# Activity Recognition in Temporally Untrimmed Videos

Bryan Anenberg
Stanford University
anenberg@stanford.edu

Norman Yu
Stanford University
normanyu@stanford.edu

## Abstract

*We investigate strategies to apply Convolutional Neural Networks (CNNs) to recognize activities in temporally untrimmed videos. We consider techniques to segment a video into scenes in order to filter frames relevant to the activity from the background and irrelevant frames. We evaluate CNNs trained on frames sampled from the videos and present an approach towards implementing the two-stream CNN architecture outlined in [8]. The system is trained and evaluated on the THUMOS Challenge 2014 data set, which is comprised of the UCF-101 and additional temporally untrimmed videos.*

## 1. Introduction

The goal of the project is to develop an algorithm for automatically recognizing a large number of action categories from videos. An action can be either a simple atomic movement performed by a single person or a complex scenario involving multiple people. The algorithm should recognize human actions from videos in a realistic setting. Videos of a given action class could involve different backgrounds, different actors, and be captured from different camera viewpoints. This project addresses the task of recognizing human activities in temporally untrimmed videos, where the action of interest may occur at any time in the video.

The overall plan to approach the problem is to first extract proposals in the form of time ranges about where the activity may occur in the video. With this information, our system decomposes the video into spatial and temporal components and trains a separate Convolutional Neural Network (CNN) on either stream. We demonstrate the performance of different region proposal heuristics, different frame sampling techniques, and the spatial vs. temporal streams.

Figure 1: Sample activity classes from the UCF 101 data set

## 2. Related Work

### 2.1. Review of Previous Work

#### 2.1.1 Feature Based Approaches

The most popular state-of-the-art video classification systems rely on well designed features. The typical pipeline involves feature extraction, feature encoding, and then classification. For instance, [10] uses Improved Trajectory Features to densely samples feature points in each frame and tracks their motion using optical flow. Feature descriptors such as trajectory, histogram of oriented gradient (HOG), histogram of optical flow (HOF), and motion boundary histograms (MBH) are computed along the trajectories of feature points to capture shape, appearance, and motion information. The features are then encoded into a Bag of Features representation using a learned k-means dictionary. A classifier such as an SVM is trained on the quantized features to distinguish the video classes.

#### 2.1.2 Convolutional Neural Networks for Activity Recognition in Video

Recently [8] and [4] demonstrated impressive performance classifying activities in videos purely using CNNs. [4] treats every video as a bag of short, fixed-sized clips and extends the connectivity of the network across frames in an attempt to let the network discover spatio-temporal features. [4] evaluates various CNN architectures such as the Slow Fusion model, concludes that the single-frame architecture already exhibits strong performance, and demonstrates that the features learned no teh large Sports-1M data set are generic and generalize to video classification on the UCF-101 data set.

[8] introduces an architecture based on two separate recognition streams (spatial and temporal) which are combined by either averaging or training a multi-class SVM on the L2-normalized softmax scores as features. The spatial stream performs action recognition from still video frames, while the temporal stream recognizes actions from motion in the form of dense optical flow. [8] demonstrates that it is useful to pre-train the spatial stream CNN on ImageNet [7] since training on the UCF-101 data set alone leads to overfitting, and that a CNN trained on static images video frames alone is fairly competitive. The input to the temporal stream architecture is a stack of optical flow $dX$ and $dY$ channels of $L$ consecutive frames to form a total of $2L$ input channels. The key conclusion is that the CNN trained on stacked optical flow images is highly beneficial as it provides the network with more long-term motion information. Further, the temporal and spatial recognition streams are complementary as their fusion significantly improves either stream's individual performance to achieve an overall 87% accuracy.

### 2.2. Contribution of this Paper

The CNN systems for activity recognition described in [8; 4] do not exceed the performance of the best feature baselines. This paper does not present a algorithm to beat the state-of-the-art systems. Rather, this paper attempts to extend the CNNs presented in [8; 4] to classify temporally untrimmed videos. This paper implements a CNN to perform activity recognition, and explores pre-processing techniques to extend the system to classify untrimmed videos

## 3. Approach

The goal of the paper is to present a strategy for applying activity recognition to temporally untrimmed videos. Relying on the evidence from [8] and [4], we pre-train the weights of our CNN architecture on ImageNet [7]. We first establish the baseline performance of a system trained and tested on the UCF-101 temporally trimmed videos. We then extend the experiment to activity recognition on the temporally untrimmed videos without any pre-processing.Next, we evaluate the performance of two different temporal scene proposal heuristics. In parallel, we design an CNN trained on the stacked optical flow frames.

### 3.1. Scene Proposal Pre-processing

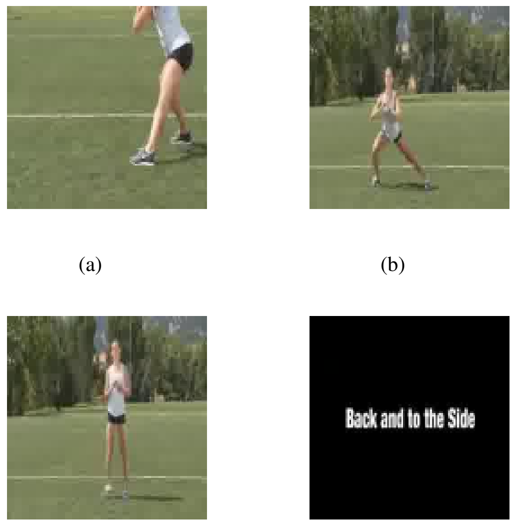
#### 3.1.1 Shot Boundary Detection

A video is usually composed of hundreds of disjoint shots concatenated together into a single file. A shot is a set of continuous frames captured in a single sequence by one camera. Shot boundary detection scores the potential for a boundary between two adjacent frames using such metrics as the chi-squared distance between the color histograms of frames. The algorithm predicts the existence of a boundary between frames if the score is about a set threshold. This project relies on the Shotdetect [6] to perform this analysis. Given the frame ids of the scene boundaries, we apply a heuristic with the goal of eliminating scenes which have a low likelihood of containing an activity. Through observation, we noticed that short scenes usually correspond to advertisements or introductions. As a first attempt we decided to retain video frames that belong to scenes of longer duration than the median length scene. This approach ignores the fact that long scenes are likely to portray imagery of the surrounding environment and background, rather than the actions of interest. Thus, we consider the performance of the CNN when trained on frames sampled from scenes of length $0.2D$ to $0.8D$ where $D$ is the duration of the longest scene.

It would be useful to consider the distribution of scene lengths in every video to understand the what percentage of frames are removes using the scene selection heuristic. As illustrated in Figure 2(a) and (b) a scene of an relevant activity may be segmented due to a different camera position. As a result the average length of a relevant action frame is shorter. Furthermore, transitions between scenes and advertisements are of different durations.

#### 3.1.2 Tubelets

During this project we attempted to extract accurate spatio-temporal activity proposals using the "tubelet" algorithm presented in [2]. The algorithm returns a sequence of bounding box predictions across consecutive frames, which are referred to as "tubelets." A tubelet describes a sub-volume with a high likelihood of encompassing the action of interest. The idea was to trim the set of tubelet proposals down to a subset of proposals with the greatest likelihood of containing the activity, and then pool the CNN activity predictions for frames belonging to these proposal regions. Since each tubelet provides a series of bounding box predictions, we could dynamically crop and re-size each frame in the video to a constant size to serve as input for the CNN.



(a) (b) (c) (d)

Figure 2: Shot Boundary Detection: (a), (b): a pair of frames immediately before and after a shot boundary. (c): a frame in the same scene as frame (b). (d): The first frame of the next scene.

Unfortunately even when using the most aggressive tubelet extraction settings, the tubelets required on the order of 5 minutes of processing for 100 frame videos and over an hour for videos with beyond 200 frames. As a result, we do not evaluate the CNN performance using the tubelet predictions.

## 3.2. Spatial Stream CNN

The spatial stream CNN operates on individual video frames or sequences of frames, where the RGB channels of each frame are stacked to form a 3L dimensional input channel. Since the frames are 224x244 in size, the resulting input dimensions are 224x224x3L. Through the experiments we found that static image features alone provide enough information to train a competitive activity recognition system.

All of the spatial stream CNN experiments rely on the same CNN architecture: We use shorthand notation $C(d, f, s)$, $d$ filters of spatial size $f \times f$, applied to the input with stride $s$. All pooling layers P(k,s) where $k$=kernel size, $s$=stride use max-pooling. $FC(n)$ is a fully connected layer with $n$ nodes. All normalization layers $N$ are described in Krizhevsky et al [5] and use the same parameters: $n = 5$, $\alpha = 10^{-4}$, and $\beta = 0.75$.

Inputs of size

$$224 \times 224 \times 3$$

$$C(96, 11, 4), RELU, P(3, 2)$$

$$C(256, 5, 1), RELU$$

$$P(3, 2), N$$

$$C(384, 3, 1), RELU$$

$$C(384, 3, 1), RELU$$

$$C(256, 3, 1), RELU, P(3, 2)$$

$$FC(4096), RELU, Dropout$$

$$FC(4096), RELU, Dropout$$

$$FC(101), Softmax$$

The experiment section elaborates on the variety of experiments we conducted using the spatial stream architecture.

## 3.3. Optical Flow CNN

Encouraged by the positive results of [8], we developed a CNN trained on optical flow images. [8] reported high test accuracies achieved by training a CNN on single-frame optical flow and stacking up to 10 consecutive frames. As a first step we decided to extract the optical flow between a pair of frames 30 apart. Extracting the optical flow between frames 30 apart ensures that the average displacement vector field is large. However, since the video frame rate is 24 fps, computing the optical flow between two frames 30 apart could yield inconsistent results for activities with lots of fast movement.

It is important to consider that camera movement could contribute to the optical flow displacement between a pair of frames. This problem is even more likely to occur if the two optical flow frames are not consecutive. We also consider the possibility of pairs of optical flow frames existing in different scenes as segmented by shot detect. However unlikely, if the optical flow bridges between shot detection boundaries, then the optical flow would be exceedingly exaggerated. Thus, we prevent pairs of optical flow frames from existing in separate scenes.

One approach to account for camera motion between a pair of frames is to subtract the mean $dX$ and $dY$ from either optical flow displacement field $d$. Although we did not subtract the mean displacement field, we did take the absolute value of the raw optical flow displacement field. Applying the absolute value to the optical flow implicitly assumes that movement in either direction in the horizontal is equivalent and movement in either direction in the vertical is equivalent. This assumption is valid for some activities such as swimming, but not for others such as rock climbing.

The optical flow was precomputed before training the CNN using the GPU implementation of [9] from OpenCV. To avoid storing the optical flow displacement fields as floats, the horizontal and vertical components are linearly rescaled to a [0,255] range and compressed using JPEG.

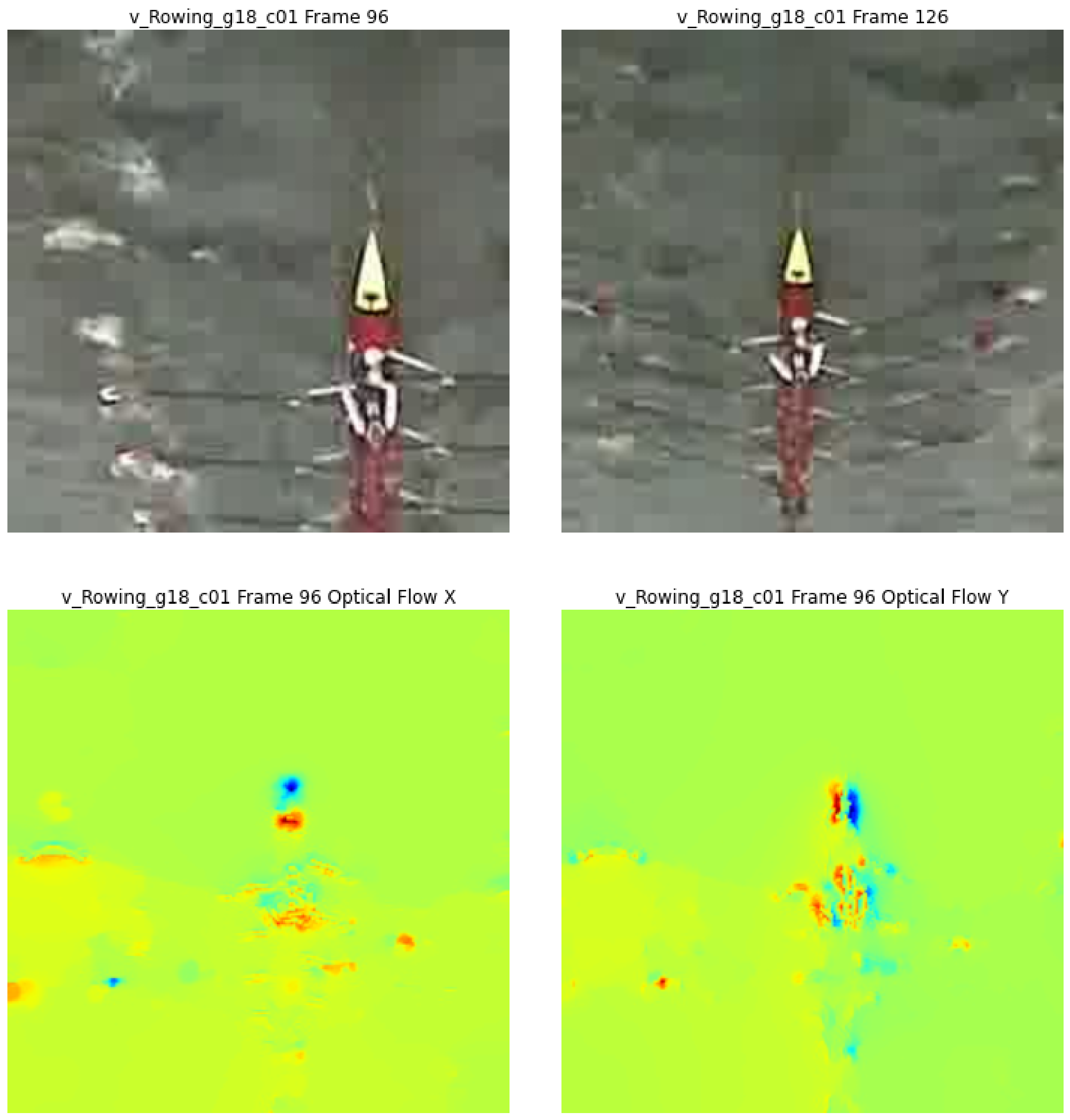

v_Rowing_g18_c01 Frame 96
v_Rowing_g18_c01 Frame 126
v_Rowing_g18_c01 Frame 96 Optical Flow X
v_Rowing_g18_c01 Frame 96 Optical Flow Y

Figure 3: Optical Flow: The above frames illustrate the horizontal and vertical components of the optical flow between a pair of frames 30 frames apart.

## 4. Experiment

### 4.1. Dataset

For data, we use

- THUMOS Challenge 2014 data [3]: The THUMOS challenge provides 4 separate video data sets: (training, validation, background, and testing).
- Training: The training data set is the full UCF101 action data set. It consists of 101 human action categories with 13,320 videos in total. Each category has more than 100 video clips, all of which are temporally trimmed.
- Validation: The validation data set contains 1,000 videos. In general, there is one primary action class shown in each video; however, some videos may include one or more instances from other action classes. Additionally, these videos are not temporally trimmed; the relevant action is not guaranteed to occur throughout the entire video.
- Background: The background data set contains 2,500 videos that are verified to make sure they do not include an instance of any of the 101 action classes.

Each video is relevant to one of the action classes. For instance, the background videos related to the action class Basketball Dunk may show the basketball court when the game is not being played.

- Test: The test data set contains 1,574 temporally untrimmed videos are provided as the test data. Some videos may contain one or multiple instances from one or multiple action classes, and some videos may not include any actions from the 101 classes. A significant portion of the video may not include any particular action, and multiple instances may occur at different time stamps within the video.

## 4.2. Evaluation

We will use Interpolated Average Precision (AP) as the official measure for evaluating the results on each action class. Given a descending-score-rank of videos for the test class c, the AP(c) is computed as:

$$AP(c) = \frac{\sum_{k=1}^{n} P(k) \cdot rel(k)}{\sum_{k=1}^{n} rel(k)}$$

where $n$ is the total number videos, $P(k)$ is the precision at cut-off $k$ of the list, $rel(k)$ is an indicator function equaling to 1 if the video ranked $k$ is a true positive, and to zero otherwise. The denominator is the total number of true positives in the list.

Mean Average Precision (mAP) is the official measure used to evaluate the performance of one run, which is computed as:

$$mAP = \frac{1}{C} \sum_{i=1}^{C} AP(c)$$

We will also report the average accuracy.

## 4.3. Experimental Results

### 4.3.1 Baseline on UCF-101 data set exclusively

We establish the baseline performance of pre-training the CNN on ImageNet and fine-tuning on a portion of the UCF-101 data set. The fine-tuning data set is comprised of the first frame from each UCF-101 video. This process yields 61.7% mAP.

The UCF videos are trimmed, therefore we can be confident that the frames from which we are sampling from actually correspond to the labeled action. Using untrimmed videos is much more challenging - only the whole video is labeled rather than the individual frames. So there is a much greater chance that the frame chosen from random sampling will correspond to noise.

### 4.3.2 Baseline: fine tune on random sample of (TrainVal)

As an alternate approach, we attempted to also sample 5 frames at random and 100 frames at random from the the UCF101 and THUMOS videos. In this case, the CNN is trained on both the UCF-101 and Validation (TrainVal) data sets and tested on the Test data set. Sampling 5 frames at random per video yielded a 31.9% mAP. Sampling 100 frames at random yielded 37.0% mAP. Increasing the sampling rate increased the model performance.

### 4.3.3 Fine tune on (TrainVal) using shot detection

We used shot detection to identify points in the video where the scene changes. Thus every scene in the video has a corresponding length. Our hypothesis was that very short scenes did not correspond to meaningful events in the video.

Our first method for shot detection was to only sample on scenes whose lengths were greater than the median scene length. For 5 frames, this yielded 32.7% mAP, which is an improvement over selecting 5 frames at random. For 100 frames, the shot selection method yielded a similar result (37.0 % mAP).

This suggests that shot detection can help if we can only sample a few frames, and that the benefit saturates when we have the ability to sample from many frames.

We also experiments with other heuristics for detecting significant scenes - for instance, we considered only sampling from scenes who lengths were between the 20th and 80th percentile. The intuition for removing the longest scenes came from observing a few videos whose longest scenes were mostly idle. This performed slightly worse than just selecting only those scenes whose length were greater than the median.

Shot detection is a potentially interesting method to apply if we develop a better heuristic for detecting the critical scenes.

### 4.3.4 Relative performance when tuning on TrainVal data set

Table 1 presents mAP performance of a CNN tested on temporally untrimmed videos. The CNNs were pre-trained on ImageNet followed by (i) fine-tuning on the TrainVal or (ii) keeping the pre-trained weights fixed and only training the last fully connected layer. In either case, we experiment with drop out of 0.5 and 0.9 after the fully connected layers. Although we interrupted training pre-maturely, the results indicate that only training the final fully connected layer yields marginally better performance.

| Training setting | Dropout Ratio | |
|---|---|---|
| | 0.5 | 0.9 |
| fine-tuning | 0.22450 | 0.23529 |
| last layer | 0.2397 | 0.24061 |

Table 1: Using only 20k iterations

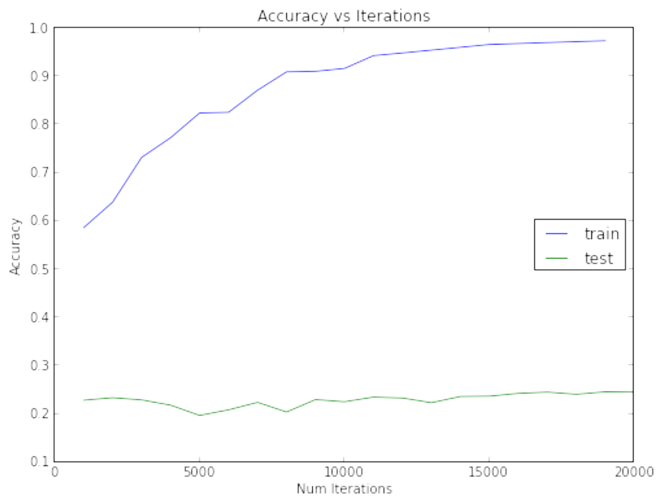


Figure 5: Accuracy plots after fine-tuning with dropout 0.8 and shot detection: Shot Detect 5 with Dropout 0.8: After pre-training on ImageNet, the CNN exhibits severe overfitting on the TrainVal data set. The training and testing data set consists of 5 sampled at random using the shot detection heuristic. Increasing the regularization strength, decreasing the model complexity, and increasing drop-out could potentially decrease the margin between training and testing accuracy.

#### 4.3.5 Optical Flow

For this experiment, from am pair of 3x224x224 frames in a video separated by 30 frames, we computed the optical flow in the $x$ and $y$ direction. Each optical flow is $1 \times 224 \times 224$. We then trained the network by stacking the optical flow images to create a $2 \times 224 \times 224$ image. the pixel values for the optical flow channels were rescaled to have values between 0 and 255.

Training on the optical flow images only yielded 7.3% mAP, which is better than chance, but still worse than training on actual frames.

A potential issue with training the optical flow is that the method does not account for the motion of the camera. It would be useful to consider subtracting the mean displacement from the optical flow frames to mitigate the camera motion to some degree. In the future we would like to capture the incremental motion between multiple frames by stacking the channels of consecutive optical flow frames. If we had more time, we would consider augmenting the optical flow with the original images to try to enhance our results since it does seem that optical flow does have some predictive power.

#### 4.3.6 Stacking Images

Another approach we considered was stacking images. We would choose $K$ frames in order from a particular video and create a $(3 \cdot K) \times 224 \times 224$ image object. This method would have the benefits of optical flow as the network should be able to detect the change in the scene over time, which is what we wanted from optical flow. Moreover, this method would not lose the original information contained in the image.

## 5. Conclusion and Directions for Improvement

We proposed strategies to extend activity recognition to untrimmed videos and evaluated the performance of separate temporal and spatial stream Convolutional Neural Networks. We considered how to accurately sample relevant frames from videos and how to format these frames as input to a CNN. Although promising in theory, stacking the channels of consecutive frames, and computing the optical flow between non-adjacent frames did not lead to significant improvements in performance.

We learned the challenges of trying to classify untrimmed videos, as well as different methods to extract the most critical information from the videos.

We'd like to experiment more with training different network architectures and with investigating heuristics for sampling the most from the frames that are most representative of the action occurring in the video. We would like to extend the optical flow analysis to stacked optical flow channels from consecutive frames. Further, we would like to investigate alternative techniques other than subtracting the mean optical flow displacement to normalize for camera motion between frames. For example, rather than using optical flow, it would be interesting to explicitly use the w-flow described in [1] as input to the CNN. W-flow assumes an affine model to accurately subtract the dominant motion, described by the affine flow vector, from the optical flow vector. The w-flow exaggerates the optical flow of the actors in the foreground and depresses the camera motion.

## References

[1] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2555–2562, June 2013.

[2] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. Snoek. Action localization with tubelets from motion. In

*CVPR - International Conference on Computer Vision and Pattern Recognition*, Columbus, United States, June 2014.

[3] Y.-G. Jiang, J. Liu, A. Roshan 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.

[4] 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.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[6] J. Mathe. Shotdetect. `https://github.com/johmathe/Shotdetect`, 2013.

[7] 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, 2014.

[8] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[9] N. P. T.Brox, A. Bruhn and J. Weickert. High accuracy optical flow estimation based on theory of warping. pages 25–36, 2004.

[10] H. Wang and C. Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision*, Sydney, Australia, 2013.

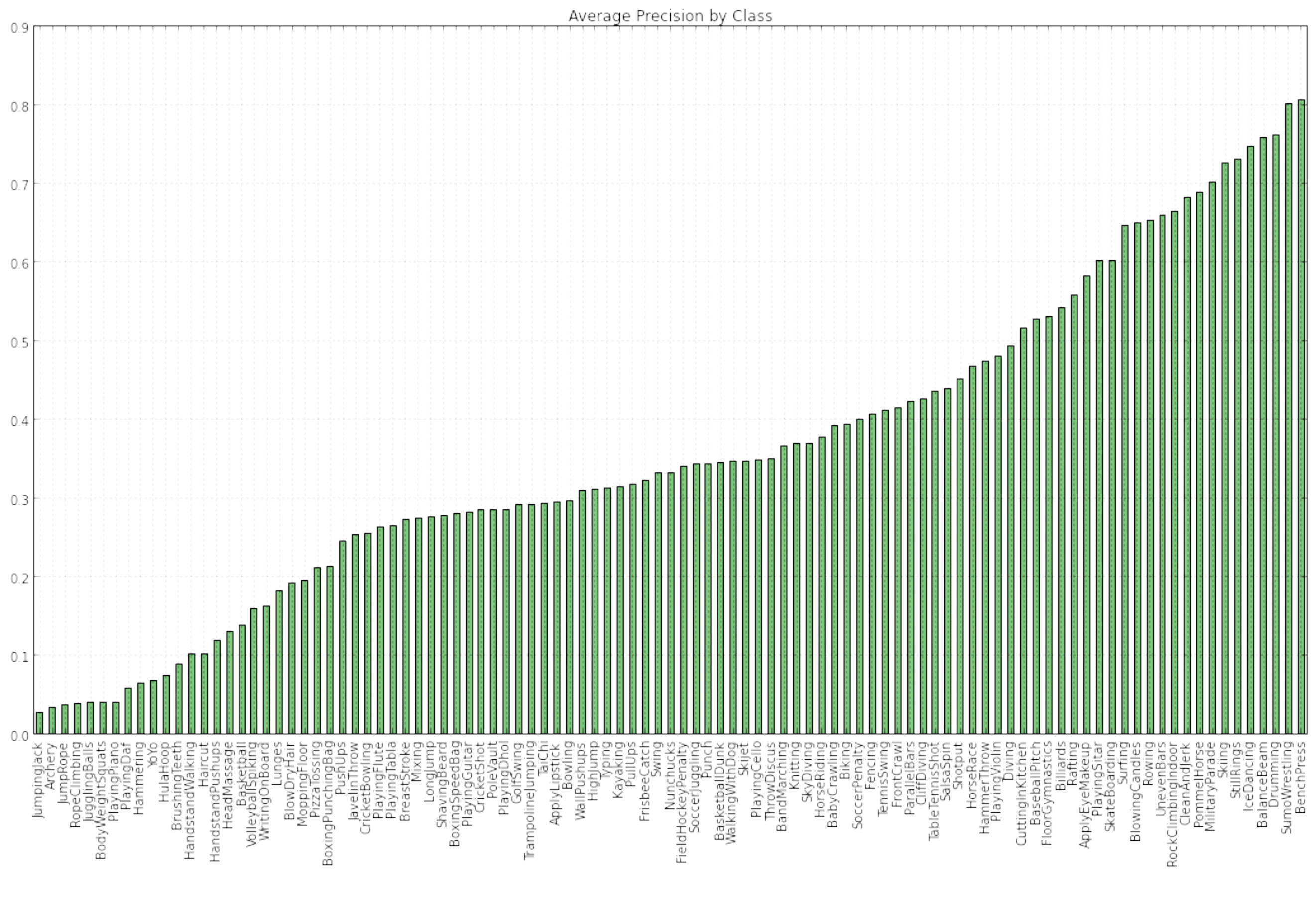


Figure 4: Shot Detect 100 Trial 2: Average Precision by Class