# CS 221 Project: Activity Recognition in Videos

Bryan Anenberg
Stanford University
anenberg@stanford.edu

## Abstract

*This paper presents a system to recognize a wide variety of action categories from videos. The system is trained on the UCF101 action recognition data set of temporally trimmed videos. For each video, the system extracts dense trajectory features and encodes the features as Fisher Vector. A one-vs-rest classifier using SVMs is employed to classify each feature vector. The experimental results demonstrate significant improvement against the baseline.*

## 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.

## 2. UCF101 Data Set

This project exclusively relies upon the UCF101action recognition data set [2] of realistic action videos. The UCF101 data set was collected from YouTube and contains 101 unique action categories. With 13320 videos from 101 action categories, UCF101 gives the largest diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, and illumination conditions. The UCF101 data set presents realistic action categories as opposed to other data sets of staged activities performed by actors. For example, activities performed in videos range from daily-life actions such as "Blow Dry Hair" and "Brushing Teeth" to sports actions such as "Driving" and "Golf Swing." Each video contains an action from a single category and without irrelevant frames. Due to this simplification, the classification algorithm will not need to identify the temporal extent of an activity in a video. It can assume that the entire video comprises a single activity.


Figure 1. A few example action categories from UCF101 data set. The full dataset can be downloaded at http://crcv.ucf.edu/data/UCF101.php

## 3. The Algorithm

This project implements an activity recognition system similar to [7]. At a high level, the system is composed of 5 components.

1. Video-preprocessing
2. Feature Extraction
3. Feature Encoding
4. Dimensionality Reduction
5. Classification

### 3.1. Video-preprocessing

Video-preprocessing consists of rescaling and possibly resampling frames from the raw input videos. Even though the videos in the UCF101 data set have been temporally trimmed, their lengths and resolutions may vary. I rescale the videos so that the dimensions of each frame are $320 \times 240$.

### 3.2. Feature Extraction

Feature extraction is the most complex component of the activity recognition system. I have decided to rely upon the Improved Trajectory Features (IDTF) [7]. This feature extraction technique densely samples feature points in each frame and tracks them in the video based on optical flow. Optical flow uses temporal image brightness variations to recover the image motion at each pixel and to extrapolate the apparent motion of objects and edges in a video. Points are only tracked for a maximum of 15 frames.

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.

- Trajectory: The Trajectory descriptor is a concatenation of normalized displacement vectors. Each trajectory is 30 dimensional.
- HOG: The HOG is constructed by dividing the image into cells and for each cell computing the distribution of intensity gradients or edge directions. The concatenating each of these gradient orientation histograms yields the HOG. Each HOG descriptor is 96 dimensional.
- HOF: The HOF descriptor directly quantizes the orientation of the optical flow vectors. Each HOF is 108 dimensional.
- MBH: The MBH descriptors divide the optical flow into horizontal (MBHx) and vertical components (MBHy), and quantize the derivatives of each component. Both the MBHx and MBHy are 96 dimensional.

Wang's implementation IDTF, yields precise calculations of these descriptors since it identifies and subtracts the effect of camera motion from the frame. It is important to remove camera motion since it generates many irrelevant trajectories. Wang removes the background trajectories, warps the optical flow through a homography that approximates the camera motion, and leverages a human detector to avoid confusing human motion with camera motion. As a result, the motion vectors of the human actors is made independent of the camera motion. The Trajectory, HOF, and MBH descriptors are all compute based on the motion of the warped optical flow. For each video, the IDTF extractor yields thousands of Improved Trajectory Features, where each feature includes a Trajectory, HOG, HOF, MBHx, and MBHy.

### 3.3. Feature Encoding

The IDTF extraction process yields numerous IDTFs for each video. The feature encoding step condenses the local feature (the IDTFs) into a fixed length vector that describes the video as a whole. This procedure is usually accomplished by the Bag-of-words model. This approach treats the video as a document and the image features (IDTFs) as "words." A video is represented as a histogram of word occurrence counts in a vocabulary of video descriptors. The Bag-of-words model must generate a "code book" (equivalent to a word dictionary or vocabulary) by using k-means to cluster all of the image features. The "codewords" in the vocabulary are defined as the centroids learned by the k-means clustering. Each of the original IDTFs is mapped to closest "codeword" and a video is represented as a histogram of these "codeword" occurrences.

Rather than adopting the Bag-of-words approach, I encode the features of each video in terms of a Fisher Vector. Fisher Vectors have recently been show to improve activity classification in videos [6].

The first step in representing a video as a Fisher Vector is to construct a Gaussian Mixture Model. A Gaussian Mixture Model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number ($K$) Gaussian distributions with unknown parameters.

A multidimensional Gaussian distribution is:

$p(x, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$

In our case, each $x$ corresponds to an example descriptor. Each example descriptor can be thought of as drawn from one of $K$ gaussians as given by the multidimensional Gaussian distribution above.

I rely on the Yael library [3] to construct a GMMs from the IDTF descriptors. A separate GMM is constructed for each of the descriptors: Trajectory, HOG, HOF, MBHx, and MBHy. In this experiment I arbitrarily set $K = 120$ since the computational costs of a greater number of modes is too expensive.

Fisher Vector encoding:

The GMM associates each vector $x_i$ to a model $k$ in the mixture with a strength given by the posterior probability:

$q_{ik} = \frac{\exp\left(-\frac{1}{2}(x_i-\mu_k)^T \Sigma_k^{-1}(x_i-\mu_k)\right)}{\sum_{t=1}^{K} \exp\left(-\frac{1}{2}(x_i-\mu_t)^T \Sigma_k^{-1}(x_i-\mu_t)\right)}$

For each of the $K$ modes, the mean and covariance deviation vectors are

$u_{jk} = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ik} \frac{x_{ji}-\mu_{jk}}{\sigma_{jk}}$

$v_{jk} = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ik} \left[\left(\frac{x_{ji}-\mu_{jk}}{\sigma_{jk}}\right)^2 - 1\right]$

$j = 1...D$ for a $D$ dimensional descriptor.

The Fisher vector for a particular descriptor type for a given video is found by stacking the $u_k$ and $v_k$ vectors for each of the $K$ modes of the Gaussian mixtures.

$$FV = \begin{bmatrix} \vdots \\ u_k \\ \vdots \\ v_k \\ \vdots \end{bmatrix},$$

The Fisher Vector leverages the information encoded in the GMM. For each input video (which is described as a set of IDTFs) I construct a Fisher Vector for each of the descriptors: Trajectory, HOG, HOF, MBHx, and MBHy. Each Fisher Vector represents the set of descriptors by the gradient of its log-likelihood with respect to the model parameters of the associated GMM. For each of the $K$ modes of the GMM we measure the mean and covariance deviation of the set of descriptors. The Fisher Vector is the list of the mean and covariance deviations. After building a Fisher Vector for each of the descriptor types, I concatenate the Fisher Vectors together to yield a single Fisher Vector that represents a video.

The Fisher vector encoding process is summarized below:

1. Number of gaussians for (GMM mixtures) set to K=120
2. Randomly sample a subset of 1,500,000 features from the training set to estimate the GMM for each of the 5 descriptor types.
3. The Fisher Vector for each of the 5 descriptor types is power normalized and $l_2$ normalized before being concatenating together to form the full Fisher Vector for the video.
   (a) Power Normalization: $f(z) = sign(z)|z|^\alpha$ where $0 \leq \alpha \leq 1$.
      i. As the number of Gaussians increases, Fisher vectors become sparser. Fewer descriptors are assigned to each Gaussian. When no descriptors are assigned to a particular Gaussian, the entries in the Fisher vector $G_\mu$ and $G_\sigma$ are 0 ($G_\mu$ and $G_\sigma$ are the mean and covariance deviations). Power normalization reduces the relative weight of the Fisher Vector dimensions with greater values to account for those dimensions that correspond to gaussians that fewer descriptors were assigned to.
      ii. We set $\alpha = 0.5$
   (b) L2 Normalization.
      i. Fisher Vector $K$, $K' = \frac{K}{\sqrt{\sum_{i=1}^{n}(K_i)^2}}$
4. Each Fisher Vector is $2DK$ in dimension. The 2 corresponds to $G_\mu$ and $G_\sigma$, the mean and covariance deviations, $D$ to the original dimension of each descriptor, and $K$ to the number of modes in the Gaussian mixture model.
5. The Fisher Vectors for each of the descriptors (HOG, HOF, MBHx, MBHy) are concatenated together. This long concatenated vector is again normalized using $l_2$ normalization.
6. A single video as a whole is represented by the concatenation of the $2DK$ dimensional Fisher Vectors. Since the total sum of the descriptor dimensions is 426. The total Fisher Vector dimensions for a single video is $2 \cdot 120 \cdot 426 = 102240$.

### 3.4. Dimensionality Reduction

The dimension of each Fisher Vector is 102240. This large dimension results in a very slow computation of the One-vs-Rest classifier. As a result, I decided to reduce the dimensions of the Fisher Vector to $1000$ using Principle Component Analysis, which was computed with Singular Value Decomposition (SVD). The training matrix $X$ was of dimensions $m \times n$ where $n = 102240$. SVD decomposes this matrix as follows: $X_{m\times n} = U_{m\times r}\Sigma_{r\times r}(V_{n\times r})^T$.

The elements along the diagonal of the diagonal matrix $\Sigma$ represent the principle direction of the data. The magnitude of each diagonal element is proportional to the amount of variation of the data in that principle direction.

We can transform the Fisher Vector for a single video, $x \in \mathbb{R}^{102240}$ into a reduced vector in the dimension $x' \in \mathbb{R}^{1000}$ with the product: $x' = x^T V$.

For computational reasons, SVD is performed on a sample of the full training data matrix $X$ (approximately 50% of the matrix). Performing SVD on the full matrix $X$ was too computationally expensive.

Scipy.linalg's efficient implementation of singular value decomposition is employed here.

### 3.5. Classification

After reducing the description of a video through Fisher Vector encoding and PCA, a video example $x$ can be represented concisely by a feature vector $\phi(x)$.

#### 3.5.1 One-Vs-Rest Classification

The experiment evaluates the performance of a variety of different classification algorithms. Each classification algorithm operates in the One-vs-Rest scheme. The One-vs-Rest approach involves training a separate binary classifier for each of the 101 classes. The positive examples are the videos whose true label is of class $c$ and the negative examples are the videos whose true label is not class $c$. As mentioned in the discussion of Interpolated Average Precision, each binary classifier assigns a score analogous to $w\cdot\phi(x)$ to each video. To classify a video, each of the 101 binary classifiers assigns a score to the video. The predicted class of the video is the class of the classifier that yields the greatest score. One of the drawbacks of this approach is that each of the 101 binary classifiers views a disproportionately greater fraction of negative examples than positive examples since

the dataset consists of approximately an equal number of videos of each class.

In all cases, the binary classifiers are Support Vector Machines (SVM).

#### 3.5.2 SVM Mathematical Formulation

The support vector machine is a supervised learning algorithm that constructs a hyper-plane to divide the training examples into two classes. In the simple linear classification case, the margin can be defined as $(w \cdot \phi(x))\, y$. In the context of the SVM, we refer to this margin as the functional margin and express the intercept term as $b$ (where $b$ was previously $w_0$, the artificial first term of $w$, with the first term of $\phi(x)$ set as $\phi(x)_0 = 1$). The classifier can be written as $w \cdot \phi(x) + b$ rather than $w \cdot \phi(x)$. The SVM assumes the best hyperplane is the one that has the largest distance (largest margin) to the nearest training data points of any class.

[4][5]

The definition of the SVM classifier is as follows:

Provided training vectors $x^{(i)} \in \mathbb{R}^n$ ($n = 1000$ in our case), and $i = 1, ..., m$ where $m$ is the number of training vectors, and a label vector $y \in \mathbb{R}^n$ such that $y^{(i)} \in \{1, -1\}$. Remember, the positive labels (1) correspond to videos of class $c$ and the negative labels (-1) correspond to all other videos.

The primal problem of the SVM is the following minimization

$\min_{\gamma,w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i$

such that $y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i\,, i = 1, ..., m$

$\xi_i \geq 0, i = 1, ..., m$

The dual problem of the SVM can be written as:

$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j \left\langle x^{(i)}, x^{(j)}\right\rangle$

such that $0 \leq \alpha_i \leq C\,, i = 1, ..., m$

$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

where the $\alpha_i$'s correspond to the Lagrange multipliers used to solve the maximization.

The parameter $C$ (where $C > 0$) controls the relative weight between the goal of minimizing $||w||^2$ and ensuring that most training examples have functional margin of at least 1. Training examples are allowed to have a functional margin less than 1, but incur a cost of $C\xi_i$ if they do so. A small value of $C$ results in the constraint $C\sum_{i=1}^{m} \xi_i$ being ignored and functional margins less than 1 not incurring a large cost. A larger value of $C$ threatens a large cost if the functional margin is less than 1. In this case the classifier becomes more sensitive to noise in the data since the algorithm will even try to force outlier training examples to have a functional margin of at least 1. A smaller choice of $C$ will yield a decision boundary that is more flexible provided non-linearly separable data.

**Kernels** Notice that in the dual formulation of the SVM problem, the feature vectors are expressed in terms of an inner product $\left\langle x^{(i)}, x^{(j)}\right\rangle$. This inner product can more generally be referred to as the kernel function $K(x, x')$ between two feature vectors $x$ and $x'$. In the linear case, the $K(x, x') = \langle x, x'\rangle$. However, it is possible to replace this linear kernel with a non-linear kernel such as:

- radial basis function: $K(x, x') = \exp\left(\gamma||x - x'||^2\right)$ where $\gamma = -\frac{1}{2\sigma^2}$. $\gamma$ is a constant that can be tuned.
- polynomial: $\left(\gamma\left\langle x, x'\right\rangle + r\right)^d$ where $\gamma$ is a constant parameter, $r$ is the linear offset, and $d$ is the polynomial degree.
- sigmoid: $\tanh\left(\gamma\left\langle x, x'\right\rangle + r\right)$ where $\gamma$ is a constant parameter and $r$ is the linear offset.

**Loss Function** The loss function used by default in the derivation of the SVM is the the hinge loss, denoted by $l_1$. Notice, $y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i$ is the hinge loss expression. This expression is equivalent $\max\left\{1 - y^{(i)}\left(w^T x^{(i)} + b\right), 0\right\}$. In the case of the linear kernel SVM, I also experiment using the squared hinge loss $\max\left(\left\{1 - y^{(i)}\left(w^T x^{(i)} + b\right), 0\right\}\right)^2$ denoted by $l_2$.

**Penalty Norm** This experiment use the $l_2$ norm in the penalization.

Scikit-learn's SVM implementation is leveraged in this project.[5]

## 4. Baseline Algorithm

The baseline algorithm provides a lower bound on the expected performance of the presented algorithm. The baseline algorithm is very naive. It reduces the complex video input to the first frame. The input to the baseline algorithm is first frame from every video, resized to $240 \times 320$ pixels. Principle Component Analysis is employed to reduce the dimensions of each video's first frame from 76800 dimensions to a feature vector of size 1000. As in the case of the presented algorithm, a one-vs-rest classifier using SVMs is employed to classify each video.

## 5. Experimental Results

The following experiments were performed using the full UCF101 data set. Specifically, the data set was partitioned into disjoint training and testing sets. The training set consisted of 9535 videos from 101 classes with an average of 94 videos per class, a minimum of 72 from a single class, and a maximum of 121 from another class. The testing set consisted of 3676 videos from 101 classes with an average of 36 videos per class with, a minimum of 23 videos from

a single class, and a maximum of 49 videos from another class.

### 5.1. Performance Metrics

The performance of the system is evaluated using the following metrics:

- First are definitions of statistical measures of performance. For a given class $c$, a binary classifier can either predict a video as positive (a member of class $c$) or negative (not a member of class $c$): $Y_{pred} \in \{1, -1\}$. The true classification of a video is $Y_{true} \in \{1, -1\}$ where $1$ indicates the video is of class $c$ and $-1$ indicates the video is not of class $c$.
  - example x is either class C or not. y_true = C or not. y_pred = C or not.
  - True positives: Correctly identified ($Y_{true} = 1$ and $Y_{pred} = 1$)
  - True negatives: Correctly rejected ($Y_{true} = -1$ and $Y_{pred} = -1$)
  - False positives: Incorrectly identified ($Y_{true} = -1$ and $Y_{pred} = 1$)
  - False negatives: Incorrectly rejected ($Y_{true} = 1$ and $Y_{pred} = -1$)
- Interpolated Average Precision (AP) is calculated for each of the 101 action classes.
  - Provided a class specific descending-score rank of videos, Interpolated Average Precision $AP$ is defined as follows:
    - $AP(c) = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{\sum_{k=1}^{n} rel(k)}$
    - Where $c$ is one of the $101$ classes, $n$ is the total number of 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 descending-score rank of videos is found by considering a binary classifier for the class $c$ trained on the entire training set where positive labels $(1)$ represent videos of the class $c$ and negative labels $(-1)$ represent videos not of the class $c$. The descending-score rank can be found by calculating the distance of each testing example from the binary classification boundary. For instance, in a linear classifier setting with a decision boundary given by $w$. Each video, which is represented by a feature vector $\phi(x)$, is ordered by decreasing score $w \cdot \phi(x)$.
    - The denominator is the expression for $AP$ is the total number of true positives in the list.
- Mean Average Precision (mAP) is the measure used to evaluate the performance of one run.
  - $mAP = \frac{1}{C} \sum_{c=1}^{C} AP(c)$
  - $C$ is the total number of test classes, which is equal to $101$
- Sensitivity (or recall)
  - Sensitivity measures the proportion of actual positives which are correctly identified. This is the ability of the classifier to find all the positive samples.
  - Sensitivity is given by the ratio $\frac{t_p}{t_p+f_n}$ where $t_p$ is the number of true positives and $f_n$ the number of false negatives.
- Specificity
  - Specificity measures the proportion of negatives which are correctly identified.
  - Specificity is given by the ratio $\frac{t_n}{f_p+t_n}$
- Precision score
  - The precision score is a measure of the ability of the classifier to not label as positive a sample that is negative.
  - The precision is the ratio $\frac{t_p}{t_p+f_p}$ where $t_p$ is the number of true positives and $f_p$ is the number of false positives.
- Receiver Operating Characteristic (ROC)
  - The receiver operating characteristic curve is a graphical representation of the performance of a binary classifier as the discrimination threshold is varied. The ROC curve plots the sensitivity against 1-specificity (also known as the false positive rate).
  - The area under the ROC curve is referred to as the AUC and is equal to the probability that a classifier will rank an randomly chosen positive video higher than a randomly chosen negative one.
- Confusion matrix
  - The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix indicates more correct predictions.

- Accuracy Score
  - In the multi-class regime, each of the video is assigned a predicted class. The accuracy score is the percentage of videos that were classified correctly (where the predicted class is equal to the true class).

### 5.2. Evaluating Parameter Performance

The performance of a variety of classification models with different parameter settings are experimentally compared in the Appendix. Table 1 presents the SVM classifiers that yielded the best classification performance when tested on the 3676 video test set. All of the classifiers were trained using the same 9535 videos from 101 classes.

This experiment runs the risk of overfitting on the test set and could have been improved by performing cross-validation. The relative performance of each model would be compared by classifying a separate validation set, different than the test set. After the models are compared based on the performance on the validation set, the final performance of the successful models can be measured relative to the test set.

Notice that the non-linear kernels are given by:

- radial basis function: $K(x, x') = \exp\left(\gamma||x - x'||^2\right)$
- polynomial: $\left(\gamma \langle x, x' \rangle + r\right)^d$
- sigmoid: $\tanh\left(\gamma \langle x, x' \rangle + r\right)$

The experiments only tune the $\gamma$ parameter. $r$ and $d$ remain fixed at $r = 0$ and $d = 3$. Further experiments could involve adjusting these parameters as well.

### 5.3. Performance vs. baseline

The experimental results show that a variety of SVM classifiers performed well classifying the test videos. For example, the linear kernel SVM with $C = 1$, $L_2$ loss, and $L_2$ penalty yielded a 79.25% Mean Average Precision, 77.80% accuracy score, 78.06% Average Precision across the 101 class, and 77.15% Average Sensitivity across the classes. These results demonstrate significant improvement over the performance of the baseline classifier. With the same SVM settings and training set and test set, the baseline achieved 1.38% Mean Average Precision, 1.88% accuracy score, 1.58% average precision, and 2.11% average sensitivity. The discrepancy in performance between the baseline and the presented framework is attributed to the improved feature extraction pipeline. The improved dense trajectories capture the local motion information of the video. They cover the foreground and surrounding context of the activity. The Fisher Vector that represents each video encodes this information. The goal of extracting the detailed motion information the videos, is to produce feature vectors that enable a classifier, such as the linear kernel SVM, the easily distinguish between feature vectors from different classes. The presented algorithm yields feature vectors that enable the classifier to more easily discriminate between classes. [1]

Linear Kernel SVM

| Trial: | $C$ | l | p | mAP | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | $l_1$ | $l_2$ | 78.78% | 77.04% | 77.09% | 76.25% |
| 2 | 1 | $l_2$ | $l_2$ | 79.25% | 77.80% | 78.06% | 77.15% |
| 3 | 10 | $l_1$ | $l_2$ | 78.27% | 76.88% | 77.00% | 76.37% |
| 4 | 10 | $l_2$ | $l_2$ | 78.74% | 77.45% | 77.86% | 76.93% |

Kernel SVMs

| Tr: | Ker | $C$ | $\gamma$ | mAP | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|---|---|---|---|
| 22 | rbf | 50 | 0.01 | 78.74% | 77.15% | 77.18% | 76.34% |
| 31 | rbf | 100 | 0.01 | 79.34% | 77.45% | 77.35% | 76.80% |
| 34 | sig | 100 | 0.01 | 78.71% | 77.01% | 77.05% | 76.22% |
| 41 | rbf | 1000 | 0.001 | 79.28% | 77.31% | 77.25% | 76.68% |
| 43 | sig | 1000 | 0.01 | 78.22% | 76.96% | 77.07% | 76.44% |
| 44 | sig | 1000 | 0.001 | 78.71% | 77.01% | 77.06% | 76.22% |

- l=loss function
- p=penalty function
- Ker=Kernel Function
- rbf=Radial Basis Function
- poly = Polynomial,
- sig= Sigmoid
- mAP= Mean Average Precision
- $A_1$ = Accuracy
- $A_2$ = Average Precision
- $A_3$ = Average Sensitivity

Table 1. The best results from an expansive test of different SVM kernels and parameters. The full results are included in the appendix.

### 5.4. Confusion Matrix

Now we take a closer look at the performance of the linear kernel SVM with $C = 1$, $L_2$ loss, and $L_2$ penalty. Figure 2 presents the Confusion Matrix after performing classification on the test set with this classifier.

The matrix diagonal is very distinct, indicating that the majority of the test examples were classified correctly.

Table 2 illustrates the variance in Interpolated Average Precision of a few example action classes. The

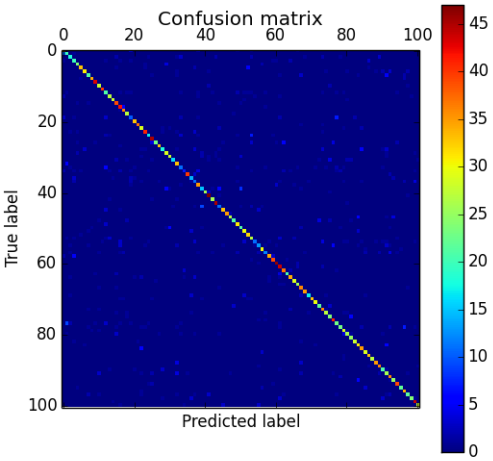


Figure 2. Confusion Matrix illustrating the performance on the full data of 101 action classes.

| class | AP |
|---|---|
| apply eye makeup | 76.2% |
| archery | 37.2% |
| baby crawling | 70.1% |
| basketball | 73.2% |
| basketball dunk | 95.1% |
| biking | 93.7% |
| boxing punching bag | 92.0% |
| diving | 98.2% |
| golf swing | 90.6% |
| hand stand push-ups | 61.2% |
| hand stand walking | 34.4% |
| tennis swing | 27.4% |
| yoyo | 81.2% |

Table 2. The Interpolated Average Precision for a sample of the 101 classes. This represents the linear kernel SVM with $C = 1$, $L_2$ loss, and $L_2$ penalty.

The lower AP scores for some of the action categories such as golf swing and hammer throw could be attributed to the One-vs-Rest classification approach. Action categories such as golf swing and hammer throw involve similar motions motions. "Hand stand push-ups" and "Hand stand walking" feature vectors could be located relatively close to each other in feature space. However, when calculating the binary classification boundary for the "Hand stand push-ups" class, the positive examples are the training videos labeled as "Hand stand push-ups" and the negative example are the rest of the training videos. The presence of the "Hand stand walking" videos in the negative training examples could make it more difficult for the classifier

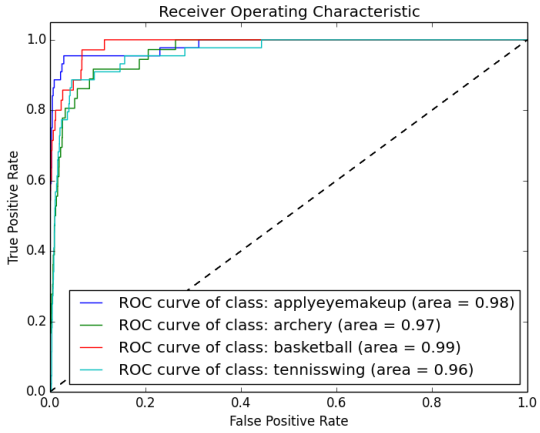


Figure 3. ROC curves for a few of the classes. ROC curves were generated using the linear kernel SVM with $C = 1$, $L_2$ loss, and $L_2$ penalty.

"Hand stand pushup-vs-rest" classifier to identify a discriminating hyperplane than if the "Hand stand walking"videos were not present.

Further reasons for the variance in AP scores could be attributed to the class specific video variance. Some classes such as archery involve videos taken from multiple perspectives where as the apply eye makeup class usually involves videos taken from the same camera angle.

### 5.5. ROC Curve

Even though the interpolated average precision of each of the classes varies widely, the ROC curves and associated AUC values are relatively high. The apply eye makeup and basketball have the ROC curves closest to the left-hand border and the top of the ROC space and the greatest AUC values. This results agrees with the results in the interpolated average precision table where apply eye makeup and basketball presented greater values than the archery and tennis swing classes.

### 5.6. Learning Curve

The positive slope of the cross-validation score curve indicates that the performance of the algorithm can improve if provided more data.

The test and training learning curves are calculated using the full data set (13211 videos). To generate the displayed learning curve, the whole data set is split 10 times into training and test sets. The training set comprises 80% of the data set whereas the test set is 20%. Varying size subsets of the training set are used to train the linear SVM and a score for each training subset size and the test set is computed. The score corresponds to the accuracy. The scores are averaged over the 10 runs for each training subset sizes.

The learning curve is useful to quantify the amount of

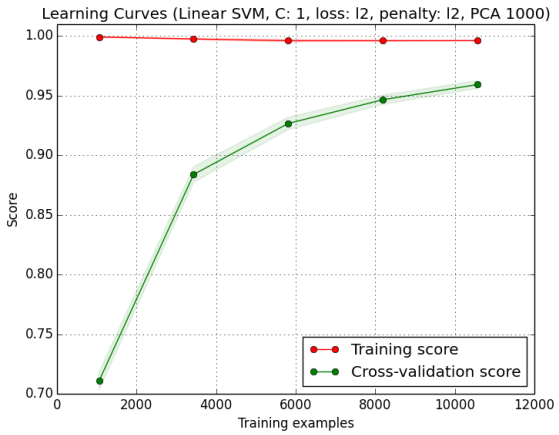


Figure 4. Learning curve for the linear kernel SVM with $C = 1$, $L_2$ loss, and $L_2$ penalty.

performance improvement one can achieve by adding more training data and to determine whether the classifier suffers from bias or variance error.

We can observe from the learning curve that the training curve score is still greater than the validation curve score even after training on the maximum number of samples. This behavior indicates that adding more training samples will likely continue to improve the performance of the classifier.

Also note that validation curve achieves a relatively high accuracy, which indicates that the linear SVM classifier does not suffer from overfitting on the training data set.

## 6. Conclusion

This paper presents an improved an algorithm for classifying the activities in videos across a diverse data set. We show that the performance of the activity classifier can be significantly improved over the baseline by describing the motion information of the video in terms of improved dense trajectories, and encoding this information in Fisher Vectors. Although there are many parameters to tune at different stages of pipeline, the results of this experiment are promising.

## References

[1] Cordelia Schmid Cheng-Lin Liu Heng Wang, Alexander Klaser. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 2013. 5.3

[2] Amir Roshan Zamir Khurram Soomro and Mubarak Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. 2012. 2

[3] Hervé Jégou Matthijs Douze. Yael library. 3.3

[4] Andrew Ng. Cs 229 lecture notes. 3.5.2

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 3.5.2, 3.5.2

[6] Chen Sun and R. Nevatia. Large-scale web video event classification by use of fisher vectors. *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, 2013. 3.3

[7] Heng Wang and C. Schmid. Action recognition with improved trajectories. *IEEE International Conference on Computer Vision*, 2013. 3, 3.2

## 7. Appendix

| Linear Kernel SVM | | | | | | | |
|---|---|---|---|---|---|---|---|
| Trial: | $C$ | l | p | mAP | $A_1$ | $A_2$ | $A_3$ |
| 1 | 1 | $l_1$ | $l_2$ | 78.78% | 77.04% | 77.09% | 76.25% |
| 2 | 1 | $l_2$ | $l_2$ | 79.25% | 77.80% | 78.06% | 77.15% |
| 3 | 10 | $l_1$ | $l_2$ | 78.27% | 76.88% | 77.00% | 76.37% |
| 4 | 10 | $l_2$ | $l_2$ | 78.74% | 77.45% | 77.86% | 76.93% |
| 5 | 50 | $l_1$ | $l_2$ | 77.55% | 76.25% | 76.71% | 75.81% |
| 6 | 50 | $l_2$ | $l_2$ | 77.90% | 76.58% | 77.05% | 76.10% |
| 7 | 100 | $l_1$ | $l_2$ | 77.55% | 76.25% | 76.71% | 75.81% |
| 8 | 100 | $l_2$ | $l_2$ | 77.73% | 76.41% | 76.87% | 75.93% |
| 9 | 1000 | $l_1$ | $l_2$ | 77.55% | 76.25% | 76.71% | 75.81% |
| 10 | 1000 | $l_2$ | $l_2$ | 77.57% | 76.31% | 76.77% | 75.86% |

- l=loss function
- p=penalty function
- Ker=Kernel Function
- rbf=Radial Basis Function
- poly = Polynomial,
- sig= Sigmoid
- mAP= Mean Average Precision
- $A_1$ = Accuracy
- $A_2$ = Average Precision
- $A_3$ = Average Sensitivity

Table 3. All of the reported results have reduced the dimensions of the Fisher Vector to 1000. The scores represent the classification performance on the testing set of 3676 videos after training the One-Vs-Rest SVM on a training set of size 9535. The PCA transformation matrix was calculated using Fisher Vectors randomly selected from the training set.

| Kernel SVMs | | | | | | | |
|---|---|---|---|---|---|---|---|
| Tr: | Ker | $C$ | $\gamma$ | mAP | $A_1$ | $A_2$ | $A_3$ |
| 1 | poly | 1 | 0.01 | 67.24% | 33.73% | 67.71% | 33.15% |
| 2 | poly | 1 | 0.001 | 67.24% | 33.73% | 67.71% | 33.15% |
| 3 | poly | 1 | 0.0001 | 67.01% | 50.22% | 70.49% | 50.10% |
| 4 | rbf | 1 | 0.01 | 77.26% | 72.61% | 73.17% | 71.64% |
| 5 | rbf | 1 | 0.001 | 76.61% | 72.23% | 72.70% | 71.30% |
| 6 | rbf | 1 | 0.0001 | 74.75% | 61.78% | 71.49% | 60.71% |
| 7 | sig | 1 | 0.01 | 77.25% | 72.44% | 73.04% | 71.49% |
| 8 | sig | 1 | 0.001 | 75.95% | 71.71% | 71.78% | 70.72% |
| 9 | sig | 1 | 0.0001 | 74.97% | 60.86% | 70.50% | 59.66% |
| 10 | poly | 10 | 0.01 | 67.24% | 33.73% | 67.71% | 33.15% |
| 11 | poly | 10 | 0.001 | 67.24% | 33.72% | 67.71% | 33.15% |
| 12 | poly | 10 | 0.0001 | 67.35% | 33.24% | 66.78% | 32.63% |
| 13 | rbf | 10 | 0.01 | 77.32% | 73.01% | 73.43% | 72.04% |
| 14 | rbf | 10 | 0.001 | 77.26% | 72.55% | 73.20% | 71.59% |
| 15 | rbf | 10 | 0.0001 | 76.69% | 72.03% | 72.08% | 71.15% |
| 16 | sig | 10 | 0.01 | 77.26% | 72.66% | 73.20% | 71.70% |
| 17 | sig | 10 | 0.001 | 77.26% | 72.50% | 73.13% | 71.54% |
| 18 | sig | 10 | 0.0001 | 75.95% | 71.71% | 71.80% | 70.72% |
| 19 | poly | 50 | 0.01 | 67.24% | 33.73% | 67.71% | 33.15% |
| 20 | poly | 50 | 0.001 | 67.24% | 33.73% | 67.71% | 33.15% |
| 21 | poly | 50 | 0.0001 | 67.33% | 33.22% | 67.66% | 32.63% |
| 22 | rbf | 50 | 0.01 | 78.74% | 77.15% | 77.18% | 76.34% |
| 23 | rbf | 50 | 0.001 | 77.26% | 72.66% | 73.19% | 71.70% |
| 24 | rbf | 50 | 0.0001 | 77.26% | 72.50% | 73.06% | 71.52% |
| 25 | sig | 50 | 0.01 | 77.81% | 75.41% | 75.54% | 74.41% |
| 26 | sig | 50 | 0.001 | 77.30% | 72.66% | 73.22% | 71.71% |
| 27 | sig | 50 | 0.0001 | 77.18% | 72.39% | 72.97% | 71.47% |
| 28 | poly | 100 | 0.01 | 67.24% | 33.73% | 67.71 | 33.15 |
| 29 | poly | 100 | 0.001 | 67.24% | 33.73% | 67.71 | 33.15 |
| 30 | poly | 100 | 0.0001 | 67.30% | 33.71% | 67.71 | 33.14 |
| 31 | rbf | 100 | 0.01 | 79.34% | 77.45% | 77.35 | 76.80 |
| 32 | rbf | 100 | 0.001 | 77.29% | 73.01% | 73.43 | 72.04 |
| 33 | rbf | 100 | 0.0001 | 77.29% | 72.60% | 73.17 | 71.65 |
| 34 | sig | 100 | 0.01 | 78.71% | 77.01% | 77.05 | 76.22 |
| 35 | sig | 100 | 0.001 | 77.25% | 72.66% | 73.20 | 71.70 |
| 36 | sig | 100 | 0.0001 | 77.25% | 72.47% | 73.09 | 71.51 |
| 37 | poly | 1000 | 0.01 | 67.27% | 33.73% | 66.74 | 33.15 |
| 38 | poly | 1000 | 0.001 | 67.24% | 33.73% | 67.71 | 33.15 |
| 39 | poly | 1000 | 0.0001 | 67.24% | 33.73% | 67.71 | 33.15 |
| 40 | rbf | 1000 | 0.01 | 77.88% | 76.28% | 76.68 | 75.81 |
| 41 | rbf | 1000 | 0.001 | 79.28% | 77.31 | 77.25 | 76.68 |
| 42 | rbf | 1000 | 0.0001 | 77.28% | 73.01 | 73.45 | 72.04 |
| 43 | sig | 1000 | 0.01 | 78.22% | 76.96 | 77.07 | 76.44 |
| 44 | sig | 1000 | 0.001 | 78.71 | 77.01 | 77.06 | 76.22 |
| 45 | sig | 1000 | 0.0001 | 77.25 | 72.66 | 73.20 | 71.70 |

Table 4. Continuation of results from the previous table.