Video Copy Detection motivated by Image Classification using Sparse Coding

Aniket Anand Deshmukh
University of Michigan, Ann Arbor
Department of EECS
`aniketde@umich.edu`

Naveen Murthy
University of Michigan, Ann Arbor
Department of EECS
`nnmurthy@umich.edu`

Abstract

In the presence of a vast amount of digital video data, the protection of intellectual property of the creator is of utmost importance. Video copy detection, which is used to find copyright infringements, is also useful in video information retrieval. We propose a novel technique for video copy detection using an image classification framework and sparse coding. The underlying image classification framework is based on non-negative sparse coding, low-rank and sparse matrix decomposition techniques along with Spatial Pyramid Matching (LR-Sc+SPM). SIFT features from each image are encoded using non-negative sparse coding. Using Spatial Pyramid Matching + Max pooling, we capture the spatial relations between the sparse codes. Low-rank and sparse matrix decomposition is then used to exploit correlations and dissimilarities between images of the same class. Extending this to video copy detection, we create a framework where scene change detection is performed using edge maps. The resulting scenes are divided into classes and, using the same image classification framework, a multi-class linear SVM is trained. We evaluate our proposed algorithm against two state-of-the-art techniques for video copy detection with accuracy and computational time being the metrics.

1 Introduction

In the last few years, many researchers have used bag-of-visual words (BoW) model for image classification [1]. There is a loss of spatial information in BoW models and to avoid that, Lazebnik et al. [2] proposed the spatial pyramid matching (SPM) which has proven very effective and has been widely used. Recently, Yang et al. [3] proposed an extension of the SPM approach by leveraging sparse coding and achieved state-of-the-art performance for image classification where only one type of local feature (SIFT) is used. This method can automatically learn the optimal codebook and search for the optimal coding weights for each local feature. After this, max pooling along with SPM is used to get the feature representation of images. Inspired by this, Wang et al. [4] proposed the use of locality to constrain the sparse coding process which can be computed faster and yields better performance. Motivated by these two methods, Zhang et al. [5] (original paper implemented by us) propose non-negative sparse coding, low-rank and sparse matrix decomposition techniques (LR-Sc+SPM) which is a little better in terms of accuracy but computationally more intensive. We will describe all algorithms and results of this method in the next few sections.

We extend this image classification framework to the domain of video copy detection. The explosive growth of technology means the number of videos being uploaded onto the internet has exponentially increased. Keeping track of this is a Herculean task and is a critical issue. Video Copy Detection is the technique of detecting videos that have been illegally copied from other original content, with a view to protecting the creator's intellectual property. For the protection of copyright, existing methods usually extract features called fingerprints from videos in the database. Several video fingerprinting methods have been proposed till now. Oostveen et al. in [6] proposed a spatio-temporal fingerprint based on the differential of luminance of partitioned grids in spatial and temporal regions. This method, while performing quite well for pristine test videos, does not work that well for camera queries or non linear transformations. S. Lee et. al. [7] proposed robust but low-complexity local feature-based algorithms for video ngerprinting. But these gradient based features are very sensitive to noise, and therefore not robust to artifacts which affect the high frequency content of the video. Shanmuga Vadivel et al., [8] proposed Perceptual Distance Metric (PDM) between two videos using Structural Similarity Index (SSIM). This method works really well for almost all sorts of transformations but suffers from high computational time. To solve the issue of high computational time, we propose a new fingerprinting system motivated by image classification. We will discuss more about this system in later sections.

The report is organized as follows. Section 2 contains a well-defined hypothesis of our original problem statement for image classification (excluding the extension). Section 3 comprises of the key algorithms used for Image Classification, while Section 4 includes the corresponding experimental results. Section 5 explains the steps involved in Video Copy Detection, which is our extension, and the results obtained using our proposed method. Section 6 contains our concluding remarks while Section 7 details a few of the challenges that we faced.

2 Research Hypothesis

Table 1: Algorithms and their accuracy [5]

Algorithm	15 training	30 training
KSPM [2]	56.40	64.40
KCSPM [9]	-	64.14
NBNN [10]	65.00	70.40
SVM-KNN [11]	59.10	66.20
KMTJSRC [12]	65.00	-
ScSPM [3]	67.00	73.20
LLC [4]	65.43	73.44
LR-SC+SPM (this paper) [5]	69.58	75.68

The LR-Sc+SPM image classification algorithm achieves state-of-the-art performance on several public datasets. The dataset that we are specifically going to concentrate on is "Caltech-101". The Caltech-101 dataset contains 101 classes with high intra-class appearance shape variability. The number of images per category varies from 31 to 800 images. A similar experimental setup has been followed in Refs. [12, 3]. Table 1 gives the exact comparison of results of different image classification algorithms used on the Caltech 101 dataset. So, LR-Sc+SPM proposed in Ref. [5], outperforms other algorithms. It achieves an accuracy of 69.68% for 15 training images and 75.68% for 30 training images which is 4.1% better for 15 training images and 2.2% better for 30 training images when compared to Locality-constrained Linear Coding (LLC) [4].

3 Algorithms used in Image Classification

The block diagram of the image classification framework used in [5] is given in Fig. 1. In this section, we will discuss all algorithms that we are going to use. Let us define the variables and experimental setup here. Let $X = [x_1, x_2, ..., x_N](x_i \in R^{(D\times1)}$ be the set of N local image descriptors), $V = [v_1, v_2, ...v_K](v_i \in R^{(D\times1)})$ cluster centers to be learned and let $U = [u_1, u_2, ...u_N](u_i \in R^{(K\times1)})$ be the sparse representations of the descriptors. In our setup, X contains the 128-dimensional descriptors obtained after SIFT feature extraction.

We solve Vector Quantization (using k-means) using following optimization problem:

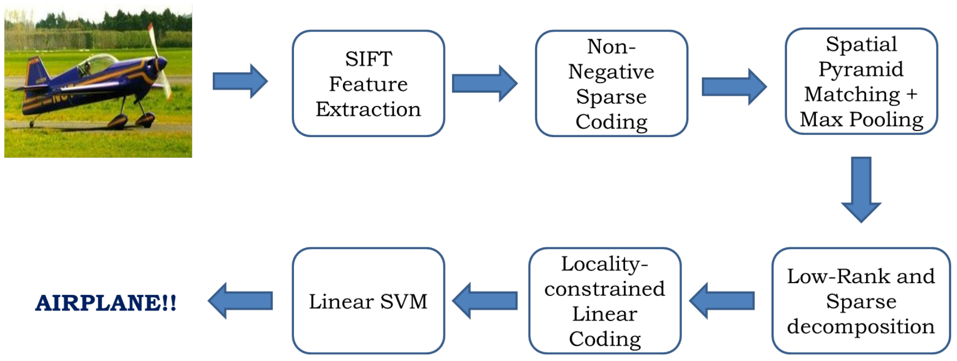

SIFT Feature Extraction
Non-Negative Sparse Coding
Spatial Pyramid Matching + Max Pooling
Low-Rank and Sparse decomposition
Locality-constrained Linear Coding
Linear SVM
AIRPLANE!!

Figure 1: Block diagram of the Image Classification system in [5]

$$\min_{U,V} \sum_{n=1}^{N} \| x_n - Vu_n \|^2, \ s.t.\ card(u_n) = 1, |u_n| = 1, u_n \geq 0, \forall n. \tag{1}$$

Where $card(u_n) = 1$ is the cardinality constraint and is too strict because each local feature can be assigned to only one visual word. This is a constraint which has a high quantization error and to reduce this loss, we allow multiple bases in V to represent X. We relax the cardinality constraint using l_1 norm regularization which turns this problem into sparse coding, where λ is regularization parameter:

$$\min_{U,V} \sum_{n=1}^{N} \| x_n - Vu_n \|^2 + \lambda \| u_n \|_1, \ s.t.\ \| v_k \|^2 \leq 1, \forall k. \tag{2}$$

Zhang et al. [5] say there is a problem with this sparse coding plus max pooling strategy. When we do max pooling, zero coefficients will be chosen over negative coefficients. But coefficients of sparse coding which are close to zero indicate that the corresponding bases have no (or very small) influence. That means some useful information may have been lost which hinders the final classification performance. There is an ambiguity here because we do max pooling of *absolute* values of sparse coefficients. So positive or negative does not matter here and Yang et al. [3] who proposed max pooling explicitly mention that the sign does not matter and which makes more sense. But we continue with what the authors in [5] say. To reduce the loss of information, they propose non negative sparse coding which solves

$$\min_{U,V} \sum_{n=1}^{N} \| x_n - Vu_n \|^2 + \lambda \| u_n \|_1 \ \ s.t.\ \| v_k \|^2 \leq 1, u_n \geq 0, \forall n, k. \tag{3}$$

This optimization problem is solved in Section 3.2. After non-negative sparse coding, each image is represented as a single feature using Spatial Pyramid Matching + Multi-scale Max Pooling, explained in Section 3.3. Then, we exploit the correlations and dissimilarities between images of the same class, and decompose the feature matrix of the entire database into a low-rank and sparse matrix, explained in Section 3.5. The final step is classification using a multi-class linear SVM, given in Section 3.7.

3.1 Scale Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform is one of the most widely used feature descriptors [13]. The key steps of SIFT can be explained as follows:

- SIFT searches over all locations and image scales to find extrema (potential keypoints).
- At each location and scale, keypoints are chosen depending on how stable they are.

- Orientations are assigned to the keypoint locations depending on the local image gradients. By considering all operations relative to the assigned orientation, scale, location, invariance to these transformations is achieved.
- The local image gradients are measured at the selected scale, for each keypoint, and are converted to an appropriate representation. This is 128-dimensional and is normalized to achieve illumination invariance.

The authors in [5] use the implementation provided by Lazebnik et al. in [2]. We used the same implementation of SIFT in our experiments. In our experimental set-up, we densely compute SIFT descriptors on overlapping 16×16 pixels with an overlap of 6 pixels.

3.2 Efficient Sparse Coding

3.2.1 Introduction

The goal of Sparse coding is to represent input vectors approximately as a linear combination of small number of unknown basis vectors. Sparse coding represents each input vector $x_i \in R^{(D \times 1)}$, using sparse vector coefficients or weights $u_i \in R^{(K \times 1)}$ and basis vectors $v_1, v_2 ... v_K \in R^{(D \times 1)}$. To write equations more concisely in Matrix form consider a training set of N input vectors (with each column as an input vector) $X \in R^{(D \times N)}$, their (unknown) corresponding basis matrix (with each column as a basis vector) $V \in R^{(D \times K)}$ and sparse coefficient matrix (with each column as a coefficient vector) $U \in R^{(K \times N)}$. Then the best estimate of the bases and coefficient is the solution to optimization problem given in Eqn. 2 or 3.

Here, L_1 regularization is chosen because we want a sparse representation. The optimization problem in Eqn. 3 is convex in V - bases (while holding U fixed) and convex in U - coefficients (while holding V fixed), but not convex in both simultaneously. So, to solve this problem, the idea is to iteratively optimize the objective function in Eqn. 3 by alternatively optimizing with respect to V and U. Solving for U is a regularized least square problem and can be solved using generic QP solvers like CVX. However, these are very slow and computationally expensive. Solving for V is a constrained least square problem and can be solved using generic convex optimization solvers or gradient descent. Authors in [14] derive and solve the Lagrange dual for solving constrained least square problem and show that it is way more efficient than existing methods.

3.2.2 Solving for U

This algorithm solves for U (while holding V fixed), which is an L_1 regularized least square problem:

$$\min_U \sum_{n=1}^{N} \| x_n - V u_n \|^2 + \lambda \| u_n \|_1 \; s.t. u_n \geq 0, \forall n. \tag{4}$$

We solve this using A Fast Iterative Shrinkage-Thresholding Algorithm (FISTA). We use FISTA with constant step size as in [15] and we tested it using a small database which works fine. Using a trial and error approach, we chose $\lambda = 0.15$. But we found a Matlab-executable "C" implementation for solving non-negative sparse coding problems (using Least Angle Regression algorithm [16]) which was extremely helpful as it was really fast. We integrated this "C" code in our module.

3.2.3 Lagrange dual to solve for V

In this case, the optimization problem is reduced to

$$\min_V \| X - VU \|_F^2, \; s.t. \sum_i |V_{i,j}^2| \; \leq c \, \forall j = 1, 2, ... n \tag{5}$$

This can be solved using a Lagrange dual. Let us take a Lagrangian first:

$$L(B, \overrightarrow{\lambda}) = trace((X - VU)^T (X - VU)) + \sum_{j=1}^{n} \lambda_j (\sum_{i=1}^{n} V_{(i,j)}^2 - c), \; where \; \lambda_j \geq 0. \tag{6}$$

Analytically minimizing over V gives us,

$$D(\overrightarrow{\lambda}) = \min_{V} L(V, \overrightarrow{\lambda}) = trace((X^T X - XU^T(UU^T + \Lambda)^{-1}(XU^T)^T - c\Lambda), \; where\, \Lambda = diag(\overrightarrow{\lambda}) \tag{7}$$

Now, Eqn. 7 can be optimized using Newton's method or conjugate gradient. Solving this, we get the optimal bases:

$$V^T = (UU^T + \Lambda)^{-1}(XU^T)^T. \tag{8}$$

We use Prof. Honglak Lee's toolbox for solving Eqn. 5 [14].

3.3 Spatial Pyramid Matching (SPM)

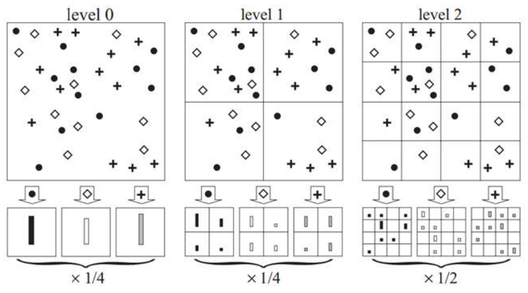
The popular Bag-of-Words (BoW) method represents an image as an orderless collection of features (SIFT features in our case). Since these methods disregard all information about the spatial layout of the features, they are incapable of describing an object through characteristics such as shape. In [2], Lazebnik et al. proposed Spatial Pyramid Matching (SPM), a method which incorporated the spatial correspondence of features. In this technique, we partition the image into increasingly fine sub-regions and compute histograms of local features found inside each sub-region. This is illustrated using an example in Fig. 2a. The sample image has three feature types, indicated by circles, diamonds, and crosses. We sub-divide the image at three levels of resolution, and at each level, we construct histograms of local features that fall in each sub-region. The final feature representation of the image is obtained as a concatenation of the local histograms.

We follow a similar procedure in our Image Classification algorithm. Each image is analysed at three scales (1,2,4), i.e., at scale i, the image is divided into 2^i regions. In each sub-region, we count the occurrences of local features and construct a weighted histogram, as explained in Fig. 2a. However, instead of taking a histogram, we implement Max Pooling, which was proposed by Yang et al. in [3]. The need for pooling is discussed in detail in the next subsection.

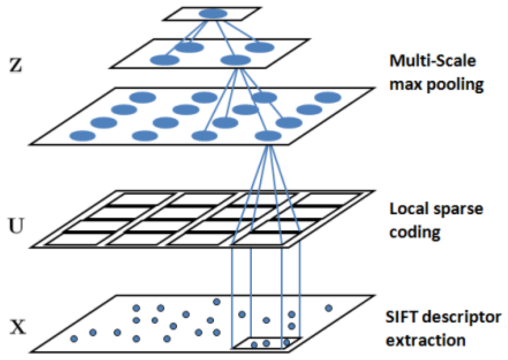
3.4 Multi-scale Max Pooling

Modern vision algorithms have a spatial pooling step, which combines the responses of local feature detectors of a region into a single statistic. This represents the overall feature response of the region of interest. Pooling is performed to achieve invariance to image transformations, compact representations and robustness to noise [17]. A histogram is one such example of a pooling operation, which was used by Lazebnik et al. in [2] for scene categorization.

Recall, from Section 3.2, that $U = [u_1, u_2, ...u_N] \in R^{(K \times N)}$ contains the sparse representations of all descriptors. Each column of U contains the sparse representation of a feature while each row



(a)



(b)

Figure 2: (a) Spatial Pyramid Matching - an example [2] (b) Multi-scale Max Pooling [3]

corresponds to the responses of all feature descriptors to one specific codeword in the dictionary V. The statistic obtained after pooling is

$$Z = f\,(U) \tag{9}$$

where $f()$ is the pooling function. A typically used pooling function is the *averaging* function, which yields the histogram. In [3], Yang et al. discuss the *max* pooling function as

$$z_j = max\left\{ \left|u_{j1}\right|, \left|u_{j2}\right| ..., \left|u_{jM}\right| \right\} \tag{10}$$

where z_j is the j^{th} element of the statistic Z, u_{ij} is the element at the i^{th} row and j^{th} column of U, and M is the number of local descriptors in the region. *Here, z_j could be interpreted as the maximum response of the j^{th} codeword among all local descriptors in the region.* The max pooling procedure is established by previous research in [18] and used successfully for image classification in [3]. An illustration of max pooling is shown in Fig. 2b with $f()$ being the max pooling function.

We wrote our own code for both Spatial Pyramid Matching and Multi-scale Max Pooling based on details provided in [2] and [3] respectively. We worked with a dictionary V of size 1024 codewords, and hence each vector in U is of dimension $u_m \in R^{(1024 \times 1)}$. Max pooling yields a feature vector of dimension 1024×1 for each sub-region. Considering all scales, we obtain a feature matrix $\in R^{(1024 \times 21)}$, since we have a total of $1 + 4 + 16 = 21$ regions across three scales. Converting this matrix to a vector, we represent each image in the database with a vector of dimensions 21504×1. Thus, the entire training set can be represented by a Bag-of-Words (BoW) matrix $H \in R^{21504 \times T}$ where T denotes the total number of training images.

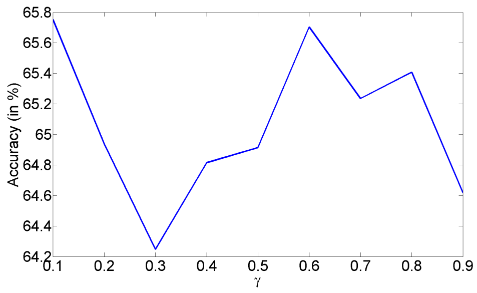
3.5 Sparse and Low Matrix Decomposition

The authors exploit the fact that there exist correlated (or common) items and specic (or noisy) items among images of the same class [5]. For instance, the similarities between images from the same class are characteristics that are common to that object while the clutter/noise in the scene differs from image to image. The correlations are captured in a low-rank matrix and the dissimilarities in a sparse noisy matrix. The authors propose the decomposition of the BoW matrix H as

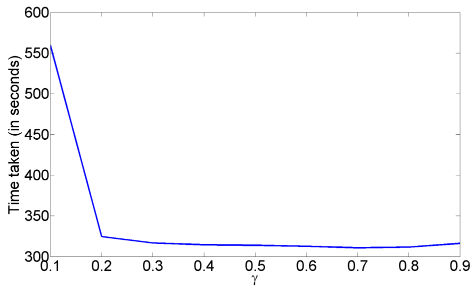
$$H_i = L_i + N_i \tag{11}$$

where H_i is the BoW representation of all images in class i, with L_i and N_i being the low-rank and noise matrices of class i respectively. This problem can be solved by optimizing

$$\min_{L_i, N_i} rank(L_i) + \gamma \left\| N_i \right\|_0 \tag{12}$$



(a)



(b)

Figure 3: Effect of parameter γ in Low-rank and Sparse decomposition (a) Accuracy vs γ (b) Time taken vs γ

$$s.t.\ H_i = L_i + N_i$$

where $\gamma > 0$ is a trade-off between low-rank and sparsity. Further, it is shown in [19] that, under certain conditions, it is equivalent to solving

$$\min_{L_i, N_i} \|L_i\|_* + \gamma \|N_i\|_1 \tag{13}$$

$$s.t.\ H_i = L_i + N_i$$

Here, $\| \ \|_*$ is the nuclear sum, defined as the sum of all singular values. The authors use the Augmented Lagrange Multiplier (ALM) [20] to solve this problem. We used the software implementation of ALM provided in [20].

The parameter γ is the weight on the sparsity regularizer in Eqn. 13. To find the optimal value of γ, we varied γ between 0.1 and 0.9 in steps of 0.1. Beyond 0.9, the sparsity matrix consisted of all zeros and so we stopped at 0.9. Fig. 3 shows the effects of γ on classification accuracy and time taken. In Fig. 3a, we see that we obtain a high accuracy for γ = 0.1, but the corresponding time taken is large (Refer Fig. 3b. We can see that at about γ = 0.6, we obtain a reasonably good accuracy and a lower value of time taken. Thus, γ = 0.6 looks like a good trade-off between sparsity of N_i and the low-rank nature of L_i.

3.6 Locality-constrained Linear Coding (LLC)

Locality-constrained Linear Coding (LLC) was proposed by Wang et al. in [4] when they used it in their image classification algorithm. As illustrated in Fig. 14, LLC, like Sparse Coding (SC), uses multiple bases for representation. SC does not have a notion of locality and similar patches might be represented using quite a different set of bases. However, in LLC, an explicit locality adaptor ensures that only local bases are used for representation and hence, similar patches are represented in an almost similar way. We describe the implementation of LLC below.

We use the low-rank and sparse matrices obtained in Section 3.5 to encode the histogram information of images H. We compute the low-rank and sparse matrices for all M classes to obtain $L = [L_1, L_2, ..., L_M]$ and $N = [N_1, N_2, ..., N_M]$. These are combined to form $B = [L, N]$, which are the new bases of coding. If one image belongs to the i^{th} class, it will probably be reconstructed by vectors of the i^{th} low-rank matrix L_i and sparse matrix N_i instead of vectors of other classes. In LLC, we reconstruct the BoW representation of images by using the new bases in B. Formally, LLC solves

$$\min_{c_p} \sum_{p=1}^{P} \|h_p - Bc_p\|^2 + \beta \left\| d_p \bigodot c_p \right\|^2 \tag{14}$$

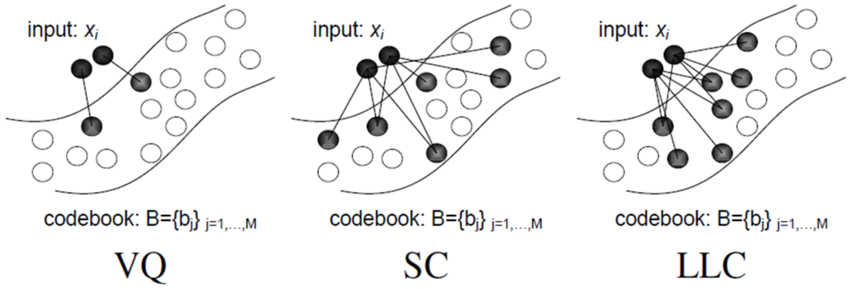

input: x_i
codebook: B={b_j} j=1,...,M
VQ
input: x_i
codebook: B={b_j} j=1,...,M
SC
input: x_i
codebook: B={b_j} j=1,...,M
LLC

Figure 4: Comparison between VQ, SC and LLC [4]

$$s.t. 1^T c_p = 1, \forall p$$

where $\bigodot$ represents element-wise multiplication and d_p is a distance scoring function defined as

$$d_p = exp(\frac{dist(h_p, B)}{\sigma}) \tag{15}$$

Here, $dist(h_p, B)$ is a column vector containing the Euclidean distances between h_p and each base in B. σ is a weighting term for the scoring function.

We implemented LLC using the analytical solution in [4], given by

$$\tilde{c_i} = (C_i \, + \, \lambda \, diag(d)) \setminus 1 \tag{16}$$

$$c_i = \tilde{c_i} \, / \, 1^T \tilde{c_i} \tag{17}$$

where $C_i = \left(B \; - 1 \, h_i^T\right) \left(B \; - 1 \, h_i^T\right)^T$ is the data covariance matrix. We chose a parameter of β = 0.0001.

Another difference between SC and LLC is that the l_1 norm regularizer is replaced by the locality adaptor. As explained by Yu e al. in [21], locality is more essential than sparsity, since locality must lead to sparsity but sparsity need not necessarily lead to locality. This is shown in Fig. 4, where SC leads to spread out bases while LLC ensures local bases.

3.7 Classification

3.7.1 Linear SVM

The authors of our chosen paper use the multi-class linear SVM provided by Yang et al. [3] and hence, we have also used the implementation provided by Yang et al. The authors take a one-against-all strategy to train M binary linear SVMs and obtain a multi-class linear SVM for M classes. Each binary linear SVM has a squared hinge loss function and is trained using LBFGS [3], which is an optimization algorithm.

4 Simulations and Results for Image Classification

4.1 Caltech-101 Dataset

Caltech-101 is a widely used database for object classification. It has 101 categories of objects and about 31-800 images in each class [22]. The size of each image is roughly 300 $\times$ 200 pixels.

Figure 5: Sample images taken from Caltech-101

Algorithm	Original Paper[5]	Our Implementation
LR-Sc+SPM	75.7 ± 0.9	70.8 ± 0.8

(a)

Training Images	Accuracy
5 images	60.5 ± 2.1
10 images	67.1 ± 0.9

(b)

Table 2: Classification accuracy for (a) Entire dataset (all classes) (b) Reduced dataset (18 classes and 25 training images). All accuracies are given in terms of percentage.

4.2 Experiment 1 : Entire Dataset

We used all 101 categories in this setup. 30 images from each class were used for training, and the first 30 images were chosen owing to considerable time taken if we randomized this. For testing, we considered 10 images from each class randomly (lesser for a few classes which do not have 10 images for testing). The classification accuracy obtained by us is shown in Table 2a and is compared with that obtained in [5]. The disparity in results could be attributed to a factor: The authors in [5] choose 30 images randomly in each trial and train an SVM. However, due to the enormous time taken, we chose the first 30 images of each class and trained an SVM only once. Only the testing images were randomized in each trial and classified.

4.3 Experiment 1 : Reduced Dataset

We created a reduced database of 18 classes and 25 images from each class. Using a portion of the dataset for training, we evaluated the system using the remaining images from the reduced dataset. The effect of number of training images on classification accuracy is shown in Table 2b.

5 Video Copy Detection

Copyright infringement or piracy is very important for the motion picture industry. Also, methods used for video copy detection can also be used for video information retrieval - for instance, given a short video query, we may need to identify which video it belongs to in the database. In this section, we propose a novel video copy detection system motivated by the Image Classification framework discussed in the previous section. Roughly speaking, we can classify Video Copy Detection system into three classes. In the first phase, we do pre-processing of videos and scene change analysis. In the second phase, we use the image classification framework and train an SVM using videos from the database . In the third phase, we try to find the closest match for the video query. The block diagram of the proposed Video Copy Detection system is shown in Fig. 6.

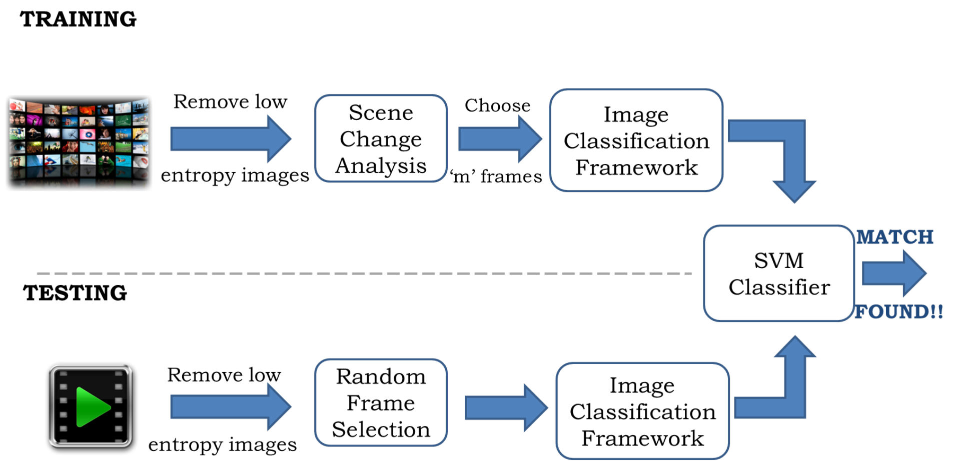
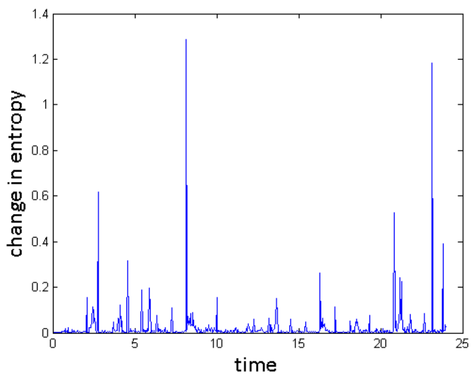
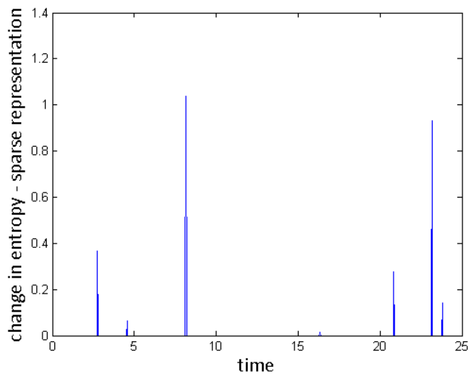


Figure 6: Block diagram of the proposed Video Copy Detection system



(a)



(b)

Figure 7: Change in entropy (a) Original and (b) Sparse representation

5.1 Scene Change Detection

Inspired by Image Classification, we divide the video into different classes of images using scene change analysis. From one scene change to the next scene change, we have one class of images. Thus, we divide each video into such image classes and select few frames from each class for training. Before scene change detection, we delete low entropy images because we do not want to match background images.

5.1.1 Entropy based scene change detection

We calculate the entropy of each image in a given video and then take difference of entropy. The plot of difference in entropy is shown in Fig. 7a. We then find the sparse estimate for this difference in entropy which will give us the scene change locations, as illustrated in Fig. 7b. This works well for many videos but for a few videos where there is a gradual change in entropy or overlap of frames at the location of the scene change, this method fails.

5.1.2 Edge map based scene change detection

Instead of entropy, we use edge detection for scene change detection. First we calculate the edge map using Sobel edge detection. Then we divide this edge map into 32 blocks and calculate the mean of each block. If the mean of a particular block from one image to the next image changes beyond some threshold, then we say that that block has changed. If the number of block changes crosses some threshold (equal to 4-5) then we say it is a scene change. Loosely speaking, if scene change analysis gives p classes per video (on an average) and assuming a total of q videos, there are pq overall classes for the database.

5.2 Image Classification Framework

5.2.1 Training

We treat each scene from each video as a different class and we select a maximum of 30 frames from each class/scene. These frames are taken through a similar procedure as detailed in the image classification framework in Fig. 1, and finally used to train a multi-class SVM.

5.2.2 Testing

First, we delete all low entropy images from query video. We do not perform scene change analysis since we desire a real-time output. Then we choose 10 frames from the rest of the video. We classify these 10 images using the trained SVM and see which class it belongs to. Then we find which videos these output classes belong to. We perform majority voting to get the matched video in the dataset.

Table 3: Algorithms and their accuracy

Video Transformation	Oostveen [6]	PDM [8]	Proposed Method
Pristine Queries	19/19	19/19	19/19
Noise added (Gaussian, $\mu = 0, \sigma^2 = 0.02$)	19/19	19/19	18/19
Camera Queries	7/19	19/19	18/19
Average time	2.72 sec	705 sec	25.44 sec

5.3 Results

We evaluated performance of proposed method and compared it against two existing methods. We created a databse of 19 videos, each of length 25 seconds. We then removed low entropy images from all those videos and then divided videos into different classes based on scene change detection. We then considered a maximum of 30 images for each class and trained the system using the Image Classification framework built in Section 3.

We created different queries as follows. A pristine query is created by extracting 3 seconds of clip from each video. There are no transformations applied in case of the pristine query. Noisy queries are created by adding Gaussian noise to these 3 seconds of pristine queries. Camera queries are created by recording video using a digital camera.

As we can see in Table 3, all three methods perform really well for pristine queries. But for noisy queries, the proposed method does not perform as well as the other two. However, for camera queries, the proposed method outperforms the method proposed by Oostveen et al. [6] and performs equally well as PDM [8]. The major advantage of the proposed method over PDM is computational time. PDM uses SSIM [23] which needs to compute the covariance between two images. So while sweeping through all videos in the database, PDM has to do all these calculation when the query is passed. But in our case, computational time just depends on number of images that we select from the query.

6 Conclusion

In this project we first implemented image classification framework as in [5]. The original authors have used spatial pyramid matching + max pooling approach based on SIFT sparse codes for image classification. Before spatial pyramid matching, the method uses non-negative sparse coding instead of traditional vector quantization. The authors argue that there is information loss when we do max pooling on top of sparse coding but there is no such information loss for non-negative sparse coding. However, according to the original definition of max pooling in [3], max pooling would give us the desired result even with conventional sparse coding since we take absolute values of sparse codes. This was the most confusing part in the entire paper.

To summarize, authors get BoW representation of images using non-negative sparse coding, SPM and max pooling. Authors have further proposed low-rank and sparse matrix decomposition technique to get more discriminative bases for sparse representation. After locality constrained linear coding these codewords have been trained in the dictionary. Experimental results show that we get close to the accuracy obtained by the authors.

After implementing this paper, we extended the same framework and proposed a novel methodology for video copy detection. We first detect scene changes using edge maps and then divide videos into different classes of images. We then use the already built image classification framework for training these classes. When query video is passed, we randomly select 10 frames and classify it using the trained SVM. We then perform majority voting to get the desired video match. We achieve close to (in terms of accuracy) or even better (in terms of computational time) results compared to state of art techniques.

Table 4: Algorithms

Algorithm	Source
SIFT	Lazebnik et al. [2]
Solving for U	We implemented FISTA and used [16]
Solving for V	Used Prof. Honglak's code [14]
SPM	We implemented
Max Pooling	We implemented
L-R Sparse Decomposition 3.5	ALM [20]
LLC	We implemented
SVM	Yang et al. [3]
GUI for Image Classification	We implemented
Scene Change analysis	We implemented
Framework for Video Copy Detection	We implemented
GUI for Video Copy Detection	We implemented

7 Challenges

We learnt a great deal from this project. We used concepts from EECS 556 course such as image restoration and edge detection. We learnt about Lagrange Dual, Low-Rank and Sparse Decomposition, Locality-constrained Linear Coding, Scene Change Analysis and Support Vector Machine. But there are a few challenges that we faced during the course of presentation. There were a few missing links or assumptions in the paper that are not as obvious to us as authors. For example, why authors have used non-negative sparse coding is not clear. How authors chose values of gamma, lambda or beta in Eqns. 13, 14 and 4 is not clear either. The next challenge was the time required for each run of simulations. Playing with close to 10000 images is a time consuming task and we need faster training and testing methods. Though the proposed method is very fast on the testing side, it is really slow on the training side.

The same is true for video copy detection where the training time is huge. In video copy detection, SVM may fail if there are too many classes. So if we really want to make this algorithm feasible we may have to come up with different training and classifying strategies. This would be a really nice extension as these algorithms may have more demand in the future from Video hosting/sharing service providers.

References

[1] Sivic, Josef, and Andrew Zisserman. “Video Google: A text retrieval approach to object matching in videos.” In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pp. 1470-1477. IEEE, 2003.

[2] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol. 2, pp. 2169-2178. IEEE, 2006.

[3] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang, “Linear spatial pyramid matching using sparse coding for image classification,” In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 1794-1801. IEEE, 2009.

[4] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong, “Locality-constrained linear coding for image classification,” In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 3360-3367. IEEE, 2010.

[5] Chunjie Zhang, Jing Liu, Qi Tian, Changsheng Xu, Hanqing Lu, and Songde Ma, “Image Classification by Non-negative Sparse Coding, Low-rank and Sparse Decomposition.” In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 1673-1680. IEEE, 2011.

[6] Oostveen, Job, Ton Kalker, and Jaap Haitsma. ”Feature extraction and a database strategy for video fingerprinting.” In Recent Advances in Visual Information Systems, pp. 117-128. Springer Berlin Heidelberg, 2002.

[7] S. Lee and C. D. Yoo, “Robust video ngerprinting for content based video identication,” IEEE Transactions on CSVT, 2008

[8] Shanmuga Vadivel, K., Felix Fernandes, Zhan Ma, PoLin Lai, and Ankur Saxena. “Perceptual similarity based robust low-complexity video fingerprinting.” In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, pp. 1337-1340. IEEE, 2012.

[9] Gemert Van, Jan C., Cor J. Veenman, Arnold WM Smeulders, and J-M. Geusebroek, “Visual word ambiguity,” Pattern Analysis and Machine Intelligence, IEEE Transactions on 32, no. 7 (2010): 1271-1283.

[10] Oren Boiman, Eli Shechtman, and Michal Irani, “In defense of nearest-neighbor based image classification,” In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1-8. IEEE, 2008.

[11] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik, “SVM-KNN: Discriminative nearest neighbor classification for visual category recognition.” In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol. 2, pp. 2126-2136. IEEE, 2006.

[12] Yuan, Xiao-Tong, Xiaobai Liu, and Shuicheng Yan, “Visual classification with multitask joint sparse representation,” Image Processing, IEEE Transactions on 21, no. 10 (2012): 4349-4360.

[13] Lowe, David G. ”Distinctive image features from scale-invariant keypoints.” International journal of computer vision 60, no. 2 (2004): 91-110.

[14] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng, “Efficient sparse coding algorithms,” Advances in neural information processing systems 19 (2007): 801.

[15] Beck, Amir, and Marc Teboulle. ”A fast iterative shrinkage-thresholding algorithm for linear inverse problems.” SIAM Journal on Imaging Sciences 2, no. 1 (2009): 183-202.

[16] `http://spams-devel.gforge.inria.fr/documentation.html`

[17] Boureau, Y-Lan, Jean Ponce, and Yann LeCun. ”A theoretical analysis of feature pooling in visual recognition.” In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 111-118. 2010.

[18] Serre, T., Wolf, L., and Poggio, T, “Object recognition with features inspired by visual cortex, In Computer Vision and Pattern Recognition (CVPR), 2005, IEEE Computer Society Conference on (Vol. 2, pp. 994-1000). IEEE.

[19] Cands, E. J., Li, X., Ma, Y., and Wright, J. (2011), “Robust principal component analysis” Journal of the ACM (JACM), 58(3), 11.

[20] Lin, Z., Chen, M., and Ma, Y. (2010), “The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices,” arXiv preprint arXiv:1009.5055.

[21] Yu, Kai, Tong Zhang, and Yihong Gong. ”Nonlinear Learning using Local Coordinate Coding.” In NIPS, vol. 9, p. 1. 2009.

[22] Fei-Fei, Li, Rob Fergus, and Pietro Perona. ”Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories.” Computer Vision and Image Understanding 106, no. 1 (2007): 59-70.

[23] Wang, Zhou, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. ”The SSIM index for image quality assessment.” MATLAB implementation available online from: http://www. cns. nyu. edu/ lcv/ssim (2003).

Table 5: Group Project Effort Form

Task	Aniket	Naveen
leadership	50	50
planning/design	55	45
programming	55	45
testing	55	45
analysis	45	55
proposal	50	50
presentation	50	50
report	45	55