

503 Final Tmp

15 April, 2018

Problem of Interest

Images have played a great part of our life after data storage became cheaper and convenient in the 21st century. In 2001, Google enabled users to search images with a 250 million image database, and the database 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 networks was first introduced back in the 1940s, computers were not available nor able to undertake massive calculations when it was first introduced, but nowadays we are able to compute complicated models and algorithms such as Convolutional Neural Networks (CNN), Deep Neural Networks (DNN), ... etc. Popular algorithms were introduced to solve the image classification problems.

Data Exploration Challenge with Image data/ Difference with tabular data

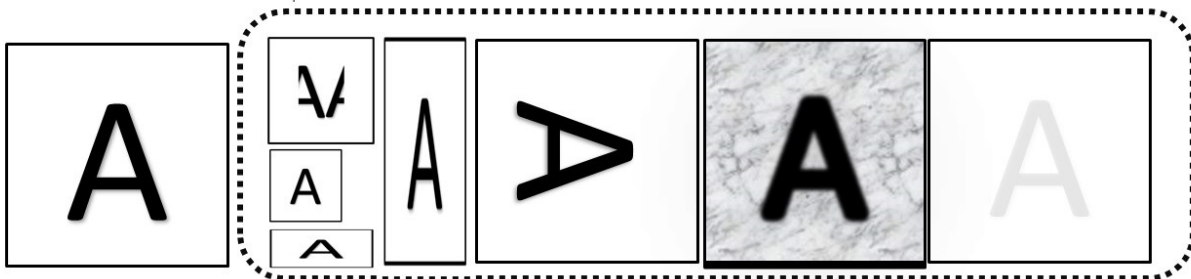
[https://en.wikipedia.org/wiki/Channel_\(digital_image\)](https://en.wikipedia.org/wiki/Channel_(digital_image)).

Data Structure

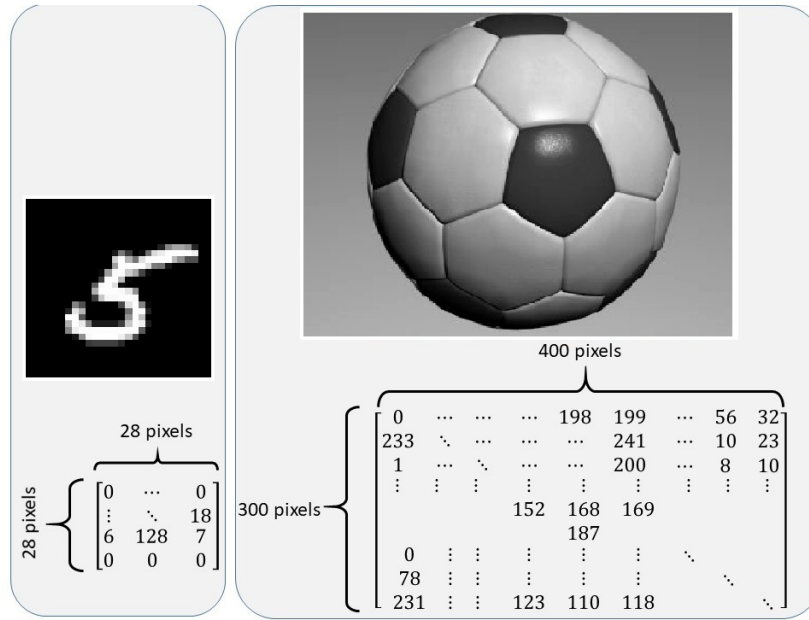
A color digital image can be decompose into an 3-dimensional array with each a n by m matrix (*or a nm length vector*), which each matrix (*vector*) correspond to a specific channel **R,G,and B**.

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 converting 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 maximum singular value of matrix M .

Algorithm

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

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

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

- 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 minimum distance, i.e

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

Algorithm 1: Nearest Neighbor Algorithm

Data: $X_{Train}, X_{Test}, Y_{Train}$
Result: Classification prediction for $X_{Test}, Y_{Test}^{\hat{}}$
begin

 foreach X_{Test} **do**

 foreach X_{Train} **do**

 tmp \leftarrow Calculate difference of the two matrix(Element-wise)

 D[j] \leftarrow Calculate L_p -distance(tmp)

 Find idx_{Train} s.t. $D[idx_{Train}] = \underset{j}{\operatorname{argmin}} D[i,j]$

 $Y_{Test}^{\hat{}}[i] \leftarrow Y_{Train}[idx_{Train}]$

 Remove (D, idx_{Train}) , release memory space

 return $Y_{Test}^{\hat{}}$

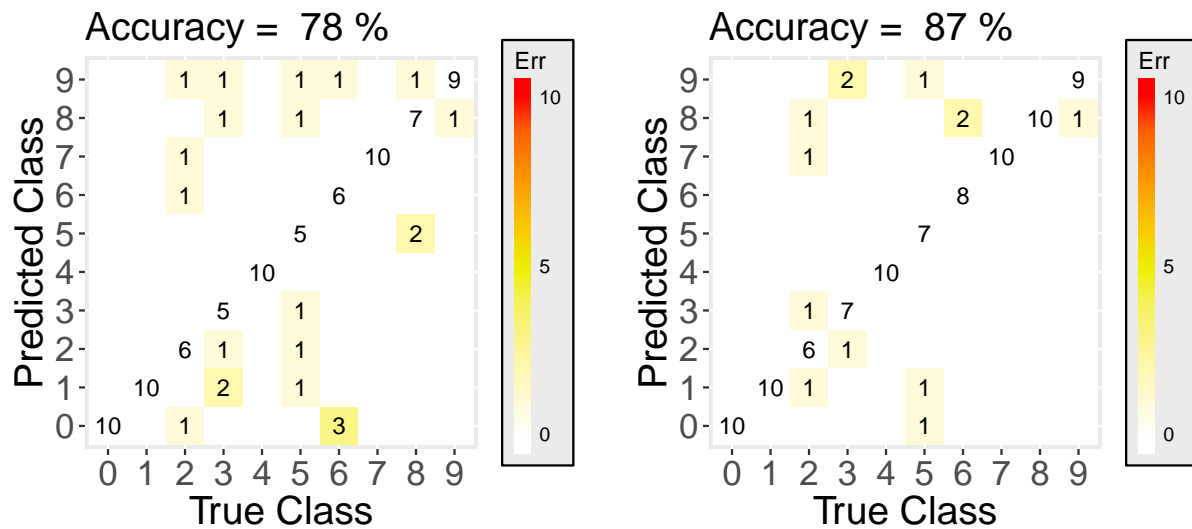
Data experieiment

From the MNIST hand writing 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.

QUESTION: where to deliver high- low -res problem

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

Classification Matrix for Low-Resolution Images



L1 vs. L2.

As we can see 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- vs. Low- resolution images

The main difference between low- and high- resolution images are the information contained in the image vector. Results from applying naïve classifier to both high- and low- resolution images are what we expected to be quite 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 consistent in width and height, otherwise it will fail because the nature of the algorithm involves matrix element-wise subtraction 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 transformations; this means that the same set of features are identified from an image if it is transformed 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 defined in the upper-left corner of the image, blocks are formed in a sliding 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,

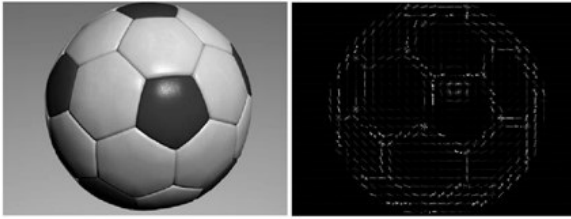
$$\nabla D \approx \begin{bmatrix} \frac{D(x+\delta, y) - D(x-\delta, y)}{2\delta} \\ \frac{D(x, y+\delta) - D(x, y-\delta)}{2\delta} \end{bmatrix}$$

The gradient provides two important pieces of information about the image's local structure. $\|\nabla D\|$ 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.

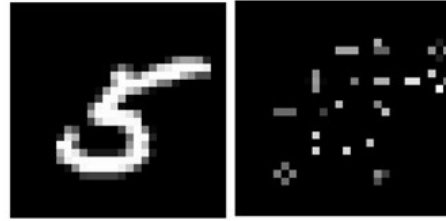
The vector h_i , called the HOG of block B_i , is calculated by computing the $(\Theta, \|\nabla D\|)$ for each pixel in the block. Downsampling by binning the values of Θ is used to reduce the gradients per block. Finally, the HOG of image D is defined by stacking all h_i in order. Therefore, the HOG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

TODO: Discuss HOG on low- and high- resolution data set.

HOG Results for Hi-Res Image



HOG Results for Lo-Res Image



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 $i = \{1, \dots, n_j\}$ and $j = \{0, \dots, K-1\}$, we compute the HOG vector $h_i^{(j)}$. For each class j , the HOG vectors are arranged in a feature matrix $H^{(j)} = [h_1^{(j)} h_2^{(j)} \dots h_{n_j}^{(j)}]$.

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

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

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

and then perform eigen-decomposition of C_j as

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

and construct matrix $T_P^{(j)} = [t_1^{(j)} t_2^{(j)} \dots t_P^{(j)}]$ 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 thought 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 $z_j = T_P^{(j)T}(h - \bar{h}_j)$. 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 $j = \{0, \dots, K-1\}$ and classify x to the class j^* that minimizes $\{E_j, j = 0, \dots, K-1\}$.

Algorithm 2: Multi-category Classification using HOG + PCA

Input: K , Number of principle components to retained

Data: $X_{Train}^{(j)}, X_{Test}^{(j)}, Y_{Train}^{(j)}$, where $j = 0, 1, \dots, \#Class$

Result: Classification prediction for $X_{Test}^{(j)}, Y_{Test}^{(j)}$

begin

foreach j **do**

Step 1 : HOG feature

 Transform the raw data $X_{Train}^{(j)}$ into HOG feature matrix

$$H^{(j)} = [h_1^{(j)} h_2^{(j)} \dots h_{n_j}^{(j)}]$$

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

$$\Rightarrow T_K^{(j)} = [t_1^{(j)} t_2^{(j)} \dots t_K^{(j)}]$$

Step 3 : Test data classification

foreach $x \in X_{Test}$ **do**

 Compute HOG feature h of x

foreach j **do**

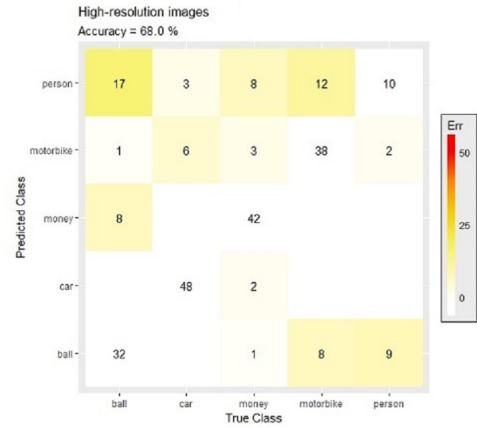
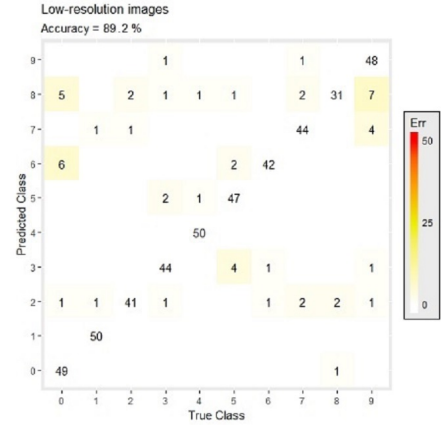
 Compute

- projection $z_j = T_K^{(j)T}(h - \bar{h}_j)$
- 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 $Y_{Test}^{(j)}$

return $Y_{Test}^{(j)}$



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

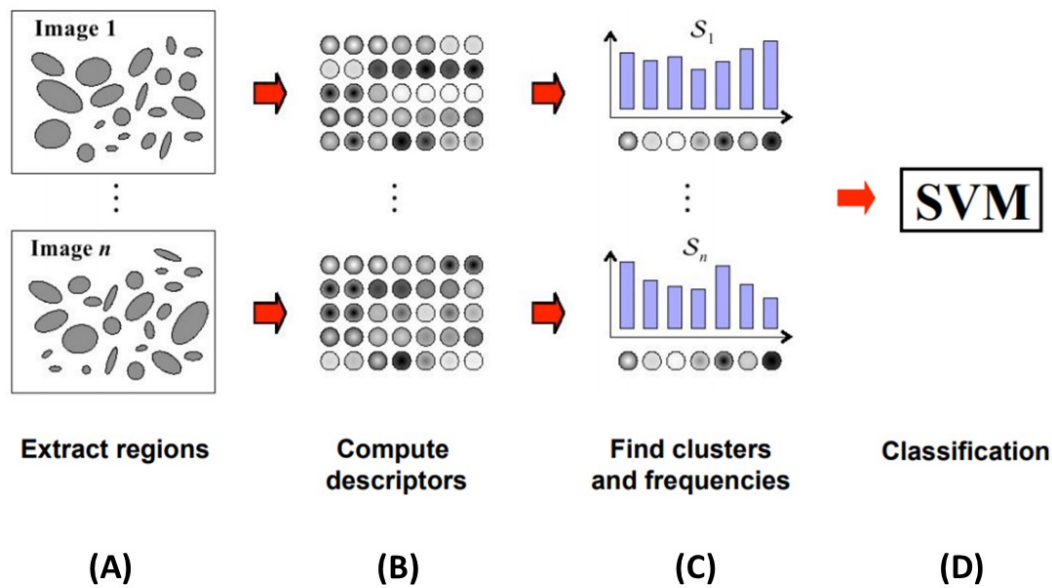


Figure 1: Bag Of Features: Flow Chart

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 spatial locality information.

Bag of Feature

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

TODO: Explain model using flow chart

Bag of feature(BoF) is an implementation of bag of words from text mining in image classification. Instead of capturing common words and build up character/ string code book, BoF classification is based on feature descriptors, of which there are various source of technique such as SIFT, HOG implemented in this report or can be replaced to PCA of an image,... etc. When a new test image needs to be classified, it counts the feature of the image and **looks up** the code up and counts the frequencies of the feature descriptors, and feed the information of counts into classifier for classification.

- **Step (A):**

First we scan over the image and pick region of area in interest for feature

- **Step (B):**

For each image, we may find n_i descriptors, we collect all feature descriptors among all i images and using cluster technique(K-mean in the report) to label descriptors in to selected N clusters.

- **Step (C):**

For each train data, we map the extracted n_i descriptors into N categories by following the cluster result and count the frequencies of each categories, the process will reduce the image into a vector of size N containing counts of categories, which can be read as a representative profile of an image.

- **Step (D):**

For test images, we perform the same algorithm as in **Step A** and **Step C** so that the test image is processed into a length N vector, then use classifier(in our case SVM) for classification result.

TODO: Pseudo code

Algorithm 3: Bag of Features

Input: N , Code-book size
Data: $X_{Train}^{(j)}, X_{Test}, Y_{Train}$, where $j = 0, 1, \dots, \#Class$
Result: Classification prediction for X_{Test}, \hat{Y}_{Test}

```

begin
  i ← #XTrain
  foreach  $x_i \in X_{Train}$  do
    Step 1 : Feature extraction
    • Extract regions
    • Compute descriptors
    Add extracted descriptors,  $\{d_{i1}, d_{i2}, \dots\}$  into descriptor sets,  $D$ 
  Step 2 : Build Code Book
  For  $D$ , find  $N$  cluster using clustering method s.t. Code Book has  $N$  clusters
  foreach  $d \in D$  do
    Assign descriptors  $\{d_{i1}, d_{i2}, \dots, d_{iN}\}$  cluster label  $\Rightarrow$ 
     $\{d_{i1}^{(0)}, d_{i2}^{(3)}, \dots, d_{iN}^{(j)}\}$ , where  $j \in \{0, 1, \dots, N-1\}$ 
    Feature Dictionary,  $F_j$  contains descriptors with cluster label  $j$ 
  for  $i$  do
    Step 3 : Compute Frequencies
    Compute frequencies of extracted descriptors in the Code Book,  $F$ 
    foreach  $j$  do
       $S_{ij} = \#\{d_{i.}^{(j)} \in F_j\}$ 
    Step 4 : Test data classification
    foreach  $x \in X_{Test}$  do
      Perform Step 1 + Step 3  $\Rightarrow$  Classifier (e.g. SVM)
      Append classifier result to  $\hat{Y}_{Test}$ 
    return  $\hat{Y}_{Test}$ 

```

BoF with HOG

TODO: Explain BoF with HOG

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

TODO: Bof with HOG interpret result

SIFT

TODO: Introduce SIFT

BoF with SIFT

TODO: Explain BoF with SIFT

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

TODO: Bof with SIFT interpret result

Feature Descriptors + SVM

TODO: flow chart

TODO: HOG + SVM

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

TODO: HOG + SVM interpret result