Image Classification: Modern Approaches

Image Classification: Modern Approaches

Giacomo Boracchi, Alessandro Giusti
DEIB, Politecnico di Milano

February, 12th, 2018

giacomo.boracchi@polimi.it
home.deib.polimi.it/boracchi/

What is Image Classification About

Image Classification Problem

The problem: assigning to an *input image* I one *label* l from a *fixed set of categories* Λ

I ⇨ “wheel”

I ⇨ “castle”

Λ = {"wheel", "cars" ...."castle", "baboon", ... }

Image Classification Problem

The problem: assigning to an *input image I* one *label l* from a *fixed set of categories* Λ

I ⇨ "wheel" 65%, "tyre" 30%..

I ⇨ "castle" 55%, "tower" 43%..

Λ = {"wheel", "cars" ...."castle", "baboon", ... }

Extended IC Problems: Object Detection

Semantic Segmentation

Objects appearing in the image:

Boat | Dining table | Person

Zheng et al. “Conditional Random Fields as Recurrent Neural Networks”, ICCV 2015

Image Captioning

"little girl is eating piece of cake."

"black cat is sitting on top of suitcase."

Andrej Karpathy, Li Fei-Fei "Deep Visual-Semantic Alignments for Generating Image Descriptions" CVPR 2015

ImageNet

The **ImageNet project** is a large database visual object recognition.

It contains over **14 million hand-annotated images** in **over 20 thousand categories**

Since 2015 Imagenet organizes **ILSVRC** (ImageNet Large Scale Visual Recognition Challenge)

Classification error rate (top 5 accuracy):

- In 2011: 25%

Deep learning

- In 2012: 16% (achieved by a CNN)
- In 2017: < 5% (for 29 of 38 competing teams, deep learning)

Deep learning was boosted by:

- Advances in parallel hardware (e.g. GPU)
- Availablilty of large annotated dataset (e.g. ImageNet)

The Image Classification Course

Course Outline

Feb 12: **Introduction to IC** and **Basics of Image Handling**

Feb 14: IC by **hand-crafted features**, transform domain methods

Feb 16: IC and Object Recognition by **computer-vision features**

Feb 19: **Data-driven features**: CNN

Feb 21: **Advanced CNN** and best practices

Feb 23: **Extended Problems** and Projects

Goals

Understanding the challenges of handling images in a ML

Get an **algorithmic insight of image processing / analysis** techniques that are relevant for IC

Get an overview of **feature extraction** for IC

- hand-crafted features
- computer-vision features
- data-driven features

Understanding CNN and deep-learning architectures

Understand how to address **practical issues in IC**: performance assessment, dataset augmentation, transfer learning

Provide an **overview of IC tools in Python**

Code-sharing with Deep Learning

Date	Deep Learnig Classes (09:30-13:00)	Image Classification Classes (14:15-17:45)
12/02/2018	Introduction to Deep Learning, Classification and Feed Forward Neural Networks	Introduction to Image Classification and basics of image handling in Python
14/02/2018	Overfitting and regularization, gradient descent variations, tips & tricks	Hand-crafted features for image classification
16/02/2018	Recurrent neural networks, vanishing gradient issues, Long-Short Term Memories	Computer Vision features for image classification
19/02/2018	TensorFlow and PyTorch	Data-driven feature extraction and Convolutional Neural Networks
21/02/2018	Deep neural networks architectures for image classification and structural learning (with guests)	Advanced CNNs and Best practices in image classification
23/02/2018	Special guests: Variational Autoencoder, Shape Classification, Overview of DeepMind research.	An overview on extended problems in image classification

Code-sharing with Deep Learning

Date	Deep Learnig Classes (09:30-13:00)	Image Classification Classes (14:15-17:45)
12/02/2018	**Introduction to Deep Learning, Classification and Feed Forward Neural Networks**	**Introduction to Image Classification and basics of image handling in Python**
14/02/2018	**Overfitting and regularization, gradient descent variations, tips & tricks**	**Hand-crafted features for image classification**
16/02/2018	Recurrent neural networks, vanishing gradient issues, Long-Short Term Memories	**Computer Vision features for image classification**
19/02/2018	**TensorFlow and PyTorch**	**Data-driven feature extraction and Convolutional Neural Networks**
21/02/2018	**Deep neural networks architectures for image classification** and structural learning (with guests)	**Advanced CNNs and Best practices in image classification**
23/02/2018	Special guests: Variational Autoencoder, Shape Classification, Overview of DeepMind research.	**An overview on extended problems in image classification**

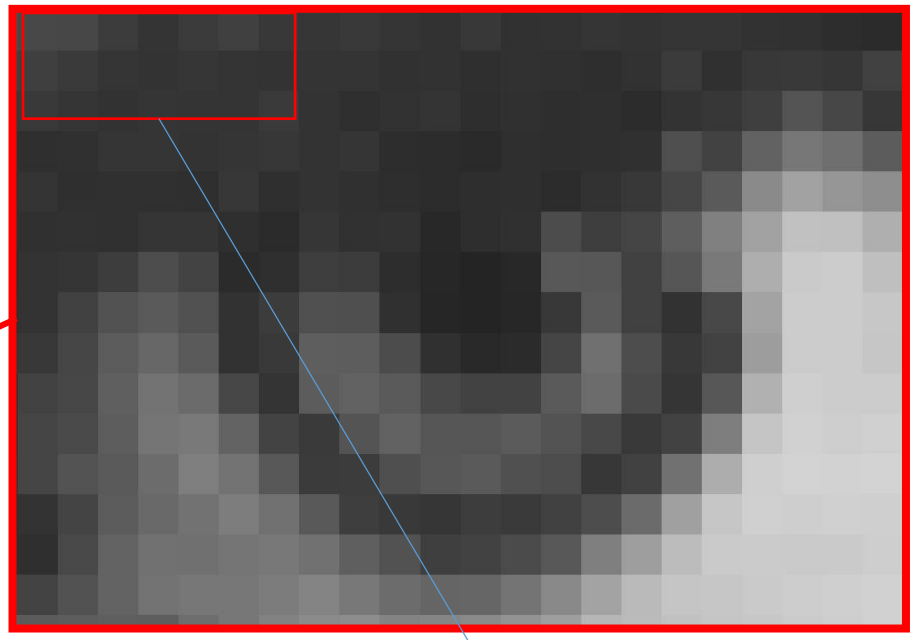
Outline

Materials are from:

Notes accompanying the Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Images

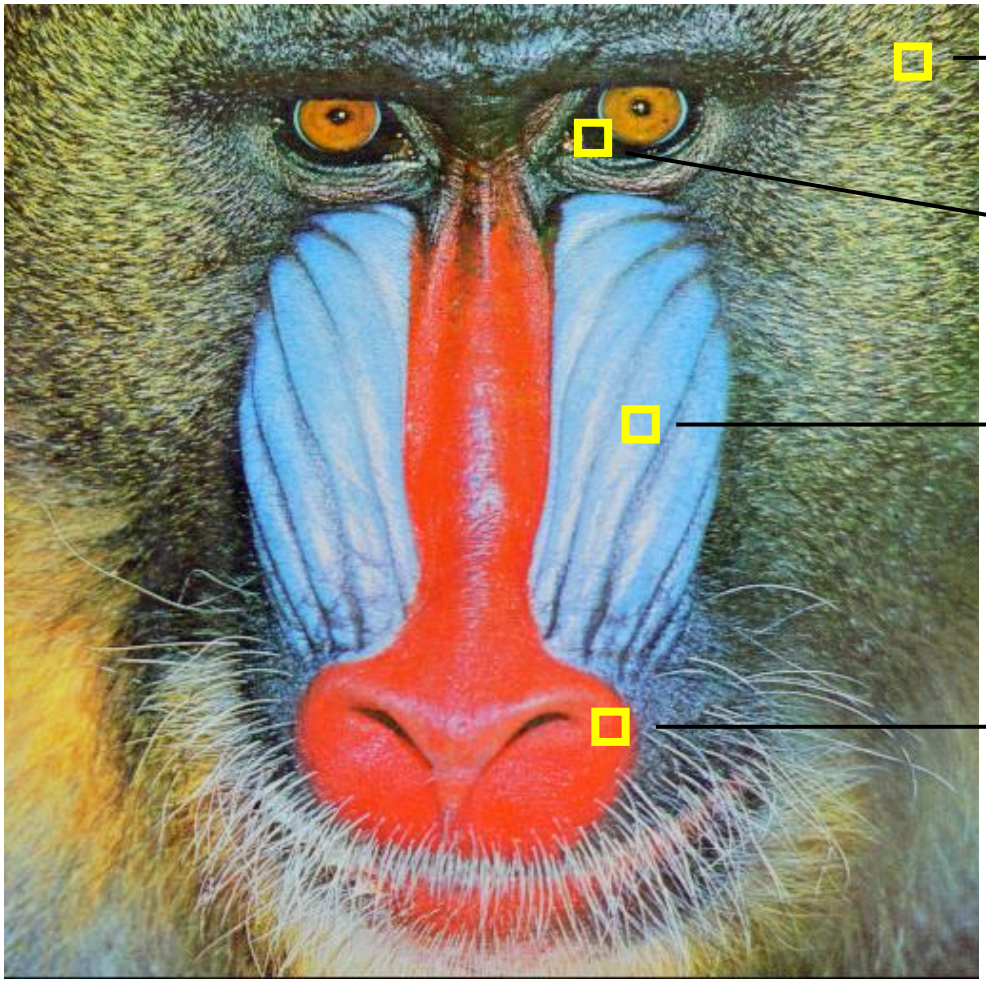
Images: the Classifier's input



123	122	134	121	132	133	145	134
122	121	125	132	124	121	116	126
119	127	137	119	139	127	128	131

x

RGB images



Green region

$I(r,c) = [120, \mathbf{150}, 30]$

Dark region

$I(r,c) = [40, 30, 11]$

Blue region

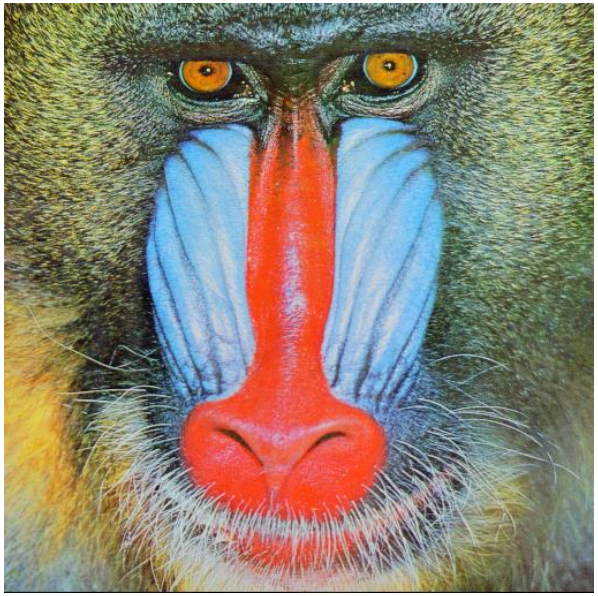
$I(r,c) = [100, 190, \mathbf{240}]$

Red region

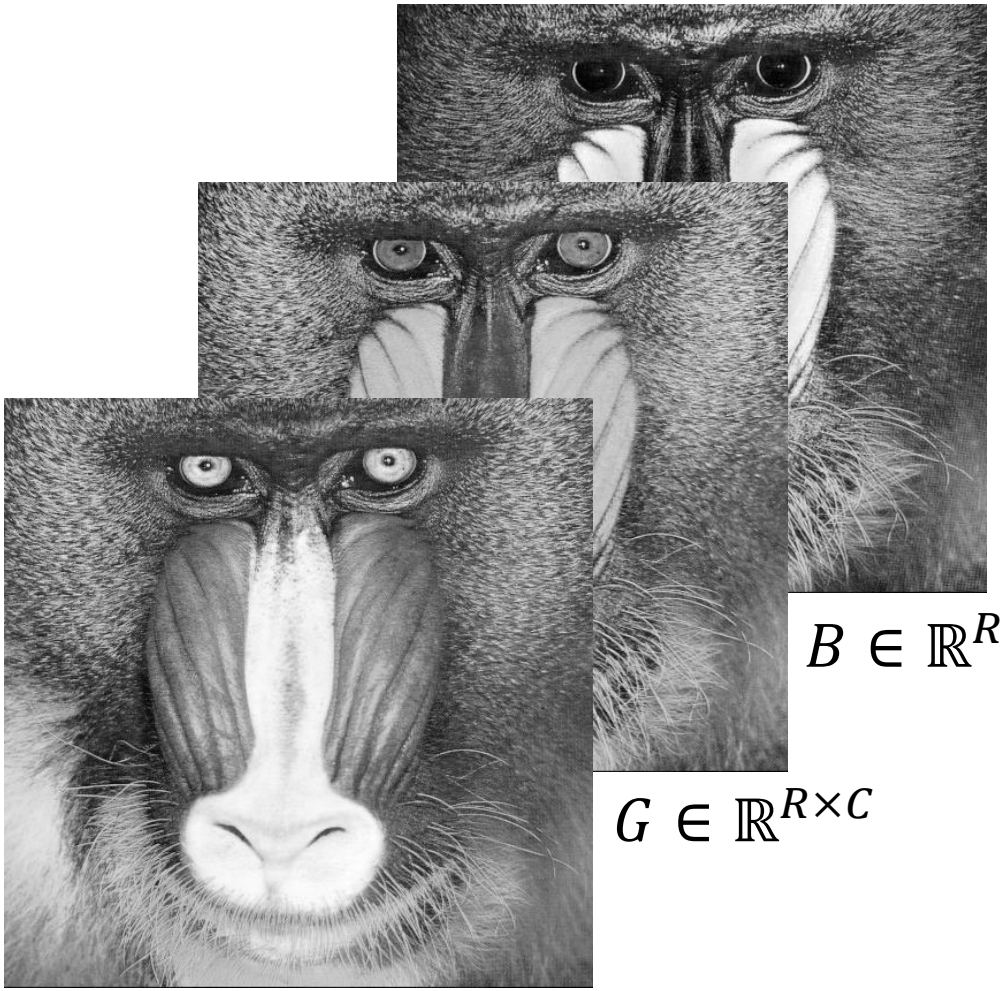
$I(r,c) = [\mathbf{240}, 80, 70]$

In practice, intensity values integers $[0 - 255]$

RGB images



$I \in \mathbb{R}^{R \times C \times 3}$



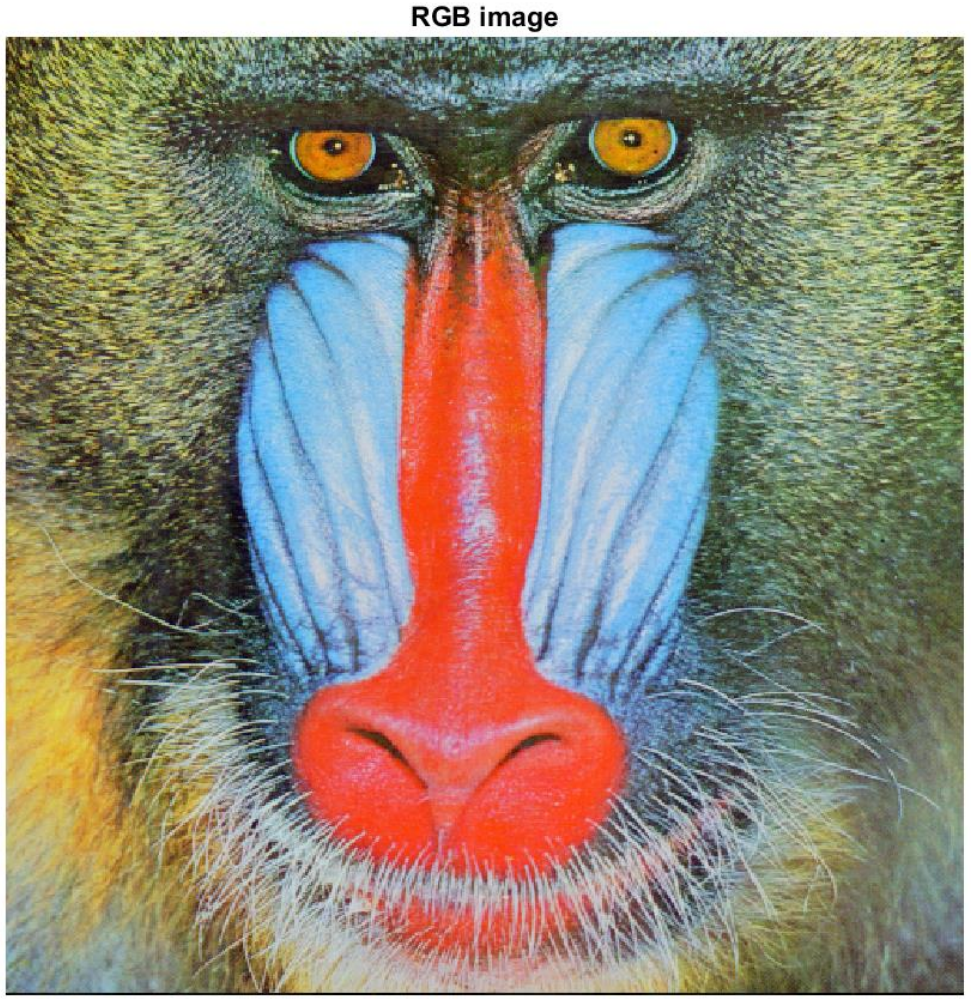
$B \in \mathbb{R}^{R \times C}$

$G \in \mathbb{R}^{R \times C}$

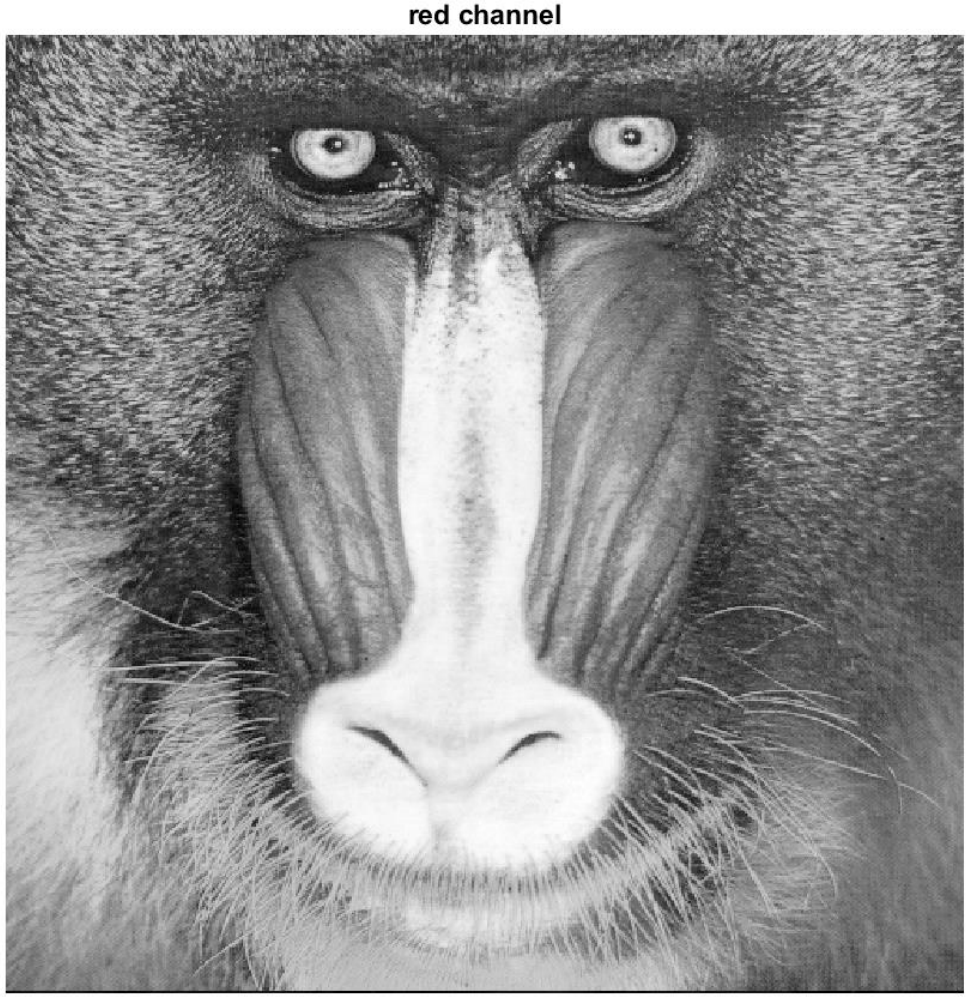
$R \in \mathbb{R}^{R \times C}$

This image is 512 x 512 pixels: $I \in \mathbb{R}^{512 \times 512 \times 3}$

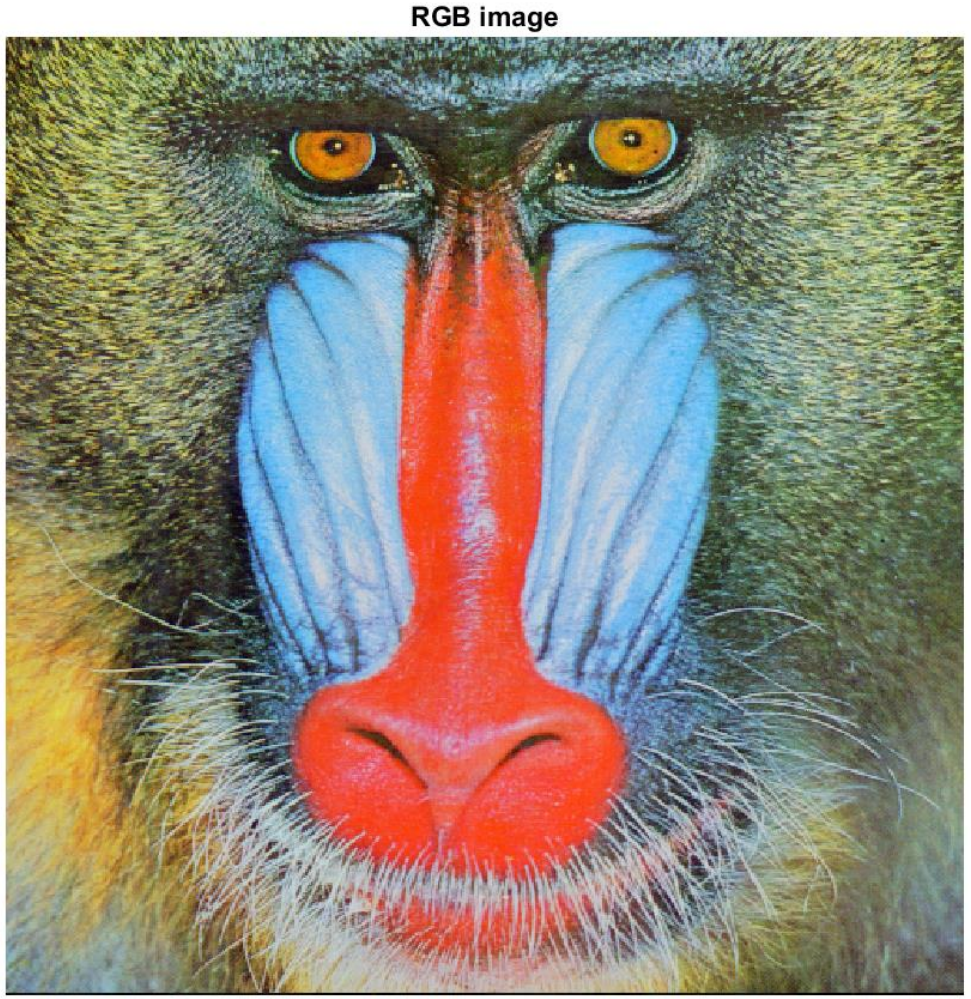
RGB image



Red Channel

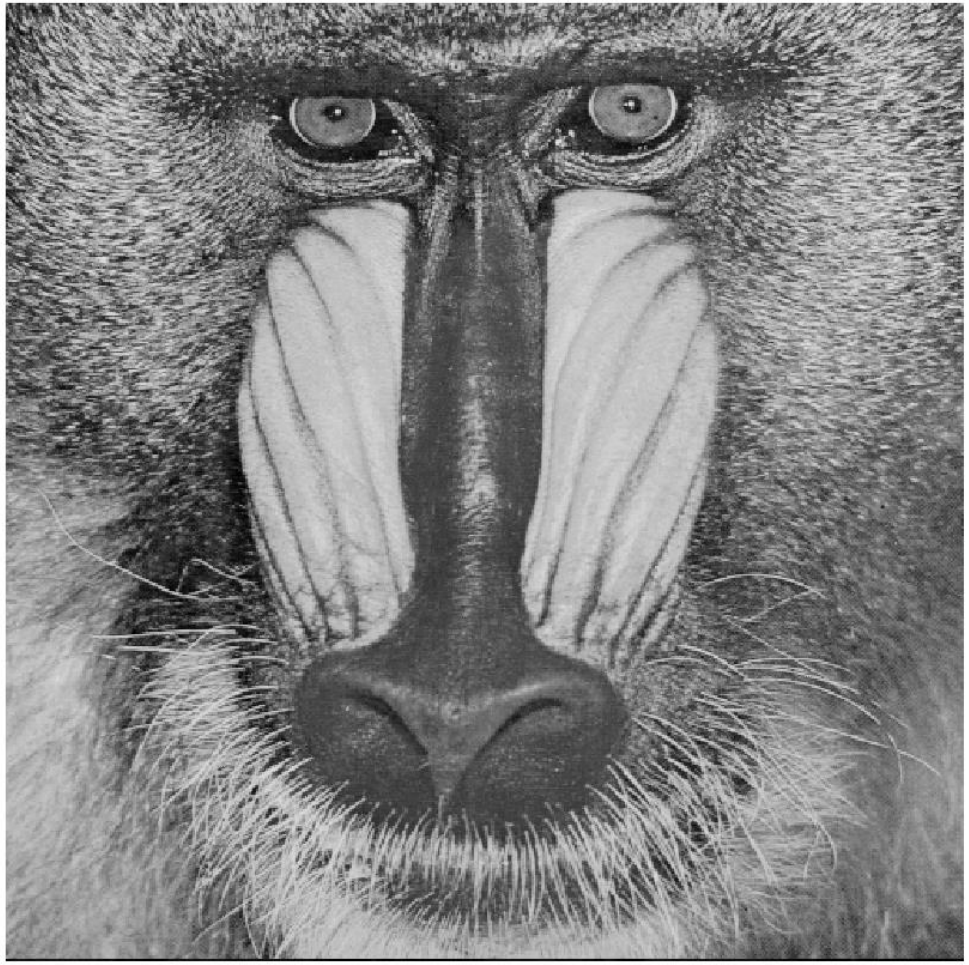


RGB image

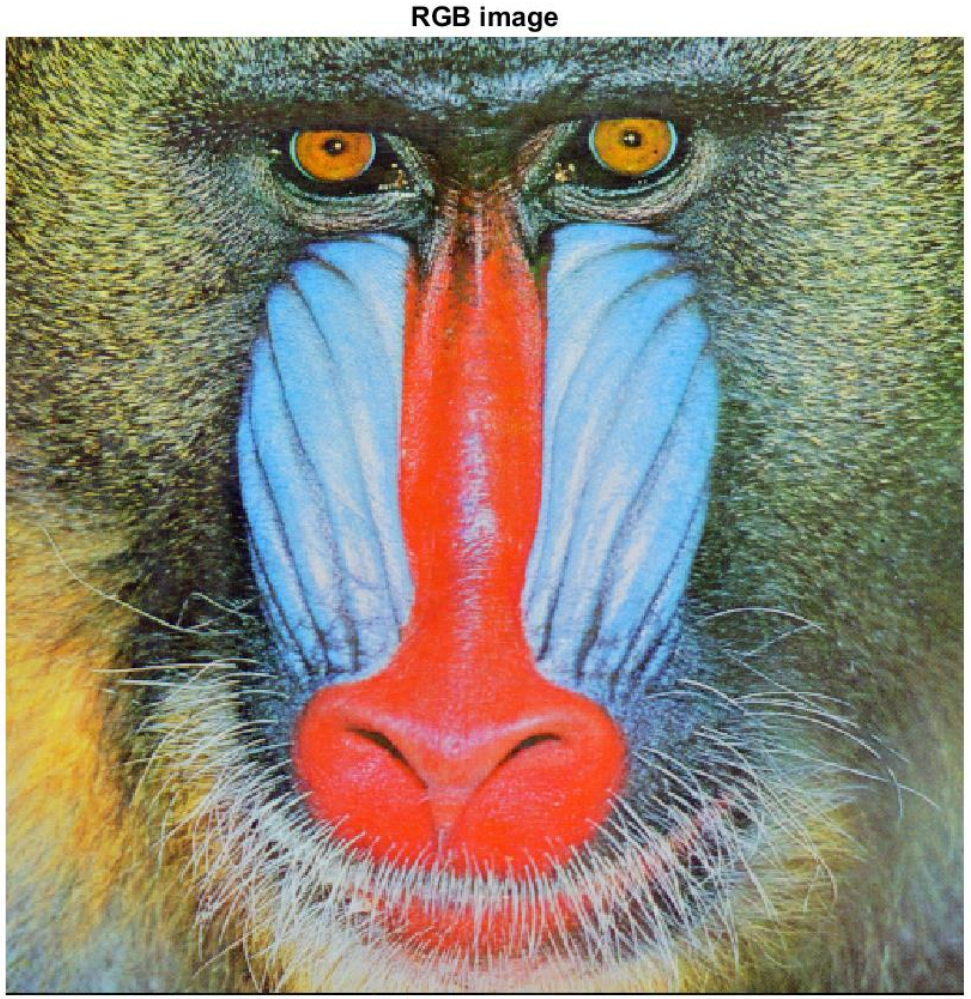


Green Channel

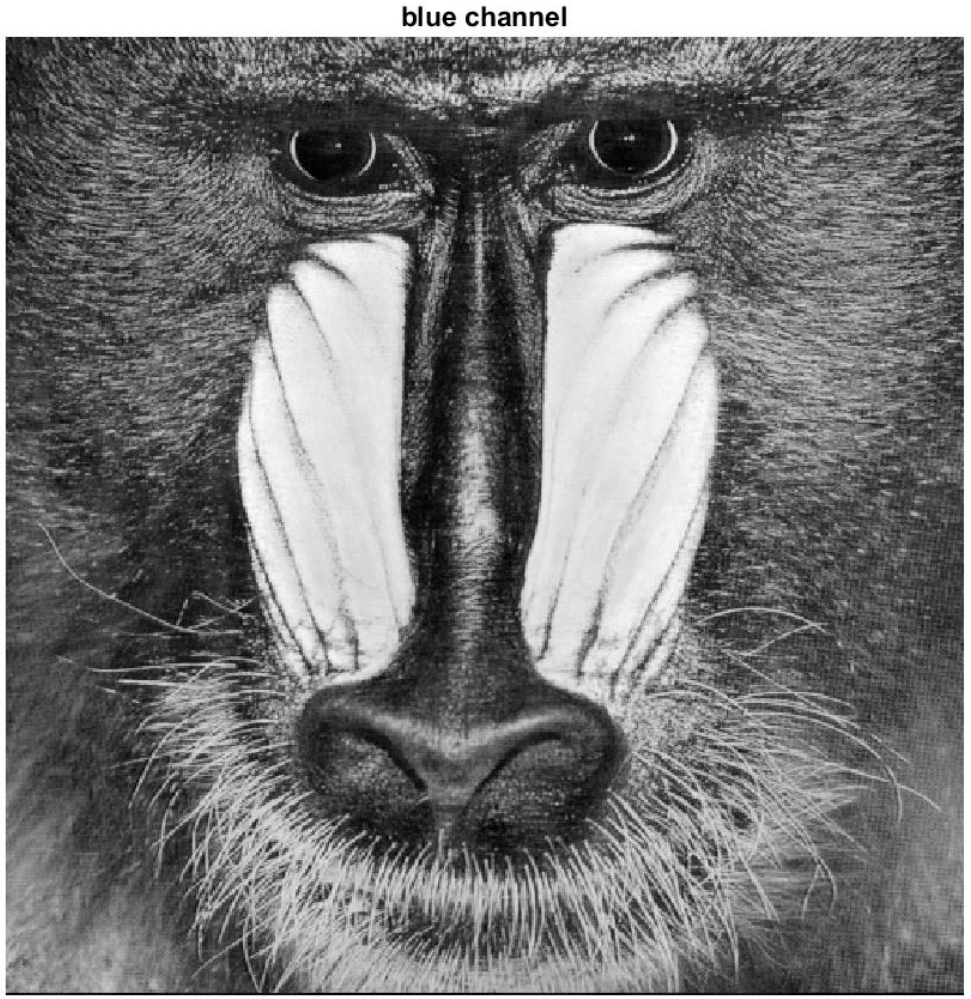
green channel



RGB image



Blue Channel



Higher dimensional images

Videos are sequences of images (frames)

If a frame

$$I \in \mathbb{R}^{R \times C \times 3}$$

a video of T frames

$$V \in \mathbb{R}^{R \times C \times 3 \times T}$$

In this example: $R = 144, C = 180$, thus these 5 color frames contains: 388.800 values in [0,255], thus in principle, 388 KB

Dimension Increases very quickly

Without compression: 1Byte per pixel

- 1 frame in full HD: $R = 1080, C = 1920 \approx 6MB$
- 1 sec in full HD (24fps) $\approx 150MB$

Fortunately, visual data are very redundant, thus compressible

This has to be taken into account when you design a Machine learning algorithm to be used on images

Higher dimensional Images

These images are stacking multiple layers as color-planes

Multi-spectral or Hyper-spectral images:

- each band covers a certain wavelength that is meaningful for interpretation soil in areal images

1. 0.45-0.52 µm Blue-Green
2. 0.52-0.60 µm Green
3. 0.63-0.69 µm Red
4. 0.76-0.90 µm Near IR
5. 1.55-1.75 µm Mid-IR
6. 10.40-12.50 µm Thermal IR
7. 2.08-2.35 µm Mid-IR

Classification of multispectral images: infer the soil coverage from the vaues in the different spectral bands

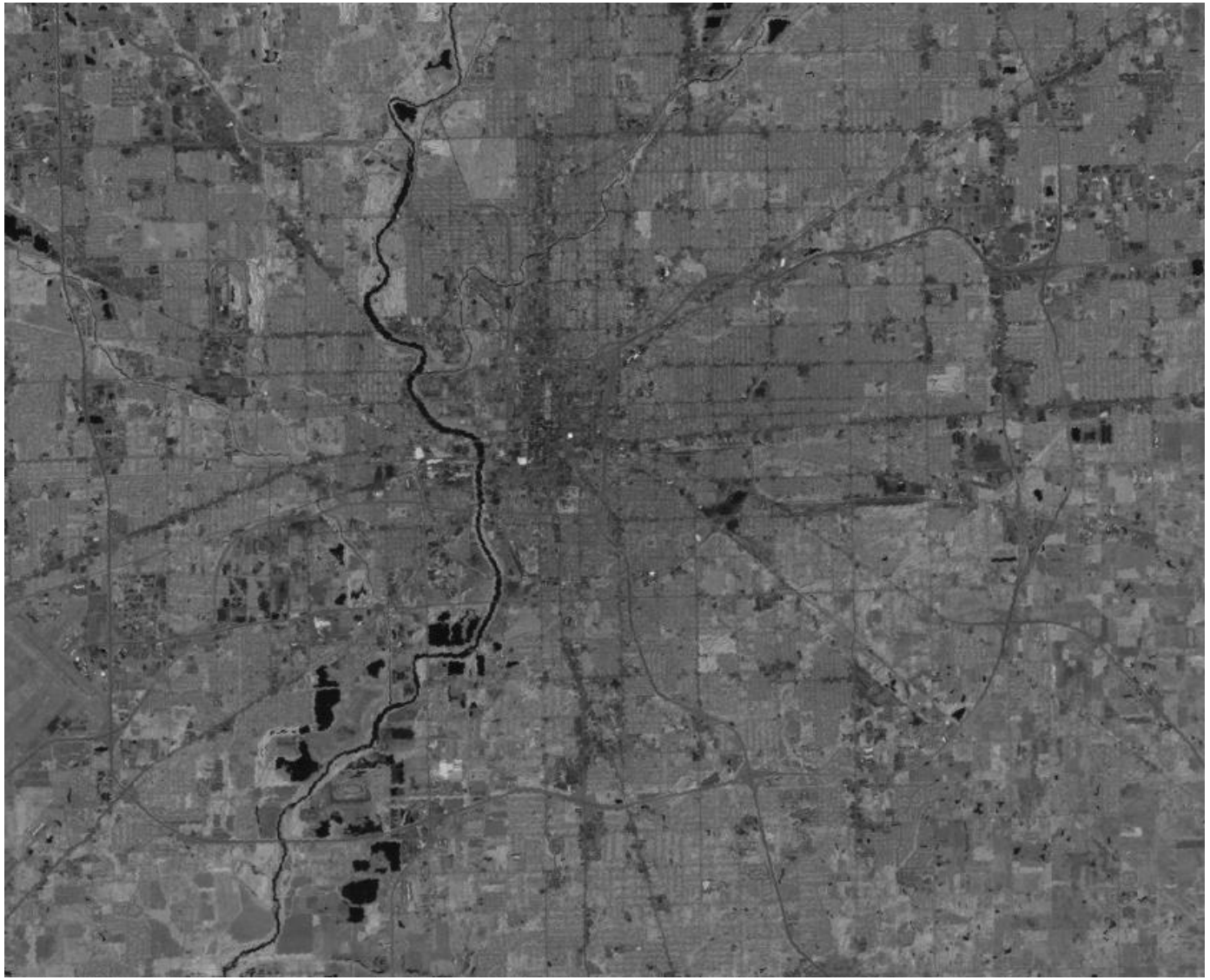
- In hyperspectral images the **number of bands becomes larger** and you get a whole energy spectrum in each pixel

Band 1

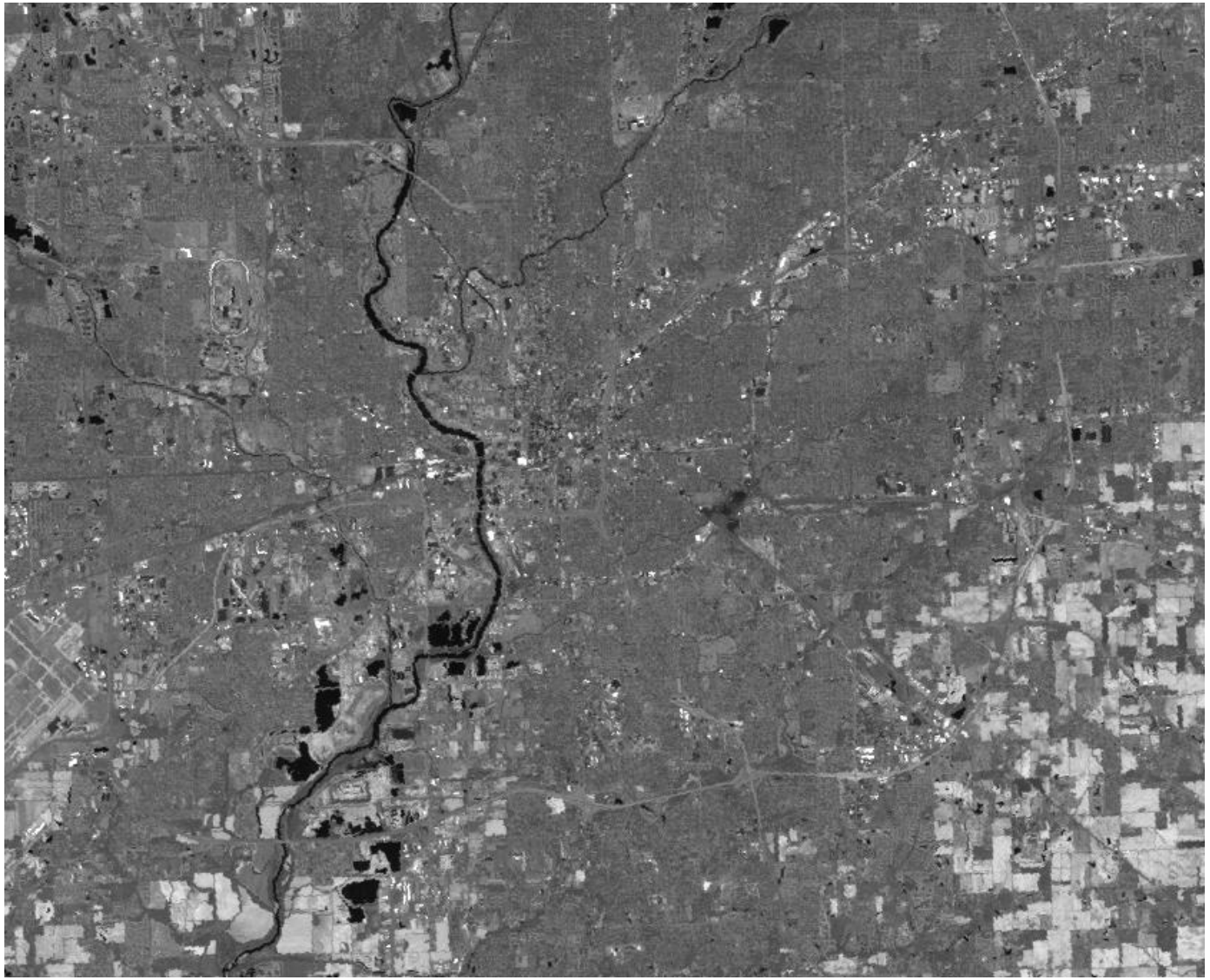
Band 2

Band 3

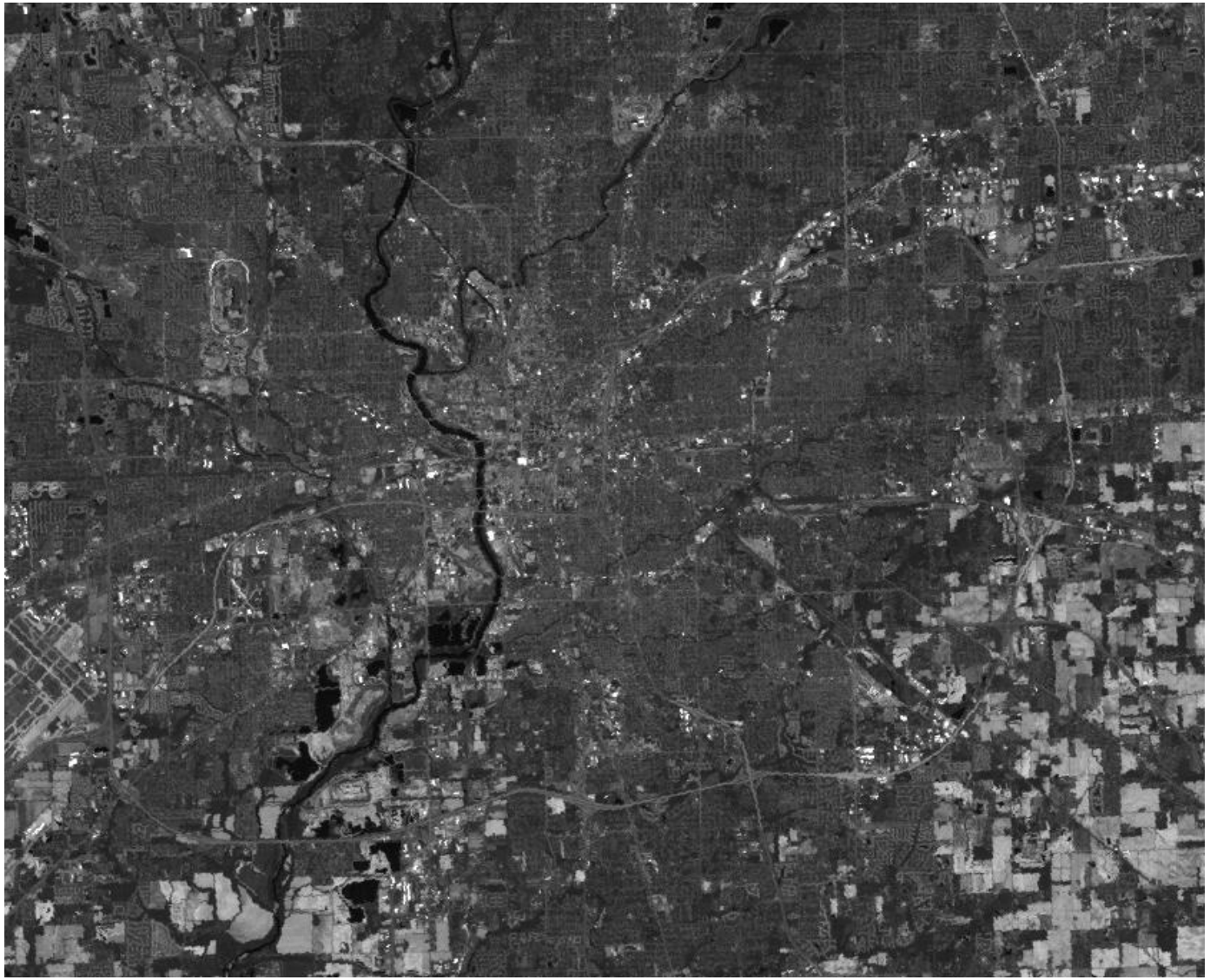
Band 4



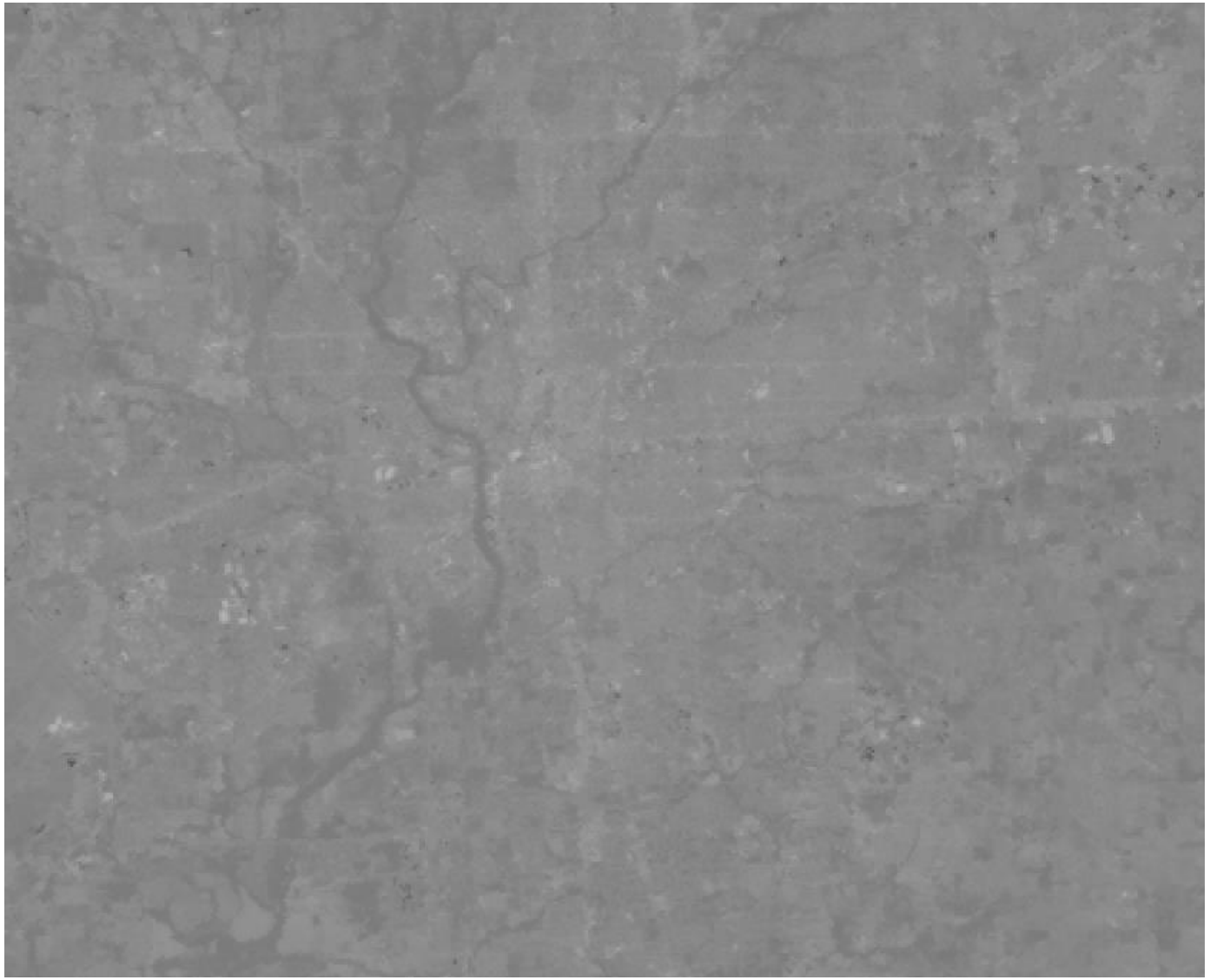
Band 5



Band 6



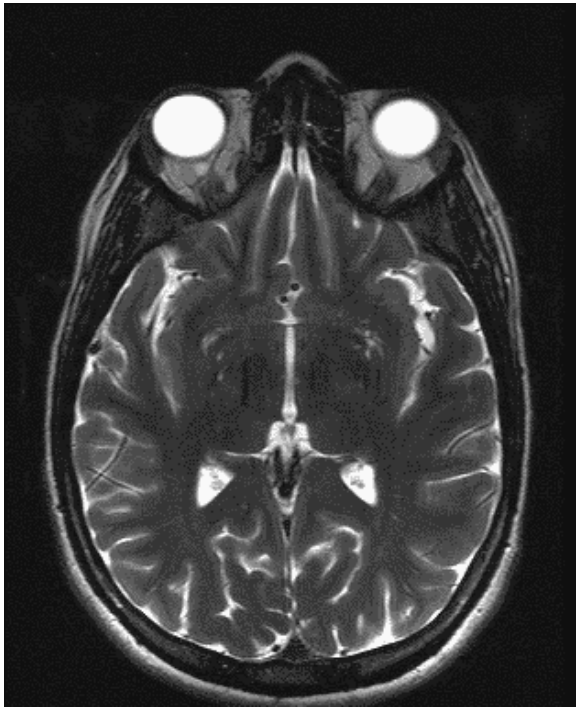
Band 7

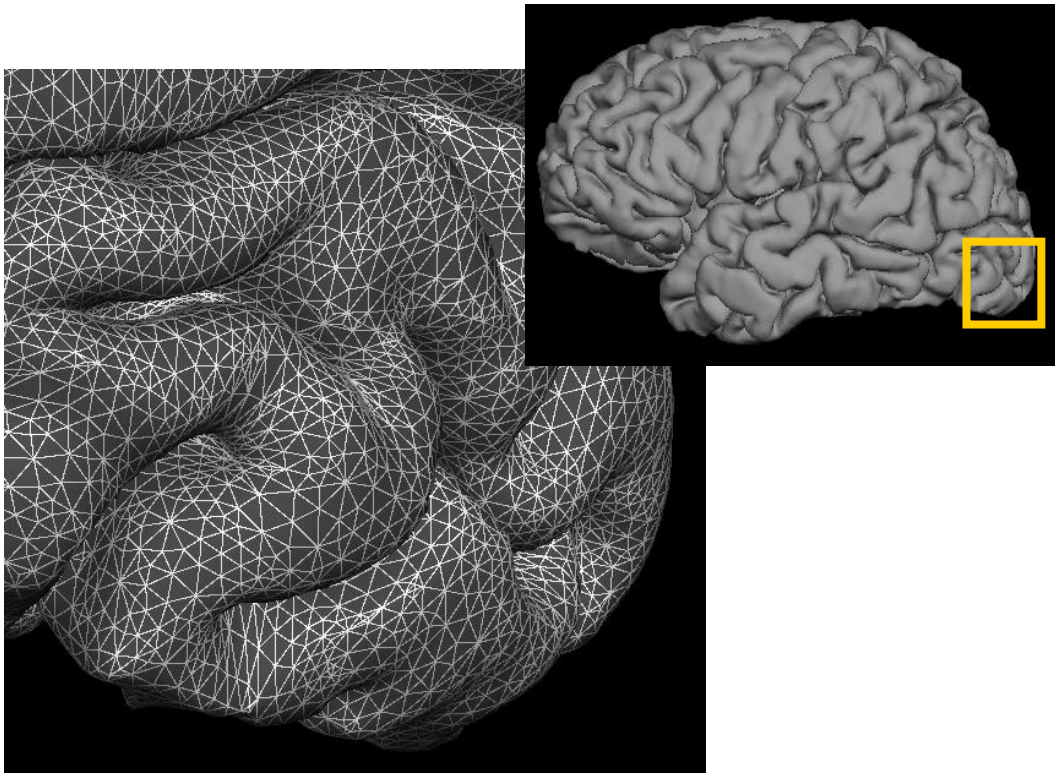


MRI and TAC

(Structural) MRI provide a 3D stack of grayscale images that are analyzing the body at different depth levels

These can be used to perform 3D reconstruction of bones, organs and membranes, e.g. cortical sufaces



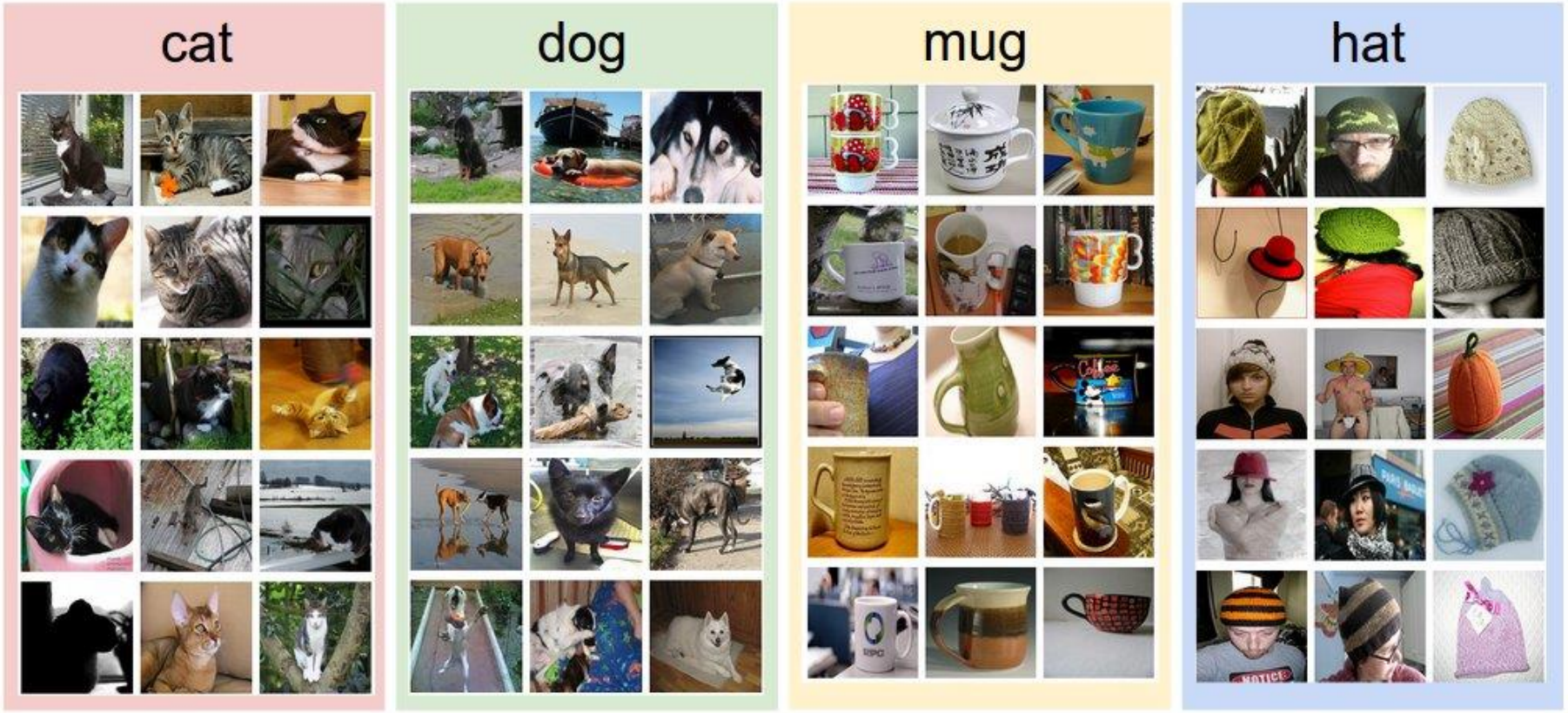


The Image Classification Problem

Problem Statement

The Problem

Image Classification: the task of assigning an input image one label from a fixed set of categories.



Problem Statement

Let us denote:

- $\boldsymbol{x} \in \mathbb{R}^d$ be the image to be classified (d = #of pixels)
- $y \in \Lambda = \{1, \dots, L\}$ the label associated to I (L is the number of classes)

The problem:

Given a training set $TR = \{(\boldsymbol{x}_i, y_i), i = 1, \dots, N\}$ of labelled images, learn/define $\mathcal{K} : \mathbb{R}^d \to \Lambda$, a **classifier** such that

$$\hat{y}_j = \mathcal{K}(\boldsymbol{x}_j)$$

it's an estimate of $y_j, \forall j$.

Here $\hat{y}_j$ is the class estimated by $\mathcal{K}$ and the test samples $\{(x_j, y_j), j = 1, \dots\}$ are not in the training set.

Image Classification challenges

Here are the major challenges to be considered in image classification:

- Labels might not uniquely identify the image (images typically contain multiple objects)
- There are many **transformation** that change the image dramatically, while not its label
- There is a lot of **inter-class variations**

Image Classification challenges

Viewpoint variation

Scale variation

Deformation

Occlusion

Illumination conditions

Background clutter

Intra-class variation

Straightforward Approaches

Train a K-NN on images

A Straightforward Approach

Treat an image as a vector and simply use a classifier from the machine learning literature.

There are two main issues:

- Images are very high-dimensional data
- Perceptual similarity in images is not related to pixel-similarity

Nearest Neighborhood Classifiers for Images

Nearest neighborhood

Assign to a each test image, the label of the closest image in the training set

$$\hat{y}_j = y_{j^*}, \qquad \text{being} \quad j^* = \underset{i=1\ldots N}{\operatorname{argmin}}\, d(\boldsymbol{x}_j, \boldsymbol{x}_i)$$

Distance between images is typically measured as

$$d(\boldsymbol{x}_j, \boldsymbol{x}_i) = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2 = \sqrt{\sum_k \left([\boldsymbol{x}_j]_k - [\boldsymbol{x}_i]_k\right)^2}$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

Nearest Neighborhood Classifiers for Images

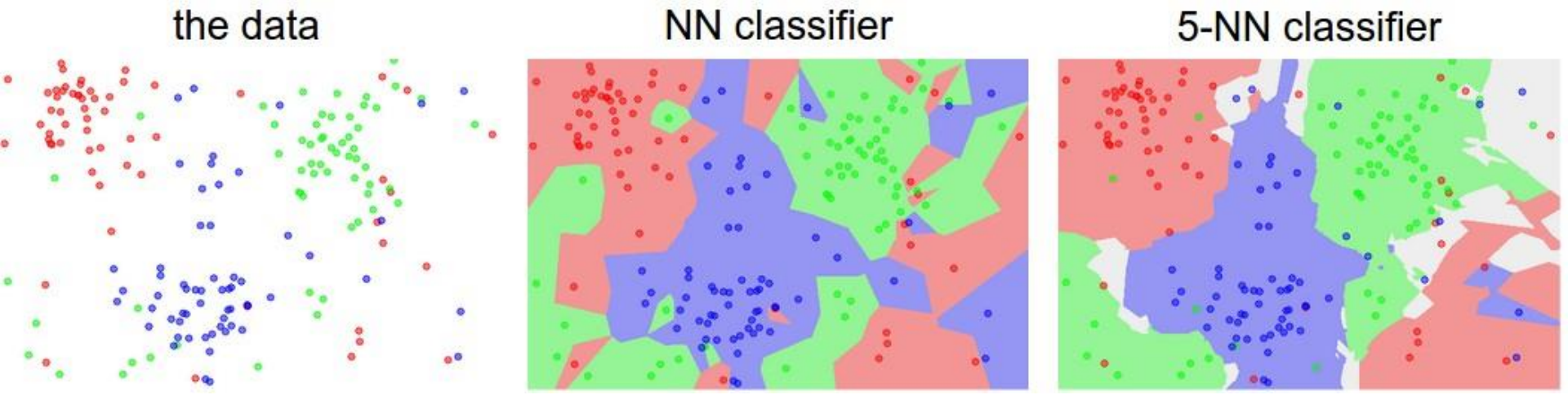
K-Nearest neighborhood (K-NN)

Assign to a each test image, the most frequent label among the K −closest images in the training set

$$\hat{y}_j = y_{j^*}, \qquad \text{being} \quad j^* \text{ the mode of } \mathcal{U}_K(\boldsymbol{x}_j)$$

where $\mathcal{U}_K(\boldsymbol{x}_j)$ contains the K training images that are closer to $\boldsymbol{x}_j$

The **issue** is how to choose the parameter K (and the distance).



Nearest Neighborhood Classifiers for Images

K-Nearest neighborhood (K-NN)

Pros:

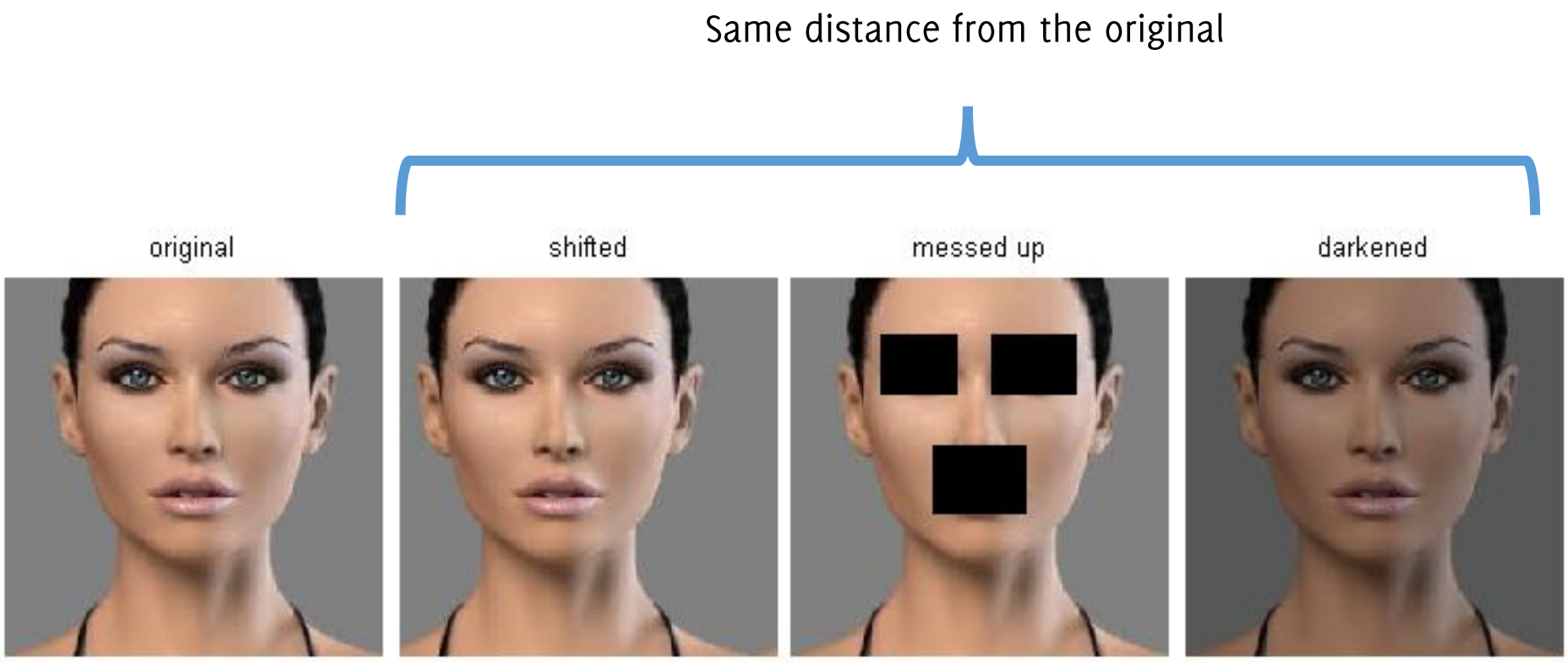
- Easy to understand and implement
- It takes no training time

Cons:

- Computationally demanding at test time (in particular when TR is large and d is also large)
- Large training sets have to be stored in memory
- Rarely practical on images: distances on high-dimensional objects are difficult to interpret

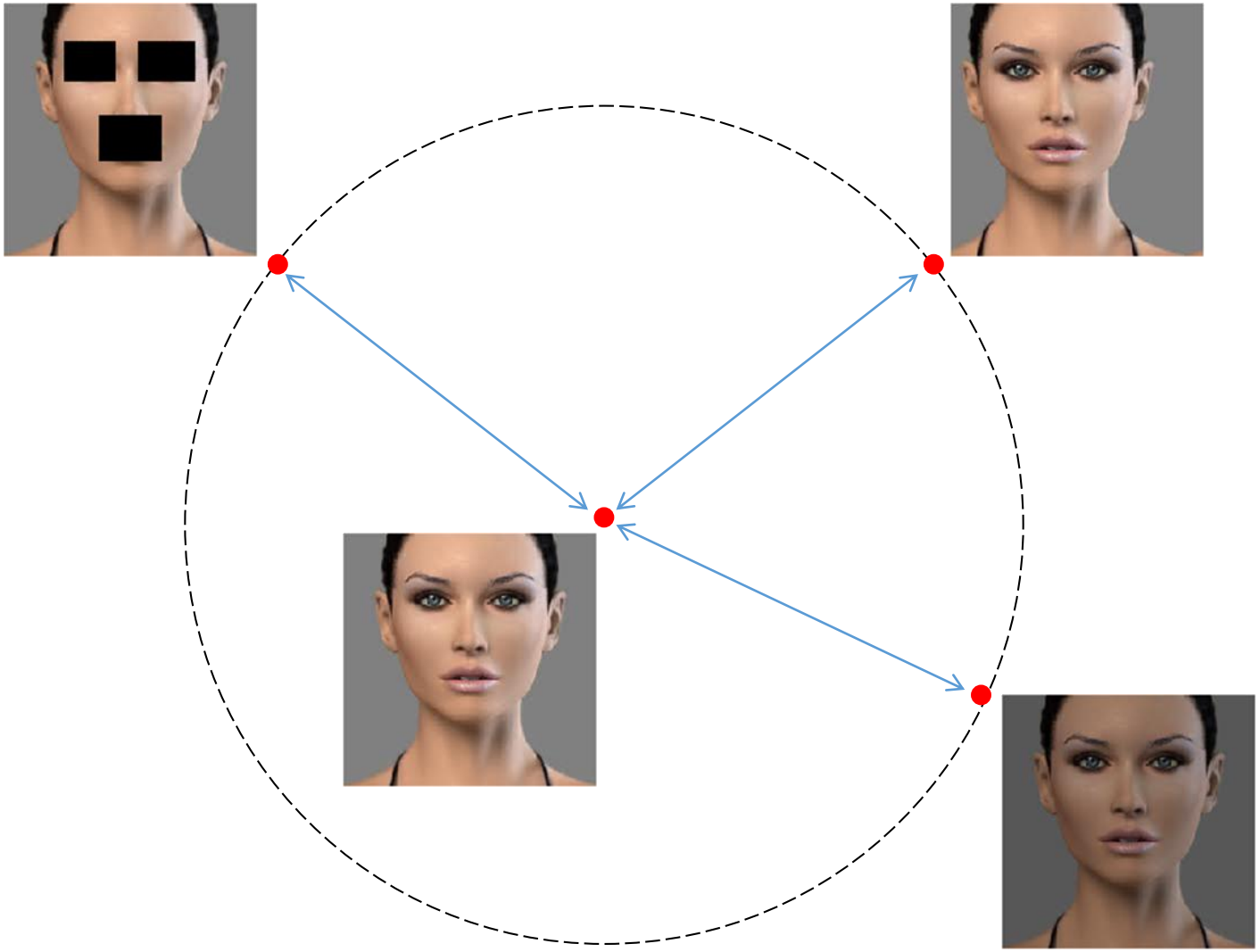
Nearest Neighborhood Classifiers for Images

These three images have the same ℓ^2 distance from the original one but perceptually they are very different



Nearest Neighbourhood Classifiers for Images

Think of these images as points in $\mathbb{R}^d$



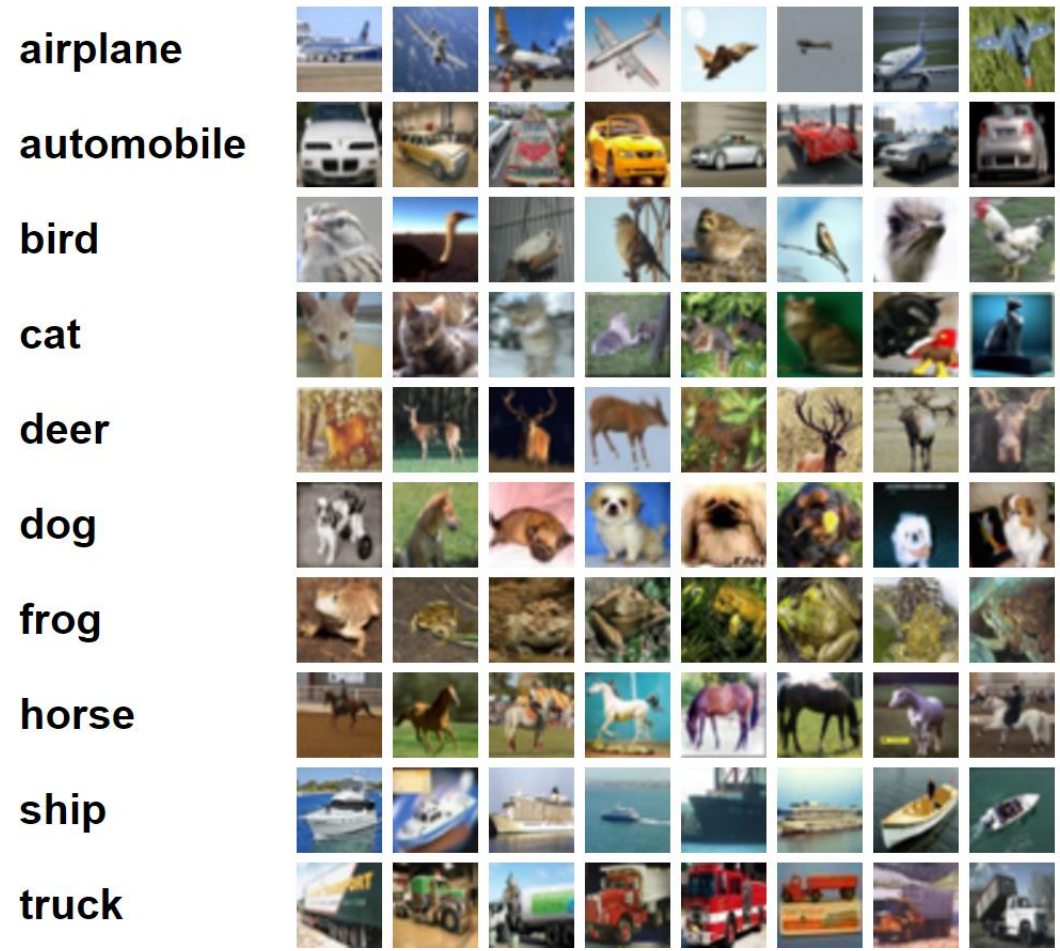
CIFAR-10 dataset

The CIFAR-10 dataset contains 60000 images:

- Each image is 32x32 RGB
- Images are divided in10 classes
- 6000 images per class

Extremely small images, but very high-dimensional data:

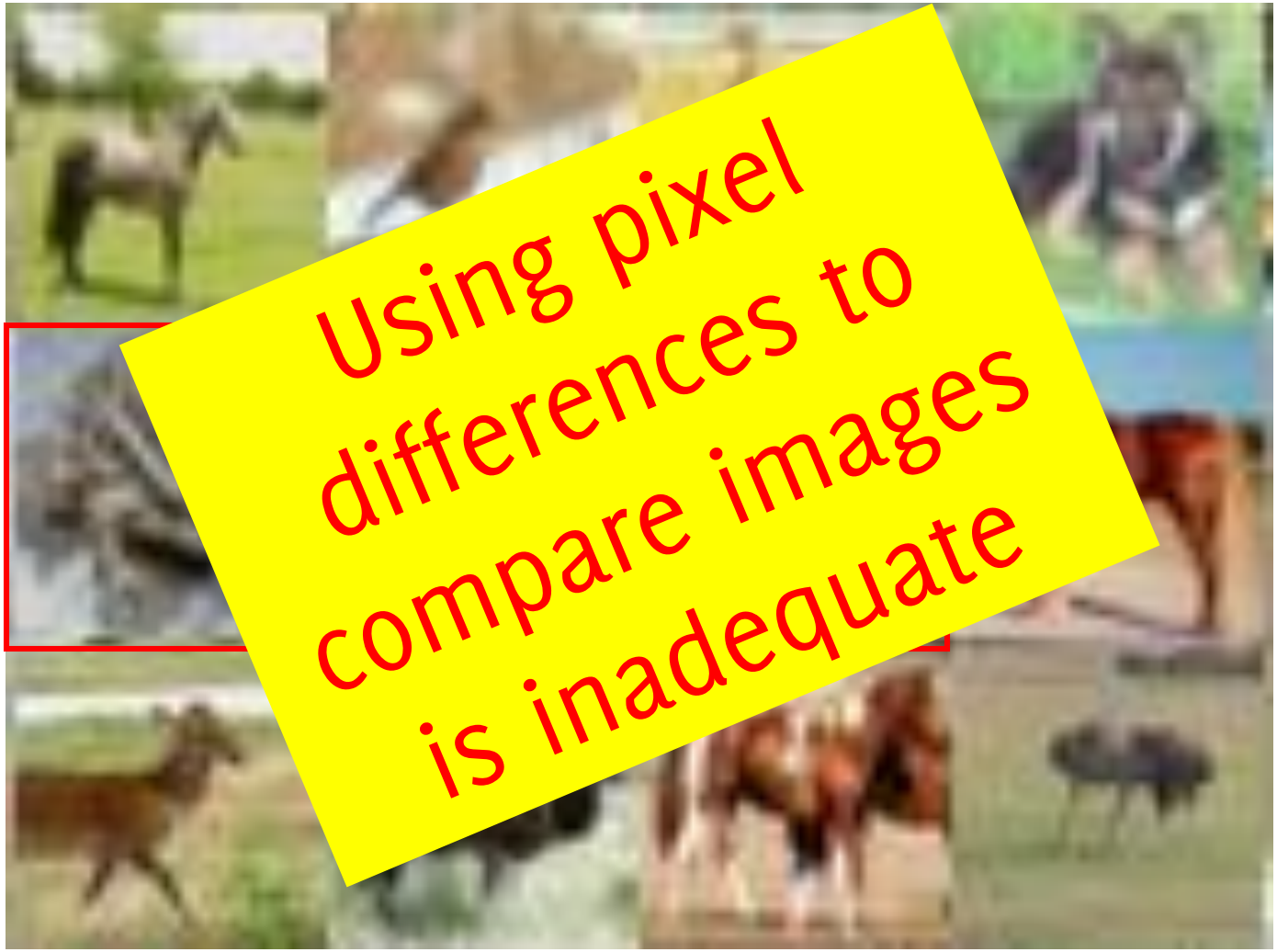
$$d = 32 \times 32 \times 3 = 3072$$



Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

On CIFAR10 we see exactly this problem

On CIFAR10 we see exactly this problem



Linear Classifier

Linear Classifiers for images

It is very important as it represents the basic building block for deep architectures

Linear Classifiers for Images

A classifier can be seen as a function that maps an image $\boldsymbol{x}$ to a **confidence scores for each** of the L **class**:

$$\mathcal{K}\colon \mathbb{R}^d \to \mathbb{R}^L$$

where$\mathcal{K}(\boldsymbol{x})$ is a L −dimensional vector and the i −th component

$$s_i = [\mathcal{K}(\boldsymbol{x})]_i$$

contains a **score of how likely $\boldsymbol{x}$ belongs to class $\boldsymbol{i}$.**

Intuitively, a good classifier associates to the correct class a score that is larger scores associated to incorrect classes.

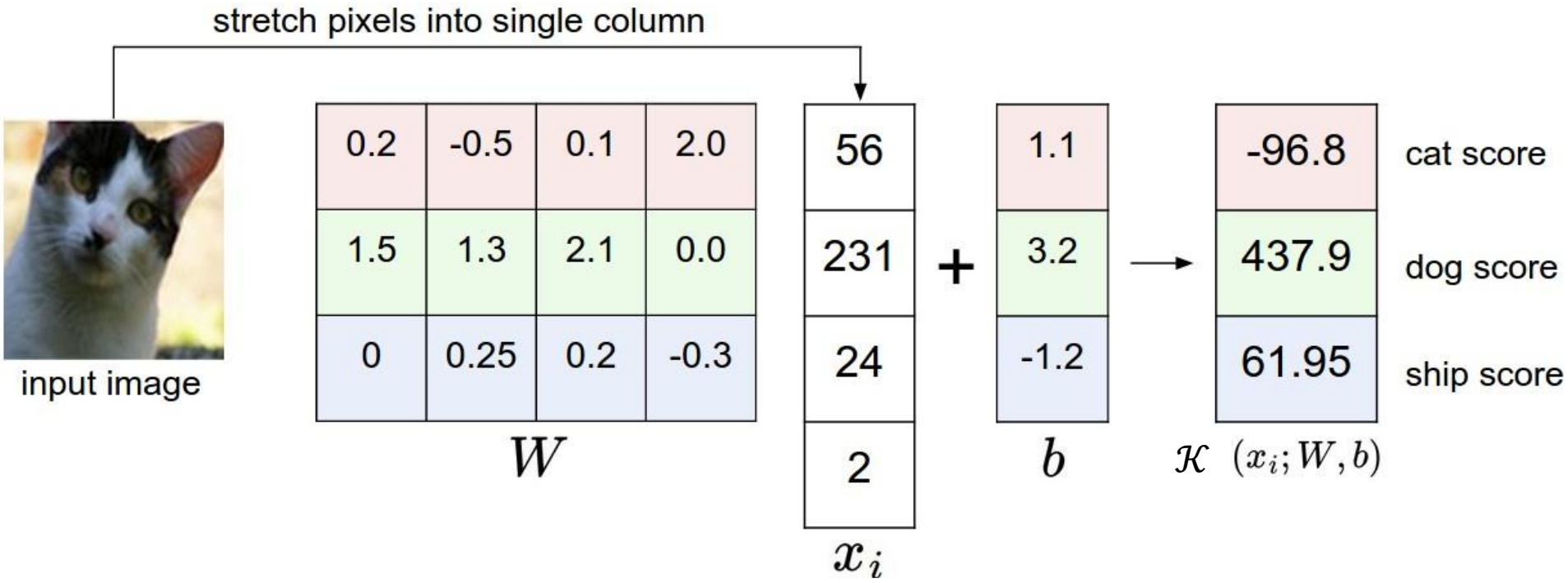
Linear Classifiers for Images

In linear classification $\mathcal{K}$ **is a linear function:**

$$\mathcal{K}(\boldsymbol{x}) = W\boldsymbol{x} + b$$

where $W \in \mathbb{R}^{L \times d}$, $b \in \mathbb{R}^{L}$ are the parameters of the classifier $\mathcal{K}$.

W are referred to as the weights, b the bias.



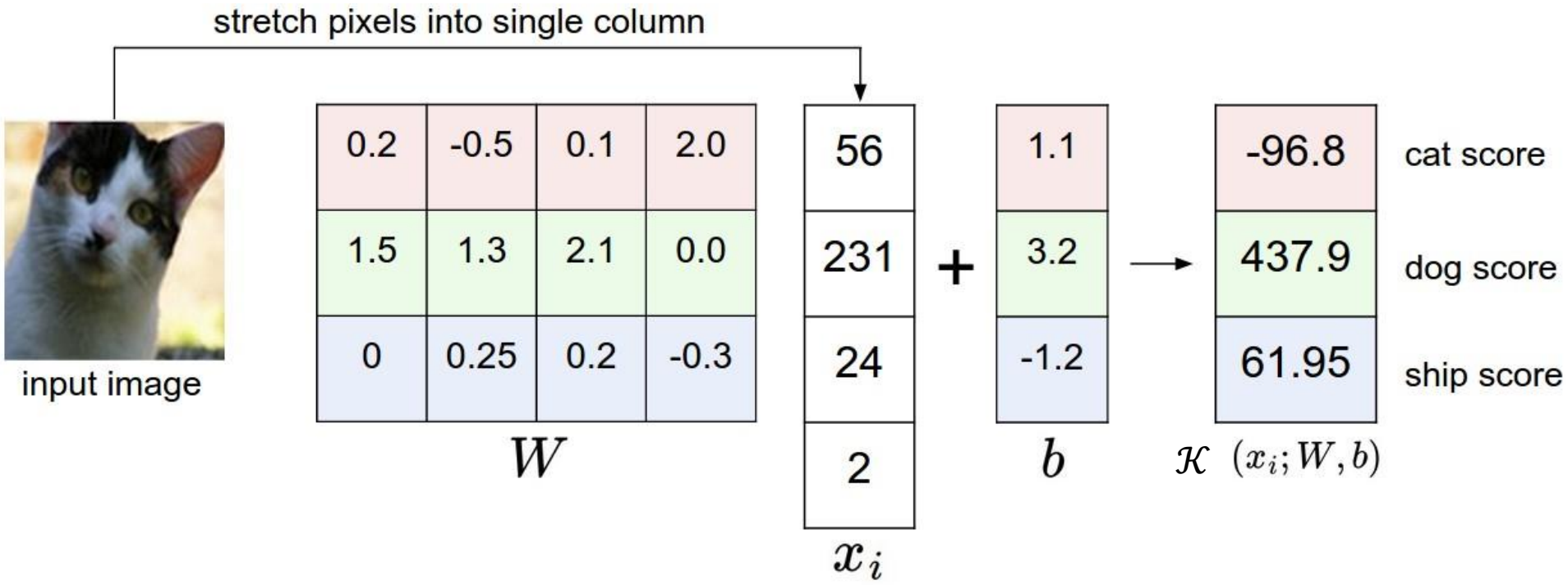
Linear Classifiers for Images

The classifier assign to an input image the class corresponding to the largest score

$$\hat{y}_j = \underset{i=1,..,L}{\operatorname{argmax}} \left[s_j\right]_i$$

being $\left[\boldsymbol{s}_j\right]_i$ the i −th component of the vector

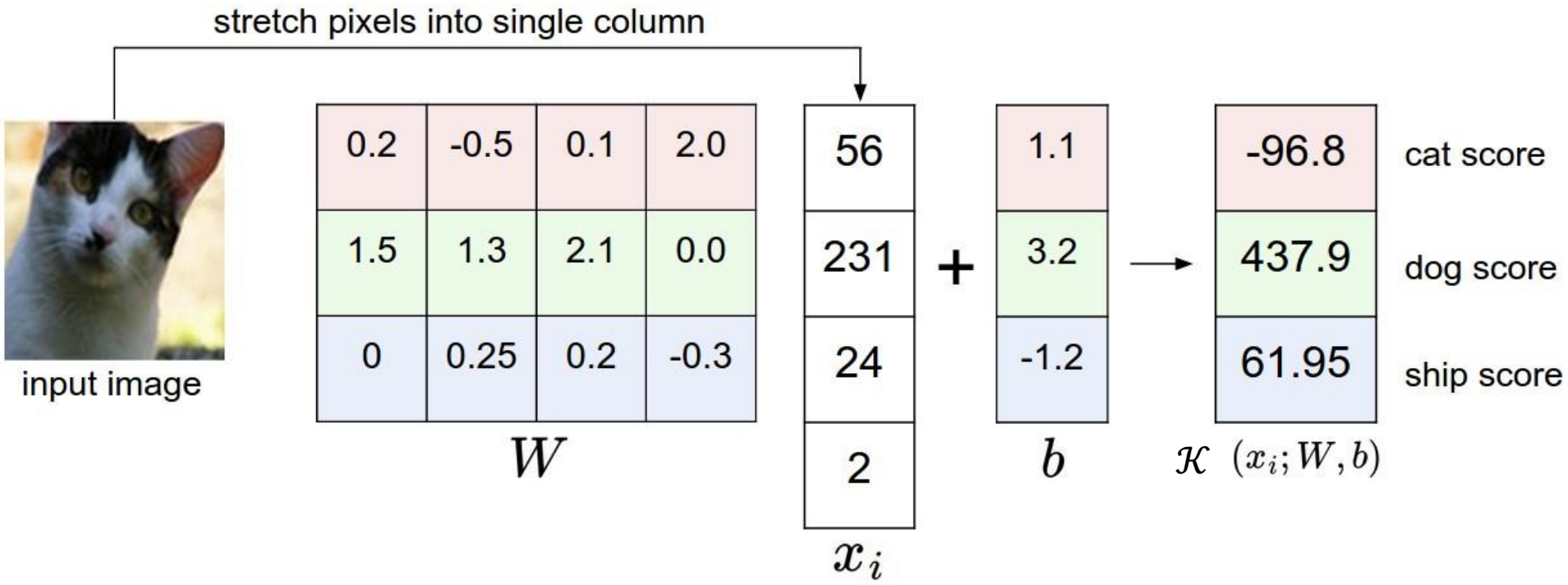
$$\mathcal{K}(\boldsymbol{x}_j) = W\boldsymbol{x}_j + \boldsymbol{b}$$



Linear Classifiers for Images

The score of a class is the weighted sum of all the image pixels. Weights are actually the classifier parameters.

Weights indicate which are the most important pixels / colors



Training a Classifier

Given a training set TR and a loss function, define the parameters that minimize the loss function over the whole TR

In case of linear classifier

$$[W, b] = \underset{W \in \mathbb{R}^{L \times d},\ b \in \mathbb{R}^{L}}{\operatorname{argmin}} \sum_{(\boldsymbol{x}_i, y_i) \in TR} \mathcal{L}(\boldsymbol{x}, y_i)$$

This problem is solved by specific algorithms (see DL class)

The loss function has to be typically regularized to achieve a unique solution satisfying some desired property

$$[W, b] = \underset{W \in \mathbb{R}^{L \times d},\ b \in \mathbb{R}^{L}}{\operatorname{argmin}} \sum_{(\boldsymbol{x}_i, y_i) \in TR} \mathcal{L}(\boldsymbol{x}, y_i) + \lambda\, \mathcal{R}(W, b)$$

being $\lambda > 0$ a parameter balancing the two terms

Once Trained

The classifier is expected to provide to the correct class a score that is larger than that assigned to the incorrect classes.

The training data is used to learn the parameters **W,b**,

Once the training is completed, it is possible to discard the whole training set and only keep the learned parameters.

Geometric Interpretation of a Linear Classifier

In Matlab/Python notation:

$W(i,:)$ is a d −dimensional vector containing the weights of the score function for the i −th class

Computing the score function for the i −th class corresponds to computing the inner product

$$W(i,:) * \boldsymbol{x}$$

Thus, linear classifiers identify an hyperplane in a d −dimensional

Geometric Interpretation

Interpret each image as a point in $\mathbb{R}^d$.

Each classifier is a weighted sum of pixels, which corresponds to a linear function in $\mathbb{R}^d$

In $\mathbb{R}^2$ these would be
$f([x_1, x_2]) = w_1 x_1 + w_2 x_2 + b$

Then, points $[x_1, x_2]$ yielding
$$f([x_1, x_2]) = 0$$

would be lines.

Thus, in $\mathbb{R}^2$ the region that separates positive from negative scores for each class is a line. This region becomes an hyperplane in $\mathbb{R}^d$.

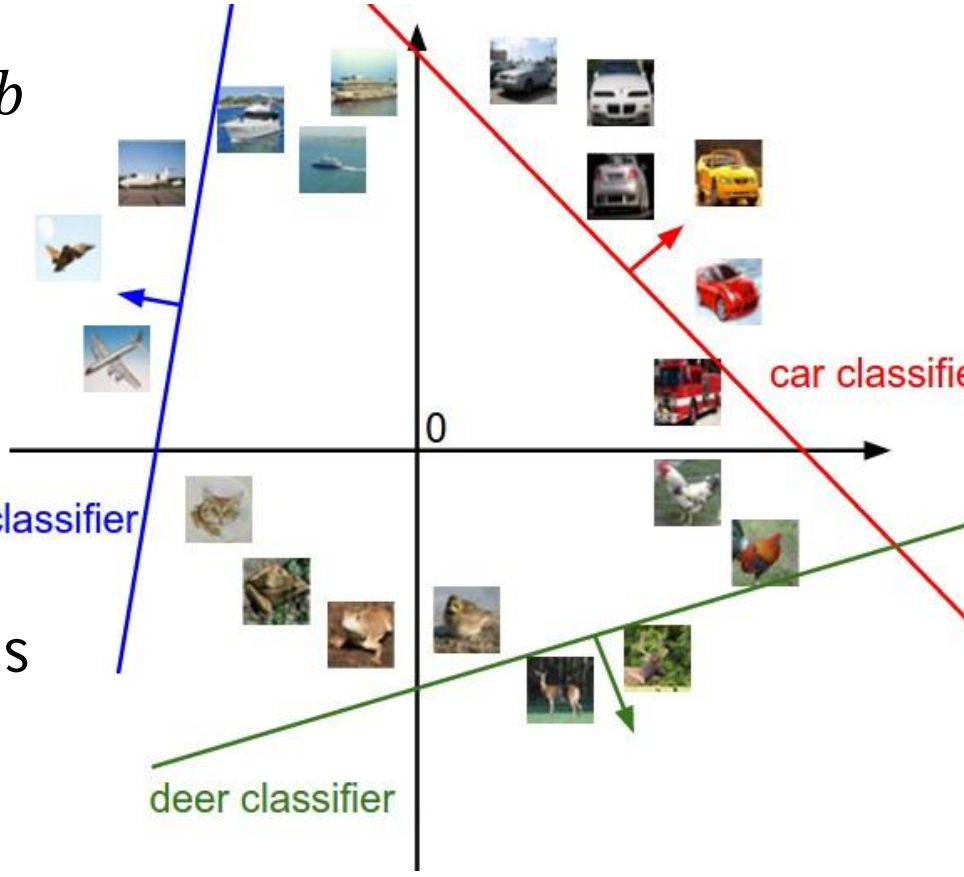


Image Classification By Feature Extraction

The Feature Extraction Perspective

Images can not be directly fed to a classifier

We need some intermediate step to:

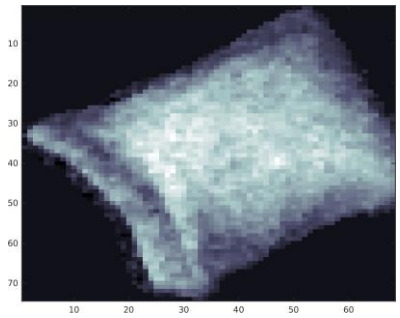
- Extract meaningful information (to our understanding)
- Reduce data-dimension

We need to extract features:

- The better our features, the better the classifier

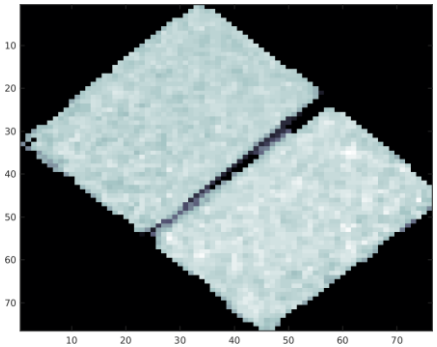
The Feature Extraction Perspective

Input image



$I_1 \in \mathbb{R}^{r_1 \times c_1}$

Input image



$I_1 \in \mathbb{R}^{r_2 \times c_2}$

Feature Extraction Algorithms

$\mathbf{x} \in \mathbb{R}^d$

$\mathbf{x} \in \mathbb{R}^d$

$(d \ll r \times c)$

Classifier

"parcel"

$y \in \Lambda$

"double"

$y \in \Lambda$

The Feature Extraction Perspective

Images can not be directly fed to a classifier

We need some intermediate step to:

- Extract meaningful information (to our understanding)
- Reduce data-dimension

We need to extract features:

- The better our features, the better the classifier

Different Options (to be covered in IC)

- **Hand Crafted Features**
- **Computer Vision Features**
- **Learned Features**