\%$ boost.

Our contributions include: (a) A novel graph representation with variant relationships between different objects in a long range video; (b) A graph convolutional network model

for reasoning with multiple relation edges; (c) state-of-the-art performance with a significant gain in action recognition in complex environments.

## 6.1 Background

**Video Understanding Models.** Spatio-temporal reasoning is one of the core research areas in the field of video understanding and action recognition. However, most of the early work has focused on using spatio-temporal appearance features. For example, a large effort has been spent on manually designing the video features [41, 165, 179–181, 236, 264, 326, 348, 398]. Some of the hand-designed features such as the Improved Dense Trajectory (IDT) [326] are still widely applied and show very competitive results in different video related tasks. However, instead of designing hand-crafted features, recent researches have focused towards learning deep representations from the video data [153, 186, 281, 304, 331, 333, 335, 390]. One of the most popular model is the two-Stream ConvNets [281] where temporal information is model by a network with 10 optical flow frames as inputs ($< 1$ second). To better model longer-term information, a lot of work has been focused on using Recurrent Neural Networks (RNNs) [16, 50, 77, 195, 233, 293, 297, 364, 381] and 3D ConvNets [27, 62, 145, 244, 307, 308, 365]. However, these frameworks focus on extracting features from the whole scenes and can hardly model the relationships between different object instances in space and time.

**Visual Relationships.** Reasoning about the pairwise relationships has been proven to be very helpful in a variety of computer vision tasks [103, 175, 263, 374, 375]. For example, object detection in cluttered scenes can be significantly improved by modeling the human-object interactions [374]. Recently, the visual relationships have been widely applied together with deep networks in the area of visual question answering [266], object recognition [88, 130, 376] and intuitive physics [10, 350]. In the case of action recognition, a lot of effort has been made on modeling pairwise human-object and object-object relationships [86, 216, 230]. However, the interaction reasoning framework in these efforts focus on static images and the temporal information is usually modeled by a RNN on image level features. Thus, these approaches still cannot capture how a certain object state changes or rather transformations over time.

**Graphical Models.** The long range relationships in images and videos are usually captured by graphical models. One popular direction is using the Conditional Random Fields (CRF) [168, 177]. In the context of deep learning, especially for semantic segmentation, the CRF model is often applied on the outputs of the ConvNets by performing mean-field inference [28, 31, 111, 169, 268, 388]. Instead of using mean-field inference, variant simpler feed-forward graph based neural network have been proposed recently [162, 200, 208, 218, 267, 370]. In this chapter, we apply the Graph Convolutional Networks (GCNs) [162] which was originally proposed for applications in Natural Language Processing. Our GCN is built by stacking multiple layers of graph convolutions with similarity relations and spatial-temporal relations. The outputs of the GCNs are updated features for each object node, which can be used to perform classification.

Our work is also related to video recognition with object cues [4, 115, 363] and object graph models [17, 30, 138, 380]. For example, Structural-RNN [138] is proposed to model the spatial-temporal relations between objects (adjacent in time) for video recognition tasks. Different from these works, our space-time graph representation encodes not only local relations but also long range dependencies between any pairs of objects across space and time. By using graph convolutions with long range relations, it enables efficient message passing between starting states and ending states of the objects. This global graph reasoning

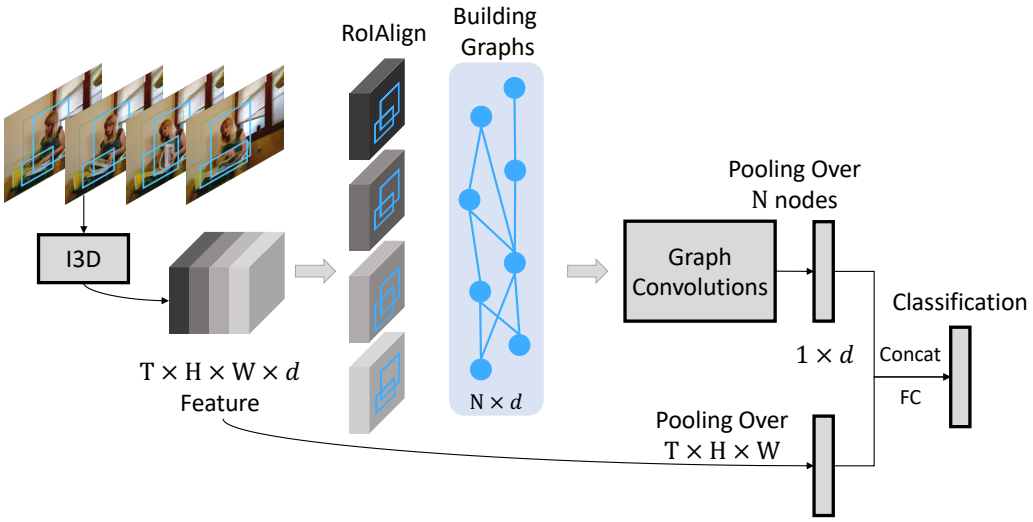



Figure 6.2: Model Overview. Our model uses 3D convolutions to extract visual features followed by RoIAlign extracting $d$-dimension feature for each object proposal. These features are provided as inputs to the Graph Convolutional Network which performs information propagation based on spatiotemporal edges. Finally, a $d$-dimension feature is extracted and appended to another $d$-dimension video feature to perform classification.

framework provides significant boost over the state-of-the-art.

## 6.2 Overview

Our goal is to represent the video as a graph of objects and perform reasoning on the graph for action recognition. The overview of our model is visualized in Figure 6.2. Our model takes inputs as a long clip of video frames (more than 5 seconds) and forward them to a 3D Convolutional Neural Network [27, 340]. The output of this 3D ConvNet is a feature map with the dimensions $\mathrm{T} \times \mathrm{H} \times \mathrm{W} \times d$, where T represents the temporal dimension, $\mathrm{H} \times \mathrm{W}$ represents the spatial dimensions and $d$ represents the channel number.

Besides extracting the video features, we also apply a Region Proposal Network (RPN) [249] to extract the object bounding boxes (We have not visualized the RPN in Figure 6.2 for simplicity). Given the bounding boxes for each of the T feature frames, we apply RoIAlign [81, 112] to extract the features for each bounding box. Note that the RoIAlign is applied on each feature frame independently. The feature vector for each object has $d$ dimensions (first aligned to $7 \times 7 \times d$ and then maxpooled to $1 \times 1 \times d$). We denote the object number as N, thus the feature dimension is $\mathrm{N} \times d$ after RoIAlign.

We now construct a graph which contains N nodes corresponding to N object proposals aggregated over T frames. There are mainly two types of relations in the graph: similarity relations and spatial-temporal relations. For simplicity, we decompose this big graph into two sub-graphs with the same nodes but two different relations: the similarity graph and the spatial-temporal graph.

With the graph representations, we apply the Graph Convolutional Networks (GCNs) to perform reasoning. The output for the GCNs are in the same dimension as the input features

| layer | | output size |
|---|---|---|
| $\text{conv}_1$ | 5×7×7, 64, stride 1, 2, 2 | 32×112×112 |
| $\text{pool}_1$ | 1×3×3 max, stride 1, 2, 2 | 32×56×56 |
| $\text{res}_2$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$ | 32×56×56 |
| $\text{pool}_2$ | 3×1×1 max, stride 2, 1, 1 | 16×56×56 |
| $\text{res}_3$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$ | 16×28×28 |
| $\text{res}_4$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$ | 16×14×14 |
| $\text{res}_5$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$ | 16×14×14 |
| global average pool, fc | | 1×1×1 |

Table 6.1: Our baseline ResNet-50 I3D model. We use T×H×W to represent the dimensions of filter kernels and 3D output feature maps. For filter kernels, we also have number of channels following T×H×W. The input is in 32×224×224 dimensions and the residual blocks are shown in brackets.

which is $\mathrm{N} \times d$. We perform average pooling over all the object nodes to obtain a $d$-dimension feature. Besides the GCN features, we also perform average pooling on the whole video representation ($\mathrm{T} \times \mathrm{H} \times \mathrm{W} \times d$) to obtain the same $d$-dimension feature as a global feature. These two features are then concatenated together for video level classification.

We will introduce the details of each component in the following sections. We introduce the process of feature extraction and graph representations in Section 4 and the Graph Convolutional Networks (GCNs) in Section 5.

## 6.3 Graph Representations in Videos

In this section, we will first introduce the feature extraction process for our model with 3D ConvNets and then describe the construction of the similarity graph as well as the spatial-temporal graph.

### 6.3.1 Video Representation

**Video Backbone Model.** Given a long clip of video (around 5 seconds), we sample 32 video frames from it with the same temporal duration between every two frames. We extract the features on these frames via a 3D ConvNet. Table 6.1 shows our backbone model based on the ResNet-50 architecture, which are motivated by the model architecture mentioned in [340]. The model takes input as 32 video frames with $224 \times 224$ dimensions and the output of the last convolutional layer is a $16 \times 14 \times 14$ feature map (i.e., 16 frames in the temporal dimension and $14 \times 14$ in the spatial dimension). The baseline method in this chapter adopts

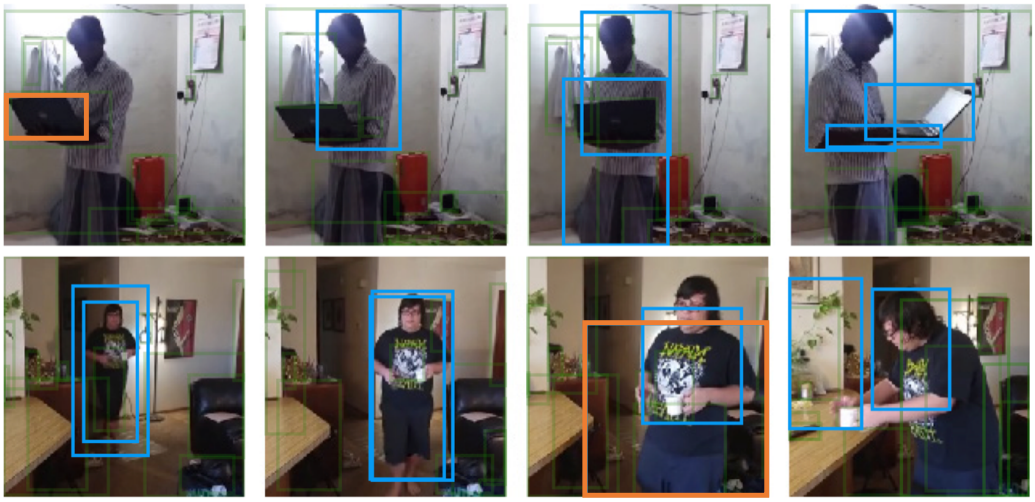


Figure 6.3: Similarity Graph $\mathbf{G}^{sim}$. Above figure shows our similarity graph not only captures similarity in visual space but also correlations (similarity in functional space). The query box is shown in orange, the nearest neighbors are shown in blue. The transparent green boxes are the other unselected object proposals.

the same architecture, and the classification is simply performed by using a global average pooling on the final convolutional features and then following by a fully connected layer.

This backbone model is called Inflated 3D ConvNet (I3D) [27, 62, 340] as one can turn a 2D ConvNet into a 3D ConvNet by inflating the kernels during initialization. That is, a 3D kernel with $t \times k \times k$ dimensions can be inflated from a 2D $k \times k$ kernel by copying the weights $t$ times and rescaling by $1/t$. Please refer to [27, 62, 340] for more initialization details.

**Region Proposal Network.** We apply the Region Proposal Network (RPN) in [85, 249] to generate the object bounding boxes of interest on each video frame. More specifically, we use the RPN with ResNet-50 backbone and FPN [204]. The RPN is pre-trained with the MSCOCO object detection dataset [205] and there is no weight sharing between the RPN and our I3D video backbone model. Note that the bounding boxes extracted by the RPN are class-agnostic.

To extract object features on top of the last convolutional layer, we project the bounding boxes from the 16 input RGB frames (which are sampled from the 32 input frames for I3D, with the sampling rates of 1 frame every 2 frames) to the 16 output feature frames. Taking the video features and projected bounding boxes, we apply RoIAlign [112] to extract the feature for each object proposal. Note that RoIAlign is similar to RoIPooling [81] which crops and rescales the object features into the same dimensions. In RoIAlign, each output frame is processed independently. The RoIAlign generates a $7 \times 7 \times d$ output features for each object which is then max-pooled to $1 \times 1 \times d$ dimensions.

### 6.3.2 Similarity Graph

We measure the similarity between objects in the feature space to construct the similarity graph. In this graph, we connect pairs of semantically related objects together. More

specifically, we will have a high confidence edge between two instances which are: (i) the same object in different states in different video frames or (ii) highly correlated for recognizing the actions. Note that the similarity edges are computed between any pairs of objects.

Formally, assuming we have the features for all the object proposals in the video as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_\mathrm{N}\}$, where $\mathrm{N}$ represents the number of object proposals and each object proposal feature $\mathbf{x}_i$ is a $d$ dimensional vector. The pairwise similarity or the affinity between every two proposals can be represented as,

$$F(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi'(\mathbf{x}_j), \tag{6.1}$$

where $\phi$ and $\phi'$ represents two different transformations of the original features. More specifically, we have $\phi(\mathbf{x}) = \mathbf{w}\mathbf{x}$ and $\phi'(\mathbf{x}) = \mathbf{w}'\mathbf{x}$. The parameters $\mathbf{w}$ and $\mathbf{w}'$ are both $d \times d$ dimensions weights which can be learned via back propagation. By adding the transformation weights $\mathbf{w}$ and $\mathbf{w}'$, it allows us to not only learn the correlations between different states of the same object instance across frame, but also the relations between different objects. We visualize the top nearest neighbors for the object proposals in Figure 6.3. In the first example, we can see the nearest neighbors of the laptop not only include the other laptop instances in other frames, but also the human who is operating it.

After computing the affinity matrix with Eq. 6.1, we perform normalization on each row of the matrix so that the sum of all the edge values connected to one proposal $i$ will be $1$. Motivated by the recent works [314, 340], we adopt the softmax function for normalization as,

$$\mathbf{G}_{ij}^{sim} = \frac{\exp F(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j=1}^{\mathrm{N}} \exp F(\mathbf{x}_i, \mathbf{x}_j)}. \tag{6.2}$$

The normalized $\mathbf{G}^{sim}$ is taken as the adjacency matrix representing the similarity graph.

### 6.3.3 Spatial-Temporal Graph

Although the similarity graph captures even the long term dependencies between any two object proposals, it does not capture the relative spatial relation between objects and the ordering of the state changes. To encode these spatial and temporal relations between objects, we propose to use spatial-temporal graphs, where objects in nearby locations in space and time are connected together.

Given a object proposal in frame $t$, we calculate the value of Intersection Over Unions (IoUs) between this object bounding box and all other object bounding boxes in frame $t+1$. We denote the IoU between object $i$ in frame $t$ and object $j$ in frame $t+1$ as $\sigma_{ij}$. If $\sigma_{ij}$ is larger than $0$, we will link object $i$ to object $j$ using a directed edge $i \to j$ with value $\sigma_{ij}$. After assigning the edge values, we normalize the graph so that the sum of the edge values connected to proposal $i$ will be $1$ by

$$\mathbf{G}_{ij}^{front} = \frac{\sigma_{ij}}{\sum_{j=1}^{\mathrm{N}} \sigma_{ij}}, \tag{6.3}$$

where $\mathbf{G}^{front}$ is taken as the adjacency matrix for a spatial-temporal graph. We visualize some of the object proposals and the trajectories in Figure 6.4.

Besides building the forward graph which connects objects from frame $t$ to frame $t+1$, we also construct a backward graph in a similar way which connect objects from frame $t+1$

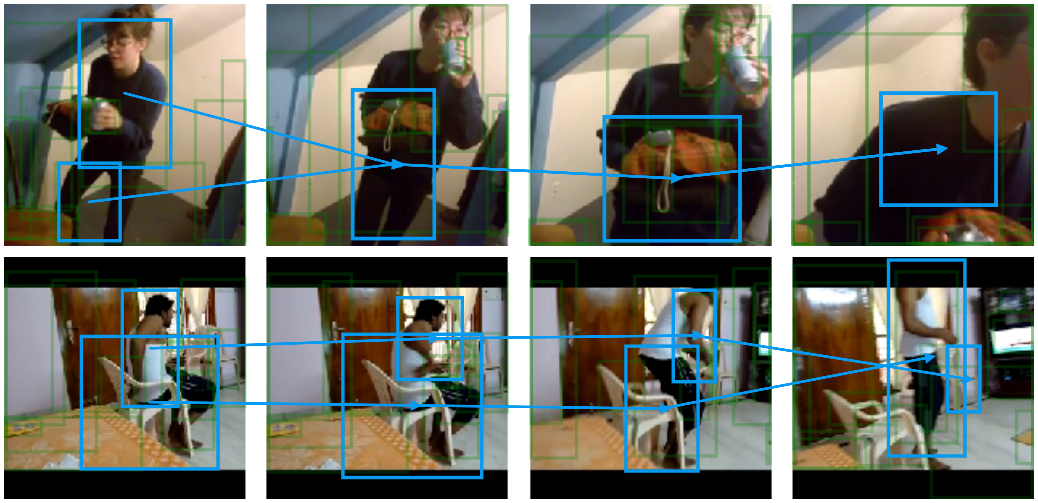


Figure 6.4: Spatial-Temporal Graph $\mathbf{G}^{front}$. Highly overlapping object proposals across neighboring frames are linked by directed edge. We plot some example trajectories with blue boxes and the direction shows the arrow of time.

to frame $t$. We denote the adjacency matrix of this backward graph as $\mathbf{G}^{back}$. Specifically, for the overlapping object $i$ in frame $t$ and object $j$ in frame $t+1$, we construct an edge $i \leftarrow j$ and assign the values to $\mathbf{G}^{back}_{ji}$ according to the IoU values. By building the spatial-temporal graphs in a bidirectional manner, we can obtain richer structure information and enlarge the number of propagation neighborhoods during graph convolutions.

## 6.4 Convolutions on Graphs

To perform reasoning on the graph, we apply the Graph Convolutional Networks (GCNs) proposed in [162]. Different from standard convolutions which operates on a local regular grid, the graph convolutions allow us to compute the response of a node based on its neighbors defined by the graph relations. Thus performing graph convolutions is equal to performing message passing inside the graphs. The outputs of the GCNs are updated features of each object node, which can be aggregated together for video classification. Formally, we can represent the one layer of graph convolutions as,

$$\mathbf{Z} = \mathbf{G}\mathbf{X}\mathbf{W}, \tag{6.4}$$

where $\mathbf{G}$ represents one of the adjacency graph we have introduced ($\mathbf{G}^{sim}$, $\mathbf{G}^{front}$ or $\mathbf{G}^{back}$) with $\mathrm{N} \times \mathrm{N}$ dimensions, $\mathbf{X}$ is the input features of the object nodes in the graph with $\mathrm{N} \times d$ dimensions, and $\mathbf{W}$ is the weight matrix of the layer with dimension $d \times d$ in our case. Thus the output of one graph convoltuional layer $\mathbf{Z}$ is still in $N \times d$ dimensions. The graph convolution operation can be stacked into multiple layers. After each layer of graph convolutions, we apply two non-linear functions including the Layer Normalization [6] and then ReLU before the feature $\mathbf{Z}$ is forwarded to the next layer.

**Connecting GCN and Non-local Net.** We would also like to draw the connections between the Graph Convolutional Networks with the Similarity Graph ($\mathbf{G}^{sim}$) and the recent proposed Non-local Neural Networks [340]. If we apply the non-local operation on the

region proposals, it can be represented as,

$$\mathbf{Y} = \mathbf{G}^{sim} g(\mathbf{X}), \tag{6.5}$$

where $g$ is a function with a convolutional layer. The non-local block in [340] can be further formulated as,

$$\mathbf{Z} = \mathbf{Y}\mathbf{W} + \mathbf{X} = \mathbf{G}^{sim} g(\mathbf{X})\mathbf{W} + \mathbf{X}, \tag{6.6}$$

which is very similar to the graph convolution operation in Eq. 6.4. Inspired by this, we modify the graph convolution operations by adding a convolutional operator on the input $\mathbf{X} = g(\mathbf{X})$ for the first layer (note that $\mathbf{G}^{sim}$ is still computed based on the features before applying $g$). We also add a residual connection in every layer of GCN, which extends Eq. 6.4 as,

$$\mathbf{Z} = \mathbf{G}\mathbf{X}\mathbf{W} + \mathbf{X}. \tag{6.7}$$

**Combining Multiple Graphs.** To combine multiple graphs in GCNs, we can simply extend Eq. 6.7 as,

$$\mathbf{Z} = \sum_i \mathbf{G}_i \mathbf{X} \mathbf{W}_i + \mathbf{X}, \tag{6.8}$$

where $\mathbf{G}_i$ indicates different types of graphs, and the weights for different graphs $\mathbf{W}_i$ are not shared. Note that in this way, each hidden layer of the GCN is updated though the relationships from different graphs. However, we find that the direct combination of 3 graphs ($\mathbf{G}^{sim}$, $\mathbf{G}^{front}$ and $\mathbf{G}^{back}$) with Eq. 6.8 actually hurts the performance compared to the situation with a single similarity graph.

The reason is that our similarity graph $\mathbf{G}^{sim}$ contains learnable parameters (Eq. 6.1) and requires back propagation for updating, while the other two graphs do not require learning. Fusing these graphs together in every GCN layer increases the optimization difficulties. Thus we create two branches of graph convolutional networks, and only fuse the results from two GCNs in the end: one GCN adopts Eq. 6.4 with $\mathbf{G}^{sim}$ and the other GCN adopts Eq. 6.8 with $\mathbf{G}^{front}$ and $\mathbf{G}^{back}$. These two branches of GCNs perform convolutions separately for $L$ layers and the final layer features are summed together, which is in $\mathrm{N} \times d$ dimensions.

**Video Classification.** As illustrated in Figure 6.2, the updated features after graph convolutions are forwarded to an average pooling layer, which calculates the mean of all the proposal features and leads to a $1 \times d$ dimension representation. Besides the GCN features, we also perform average pooling on the whole video level representation and obtain the another $1 \times d$ dimensions of global features. These two features are then concatenated together for video classification. The classification training loss is defined depending on the tasks (multi-label or single label classifications).

## 6.5 Experiments

We perform the experiments on two recent challenging datasets: Charades [273] and Something-Something [98]. We first introduce the implementation details of our approach and then the evaluation results on these datasets.

### 6.5.1 Implementation Details

**Training.** The training of our backbone models involves pre-training on 2 different datasets following [27, 340]. The model is first pre-trained as a 2D ConvNet with the ImageNet

dataset [260] and then inflated into a 3D ConvNet (i.e., I3D) as [27]. We then fine-tuned the 3D ConvNet with the Kinetics action recognition dataset [154] following the same training scheme for longer sequences (around 5 second video) in [340]. Given this initialization, we now introduce how to further fine-tune the network on our target datasets (e.g. Charades or Something-Something) as following.

As specified in Table 6.1, our network takes 32 video frames as inputs. These 32 video frames are sampled in the frame rate of 6fps, thus the temporal length of the video clip is around 5 seconds. The spatial dimensions for input is $224 \times 224$. Following [282], the input frames are randomly cropped from a randomly scaled video whose shorter side is sampled in $[256, 320]$ dimensions. To reduce the number of GCN parameters, we add one more $1 \times 1 \times 1$ convolutional layer on top of the I3D baseline model, which reduces the output channel number from $2048$ to $d = 512$. Since both Charades and Something-Something dataset are in similar scales in number of video frames, we adopt the same learning rate schedule for both datasets.

Our baseline I3D model is trained with a 4-GPU machine where each GPU has 2 video clips in a mini-batch. Thus the total batch size is 8 clips during training. Note that we freeze the parameters in all Batch Normalization (BN) layers during training. Our model is trained for 100K iterations in total, with learning rate $0.00125$ in the first 90K iterations and it is reduced by a factor of 10 during training the last 10K iterations. Dropout [123] is applied on the last global pooling layer with a ratio of $0.3$.

We set the layer number of our Graph Convolutional Network to 3. The parameters of the convolutional operations $\phi, \phi'$ and $g$ are initialized with Gaussian distribution having standard deviation of $0.01$. The parameters of kernels in graph convolutions $W$ are initialized as zero inspired by [97]. To train the GCN together with the I3D backbone, we propose to apply stage-wise training. We first finetune the I3D model as mentioned above, then we apply RoIAlign and GCN on top of the final convolutional features as shown in Figure 6.2. We fix the I3D features and train the GCN with the same learning rate schedules as for training the backbone. Then we train the I3D and GCN together end-to-end for 30K more iterations with the reduced learning rate.

*Task specific settings.* We apply different loss functions when training for Charades and Something-Something datasets. For Something-Something dataset, we can simply apply the softmax loss function. For Charades, we apply binary sigmoid loss, one for each action class, to handle the multi-label property. We also extract different numbers of object bounding boxes with RPN in two different datasets. As for Charades, the scenes are more cluttered and we extract 50 object proposals for each frame. However, for Something-Something, there is usually only one or two objects in the center of video frame and one hand is interacting with it. We find that extracting 10 object proposals each frame is enough for the Something-Something dataset.

**Inference.** We perform fully-convolutional inference in space as [282, 340] during inference. Note that we rescale the shorter side of each video frame to $256$ while maintaining the aspect ratios. To perform inference on one whole video, we sample 10 clips for Charades and 2 clips for Something-Something according to the average video length in two different datasets. Results from multiple clips are aggregated together by Max-Pooling over the scores.

| model, R50, I3D | mAP |
|---|---|
| baseline | 31.8 |
| Proposal+AvgPool | 32.1 |
| Spatial-Temporal GCN | 34.2 |
| Similarity GCN | 35.0 |
| Joint GCN | **36.2** |

(a) We perform ablation studies with GCN using the ResNet-50, I3D backbone.

| model, R50, I3D | mAP |
|---|---|
| baseline | 31.8 |
| Non-local | 33.5 |
| Joint GCN | 36.2 |
| Non-local + Joint GCN | **37.5** |

(b) We first compare our approach with Non-local Net and then combine Non-local Net with our model.

Table 6.2: **Ablations** on Charades. We show the mean Average Precision (mAP%).

### 6.5.2 Experiments on Charades

In the Charades experiments, following the official split, we use the 8K training videos to train our model and perform testing on the 1.8K validation videos. The average video duration is around 30 seconds. There are 157 action classes and multiple actions can happen at the same time.

**How much each graph helps?** We first perform analysis on each component of our framework, with the backbone of ResNet-50 I3D, as illustrated in Table 6.2a. We first show that the result of I3D baseline without any proposal extractions and graph convolutions is $31.8\%$ mAP on the validation set.

One simple extension on this baseline is: obtain the region proposals with RPN, extract the features for each proposal and perform average pooling over them as an extra feature. We concatenate the video level feature and the proposal feature together for classification. However, we can only obtain $0.3\%$ boost with this approach. Thus, a naive aggregation of proposal features does not help much.

We then perform evaluations by applying GCNs with the similarity graph and the spatial-temporal graph individually. We observe that our GCN model with only spatial-temporal graph can obtain a boost of $2.4\%$ over the baseline model and achieve $34.2\%$. With the similarity graph, we can achieve a better performance of $35.0\%$. By combining two graphs together and train GCNs with multiple relations, our method achieves $36.2\%$ mAP which is a significant boost of $4.4\%$ over the baseline.

**Robustness to Proposal Numbers.** Besides studying on each sub-graph, we also analyze how the number of object proposals generated by the RPN affect our method. Note that the baseline achieves $31.8\%$ and our method achieves $36.2\%$ with extracting 50 object proposals per video frame. If we reduce the number of object proposals and extract 25 proposals per frame, the mAP of our method is $35.9\%$. If we increase and double the number of object proposals to 100 proposals per frame, the performance of our method is $36.1\%$ mAP. Thus our approach is actually very stable with the changes of RPN.

**Model Complexity.** Given this large improvement in performance, the extra computation cost of the GCN over the baseline is actually very small. In the Charades dataset, our graph is defined based on 800 object nodes per video (with 16 output frames and 50 object proposals per frame). The computations of GCN with an 800-node graph is very small. The FLOPs of the baseline I3D model is $153 \times 10^9$ and the total FLOPs of our model (I3D + Joint GCN)

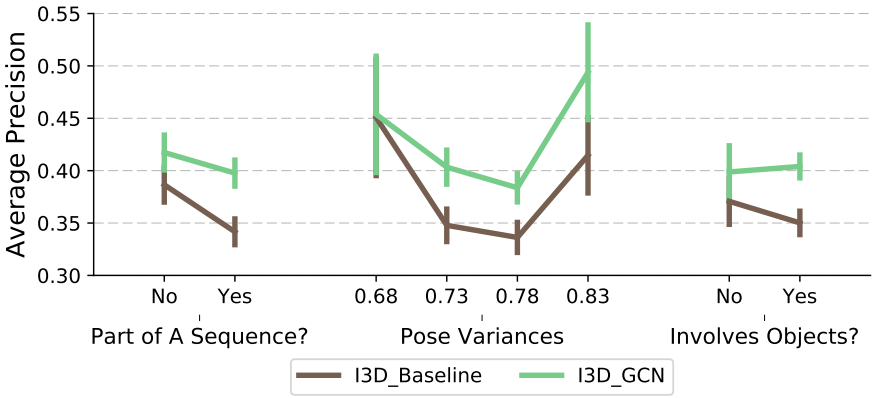



Figure 6.5: Error Analysis. We compare our approach against baseline I3D approach across three different attributes. Our approach improves significantly when action is part of sequence, involves interaction with objects and has high pose variance.

is $158 \times 10^9$. Thus there is only around $3\%$ increase in FLOPs. In fact, we barely observe training and inference time difference between baseline and our model.

**Comparing to the Non-local Net.** One of the related work is the recent proposed Non-local Neural Networks [340], where they propose to perform non-local operations on different layers of feature maps for spatial-temporal reasoning. The comparisons between Non-local Nets and our approach is shown in Table 6.2b. We can see that the Non-local operations gives $1.7\%$ improvements over the baseline and our approach performs $2.7\%$ better than the Non-local Net. We also show that these two approaches are actually complementary to each other. By replacing the I3D backbone with Non-local Net, we have another $1.3\%$ boost, leading to $37.5\%$.

**Error analysis** Given this significant improvements, we will also like to find out in what cases our methods improve over the baselines most. Following the attributes set up in [272], we show 3 different situations where our approach get more significant gains over the baselines in Figure 6.5. More specifically, for each video in Charades, besides the action class labels, it is also labeled with different attributes (e.g., whether the actions are happening in a sequence? Is the pose variant a lot though the actions? Is the action involving objects?).

*Part of A Sequence?* This attribute specifies whether an action category is part of a sequence of actions. For example, "holding a cup" and then "sitting down" are usually in a sequence of actions, while "running" often happens in isolation. As shown in the left plots in Figure 6.5, the baseline I3D method fails dramatically when an action is part of a sequence of actions, while our approach is more stable. If an action is not happening in isolation, we have actually more than $5\%$ gain over the baseline.

*Pose Variances.* This attribute is computed by averaging the Procrustes distance [155] between any two poses in an action category. If the average distance is large, it means the poses change a lot in an action. As visualized in the middle plots in Figure 6.5, we can see that our approach has similar performance as the baseline when the pose variance is small. However, the performance of the baseline drops dramatically again as the variance of pose becomes larger (from $0.68$ to $0.73$) in the action, while the slope of our curve is much smaller.

| model | backbone | modality | mAP |
|---|---|---|---|
| 2-Stream [271] | VGG16 | RGB + flow | 18.6 |
| 2-Stream +LSTM [271] | VGG16 | RGB + flow | 17.8 |
| Asyn-TF [271] | VGG16 | RGB + flow | 22.4 |
| MultiScale TRN [390] | Inception | RGB | 25.2 |
| I3D [27] | Inception | RGB | 32.9 |
| I3D [340] | ResNet-101 | RGB | 35.5 |
| NL I3D [340] | ResNet-101 | RGB | 37.5 |
| NL I3D + GCN | ResNet-50 | RGB | 37.5 |
| I3D + GCN | ResNet-101 | RGB | 39.1 |
| NL I3D + GCN | ResNet-101 | RGB | **39.7** |

Table 6.3: Classification mAP (%) in the **Charades** dataset [273]. NL is short for Non-Local.

The performance of both approaches improve as the pose variability reaches $0.83$, where our approach has around $8\% \sim 9\%$ boost over the baseline.

*Involves Objects?* This attribute specifies whether an object is involved in the action. For example, "drinking from a cup" involves the object cup while "running" does not require interactions with objects. As shown in the right plots in Figure 6.5, we can see the baseline perform worse when the actions require interactions with objects. Interestingly, our approach actually performs slightly better when objects are involved.

As a short summary, our approach is better in modeling a long term sequence of actions and actions that require object interactions. Our approach is also more robust to pose changes and is able to utilize the motion from poses.

**Training with a larger backbone.** Besides the ResNet-50 backbone architecture, we also verify our method on a much larger backbone model which is applied in [340]. This backbone is larger than our baseline in 3 aspects: (i) instead of using ResNet-50, this backbone is based on the ResNet-101 architecture; (ii) instead of using $224 \times 224$ spatial inputs, this backbone takes in $288 \times 288$ images; (iii) instead of sampling 32 frames with 6fps, this backbone performs sampling more densely by using 128 frames with 24fps as inputs. Note that the temporal output dimension of both our baseline model and this ResNet-101 backbone are still the same ($16$ dimensions). With all the modifications on the backbone architecture, the FLOPs are 3 times as many as our ResNet-50 baseline model.

We show the results together with all the state-of-the-art methods in Table 6.3. The Non-local Net [340] with ResNet-101 backbone achieves the mAP of $37.5\%$. We can actually obtain the same performance with our method by using a much smaller ResNet-50 backbone (with around $1/3$ FLOPs). By applying our method with the ResNet-101 backbone, our method (I3D+GCN) can still give $3.6\%$ improvements and reaches $39.1\%$. This is another evidence showing that our method is modeling very different things from just increasing the spatial inputs and the depth of the ConvNets. By combining the non-local operation together with our approach, we obtain the final performance of $39.7\%$.

### 6.5.3 Experiments on Something-Something

In the Something-Something dataset, there are 86K training videos, around 12K validation videos and 11K testing videos. Each video has the duration ranging from 3 seconds to 6 seconds. The total number of classes is 174.

The data in the Something-Something dataset is very different from the Charades dataset.

| model | backbone | *val* top-1 | *val* top-5 | *test* top-1 |
|---|---|---|---|---|
| C3D [98] | C3D [306] | - | - | 27.2 |
| MultiScale TRN [390] | Inception | 34.4 | 63.2 | 33.6 |
| I3D | ResNet-50 | 41.6 | 72.2 | - |
| I3D + Spatial-Temporal GCN | ResNet-50 | 42.8 | 74.7 | - |
| I3D + Similarity GCN | ResNet-50 | 42.7 | 74.6 | - |
| I3D + Joint GCN | ResNet-50 | 43.3 | 75.1 | - |
| NL I3D | ResNet-50 | 44.4 | 76.0 | - |
| NL I3D + Joint GCN | ResNet-50 | 46.1 | 76.8 | 45.0 |

Table 6.4: Classification accuracy (%) in the **Something-Something** dataset [98]. NL is short for Non-Local.

In the Charades dataset, most of the actions are performed by agents in a cluttered indoor scenes. However, in the Something-Something dataset, all videos are object centric and there is usually only one or two hands interacting with the center objects. The background in the Something-Something dataset is also very clean in most cases.

We report our results in Table 6.4. The evaluations are performed on both validation set and testing set. The baseline I3D approach achieves $41.6\%$ in top-1 accuracy and $72.2\%$ in top-5 accuracy. By applying our method with the I3D backbone (I3D + Joint GCN), we achieve $1.7\%$ improvements in the top-1 accuracy. We observe that the improvement of top-1 accuracy here is not as huge as the gains we have in the Charades dataset. The reason is mainly because the videos are already well calibrated with objects in the center of the frames. But interestingly, we still have a relative larger boost $2.9\%$ on the top-5 metric compared to the top-1 metric. We have also studied the performance of each sub-graph. If we only use spatial-temporal graph we obtain $42.8\%$ top-1 accuracy. With only similarity graph, we obtain $42.7\%$ accuracy.

We have also combined our method with the Non-local Net. As shown in Table 6.4, the Non-local I3D method achieves $44.4\%$ in top-1 accuracy. By combining our approach with the Non-local Net, we achieve another $1.7\%$ gain in top-1 accuracy, which leads to the state-of-the-art results $46.1\%$. We also test our final model on the test set by submitting to the official website. By using a single RGB model, we achieve the best result $45.0\%$ in the leaderboard.

## 6.6 Discussion

We propose a novel graph based network to model the long range relationships in videos for action recognition. Large performance gain over state-of-the-art demonstrate the effectiveness of our framework. But more importantly, our error analysis shows how our model is doing better in capturing the object interactions, pose changes and actions in a sequence. This shows that our model has a large potential in not only video classification, but also variant tasks including detection and tracking in videos.

# Chapter 7

# Modeling Actions as Transformations

Besides recognizing the action by taking a video as a space-time cube. In this chapter, we propose an alternative representation to understand human action. Consider the "soccer kicking" action shown in Figure 7.1. What is the right representation for the recognition of such an action? Traditionally, most research in action recognition has focused on learning discriminative classifiers on hand-designed features such as HOG3D [165] and IDT [327]. Recently, with the success of deep learning approaches, the focus has moved from hand-designed features to building end-to-end learning systems. However, the basic philosophy remains the same: representing action implies encoding the appearance and motion of the actor. But are actions all about appearance and motion?

We argue that the true essence of an action lies in the change or the transformation an action brings to the environment and most often these changes can be encoded visually. For example, the essence of "soccer kicking" lies in the state change of the ball (acceleration) caused by the leg of the player. What if we try to represent actions based on these changes rather than appearance and motion?

In this chapter, we propose representing actions as transformations in the visual world. We argue that current action recognition approaches tend to overfit by focusing on scene context and hence do not generalize well. This is partly because of the lack of diversity in action recognition datasets compared to object recognition counterparts. In this chapter, we overcome this problem by forcing a representation to explicitly encode the change in the environment: the inherent reason that convinced the agent to perform the action. Specifically, each action is represented as a transformation that changes the state of the environment from what it was before the action to what it will be after it. Borrowing the terminology from NLP, we refer to the state before the action as the precondition state and the state after the action as the effect state, as Fig. 7.1 illustrates. We build a discriminative model of transformation by using a Siamese network architecture (similar to [18, 35]) where the action is represented as a linear transformation between the final fully connected layers of the two towers representing the precondition and effect states of an action.

Our experimental evaluations show that our representation is well suited for classical action recognition and gives state of the art results on standard action recognition dataset

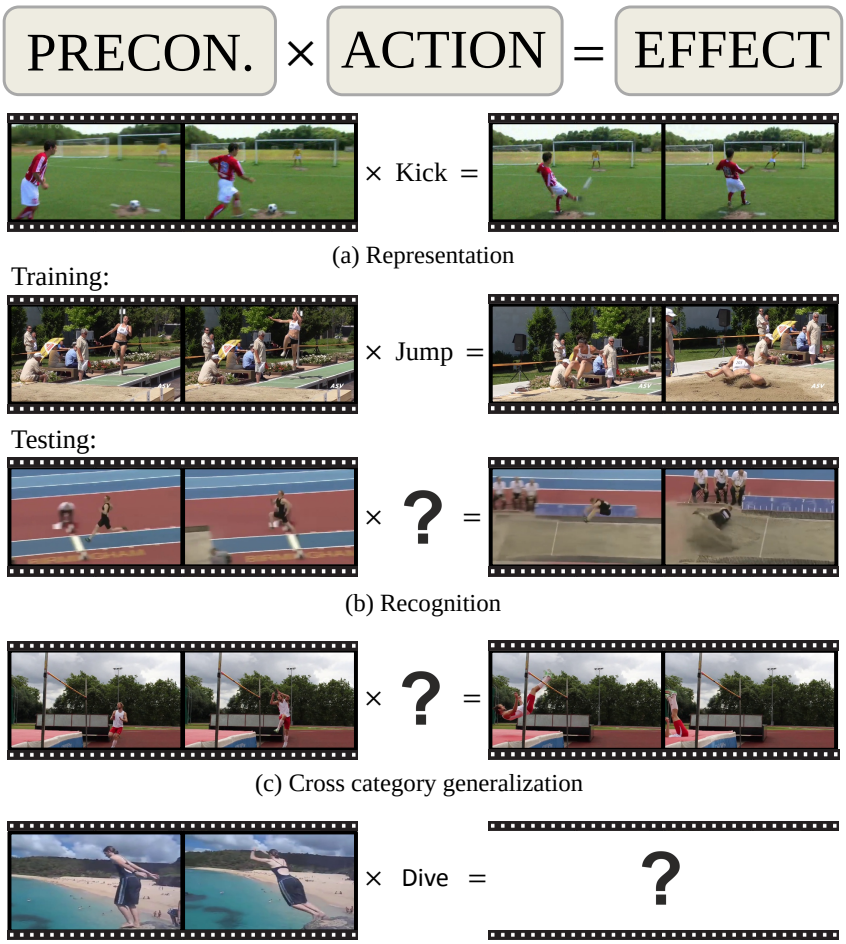



Figure 7.1: We represent actions as the transformations from precondition to effect. (a) For example, the precondition of kicking is the player running towards the ball and the effect is the ball flies away. By using this representation, we can (b) perform action recognition given the training data on long jump, and (c) generalize the classifier to high jump in testing. (d) Moreover, we can perform visual prediction given the precondition.

such as UCF101 [288]. However, in order to test the robustness of our representation, we also test our model for cross-category generalization. While overfitted representations would perform competitively on current action recognition datasets (due to lack of diversity), the true test lies in their ability to generalize beyond learned action categories. For example, how would a model learned on "opening a window" generalize to recognize "opening the trunk of the car"? How about generalizing from a model trained on climbing a cliff to recognize climbing a tree? Our experimental evaluations show that our representation allows successful transfer of models across action categories (Fig. 7.1 (c)). Finally, our transformation model can also be used to predict what is about to happen (Fig. 7.1 (d)).

Our contributions include: (a) a new representation of actions based on transformations in visual world; (b) state of the art performance on an existing action recognition dataset:

UCF101; (c) addressing the cross-category generalization task and proposing a new dataset, ACT, consisting of 43 categories of actions, which can be further grouped to 16 classes, and 11234 videos; (d) results on prediction task for our ACT dataset.

## 7.1 Background

Action recognition has been extensively studied in computer vision. Lack of space does not allow a comprehensive literature review (see [242] for a survey).

**Hand-crafted representations** have been conventionally used to describe patches centered at Space Time Interest Points (STIP) [180]. Most successful examples are 3D Histogram of Gradient (HOG3D) [165], Histogram of Optical Flow (HOF) [181], and Motion Boundary Histogram [41]. Mid- to high-level representation are also used to model complex actions [137,178,264,287,348,398]. More recently, trajectory based approaches [149,179,221,235,236, 325,327] have shown significant improvement in action recognition.

**Learned representations** with deep learning have recently produced state of the art results in action recognition [89,145,153,186,279,304,307,331,368]. Karpathy et al. [153] proposed to train Convolutional Neural Networks (ConvNets) for video classification on the Sports-1M dataset. To better capture motion information in video, Simonyan et al. [279] introduced a Two Stream framework to train two separate ConvNets for motion and color. Based on this work, Wang et al. [331] extracted deep feature and conducted trajectory constrained pooling to aggregate convolutional feature as video representations. In this chapter, we also train the networks taking RGB frames and optical flows as inputs.

**Temporal structure** of videos have also been shown effective in action recognition [63, 135,256,296,301]. For example, Tang et al. [301] proposed an HMM model to model the duration as well as the transitions of states in event video. Fernando et al. [63] learned ranking functions for each video and tried to capture video-wide temporal information for action recognition. Recurrent Neural Networks have also been used to encode temporal information for learning video representations [49, 229, 293, 297, 300, 364]. Srivastava et al. [293] proposed to learn video representations with LSTM in an unsupervised manner. Ng et al. [229] proposed to extract features with a Two Stream framework and perform LSTM fusion for action recognition. However, these HMM, RNN and recent LSTM approaches model a sequence of transformation across frames or key frames; whereas in our framework we model action as a transformation between precondition and effect of action. Note that the location of these frames are latent in our model.

The most similar work to ours is from Fathi et al. [61] where the change in the state of objects are modeled using hand-crafted features in ego-centric videos of 7 activities. We differ from [61] in that we learn representations which enable explicit encoding of actions as transformations. Our representations not only produce state of the art generic action recognition results, but also allow predictions of the outcome of actions as well as cross-category model generalization. To model the transformation, we apply a Siamese network architecture in this chapter, which is also related to the literature using deep metric learning [105,125,131,142,276,341].

## 7.2 Dataset

To study action recognition, several datasets have been compiled. Early datasets (e.g. Weizmann, KTH, Hollywood2, UCF Sports, UCF50) are too small for training ConvNets. Recently,

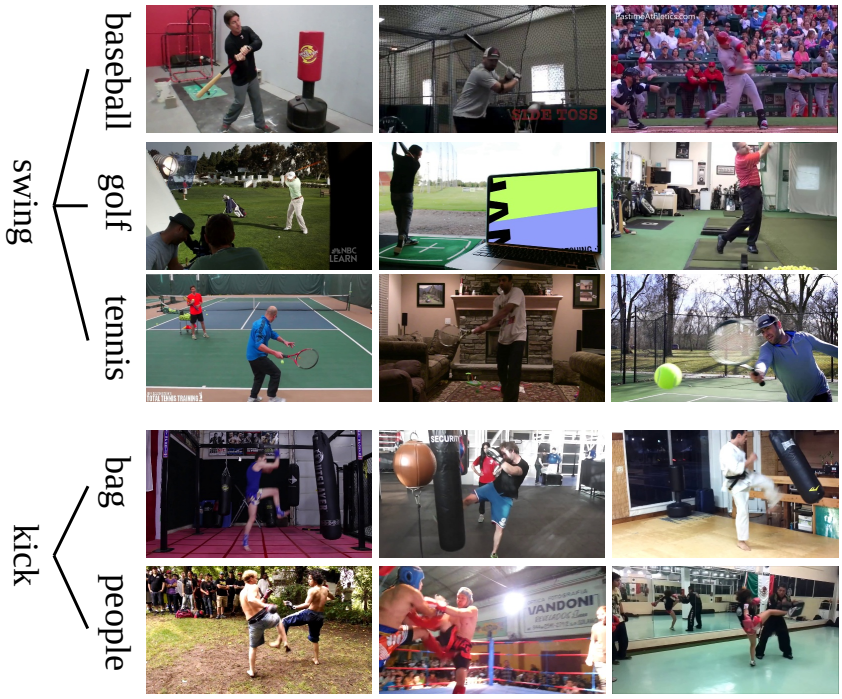



Figure 7.2: Samples in our ACT dataset. The action classes are arranged in a two-layer hierarchy. For each class, we collected hundreds of video clips with large diversities.

a few large-scale video datasets have been introduced (e.g. CCV [150], Sports-1M [153], ActivityNet [116], THUMOS [148] and FGA-240 datasets [297]). Unfortunately, some of these datasets have untrimmed videos without localization information for short term actions. The most commonly studied datasets are UCF101 [288] and HMDB51 [173]. The UCF101 dataset lacks the desired diversity among videos in each class. The HMDB51 dataset does not have enough videos compared to UCF101, and some of the videos are hard to recognize. But more importantly, none of these datasets is suitable for our task of examining cross category generalization of actions.

In this chapter, we argue for cross-category generalization as a litmus test for action recognition. We believe that the cross-category recognition task should not allow approaches to overfit to action classes based on contextual information. For this task, we propose a dataset, namely ACT dataset. In this dataset, we collected 11234 video clips with 43 classes. These 43 classes can be further grouped into 16 super-classes. For example, we have classes such as kicking bag and kicking people, they all belong to the super-class kicking; swinging baseball, swinging golf and swinging tennis can be grouped into swinging. Thus, the categories are arranged in a 2-layer hierarchy. The higher layer represents super-classes of actions such as kicking and swinging. Each super-class has different sub-categories which are the same action under different subjects, objects and scenes. During the dataset collection, we also ensured that we only consider high resolution and diverse videos.

**Dataset collection.** To collect our dataset we used YouTube videos. We used 50 keywords to retrieve videos which belong to one of the 43 classes. For each keyword, we downloaded

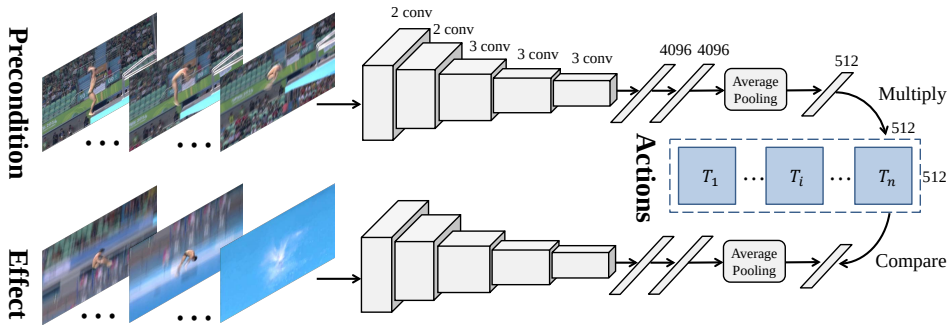



Figure 7.3: Siamese network architecture. Given a video, we feed the precondition state frames to the top network and effect state frames to the bottom network. Each tower of the ConvNet computes the feature for each frame independently, and aggregates the features via average pooling. The pooling results are fully connected to 512-D embedding outputs. We apply $n$ transformations (actions) on the precondition embedding and compare with the effect embedding to decide the action class.

around 500 high quality videos which have length within 15 minutes. The videos were labeled by a commercial crowd-sourcing organization. We asked the workers to label the starting and ending frames for actions in the video. For each action class, we provided a detailed description and 3 annotation examples from different videos. To increase the diversity of actions, we required the workers to label no more than 8 action clips in each video, and between each clips there should be a temporal gap of at least 40 frames. We also set temporal length limitations that each annotated clip should have at least 1 second and at most 10 seconds. As Figure 7.2 illustrates, our dataset has large intra-class diversities.

**Task design.** We design two tasks for our ACT dataset. The first task is standard action classification over 43 categories. The split used for this task included $65\%$ of videos as training and the rest as testing data, resulting in 7260 training videos and 3974 for testing in total. The second proposed task for this dataset is cross-category generalization. For each of the 16 super-classes, we consider one of its sub-category as testing and the other sub-categories are used for training. For example, for super-class "swinging", we want to see if the model trained on swinging baseball and swinging golf can recognize swinging tennis as "swinging". We create 3 different random splits for the second task. There are around 7000 training samples and 4000 testing samples on average. Our dataset can be downloaded from the project website[1].

## 7.3 Modeling Actions as Transformations

Given an input video $X$ consisting of $t$ frames, we denote each frame as $x_i$ and the whole video as $X = \{x_1, x_2, ..., x_t\}$. We make an assumption that the precondition state of an action corresponds to the first $z_p$ frames and the effect of the action can be seen after from $z_e$ until the end. We denote precondition and effect frames as: $X_p = \{x_1 \dots x_{z_p}\}$ and $X_e = \{x_{z_e} \dots x_t\}$. Note that we do not manually define how many frames are used of representing precondition and effect. Therefore $z_p$ and $z_e$ are treated as latent variables, which will be inferred automatically by our model during training and testing.

Instead of representing the precondition and effect by pixels and modeling the transfor-

[1] http://www.cs.cmu.edu/~xiaolonw/actioncvpr.html

mation in pixel space, we want to represent them using higher-level semantic features (e.g., the last fully connected layer of a ConvNet). The action then corresponds to transformation in higher-level feature subspace. This can be modeled via a Siamese ConvNet as shown in Figure 7.3. Given the video frames for the precondition and effect states, we feed the frames of precondition state $X_p$ as inputs for the ConvNet on the top and the frames of effect state $X_e$ are fed to the ConvNet on the bottom. For each tower of ConvNet, it computes the features for each frame independently, and then the features are aggregated via average pooling. We add a final $d$-dimensional fully connected layer after average pooling which represent the feature space where precondition, effect and the transformation between the two are modeled. Formally, we use $f_p(X_p)$ to represent the $d$-dimension embedding for the precondition state generated from the network on the top (in Figure 7.3) given input $X_p$. For the network on the bottom, we represent the embedding for the effect state as $f_e(X_e)$ with the same dimension $d$ given input $X_e$.

Finally, we model the action as the transformation between these two states. Specifically, we use a linear transformation matrix to model this. For a dataset with $n$ categories of actions, we have a set of $n$ corresponding transformation matrices $\{T_1, ..., T_n\}$ to represent them. Each $T_i$ is a $d \times d$ dimensions matrix. At the training time, given an input video $X$ that belongs to action category $i$, we first obtain the embedding for the precondition and effect states of the action as $f_p(X_p)$ and $f_e(X_e)$. We then apply the transformation matrix $T_i$ on the embedding of the precondition state as $T_i f_p(X_p)$, which is also a $d$-dimension vector. The objective of learning is making the distance $D(T_i f_p(X_p), f_e(X_e))$ between the two embeddings small. Note that while training, the gradients are back propagated throughout the Siamese networks; thus we learn both the embedding space and the transformation simultaneously.

**Network Architecture** We applied the VGG-16 network architecture [277] for both sides of our Siamese network. The VGG-16 network is a 16-layer ConvNet with 13 convolutional layers and 3 fully connected layers. As we mentioned before, we perform forward propagation for each video frame independently. We extract the feature of the second-to-last fully connected layer for each frame, which is a $4096$-D vector. In each side of our model, the features are aggregated via average pooling, and we use a final fully connected layer on these pooling outputs. The dimension of the last embedding layer outputs is $d = 512$. In our model, we do not share the model parameters between two ConvNets. Intuitively, we want to learn different semantic representations for precondition and effect of actions.

**Two Stream Siamese: RGB and Optical Flow as Inputs.** For each input frame, we rescale it by keeping the aspect ratio and make the smaller side $256$ pixels. To represent the video frames, we follow the same strategy as [279], which represents the frames with RGB images as well as optical flow fields. We train two separate models for RGB and optical flow fields as inputs. For the model using RGB images as inputs, the input size is $224 \times 224 \times 3$. For the model using optical flow as inputs, we represent each frame by stacking $10$ optical flow fields extracted from $10$ consecutive frames in the video starting from the current one. The optical flow field can be represented by a 2-channel image including horizontal and vertical flow directions. Therefore, the input size is $224 \times 224 \times 20$ as mentioned in [279].

**Implementation Details.** During training and testing, we re-sample the videos to be 25 frames in length ($t = 25$) as [279]. Note that the optical flow fields are still computed in the original video without re-sampling. We also constrain the latent variables $z_p$ and $z_e$, which are the indexes for the end frame of the precondition state and start frame of the effect state such that: $z_p \in [\frac{1}{3}t, \frac{1}{2}t)$ and $z_e \in (\frac{1}{2}t, \frac{2}{3}t]$.

## Training

We now introduce the training procedure for our model. Suppose we have a dataset of $N$ samples with $n$ categories $\{(X_i, y_i)\}_{i=1}^N$, where $y \in \{1, ..., n\}$ is the label. Our goal is to optimize the parameters for the ConvNets and transformation matrices $\{T_i\}_{i=1}^n$. We also need to calculate the latent variables $z_p$ and $z_e$ for each sample. Thus, we propose an EM-type algorithm, which performs a two-step procedure in each iteration: (i) learning model parameters and (ii) estimating latent variables .

**(i) Learning model parameters.** We first discuss the optimization of model parameters given the latent variables. Our objective is to minimize the distance between the embedding of the precondition state after transformation and the embedding of the effect state computed by the second ConvNet. This can be written as:

$$\min D(T_y f_p(X_p), f_e(X_e)), \tag{7.1}$$

where $T_y$ is the transformation matrix for the ground truth class $y$, $f_p(X_p)$ is the embedding for the precondition of the action and $f_e(X_e)$ is the embedding for the effect of the action. We use cosine distance here such that $D(v_1, v_2) = 1 - \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|}$ between any two vectors $v_1, v_2$.

To make our model discriminative, it is not enough to minimize the distance between two embeddings given the corresponding transformation $T_y$. We also need to maximize the distances for other incorrect transformations $T_i (i \neq y)$, which equals to minimize the negative of them. Thus, we have another term in the objective,

$$\min \sum_{i \neq y}^{n} max(0, M - D(T_i f_p(X_p), f_e(X_e))), \tag{7.2}$$

where $M$ is the margin so that we will not penalize the loss if the distance is already larger than $M$. In our experiment, we set $M = 0.5$. By combining Eq.(7.1) and Eq.(7.2), we have the final loss for training. It is a contrastive loss given sample $(X, y)$ and latent variables $z_p$ and $z_e$ as inputs. We train our model with back-propagation using this loss.

**(ii) Estimating latent variables.** Given the model parameters, we want to estimate the end frame index $z_p$ of the precondition state and start frame index $z_e$ of the effect state. That is, we want to estimate the latent variables to give a reasonable temporal segmentation for the input video. Since the latent variable only depends on the ground truth action category; we only use the first term in the loss function, Eq.(7.1), to estimate $z_p$ and $z_e$ as follows:

$$(z_p^*, z_e^*) = argmin_{(z_p, z_e)} D(T_y f_p(X_p), f_e(X_e)), \tag{7.3}$$

where $(z_p^*, z_e^*)$ are the estimation results given current model parameters. To estimate these latent variables, we use a brute force search through the space of latent variables. For computation efficiency, we first compute features for all frames, and then a brute force search only requires the average pooling step to be redone for each configuration of $(z_p, z_e)$

**Model pre-training.** We first initialize the ConvNets using ImageNet [261] pre-training and adapt them to action classification with fine-tuning as [332]. We transfer the convolutional parameters to both ConvNet towers and randomly initialize the fully connected layers in our model.

**Training detail discussions.** During training, we have not explicitly enforced the representations of $f_p(X_p)$ and $f_e(X_e)$ to be different. We have tried to use Softmax loss to

classify precondition and effect as two different classes. We found it does not help improve the performance. The reason is that the two towers of networks are learned on different data with different parameters (no sharing) and initialization, which automatically leads to different representations.

### Inference

During inference, given a video $X$ and our trained model, our goal is to infer its action class label $y$ and segment the action into the precondition and effect states at the same time. The inference objective can be represented as,

$$\min_{y,z_p,z_e} D(T_y f_p(X_p), f_e(X_e)). \tag{7.4}$$

More specifically, we first calculate the ConvNet feature before the average pooling layer for all the frames. Note that the first half of the frames are fed into the network for the precondition state and the second half of the frames are fed into the network for the effect state. We do a brute force search over the space of $(y, z_p, z_e)$ to estimate the action category and segmentation into precondition and effect. We visualize reasonable segmentation results for precondition and effect states during inference as Figure 7.4.

**Model fusion.** We perform the inference with the models using RGB images and optical flow as inputs separately. For each video, we have $n$ distance scores from each model. We fuse these two sets of scores by using weighted average. As suggested by [332] we weight the flow twice as much as the RGB ConvNet.

## 7.4 Experiment

In this section, we first introduce the details of the datasets and our experimental settings. Then we provide quantitative and qualitative results on different datasets.

**Datasets.** We evaluate our method on three datasets, including UCF101 [288], HMDB51 [173] and our ACT dataset. UCF101 dataset contains 13320 videos with 101 classes. The HMDB51 dataset is composed of 6766 videos with 51 action categories. For both datasets, we follow the standard evaluation criteria using 3 official training/testing splits. Our ACT dataset has 11234 video clips from 43 action classes and 16 super-classes. For this dataset, we conduct our experiments on two tasks. The first task is standard action classification of 43 categories. The second task is to test the generalization ability of our model. We conduct the experiments using 3 training/testing splits for 16-class classification. For each super-class, one sub-category is used for testing and the others for training.

**Implementation Details.** We first pre-train our networks using Softmax loss on action classification as the Two Stream baseline [332] and then transfer the convolutional parameters to our model. For HMDB51 dataset, as the number of training videos are relatively small (around 3.6K), we borrowed the ConvNet trained on UCF101 dataset with Softmax loss to initialize our model as [279, 331]. For the ACT dataset, we did not use UCF101 or HMDB51 during pre-training. Note that there are two types of inputs for our model: RGB and optical flow. We trained two different models for different inputs. For the model using RGB images as input, we set the initial learning rate as $10^{-5}$ and reduce it by 10 times after every $10K$ iterations and stop at $30K$ iterations. For the model using optical flow, the initial learning rate is set to $10^{-5}$. We reduce learning rate by order of 10 for every $20K$ iterations and stop

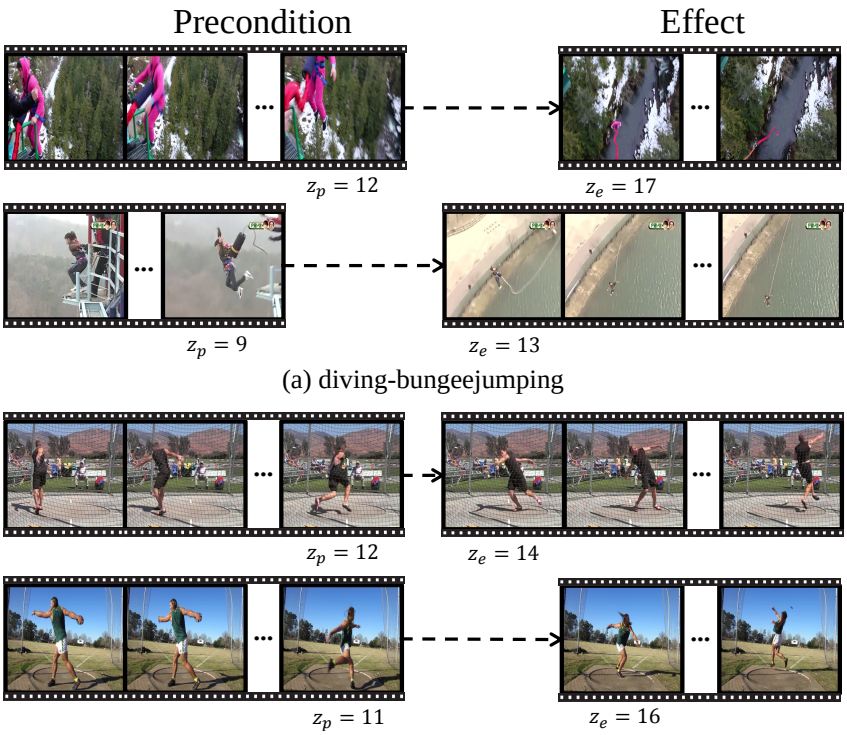



Figure 7.4: Temporal segmentation results after inference. $z_p$ and $z_e$ are the latent variables indexing the end frame for precondition and start frame for effect. Our model can provide reasonable temporal alignment of states between different videos in the same category.

| Method | RGB | Optical Flow | Fusion |
|---|---|---|---|
| Two Stream [279] | 73.0% | 83.7% | 88.0% |
| Two Stream (VGG16) [332] | 78.4% | 87.0% | 91.4% |
| Ours | **80.8%** | **87.8%** | **92.4%** |

Table 7.1: Average accuracies on UCF101 over 3 splits.

at $50K$ iterations. We set the size of the batch to $50$ videos. To compute the optical flow, we apply the TVL1 optical flow algorithm [382] and discretize the values to the range of $[0, 255]$.

### 7.4.1 Experimental Results.

**UCF101 dataset.** The accuracies over 3 training/testing splits of UCF101 are shown in Table 7.1. The Two Stream [279] method using an 8-layer CNNs achieves $88.0\%$. By replacing the base network with VGG-16 architecture, [332] obtains $91.4\%$ accuracy. We use [332] as a baseline for comparisons. Our model outperforms both models using RGB images and optical flow fields. After fusion of two models, we achieve state of the art performance at $92.4\%$. See Table 7.4 for more comparisons.

To further analyze our results and factor out the contributions of the appearance of

| Method (for split1) | RGB | Optical Flow | Fusion |
|---|---|---|---|
| Two Stream (VGG16) [332] | 79.8% | 85.7% | 90.9% |
| Two Stream First Half | 80.0% | 84.7% | 90.0% |
| Two Stream Second Half | 79.5% | 84.8% | |
| Ours | **81.9%** | **86.4%** | **92.0%** |

Table 7.2: Comparing method training different networks for different parts of the videos on UCF split1.

| Method | RGB | Optical Flow | Ave Fusion |
|---|---|---|---|
| Two Stream [279] | 40.5% | 54.6% | 58.0% |
| Two Stream (VGG16) | 42.2% | 55.0% | 58.5% |
| Ours | **44.1%** | **57.1%** | **62.0%** |

Table 7.3: Average accuracies on HMDB51 over 3 splits.

the precondition and effect segments, we perform an ablative analysis on the first split of UCF101. We train different networks using Softmax loss for different segments of the videos independently. For each video, we cut it by half and train one network using the first half of frames and another network is trained using the second half. In total we train 2 RGB ConvNets and 2 optical flow ConvNets. For model fusion, we average the results from these 4 networks. We show the results in Table 7.2. Compared to the baseline method [332], we find that the performance decreases most of the time and the fusion model is $0.9\%$ worse than the Two Stream baseline. This is mainly because decreasing the number of training samples leads to over-fitting. This analysis shows that modeling the precondition and effect separately does not work as well as the baseline, but modeling the transformations between them is $1.1\%$ better than the Two Stream baseline.

**HMDB51 dataset.** For HMDB51, we also report the average accuracies over 3 splits in Table 7.3. As a baseline, we apply Two Stream method [279,332] with VGG-16. The baseline method is better than [279] with average fusion. However, it is not as good as the best results ($59.4\%$) in [279] using SVM for model fusion. We show that our method has $1.9\%$ gain for models using RGB images and $2.1\%$ gain for model using optical flow. After model fusion, our method reach $62\%$ accuracy, which is $3.5\%$ better than the baseline.

More interestingly, we show that our method and the Two Stream method are complimentary to each other by combining the classification results from two methods. Since our outputs are distances and the Two Stream outputs are probabilities, we convert the distances outputs to probabilities. To do this, we extract the precondition embedding after ground truth transformation and the effect embedding on training data. We concatenate the two embeddings and train a Softmax classifier. In testing, we apply the same inference method as before and use the newly trained Softmax classifier to generate the probabilities, which gives the same performance as before. We average the outputs from two methods with equal weights as our final result. As Table 7.4 shows, by combining our method and the Two Stream method we have $1.4\%$ boost, which leads to $63.4\%$ accuracy. However, the state of the art results in this dataset are based on Improved Dense Trajectories (IDT) and Fisher vector encoding ($63.7\%$ [63] and $66.8\%$ [236]).

**ACT dataset.** We evaluate on two tasks in this dataset as proposed in the Dataset section. Motivated by the recent literature [49,229,364] in using LSTM to model the temporal information of videos, we also conduct experiments using the LSTM as another baseline for comparison. All the ConvNets are based on the VGG-16 architecture. For this LSTM

| HMDB51 | | UCF101 | |
|---|---|---|---|
| STIP+BoW [173] | 23.0% | STIP+BoW [288] | 43.9% |
| DCS+VLAD [139] | 52.1% | CNN [153] | 65.4% |
| VLAD Encoding [361] | 56.4% | IDT+FV [327] | 85.9% |
| IDT+FV [327] | 57.2% | IDT+HSV [235] | 87.9% |
| Two Stream [279] | 59.4% | Two Stream [279] | 88.0% |
| IDT+HSV [235] | 61.1% | LSTM with Two Stream [229] | 88.6% |
| TDD+FV [331] | 63.2% | TDD+FV [331] | 90.3% |
| Two Stream (VGG16) by us | 58.5% | Hybrid LSTM [364] | 91.3% |
| Ours | 62.0% | Two Stream (VGG16) [332] | 91.4% |
| Ours+Two Stream | **63.4%** | Ours | **92.4%** |

Table 7.4: Comparison to state of the art results.

| Method | RGB | Optical Flow | Fusion |
|---|---|---|---|
| Two Stream | 66.8% | 71.4% | 78.7% |
| LSTM+Two Stream | 68.7% | 72.1% | 78.6% |
| Ours | **69.5%** | **73.7%** | **80.6%** |

Table 7.5: Accuracies for the first task on ACT dataset.

approach, we first perform forward propagation with Two Stream model on each of the 25 frames and extract the feature from the fully connected layer before the classification outputs. The features are fed into LSTM to generate the classification scores. During training, we train the LSTM and Two Stream models jointly.

The first task is the standard action classification over 43 classes. As Table 7.5 illustrates, the LSTM method is better than the Two Stream method given RGB and optical flow inputs individually, however the results after fusion are very close. On the other hand, our method reach $80.6\%$, $1.9\%$ higher than the baseline.

For the second task, we examine the cross category generalization ability of our model. We perform 16-class classification on 3 different splits. For each class, one sub-category is left out for testing and the other sub-categories are used for training. As Table 7.6 shows, compared to the Two Stream baseline, we have $1.9\%$ improvements in models using RGB images as inputs and $3.7\%$ improvements using optical flow fields as inputs in average. After fusing the two models with different inputs, we have $65.5\%$ accuracy, $2.3\%$ higher than the baseline. From these results, we can see that modeling actions as transformations leads to better category generalization. To explain why our model works better, we perform several

| | RGB | | | | Optical Flow | | | | Fusion | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Split1 | Split2 | Split3 | Average | Split1 | Split2 | Split3 | Average | Split1 | Split2 | Split3 | Average |
| Two Stream | 48.3% | 53.2% | 49.7% | 50.4% | 53.7% | 56.6% | 56.3% | 55.5% | 59.5% | 67.6% | 62.2% | 63.2% |
| LSTM+Two Stream | 47.8% | 53.6% | 50.2% | 50.5% | 55.6% | 57.0% | 57.4% | 56.7% | 59.9% | 67.3% | 61.1% | 62.8% |
| Ours | 49.3% | 54.8% | 52.5% | **52.2%** | 57.6% | 59.5% | 60.4% | **59.2%** | 62.4% | 68.7% | 65.4% | **65.5%** |

Table 7.6: Accuracies over 3 splits for the second task on ACT dataset.


Figure 7.5: Nearest neighbor test. Given the query videos on the left, our method can retrieve semantically related videos in the middle while the Two Stream method retrieves videos with similar appearance and motion on the right.

visualizations in the following section.

### 7.4.2 Visualization

Just showing quantitative results does not give a full understanding of our method. What is our model learning? In which way our model is doing better? To answer these questions, we perform several visualization tasks on our ACT dataset.

**Nearest neighbor test.** In this experiment, we want to visualize in what cases our method is doing better via nearest neighbor on the features, which are extracted by the models trained in the second task of the ACT dataset. To extract the feature with our model, we perform inference on the video and concatenate the precondition embedding after transformation with the effect embedding. For the baseline Two Stream method, we extract the second-to-last fully connected layer feature and compute the average feature over the frames in the video. For both methods, we extract the feature for the RGB and optical flow models and then average them. As shown in Figure 7.5, given the query from testing samples on the left, we show the retrieved videos from the training videos. In the middle is our result and the retrieval results of the Two Stream method are on the right. For instance, on the first row, the query is a baby pushing a cart, we can retrieve the video in which a man is pushing a car, while the Two Stream method gets the baby crawling on the floor as the nearest neighbor. As the Two Stream method tries to match videos with similar appearance and motions, modeling the actions as transformations give us more semantically related retrieval results.

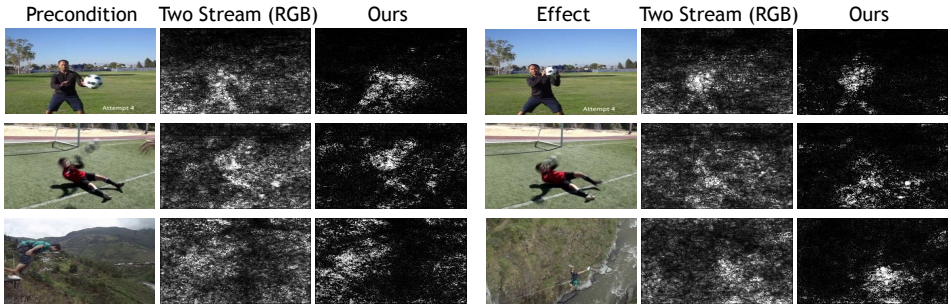



Figure 7.6: Given the precondition and effect frames as inputs, we visualize the magnitude of gradients via back-propagation. While the network trained with RGB images in Two Stream method are sensitive to the object and scenes, our model focus more on the changes.


Figure 7.7: Prediction results. Given the precondition frames of test videos on the left, we retrieve the effect frames of training samples. In the middle are the retrieval results with the same class as the query, on the right are results from different classes.

**Visualization of what the models focus on.** In this experiment, we visualize what the models focus on during inference using similar techniques as [76, 278]. We select one frame from the precondition state and another frame from the effect state, we feed these two RGB images into two towers of our model. We calculate the transformation loss given the video label and perform back-propagation. We visualize the magnitude of the gradients passed to the input image. This visualization shows which parts of the image force the network to decide on an action class. We visualize the RGB network from the Two Stream method (using Softmax loss) in the same way. As Figure 7.6 shows, given the precondition and effect frames, our model focuses more on the changes happening to the human while the RGB network from the Two Stream method performs classification based on the entire scene. In the first row, our model focus on the man and the ball when the ball is flying towards to him. As the man catches the ball, our model has high activations on the man with the ball. In the second row, our model can also capture locations where the ball is flying towards the man. In both cases, the direct classification model fires on the whole scene.

**Visual prediction.** Given the precondition embedding after transformation on the testing video, we retrieve the effect embedding from the same or different classes among the training data. We use the model trained on the first task of the experiment on the ACT dataset (43-class classification). If our model is actually learning the transformation, then it should be able to find effects frames that are similar to what we expect from applying the transformation to the precondition frames. We visualize the results in Figure 7.7. In each row, the first two

images are the beginning and ending frames of precondition segments of testing videos. The two images in the middle are the retrieval results with the same label as query. The two images on the right are the retrieval results with different class label as query. We show that our method can retrieve reasonable and smooth prediction results. For example, when our model is asked to predict what is going to happen when a man jumps from a diving board, it retrieves getting into the pool as the within-category result, and getting into the lake as the cross-category result. This is another information showing that our model is in fact learning actions as transformations from preconditions to effects.

## 7.5 Discussion

We propose a novel representation for action as the transformation from precondition to effect. We show promising action recognition results on UCF101, HMDB51 and ACT datasets. We also show that our model has better ability in cross category generalization. We show qualitative results which indicate that our method can explicitly model an action as the change or transformation it brings to the environment.

# Part III

# Interaction Reasoning with Correspondence

# Chapter 8

# Binge Watching for Affordance Learning

Going beyond recognition tasks in the previous chapters, one of the long-term goals of computer vision, as it integrates with robotics, is to translate perception into action. While vision tasks such as semantic or 3D understanding have seen remarkable improvements in performance, the task of translating perception into actions has not seen any major gains. For example, the state of the art approaches in predicting affordances still do not use any ConvNets with the exception of [71]. Why is that? What is common across the tasks affected by ConvNets is the availability of large scale supervisions. For example, in semantic tasks, the supervision comes from crowd-sourcing tools like Amazon Mechanical Turk; and in 3D tasks, supervision comes from structured light cameras such as the Kinect. But no such datasets exist for supervising actions afforded by a scene. Can we create a large-scale dataset that can alter the course in this field as well?

There are several possible ways to create a large-scale dataset for affordances: (a) First option is to label the data: given empty images of room, we can ask mechanical turkers to label what actions can be done at different locations. However, labeling images with affordances is extremely difficult and an unscalable solution. (b) The second option is to automatically generate data by doing actions themselves. One can either use robots and reinforcement learning to explore the world and the affordances. However, collecting large-scale diverse data is not yet scalable in this manner. (c) A third option is to use simulation: one such example is [71] where they use the block geometric model of the world to know where human skeletons would fit. However, this model only captures physically likely actions and does not capture the statistical probabilities behind every action. For example, it allows predictions such as humans can sit on top of stoves; and for the open space near doors it predicts walking as the top prediction (even though it should be reaching the door).

In this chapter, we propose another alternative: watch the humans doing the actions and use those to learn affordances of objects. But how do we find large-scale data to do that? We propose to binge-watch sitcoms to extract one of the largest affordance datasets ever. Specifically, we use every episode and every frame of seven sitcoms [1] which amounts to

[1] How I Met Your Mother, Friends, Two and a Half Men, Frasier, Seinfield, The Big Bang Theory, Everybody Loves Raymond

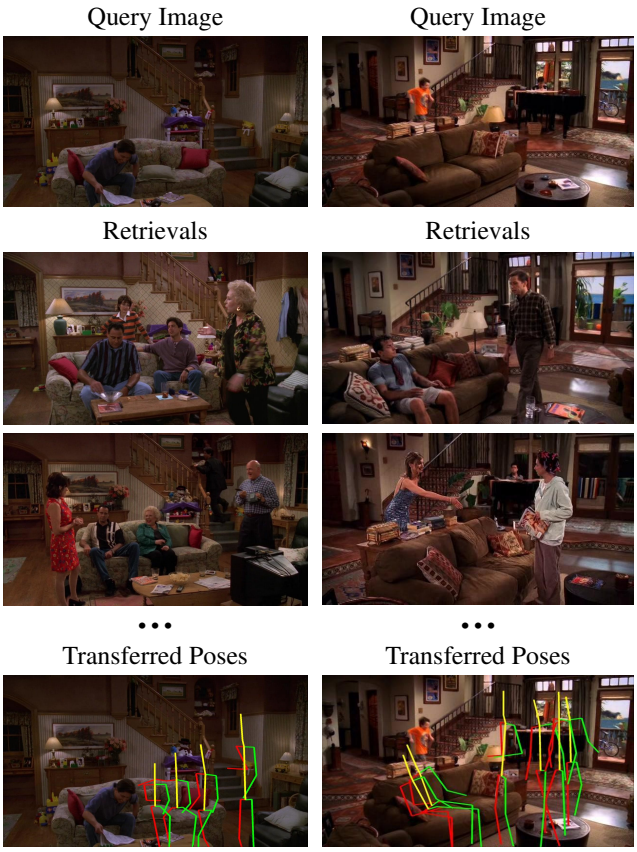



Figure 8.1: We propose to binge-watch sitcoms to extract one of the largest affordance datasets ever. We use more than 100M frames from seven different sitcoms to find empty scenes and same scene with humans. This allows us to create a large-scale dataset with scenes and their affordances.

processing more than 100 Million frames to extract parts of scenes with and without humans. We then perform automatic registration techniques followed by manual cleaning to transfer poses from scenes with humans to scenes without humans. This leads to a dataset of 28882 poses in empty scenes.

We then use this data to learn a mapping from scenes to affordances. Specifically, we propose a two-step approach. In the first step, given a location in the scene we classify which of the 30 pose classes (learned from training data) is the likely affordance pose. Given the pose class and the scene, we then use the Variational Autoencoder (VAE) to extract the scale and deformation of the pose. Instead of giving a single answer or averaging the deformations, VAE allows us to sample the distribution of possible poses at test time. We show that training an affordance model on large-scale dataset leads to a more generalizable and robust model.

## 8.1 Background

The idea of affordances [79] was proposed by James J. Gibson in late seventies, where he described affordances as "opportunities for interactions" provided by the environment. Inspired by Gibson's ideas, our field has time and again fiddled with the idea of functional recognition [253, 294]. In most cases, the common approach is to first estimate physical attributes and then reason about affordances. Specifically, manually-defined rules are used to reason about shape and geometry to predict affordances [294, 352]. However, over years, the idea of functional recognition took backstage because these approaches lacked the ability to learn from data and handle the noisy input images.

On the other hand, we have made substantial progress in the field of semantic image understanding. This is primarily due to the result of availability of large scale training datasets [43, 205] and high capacity models like ConvNets [172, 187]. However, the success of ConvNets has not resulted in significant gains for the field of functional recognition. Our hypothesis is that this is due to the lack of large scale training datasets for affordances. While it is easy to label objects and scenes, labeling affordances is still manually intensive.

There are two alternatives to overcome this problem. First is to estimate affordances by using reasoning on top of semantic [32, 38, 113] and 3D [8, 56, 80, 359] scene understanding. There has been a lot of recent work which follow this alternative: [101, 164] model relationship between semantic object classes and actions; Yao et al. [373] model relationships between object and poses. These relationships can be learned from videos [101], static images [373] or even time-lapse videos [42]. Recently, [402] proposed a way to reason about object affordances by combining object categories and attributes in a knowledge base manner. Apart from using semantics, 3D properties have also been used to estimate affordances [36, 67, 99, 102, 386]. Finally, there have been efforts to use specialized sensors such as Kinect to estimate geometry followed by estimating affordances as well [147, 166, 167, 400].

While the first alternative tries to estimate affordances in low-data regime, a second alternative is to collect data for affordances without asking humans to label each and every pixel. One possible way is to have robots themselves explore the world and collect data for how different objects can used. For example, [240] uses self-supervised learning to learn grasping affordances of objects or [3, 238] focus on learning pushing affordances. However, using robots for affordance supervision is still not a scalable solution since collecting this data requires a lot of effort. Another possibility is to use simulations [227]. For example, Fouhey et al. [71] propose a 3D-Human pose simulator which can collect large scale data using 3D pose fitting. But this data only captures physical notion of affordances and does not capture the statistical probabilities behind every action. In this work, we propose to collect one of the biggest affordance datasets using sitcoms and minimal human inputs. Our approach sifts through more than 100M frames to find high-quality scenes and corresponding human poses to learn affordance properties of objects.

## 8.2 Sitcom Affordance Dataset

Our first goal towards data-driven affordances is to collect a large scale dataset for affordances. What we need is an image dataset of scenes such as living rooms, bedrooms etc and what actions can be performed in different parts of the scene. In this chapter, inspired by some recent work [104], we represent the output space of affordances in terms of human poses.

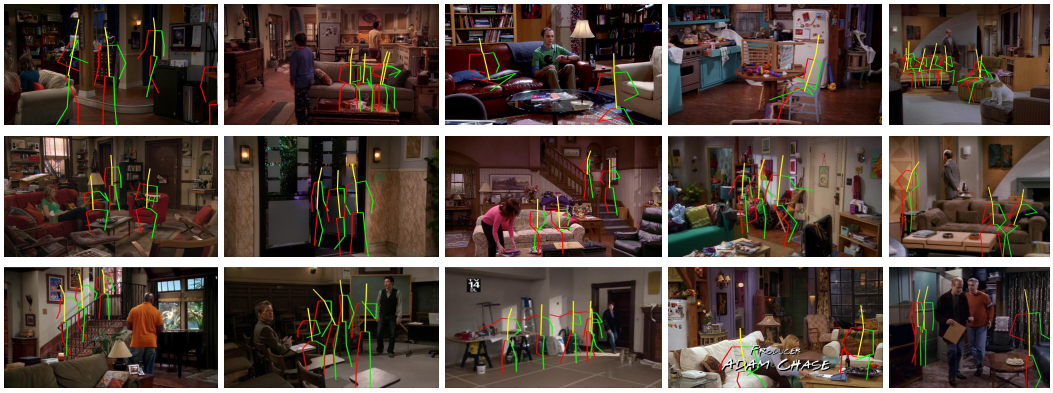


Figure 8.2: Some example images from Sitcom Affordance dataset. Note that our images are quite diverse and we have large samples of possible actions per image.

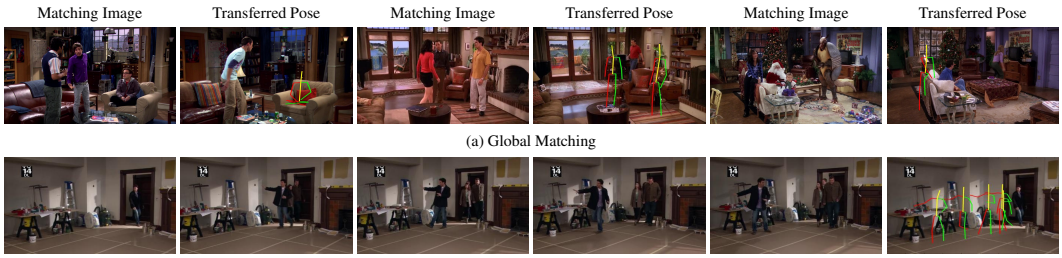
But where can we find images of the same scene with or without people in it?

The answer to the above question lies in exploiting the TV Sitcoms. In sitcoms, characters share a common environment, such as a home or workplace. A scene with exact configuration of objects appears again and again as multiple episodes are shot in it. For example, the living room in Friends appears in all the 10 seasons and 240 episodes and each actor labels the action space in the scene one by one as they perform different activities.

We use seven such sitcoms and process more than 100M frames of video to create the largest affordance dataset. We follow a three-step approach to create the dataset: (1) As a first step, we mine the 100M frames to find empty scenes or sub-scenes. We use an empty scene classifier in conjunction with face and person detector to find these scenes; (2) In the second step, we use the empty scenes to find the same scenes but with people performing actions. We use two strategies to search for frames with people performing actions and transfer the estimated poses [26] to empty scenes by simple alignment procedure; (3) In the final step, we perform manual filtering and cleaning to create the dataset. We now describe each of these steps in detail.

### Extracting Empty Scenes

We use a combination of three different models to extract empty scenes from 100M frames: face detection, human detection and scene classification scores. In our experiment, we find face detection [203] is the most reliable criteria. Thus, we first filter out scenes based on the size of the largest face detected in the scenes. We also applied Fast-RCNN to detect humans [82] in the scene. We reject the scenes where humans are detected. Finally, we have also trained a CNN classifier for empty scenes. The positive training data for this classifier are scenes in SUN-RGBD [286] and MIT67 [245]; the negative data are random video frames from the TV series and Images-of-Groups [75]. The classifier is finetuned on PlaceNet [389]. After training this classifier, we apply it back on the TV series training data and select 1000 samples with the highest prediction scores. We manually label these 1000 images and use them to fine-tuned the classifier again. This "hard negative" mining procedure turns out to be very effective and improve the generalization power of the CNN across all TV series.




(a) Global Matching

(b) Local Matching

Figure 8.3: We propose to use two approaches to transfer poses. In global matching approach, we match an empty scene to all the images in the sitcom. Sometimes the matches occur in different seasons. Given these matches, we transfer poses to the image. In the local matching approach, we use the next 5-10 sec of video to transfer poses via optical-flow registration scheme.

### People Watching: Finding Scenes with People

We use two search strategies to find scenes with people. Our first strategy is to use image retrieval where we use empty scenes as query images and all the frames in the TV-series as retrieval dataset. We use cosine distance on the pool5 features extracted by ImageNet pre-trained AlexNet. In our experiments, we find the pool5 features are robust to small changes of the image, such as the decorations and number of people in the room, while still be able to capture the spatial information. This allows us to directly transfer human skeletons from matching images to the query image. We show some examples of data generated using this approach in the top two rows of Fig. 8.3.

Besides global matching of frames across different episodes of TV shows, we also transfer human poses within local shots (short clips at most 10 seconds) of videos. Specifically, given one empty frame we look into the video frames ranging from 5 seconds before this frame to 5 seconds after it. We perform pose estimation on every frame. We then compute the camera motions of each frame with respect to the empty frame by accumulating the optical flows. Given these motion information, we can map the detected poses to the empty frame, as shown in the bottom two rows in Fig. 8.3.

### Manual Annotations

Our goal is to use the automated procedure above to generate valid hypothesis of possible poses in empty scenes. However, the alignment procedure is not perfect by any means. Thus, we also need human annotators to manually adjust pose joints by scaling and translating. In this way, the pose in the empty scene can be aligned with the human in the image where the pose is transferred from. In cases where the poses are not fitting with the scene, due to occlusions or incorrect matching, we remove such poses. The final dataset we obtain contains 28882 human poses inside 11449 indoor scenes. The detailed statistics of how many poses for each TV series are summarized in the right table.

| Source | #Datapoints |
|---|---|
| HIMYM | 3506 |
| TBBT | 3997 |
| Friends | 3872 |
| TAAHM | 3212 |
| ELR | 5210 |
| Frasier | 6018 |
| Seinfeld | 3067 |
| Total | 28882 |

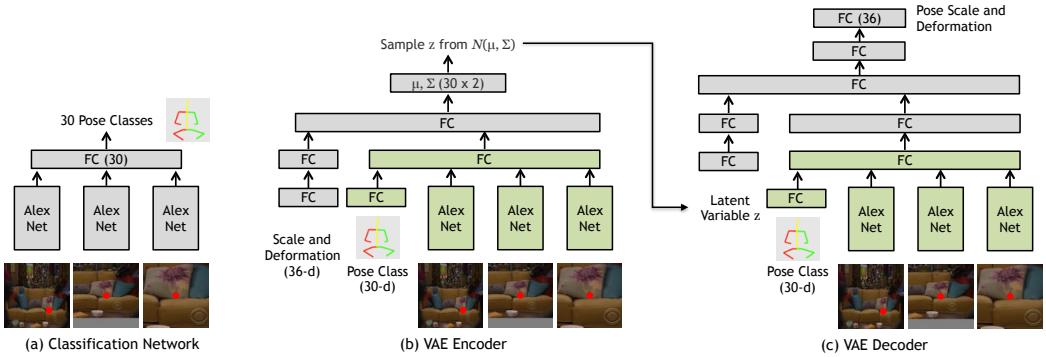



Figure 8.4: Our Affordance Prediction Model. The encoder and decoder in VAE share the weights which are highlighted as green. All fully connected layers have 512 neurons unless it is specified in the figure.

## 8.3 VAEs for Estimating Affordances

Given an indoor scene and the location, we want to predict what is the most likely human pose. One naive approach is training a ConvNet with the image and the location as input, predict the heat maps for each joint of the pose as state-of-the-art pose estimation approaches [26]. However, our problem is very different from standard pose estimation, since we do not have the actual human which can provide the pose structure and regularize the output.

We explore an alternative way: we decompose the process of predicting poses into two stages: (i) categorical prediction: we first cluster all the human poses in the dataset into 30 clusters, and predict which pose cluster is most likely given the location in the scene; (ii) given the pose cluster center, we predict its scale as well as the deformations for pose joints such that it fits into the real scene.

### 8.3.1 Pose Classification

As a first step, given an input image and a location, we first do a categorical prediction of human poses. But what are the right categories? We use a data-driven vocabulary in our case. Specifically, we use randomly sampled 10K poses from the training videos. We then compute the distances between each pair of poses using procrustes analysis over the 2D joint coordinates, and cluster them into 30 clusters using k-mediod clustering. We visualize the centers of the clusters as Fig. 8.5.

In the first stage of prediction, we train a ConvNet which uses the location and the scene as input and predict the likely pose class. Note that multiple pose classes could be reasonable in a particular location (e.g. one can stand before the chair or sit on the chair), thus we are not trying to regress the exact pose class. Instead we predict a probability distribution over all classes and select the most likely one. The selected pose center can be further adjusted to fit in the scene in the second stage.

**Technical Details:** The input to the ConvNet is the image and the location where to predict likely pose. To represent this point in the image, we crop a square patch using it as the center

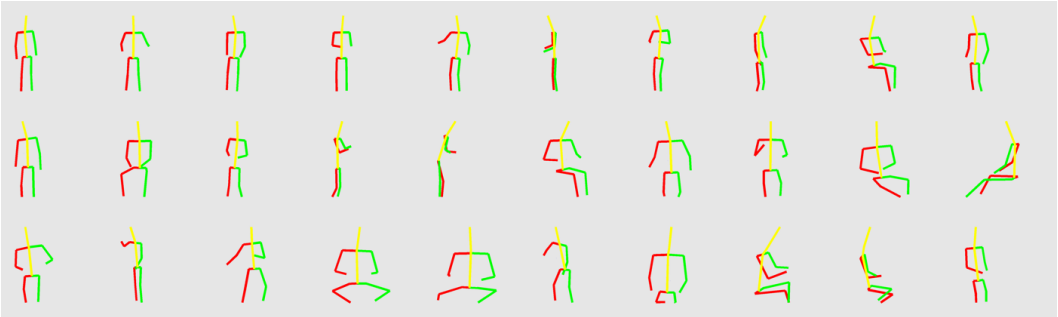


Figure 8.5: Cluster centers of human poses in sitcom dataset. These clusters are used as pose categories predicted by classification network.

and the side length is the height of the image frame. We also crop another patch in a similar way except that the length is half the height of the image. As illustrates in Fig. 8.4 (a), the red dots on the input images represent the location. The two cropped patches can offer different scales of information and we also take the whole image as input. The 3 input images are all re-scaled to $227 \times 227$.

Given the 3 input images, they are forwarded to 3 ConvNets which share the weights between them. We apply the AlexNet architecture [172] for the ConvNet here and concatenate the 3 $fc7$ outputs. The concatenated feature is further fully connected to 30 outputs, which represents 30 pose classes. During training, we apply SoftMax classification loss and the AlexNet is pre-trained with ImageNet dataset [44].

### 8.3.2 Scale and Deformation Estimation

Given the pose class and scene, we need to predict the scale and the deformations of each joint to fit the pose into the scene. However, the scale and deformations of the pose are not deterministic and there could be ambiguities. For example, given an empty floor and a standing pose class, it could be a child standing there (which corresponds to a smaller scale) or an adult standing there (which corresponds to a larger scale). Thus instead of directly training a ConvNet to regress the scale and deformations, we apply the conditional Variational Auto-Encoder (VAE) [161,321] to generate the scale and deformations conditioned on the input scene and pose class.

**Formulations for the conditional VAE.** We applied the conditional VAE to model the scale and deformations of the estimated pose class. For each sample, we define the deformations and scale as $y$, the conditioned input images and pose class as $x$, and the latent variables sampled from a distribution $Q$ as $z$. Then the standard variational equality can be represented as,

$$\begin{aligned} &\log P(y|x) - KL[Q(z|x,y)||P(z|x,y)] \\ &= E_{z\sim Q}[\log P(y|z,x)] - KL[Q(z|x,y)||P(z|x)], \end{aligned} \tag{8.1}$$

where $KL$ represents the KL-divergence between the distribution $Q(z|x,y)$ and the distribution $P(z|x)$. Note that in VAE, we assume $P(z|x)$ is a normal distribution $\mathcal{N}(0,1)$. The distribution $Q$ is another normal distribution which can be represented as $Q(z|x,y) =$

$Q(z|\mu(x,y),\sigma(x,y))$, where $\mu(x,y)$ and $\sigma(x,y)$ are estimated via the encoder in VAE. The log-likelihood $\log P(y|z,x)$ is modeled by the decoder in VAE. The details are explained as below.

**Encoder.** As Fig. 8.4(b) illustrates, the inputs of model include 3 images which are obtained in the same way as the classification network, a 30-d binary vector indicating the pose class (only one dimension is activated as 1), and a 36-d vector representing the ground truth scale and deformations. The images are fed into the AlexNet model and we extract the $fc7$ feature for each of them. The pose binary vector and the vector of scale and deformations are both forwarded though two fully connected layers. Each fully connected layer has 512 neurons. The outputs for each components are then concatenated together and fully connected to the outputs. The dimension of the outputs is $30 \times 2$ which are two vectors of mean $\mu(x,y)$ and variance $\sigma(x,y)$ of the distribution $Q$.

We calculate the ground truth scale for the height of pose $s_h$ as the actual pose height divided by the normalized height (ranging from 0 to 1) of cluster center. The ground truth scale for the width $s_w$ is calculated in a similar way. Given the ground truth $(s_h, s_w)$, we can re-scale the cluster center and aligned it to the input location. The deformation for each joint is the spatial distance $(dx, dy)$ between the scaled center and ground truth pose. Since we have 17 pose joints $(dX, dY) = (dx_1, dy_1, ..., dx_{17}, dy_{17})$, there are 34 outputs representing the deformations. Together with the scale $s_h, s_w$, the outputs of the generator are 36 real numbers.

**Decoder.** As Fig. 8.4(c) shows, the decoder has a similar architecture as the encoder. Instead of taking a vector of scale and deformations as input, we replace it with the latent variables $z$. The output of the network is changed to a 36-d vector of scale and deformations whose groundtruth is identical to the 36-d input vector of the encoder. Note that we share the feature representations for the conditional variables (images and classes) between the encoder and decoder.

**Training.** As indicated by Eq. 8.1, we have two losses during training the VAE model. To maximize the log-likelihood, we apply a Euclidean distance loss to minimize distance between the estimated scale and deformations $y^*$ and the ground truths as,

$$L_1 = ||y^* - y||^2. \tag{8.2}$$

And the other loss is to minimize the KL-divergence between the estimated distribution $Q$ and the normal distribution $\mathcal{N}(0,1)$ as,

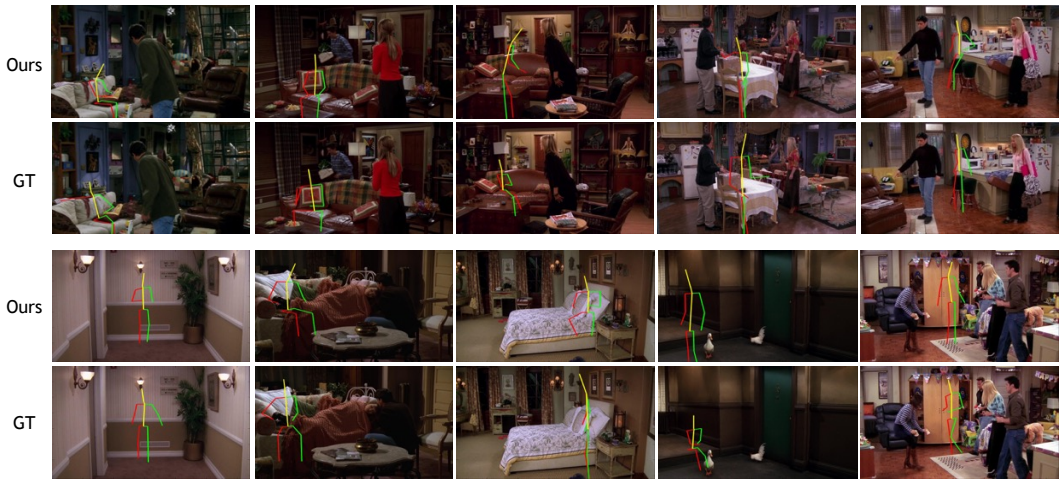
$$L_2 = KL[Q(z|\mu(x,y),\sigma(x,y))||\mathcal{N}(0,1)]. \tag{8.3}$$

Note that the first loss $L_1$ is applied on top of the decoder model, and its gradient is backpropagated though all the layers. To do this, we follow the reparameterization trick introduced in [159]: we represent the latent variables as $z = \mu(x,y) + \sigma(x,y) \cdot \alpha$, where $\alpha$ is a variable sampled from $\mathcal{N}(0,1)$. In this way, the latent variables $z$ is differentiable with respect to $\mu$ and $\sigma$.

### 8.3.3 Inference

Given the trained models, we want to tackle two tasks: (i) generating poses on an empty location of a scene and (ii) estimate if a pose fits the scene or not.

(a) Predicted Poses




(b) Comparing to Ground Truth

Figure 8.6: Qualitative results of generating poses: We show qualitative results on scenes from Friends dataset. (a) As we can see the human poses generated seem very reasonable. (b) We also compare the poses generated using our approach with the ground truth poses.

For the first task, given an image and a point representing the location in the image, we first perform pose classification and obtain the normalized center pose of the corresponding class. The classification scores, together with the latent variables $z$ sampled from $\mathcal{N}(0, 1)$ and images are forwarded to the VAE decoder model (Fig. 8.4 (c)). We scale the normalized center with the inferred scale $(s_h^*, s_w^*)$ and align the pose with the input point. Then we adjust each joint of the pose by adding the deformations $(dX^*, dY^*)$.

For the second task, we want to estimate whether a given pose fits the scene or not. To do this, we first perform the same estimation of the pose given an empty scene as the first task, then we compute the euclidean distance $D$ between the estimated pose and the given pose. To ensure the robustness of the estimation, we repeat this procedure by sampling different $z$ for $m = 10$ times, and calculate the average distance $\frac{1}{m}\sum_1^m D_i$ as the final result. If the final distance is less than a threshold $\delta$, then the given pose is taken as a reasonable pose.

## 8.4 Experiments and Results

We are going to evaluate our approach on two tasks: (i) affordance prediction: given an input image and a location, generate the likely human pose at that location; (ii) classify whether a given pose in a scene is possible or not.

We train our models using data collected from the TV series of "How I Met Your Mother", "The Big Bang Theory", "Two and A Half Man", "Everyone Loves Raymond", "Frasier" and "Seinfeld". The models are tested on the frames collected from "Friends". For training data, we have manually filtered and labeled 25010 accurate poses over 10009 different scenes. For testing data, we have collected 3872 accurate poses over 1490 different scenes and we have also artificially generated 9572 poses in the same scenes which are either physically impossible or very unlikely to happen in the real world.

During training, we initialize the AlexNet image feature extractor with ImageNet pre-training and the other layers are initialized randomly. We apply the Adam optimizer during training with learning rate $0.0002$ and momentum term $\beta_1 = 0.5, \beta_2 = 0.999$. To show that large scale of data matters, we perform the experiments on different size of the dataset.

We also evaluate the performance of our approach as the training dataset size increases. Specifically, we randomly sample 2.5K and 10K of data for training, and compare these models with our full model which uses 25K data for training.

**Baseline** We compare our VAE approach to a heatmap regression based baseline. Essentially, we represent the human skeletons as a 17-channel heatmap, one for each joint. We train a three-tower AlexNet (upto conv5) architecture, where each tower looks at a different scale of the image around the given point. The towers have shared parameters and are initialized with ImageNet. The outputs are concatenated across the towers and passed through a convolution and deconvolution layer to produce a 17 channel heatmap, which is trained with euclidean loss.

### 8.4.1 Generating poses in the scenes

As we mentioned in the approach, we generate the human pose via estimating the pose class and the scale as well as deformations.

**Qualitative results.** We show our prediction results as Fig. 8.6(a). We have shown that our model can generate very reasonable poses including sitting on a coach, closing the door and reaching to the table, etc. We also compare our results with the ground truth as Fig. 8.6(b). We show that we can generate reasonable results even though it can be very different from the ground truth poses. For example, in the 3rd column of the 2nd row, we predict a pose sitting on a bed while the ground truth is standing in front of the bed.

We have also visualized the results given different noise $z$ as inputs to the VAE Decoder during testing. For the same scene and location, we can generate different poses as Fig. 8.7.

**Quantitative results.** To show that the generated poses are in reasonable, we first evaluate the performance of our pose classification network. Note that there could be multiple reasonable poses in a given location, thus we show our 30-class classification accuracies given top 1 to top 5 guesses. We test our model on the 3872 samples from "Friends", the results is shown in Table 8.1. We compare models trained on three different sizes of our dataset (2.5K, 10K and 25K). We show that the more data we have, the higher accuracies

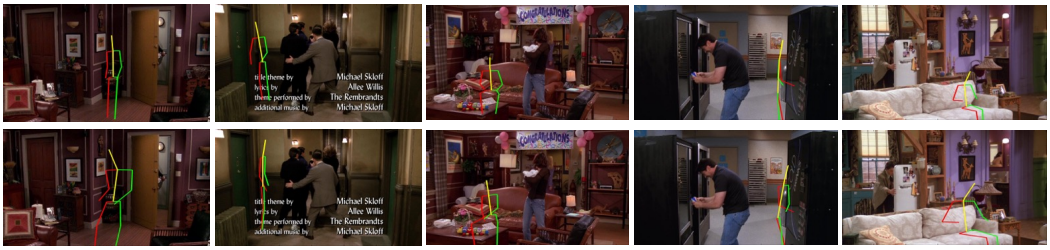


Figure 8.7: Affordance results for VAE with different noise inputs. Each column includes two predictions results for the same inputs. Given the human pose class, we show how VAE can be used to sample multiple scale and deformations.

Figure 8.8: Negative samples added in the test dataset. 71% of the test data is such images and 29% is positive examples.

we can get. For the heatmap baseline, we use the inner product of the predicted heatmap to assign the output to the cluster centers. We obtain the top-5 assignments and standard evaluation as above for classification performance. As the numbers show, our approach clearly outperforms this heatmap based baseline.

**Human evaluation.** We also perform human evaluation on our approach. Given the ground truth pose and predicted pose in the same location of the same image, we ask human which one is more realistic. We find that $46\%$ of the time the turkers think our prediction is more realistic. Note that a random guess is $50\%$, which means our prediction results are almost as real as the ground truth and the turkers can not tell which is generated by our model.

### 8.4.2 Classifying given poses in the scenes

Given a pose in a location of the scene, we want to estimate how likely the pose is using our model. We perform our experiments on 3872 positive samples from "Friends" and 9572 negative samples in the same scenes. We show some of the negative samples as Fig. 8.8. Note that although we use negative data in testing, but there is no negative data involved in training. We show the Precision-Recall curve as Fig. 8.9. Among all of our approaches, we find that training with 25K data points give best results, which is consistent with the first task. For the heatmap baseline, we again score each sample as the inner product of

| Method | Top-1 | Top-2 | Top-3 | Top-4 | Top-5 |
|---|---|---|---|---|---|
| HeatMap (Baseline) | 8.4 % | 19.9% | 30.1% | 39.1% | 47.3% |
| Training with 2.5K | 11.7% | 21.9% | 29.7% | 36.1% | 41.8% |
| Training with 10K | 13.3% | 23.7% | 32.3% | 39.7% | 46.8% |
| Training with 25K | **14.9%** | **26.0%** | **36.0%** | **43.6%** | **50.9%** |

Table 8.1: Classification results on the test set.

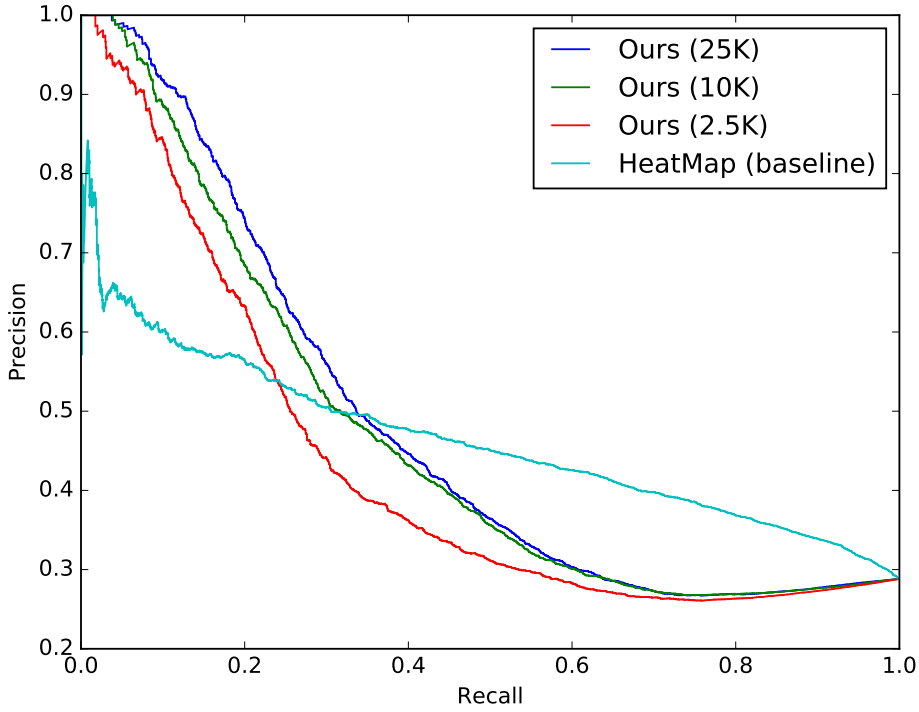



Figure 8.9: PR curve for our second experiment: given an image and pose, we produce a score on how probable it is. We use these scores to compute the recall-precision graph.

predicted heatmap with a heatmap representation of the labeled pose. We observe that the baseline does better than our approach in high-recall regimes, which can be explained by the fact that training with euclidean loss generates an averaged-out output, which is less likely to miss a pose.

## 8.5 Discussion

In this chapter, we present one of the biggest affordance dataset. We use 100 Million frames from seven sitcoms to extract diverse set of scenes and how actors interact with different objects in those scenes. Our dataset consist of more than 10K scenes and 28K ways humans can interact with these 10K images. We also propose a two step approach to predict affordance pose given an input image and the location. In the first step, we classify which of the 30 pose classes is the likely affordance pose. Given the pose class and the scene,we then use Variational Autoencoder (VAE) to extract the scale and deformation of the pose. VAE allows us to sample the distribution of possible poses at test time. Our results indicate that the poses generated by using our approach are quite realistic.

# Chapter 9

# Interpretable Intuitive Physics Model

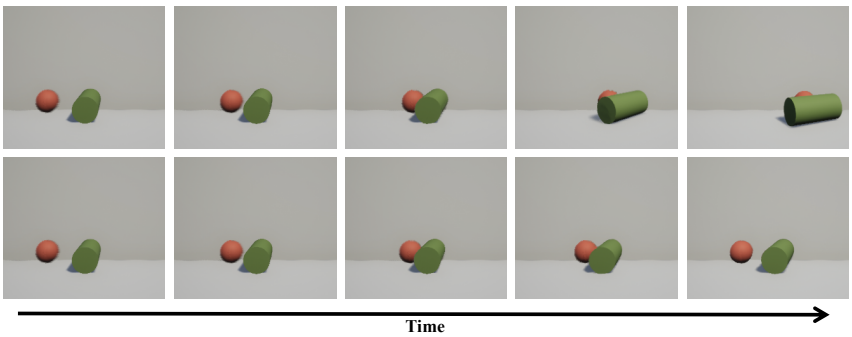



Figure 9.1: Interpretable Physics Models. Consider the sequences shown above. Not only we can predict the future frames of collisions but we can also predict the underlying factors that lead to such an inference. For example, we can infer the mass of cylinder is much higher in second sequence and therefore it hardly moves in the image. Our ability to infer meaningful underlying latent factors inspires us in this chapter to learn an interpretable intuitive physics model.

Besides studying on the interactions between human and the scene, in this chapter, we want to focus on the physical interactions between objects. Consider the collision image sequences shown in Figure 9.1. When people see these images, they not only recognize the shapes and color of objects but also predict what is going to happen. For example, in the first sequence people can predict that the cylinder is going to rotate while in the second sequence the ball will bounce with no motion on cylinder. But beyond visual prediction, we can even infer the underlying latent factors which can help us explain the difference in visual predictions. For example, a possible explanation of the behavior between the two sequences, if we knew the ball's mass didn't change, is that the first sequence's cylinder was lighter than the ball whereas in the second sequence the cylinder was heavier than the ball. Beyond this we can deduce that the cylinder in the first sequence was much lighter than the

one in the second.

Humans demonstrate the profound ability to understand the underlying physics of the world [108, 109] and use it to predict the future. We use this physical commonsense for not only rich understanding but also for physical interactions. The question arises as to whether this physical commonsense is just an end-to-end model with intermediate representations being a black-box, or explicit and meaningful intermediate representations? For humans, the answer appears to be the latter. We can predict the future if some underlying conditions are changed. For example, we can predict that if we throw the ball in the second sequence with 10x initial speed then the cylinder might rotate.

In this chapter, we focus on learning an intuitive model of physics [10, 192, 227]. Unlike some recent efforts, where the goal is to learn physics in an end-to-end manner with little-to-no constraints on intermediary layers, we focus on learning an **interpretable** model. More specifically, the bottleneck layers in our network model physical properties such as mass, friction, etc.

Learning an interpretable intuitive physics model is, however, quite a challenging task. For example, Wu et al. [358] attempts to build a model but the inverse graphics engine infers physical properties such as mass and friction. These properties are then used with neural physics engine or simulators for prediction. But can we really infer physical properties from the few frames of such collisions? Can we separate friction from mass, restitution by observing the frames? The fact is most of these physical factors are so dependent that it is infeasible to infer the exact values of physical properties. For example we can determine ratios between properties but not the precise values of both (e.g., we can determine the relative mass between two objects but not the exact values for both). This is precisely why in [358] only one factor is inferred from motion and the other factor is directly correlated to the appearance. Furthermore, the learned physics model is domain-specific and will not generalize–even across different shapes.

To tackle these challenges, we propose an interpretable intuitive physics model, where specific dimensions in the bottleneck layers correspond to different physical properties. The bottleneck layer models the distribution rather than infer precise values of mass, speed and friction. In order to demonstrate that our system models these underlying physical properties, we train our model on collision of different shapes (cube, cone, cylinder, spheres etc.) and test on collisions of unseen combinations of shapes altogether. We also demonstrate the richness of our model by predicting the future states under different physical conditions (*e.g.*, how the future frames will look if the friction is doubled).

Our contributions include: (a) an intuitive physics model that disentangles different physical properties in an interpretable way; (b) a staggered training algorithm designed to distinguish the subtleties between different physical quantities; (c) generalization to different shapes and physical quantity combinations; most importantly, (d) the ability to adapt future predictions when physical environments change. Note (d) is different from generalization: the hallucination/prediction is done for a physical scene completely different from the observed first four frames.

## 9.1 Background

Physical reasoning and learning physical commonsense has raised a lot of interest in recent years [3, 54, 226, 227, 237, 383, 387, 401]. There has been multiple efforts to learn implicit and

explicit models of physics commonsense. The underlying goal of most of these systems is to use physics to predict what is going to happen next [64, 72, 100, 192, 198, 357, 360]. The hope is that if the model can predict what is going to happen next after interacting with objects, it will be forced to understand the physical properties of the objects. For example, [192] tries to learn the physical properties by predicting whether a tower of blocks will fall. [72] proposed to learn a visual predictive model for playing billiards.

However, the first issue is what is the right data to learn this physics model. Researchers have tried a wide spectrum of approaches. For example, many researchers have focused on the task of visual prediction using real-world videos, based on the hypothesis that the predictive model will contain some underlying physical properties [220, 318, 321]. While videos provide realistic data, there is little to no control on how the data is collected and therefore the implicit models end up learning dynamic models of texture. In order to force physical commonsense learning, people have even tried using videos of physical interactions. For example, Physics101 dataset [357] collects sequences of collisions for this task. But most of the learning still happens passively (random batches). In order to overcome that, recent approaches have tried to learn physics by active interaction using robots [3, 64, 237]. While there is more control in the process of data collection, there are still issues with lack of diverse data due to most experiments being performed in lab setting with few objects. Finally, one can collect data with full control over several physical parameters using simulation. There has been lot of recent efforts in using simulation to learn physical models [72, 192, 226, 227]. One limitation of these approaches, in terms of data, is the lack of diversity during training, which forces them to learn physics models specific to particular shapes such as blocks, spheres etc. Furthermore, none of these approaches use the full power of simulation to generate a dense set of videos with multiple conditions. Most importantly, none of these approaches learn an interpretable model.

Apart from the question of data, another core issue is how explicit is the representation of physics in these models. To truly understand the object physical properties, it requires our model to be interpretable [10, 29, 174, 350, 358]. That is, the model should not only be able to predict the futures, but the latent representations should also indicate the physical properties (e.g., mass, friction and speed) implicitly or explicitly. For example, [10] proposed an Interaction Network which learns to predict the rigid body dynamics of gravitational systems. [358] proposed to explicitly estimate the physical object states and forward this state information to a physics engine for prediction. However, we argue exact values of these physical properties might not be possible due to entanglement of various factors. Instead of estimating the physics states explicitly, our work focuses on separating the dimensions in the bottleneck layer.

Our work is mostly related to the Inverse Graphics Network [174]. It learns a disentangled representation in the graphics code layer where different neurons are encouraged to represent different transformations including pose and light. The system can be trained in an end-to-end manner without providing an explicit state value as supervisions for the graphics code layer. However, unlike the Inverse Graphics Network, where pose and light can be separately inferred from the input images, the dynamics are dependent on the joint set of physical properties in our model (mass, friction and speed), which confound future predictions.

Our model is also related to the visual prediction models [163, 220, 243, 293, 321, 369, 396] in computer vision. For example, [293] proposed to directly predict a sequence of video frames in raw pixels given a sequence of former frames as inputs. Instead of directly predicting the pixels, [321] proposed to predict the optical flows given an input image and then warp the

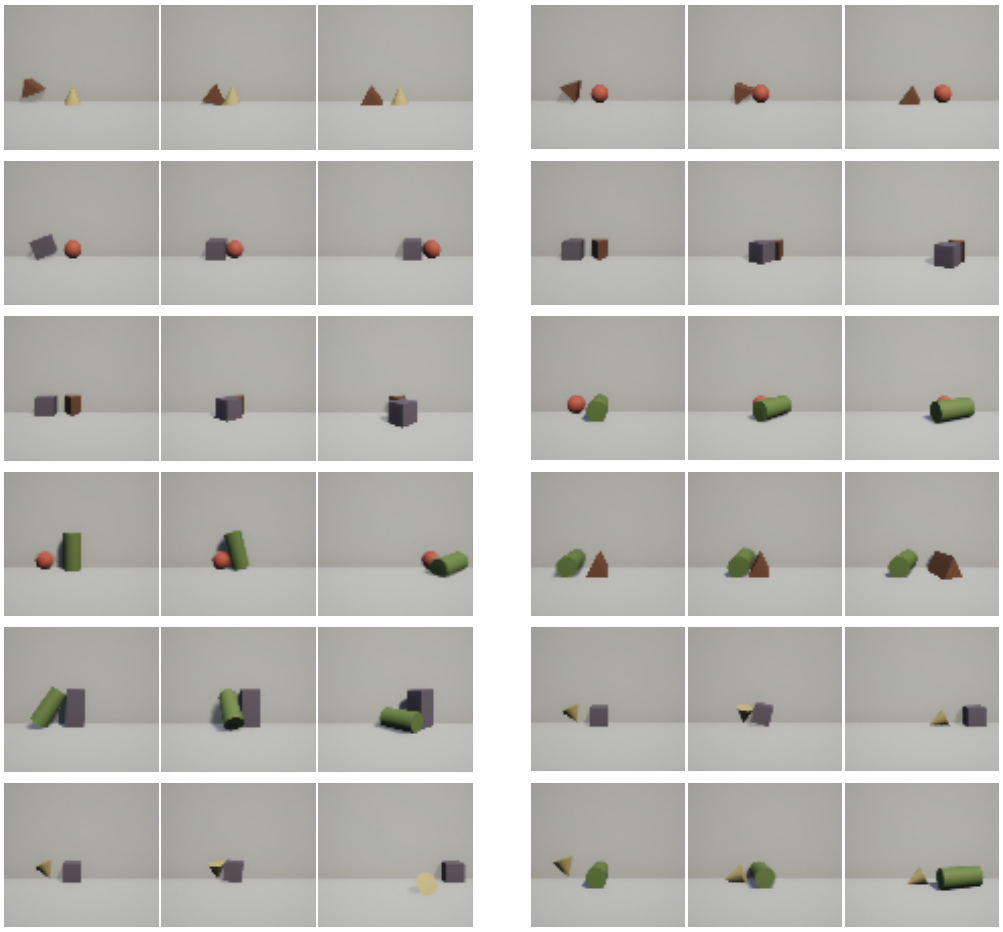


Figure 9.2: Our dataset includes 2 object collisions with a variety of shapes. Unlike existing physics datasets which have only one type of shape, our dataset is diverse in terms of different shapes and physical properties of objects.

flows on the input images to generate future frames. However, the optical flow estimation is not always correct, introducing errors in the supervisions for training. To tackle this, [396] proposed a bilinear sampling layer which makes the warping process differentiable. This enables them to train their prediction model from pixels to pixels in an end-to-end manner.

## 9.2 Dataset

We create a new dataset for our experiments in this chapter. The advantage of our proposed dataset is that we have rich combinations of different physical properties as well as different object appearances for different types of collisions (falling over, twisting, bouncing, etc.). Unlike previous datasets, the physical properties in our dataset are independent from the object shapes and appearance. In this way, we can train models which force estimation of

| | $scale_1$ | $scale_2$ | $scale_3$ | $scale_4$ | $scale_5$ |
|---|---|---|---|---|---|
| Mass | 100 | 200 | 300 | 400 | 500 |
| Speed | 10000 | 20000 | 30000 | 40000 | 50000 |
| Friction | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |

Table 9.1: Dataset Settings

physical properties by observing the collisions. More importantly, our testing sets contain novel combinations of object shapes or physical properties that are unseen in the training set. The details of dataset generation is illustrated as following.

We generate our data using the Unreal Engine 4 (UE4) game engine. We use 11 different object combinations with 5 unique basic objects: sphere, cube, cylinder, cone, and wedge. We select 3 different physical properties including mass of static object, initial speed of colliding object and friction of floor. For each property, we choose 5 different scales of values as shown in Table 9.1. For simplicity, we specify a certain scale of parameter by the format $\{parameter\ name\}_{\{scale\}}$ (e.g., $mass_1$, $friction_4$, $speed_2$). We simulate all the $5 \times 5 \times 5 = 125$ sets of physical combinations. For each set of physical property combination, there are 11 different object combinations and 15 different initial rotation and restitution. Thus in total there are $125 \times 15 \times 11 = 20625$ collisions. Each collision is represented by 5 sample frames with 0.5s time intervals between them.

The diversity in our dataset is highlighted in Figure 9.2. For example, our dataset has cones toppling over; cylinders falling down when hit by a ball and rolling cylinders. We believe this large diversity makes it one of the most challenging datasets to learn and disentangle physical properties.

For training, we use $124$ sets of physics combination with $9$ different object combinations ($16740$ collisions). The remaining data are used for two types of testing: (i) parameter testing and (ii) shape testing. The parameter testing set contains $135$ collisions with unseen physical parameter combinations ($mass_3$, $speed_3$, $friction_3$) but seen object shape combinations. The shape testing set on the other hand, contains $3750$ collisions with $2$ unseen shape combinations yet seen physical parameter combinations. We show the generalization ability of our physics model on both testing conditions.

## 9.3 Interpretable Physics Model

Our goal is to develop a physics-based reasoning network to solve prediction tasks, *e.g.*, physical collisions, while having interepretable intermediate representations.

### 9.3.1 Visual Prediction Model

As illustrated in Figure 9.3, our model takes in 4 RGB video frames as input and learns to predict the future 5th RGB frame after the collisions. The model is composed with two parts: an encoder for extracting abstract physical representations and a decoder for future frame prediction.

**Encoder for physics representations.** The encoder is designed to capture the motion of two colliding objects, from which the physical properties can be inferred. Given 4 RGB frames

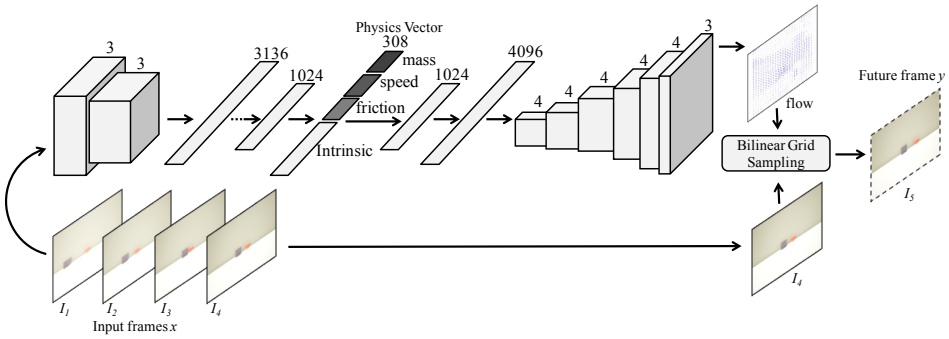



Figure 9.3: Model Architecture: we follow an encoder-decoder framework. The encoder takes 4 frames of a collision (2 before collision, 1 during collision, and 1 after collision). All inputs are first passed through a pre-trained Alexnet. The Alexnet features are further appended along channels and are sent to two convolution layers and four fully-connected layers. The resulting physics vector is passed through a decoder consisting of one fully-connected layer and six up-sampling convolution layers to produce an optical flow. The number on the convolution layers and transpose convolution layers stands for the kernel size of corresponding layer. The last bilinear grid sampling layer takes the optical flow and the $4^{th}$ input frame to produce future prediction.

as inputs, they are first forwarded to a ConvNet with AlexNet architecture and ImageNet pre-training. We extract the pool5 feature for each video frame and concatenate the features together as a representation for the input sequence. This feature is then forwarded to two convolutional layers and four fully connected layers to obtain the physics representation.

The physics representation is a 306 dimensional vector, which contains disentangled neurons of mass (dimensions 1 to 25), speed (dimensions 26 to 50), friction (dimensions 51 to 75), and other intrinsic information (dimensions 76 to 306), as shown in Figure 9.3. Note that although the vector is disentangled, there is no explicit meanings for each neuron value.

**Decoder for future prediction.** The physics representation is forwarded to a decoder for future frame prediction. Our decoder contains one fully-connected layer followed by six decovolutional layers. Inspired by [321, 396], our decoder uses optical flow fields as the output representation instead of directly outputing the RGB raw pixel values. The optical flow is then used to perform warping on the last input frame by a bilinear sampling layer [396] to generate the future frame. Since the bilinear sampling layer is differentiable, the network can be trained in an end-to-end manner with the 5th frame for direct supervision.

There are two major advantages of using optical flow as outputs: (i) it can force the model to learn the factors that cause the changes between two frames; (ii) it allows the model to focus on the changes of the foreground objects.

### 9.3.2 Learning Objective

Formally, we define the encoder as a function $f$ and the decoder as a function $g$. Given an image sequence $x$ as inputs (4 frames), our encoder transforms the images into a physically meaningful and disentangled representation $z = f(x)$ and then the decoder transforms this

representation into a future frame $y = g(z)$.

The disentangled representation $z$ can be formulated as $z = (\phi^m, \phi^s, \phi^f, \phi^i)$ where $(\cdot, \cdot)$ denotes concatenation. The first part $(\phi^m, \phi^s, \phi^f)$ denotes the combination *physics variable*, which encodes the physical quantities ($m$, $s$, $f$ stands for mass, speed, and friction respectively). The second part $\phi^i$ is the *intrinsic variable*, representing all the other intrinsic properties in the scene (*e.g.*, colors, shapes and initial rotation).

In this chapter, we study the effect of varying the values of physical quantities in a two-object collision scenario. Following the strategy in [174], we group our training sequence samples into mini-batches. Inside one mini-batch, only one physical property changes across all the samples and other physical properties remain fixed. We denote $B^p = \{(x_k, y_k)\}_{k=1}^5$ as one mini-batch with 5 sequences, where the only changing property is $p$ (i.e., we use $p$ as a variable to represent either mass, speed or friction).

For each mini-batch $B^p$ during training, we encourage only the dimensions corresponding to the property $p$ to change in $z$. For example, when training with a mini-batch where only mass is changing, we force the network to have different values in the dimensions for $\phi^m$ and same values for the rest of the dimensions in $z$. For simplicity, we further denote the dimensions which relevant to $p$ in $z$ as $\phi_k^p$ and the rest of the dimensions as $\bar{\phi}_k^p$ for example $k$.

We train our prediction model with this constraint. Assuming we are training with one batch $B^p = \{(x_k, y_k)\}_{k=1}^5$. In a maximum likelihood estimation (MLE) framework, this can be formulated as maximizing the log-probabilities under the desired constraints:

$$\begin{aligned} \text{maximize} \quad & \sum_{k=1}^{5} \log(\mathrm{P}(y_k|x_k)) \\ \text{subject to} \quad & \bar{\phi}_i^p = \bar{\phi}_j^p, \forall 1 \leq i, j \leq 5 \end{aligned} \tag{9.1}$$

where $\bar{\phi}_k^p$ contains both the intrinsic variable inferred from image sequence $x_k$ and inferred physics variables, except for the changing parameter.

In our auto-encoder architecture, the objective function is equivalent to minimizing the l1 distance between the predicted images $\hat{y}_k$ and the ground truth future images $y_k$:

$$\mathcal{L}_{mle} = \sum_k ||\hat{y}_k - y_k||_1. \tag{9.2}$$

The constraints in Eq. 9.1 can be satisfied via minimizing the loss between $\bar{\phi}_k^p$ and the mean of them within the mini-batch $\bar{\phi}^p = \frac{1}{5}\sum_k \bar{\phi}_k^p$ as,

$$\mathcal{L}_{ave} = \sum_k ||\bar{\phi}_k^p - \bar{\phi}^p||_2^2. \tag{9.3}$$

We apply both losses jointly during training our model with a constant $\lambda$ balancing between them as,

$$\mathcal{L} = L_{mle} + \lambda L_{ave}. \tag{9.4}$$

In practice, we set the $\lambda$ dynamically so that both gradients are maintained in the same magnitude. The value of $\lambda$ is around $1e-6$.

### 9.3.3 Staggered Training

Although we follow the training objective proposed in [174], it is actually non-trivial to directly optimize with this objective. There is a fundamental difference between our problem and the settings in [174]: the physical dynamics are dependent across the set of properties, which confounds training. The same sequence of inputs and output ground-truth might infer different combinations of the physical properties. For example, both large friction and slow speed can lead to small movements of the second object after collision. Thus modifications on training method is required to handle this multi-modality issue.

We propose a staggered training algorithm to alleviate this problem. We first divide the entire training set $D$ into 3 different sets $\{D^p\}$, where $p$ indicates one of the physics properties( mass, speed or friction). Each $D^p$ contains different mini-batches of $B^p$, inside which the only changing property is indicated by $p$.

The idea is: instead of training with all the physics properties at the same time in the beginning, we perform curriculum learning. We first train the network with one subset $D^p$ and then progressively add more subsets with different properties into training. In this way, our training set becomes larger and larger through time. By learning the physics properties in this sequential manner, we force the network to recognize new physical properties one by one while keeping the learned properties. In practice, we observe that in the first training session, the network behaves normally. For the following training sessions, the loss will increase in the beginning, and will decrease to roughly the same level as the previous session.

## 9.4 Experiments

We now demonstrate the effectiveness and generalization of our model. We will perform two sets of experiments with respect to two different testing sets in our dataset. One tests on unseen physical property combinations but seen shape combinations, and the other tests on unseen shape combinations with seen physical properties. Before going into further analysis, we will first describe the implementation details of our model and the baseline method.

**Implementation details** In total, we trained for 319 epochs. We used ADAM for optimization, with initial learning rate $10^{-6}$. During training, each mini-batch mentioned above has 5 sequences. During the training for the first physical quantity, each batch contains 3 mini-batches, which means 15 data in total. For the second round of staggered training, each batch contains 2 mini-batches, one for each physical quantity; similarly, in the third round of training, each batch contains 3 mini-batches, one for each physical quantity.

**Baseline model** Our baseline model learns intuitive physics in an end-to-end manner and post-hoc obtains the dimensions that correspond to different physical properties. We need the disentagled representation because we want to test the generalization when the physical properties are different from input video: *e.g.*, what happens if friction is doubled? What happens if the speed is 1/10th?

For the baseline, we use the same network architecture. Different from our approach, we do not add any constraints on the bottleneck representation layer as in Eq. 9.1 in the baseline model. However, we still want to obtain the disentangled representation from this baseline for comparison. Recall that we have a subset $D^p$ for each property $p$ (mass, friction or speed). The examples in each mini-batch inside $D^p$ specify the change of property $p$. We compute the variances for each neuron in the bottleneck representation for each $D^p$, and select 25

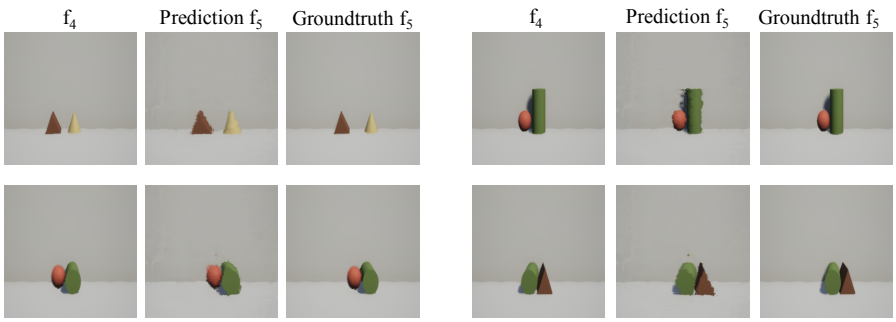



Figure 9.4: Prediction results for unseen parameters but seen shapes.

dimensions with top variances as the vector indicating property $p$.

### 9.4.1 Visual prediction

**Unseen Parameters:** First we evaluate if we can predict future pixels when we see a novel combination of physical parameters. Specifically, our model has never seen in training a combination of mass=3, friction=3 and speed=3. Figure 9.4 shows our interpretable model generalizes well and produces high quality predictions.

**Unseen Shape Combinations:** Next, we want to explore if our visual prediction model generalizes to different shape combinations using two unseen sets: (a) cone and cuboid; (b) cuboid and sphere. To demonstrate that our model understands each of these physical properties, we show contrasted prediction results for two different values. For example, we will use different friction values $(1, 5)$ but same mass and speed. Comparing these two outputs should highlight how our approach understands the underlying friction values.

As shown in Figure. 9.5, our predicted future frame has high quality compared to the ground-truth. We show that our model can generalize the physics reasoning to unseen objects and learn to output different collisions results given different physical environments. For example in the second condition, when the mass of sphere is high $(5)$, our approach can predict it will not move and instead the cube will bounce back. We also compare our approach to baseline quantitatively: our approach has pixel error of 87.3, while baseline has pixel error of 95.6.The results clearly indicate our interpretable model tends to generalize better than an end-to-end model when test conditions are very different.

In addition to the baseline, we also compare our model with two other methods based on optical flow. First, we trained another prediction network using the optical flow computed between the 4th and the 5th frame as direct supervisions, instead of using the pixels of the 5th frame. For testing, we apply the predicted optical flows on the 4th frame to generate the future frame. The loss between the future frame and the ground-truth 5th frame is 118.8. Second, we computed 3 optical flows of first 4 frames, using which to find a linear model to generate the future optical flow. We apply this optical flow on the 4th frame and compare the result to the ground-truth 5th frame. The error reaches to 292.5. The result shows that our method achieves high precision than using optical flow directly.

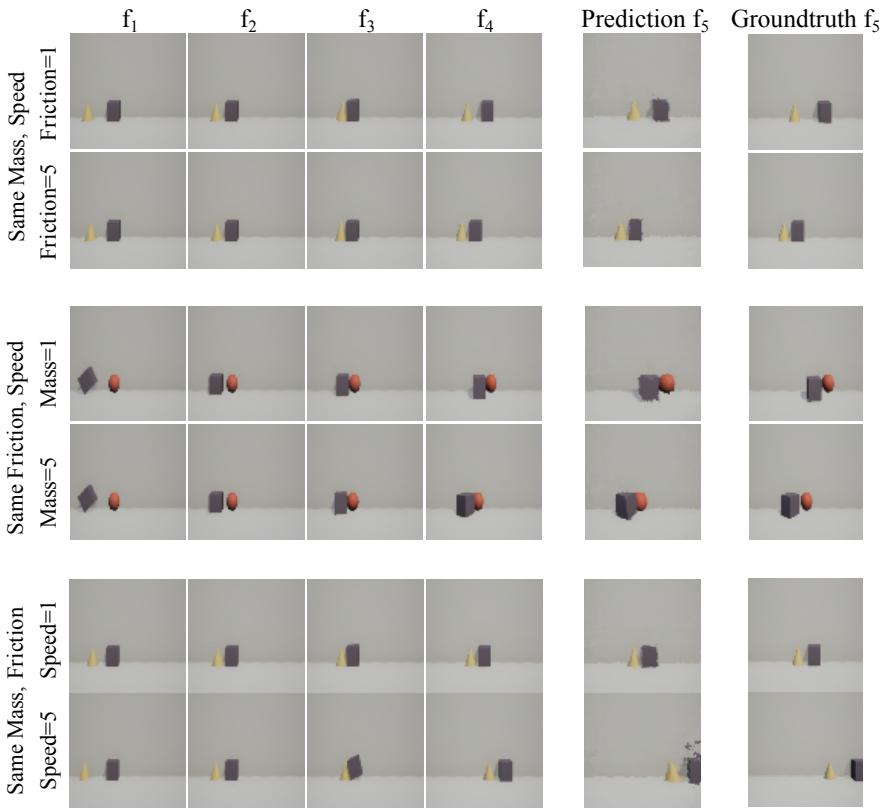



Figure 9.5: 4 input frames, the predicted 5th frame and ground-truth for collisions with unseen shape combinations. Contrast the predictions as one of physical property changes. For example, to show our approach understand these shapes, we predict for two different friction values in first case (keeping mass and speed same). The less motion in 2nd case shows that our approach understands the concept of friction.

### 9.4.2 Physical Interpolation

To show our model has actually learnt physics properties, we perform a series of interpolations on the bottleneck representation.

**Interpolating physics representation within a mini-batch.** We first show that the learned bottleneck layer is meaningful and smooth. To demonstrate this, we interpolate between different physical properties and compare our result with the ground-truth. The experiment is conducted in the following way. Let's take mass as an example: given a mini-batch where only mass changes, we use the encoder to get the physics vector $z_1 = (\phi_1^m, \phi_1^s, \phi_1^f, \phi_1^i)$ from $\text{mass}_1$ data and $z_5 = (\phi_5^m, \phi_5^s, \phi_5^f, \phi_5^i)$ from $\text{mass}_5$ data. To estimate the physics vector for $\text{mass}_i$, we interpolate a new mass variable $\hat{\phi}_i^m = (1 - 0.25i) \cdot \phi_1^m + 0.25i \cdot \phi_5^m$ and use this to create a new physics vector $\hat{z}_i = (\hat{\phi}_i^m, \phi_1^s, \phi_1^f, \phi_1^i)$. We pass the new vector to the decoder to predict the optical flows, which are warped to the 4th image in sequence $i$ via the bilinear sampling layer, and generate the future frame.

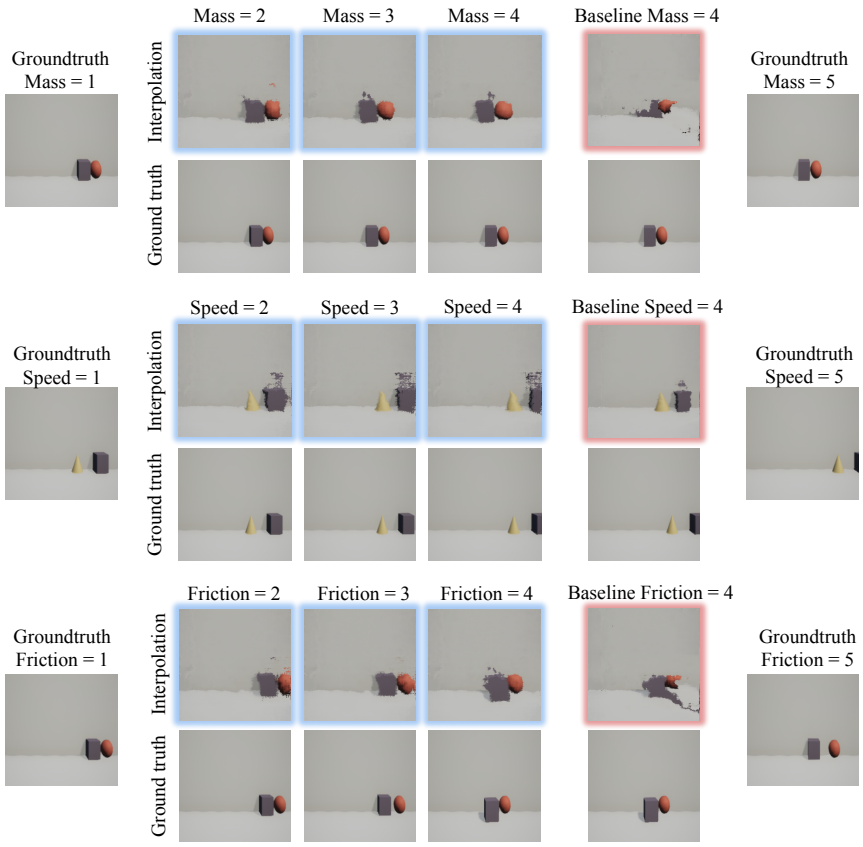



Figure 9.6: Interpolation results for physical quantity with different values. Our interpolation results are shown with blue frames. Images with red frame in last column represents the interpolation results for baseline when physical quantities equal to 4.

We perform the same set of experiments for the baseline model. Quantitatively, we evaluate the prediction using the sum of mean square error for each pixel, as shown in Table 9.2, which shows that our method is significantly better than the baseline. We also visualized the results in Figure 9.6. Interestingly, our interpolation results are also very close to the ground-truth. On the other hand, baseline models failed easily when there is a dramatic change during interpolations.

We also trained another model which takes physics parameters and the optical flows of first 4-frame as inputs, and predicts the future frame. This model performs much worse than our model in the interpolation test as shown in Figure 9.6. We believe a ground-truth physics parameter based approach focuses on classification instead of learning an intuitive physics model. In interpolation experiments, the model cannot separate physics information from the optical flow features.

From these comparison, we can see that only by learning interpretable representations, we can generate reasonable prediction results after interpolations.

**Changing physical properties.** In this experiment, we show that physics variables learned

| Method | shape 2 | shape 3 | shape 4 | shape 5 | parameter 3 |
|---|---|---|---|---|---|
| Baseline | 117.76 | 130.41 | 154.78 | 173.80 | 299.88 |
| Flow + Physics | 272.02 | 317.79 | 328.06 | 336.54 | 671.51 |
| Ours | **110.93** | **119.73** | **131.70** | **138.04** | **154.09** |

Table 9.2: Interpolation Result. The numbers are pixel prediction errors

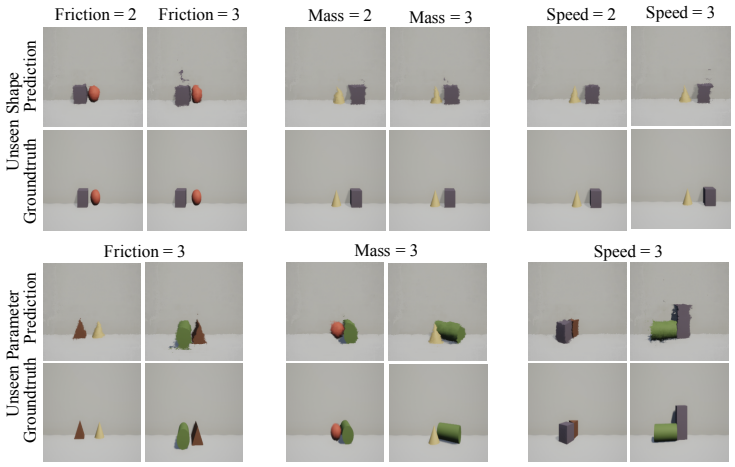



Figure 9.7: Prediction by learning double, triple ratio relation for different physical entities. Top: the result with unseen shapes. Bottom: result with unseen parameters.

by our model are interpretable by finding a mapping between different scale of the same physical property. Specifically, we want to see: can we predict the future if the mass is doubled while all other physics conditions remain the same? For each physical quantity $p$, we train two networks $F_2^p$ and $F_3^p$ which learns to double or triple the scale of a physical property. For example, we can project the physics representation of $\text{mass}_1$ to $\text{mass}_3$ by using the network $F_3^p$. The network architecture for both $F_2^p$ and $F_3^p$ is a simple 2-layer fully connected network with 256 hidden neurons per layer. These two networks can be trained using the physical representations inferred by our encoder with the training data.

In testing time, we apply the similar interpolation as the last experiment. The only difference is that instead of using an interpolation between two relevant representations, we use the fully connected network to generate the new representations. We again evaluate the quantitative results by computing the mean square error over the pixels. As shown in Table 9.3, we have a larger performance gain in this setting compared to the baseline. Figure 9.7 shows the prediction results of our model when the physics property is enlarged from scale 1 to 2 and 3, which are all very close to the ground-truth. This is another evidence showing our physics representation is interpretable and generalizes significantly better.

**Switching between the object shapes.** In experiments above, we interpolate the physics representation and apply them to the same object shape combinations. In this experiment, for a physical property $p$, we replace the corresponding variable $\phi^p$ of one collision with the variable from another collision with different objects but the same $p$ value. We visualize the results in Figure 9.8, where the first line shows the predictions when we replace current

| Method | shape ratio 2 (↓) | shape ratio 3 (↓) | parameter ratio 3 (↓) |
|---|---|---|---|
| Baseline | 345.60 | 310.37 | 490.92 |
| Ours | **110.79** | **124.00** | **157.10** |

Table 9.3: Ratio Result. Comparing visual prediction when underlying physical parameters are changed by a factor

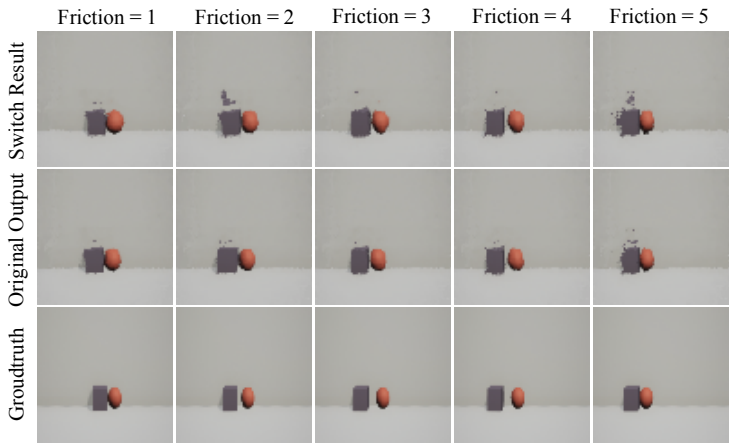



Figure 9.8: Prediction when physical property vector from one shape combination is applied to a different shape combinations. The first row shows switched result; the second row shows the prediction without switching; the third row shows ground-truth.

$\phi^p$ with one from another shape combination. The results are almost same as the original prediction and the ground-truth, which means that the physical variable of same value can be transferred among different shape combinations. It also shows that the dimensions of physics and other dimensions are independent and can be appended easily.

## 9.5 Discussion

In this chapter, we demonstrated an interpretable intuitive physics model that generalizes across scenes with different underlying properties and object shapes. Most importantly, our model is able to predict the future when physical environment changes. To achieve this we proposed a model where specific dimensions in the bottleneck layers correspond to different physical properties. However, often physical properties are dependent and intertangled, so we introduced a training curriculum and generalized loss function that was shown to outperform the baseline approaches.

# Chapter 10

# Discussion and Conclusion

In this thesis, we explored the spatio-temporal structure in videos for learning visual representations. We focused on the correspondence in time and showed its important role in different research applications. However, much work remains to be done and we believe it would be beneficial to summarize the observations, lessons and point out the potential future directions below.

**Self-supervised learning.** Going beyond the limits of semantic supervision, I believe self-supervised learning has a great potential to produce richer representation and allow for learning at a much larger scale. I see two possible goals for self-supervised learning.

- Obtain a *universal representation*. We can utilize multiple sources of structure information in data as signals to learn a *universal representation*, which can be generalized to every task. One initial attempt is mentioned in Chapter 4 in this thesis. Another potential way is to combine different signals from low levels (e.g., motion and boundary) to high levels (e.g., physical and functional properties) with curriculum learning. Intuitively, the high-level structure would be easier to learn given the emergence of low-level representation.

- Obtain *task specialized representations*. The goal is to learn representation for tasks where human annotations are hard to come by. There are already encouraging results in this direction (e.g., our work on finding dense correspondence in Chapter 2). However, the applications are still restricted to a few areas. One potential future direction is to explore task-specific, self-supervised learning on a large range of visual problems. Across these tasks, we believe there will be a shared principle in algorithm design.

**Video representation learning.** While we have shown some encouraging developments in video understanding in Chapter 5, Chapter 6 and Chapter 7 in this thesis, we have also observed two problems in the field: (i) first, the supervision from classification is not necessarily correct in capturing temporal information–for example, to classify the action "swimming," the model does not need to capture the motion but recognize the water; (ii) second, the scale of training examples is much smaller than image datasets, due to the cost of labels. These problems require us to search for other supervisions. One future direction is looking for training signals for learning spatial-temporal representation in a self-supervised

manner, which has been rarely explored so far. The other direction is to re-define the problem itself. We can design a dataset which requires the model to reason about the cause and effect of the action through time to solve the task. One possible approach is to disentangle the appearance from the target tasks in the dataset. In this way, the network will be forced to learn better abstraction beyond semantics.

**Active interaction.** One goal of this thesis is to make AI system to have a deeper understanding of how objects can be used and the functionality of scenes and how humans interact with them. As a first step, I have studied on scene affordance (Chapter 8) and object physical properties (Chapter 9) by learning from videos passively.

However, the process of acquiring this common sense knowledge should not only be in a passive manner. One future direction is to create an agent to move around and perform interactions actively. By reasoning with the interactions, new knowledge can be absorbed. Three potential future goals in active interaction would be: (i) First, when the agents are interacting with the environment, they should not only rely on semantic information of the scene but also on richer knowledge including affordances and physical properties of the objects. (ii) After interacting with the environment and observing the changes, we will allow the agent to update its knowledge base. Potentially, the updated knowledge base can then offer better guidance for future interactions. We can integrate the knowledge updates and the evolution of actions into an iterative learning procedure. (iii) Most importantly, the ultimate goal is to make this agent work in the real world beyond simulations.

**Summary.** This thesis is not the end, but rather the starting steps of my aim for research: To build an AI system that can scale up its learning ability beyond human supervision, acquire common sense knowledge, and interact with the world using the knowledge it has learned.

# Bibliography


[1] Recurrent Tracking using Multifold Consistency. *CVPR Workshop on PETS*, 2007. 9

[2] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015. 8, 34

[3] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Neural Information Processing Systems (NIPS)*, 2016. 93, 104, 105

[4] J.-B. Alayrac, J. Sivic, I. Laptev, and S. Lacoste-Julien. Joint discovery of object states and manipulation actions. In *ICCV*, 2017. 64

[5] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, 2008. 9

[6] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 69

[7] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. Recycle-gan: Unsupervised video retargeting. In *ECCV*, 2018. 9

[8] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, 2016. 93

[9] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *Proceedings of SIGGRAPH, ACM Transactions on Graphics*, 2009. 48

[10] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Neural Information Processing Systems (NIPS)*, 2016. 49, 64, 104, 105

[11] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV*, 2006. 23

[12] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV*, 2006. 36

[13] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *arXiv*, 2015. 44

[14] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *TPAMI*, 35(8):1798–1828, 2013. 21

[15] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2007. 19, 21, 34

[16] Y. Bian, C. Gan, X. Liu, F. Li, X. Long, Y. Li, H. Qi, J. Zhou, S. Wen, and Y. Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv:1708.03805*, 2017. 58, 59, 64

[17] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011. 64

[18] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a âĂIJsiameseâĂİ time delay neural network. *NIPS*, 1993. 76

[19] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *CVPR*, 2009. 9

[20] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 9

[21] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition (CVPR)*, 2005. 47, 48, 49, 50, 51

[22] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *Computer Vision and Pattern Recognition (CVPR)*, 2012. 48

[23] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of their mechanisms. *arXiv:1211.1552*, 2012. 48

[24] S. Caelles, K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017. 15

[25] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 14, 15, 16

[26] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *arXiv preprint arXiv:1507.06550*, 2015. 94, 96

[27] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 48, 49, 53, 54, 56, 58, 59, 62, 64, 65, 67, 70, 71, 74

[28] S. Chandra, N. Usunier, and I. Kokkinos. Dense and low-rank Gaussian CRFs using deep embeddings. In *International Conference on Computer Vision (ICCV)*, 2017. 49, 64

[29] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations (ICLR)*, 2017. 105

[30] C.-Y. Chen and K. Grauman. Efficient activity detection with max-subgraph search. In *CVPR*, 2012. 64

[31] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *arXiv:1412.7062*, 2014. 48, 64

[32] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. 93

[33] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 22

[34] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 34

[35] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *CVPR*, 2005. 76

[36] S. T. D. Xie and S. Zhu. Inferring âĂŸdark matterâĂŹ and âĂŸdark energyâĂŹ from videos. In *ICCV*, 2013. 93

[37] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *Transactions on Image Processing (TIP)*, 2007. 48

[38] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016. 93

[39] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 19

[40] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 43

[41] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006. 64, 78

[42] V. Delaitre, D. Fouhey, I. Laptev, J. Sivic, A. Efros, and A. Gupta. Scene semantics from long-term observation of people. In *ECCV*, 2012. 93

[43] 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. 93

[44] 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. 97

[45] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013. 21, 34, 37

[46] C. Doersch, A. Gupta, and A. A. Efros. Context as supervisory signal: Discovering objects with predictable context. In *ECCV*, 2014. 21, 34

[47] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 14

[48] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 32, 33, 34, 36, 37, 40, 41, 42, 43, 44, 45

[49] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 78, 85

[50] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 62, 64

[51] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv*, 2016. 34

[52] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 49

[53] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv*, 2016. 34

[54] M. Edmonds, F. Gao, X. Xie, H. Liu, S. Qi, Y. Zhu, B. Rothrock, and S.-C. Zhu. Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles. 2017. 104

[55] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision (ICCV)*, 1999. 48, 49

[56] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 93

[57] S. M. A. Eslami, N. Heess, and J. Winn. The shape boltzmann machine: a strong model of object shape. In *CVPR*, 2012. 21

[58] S. M. A. Eslami, N. Heess, and J. Winn. The shape boltzmann machine: a strong model of object shape. In *CVPR*, 2012. 34

[59] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, , and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 29

[60] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 2010. 41, 42

[61] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *ICCV*, 2013. 78

[62] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Neural Information Processing Systems (NIPS)*, 2016. 49, 54, 64, 67

[63] B. Fernando, E. Gavves, J. O. M., A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 78, 85

[64] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Neural Information Processing Systems (NIPS)*, 2016. 105

[65] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv*, 2015. 6, 9, 12

[66] D. A. Forsyth and J. Ponce. *Computer Vision - A Modern Approach, Second Edition.* Pitman, 2012. 9

[67] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic. People watching: Human actions as a cue for single-view geometry. In *ECCV*, 2012. 93

[68] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 30, 31, 45

[69] D. F. Fouhey, A. Gupta, and M. Hebert. Unfolding an indoor origami world. In *ECCV*, 2014. 30, 31, 45

[70] D. F. Fouhey, W. Kuo, A. A. Efros, and J. Malik. From lifestyle vlogs to everyday interactions. In *CVPR*, 2018. 12, 13

[71] D. F. Fouhey, X. Wang, and A. Gupta. In defense of the direct perception of affordances. *CoRR*, abs/1505.01085, 2015. 91, 93

[72] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. In *International Conference on Learning Representations (ICLR)*, 2016. 105

[73] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*. Springer, 1982. 47

[74] P. FÃűldiÃąk. Learning Invariance from Transformation Sequences. *Neural Computation*, 3(2):194–200, June 1991. 8

[75] A. Gallagher and T. Chen. Understanding groups of images of people. In *CVPR*, 2009. 94

[76] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. *CVPR*, 2015. 88

[77] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In *CVPR*, 2016. 64

[78] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*, 2017. 49

[79] J. Gibson. *The ecological approach to visual perception*. Boston: Houghton Mifflin, 1979. 93

[80] R. Girdhar, D. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 93

[81] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 42, 43, 65, 67

[82] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 94

[83] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 27, 29, 30

[84] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 32

[85] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018. 67

[86] G. Gkioxari, R. Girshick, and J. Malik. Actions and attributes from wholes and parts. In *ICCV*, 2015. 64

[87] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with r*cnn. In *ICCV*, 2015. 63

[88] G. Gkioxari, R. Girshick, P. Dollár, and K. He. Detecting and recognizing human-object intaractions. *CVPR*, 2018. 63, 64

[89] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 78

[90] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Computer Vision and Pattern Recognition (CVPR)*, 2009. 48

[91] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 45

[92] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 9

[93] Y. Gong, Y. Jia, T. K. Leung, A. Toshev, and S. Ioffe. Deep convolutional ranking for multilabel image annotation. In *ICLR*, 2014. 22, 34

[94] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 34

[95] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. *ICCV*, 2015. 8

[96] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. *ICCV*, 2015. 21

[97] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv:1706.02677*, 2017. 55, 71

[98] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fründ, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic. The "something something" video database for learning and evaluating visual common sense. *arXiv:1706.04261*, 2017. 63, 70, 75

[99] H. Grabner, J. Gall, and L. van Gool. What makes a chair a chair? In *CVPR*, 2011. 93

[100] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 9–20. ACM, 1998. 105

[101] A. Gupta and L. S. Davis. Objects in action: An approach for combining action understanding and object perception. In *CVPR*, 2007. 93

[102] A. Gupta, S. Satkin, A. Efros, and M. Hebert. From 3D scene geometry to human workspace. In *CVPR*, 2011. 93

[103] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2009. 63, 64

[104] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *Computer Vision and Pattern Recognition(CVPR)*, 2011. 93

[105] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 22, 78

[106] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 34

[107] B. Ham, M. Cho, C. Schmid, and J. Ponce. Proposal flow. In *CVPR*, 2016. 9

[108] J. Hamrick, P. Battaglia, and J. B. Tenenbaum. Internal physics models guide probabilistic judgments about object dynamics. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, 2011. 104

[109] J. B. Hamrick, P. W. Battaglia, T. L. Griffiths, and J. B. Tenenbaum. Inferring mass in complex scenes by mental simulation. *Cognition*, 2016. 104

[110] K. Han, R. S. Rezende, B. Ham, K.-Y. K. Wong, M. Cho, C. Schmid, and J. Ponce. Scnet: Learning semantic correspondence. *ICCV*, 2017. 9

[111] A. Harley, K. Derpanis, and I. Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *International Conference on Computer Vision (ICCV)*, 2017. 49, 64

[112] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*, 2017. 48, 59, 60, 65, 67

[113] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015. 55, 93

[114] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 11, 14, 15, 16, 17, 18, 47, 51, 52, 55

[115] F. C. Heilbron, W. Barrios, V. Escorcia, and B. Ghanem. Scc: Semantic context cascade for efficient action detection. In *CVPR*, 2017. 64

[116] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. *CVPR*, 2015. 79

[117] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016. 7, 9

[118] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 2015. 24

[119] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 2015. 38

[120] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006. 21

[121] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006. 34

[122] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995. 21

[123] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 55, 71

[124] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997. 47

[125] E. Hoffer and N. Ailon. Deep metric learning using triplet network. *CoRR*, /abs/1412.6622, 2014. 78

[126] E. Hoffer and N. Ailon. Deep metric learning using triplet network. *CoRR*, /abs/1412.6622, 2015. 22

[127] E. Hoffer and N. Ailon. Deep metric learning using triplet network. *arXiv*, 2015. 34

[128] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 1981. 9

[129] Y. Hoshen. Multi-agent predictive modeling with attentional commnets. In *Neural Information Processing Systems (NIPS)*, 2017. 49

[130] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 63, 64

[131] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. In *CVPR*, June 2014. 78

[132] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, 2013. 9

[133] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 6, 9, 14, 15, 16, 18

[134] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 55

[135] H. Izadinia and M. Shah. Recognizing complex events using large margin joint low-level event model. *ECCV*, 2012. 78

[136] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. 12

[137] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis. Representing videos using mid-level discriminative patches. In *CVPR*, 2013. 78

[138] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, 2016. 64

[139] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. *CVPR*, 2013. 86

[140] J. Janai, F. Güney, A. Ranjan, M. Black, and A. Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, 2018. 9

[141] D. Jayaraman and K. Grauman. Learning image representations tied to egomotion. In *ICCV*, 2015. 8, 34

[142] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015. 78

[143] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. 13, 15, 16

[144] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *International Conference on Machine Learning (ICML)*, 2010. 49

[145] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 2013. 64, 78

[146] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, /abs/1408.5093, 2014. 19

[147] H. Jiang and J. Xiao. A linear approach to matching cuboids in RGBD images. In *CVPR*, 2013. 93

[148] Y.-G. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes. *http://crcv.ucf.edu/THUMOS14/*, 2014. 79

[149] Y.-G. Jiang, Q. Dai, X. Xue, W. Liu, and C.-W. Ngo. Trajectory-based modeling of human actions with motion reference points. In *ECCV*, 2012. 78

[150] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *ICMR*, 2011. 79

[151] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *ICPR*, 2010. 9

[152] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *TPAMI*, 2012. 9

[153] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 64, 78, 79, 86

[154] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv:1705.06950*, 2017. 14, 48, 55, 71

[155] D. G. Kendall. A survey of the statistical theory of shape. *Statistical Science*, pages 87–99, 1989. 73

[156] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *CVPR*, 2013. 9

[157] S. Kim, D. Min, B. Ham, S. Jeon, S. Lin, and K. Sohn. Fcss: Fully convolutional self-similarity for dense semantic correspondence. In *CVPR*, 2017. 9

[158] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. 13

[159] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 98

[160] D. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 34

[161] D. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 97

[162] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 63, 64, 69

[163] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *European Conference on Computer Vision (ECCV)*, 2012. 105

[164] H. Kjellstrom, J. Romero, D. Martinez, and D. Kragic. Simultaneous visual recognition of manipulation actions and manipulated objects. In *ECCV*, 2008. 93

[165] A. Klaser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. 64, 76, 78

[166] H. Koppula and A. Saxena. Physically-grounded spatio-temporal object affordances. In *ECCV*, 2014. 93

[167] H. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *TPAMI*, 2015. 93

[168] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Neural Information Processing Systems (NIPS)*, 2011. 48, 64

[169] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 64

[170] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 32

[171] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 19, 22, 24

[172] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 93, 97

[173] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T.Serre. Hmdb: A large video database for human motion recognition. In *ICCV*, 2011. 79, 83, 86

[174] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Neural Information Processing Systems (NIPS)*, 2015. 105, 109, 110

[175] M. P. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *CVPR*, 2010. 64

[176] L. Ladický, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014. 30, 31, 45

[177] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001. 48, 64

[178] T. Lan, Y. Zhu, A. R. Zamir, and S. Savarese. Action recognition by hierarchical mid-level action elements. In *ICCV*, 2015. 78

[179] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015. 64, 78

[180] I. Laptev. On space-time interest points. *IJCV*, 64, 2005. 64, 78

[181] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 64, 78

[182] D. Larlus and A. Vedaldi. Anchornet: A weakly supervised network to learn geometry-sensitive features for semantic matching. In *CVPR*, 2017. 9

[183] Q. V. Le, M. A. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012. 19, 21

[184] Q. V. Le, M. A. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012. 34

[185] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 21

[186] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 64, 78

[187] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990. 19, 93

[188] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 32, 47

[189] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 21

[190] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 34

[191] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 48

[192] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. In *International Conference on Machine Learning (ICML)*, 2016. 104, 105

[193] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011. 9

[194] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018. 9, 12

[195] F. Li, C. Gan, X. Liu, Y. Bian, X. Long, Y. Li, Z. Li, J. Zhou, and S. Wen. Temporal modeling approaches for large-scale youtube-8m video understanding. *arXiv preprint arXiv:1707.04555*, 2017. 62, 64

[196] N. Li and J. J. DiCarlo. Unsupervised natural experience rapidly alters invariant object representation in visual cortex. *Science (New York, N.Y.)*, 321(5895):1502–1507, September 2008. 8

[197] Q. Li, A. Arnab, and P. H. Torr. Holistic, instance-level human parsing. *arXiv*, 2017. 16

[198] W. Li, S. Azimi, A. Leonardis, and M. Fritz. To fall or not to fall: A visual approach to physical stability prediction. *arXiv:1604.00066*, 2016. 105

[199] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár. Unsupervised learning of edges. In *CVPR*, 2016. 8, 32, 34, 42

[200] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. In *ICLR*, 2016. 64

[201] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, and S. Yan. Computational baby learning. *CoRR*, abs/1411.2861, 2014. 22, 27

[202] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, and S. Yan. Computational baby learning. *arXiv*, 2014. 40

[203] Z. Liang, S. Ding, and L. Lin. Unconstrained facial landmark localization with backbone-branches fully-convolutional networks. In *arXiv:1507.03409*, 2015. 94

[204] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 60, 67

[205] T. Lin, M. Maire, S. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *arXiv*, 2014. 33, 44, 48, 59, 67, 93

[206] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *TPAMI*, 2011. 9, 14, 15, 16

[207] S. Liu, X. Liang, L. Liu, X. Shen, J. Yang, C. Xu, X. Cao, and S. Yan. Matching-cnn meets knn: Quasi-parametric human parsing. In *CVPR*, 2015. 22

[208] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz. Learning affinity via spatial propagation networks. In *Neural Information Processing Systems (NIPS)*, 2017. 49, 64

[209] S. Liu, G. Zhong, S. De Mello, J. Gu, M.-H. Yang, and J. Kautz. Switchable temporal propagation network. *ECCV*, 2018. 8

[210] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 32, 45

[211] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004. 19

[212] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. *ECCV*, 2016. 63

[213] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *IJCAI*, 1981. 9

[214] P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In *CVPR*, 2012. 21

[215] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei. Unsupervised learning of long-term motion dynamics for videos. *CVPR*, 2017. 8

[216] C.-Y. Ma, A. Kadav, I. Melvin, Z. Kira, G. AlRegib, and H. P. Graf. Attend and interact: Higher-order object interactions for video understanding. In *CVPR*, 2018. 64

[217] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: A path-based method for plausible image interpolation. In *Proceedings of SIGGRAPH, ACM Transactions on Graphics*, 2009. 9

[218] K. Marino, R. Salakhutdinov, and A. Gupta. The more you know: Using knowledge graphs for image classification. In *CVPR*, 2017. 64

[219] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv*, 2015. 8, 34

[220] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations (ICLR)*, 2016. 105

[221] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV Workshops*, 2009. 78

[222] S. Meister, J. Hur, and S. Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. *AAAI*, 2018. 9

[223] E. Mémin and P. Pérez. Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Transactions on Image Processing*, 1998. 9

[224] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017. 62

[225] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *ICML*, 2009. 19, 21

[226] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *CVPR*, 2016. 104, 105

[227] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi. "what happens if..." learning to predict the effect of forces in images. In *ECCV*, 2016. 93, 104, 105

[228] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010. 51

[229] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 78, 85, 86

[230] B. Ni, X. Yang, and S. Gao. Progressively parsing interactional objects for fine grained action detection. In *CVPR*, 2016. 64

[231] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 1997. 19, 21, 34

[232] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016. 49

[233] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, 2016. 64

[234] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 8, 32, 34

[235] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CoRR*, /abs/1405.4506, 2014. 78, 86

[236] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014. 64, 78, 85

[237] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, , and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision (ECCV)*, 2016. 104, 105

[238] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016. 93

[239] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA*, 2016. 32

[240] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA*, 2016. 93

[241] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 13, 14, 15, 17

[242] R. Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010. 78

[243] S. Qi, B. Jia, and S.-C. Zhu. Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction. In *International Conference on Machine Learning (ICML)*, 2018. 105

[244] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 64

[245] A. Quattoni and A.Torralba. Recognizing indoor scenes. In *CVPR*, 2009. 94

[246] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *CVPR*, 2005. 9

[247] A. Ranjan and M. Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017. 6, 9

[248] M. A. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007. 19

[249] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 43, 44, 65, 67

[250] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 60

[251] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 9

[252] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Deepmatching: Hierarchical deformable dense matching. *IJCV*, 2016. 9

[253] E. Rivlin, S. Dickinson, and A. Rosenfeld. Recognition by functional parts. In *CVIU*, 1995. 93

[254] I. Rocco, R. Arandjelović, and J. Sivic. Convolutional neural network architecture for geometric matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 9, 12

[255] I. Rocco, R. Arandjelovic, and J. Sivic. End-to-end weakly-supervised semantic alignment. In *CVPR*, 2018. 9, 11

[256] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele. Script data for attribute-based recognition of composite activities. *ECCV*, 2012. 78

[257] M. Rubinstein, C. Liu, and W. T. Freeman. Towards longer long-range motion trajectories. *BMVC*, 2012. 9

[258] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 1986. 47

[259] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 21, 32

[260] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. 55, 71

[261] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 82

[262] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006. 21, 34

[263] 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. 64

[264] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 64, 78

[265] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *ICCV*, 2008. 9

[266] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Neural Information Processing Systems (NIPS)*, 2017. 49, 51, 64

[267] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009. 49, 64

[268] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015. 49, 64

[269] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, 2013. 21

[270] I. K. Sethi and R. Jain. Finding Trajectories of Feature Points in a Monocular Image Sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(1):56–73, January 1987. 9

[271] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 59, 74

[272] G. A. Sigurdsson, O. Russakovsky, and A. Gupta. What actions are needed for understanding human actions in videos? In *ICCV*, 2017. 62, 73

[273] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision (ECCV)*, 2016. 48, 55, 59, 63, 70, 74

[274] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 30

[275] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 44

[276] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer. Fracking deep convolutional image descriptors. *CoRR*, /abs/1412.6537, 2014. 78

[277] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 81

[278] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, /abs/1312.6034, 2013. 88

[279] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 49, 78, 81, 83, 84, 85, 86

[280] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. 14

[281] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. 33, 36, 39, 62, 64

[282] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 55, 71

[283] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012. 21, 34, 37

[284] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *ICCV*, 2005. 21, 34

[285] J. Song, L. Wang, L. Van Gool, and O. Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos. In *CVPR*, 2017. 16

[286] S. Song, S. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 94

[287] Y. Song, L.-P. Morency, and R. Davis. Action recognition by hierarchical sequence summarization. In *CVPR*, 2013. 78

[288] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, /abs/1212.0402, 2012. 77, 79, 83, 86

[289] N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, 2012. 19, 21

[290] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. *arXiv*, 2015. 8

[291] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015. 21

[292] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. *arXiv*, 2015. 34

[293] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, /abs/1502.04681, 2015. 64, 78, 105

[294] L. Stark and K. Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. In *PAMI*, 1991. 93

[295] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed dirichlet processes. In *NIPS*, 2005. 21

[296] C. Sun and R. Nevatia. Active: Activity concept transitions in video event classification. *ICCV*, 2013. 78

[297] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM Multimedia*, 2015. 64, 78, 79

[298] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010. 9

[299] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 6, 9

[300] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 78

[301] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012. 78

[302] Y. Tang, R. Salakhutdinov, and G. Hinton. Robust boltzmann machines for recognition and denoising. In *CVPR*, 2012. 21

[303] Y. Tang, R. Salakhutdinov, and G. Hinton. Robust boltzmann machines for recognition and denoising. In *CVPR*, 2012. 34

[304] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *ECCV*, 2010. 19, 21, 64, 78

[305] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision (ICCV)*, 1998. 50

[306] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015. 48, 49, 53, 62, 75

[307] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 64, 78

[308] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 62, 64

[309] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *NIPS*, 2017. 8

[310] N. Ufer and B. Ommer. Deep semantic feature matching. In *CVPR*, 2017. 9

[311] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013. 42, 43

[312] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017. 7, 9

[313] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. 8

[314] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NIPS)*, 2017. 12, 49, 51, 57, 68

[315] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv*, 2017. 9

[316] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 19, 21

[317] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 34

[318] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Neural Information Processing Systems (NIPS)*, 2016. 105

[319] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. *ECCV*, 2018. 8, 14, 15, 16

[320] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from variational autoencoders. In *ECCV*, 2016. 34

[321] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from variational autoencoders. In *European Conference on Computer Vision*, 2016. 97, 105, 108

[322] G. Wallis. Spatio-temporal influences at the neural level of object recognition. *Network: Computation in Neural Systems*, 9(2):265–278, January 1998. 8

[323] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 23

[324] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 36

[325] H. Wang, A. Klaser, C. Schmid, and L. Cheng-Lin. Action recognition by dense trajectories. In *CVPR*, 2011. 78

[326] H. Wang and C. Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision (ICCV)*, 2013. 49, 64

[327] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 76, 78, 86

[328] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 22, 23

[329] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 34, 39

[330] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 49

[331] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 64, 78, 83, 86

[332] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *CoRR*, /abs/1507.02159, 2015. 82, 83, 84, 85, 86

[333] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 62, 64

[334] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013. 7, 9

[335] X. Wang, A. Farhadi, and A. Gupta. Actions $\sim$ Transformations. In *CVPR*, 2016. 3, 4, 64

[336] X. Wang, D. F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 31, 45

[337] X. Wang, R. Girdhar, and A. Gupta. Binge watching: Scaling affordance learning from sitcoms. In *CVPR*, 2017. 3, 4

[338] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *CoRR*, 2017. 3, 4

[339] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018. 12

[340] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018. 65, 66, 67, 68, 69, 70, 71, 73, 74

[341] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2, 4, 78

[342] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 8, 14

[343] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 32, 33, 34, 35, 36, 38, 39, 40, 42, 43, 44, 45

[344] X. Wang and A. Gupta. Videos as space-time region graphs. In *ECCV*, 2018. 3, 4

[345] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, 2017. 2, 4

[346] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, 2017. 8, 14, 15, 16

[347] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 2, 4

[348] Y. Wang and G. Mori. Hidden part models for human action recognition: Probabilistic vs. max-margin. *TPAMI*, 2011. 64, 78

[349] Y. Wang, Y. Yang, and W. Xu. Occlusion aware unsupervised learning of optical flow. In *CVPR*, 2018. 9

[350] N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, and D. Zoran. Visual interaction networks. In *Neural Information Processing Systems (NIPS)*, 2017. 49, 64, 105

[351] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV*, 2000. 21

[352] P. Winston, T. Binford, B. Katz, and M. Lowry. Learning physical description from functional definitions, examples and precedents. In *MIT Press*, 1984. 93

[353] L. Wiskott and T. J. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, April 2002. 8

[354] J. N. Wood. A smoothness constraint on the development of object recognition. *Cognition*, 153:140–145, 2016. 1, 8

[355] J. N. Wood and S. M. W. Wood. The development of newborn object recognition in fast and slow visual worlds. *Proceedings. Biological Sciences*, 283(1829), April 2016. 8

[356] H. Wu, A. C. Sankaranarayanan, and R. Chellappa. In Situ Evaluation of Tracking Algorithms Using Time Reversed Chains. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 9

[357] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, 2016. 105

[358] J. Wu, E. Lu, P. Kohli, W. T. Freeman, and J. B. Tenenbaum. Learning to see physics via visual de-animation. In *Neural Information Processing Systems (NIPS)*, 2017. 104, 105

[359] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single image 3d interpreter network. In *European Conference on Computer Vision (ECCV)*, 2016. 93

[360] J. Wu, I. Yildirim, J. J. Lim, W. T. Freeman, and J. B. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Neural Information Processing Systems (NIPS)*, 2015. 105

[361] J. Wu, Y. Zhang, and W. Lin. Towards good practices for action video encoding. *CVPR*, 2014. 86

[362] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 9

[363] Z. Wu, Y. Fu, Y.-G. Jiang, and L. Sigal. Harnessing object and scene semantics for large-scale video understanding. In *CVPR*, 2016. 64

[364] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *ACM Multimedia*, 2015. 64, 78, 85, 86

[365] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 60, 64

[366] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. In *arXiv:1712.04851*, 2017. 62

[367] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. The Microsoft 2016 Conversational Speech Recognition System. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017. 49

[368] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. *CVPR*, 2015. 78

[369] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Neural Information Processing Systems (NIPS)*, 2016. 105

[370] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 64

[371] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. *arXiv*, 2018. 15

[372] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *TPAMI*, 2013. 15

[373] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010. 93

[374] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *Computer Vision and Pattern Recognition (CVPR)*, 2010. 63, 64

[375] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 64

[376] M. Yatskar, L. Zettlemoyer, and A. Farhadi. Situation recognition: Visual semantic role labeling for image understanding. In *CVPR*, 2016. 63, 64

[377] T. Ye, X. Wang, J. Davidson, and A. Gupta. Interpretable intuitive physics model. In *ECCV*, 2018. 3, 4

[378] D.-J. Yi, N. B. Turk-Browne, J. I. Flombaum, M.-S. Kim, B. J. Scholl, and M. M. Chun. Spatiotemporal object continuity in human ventral visual cortex. *Proceedings of the National Academy of Sciences*, 105(26):8840–8845, July 2008. 8

[379] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018. 9

[380] Y. Yuan, X. Liang, X. Wang, D.-Y. Yeung, and A. Gupta. Temporal dynamic graph lstm for action-driven video object detection. In *ICCV*, 2017. 64

[381] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 49, 62, 64

[382] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. *29th DAGM Symposium on Pattern Recognition*, 2007. 84

[383] R. Zhang, J. Wu, C. Zhang, W. T. Freeman, and J. B. Tenenbaum. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. In *Proceedings of the 38th AnnualConference of the Cognitive Science Society*, 2016. 104

[384] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 32, 34

[385] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 32, 34

[386] Y. Zhao and S. Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, 2013. 93

[387] B. Zheng, Y. Zhao, J. Yu, K. Ikeuchi, and S.-C. Zhu. Scene understanding by reasoning stability and safety. *International Journal of Computer Vision (IJCV)*, 2015. 104

[388] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, 2015. 49, 64

[389] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 94

[390] B. Zhou, A. Andonian, and A. Torralba. Temporal relational reasoning in videos. *arXiv*, 2017. 64, 74, 75

[391] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015. 41

[392] Q. Zhou, X. Liang, K. Gong, and L. Lin. Adaptive temporal encoding network for video instance-level human parsing. In *ACM MM*, 2018. 13, 15, 16, 17

[393] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 9

[394] T. Zhou, Y. Jae Lee, S. X. Yu, and A. A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *CVPR*, 2015. 9

[395] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, 2016. 7, 9

[396] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision (ECCV)*, 2016. 105, 106, 108

[397] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *ICCV*, 2015. 9

[398] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *ICCV*, 2013. 64, 78

[399] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017. 9

[400] Y. Zhu, Y. Zhao, and S. Zhu. Understanding tools: Task-oriented object modeling, learning and recognition. In *CVPR*, 2015. 93

[401] Y. Zhu, C. Jiang, Y. Zhao, D. Terzopoulos, and S.-C. Zhu. Inferring forces and learning human utilities from videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 104

[402] Y. Zhu, A. Fathi, and L. Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014. 93

[403] W. Zou, S. Zhu, K. Yu, and A. Y. Ng. Deep Learning of Invariant Features via Simulated Fixations in Video. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3203–3211. Curran Associates, Inc., 2012. 8

[404] W. Y. Zou, S. Zhu, A. Y. Ng, and K. Yu. Deep learning of invariant features via simulated fixations in video. In *NIPS*, 2012. 19, 21