Two-Stream Convolutional Networks for Action Recognition in Videos

1 Citation

Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." Advances in neural information processing systems. 2014.

`http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf`

2 Abstract

We do action recognition by merging two CNNs (one for spatial info and one for temporal info). We also show how you can leverage multiple datasets to train your CNN.

3 Introduction

Our spatial CNN is trained on static image frames (e.g. from ImageNet). Our temporal CNN is trained on dense optical flow. This two stream structure is similar to what happens in human vision system.

4 Related Work

Previous work would compute Histogram of Oriented Gradients (HoG) and Histogram of Optical Flow (HoF) features, quantize them with a Bag of Features (BoF) representation, pool over several grids (like spatial pyramid pooling), and classify with an SVM.

State of the art work computes features of dense trajectories (Motion Boundary Histogram), corrects for camera motion, and encodes info with Fisher Vectors.

People have tried making deep models. The HMAX model was deep, but the layers were handcrafted. Other work has trained end to end CNNs, but their learned features were worse than handcrafted features.

5 Two Stream Architecture for Video Recognition

We train a spatial CNN and a temporal CNN. To fuse them, we consider both averaging and stacking an SVM on the L2-normalized softmax scores.

The spatial CNN works pretty well, because you can often identify an action with one frame. We can also use ImageNet data here.

6 Optical Flow ConvNets

Let $\mathbf{d}_t(u, v)$ be the displacement of the point at (u, v) between frame t and $t + 1$. We have both a horizontal and vertical displacement. If we consider L frames, we then get $2L$ input channels. If the frame has size $w \times h$, our ConvNet input is $w \times h \times 2L$.

Thus, our input for frame τ is:

$$I_\tau(u, v, 2k - 1) = d^x_{r+k-1}(u, v)$$

$$I_\tau(u, v, 2k) = d^y_{r+k-1}(u, v)$$

for $u \in [1...w]$, $v \in [1...h]$, $k \in [1...L]$.

An alternative to this approach is to find trajectories of points and compute optical flows of those. That means the argument to the d function is $\mathbf{p}_k$ instead of x or y. $\mathbf{p}_k$ is the kth point on the trajectory that starts at (u, v) in frame τ.

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \mathbf{d}_{r+k-2}(\mathbf{p_{k-1}})$$

where $\mathbf{p}_0 = (u, v)$

Thus, the former approach holds the location fixed and samples the trajectories. The latter holds the trajectory fixed and samples the locations.

In the design above, we only consider the flow from τ to $\tau + L$. Another option is to consider the forward flow from τ to $\tau + L/2$ and the backward flow from τ to $\tau - L/2$ and stack them.

We set $w = h = 224$.

7 Multi-task Learning

We can't pretrain the temporal ConvNet. We pretrain on UCF-101 (9.5K videos) and HMDB (3.7K). To enable this, we put two sibling classification layers (one for UCF-101 and one for HMDB) and our loss function sums both classification losses.

8 Implementation Details

Both convnets have five conv layers, two fully fonnected layers, and a softmax. Pooling happens at layers 1, 2, and 5. Dropout is used on the FC layers. We use local response localization (LRN). The only difference between the nets is that the temporal net does not have a second LRN layer to reduce memory usage.

We use minibatch (256) SGD with momentum (0.9). We do random cropping, flipping, and RGB jittering. We have scheduled learning rate drops. Training stops at 80K iterations.

At test time, we sample 25 frames with equal temporal spacing from the video. For each frame, we take the corners and centers and their flips and feed them into our ConvNets. We average across each of these inputs to compute overall class scores.

We pretrain our spatial ConvNet on ImageNet. We actually beat AlexNet because he samples from the middle 256×256 images, but we sample from the whole image.

We train with Caffe on 4 NVIDIA Titans - it takes a day. Optical flow is computed from OpenCV's GPU implementation. We pre-compute the optical flow and compress them.

9 Evaluation

We evaluate on UCF-101 (101 actions) and HMDB (51 actions).

For the spatial CNN, we consider three approaches (1) train from scratch on UCF-101, (2) pretrain on ImageNet and fine-tune, (3) pretrain on ImageNet and fine-tune FC layers only. Pretraining on ImageNet gives a huge boost. Fine-tuning on all layers is marginally better than fine-tuning on the FC layers only.

For the temporal CNN, $L > 1$ helps - we pick $L = 10$. Mean subtraction helps. Optical flow tracking is slightly better than trajectory stacking. Bi-directional is slightly better than uni-directional. Teh temporal CNN is way better than the spatial CNN. Multi-task learning is the best way to combine the two datasets.

To combine the CNNs, we could theoretically add FC layers after both of them, but this would overfit. So, we consider averaging and stacking an SVM instead. The fusion is better than each CNN taken separately. The SVM beats averaging. Bi-directional optical flow does not help if fusing. The temporal CNN is better than the spatial CNN.

We set state of the art.

10 Conclusions and Directions for Improvement

The temporal CNN is better than the spatial CNN. It is trained on optical flow instead of raw pixels. Multi-task learning helps a lot. It may help to pool over trajectories somehow. Also, we handle camera motion with mean subtraction, but it may be worth it to add a better technique.