503 Final Tmp

15 April, 2018

Problem of Interest

Images have played a great part of our life after dat storage become cheaper and convenient in the 21st century. In 2001, Google enable users to search image with a 250 millions of images, and the data base has grown exponentially ever since its launch. Within a decade, Reverse image search[content-based image retrieval(CBIR)] was introduced. The importance of image recognition and automated image classification has become popular over time. Even though the concept of neural network was first introduced back in 1940s, computors was not available nor able to undertake massive calculation when it was first introduced, but nowadays we are able to compute complicated models and algorithmns such as Convolutional Neural Network(CNN), Deep Neural Network(DNN) ,. . . etc. Popular algorithmns were introduced to solve the image classification problems.

Data Exploration Challenge with Image data/ Difference with tabular data

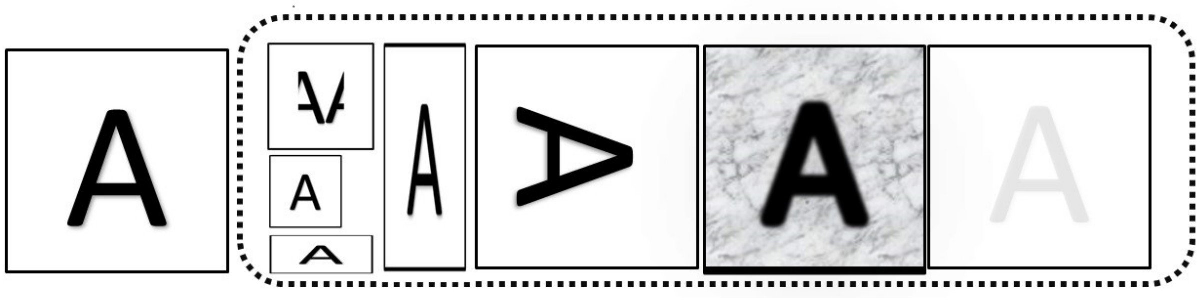
https://en.wikipedia.org/wiki/Channel_(digital_image).

Data Structure

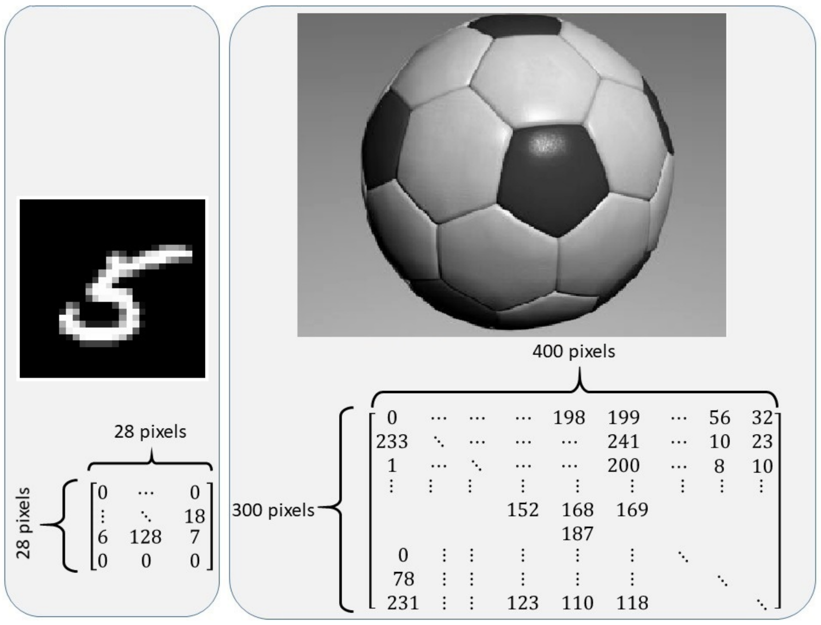
A color digital image can be decompose into an 3-dimensional array with each a n by m matrix *(or a nm length vector)*, which each matrix *(vector)* correspond to a specific channel **R,G,and B**.

Properties

The following figure displays variation of letter A, which in essence is still regonizable for human, be in the matrix representation is quiet different.



- Ordered/ Not exchangable: Even the data storage is tabular, but the columns are not exchangable, and pixels cells are highly correlated with its neighbors. Another problem arised when comverting vector back to original image, we need to be careful of the dimensions otherwise the image will be distorted.
- Rotation: Rotaion of an image may cause classifier to fail if not handle carefully, casue machine takes the matrix/vector representation of an image and and mention before, pixel are highly correlated and not exchangable.
- Noise: If the object is blend-in/ merge into the background capturing the main object image will be challenging.
- Scale: Size of the image grows in a non-linear rate, which results in the high dimensionality of the image. As an example, an image with 30x40 pixels ($\mathcal{R}^{1200}$) contains 1/100 entries compared with 300x400 pixels image ($\mathcal{R}^{120000}$).



Naïve Classifier - Nearest Neighbor Classifier

Introduction

First, we start with a naïve classifier which only consider the distance of the two matrices, the prediction of the test data is solely decided by the closet neighbor it can find among the training data space, we picked **L-1 norm** and **L-2 norm** for the disatnce calculation process.

- $$||M||_1 = max_{1\leq j\leq m}\sum_{i=1}^{m}|m_{ij}|$$

Which is the maximum absolute column sum of the matrix.

- $$||M||_2 = \sigma_{max}(M) \leq \left(\sum_{i=1}^{n}\sum_{j=1}^{m}|m_{ij}|^2\right)^{1/2}$$

  where $\sigma_{max}(M)$ is the maximun singular value of matrix M.

Algorithm

Suppose we have a test image, say M_{test} and set of training imgaes $M_{train_1}, M_{train_2}, \cdots, M_{train_n}$ with known class labels

- First calculate the matrix difference of the test image M_{test} with **ALL** training images $M_{train_1}, M_{train_2}, \cdots, M_{train_n}$, the difference of any two matrices is definded as the element-wise substraction, i.e.

$$A_{m\times n} - B_{m\times n} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} - \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix}$$
$$= \begin{bmatrix} a_{11}-b_{11} & \cdots & a_{1n}-b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}-b_{m1} & \cdots & a_{mn}-b_{mn} \end{bmatrix} = M_{m\times n}$$

- Second, we calculate the choose of *L-1 norm/L-2 norm* distance of the matrix from previous step.
- The prediction is made by comparing all *L-1/L-2* distances of the target test data to be classified, such that the predicted class is the one with minimun distance, i,e

$$Y_{\hat{Test}} = \left\{ N \in \{Class\ of\ ||M_i||_p\}; ||M_i||_p = min\Big\{||M_1||_p, \cdots, ||M_m||_p\Big\} \right\}, \text{where } p \text{ denotes } L_p \text{distance}$$

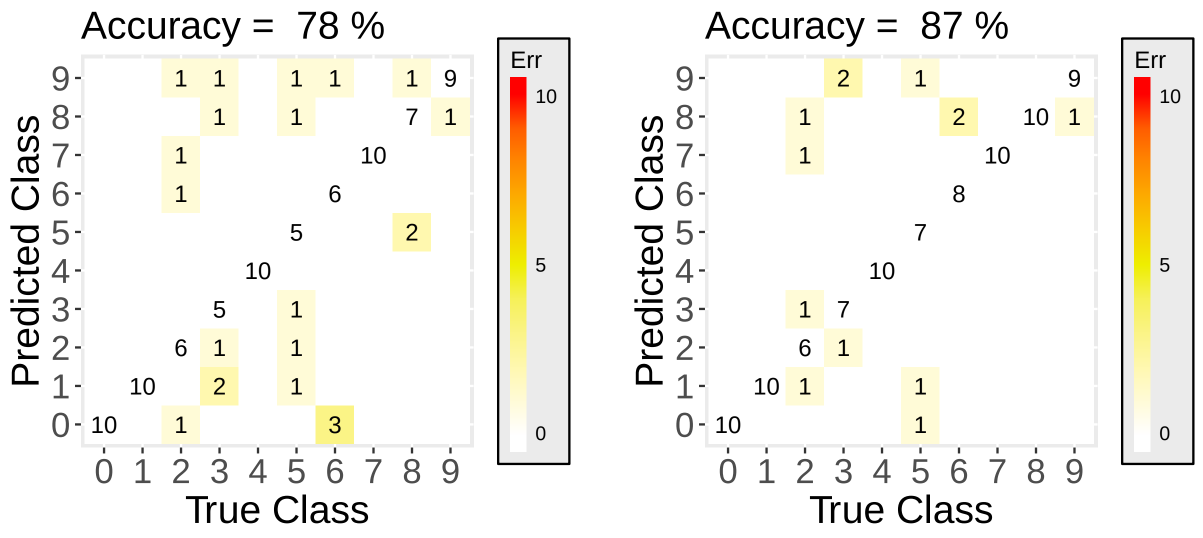
```
Algorithm 1: Nearest Neighbor Algorithm
  Data: $X_{Train}, X_{Test}, Y_{Train}$
  Result: Classification prediction for$X_{Test}$, $Y_{\hat{Test}}$
  begin
      foreach $X_{Test}$ do
          foreach $X_{Train}$ do
              tmp ← Calculate difference of the two matrix(Element-wise)
              D[j] ← Calculate $L_p$-distance(tmp)
          Find $idx_{Train}$ $s.t.$ D[$idx_{Train}$] = $\underset{j}{\text{argmin}}$ D[i,j]
          $Y_{\hat{Test}}$[i] ⟵ $Y_{Train}$[$idx_{Train}$]
          Remove $(D, idx_{Train})$, release memory space
      return $Y_{\hat{Test}}$
```

Data experiement

From the MNIST hand writting data set, we selected: + Training Set: First 100 observations per class (0-9), total of 1000 observations. + Test Set: First 10 observations per class (0-9), total of 100 observations.

The result from the test data (consist of 100 observations per class):

Classification Matrix for Low−Resolution Images



L1 vs. L2.

As we can seen that $L_2 - norm$ takes more information from the matrix as it penalizes among all the differences instead of the algorithm for $L_1 - norm$ which only takes one of all columns.

High- vc. Low- resolution images

The main difference between low- and high- resolution images are the information contained the the image vector. Results from applying naïve classifier to both high- and low- resolution images are what we expected to be quiet different. Even we only have fewer classes in the high-resolution set, the accuracy drop drastically.

Notes: Worth mentioning that this algorithm requires *ALL* images from train and test set to be consistant in width and height, ortherwise it will fail becasue the nature of the algorithm involves matrix element-wise substraction and hence calculate matrix-norm.

Introducing Feature Descriptors

Feature descriptors are algorithms motivated by recent works in computer vision and image processing that take an image and output a set of numerical feature vectors that encode *interesting* information from the image. This set of numerical feature vectors can be used to define a unique "fingerprint" for any given image. Recent works have uncovered ways of extracting information from images that is invariant under image transforamtions; this means that the same set of features are identified from an image if it is tranformed in ways such as scale, rotation, shift etc.

Histogram of Oriented Gradient (HOG)

The histogram of oriented gradient was proposed as a feature descriptor for object detection by Dalal and Triggs in 2005. For simplicity of presentation, HOG is considered on gray-scale images throughout our report.

Let D be an image, we would like to describe using HOG. We divide D into non-overlapping sub-images called cells, denoted C. Next, we form image blocks, denoted B, by combining cell sub-images. An important difference between cells and blocks is that cells are non-overlapping, while blocks can overlap. Starting with the cells definded in the upper-left corner of the image, blocks are formed in a slidding window format per row, with the last block consisting of the cells in the bottom-right corner.

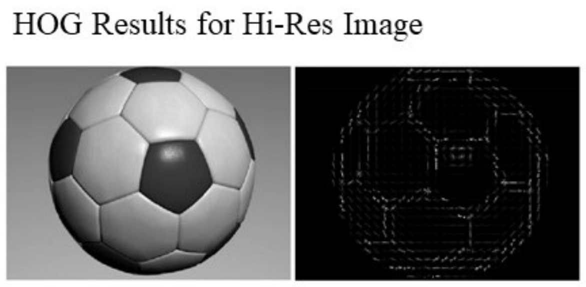
The gradient ∇D of image D is vector field over the image's domain (x, y) , where (x, y) is a pixel location in the image. This can be approximated by partial derivative along the x and y directions as,

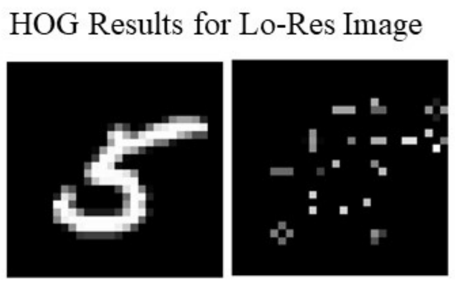
$$\nabla D \approx \begin{bmatrix} \frac{D(x+\delta,y)-D(x-\delta,y)}{2\delta} \\ \frac{D(x,y+\delta)-D(x,y-\delta)}{2\delta} \end{bmatrix}$$

The gradient provides two important pieces of information about the image's local stucture. $||\nabla D||$ tells us the magnitude of the local gradient and $\Theta = tan^{-1}\left(\frac{\nabla_y D}{\nabla_x D}\right)$ gives us the angle of the gradient.

The vector h_i, called the HOG of block B_i, is calculated by computing the $\big(\Theta, ||\nabla D||\big)$ for each pixel in the block. Downsampling by binning the values of Θ is used to reduce the gradients per block. Finally, the HOG of image D is defined by stacking all h_i in order. Therefore, the HOG feature descriptor for an image can be represented as a 2-dimensional vector.

The images below display image features identified by HOG for high and low resolution images.





PCA of Histogram of gradient(HOG)

We explore a multi-class classifier based on principle component analysis (PCA) of histogram of oriented gradient (HOG) for accurate and fast image classification.

Supposed we are given K classes of labeled images. For each image $x_i^{(j)}$ in class j where $i = \{1, \cdots, n_j\}$ and $j = \{0, \cdots, K-1\}$, we compute the HOG vector $h_i^{(j)}$. For each class j, the HOG vectors are arranged in a feature matrix $H^{(j)} = \big[h_1^{(j)}\, h_2^{(j)} \cdots h_{n_j}^{(j)}\big]$.

For each class j, we compute the mean vector and covariance matrix

$$\bar{h_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} h_i^{(j)}$$

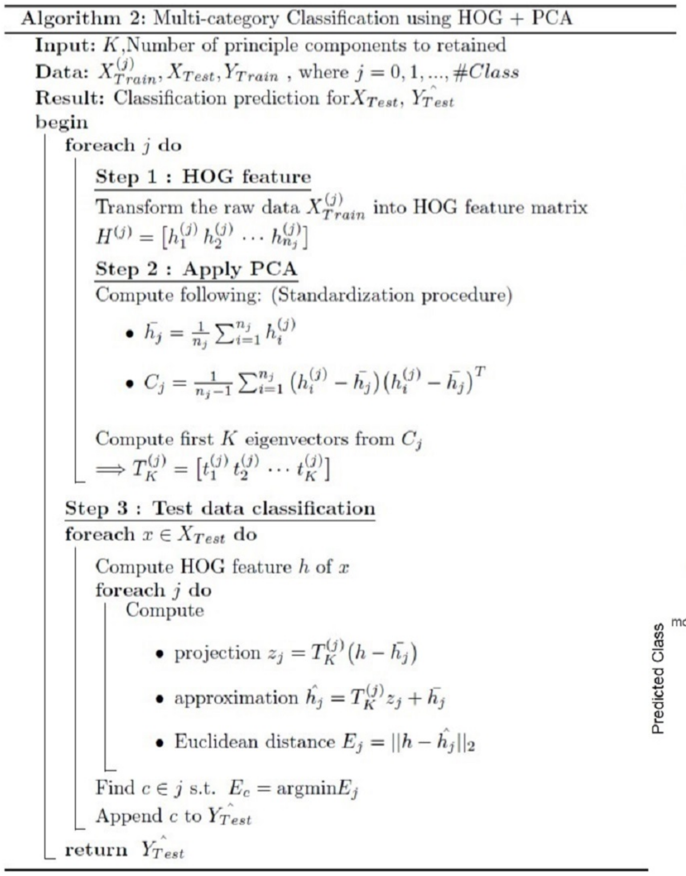
$$C_j = \frac{1}{n_j - 1}\sum_{i=1}^{n_j} \big(h_i^{(j)} - \bar{h_j}\big)\big(h_i^{(j)} - \bar{h_j}\big)^T$$

and then perform eigen-decomposition of C_j as

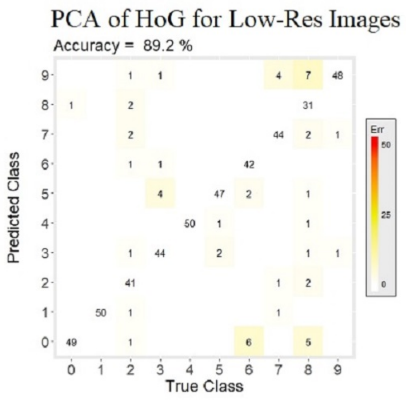
$$C_j = T_j \Lambda_j T_j^T$$

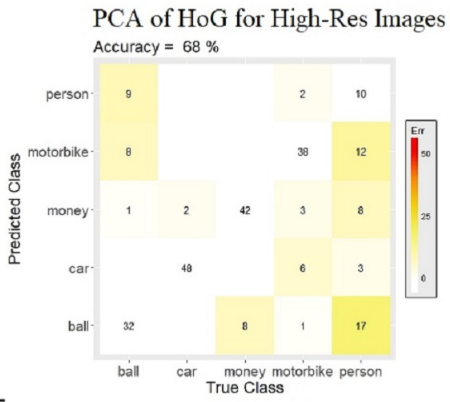
and construct matrix $T_P^{(j)} = \left[t_1^{(j)}\, t_2^{(j)}\, \cdots t_P^{(j)}\right]$ which contains P eigenvectors of C_j that are associated with the P largest eigenvalues. As a dimension reduction technique, the K image classes can be though of as being well represented by the reduced image set $\{\bar{h_j}, T_P^{(j)}\}$. $T_P^{(j)}$ defines a P dimensional subspace representing the variation of the individual images in class j from $\bar{h_j}$.

To classify a test image x, we first compute HOG features h of x. Then for each class j, we project the image to the P-dimensional subspace by computing $z_j = T_P^{(j)T}\left(h - \bar{h_j}\right)$. Next, we reconstruct the image by mapping it back to its original HOG dimensions by inverse transforming as $\hat{h_j} = T_P^{(j)} z_j + \bar{h_j}$. Finally, we compute the euclidean distance $E_j = ||h - \hat{h_j}||_2$ for $j = \{0, \cdots, K-1\}$ and classify x to the class j^* that minimizes $\left\{E_j, j = 0, \cdots, K-1\right\}$.



```
Algorithm 2: Multi-category Classification using HOG + PCA
  Input: $K$,Number of principle components to retained
  Data: $X_{Train}^{(j)}, X_{Test}, Y_{Train}$ , where $j = 0, 1, ..., \#Class$
  Result: Classification prediction for$X_{Test}$, $\hat{Y_{Test}}$
  begin
    foreach $j$ do
      Step 1 : HOG feature
      Transform the raw data $X_{Train}^{(j)}$ into HOG feature matrix
      $H^{(j)} = [h_1^{(j)}\, h_2^{(j)} \cdots h_{n_j}^{(j)}]$
      Step 2 : Apply PCA
      Compute following: (Standardization procedure)
        • $\bar{h_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} h_i^{(j)}$
        • $C_j = \frac{1}{n_j - 1}\sum_{i=1}^{n_j} (h_i^{(j)} - \bar{h_j})(h_i^{(j)} - \bar{h_j})^T$
      Compute first $K$ eigenvectors from $C_j$
      $\Longrightarrow T_K^{(j)} = [t_1^{(j)}\, t_2^{(j)} \cdots t_K^{(j)}]$
    Step 3 : Test data classification
    foreach $x \in X_{Test}$ do
      Compute HOG feature $h$ of $x$
      foreach $j$ do
        Compute
          • projection $z_j = T_K^{(j)}(h - \bar{h_j})$
          • approximation $\hat{h_j} = T_K^{(j)} z_j + \bar{h_j}$
          • Euclidean distance $E_j = ||h - \hat{h_j}||_2$
      Find $c \in j$ s.t. $E_c = \text{argmin} E_j$
      Append $c$ to $\hat{Y_{Test}}$
    return $\hat{Y_{Test}}$
```





As seen in the classification matrices above, the PCA of HOG method is able to achieve an impressive 89.2% classification accuracy for low-resolution images. However this method does not fare as well in classifying high-resolution images; it is only able to achieve 68% classification accuracy for high-resolution images.

HOG+PCA	Low-resolution image		High-resolution image	
#PCs	Accuracy	Classify Time (ms)	Accuracy	Classify Time (ms)
10	0.876	9.98	0.68	175.2
20	0.868	10.5	0.68	179.2
50	0.892	11.5	0.68	179.6
100	0.848	11.98	0.68	262.8

A notable feature of the PCA of HOG model is its fast classification performance. The table above shows that on a 2-core Intel-i7 processor, this model is able to classify low-resolution images in approximately 10 milliseconds. This benchmark is important because the broadcasting standard for videos is 24 frames per second. This means that each frame (or image) in a video is on screen for 42 milliseconds. With PCA of HOG, we should be able to perform real-time streaming low-resolution video classification.

Another important interpretation of the results above is that the performance of classification in high-resolution images does not improve as we reduce the effect of dimensionality reduction (or increase the number of principal components). This result informs us of the limitation of HOG as a feature descriptor for high-resolution images. This is why, in the following models proposed in this report, we will try to improve HOG with spstial locality information.

Bag of Feature

Bag-of-features model (BoF) for image processing is an adaptation of the bag-of-words model from text mining. In this approach, image features are treated analogously to words, and images to documents. The complete space of image features defines the visual vocabulary, and each image is represented as a set of words (features) from this vocabulary. The flowchart below displays the steps in the BoF model.

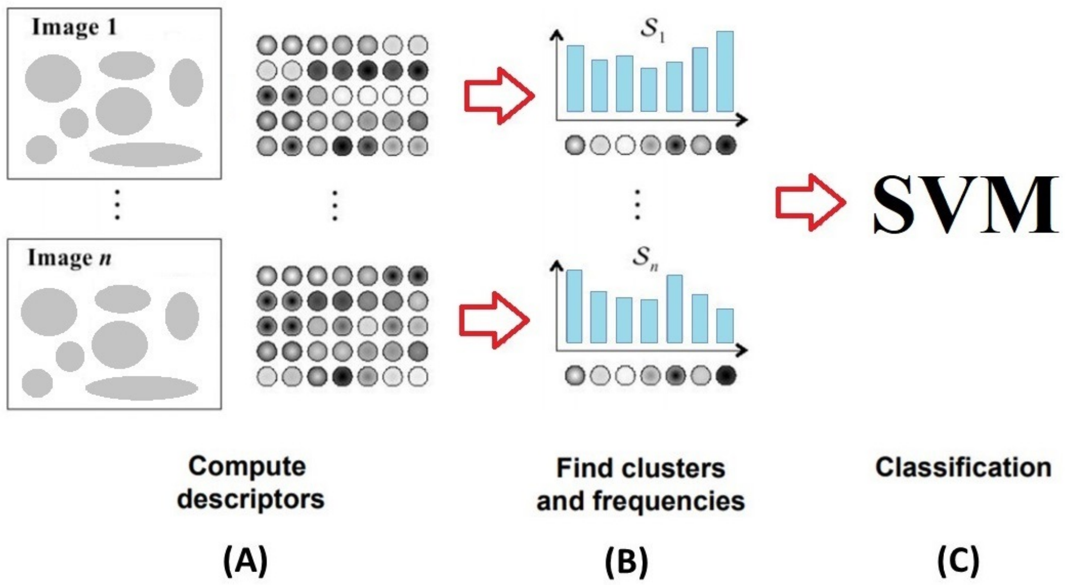


Image credit (BOF Flow Chart): Image adapted from Arénaire project, Laboratoire de l'Informatique du Parallélisme(LIP)

- **Step (A): Compute descriptors**

Feature detection and description algorithms must be applied to each image to extract local image features. These features are represented as numerical vectors.

- **Step (B): Find cluster and frequencies**

The numerical vector representation of a local image features are converted to “codewords” (analogous to words in text documents). These “codewords” are collected to produce a visual vocabulary (analogous to a word dictionary). One simple method of converting feature vectors to the visual vocabulary is by performing K-means clustering over all local image features. The learned cluster centers define the “codewords” and the number of clusters defines the vocabulary size.

At this stage, each image can be represented as a combination of “codewords” from our visual vocabulary. It is worth noting that an image may contain multiple instances of the same “codeword”, which is why each image is typically described as a histogram of “codewords”.

- **Step (C): Classification**

A classification decision boundry can be computed using the set-of-codewords representation of training images for each class. A variety of high-dimensional supervised learning models can be used for this step (such as SVM and AdaBoost).

To classify a test image, feature extraction is conducted as in step A. These test image features are mapped to the nearest "codewords" in our visual vocabulary. In the case where K-means is used to originally define the visual dictionary, this nearest mapping is simply minimizing the Euclidean Distance between each test feature vector and "codewords". Now, the test image can be represented as a collection of "codewords" and the classifier from step C can be used to predict the label of the test image.

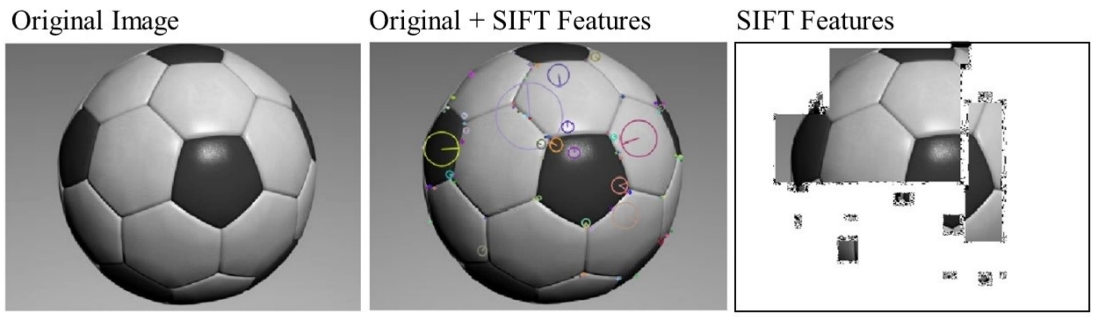
Bag-of-Features with SIFT

Scale-Invariant Feature Transform (SIFT)

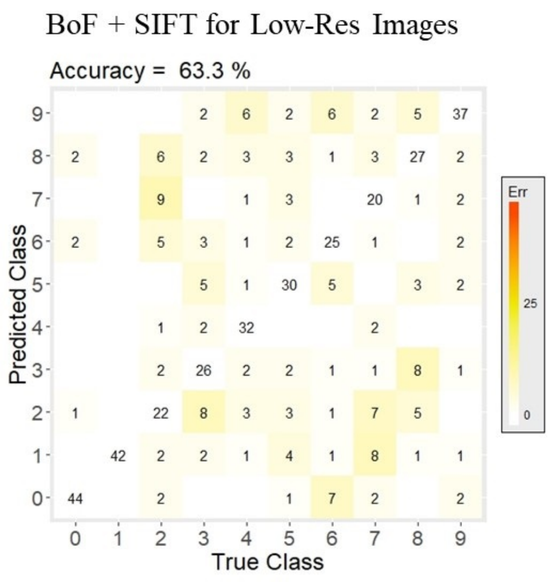
Scale-Invariant Feature Transform (SIFT) as a computer vision algorithm that was published by Lowe in 1999. This descriptor detects and describes a collection of local features in images that are invariant to image transformations of scaling, rotation, illumination changes, and geometric distortion.

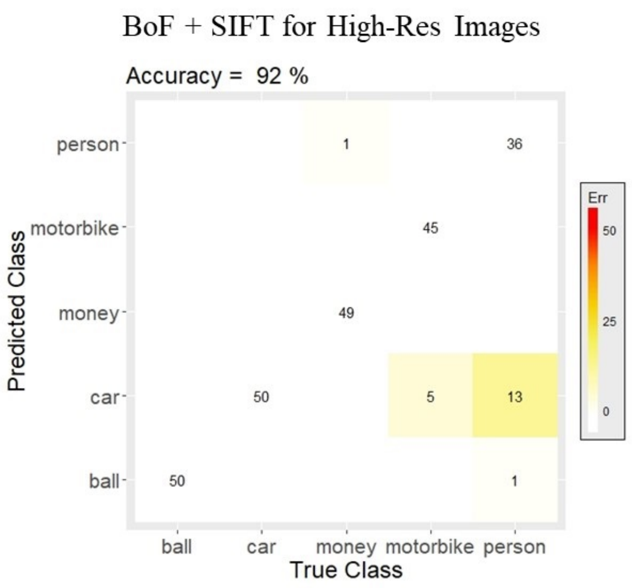
The features highlighted by SIFT are computed by finding the local minimums and maximums of the difference of gaussians at various scales. The direction of local gradient, computed with respect to neighboring pixels in Gaussian-blurred image, is used to assign one or more dominant orientation to each feature.

SIFT is a popular feature detection and description algorithm, and we propose using it in the first step of the bag-of-features model. Next, for the clustering step to construct the visual vocabulary using the SIFT identified features, we suggest K-means for its simplicity. Finally, for the multi-class classification of image feature vectors, we employ SVM with the radial basis function kernel.



The classification matrices below show that the combination of a BoF and SIFT is able to achieve an impressive 92% classification accuracy for high-resolution images. However, this model does not fare well in classifying low-resolution images; it is only able to achive 63% classification accuracy for low-resolution images.





The table below summarizes the performace of the BoF and SIFT models over a range of visual vocabulary sizes. The classification accuracy of this model increases as the vocabulary sizes increases. However, these gains only contiue up to a certain point, after which the model's performance stabilizes. Also, previously we mentioned the 42 millisecond classification-time threshold for real-time streaming video classification. As seen below, with a few technical refinements, the Bof and SIFT model should be able to classify high-resolution video in real-time.

SIFT	Low-resolution image		High-resolution image	
Vocab. Size	Accuracy	Classify Time (ms)	Accuracy	Classify Time (ms)
30	0.474	0.7	0.8	47.2
50	0.552	1.268	0.84	48
100	0.602	1.4	0.92	49.6
200	0.633	1.78	0.92	51.2

Bag-of-Features with SIFT+HOG

TODO: Explain BoF with HOG

TODO: BoF with HOG, high-/low- classification matrix

TODO: Bof with HOG interpret result

Feature Descriptors + SVM

TODO: flow chart

TODO: HOG + SVM

TODO: HOG + SVM, high-/low- classification matrix

TODO: HOG + SVM interpret result