Unsupervised Learning of Visual Representations using Videos

Xiaolong Wang, Abhinav Gupta
The Robotics Institute, Carnegie Mellon University
{xiaolonw, abhinavg}@cs.cmu.edu

Abstract

Is strong supervision necessary for learning a good visual representation? Do we really need millions of semantically-labeled images to train a ConvNet? In this paper, we present a simple yet surprisingly powerful approach for unsupervised learning of ConvNets. Specifically, we use hundreds of thousands of unlabeled videos from the web to learn visual representations. Our key idea is that we track millions of patches in these videos. Visual tracking provides the key supervision. That is, two patches connected by a track should have similar visual representation in deep feature space since they probably belong to the same object or object part. We design a Siamese-triplet network with a ranking loss function to train this ConvNet representation. Without using a single image from ImageNet, just using 100K unlabeled videos and the VOC 2012 dataset, we train an ensemble of unsupervised networks that achieves 52% mAP (no bounding box regression). This performance comes tantalizingly close to its ImageNet-supervised counterpart, an ensemble which achieves a mAP of 54.4%. We also show that our unsupervised network can perform competitive in other tasks such as surface-normal estimation.

1. Introduction

What is a good 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 [29] and HOG [5] 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 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 paper, we explore the alternative: how can we exploit the unlabeled visual data on the web to train ConvNets (e.g. AlexNet [21])? In the past, there have been several attempts at unsupervised learning using millions of static images [23, 41] or frames extracted from videos [50, 44, 31]. The most common architecture used is an auto-encoder which learns representations based on its ability to recon-
construct the input images [32, 3, 45, 33]. 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 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 from ImageNet [34], just using 100K unlabeled videos and VOC 2012 dataset, we train an ensemble of AlexNet networks that achieves 52% mAP (no bounding box regression)\(^2\). 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 paper come closest to standard supervised ConvNets approaches (which use millions of semantically-labeled images) among unsupervised approaches.

## 2. Related Work

Unsupervised learning of visual representations has a rich and diverse history starting from original auto-encoders work of Olhausen and Field [32] 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 [17, 49, 42]. These algorithms essentially capture co-occurrence 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 [39, 35]. Some of these recent algorithms also focus on learning mid-level representations rather than discovering semantic classes themselves [38, 6, 7].

The third class of algorithms and more related to our paper is unsupervised learning of visual representations from the pixels themselves using deep learning approaches [18, 23, 41, 36, 26, 43, 8, 30, 2, 45]. Starting from the seminal work of Olhausen and Field [32], the goal is to learn visual representations which are (a) sparse and (b) reconstructive. Olhausen and Field [32] 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 [18] to train a deep belief network in an unsupervised manner via stacking layer-by-layer RBMs. Similar to this, Bengio et al. [3] investigated stacking of both RBMs and autoencoders. As a next step, Le et al. [23] 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. [36] 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 feature representations using videos [24, 40, 50, 14, 44, 31]. Early work such as [50] 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. [14] proposed to learn auto-encoders based on the slowness prior. Other approaches such as Taylor et al. [44] trained convolutional gated RBMs to learn latent representations from pairs of successive images. This was extended in a recent work by Srivastava et al. [40] 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 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 [47, 28, 4, 15, 13, 19]. For example, Chopra et al. [4] proposed to learn convolutional networks in a

---

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

\(^2\)Papers referenced as [24, 40, 50, 14, 44, 31]
siamese architecture for face verification. Wang et al. [47] 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 [27], 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. Overview

Our goal is to train convolutional neural networks using hundreds and thousands of unlabeled videos from the Internet. We follow the AlexNet architecture [21] 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 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 [47] 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.

4. 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 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...
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 4). For our experiments, we take the image with size $227 \times 227$ as input. The base network is based on the AlexNet architecture [21] 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.

5.2. Ranking Loss Function

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

$$D(X_1, X_2) = 1 - \frac{f(X_1) \cdot f(X_2)}{\|f(X_1)\| \|f(X_2)\|}.$$  \hspace{1cm} (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 $S$, 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 loss as,

$$L(X_i, X_i^+, X_i^-) = \max\{0, D(X_i, X_i^+) - D(X_i, X_i^-) + M\},$$  \hspace{1cm} (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,

$$\min_{W} \frac{\lambda}{2} \|W\|^2 + \sum_{i=1}^{N} \max\{0, D(X_i, X_i^+) - D(X_i, X_i^-) + M\},$$  \hspace{1cm} (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 $\lambda = 0.0005$. 

Figure 3. Examples of patch pairs we obtain via patch mining in the videos.
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 [12]. 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 paper, 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.

### 5.5. 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 1).

5.6. 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 \( X_i^- \in \mathcal{B} \) 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. 2.

For learning, we download 100K videos from YouTube using the URLs provided by [27]. 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 networks, we set the batch size as \(|\mathcal{B}| = 100\), 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.

6. 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 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 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 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.

6.1. Object Detection

For object detection, we perform our experiments on PASCAL VOC 2012 dataset [9]. We follow the detection pipeline introduced in RCNN [12], which borrowed the
Table 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.

| 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 |
| 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 |

Figure 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.

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 settings.

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 ImageNet pre-trained network. Notice that the results for VOC 2012 reported in RCNN [12] are obtained by only fine-tuning on the train set without using the val set. The mAP for VOC 2012 reported in [12] 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 [12] 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 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 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 [12] which uses ImageNet pre-trained models. Following the procedure in [12], 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 sepa-
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.

### 6.2. 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 [37], 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 [10] which is used by different methods [10, 22, 11] on this task.

To apply deep learning to this task, we followed the same approach of ensembling, leading to huge performance boosts. For comparisons, we also trained networks from scratch as well as using ImageNet pre-trained. We show our results in Table 2. Compared to the recent results which do not use external data [10, 22, 11], we show that we can get reasonable results even using this small amounts of training data. Our approach (unsup + ft) is generally 2 ~ 3% better than network trained from scratch in 5 different metrics. We show a few qualitative results in Figure 7.

**Acknowledgement:** This work was partially supported by ONR MURI N000141010934, NSF IIS 1320083, and gifts from Google and Yahoo!. AG was partially supported by Bosch Young Faculty Fellowship. The authors would like to thank Yahoo! and Nvidia for the compute cluster and GPU donations respectively.

### Table 2. Results on NYU v2 for per-pixel surface normal estimation, evaluated over valid pixels.

| Method          | Lower Better | Higher Better |
|-----------------|--------------|---------------|
| scratch         | 36.0         | 20.5          |
| unsup + ft      | 35.2         | 23.2          |
| ImageNet + ft   | 33.3         | 20.8          |
| UNFOLD [11]     | 35.1         | 19.2          |
| Discr. [22]     | 32.5         | 22.4          |
| 3DP (MW) [10]   | 36.0         | 20.5          |
| (unsup + iterative ft) | 48% mAP (unsup + iterative ft) | 1% better than the initial fine-tuned network. |

![Figure 7. Surface normal estimation results on NYU dataset.](image)

For visualization, we use green to represent horizontal surface, blue representing facing right and red representing facing left, i.e., blue → X; green → Y; red → Z.
[45] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In ICML, 2008. 2

[46] H. Wang and C. Schmid. Action recognition with improved trajectories. In ICCV, 2013. 4

[47] 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. 2, 3

[48] X. Wang, D. F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In CVPR, 2015. 8

[49] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In ECCV, 2000. 2

[50] W. Y. Zou, S. Zhu, A. Y. Ng, and K. Yu. Deep learning of invariant features via simulated fixations in video. In NIPS, 2012. 1, 2