VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Algorithms used for classifying visual signals using methods for extracting significant features

Algoritmy pro klasifikaci visuálních signálů s využitím technik extrakce významných rysů

Bc. Vojtěch Dorňák

Diploma Thesis

Supervisor: Ing. Lukáš Pospíšil, Ph.D.

Ostrava, 2021

Abstrakt

Klasifikace vizuálních dat je důležitý úkol v moderním oboru počítačového vidění. Kvalita často používaných neuronových sítí závisí na velkém objemu trénovacích dat. Tato práce prozkoumává alternativní konvekční klasifikační metody, kde je rozměr dat redukován pomocí projekce na prostor významných bodů. Lokální a globální techniky extrakce významných bodů, a to SIFT, SURF, ORB a PCA jsou porovnány. Klasifikace pomocí Support Vector Machine a Bayeovský Model je prozkoumána na třech datasetech o malém množství trénovacích dat a různé komplexnosti

Klíčová slova

klasifikace vizuálních dat, extrakce rysů, Support Vector Machine, Bag of Words, k-means, Scale-Invariant Feature Transform, Speed-Up Robust Features, Oriented Fast and Rotated Brief, Analýza hlavních komponent

Abstract

Classification of visual data is a fundamental task in state-of-the-art computer vision. The efficiency of the commonly used neural networks crucially depends on a large amount of training data. This thesis explores alternative conventional classification methods, where the dimension of the visual data is reduced by projecting the data onto a space of significant features. The local and global feature extraction techniques, namely SIFT, SURF, ORB, and PCA are compared. The classification by Support Vector Machine and the Bayesian Model is examined on three datasets with a small number of training images of different complexity.

Keywords

visual data classification, feature extraction, Support Vector Machine, Bag of Words, k-means, Scale-Invariant Feature Transform, Speed-Up Robust Features, Oriented Fast and Rotated Brief, Principal Component Analysis

Acknowledgement

I would like thank the thesis supervisor Ing. Lukáš Pospíšil, Ph.D. and the thesis consultant Ing. Marek Pecha for their guidance throughout every step of writing this thesis. Their expertise helped form this text into the version you are reading now. I would also like to thank my girlfriend for her everyday support while this thesis was being created.

Contents

List of symbols and abbreviations

SIFT	–	Scale Invariant Feature Transform
SURF	–	Speed Up Robust Features
ORB	–	Oriented Fast and Rotated Brief
SVM	–	Support Vector Machine
PCA	–	Principal Component Analysis
BoVW	–	Bag of Visual Words

List of Figures

List of Tables

Chapter 1

Introduction

Classification of visual data is an important task in the modern computer vision field. Nowadays, as the number of photographs taken every day increases, the importance of visual data classification is even more prominent. It can be used for data organization, automatic inspection in a manufacturing process, navigation, etc. [1].

The popular approach to visual data classification is the use of neural networks. However, we explore a different approach.

In [2], we have shown that the classification of image data, transformed into a feature space by the means of SIFT feature transformation, using the SVM, is a viable option. In this thesis, we expand the idea to other feature transformation techniques and another conventional classification method, the Bayesian Model. For the feature transformation, we use SIFT, SURF, ORB, and PCA extractors. We experiment with these approaches on three datasets of different complexity.

We describe the feature transformation techniques in chapter 2 and the classification methods in chapter 3. We introduce the three datasets of different complexity in chapter 4. Finally, we present the results of this approach in chapter 5.

Chapter 2

Image Data Transformation

Currently, recording of visual signals in the form of photographs is common. Since the resolution, and consequently the storage requirement grows, it is not practical to process such data directly in raw format. Moreover, while processing such data, we come across the problem of "small data", where the number of the feature dimension is significantly higher than the number of observations. Therefore, it might be beneficial to transform the image data into a feature space using a feature extraction technique. The feature extraction techniques can be split into two categories: local and global feature extractors [3].

Local feature extractors determine such points in an image, which could be found regardless of the position of the subject in the image. These points are called key-points. The area around such key-points is described using a vector called a descriptor. The descriptors are created in a way, which allow matching similar features.

On the other hand, global feature extractors transform the whole image into a lower-dimensional vector attempting to retain the most important information. The remaining information in the image is considered to be a noise, which should not have any effect on the classification of the image. Therefore, it can be eliminated.

In this thesis, we compare three local feature extractors, i.e., SIFT, SURF and ORB, and a global feature extractor, i.e., PCA. In this section, we review the details of these four feature extraction approaches.

2.1 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) was proposed by David Lowe, and published in the original paper [4] in 1999. Compared to ordinary techniques such as the Harris Corner Detector [5] or Canny edge detector [6], the SIFT identifies the general (not only edges and corners) key-points. The descriptors, which describe the local image region around the key-points are scale-invariant. Moreover, they are invariant to rotation, illumination, and change in affine transformations. In

the text, we introduce the methodology of training the classifier that recognizes the objects in real images (photographs); therefore the properties of SIFT feature vector are essential.

2.1.1 Key-point Detection

Let us consider an input image $I_{img}(x, y)$. A convolution of the image with a Gaussian kernel

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.1}$$

at a scale $\sigma > 0$ is exploited to get a scale-space representation of the image so that:

$$L(x, y, \sigma) = G(x, y, \sigma) * I_{img}(x, y).$$

To detect scale-invariant key-points, a scale normalized Laplacian of Gaussian (LoG)

$$\triangle_{norm} L(x, y, \sigma) = \sigma \left(\frac{\partial^2 L(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 L(x, y, \sigma)}{\partial y^2} \right)$$

is required [7]. It has been shown [8], that the local extrema of LoG provide the most stable image features, compared to many other popular image functions.

Determination of the LoG would be time-consuming, therefore, it is approximated by the Difference of Gaussian (DoG) [9]. The DoG is computed from two adjacent scales separated by a constant multiplicative factor $k \in \mathbb{R}$:

$$D(x, y, \sigma) := L(x, y, k\sigma) - L(x, y, \sigma).$$

To ensure the scale invariance, the extrema are located not only in the domain of one DoG, but it is found across different scales as well. The different DoGs at the scales are constructed by progressively convolving the original image with the Gaussian kernel. The scale of consecutive the consecutive Gaussian kernel differs by the multiplicative factor k.

At each doubling of the scale k, the resolution of an image can be reduced by a factor of 2 to provide better efficiency. Each group of blurred images of the same resolution is called an octave. In each octave, we generate the DoG by subtracting the $L(x, y, \sigma)$ of neighboring scales. This is called the DoG pyramid and can be seen in Figure 2.1.

Each pixel in the DoG pyramid is compared to the 3×3 pixels in DoGs below and above, and 8 surrounding pixels. This is shown in Figure 2.2. The pixel is selected as a key-point candidate if it has a lower or a higher value than all of these 26 pixels.

To detect sub-pixel locations of extrema, the DoG is interpolated using quadratic Taylor expansion

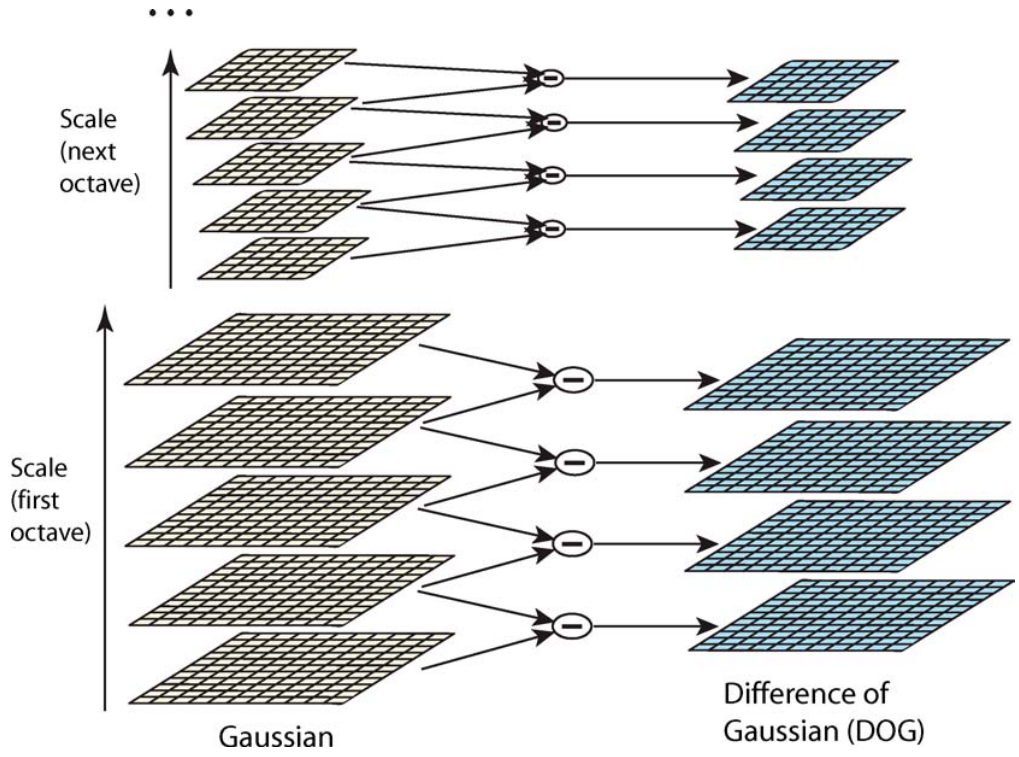


Figure 2.1: DoG pyramid [9].

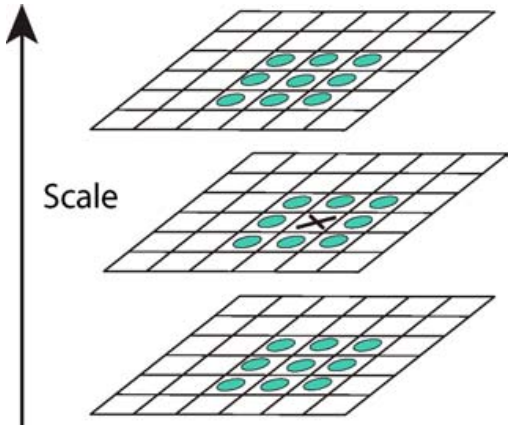


Figure 2.2: Optima of the DoG are selected by the means of comparing pixel to the 26 pixels surrounding it in the scale-space [9].

at each key-point candidate:

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\frac{\partial^2 D}{\partial \boldsymbol{x}^2}\boldsymbol{x},$$

where $D(x, y, \sigma)$ and the partial derivatives are evaluated at the key-point and $\boldsymbol{x} = (x, y, \sigma)$ is the offset from the key-point. Afterwords, extrema $\hat{\boldsymbol{x}}$ is located by setting the gradient of D to be a zero

vector:

$$\hat{\boldsymbol{x}} = -\frac{\partial^2 D^{-1}}{\partial \boldsymbol{x}^2}\frac{\partial D}{\partial \boldsymbol{x}}.$$

If the offset $\hat{\boldsymbol{x}}$ is larger than 0.5 in any dimension, it indicates that the extrema is closer to another point. In this case, the candidate point is changed to the new point, and interpolation is performed again.

The function value at the sub-pixel extrema ,$D(\hat{\boldsymbol{x}})$, is used for rejecting extrema with low contrast. In our experiments, we use the OpenCV SIFT implementation, where the default value is $\frac{0.04}{3}$ [10].

The DoG has a strong response along edges, even if the location along the edge is poorly determined. These candidate key-points have principal curvature perpendicular to the edge much larger than the principal curvature along it. The principal curvatures can be computed from a 2×2 Hessian matrix at the location of the key-point candidate

$$\boldsymbol{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix},$$

where the partial derivatives are estimated by taking the differences of neighboring sample points. The eigenvalues of $\boldsymbol{H}$ are proportional to the principal curvatures.

The computation of eigenvalues can be avoided, as only the ratio of the eigenvalues is required. Let us denote the larger eigenvalue as α and the smaller one as β. The trace of $\boldsymbol{H}$ is defined as

$$Tr(\boldsymbol{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

and the determinant as

$$Det(\boldsymbol{H}) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta.$$

Let r be the ratio of the eigenvalues, such that

$$r = \frac{\alpha}{\beta}.$$

Then,

$$\frac{Tr(H)^2}{Det(\boldsymbol{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}.$$

Therefore, to inspect the ratio of eigenvalues, we can check the value of

$$\frac{Tr(H)^2}{Det(\boldsymbol{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}.$$

The candidate key-points with this ratio below a threshold r is discarded. In our experiments, we use the default value of OpenCV SIFT implementation, which is equal to 10.

2.1.2 Key-point Description

We want to create a descriptor for each of our key-points. These descriptors must be almost identical across different scales, rotations, illuminations, and other transformations.

To ensure descriptor invariance to a rotation, we first determine the key-point orientation. We calculate the gradient magnitude

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2},$$

and orientation

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right),$$

of each point within a region around the key-point. From these values, an orientation histogram with 36 bins is created. The largest peak is selected as a key-point orientation. If there are other peaks more than 80% of the largest peak, new key-points are created at the same location with the other peaks as their orientation.

An example of key-points and their orientation, found by OpenCV SIFT implementation, can be seen in Figure 2.3.

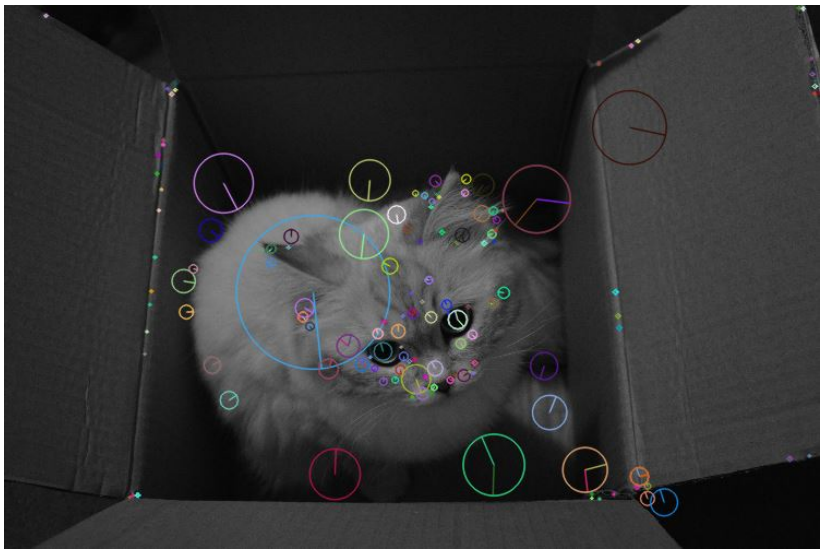


Figure 2.3: SIFT key-points and their orientation from an image of a cat.

The descriptor is constructed from a window of the size 16×16 pixels around the key-point. This window is rotated by the orientation of the key-point, which ensures the rotation invariance of the key-point. In this window, the gradient magnitude and the orientation are computed for each point. The 16×16 window is divided into 16 (4×4) sub-windows. For each sub-window, an 8 bin gradient orientation histogram, weighted by gradient magnitudes, is created. In Figure 2.4, we present the situation for a smaller (8×8) window. These histograms form a descriptor vector.

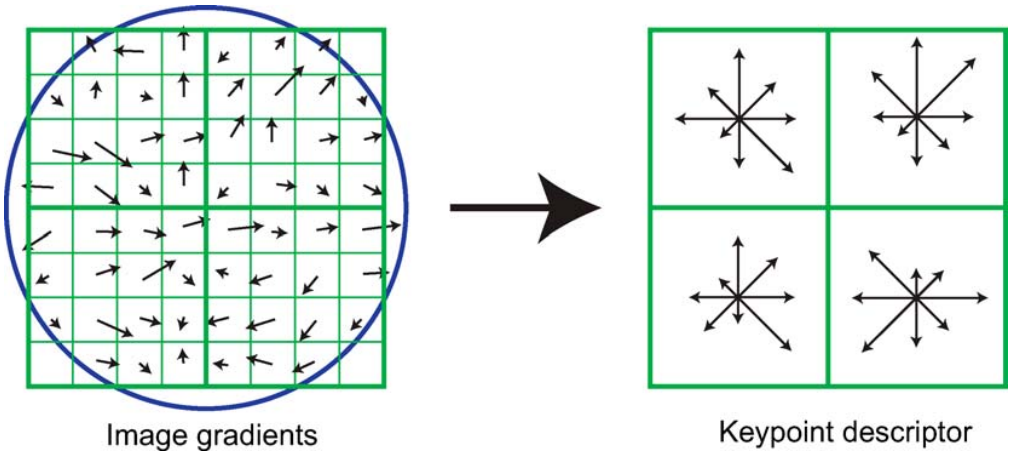


Figure 2.4: Building of the SIFT-descriptor [9].

2.2 Speeded Up Robust Features

Speeded Up Robust Features (SURF) is a local feature transformation algorithm proposed by Herbert Bay, Tinne Tuytelaars, and Luc Van Gool in 2006 [11]. Compared to SIFT [4], the authors claim the SURF detector and descriptor to be faster and more robust against various image transformations.

2.2.1 Key-point Detection

The SURF key-point detection is based on the determinant of the Hessian matrix. Let us consider an input image $I_{img}(x, y)$ and the scale-space representation of the image

$$L(x, y, \sigma) = G(x, y, \sigma) * I_{img}(x, y),$$

where $G(x, y, \sigma)$ is the Gaussian kernel defined in (2.1).

The Hessian matrix at scale σ is defined as

$$\mathcal{H}(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix},$$

where $L_{xx}(x, y, \sigma)$, $L_{xy}(x, y, \sigma)$, and $L_{yy}(x, y, \sigma)$ are the second-order derivatives of the scale-space representation of the image at a point (x, y).

The SURF authors decided to approximate the second order derivatives of Gaussian filter with box filters (shown in Figure 2.5). Because the property of Gaussian filter, that no new structures can appear while going to a lower resolution, has been shown not to apply in 2D case [7]. Box filters allow for the use of integral images, which reduces the computational complexity.

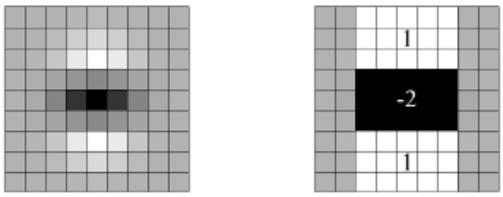


Figure 2.5: Gaussian second order derivative in the y-direction and its approximation using box filters [11].

Let us denote the approximations by D_{xx}, D_{xy}, and D_{yy}. The relative weights in the calculation of determinant of Hessian need to be weighted by 0.9, which yields

$$\det(\mathcal{H}) = D_{xx} * D_{yy} - (0.9 * D_{xy})^2.$$

Due to the use of box filters and integral images, any size of the box filter can be applied to the original image at the same speed directly. Therefore, the scale space is created by the use of up-scaled filters of sizes 9×9, 15×15, 21×21, 27×27, etc. For each octave, the difference between filter sizes is doubled (from 6 to 12 to 24).

As the box filter layout remains the same, the corresponding Gaussian filter scales accordingly. The 9×9 box filter corresponds to a Gaussian filter with the scale $\sigma = 1.2$. Based on the definition of the process, we can calculate the corresponding Gaussian filter scale for each box filter size.

To select the key-points, non-maximum suppression in the $3 \times 3 \times 3$ neighborhood of each point is applied. Afterwards, the maxima of the determinant of the Hessian matrix are interpolated the same way as in the SIFT algorithm (using quadratic Taylor expansion).

2.2.2 Key-point Description

At first, we need to ensure the descriptor's invariance to rotation. For every key-point, we calculate the Haar-wavelet responses in the x and the y direction in a circular neighborhood of $6s$, where s is the Gaussian filter scale at which the key-point was found. Integral images are used for speeding up the process.

The wavelet responses are weighted with a Gaussian($\sigma = 2.5s$) centered at the key-point. The weighted responses are represented as vectors in a space with the x response being a vector along the x-axis, and the y response being a vector along the y-axis. All the vectors within a sliding window of $\frac{\pi}{3}$ are summed and the longest of the vector is selected as the key-point orientation.

An example of key-points and their orientation, found by OpenCV SURF implementation, is presented in Figure 2.6.

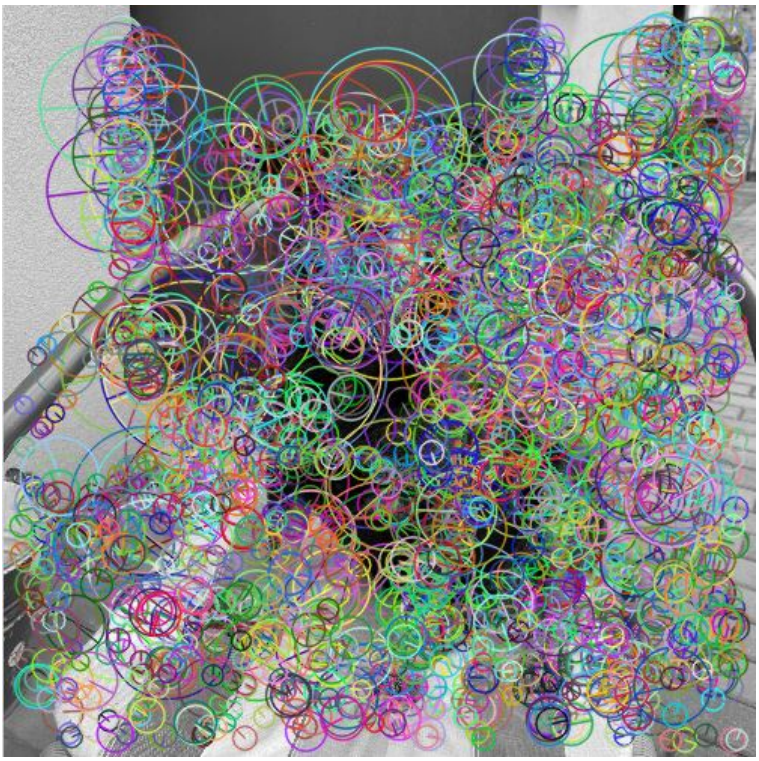


Figure 2.6: SURF key-points and their orientation from an image of a dog.

The descriptor is constructed from a square window the size of $20s$ around a key-point. The window is subsequently rotated along with the key-point orientation. Afterwards, this window is divided into 16 (4×4) square sub-windows. In each sub-window, the features are calculated using 5×5 regularly spaced sample points. From these sample points, we calculate the Haar wavelet responses in the horizontal and the vertical direction, where "horizontal" and "vertical" are defined in relation to the key-point orientation. The responses are weighted with a Gaussian ($\sigma = 3.3s$) centered at the key-point.

Let us call the horizontal responses d_x and the vertical responses d_y. Over each sub-window, we denote the sum of the responses $\sum d_x$ and $\sum d_y$, and the sum of absolute values of the responses $\sum |d_x|$ and $\sum |d_y|$. The behavior of these values for different image patterns can be seen in Figure 2.7. Combining $\sum d_x$, $\sum d_y$, $\sum |d_x|$, and $\sum |d_y|$ for each of the 16 sub-window into a vector, we get a descriptor of length 64.

2.3 Oriented FAST and Rotated BRIEF

Oriented FAST and Rotated BRIEF (ORB) is a local feature extractor proposed by Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski in 2011 [12].

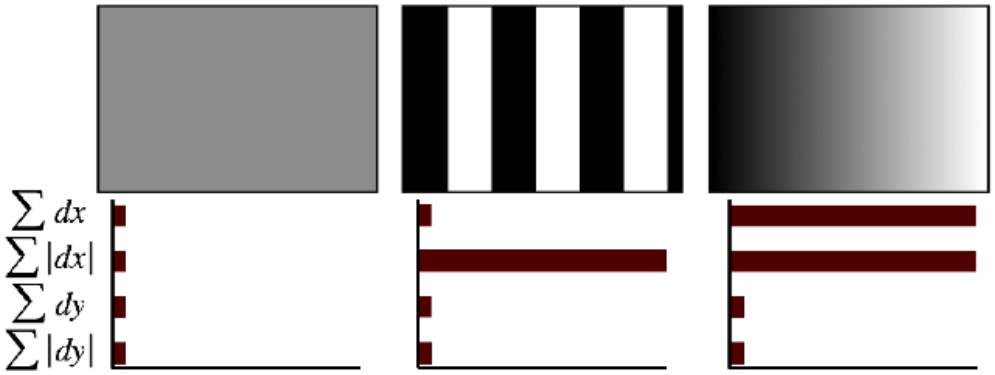


Figure 2.7: Behaviour of SURF descriptor for different image patterns [11].

The algorithm is based on the FAST (Features from Accelerated Segment Test) corner detector [13] and the BRIEF (Binary Robust Independent Elementary Features) descriptor [14].

2.3.1 Key-point Detection

The ORB algorithm uses the FAST-9 variant of the FAST corner detector. This detector compares each pixel intensity (denoted as I_p) in an image with the intensities of pixels in a circle of radius 9 around the pixel.

Let $n \in \mathbb{N}$ and the threshold $t \in \mathbb{R}^+$ be given. Let S be a set of pixels in a circle of radius 9 around the examined pixel. Let us denote I_x as the intensity of a pixel x. The examined pixel is selected as a corner if there exists a set $S_n \in S$ of n contiguous pixels, where $\forall x \in S_n : I_x + t < I_p$, or $\forall x \in S_n : I_x - t > I_p$.

The selected corners, i.e., key-points are then ordered according to a Harris corner measure [5]. For N key-points, the threshold is selected low enough to get more than N key-points. The best N key-points (according to the Harris corner measure) are then selected.

To find multi-scale features, the scale pyramid of an image is generated and key-points are located at each level in the pyramid.

2.3.2 Key-point Description

To ensure the descriptor's invariance to rotation, we need to determine the key-point orientation. To achieve this goal, the intensity centroid [15] is used. The intensity centroid expects the intensity of a key-point to be offset from its center. The vector from the center to the centroid is used for the key-point orientation.

Given image $I(x, y)$, a moment of a patch is defined as

$$m_{pq} \coloneqq \sum_{x,y} x^p y^q I(x, y),$$

and the centroid is located as

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right),$$

from which we can obtain the key-point orientation

$$\theta = \text{atan2}(m_{01}, m_{10}),$$

where atan2 is the quadrant-aware version of arctan.

An example of key-points and their orientation detected by OpenCV ORB implementation, can be seen in Figure 2.8.

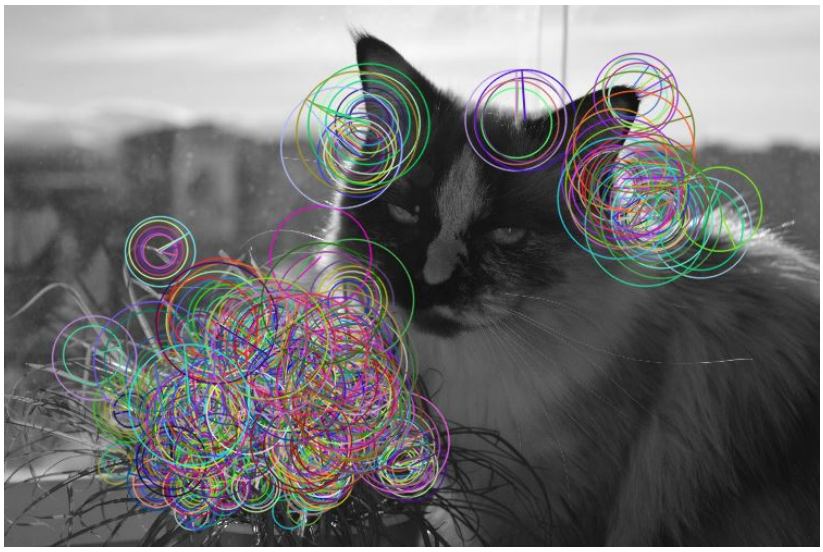


Figure 2.8: ORB key-points and their orientation from an image of a cat.

As an ORB descriptor, a variation on the BRIEF descriptor is used. The BRIEF descriptor is a vector of binary values. This allows for fast matching using a Hamming distance.

The descriptor values are computed by comparing random pairs of pixel intensities in a patch. The binary vector is based on the test defined as

$$\tau(\boldsymbol{p}; x, y) = \begin{cases} 1 & \text{if } \boldsymbol{p}(x) < \boldsymbol{p}(y), \\ 0 & \text{otherwise,} \end{cases}$$

where $\boldsymbol{p}(x)$ is the pixel intensity of a patch $\boldsymbol{p}$ at a point x. The pairs of points x and y are from a random predetermined set

$$\boldsymbol{S} = \begin{pmatrix} x_1 \dots x_n \\ y_1 \dots y_n \end{pmatrix},$$

where n is the size of our descriptor. The BRIEF descriptor is defined as a vector of n binary tests

$$f_n(\boldsymbol{p}) \coloneqq \sum_{i=1}^{n} 2^{i-1} \tau(\boldsymbol{p}; x_i, y_i).$$

The steered version of the BRIEF descriptor, according to the orientation θ of the key-point, can be created by using a corresponding rotation matrix $\boldsymbol{R}_\theta$:

$$\boldsymbol{S}_{\boldsymbol{\theta}} = \boldsymbol{R}_\theta \boldsymbol{S}.$$

The sets of pairs are precomputed in a lookup table for discretized θ in increments of $\frac{2\pi}{30}$ to improve speed. In the end, the steered BRIEF descriptor becomes

$$g_n(\boldsymbol{p}, \theta) \coloneqq f_n(\boldsymbol{p}) | (x_i, y_i) \in \boldsymbol{S}_\theta.$$

2.4 Bag of Visual Words

Local feature extractors provide us with varied number of descriptors for each image. However, for classification, we require each image to be represented by a single vector. This can be achieved using a Bag of Words technique.

Bag of Words (BoW) is a technique, which represents each sample in a dataset as a multi-set of its words. In general, these words do not have to be literal words but can be any categorization of information provided by the sample.

The first step of BoW is the dictionary generation. The dictionary is a set of $k \in \mathbb{N}$ categories of words called bins. Providing the dictionary, the BoW assigns each word to a bin. Finally, for each sample in the dataset, a histogram of frequencies of words belonging to each bin is calculated.

As an example, we explain the technique with text documents. With text documents, the dictionary consists of every unique word in all documents. The words of each document are assigned to a bin consisting of said words.

For example, in a dictionary consisting of bins [“Peter”, “run”, “talked”, “to”], a sample sentence “Peter talked to Peter” would produce the following histogram:

$$[2, 0, 1, 1].$$

In our case, we consider the descriptors found in an image as words. Since we are dealing with visual information, let us call these words “visual words” and designate the BoW technique “Bag of Visual Words” (BoVW).

We generate the categories using a k-means algorithm on all the visual words in the training dataset. Afterwards, we assign the visual words to the closest (considering euclidean distance) category and compute the histogram of frequencies.

2.4.1 k-means

The k-means algorithm is an unsupervised learning technique [16]. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a dataset of n data-points. The goal is to assign the data points into k clusters $S = \{S_1, S_2, \ldots, S_k\}$, such that

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2 ,$$

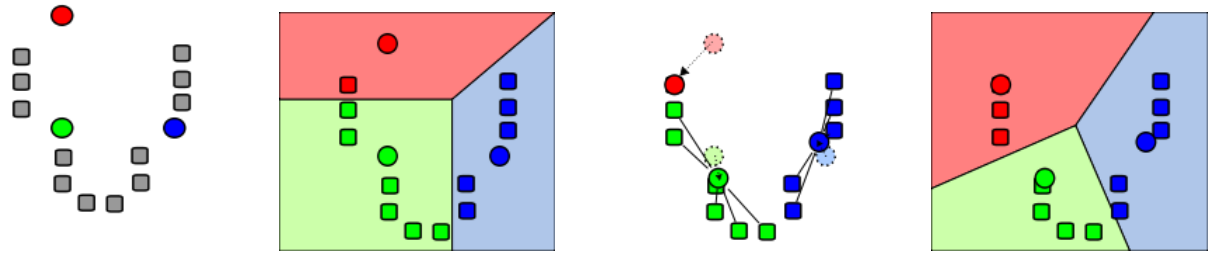
where $c_1, c_2, ..., c_k$ are the centroids of corresponding clusters $S_1, S_2, ..., S_k$. The centroids are determined as means of data points belonging to each cluster.

The algorithm starts by selecting k random data points as the centroids. The iterative process is based on the alternation between the following two consecutive steps until a termination condition is met:

Assignment step: Each data point is assigned to the cluster, such that the Euclidean distance between the centroids of the clusters and the data point is the smallest one.

Update step: For each cluster, a new centroid is computed as the mean value of data points belonging to this cluster.

The termination condition is satisfied, when the ratio of the samples, for which the assigned cluster changes, to the total amount of samples, is smaller than a specified threshold. The steps of the algorithm are demonstrated in Figure 2.9. The challenge is the determination of the number of



(a) At first, random data points are selected as the initial k centroids.

(b) By choosing the nearest centroid, each data point is assigned to a corresponding cluster.

(c) New centroids are calculated as mean values of data points in the cluster.

(d) Steps in (b) and (c) are repeated until the terminate condition is met.

Figure 2.9: Lloyd algorithm demonstration [17].

clusters k, the size of the visual dictionary. In our experiments, we examine different settings for k to find a clustering, which allows for satisfactory classification.

2.5 Principal Component Analysis

Principal Component Analysis (PCA) is a global feature extractor based on the data spectral properties [18]. It is used to reduce the dimension of data while preserving as much of the data variance as possible.

Let $x_t \in \mathbb{R}^n$ be a data point, where $t = 1, \ldots, T$ and T is the number of data points. The aim of the analysis is the determination of the optimal projector, which projects x_t onto $y_t \in \mathbb{R}^k$, where $k < n$. We want to find the optimal parameters P of the parametric reduction mapping $\psi_P : \mathbb{R}^n \to \mathbb{R}^k$ and the parametric reconstruction mapping $\psi_P^{-1} : \mathbb{R}^k \to \mathbb{R}^n$. The optimal parameters P^* minimize the reconstruction error, i.e.,

$$P^* := \arg\min_P \sum_{t=1}^{T} \left\| x_t - \psi_P^{-1}(\psi_P(x_t)) \right\|, \tag{2.2}$$

where ψ denotes the set of feasible parameters and $\left\| x_t - \psi_P^{-1}(\psi_P(x_t)) \right\|$ is the reconstruction error of a data-point.

PCA uses the linear mappings

$$\psi_{[Q,b]}^{-1}(y) := Qy + b, \quad \psi_{[Q,b]}(x) := Q^\top (x - b), \tag{2.3}$$

with parameters $b \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n,k}$ with orthonormal columns, i.e.,

$$Q^\top Q = I \in \mathbb{R}^{k,k}. \tag{2.4}$$

Substituting (2.3), (2.4) into the optimization problem (2.2) and using the square Euclidean norm for measuring the reconstruction error, we get the optimization problem

$$[Q^*, b^*] := \arg\min_{Q,b} \sum_{t=1}^{T} \left\| x_t - (QQ^\top (x_t - b) + b) \right\|_2^2 \quad \text{s.t.} Q^\top Q = I. \tag{2.5}$$

The objective function $f(Q, b)$ of (2.5) can be rewritten in the form

$$f(Q,b) = \sum_{t=1}^{T} (x_t^\top x_t - 2x_t^\top b + b^\top b - x_t^\top QQ^\top x_t + 2x_t^\top QQ^\top b - b^\top QQ^\top b), \tag{2.6}$$

and the optimality condition of the unconstrained variable b^* can be formulated as

$$\nabla_b f(Q,b) = \sum_{t=1}^{T} (-2x_t + 2b + 2QQ^\top xt - 2QQ^\top b) = 0,$$

which is equivalent to

$$(I - QQ^\top)\sum_{t=1}^{T}(b - x_t) = 0. \tag{2.7}$$

As $k < n$, $Q \in \mathbb{R}^{n,k}$ is not a full column rank and $QQ^\top \neq I$. The unique solution of (2.7) is

$$b^* = \frac{1}{T}\sum_{t=1}^{T} x_t.$$

Moreover, the Hessian matrix

$$\nabla^2_{b,b} f(Q, b) = 2(I - QQ^\top), \tag{2.8}$$

is a symmetric positive definitive matrix, therefore the objective function (2.6) is strictly convex (in the variable b) and (2.8) is a unique minimizer.

Let us denote the shifted data by

$$\hat{x_t} \coloneqq x_t - b^*, t = 1, \ldots, T$$

and write the objective function of (2.6) as

$$f(Q, b^*) = \sum_{t=1}^{T} \left\| \hat{x_t} - QQ^\top \hat{x_t} \right\|_2^2 = \sum_{t=1}^{T} (\hat{x_t}^\top \hat{x_t} - \hat{x_t}^\top QQ^\top \hat{x_t}). \tag{2.9}$$

We can simplify (2.9) using properties of the matrix trace:

$$f(Q, b^*) = \sum_{t=1}^{T} \mathrm{trace}(\hat{x_t}^\top \hat{x_t}) - \sum_{t=1}^{T} \mathrm{trace}(\hat{x_t}^\top QQ^\top \hat{x_t}) = \mathrm{trace}(\mathrm{cov}(x)) - \mathrm{trace}(Q^\top \mathrm{cov}(x) Q),$$

where $\mathrm{cov}(x)$ is the covariance matrix of x defined as

$$\mathrm{cov}(x) \coloneqq \sum_{t=1}^{T} (x_t - b^*)(x_t - b^*)^\top = \sum_{t=1}^{T} \hat{x_t} \hat{x_t}^\top.$$

The argument of the minimum is independent of additive constants in the objective function. Therefore, we can reformulate the optimization problem (2.5) (in terms of variable Q) as

$$\begin{aligned} [Q^*] &= \underset{Q^\top Q = I}{\arg\min}\, f(Q, b^*) = \underset{Q^\top Q = I}{\arg\min} -\mathrm{trace}(Q^\top \mathrm{cov}(x) Q) \\ &= \underset{Q^\top Q = I}{\arg\max}\, \mathrm{trace}(Q^\top \mathrm{cov}(x) Q) = \underset{\forall j : q_j^\top q_j = 1}{\arg\max} \sum_{j=1}^{k} q_j^\top \mathrm{cov}(x) q_j, \end{aligned} \tag{2.10}$$

where $q_j, j = 1, \ldots, k$ denote the (orthonormal) columns of the matrix $Q \in \mathbb{R}^{n,k}$. The Lagrange

function corresponding to the problem (2.10) is given by

$$L(Q, \lambda) \coloneqq \sum_{j=1}^{k} q_j^\top \mathrm{cov}(x) q_j - \sum_{j=1}^{k} \lambda_j (q_j^\top q_j - 1),$$

where $\lambda \in \mathbb{R}^K$ denotes Lagrange multiplier corresponding to equality constraints. The first Karush-Kuhn-Tucker conditions can be derived as

$$\nabla_{q_j} L(Q, \lambda) = 2\mathrm{cov}(x) q_j - 2\lambda_j q_j = 0, j = 1, \ldots, k,$$

which are the eigenvalue equations

$$\mathrm{cov}(x) q_j = \lambda_j q_j, j = 1, \ldots, k. \tag{2.11}$$

If we substitute (2.11) into objective function (2.10), we get

$$\sum_{j=1}^{k} q_j^\top \mathrm{cov}(x) q_j = \sum_{j=1}^{k} \lambda_j q_j^\top q_j = \sum_{j=1}^{k} \lambda_j.$$

Since the problem (2.10) is a maximization problem, the optimal Q^* consists of (orthonormal) eigenvectors which correspond to the k largest eigenvalues.

Chapter 3

Classifiers

For data classification in the thesis, we are using two classification techniques: the Bayes model and the Support Vector Machine (SVM). The goal of these techniques is to find a mapping from a feature space into a space of labels.

3.1 Bayesian Model

This classifier is suitable for classifying data represented by a stochastic vector. The BoVW data can be easily transformed into such vector. Instead of each component of the BoVW vector representing the number of key-points in the respective category, the component in our new vector represents the probability of key-points belonging to the respective category.

Let $x_t \in X := \{\mathsf{x}_1, \dots \mathsf{x}_{K_x}\}, t = 1, \dots, T$ be the input variables and let $y_t \in \mathsf{Y} := \{\mathsf{y}_1, \dots, \mathsf{y}_{K_Y}\}$ be output categorical variables. Let us denote the stochastic data vector $\Pi_{xt} \in \mathbb{R}^{K_x}, t = 1, \dots, T$, where T is the number of samples, K_x is the size of BoVW. Let the vector $\Pi_{yt} \in \mathbb{R}^{K_y}$ be a vector of probabilities, with which Π_{xt} belongs to each category, and K_y the number of categories.

Given a stochastic data vector Π_x, we can describe the transformation $\mathbb{R}^{K_x} \to \mathbb{R}^{K_y}$ using a matrix $\Delta \in \mathbb{R}^{K_y, K_x}$:

$$\Delta = \begin{bmatrix} P(y_t = \mathsf{y}_1 | x_t = \mathsf{x}_1) & P(y_t = \mathsf{y}_1 | x_t = \mathsf{x}_2) & \dots & P(y_t = \mathsf{y}_1 | x_t = \mathsf{x}_{K_x}) \\ P(y_t = \mathsf{y}_2 | x_t = \mathsf{x}_1) & P(y_t = \mathsf{y}_2 | x_t = \mathsf{x}_2) & \dots & P(y_t = \mathsf{y}_2 | x_t = \mathsf{x}_{K_x}) \\ \vdots & \ddots & & \\ P(y_t = \mathsf{y}_{K_y} | x_t = \mathsf{x}_1) & P(y_t = \mathsf{y}_{K_y} | x_t = \mathsf{x}_2) & \dots & P(y_t = \mathsf{y}_{K_y} | x_t = \mathsf{x}_{K_x}) \end{bmatrix},$$

where Π_x^n is the n-th element of Π_x, similar to Π_y^n, and the matrix Δ is a left stochastic matrix.

The determination of the optimal Δ^* can be written as

$$\Delta^* = \underset{\Delta \in \Omega_\Delta}{\arg\min} \sum_{t=1}^{T} \mathrm{dist}(\Pi_{yt}, \Delta \Pi_{xt}),$$

where Ω_Δ is a set of left stochastic matrices. The $\mathrm{dist}(\Pi_{yt}, \Delta\Pi_{xt})$ is calculated as Kullback-Leiber divergence [19]:

$$\mathrm{dist}(\Pi_{yt}, \Delta\Pi_{xt}) = -\sum_{i=1}^{K_y} \Pi_{yt}^i \ln \frac{[\Delta\Pi_{xt}]_i}{[\Pi_{yt}]_i} = -\sum_{i=1}^{K_y} [\Pi_{yt}]_i (\ln[\Delta\Pi_{xt}]_i - \ln[\Pi_{yt}]_i).$$

For the optimization, the term $\ln[\Pi_{yt}]_i$ is constant, therefore it can be ignored:

$$\mathrm{dist}(\Pi_{yt}, \Delta\Pi_{xt}) \propto -\sum_{i=1}^{K_y} [\Pi_{yt}]_i \ln[\Delta\Pi_{xt}]_i.$$

The analytical solution for this problem does not exist. However, $-\ln()$ is a convex function and $\Delta\Pi_{xt}$ is a convex combination, thus Jensen's inequality can be used:

$$-\sum_{i=1}^{K_y} [\Pi_{yt}]_i \ln[\Delta\Pi_{xt}]_i \leq -\sum_{i=1}^{K_y} [\Pi_{yt}]_i (\sum_{j=1}^{K_x} [\Pi_{xt}]_j \ln(\Delta_{ij})) = -\sum_{i=1}^{K_y}\sum_{j=1}^{K_x} [\Pi_{yt}]_i [\Pi_{xt}]_j \ln \Delta_{ij}.$$

From this estimation, we get an approximated optimization problem

$$\Delta^* = \arg\min_{\Delta \in \Omega_\Delta} -\sum_{t=1}^{T}\sum_{i=1}^{K_y}\sum_{j=1}^{K_x} [\Pi_{yt}]_i [\Pi_{xt}]_j \ln \Delta_{ij},$$

where

$$\Omega_\Delta = \{\Delta \in [0,1]^{K_y, K_x}, \forall j \in \{1, 2, \ldots, K_x\} : \sum_{i=1}^{K_y} \Delta_{ij} = 1\},$$

which is a feasible set of left stochastic matrices. The problem can be solved analytically.

Let Δ be the optimal stochastic matrix. There exist $\lambda_j \in \mathbb{R}^{K_x}$ such that

$$L(\Delta, \lambda) = -\sum_{t=1}^{T}\sum_{i=1}^{K_y}\sum_{j=1}^{K_x} [\Pi_{yt}]_i [\Pi_{xt}]_j \ln \Delta_{ij} + \sum_{j=1}^{K_x} \lambda_j (\sum_{i=1}^{K_y} \Delta_{ij} - 1)$$

is the Lagrange function. The Karush-Kuhn Tucker conditions are

$$\nabla_{\Delta_{\hat{i}\hat{j}}} L(\Delta_{\hat{i}\hat{j}}, \lambda) = -\frac{1}{\Delta_{\hat{i}\hat{j}}} \sum_{t=1}^{T} [\Pi_{yt}]_{\hat{i}} [\Pi_{xt}]_{\hat{j}} + \lambda_{\hat{j}} = 0, \tag{3.1}$$

and

$$\nabla_{\Delta_{\hat{j}}} L = \sum_{i=1}^{K_y} \Delta_{i\hat{j}} - 1 = 0. \tag{3.2}$$

From (3.1) and (3.2), we get the optimal Δ^* with components

$$\Delta^*_{\hat{i}\hat{j}} = \frac{\sum_{t=1}^{T}[\Pi_{yt}]_{\hat{i}}[\Pi_{xt}]_{\hat{j}}}{\sum_{i=1}^{K_y}\sum_{t=1}^{T}[\Pi_{yt}]_i[\Pi_{xt}]_{\hat{j}}}.$$

Another approach to finding the optimal Δ^* is optimizing the original problem

$$\Delta^* = \underset{\Delta\in\Omega_\Delta}{\arg\min} - \sum_{t=1}^{T}\sum_{i=1}^{K_y}[\Pi_{yt}]_i \ln[\Delta\Pi_{xt}]_i,$$

without using the Jensen inequality. Since the feasible set Ω_Δ is a closed convex set, the problem can be solved numerically using the Spectral Projected Gradient method [20].

We compare both approaches (the analytical solution using Jensen inequality and the numerical solution) in our experiments.

3.2 The Support Vector Machine

The Support Vector Machine (SVM) is a supervised learning classifier originally designed for binary classifications. It was introduced by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 [21].

Let $T := \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), ..., (\boldsymbol{x_n}, y_n)\}$, be the training dataset, where n is the number of the samples, $\boldsymbol{x_i} \in \mathbb{R}^m$, $i \in \{1, 2, \ldots, n\}$ are samples, and $y_i \in \{-1, 1\}$ are labels corresponding to samples $\boldsymbol{x_i}$. The classification model is represented by the means of the hyperplane H, defined such that:

$$H : \boldsymbol{\omega}^T\boldsymbol{x} - \widetilde{b} = 0,$$

where ω is the normalized normal vector of the hyperplane H, and

$$\widetilde{b} = \frac{b}{\|\omega\|}$$

is the bias from the origin.

At first, we consider linearly separable training data. The two classes of data are distinguished by two hyperplanes such that the distance between them is maximized. The region bounded by these hyperplanes is called the margin. The hyperplane that lies halfway between them is called the maximum-margin hyperplane. These hyperplanes are described by the following equation:

$$\boldsymbol{w}^T\boldsymbol{x} - \widetilde{b} = \pm 1. \tag{3.3}$$

Sample $x_i, i = 1, \ldots, n$ belongs to the positive class, i.e., $y_i = 1, i = 1, \ldots, n$, when

$$\boldsymbol{w}^T\boldsymbol{x_i} - \widetilde{b} \geq 1, \tag{3.4}$$

and the negative class, i.e., $y_i = -1, i = 1, \ldots, n$, when

$$\boldsymbol{w}^T \boldsymbol{x_i} - \widetilde{b} \leq -1. \tag{3.5}$$

The properties (3.4) and (3.5) can be combined into a single equation

$$y_i(\boldsymbol{w}^T \boldsymbol{x} - \widetilde{b}) \geq 1.$$

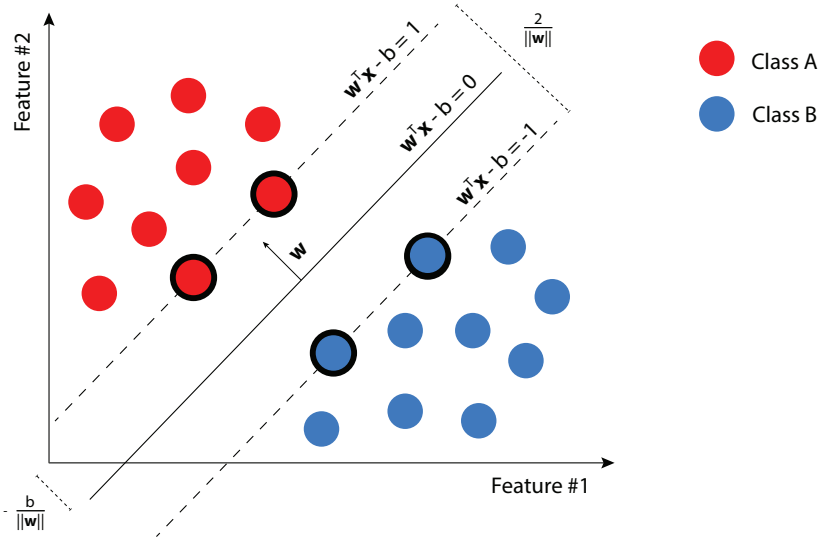


Figure 3.1: Example of the SVM model [22].

From the Figure 3.1, we can see, the distance between hyperplanes (3.3) is $\frac{2}{\|\boldsymbol{w}\|}$. Since we want to maximize this distance, we need to minimize $\|\boldsymbol{w}\|$. This leads to an optimization problem

$$\arg\min_{\boldsymbol{w},b} \|\boldsymbol{w}\| \quad \text{s.t.} \quad \begin{cases} y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b) \geq 1, \\ i = 1, ..., n, \end{cases}$$

which can be reformulated as the Quadratic Programming problem

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{s.t.} \quad \begin{cases} y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b) \geq 1, \\ i = 1, ..., n. \end{cases} \tag{3.6}$$

Because the training data is not commonly separable, the soft margin version of the SVM was proposed by Vladimir N. Vapnik and Corinna Cortes [21] in 1995. It exploits an additional function

called the hinge loss function:

$$\xi_i = \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b)). \tag{3.7}$$

The hinge loss function (3.7) equals 0 for a sample on the correct side of the corresponding hyperplane (3.3). However, for a sample on the wrong side of the corresponding hyperplane (3.3), the value of the function is proportional to the distance from the hyperplane.

If we add the hinge loss function (3.7) to optimization problem (3.6), we get a soft margin SVM optimization problem

$$\arg\min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{s.t.} \quad \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1, ..., n, \end{cases} \tag{3.8}$$

where $C \geq 0$ is a penalty for the misclassification error. The formulation (3.8) is called the $l1$-loss $l2$-regularized SVM. The primal formulation (3.8) can be modified using the Lagrange duality with Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_n]^\top$, and $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_n]^\top$ corresponding to inequality constraints. Exploiting Karush-Kuhn-Tucker conditions, we obtain the dual formulation

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{e} \quad \text{s.t.} \quad \begin{cases} \boldsymbol{o} \leq \boldsymbol{\alpha} \leq C\boldsymbol{e}, \\ \boldsymbol{B_e}\boldsymbol{\alpha} = 0, \end{cases} \tag{3.9}$$

where $\boldsymbol{e} = [1, 1, \ldots, 1]^\top$, $\boldsymbol{o} = [0, 0, \ldots, 0]^\top$, $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}]$, $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^\top$, $Y = \text{diag}(\boldsymbol{y})$, $\boldsymbol{B_e} = [\boldsymbol{y}^T]$, and $\boldsymbol{K} \coloneqq \boldsymbol{X}^T\boldsymbol{X}$ is the Gram matrix which is symmetric positive semi-definite (SPSD) [23]. The Hessian matrix in (3.9)

$$\boldsymbol{H} \coloneqq \boldsymbol{Y}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{Y}$$

is also an SPSD matrix.

To recover the normal vector, the formula

$$\boldsymbol{w} = \boldsymbol{X}\boldsymbol{Y}\boldsymbol{\alpha}.$$

is used. The bias b can be recovered as

$$b = \boldsymbol{w} \cdot \bar{\boldsymbol{x}} - y_i,$$

where $\bar{\boldsymbol{x}}$ is the mean of all support vectors.

Instead of the linear sum of the loss functions ξ_i in (3.8), we can use the square sum of the loss

functions in the objective function

$$\arg\min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \xi_i, \\ i = 1, ..., n. \end{cases} \tag{3.10}$$

The problem (3.10) is called $l2$-loss $l2$-regularized SVM. The dual formulation can again be obtained by using the Lagrange duality:

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{H} + C^{-1}\boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{\alpha}^T e \text{ s.t. } \begin{cases} \boldsymbol{o} \leq \boldsymbol{\alpha}, \\ \boldsymbol{B_e}\boldsymbol{\alpha} = 0. \end{cases}$$

In this case, the Hessian matrix $\boldsymbol{H}$, regularized by the matrix $C^{-1}\boldsymbol{I}$ is symmetric positive definite, therefore, this optimization problem should be more stable than the $l1$-loss $l2$-regularized SVM problem.

3.2.1 Hyperparameter optimization

Hyperparameter optimization is the process of selecting suitable parameters for a classifier. In SVM, we need to find an optimal penalty C. One approach of hyperparameter optimization is a Grid search.

Grid search is an exhaustive searching through a user-specified subset of parameters. The classifier is trained with each combination of the subset. The combination, which yields the best result is then selected. By the best result, we mean the highest score of a user-specified metric.

To test the hyperparameter selection, we need new, independent data from the data used in a training step. This is accomplished by splitting the data into subsets. The subsets are selected using a stratified cross-validation technique.

3.2.2 Cross-validation

Cross-validation is an approach to split a dataset into a training and validation subset [24]. For each set of hyperparameters, a dataset is split into k subsets (folds) of an equal size. One of the subsets is kept as a validation set, while the model is trained on the other $k - 1$ folds. The process is repeated k times, selecting each fold as a validation subset. The results of all k runs are averaged for a final score. This process is demonstrated in Figure 3.2.

Since a randomly selected fold might not represent the classes in the same ratio as the whole set, stratified cross-validation is used. The random folds are selected in a way, where the ratio of the classes is roughly equal to the ratio of classes in the whole dataset.

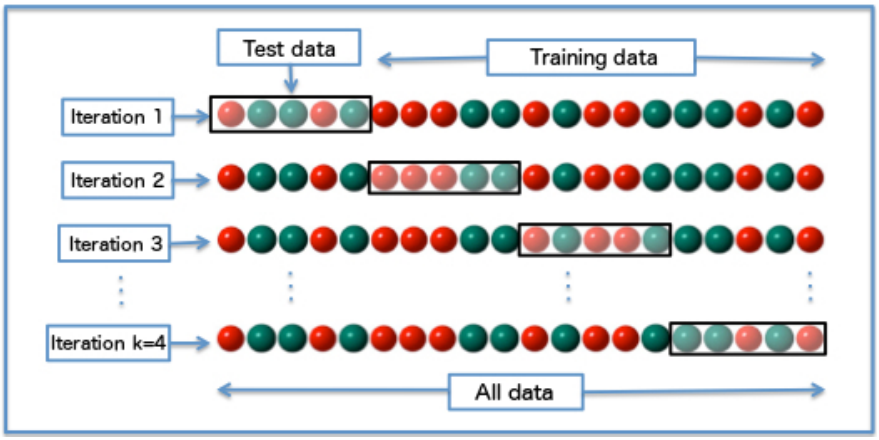


Figure 3.2: k-fold cross-validation splits data into k folds and trains the model k times always leaving out a different fold as a validation set [25].

3.3 Metrics

To assess the quality of the classification model, we need to analyze several metrics. Many of such metrics are derived from a confusion matrix (see Table 3.1). The confusion matrix is generated by

	Predicted negative	Predicted positive
Actual negative	True negatives (TN)	False positives (FP)
Actual positive	False negatives (FN)	True positives (TP)

Table 3.1: Confusion Matrix.

counting the testing data, which are supposed to belong to a negative class (Actual negative) or a positive class (Actual positive), and their predicted class (Predicted negative/Predicted positive). The diagonal confusion matrix represents the perfect classification.

We are considering the following metrics in our experiments:

Accuracy: How often is the classifier correct overall. This metric is represented in percentages.

$$\frac{TN + TP}{TN + FP + FN + TP}$$

Precision: How often is the classifier correct when it predicts positive.

$$\frac{TP}{TP + FP}$$

Sensitivity (Recall): How often is the classifier correct when it is actually positive.

$$\frac{TP}{TP + FN}$$

F_1 **Score:** A harmonic mean of precision and sensitivity.

$$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Generally, for each of these metrics, the higher value we get, the better the classification model.

Chapter 4

Datasets

To test our extraction-classification pipeline, we use three progressively more complex datasets. We start by classifying simple 2D shapes, then move on to simple 3D shapes, and last, we attempt to classify a dataset of real photographs of cats and dogs.

Each dataset is split into a training and a testing subset. This ensures that we can test, how the classifier performs on unseen data.

4.1 2D Shapes Dataset

For the simplest dataset, we use a four shapes dataset [26]. The dataset contains 16,000 images of four shapes: square, star, circle, and triangle. The dataset was created from poster board cutouts of each shape painted green. While rotating, each shape was recorded using a Garmin Virb 1080p action camera for two minutes. Shapes were then cropped out from the frames of the video and resized to 200×200 pixels. The green color of the cutouts in frames was then changed to a pure black color, while the rest of the image was changed to a pure white color. An example of the data can be seen in Figure 4.1.

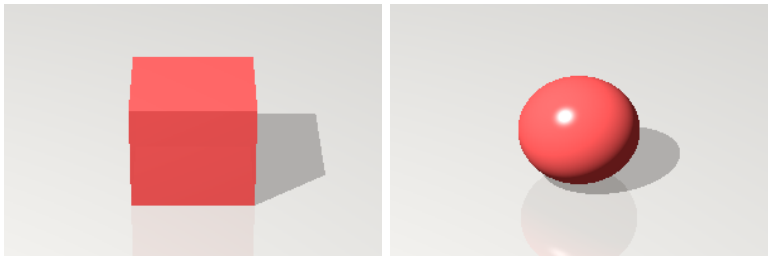
(a) Circle (b) Star

Figure 4.1: Photographs of a circle and a star from the Four Shapes dataset [26].

Since we are interested in binary classification, we use only the pair of a circle (positive class) and a star (negative class). We assign $^2/_3$ of random images from each class to a training subset, and $^1/_3$ to a testing subset.

4.2 3D Shapes Dataset

The 3D Shapes dataset contains 20 images of a box (positive class) and 20 images of a sphere (negative class) viewed from different angles. The shapes are colored red, while the background is colored white. The objects are illuminated by a single light source, providing a nice and sharp shadow and shading of each shape. The dataset was generated by Ing. Lukáš Pospíšil, Ph.D. using POV-Ray software [27]. Each of the images is 300×200 pixels. An example of each shape can be seen in Figure 4.2.



(a) Box (b) Sphere

Figure 4.2: 3D shapes, generated by Ing. Lukáš Pospíšil, Ph.D. in POV-Ray [27].

We assign $^2/_3$ of random images from each class to a training subset, and $^1/_3$ to a testing subset.

4.3 Cats and Dogs Dataset

The Cats and Dogs dataset is created from the Oxford-IIIT Pet Dataset [28]. Since the original dataset contains a few damaged images, we use a fixed version of the dataset from [29].

The dataset contains over 7,000 images of different cat and dog breeds, example of a cat and a dog image can be seen in Figure 4.3. The classes in the dataset are unbalanced, there are approximately twice as many images of dogs than images of cats.

The dataset provides us with a trimap for each image. The trimap consists of the information, which of the pixels are of the animal and which are of the background. We use these to cut out the animal from the background (set all the background pixel values to black). An example is demonstrated in Figure 4.4.

We assign the positive class to images of cats, and the negative class to images of dogs. In the original dataset,‘ we are provided with a training and testing subset. However, the ratio of training and testing samples in these classes is approximately $1 : 1$. To get a better representation of the

Figure 4.3: Example of images depicting a cat (left) and a dog (right) in the Oxford-IIIT Pet Dataset [28].

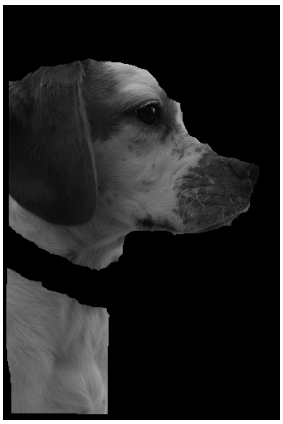


Figure 4.4: Original of a photograph of a beagle (left) and a cutout (right) [28].

data, we split the dataset ourselves, where we assign 90% of the data to a training subset and 10% of the data to a testing subset. The data is selected at random, making sure that for each breed of a cat or a dog, the ratio of amounts of training samples to testing samples is kept the same.

Chapter 5

Results

In this chapter, we take a look at results of our classification. The whole pipeline is run on the Salomon supercomputer at IT4Innovations [30]. Each task was computed on a single compute node. Each compute node contains two 2.5 GHz, 12-core Intel Xeon E5-2680v3 (Haswell) processors and 128 GB of memory.

5.1 Tools

The data preprocessing and feature extraction of the Cats and Dogs dataset are done using the tools-iiit-pet from our ml4py toolkit [31]. The tool for the data preparation and feature extraction of the other two datasets is strongly inspired by the tools-iiit-pet tool.

For the local features extraction, we use the implementation of SIFT, SURF, and ORB from the OpenCV library [32] and k-means implementation from the SciPy library [33]. The BoVW is created in Python.

The global feature extractor, PCA, is implemented using PETSc [34] and SLEPc [35] libraries.

The Bayesian Model classifier using the Jensen inequality is implemented in python. The Bayesian Model solved using the Spectral Projected Gradient method is trained using an Octave code provided by Ing. Lukáš Pospíšil, Ph.D., while the testing is done in python.

For the SVM classification, we use the PermonSVM [36]. This implementation of the SVM algorithm comes from the Department of Applied Mathematics, VSB-TUO, and the Institute of Geonics of the Czech Academy of Science in Ostrava, Czech Republic. It is build on top of a package for large scale quadratic programming, PermonQP. Both, PermonSVM and PermonQP, make use of the PETSc [34] library.

5.2 2D Shapes Dataset Classification

In the 2D Shapes classification, we attempt to train and classify images of a circle, labeled as the positive data, and images of a star, labeled as the negative data. See section 4.1 for details of benchmark formulation.

For local extractors, we examine the classification of BoVW with dictionary sizes $2, 3, 4, 6, 8, 12, 16$.

5.2.1 SVM

# of factors	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
12	0.72	0.63	0.44	0.44	0.61	0.61	7.66	794.15	0.44	5.30	0.01	10001	16

Table 5.1: 2D Shapes result for PCA extraction and SVM classification. "Ext" is shorthand for "Extraction".

In Table 5.1, we present results of the classification by the SVM model on data extracted using PCA. We keep only the first n eigenvectors, which can together explain at least 95% of the data variance. For the 2D Shapes dataset, we keep the first 12 eigenvectors.

Both, the l_1 and the l_2 SVM, classified the testing data quite well. However, by the number of iterations, we can see that the l_1-loss SVM model was unable to converge in less than $10,000$ iterations, and therefore it was stopped early.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.97	0.86	0.96	0.90	0.97	0.87	34.3	23.9	14.8	2.0	0.2	3467	308
3	0.99	0.92	1.00	0.96	0.99	0.92	33.9	29.8	33.2	0.3	0.2	636	262
4	0.92	1.00	0.85	1.00	0.92	1.00	34.4	8.2	8.3	0.2	0.1	410	227
6	1.00	1.00	1.00	1.00	1.00	1.00	33.8	4.2	3.7	0.1	0.1	172	152
8	1.00	1.00	1.00	1.00	1.00	1.00	35.4	4.9	5.0	0.1	0.1	265	148
12	1.00	1.00	1.00	1.00	1.00	1.00	36.5	4.2	3.4	0.1	0.1	149	144
16	1.00	1.00	1.00	1.00	1.00	1.00	36.4	4.5	4.0	0.1	0.1	144	132

Table 5.2: 2D Shapes results for extraction: SIFT and classification: SVM.

In Table 5.2, we show results of the classification by the SVM model on data extracted using SIFT. Since the data is quite simple, the classifier has no problem classifying the data for any size of the BoVW dictionary. However, both the SVM-l1 and SVM-l2, converged much faster for larger sizes of the BoVW dictionary. Moreover, we received perfect score for accuracy, precision, and F_1 from the BoVW sizes beyond the size of 6.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.96	0.96	0.92	0.91	0.96	0.95	10.4	6.0	10.2	0.2	0.2	385	353
3	0.74	0.72	0.48	0.44	0.65	0.61	10.3	10.3	11.5	0.3	0.3	645	500
4	0.97	0.98	1.00	1.00	0.98	0.98	10.3	19.2	19.4	0.3	0.3	614	643
6	0.98	0.98	1.00	1.00	0.98	0.98	10.4	14.2	13.9	0.1	0.1	306	307
8	1.00	1.00	1.00	1.00	1.00	1.00	10.6	11.1	11.4	0.1	0.1	281	280
12	1.00	1.00	1.00	1.00	1.00	1.00	11.1	9.8	9.9	0.1	0.1	226	225
16	1.00	1.00	1.00	1.00	1.00	1.00	11.2	8.5	8.7	0.1	0.1	210	209

Table 5.3: 2D Shapes results for extraction: SURF and classification: SVM.

In Table 5.3, we can see results of the classification by the SVM model on data extracted using SURF. Here, we can note, that for the size of BoVW dictionary 3, the classification turned out poorly. This might have been due to poorly located centroids by the k-means algorithm. However, for other sizes of BoVW dictionary, we get satisfactory, while mostly worse than in the case of extraction using SIFT, results.

Compared to SIFT, the extraction of SURF features runs much quicker. Even though the classification was in most cases slower, the time saved in extraction more than made up for it.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.89	0.90	0.90	0.89	0.89	0.90	16.0	57.2	18.3	4.4	0.4	8226	771
3	1.00	1.00	1.00	1.00	1.00	1.00	17.1	42.7	48.8	0.3	0.2	637	517
4	1.00	1.00	1.00	1.00	1.00	1.00	16.8	11.9	7.4	0.1	0.1	237	235
6	1.00	0.99	1.00	0.99	1.00	0.99	17.1	15.3	16.2	0.2	0.2	364	394
8	1.00	1.00	1.00	1.00	1.00	1.00	18.7	9.7	10.1	0.1	0.1	266	248
12	1.00	1.00	1.00	1.00	1.00	1.00	18.1	12.7	10.3	0.1	0.1	291	260
16	1.00	1.00	1.00	1.00	1.00	1.00	19.2	7.8	7.8	0.1	0.1	238	238

Table 5.4: 2D Shapes results for extraction: ORB and classification: SVM.

In Table 5.4, we can see results of the classification by the SVM model on data extracted using ORB. In this case, the classification turned out better than in the case of SURF, while the time complexity did not worsen by much.

5.2.2 Bayesian Model Classification

In Table 5.4, present results of the classification by the Bayesian model on data extracted using SIFT. In contrast with the classification using the SVM, the Bayesian model classifier has trouble

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	34.62	0.10	0.08	0.08	0.05	0.37	0.02	0.01	0.38
3	33.84	0.57	0.64	0.41	0.05	0.66	0.59	0.74	0.39
4	34.83	0.85	1.00	0.83	0.05	0.66	0.60	0.75	0.38
6	33.22	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.37
8	36.02	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.17
12	35.83	1.00	1.00	1.00	0.08	1.00	1.00	1.00	0.17
16	36.15	1.00	1.00	1.00	0.07	0.96	1.00	0.96	0.17

Table 5.5: 2D Shapes results for SIFT extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

classifying the data for smallest BoVW dictionary sizes. However, starting at the BoVW size 6, we received perfect or near perfect score for accuracy, precision, and F_1 with both models.

Looking at the training duration, we can see the advantage of the analytical solution using Jensen inequality compared to the numerical solution using Spectral Projected Gradient method. This trend continues across the different extraction techniques.

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	10.49	0.95	1.00	0.95	0.04	0.90	0.84	0.91	0.16
3	10.48	0.96	1.00	0.96	0.05	0.88	0.80	0.89	0.17
4	10.18	0.99	0.99	0.99	0.05	0.96	0.93	0.97	0.16
6	10.33	0.99	0.99	0.99	0.05	0.91	1.00	0.90	0.31
8	10.64	1.00	1.00	1.00	0.05	0.71	0.98	0.59	0.31
12	10.63	1.00	1.00	1.00	0.05	0.87	1.00	0.84	0.38
16	10.82	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.55

Table 5.6: 2D Shapes results for SURF extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

In Table 5.4, we present results of the classification by the Bayesian model on data extracted using SURF. With SURF, the Bayesian model using Jensen inequality performed well for all sizes of the BoVW dictionary. However, the Bayesian model using Spectral Projected Gradient performed quite poorly with the BoVW dictionary size of 8. We again suspect the reason to be poorly located centroids by the k-means algorithm.

In Table 5.4, we can see results of the classification by the Bayesian model on data extracted using ORB. It performed better than the classification of the extracted features using SIFT for the size 2 of the BoVW dictionary. Moreover, we can got a perfect classification for all the other sizes of the BoVW dictionary.

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	14.68	0.54	0.61	0.31	0.05	0.53	0.63	0.23	0.17
3	15.72	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.33
4	15.29	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.35
6	15.52	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.37
8	16.78	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.39
12	16.59	1.00	1.00	1.00	0.05	1.00	1.00	1.00	0.38
16	17.24	1.00	1.00	1.00	0.06	1.00	1.00	1.00	0.33

Table 5.7: 2D Shapes results for ORB extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

For our simple 2D Shapes dataset, the best classification is achieved using the analytical solution of the Bayesian model on the data extracted using ORB, with the BoVW dictionary size 3 or more.

From the standpoint of time complexity, the analytical solution of the Bayesian model shows clear advantage over the other methods. Since the SVM classifier was trained using parallelization on 24 cores, while both Bayesian models were run in linear, the advantage is even more substantial.

5.3 3D Shapes Dataset Classification

In the 3D Shapes classification, we move on to more complex images. We attempt classify images of a box, labeled as the positive data, and images of a sphere, labeled as the negative data. This dataset has much less data than the other two (only 20 images of each class). See section 4.2 for details of benchmark formulation.

Same as with the 2D shapes, for local extractors, we attempt the classification of BoVW with dictionary sizes $2, 3, 4, 6, 8, 12, 16$.

5.3.1 SVM

# of factors	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
11	0.79	0.79	0.71	0.71	0.77	0.77	0.30	4.36	1.12	0.17	< 0.01	465	11

Table 5.8: 3D Shapes result for PCA extraction and SVM classification. "Ext" is shorthand for "Extraction".

In Table 5.8, we present results of the classification by the SVM model on data extracted using PCA. For the 3D Shapes dataset, we keep the first 11 largest eigenvectors, which can explain at least 95% of the variance.

The training of both the SVM-l_1 and the SVM-l_2 converged in alloted number of iteration. However, the training of the l_1 classification model took magnitude more iteration steps. The classification score for both is acceptable.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.64	0.57	0.43	0.29	0.55	0.40	0.3	0.7	0.3	0.0	< 0.1	23	6
3	0.71	0.71	0.43	0.71	0.60	0.71	0.3	0.4	0.4	0.0	< 0.1	11	6
4	1.00	1.00	1.00	1.00	1.00	1.00	0.3	0.4	0.4	0.0	< 0.1	11	7
6	1.00	1.00	1.00	1.00	1.00	1.00	0.3	0.5	0.4	< 0.1	< 0.1	10	5
8	1.00	1.00	1.00	1.00	1.00	1.00	0.3	0.4	0.4	0.0	< 0.1	11	5
12	0.93	0.93	0.86	0.86	0.92	0.92	0.3	0.4	0.4	< 0.1	< 0.1	9	5
16	1.00	1.00	1.00	1.00	1.00	1.00	0.3	0.4	0.4	< 0.1	< 0.1	8	5

Table 5.9: 3D Shapes results for extraction: SIFT and classification: SVM.

In Table 5.9, we can see results of the classification by the SVM model on data extracted using SIFT. We achieved satisfactory results from classification of the BoVW with a dictionary size of 4 and above. Compared to the 2D dataset, we need bigger dictionary size, which can be explained by the increase in complexity of the data.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.57	0.64	0.29	0.43	0.40	0.55	0.1	nan	nan	0.0	0.0	29	8
3	0.07	0.07	0.00	0.00	nan	nan	0.1	nan	0.3	0.0	< 0.1	60	3
4	0.50	0.36	0.71	0.43	0.59	0.40	0.1	nan	0.4	0.1	< 0.1	114	4
6	0.93	1.00	0.86	1.00	0.92	1.00	0.1	0.7	0.7	0.0	0.0	69	11
8	0.93	0.93	0.86	0.86	0.92	0.92	0.2	0.5	0.5	0.0	0.0	18	15
12	1.00	1.00	1.00	1.00	1.00	1.00	0.1	0.5	0.6	0.0	0.0	13	18
16	1.00	1.00	1.00	1.00	1.00	1.00	0.2	0.6	0.5	0.0	0.0	20	15

Table 5.10: 3D Shapes results for extraction: SURF and classification: SVM.

In Table 5.10, we show results of the classification by the SVM model on data extracted using SURF. Here, both the $l1$-loss and the $l2$-loss SVM classifiers have trouble classifying the data for the size of BoVW dictionary smaller than 6. The "nan" of F_1 score for the size of BoVW 2 are due to none of the positive data (boxes) being classified as such. For the sizes of BoVW 2, 3, and 4, we came across technical difficulties with the hyperparameter optimization of the l_1 model. Same technical difficulties were encountered for the BoVW of size 2 in the case of l_2 model. As the hyperparameter optimization was not performed for these cases, we note the duration of the hyperparameter optimization as "nan".

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.93	0.86	1.00	0.86	0.93	0.86	0.2	0.5	0.4	0.0	< 0.1	25	7
3	0.57	0.71	0.71	0.57	0.63	0.67	0.2	0.7	0.6	0.0	< 0.1	25	9
4	0.50	0.57	0.43	0.29	0.46	0.40	0.2	0.6	0.6	0.0	< 0.1	24	8
6	0.29	0.43	0.43	0.43	0.38	0.43	0.2	0.6	0.6	0.0	< 0.1	22	10
8	1.00	1.00	1.00	1.00	1.00	1.00	0.2	0.7	0.6	0.0	< 0.1	20	10
12	0.86	0.86	0.71	0.71	0.83	0.83	0.2	0.6	0.5	0.0	< 0.1	15	8
16	1.00	1.00	1.00	1.00	1.00	1.00	0.2	0.6	0.5	0.0	< 0.1	12	10

Table 5.11: 3D Shapes results for extraction: ORB and classification: SVM.

In Table 5.11, we can see results of the classification by the SVM model on data extracted using ORB. Here, the results are even worse than classifying the data extracted using SURF, as the classifiers struggles with BoVW sizes smaller than 8.

For both, the SURF and ORB extracted data classification, the time complexity improvement is not worth the worse classification score.

5.3.2 Bayesian Model Classification

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	0.25	0.64	0.67	0.62	< 0.01	0.50	nan	nan	0.30
3	0.27	0.64	0.60	0.71	< 0.01	0.71	0.64	0.78	0.21
4	0.27	1.00	1.00	1.00	< 0.01	1.00	1.00	1.00	0.25
6	0.26	1.00	1.00	1.00	< 0.01	0.71	1.00	0.60	0.26
8	0.27	1.00	1.00	1.00	< 0.01	1.00	1.00	1.00	0.21
12	0.26	1.00	1.00	1.00	< 0.01	0.93	1.00	0.92	0.21
16	0.28	1.00	1.00	1.00	< 0.01	1.00	1.00	1.00	0.21

Table 5.12: 3D Shapes results for SIFT extraction and Bayesian model classification. “Acc” stands for accuracy and “Prec” stands for precision.

In Table 5.12, we present results of the classification by the Bayesian model on data extracted using SIFT. We can see, similar to classification using SVM, that for the BoVW size of 4 or more, the Bayesian classifier using Jensen inequality gives perfect results. However, the classifier using Spectral Projected Gradient method does not classify well for the BoVW of the size 6.

In Table 5.13 and Table 5.14 we can see results of the classification by the Bayesian model on data extracted using SURF and ORB respectively. Similar to the classification using SVM, the Bayes classifier struggles with the classification until larger sizes of BoVW.

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	0.18	0.57	0.67	0.40	< 0.01	0.64	0.75	0.55	0.25
3	0.17	0.57	1.00	0.25	< 0.01	0.07	0.00	nan	0.22
4	0.14	0.57	1.00	0.25	< 0.01	0.14	0.00	nan	0.22
6	0.18	0.86	1.00	0.83	< 0.01	0.86	1.00	0.83	0.21
8	0.18	1.00	1.00	1.00	< 0.01	1.00	1.00	1.00	0.25
12	0.15	1.00	1.00	1.00	< 0.01	1.00	1.00	1.00	0.33
16	0.20	1.00	1.00	1.00	< 0.01	1.00	1.00	1.00	0.30

Table 5.13: 3D Shapes results for SURF extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	0.21	0.86	0.86	0.86	< 0.01	0.46	0.33	0.12	0.34
3	0.21	0.64	0.75	0.55	< 0.01	0.57	0.67	0.40	0.30
4	0.23	0.64	0.75	0.55	< 0.01	0.57	0.67	0.40	0.35
6	0.21	0.71	0.80	0.67	< 0.01	0.57	1.00	0.25	0.36
8	0.20	0.86	1.00	0.83	< 0.01	0.79	1.00	0.73	0.29
12	0.22	0.86	1.00	0.83	< 0.01	0.93	1.00	0.92	0.33
16	0.22	1.00	1.00	1.00	< 0.01	1.00	1.00	1.00	0.34

Table 5.14: 3D Shapes results for ORB extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

On the 3D Shapes dataset, we can see an advantage of classifying data extracted using the SIFT extractor. While the SIFT extractor is slower than the SURF and the ORB extractors, the performance of the classification model is much better.

5.4 Cats and Dogs Dataset Classification

In the Cats and Dogs classification, we attempt to classify images of real photographs. The complexity of the data is much bigger than the previous two datasets. To concentrate only on classification using the data relevant to the animal, we classify images of cutouts of the animals as described in section 4.3.

5.4.1 SVM

In Table 5.15, we present results of the classification by the SVM model on data extracted using PCA. For the 3D Shapes dataset, we keep the first 81 largest eigenvectors, which can explain at

least 95% of the variance.

# of factors	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
81	0.53	0.69	0.47	0.18	0.39	0.27	46.31	nan	92.1	6.7	4.8	10001	7175

Table 5.15: Cats and Dogs result for PCA extraction and SVM classification. "Ext" is shorthand for "Extraction".

Due to technical difficulties, we were unable to perform hyperparameter optimization for the l_1-SVM model. Moreover, the l_1 model did not manage to converge in the alloted $10,000$ iterations. While the l_2-SVM model converged, it was not useful for the testing data classification, which can be seen from the classification score.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.86	0.86	0.58	0.57	0.73	0.73	137.6	144.8	136.6	1.0	0.7	1673	1128
4	0.93	0.99	0.80	0.96	0.89	0.98	145.6	486.8	703.0	1.3	2.0	2235	3410
8	0.65	1.00	1.00	1.00	0.65	1.00	156.6	161.9	157.1	0.9	0.4	1507	622
16	0.53	0.57	0.45	0.93	0.38	0.58	160.9	282.1	292.1	0.9	0.4	1349	546
32	0.74	0.90	1.00	0.68	0.71	0.81	190.8	347.2	356.6	0.3	0.5	406	600
64	0.33	0.65	1.00	1.00	0.49	0.65	221.4	288.4	274.5	0.2	0.4	294	483
128	0.95	0.86	0.92	0.57	0.92	0.72	305.5	411.4	426.3	0.6	0.4	791	494

Table 5.16: Cats and Dogs results for extraction: SIFT and classification: SVM.

In Table 5.16, we can see results of the classification by the SVM model on data extracted using SIFT. Because of the complexity of the data, we get much worse results than in the cases of the 2D Shapes and 3D Shapes datasets. However, the classification model is better than a random classifier, for most of the BoVW sizes.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.45	0.68	0.96	0.00	0.53	nan	43.2	136.2	nan	0.5	0.0	773	3
4	0.95	0.95	0.91	0.90	0.93	0.92	42.1	358.4	326.4	3.9	1.2	6608	2000
8	0.96	0.97	0.94	1.00	0.94	0.96	43.6	240.4	229.7	1.3	1.1	2111	1839
16	0.98	0.89	1.00	0.69	0.97	0.80	45.7	177.3	178.0	0.8	0.9	1193	1370
32	0.66	0.66	0.01	0.01	0.02	0.02	51.5	251.1	254.4	1.5	0.6	2025	817
64	0.68	0.68	0.00	0.01	nan	0.02	59.7	158.9	158.3	0.6	0.5	738	626
128	0.82	0.98	0.45	0.92	0.62	0.96	75.7	125.2	124.8	0.5	0.4	649	505

Table 5.17: Cats and Dogs results for extraction: SURF and classification: SVM.

In Table 5.17, we present results of the classification by the SVM model on data extracted using SURF. The classification model results are satisfactory for the BoVW of sizes 4 and 8. However, the higher sizes of BoVW dictionary show worse classification score.

Size of BoVW	Classification score						Elapsed time [s]					# Itter	
	Accuracy		Precision		F_1		Ext	HyperOpt		Training			
	l_1	l_2	l_1	l_2	l_1	l_2		l_1	l_2	l_1	l_2	l_1	l_2
2	0.68	0.68	0.00	0.00	nan	nan	61.6	176.1	128.4	0.6	0.7	949	1147
4	1.00	0.68	1.00	0.00	1.00	0.01	70.7	140.9	116.3	0.5	0.4	883	716
8	0.82	0.68	0.44	0.00	0.61	nan	73.7	82.8	80.1	0.3	0.2	539	358
16	0.86	0.68	0.57	0.00	0.73	0.01	83.5	97.7	94.6	0.4	0.3	587	518
32	0.68	0.68	0.00	0.00	0.01	nan	95.2	97.9	92.1	0.8	0.7	970	1042
64	0.98	0.89	0.95	0.65	0.97	0.79	122.8	132.9	129.8	0.5	0.5	688	691
128	0.95	0.68	0.84	0.00	0.92	nan	174.0	84.3	72.0	0.6	0.3	638	347

Table 5.18: Cats and Dogs results for extraction: ORB and classification: SVM.

In Table 5.18, we can see results of the classification by the SVM model on data extracted using ORB. The classifier behaves poorly in regard to precision and in regard to the F_1 score. In some cases, none of the data belonging to the positive class were classified as such, therefore value of the F_1 score cannot be calculated as it would lead to a division by zero. In the table, this is shown as a value "nan".

5.4.2 Bayesian Model Classification

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	187.04	0.94	1.00	0.90	0.06	0.87	1.00	0.75	0.41
4	206.78	1.00	1.00	1.00	0.06	1.00	1.00	1.00	0.11
8	215.17	1.00	1.00	1.00	0.06	0.91	1.00	0.85	0.36
16	218.44	0.83	1.00	0.63	0.07	0.71	1.00	0.20	0.38
32	268.46	1.00	1.00	1.00	0.09	0.68	1.00	0.01	0.73
64	314.75	0.93	1.00	0.89	0.08	0.85	1.00	0.71	0.44
128	438.37	0.86	0.71	0.82	0.10	0.82	1.00	0.63	1.03

Table 5.19: Cats and Dogs results for SIFT extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

In Table 5.19, we present results of the classification by the Bayesian model on data extracted using SIFT. The Bayesian model using Jensen inequality outperforms the SVM model in most cases. However, the Bayesian model using Spectral Projected gradients has trouble with BoVW sizes greater than 8 in regards to the F_1 score.

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	57.40	0.21	0.23	0.34	0.06	0.31	0.32	0.48	0.21
4	57.65	0.97	0.92	0.95	0.06	0.93	0.99	0.87	0.39
8	60.42	0.89	0.75	0.85	0.06	0.97	0.92	0.96	0.43
16	62.02	0.73	0.77	0.36	0.06	0.49	0.24	0.25	0.36
32	68.10	0.29	0.07	0.08	0.07	0.68	1.00	0.02	0.41
64	78.98	0.88	0.82	0.80	0.09	0.41	0.35	0.52	0.49
128	97.48	0.58	0.42	0.58	0.10	0.75	0.56	0.72	1.04

Table 5.20: Cats and Dogs results for SURF extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

In Table 5.20, we present results of the classification by the Bayesian model on data extracted using SURF. The testing subset classification score shows even worse drop in BoVW sizes greater than 8, than the classification trained on SIFT data.

Size of BoVW	Extraction duration [s]	Jensen inequality				Spectral Projected Gradient			
		Test score			Training duration [s]	Test score			Training duration [s]
		Acc	Prec	F_1		Acc	Prec	F_1	
2	91.34	0.66	0.06	0.01	0.08	0.64	0.07	0.01	0.37
4	95.14	0.69	1.00	0.08	0.06	0.85	1.00	0.69	0.40
8	100.60	0.68	0.45	0.04	0.06	0.22	0.15	0.19	0.37
16	110.63	0.84	1.00	0.67	0.07	0.68	nan	nan	0.42
32	128.97	1.00	1.00	1.00	0.07	1.00	1.00	0.99	0.35
64	162.12	1.00	1.00	1.00	0.08	1.00	1.00	1.00	0.39
128	231.27	1.00	1.00	1.00	0.12	1.00	1.00	1.00	0.98

Table 5.21: Cats and Dogs results for ORB extraction and Bayesian model classification. "Acc" stands for accuracy and "Prec" stands for precision.

In Table 5.21, we present results of the classification by the Bayesian model on data extracted using ORB. The model has perfect classification results for BoVW sizes greater than 16. This is true for both the Bayesian model using Jensen inequality and the model using Spectral Projected Gradient.

The Bayesian model with Jensen inequality shows perfect score for the data extracted using both SIFT and ORB. There is significant time complexity improvement of the ORB extraction over the SIFT extraction.

Chapter 6

Conclusion

The goal of the thesis was to explore different feature extraction and classification techniques, which can be used for visual signals classification. In the theoretical part of the thesis, we introduced such techniques with the fundamental mathematical background. In the practical part of the thesis, we introduced three progressively complex datasets for comparison of introduced methods.

We attempted to classify these datasets, after reducing their dimension by applying different feature extraction techniques, using different classifiers. In the final part of the thesis, we present and discuss the results of this classification.

For the feature extraction, we make use of a global feature extractor, i.e., PCA, and three local feature extractors, namely SIFT, SURF, and ORB. The classification itself was carried out by the SVM model and the Bayesian model classifiers. We used a numerical solution of the Bayesian model applying the Spectral Projected Gradient method, and an analytical solution of the model exploiting the Jensen inequality.

We started by applying this classification pipeline to the simplest dataset, i.e., the dataset of images of 2D shapes. Afterward, we moved onto a more complex dataset of images of 3D shapes. The last dataset we attempt to classify is the most complex Cats and Dogs dataset.

We run our experiments on the Salomon supercomputer at IT4Innovations. The analytical solution of the Bayesian model classifier was the fastest to train while providing similar classification results to the SVM model in most cases. While feature extraction using the SURF extractor was always the fastest, the classification of such data led to worse results than the classification of the data extracted by SIFT or ORB. While the use of SIFT and ORB lead usually to similar classification results, ORB had the advantage of lower time complexity. In our experiments, the data extracted using PCA was not suitable for the classification. We observed that this global feature extraction method is computationally the most demanding since it is based on working on the whole image. The computational complexity of solving the eigenvalue and eigenvector solver significantly dominates the whole process.

The advantage of our approach to the classification problem, over the more popular neural networks, is our ability to understand and examine each step. Additionally, the interpretation of the optimal parameters of a neural network is still not explored, and the meaning and the significance of individual neurons cannot be determined in a straightforward way. In the case of methods reviewed in this thesis, we were able to directly expound each parameter presented in every single step of the analysis. In the future, the hyperparameter optimization could be applied to the whole pipeline, from the extractor to the classifier parameters. It might also be beneficial to test different optimization algorithms in the SVM, allowing for faster convergence. In future work, it might be interesting to compare this classification pipeline directly to a neural network.

Bibliography

1. WIKIPEDIA. *Computer vision — Wikipedia, The Free Encyclopedia* [`http://en.wikipedia.org/w/index.php?title=Computer%20vision&oldid=894207185`]. [Online; accessed 28-April-2019].
2. DORŇÁK, Vojtěch. *Image feature extraction applied inclassification techniques* [`http://dspace.vsb.cz/handle/10084/136146`]. 2019. Bachelor's Thesis.
3. LEE, Yongjin; LEE, Kyunghee; PAN, Sungbum. Local and global feature extraction for face recognition. In: *International Conference on Audio-and Video-Based Biometric Person Authentication*. 2005, pp. 219–228.
4. LOWE, David G. Object recognition from local scale-invariant features. In: *Proceedings of the seventh IEEE international conference on computer vision*. 1999, vol. 2, pp. 1150–1157.
5. CHRIS HARRIS, Mike Stephens. A COMBINED CORNER AND EDGE DETECTOR. *Plessey Research Roke Manor*. 1988.
6. CANNY, John F. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986.
7. KOENDERINK, Jan J. The structure of images. *Biological cybernetics*. 1984, vol. 50, no. 5, pp. 363–370.
8. MIKOLAJCZYK, Krystian. *Detection of local features invariant to affines transformations*. 2002. PhD thesis. Institut National Polytechnique de Grenoble - INPG.
9. LOWE, David G. Distinctive Image Featuresfrom Scale-Invariant Keypoints. *International Journal of Computer Vision*. 2004.
10. BRADSKI, Gary. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.
11. BAY, Herbert; TUYTELAARS, Tinne; VAN GOOL, Luc. Surf: Speeded up robust features. In: *European conference on computer vision*. 2006, pp. 404–417.
12. RUBLEE, Ethan; RABAUD, Vincent; KONOLIGE, Kurt; BRADSKI, Gary. ORB: An efficient alternative to SIFT or SURF. In: *2011 International conference on computer vision*. 2011, pp. 2564–2571.

13. ROSTEN, Edward; DRUMMOND, Tom. Machine learning for high-speed corner detection. In: *European conference on computer vision*. 2006, pp. 430–443.

14. CALONDER, Michael; LEPETIT, Vincent; STRECHA, Christoph; FUA, Pascal. Brief: Binary robust independent elementary features. In: *European conference on computer vision*. 2010, pp. 778–792.

15. ROSIN, Paul L. Measuring corner properties. *Computer Vision and Image Understanding*. 1999, vol. 73, no. 2, pp. 291–307.

16. MACQUEEN, James et al. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. 1967, vol. 1, pp. 281–297. No. 14.

17. FLÖCK, Fabian. *K-fold cross validation* [online]. 2016 [visited on 2021]. Available from: `https://commons.wikimedia.org/wiki/File:K-fold_cross_validation_EN.jpg`.

18. PEARSON, Karl. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 1901, vol. 2, no. 11, pp. 559–572.

19. KULLBACK, Solomon; LEIBLER, Richard A. On information and sufficiency. *The annals of mathematical statistics*. 1951, vol. 22, no. 1, pp. 79–86.

20. BIRGIN, Ernesto G; MARTINEZ, Jose Mario; RAYDAN, Marcos. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*. 2000, vol. 10, no. 4, pp. 1196–1211.

21. CORINNA CORTES, Vladimir Vapnik. Support-vector networks. *Machine Learning*. 1995-09, vol. 20, no. 3, pp. 273–297. ISSN 1573-0565. Available from DOI: `10.1007/BF00994018`.

22. KRUŽÍK, Jakub; PECHA, Marek; HAPLA, Václav; HORÁK, David; ČERMÁK, Martin. Investigating Convergence of Linear SVM Implemented in PermonSVM Employing MPRGP Algorithm. In: KOZUBEK, Tomáš; ČERMÁK, Martin; TICHÝ, Petr; BLAHETA, Radim; ŠÍSTEK, Jakub; LUKÁŠ, Dalibor; JAROŠ, Jiří (eds.). *High Performance Computing in Science and Engineering*. Cham: Springer International Publishing, 2018, pp. 115–129. ISBN 978-3-319-97136-0.

23. MAREK PECHA, David Horák. Analyzing l1-loss and l2-loss Support Vector Machines Implemented in PERMON Toolbox. In: *Bioanalytical Reviews*. Springer Berlin Heidelberg, 2018-04, pp. 13–23. Available from DOI: `10.1007/978-3-030-14907-9_2`.

24. RODRÍGUEZ, Juan; PÉREZ, Aritz; LOZANO, Jose. Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 2010-04, vol. 32, pp. 569–575.

25. FLÖCK, Fabian. *K-fold cross validation* [online]. 2016 [visited on 2021]. Available from: https://commons.wikimedia.org/wiki/File:K-fold_cross_validation_EN.jpg.

26. SMESCHKE. *Four Shapes*. 2017. Retrieved 2021-04-01 from https://www.kaggle.com/smeschke/four-shapes.

27. VISION PTY. LTD. (2004), Persistence of. *Persistence of Vision Raytracer*. http://www.povray.org/download/.

28. PARKHI, Omkar M.; VEDALDI, Andrea; ZISSERMAN, Andrew; JAWAHAR, C. V. Cats and Dogs. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

29. PECHA, Marek. *Oxford IIIT Pet Dataset (fixed)*. GitHub, 2019. Available also from: https://github.com/ml4py/dataset-iiit-pet.

30. *IT4Innovations: National Supercomputing Center, VSB-Technical University of Ostrava, Salomon Cluster Documentation – Hardware Overview* [online]. 2017 [visited on 2017-07-25]. Available from: https://docs.it4i.cz/salomon-cluster-documentation/hardware-overview.

31. PECHA, Marek; DORNAK, Vojtech. *Oxford IIIT Pet Dataset (fixed)*. GitHub, 2019. Available also from: https://github.com/ml4py/toools-iiit-pet.

32. BRADSKI, Gary. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.

33. VIRTANEN, Pauli et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, vol. 17, pp. 261–272. Available from DOI: 10.1038/s41592-019-0686-2.

34. BALAY, Satish et al. *PETSc Web page* [https://www.mcs.anl.gov/petsc]. 2021. Available also from: https://www.mcs.anl.gov/petsc.

35. HERNANDEZ, Vicente; ROMAN, Jose E.; VIDAL, Vicente. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*. 2005, vol. 31, no. 3, pp. 351–362.

36. HAPLA, Václav; HORÁK, David; PECHA, Marek. *PermonSVM* [http://permon.vsb.cz/permonsvm.htm]. 2017.