Lecture Session
Week – 2

Computer Vision

Winter Semester 20/21

Goethe University

Acknowledgement: Some images are from various
sources: UCF, Stanford cs231n, etc.

Today’s class

- Image histogram
- Image classification:
    - data-driven approach
    - K-nn

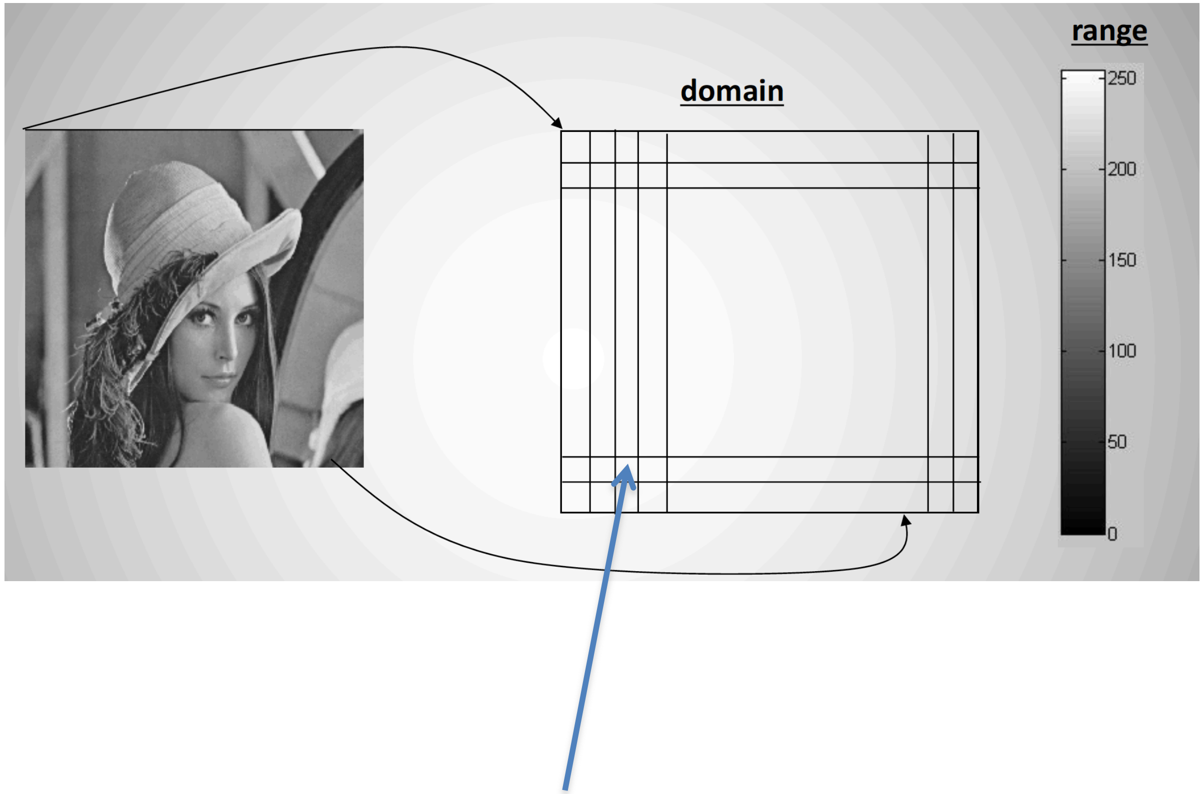
Image is an array of numbers

-Grayscale image

-2D array of numbers (pixels) / matrix

-Number indicates the intensity: [0,255] for 8-bit representation

-Image resolution / number of pixel in an image: 100x100, 1920x1080, etc.



A pixel

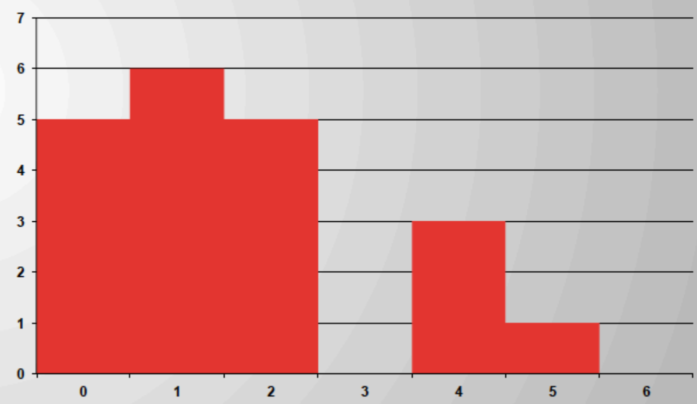
0: black, 255: white

Image Histogram

- Histogram
  - X-axis: bins of possible values
  - Y-axis: frequency of a value (number of samples)
- Normalize Y-axis => probability mass function (the probability of a pixel value in the image)
- Area = total number of pixels

0	1	1	2	4
2	1	0	0	2
5	2	0	0	4
1	1	2	4	1

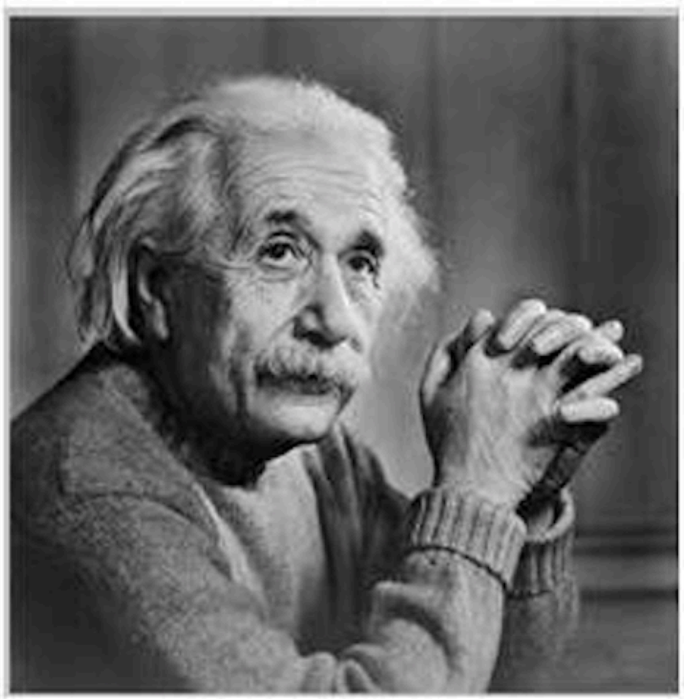
Image (In general, data)

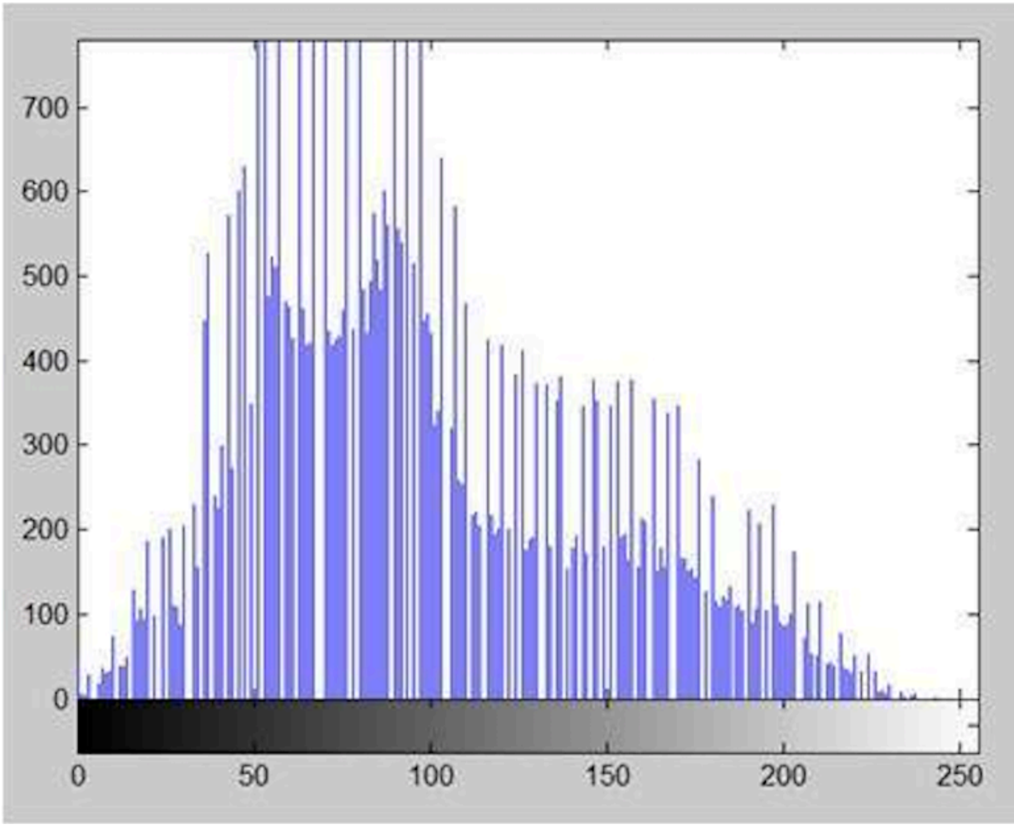


Histogram

Image Histogram

- Image Histogram
- X-axis: pixel values, i.e. 0 to 255
- Y-axis: number of pixels with a certain pixel value





Histogram equalization

- To increase the contrast of an image
  - Over or under-exposed photographs
  - Medical imaging: x-ray images, etc.
- Distribute intensities more evenly over the range: spread out the most frequent intensity values

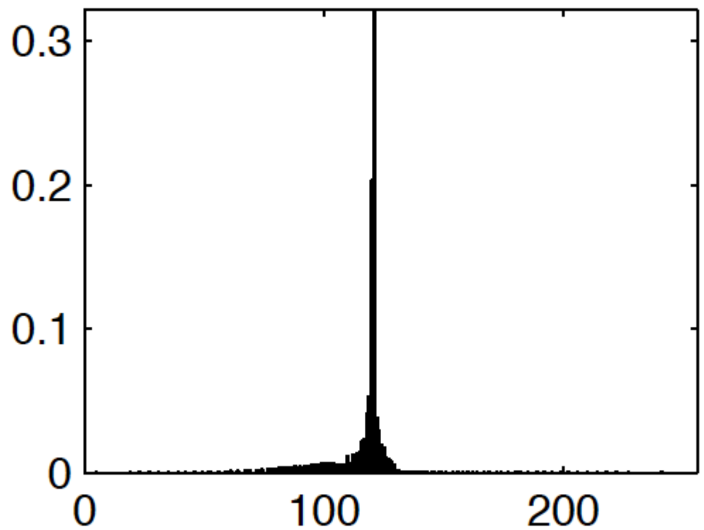
Image with low contrast

QUESTIONS:
What would be the histogram?
Why?

Histogram equalization

- To increase the contrast of an image
  - Over or under-exposed photographs
  - Medical imaging: x-ray images, etc.
- Distribute intensities more evenly over the range: spread out the most frequent intensity values

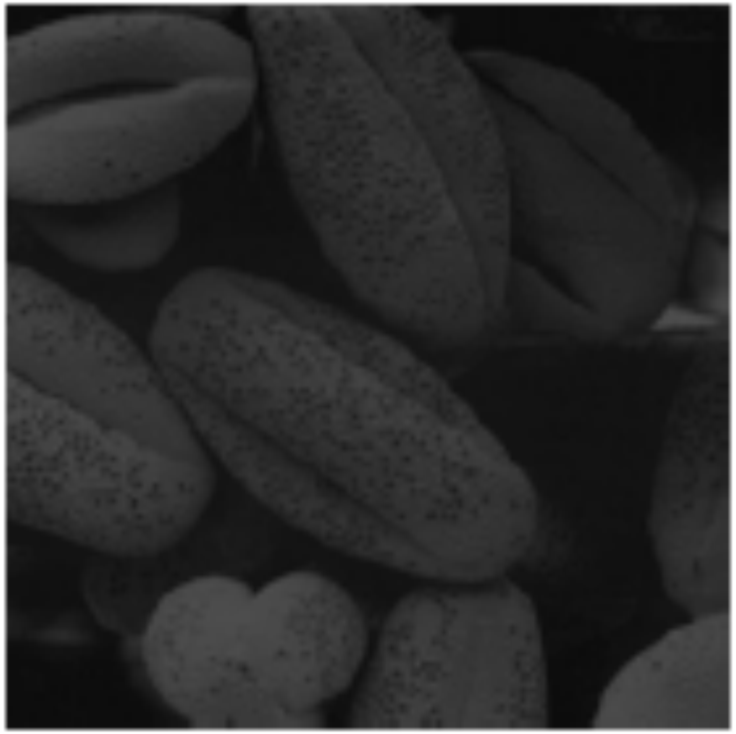
Image with low contrast



histogram

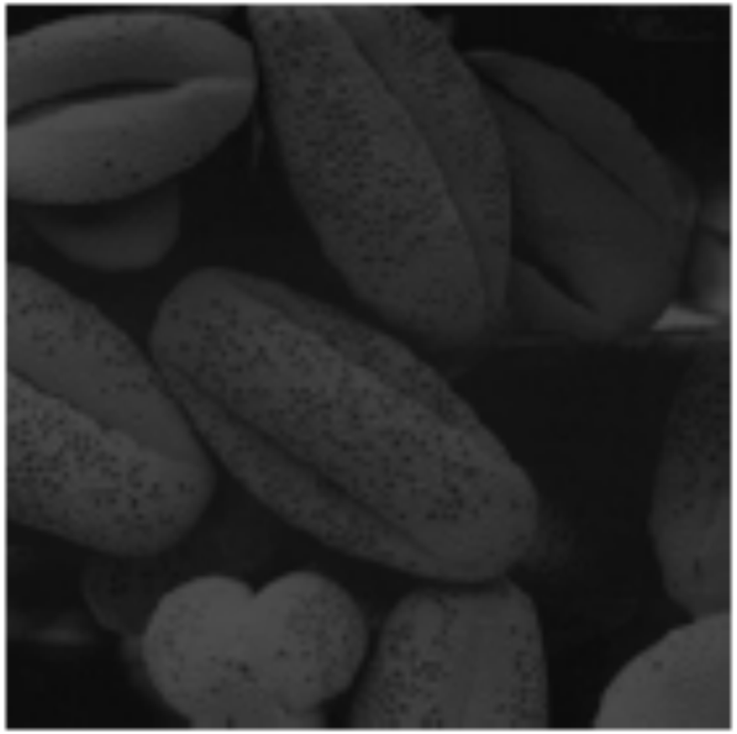
Histogram equalization

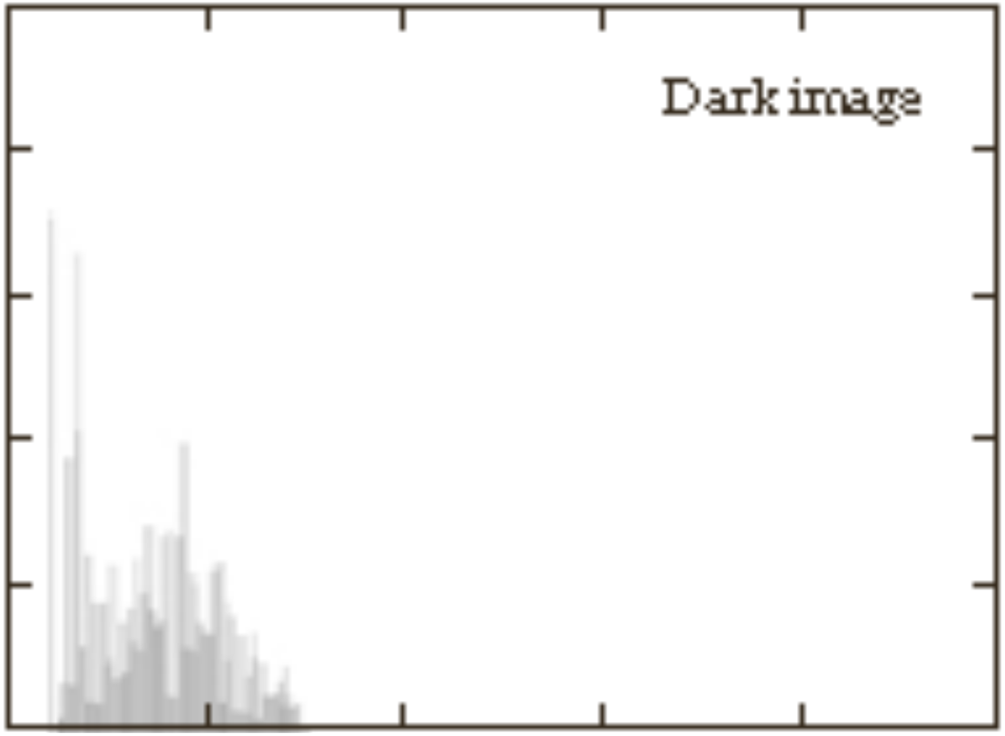
- Histogram of a dark image



Histogram equalization

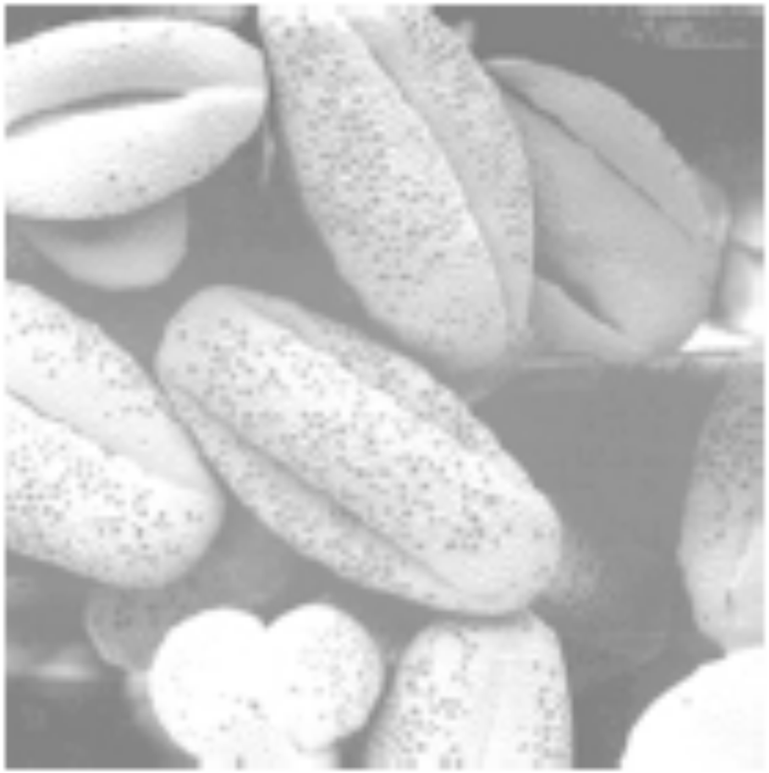
- Histogram of a dark image





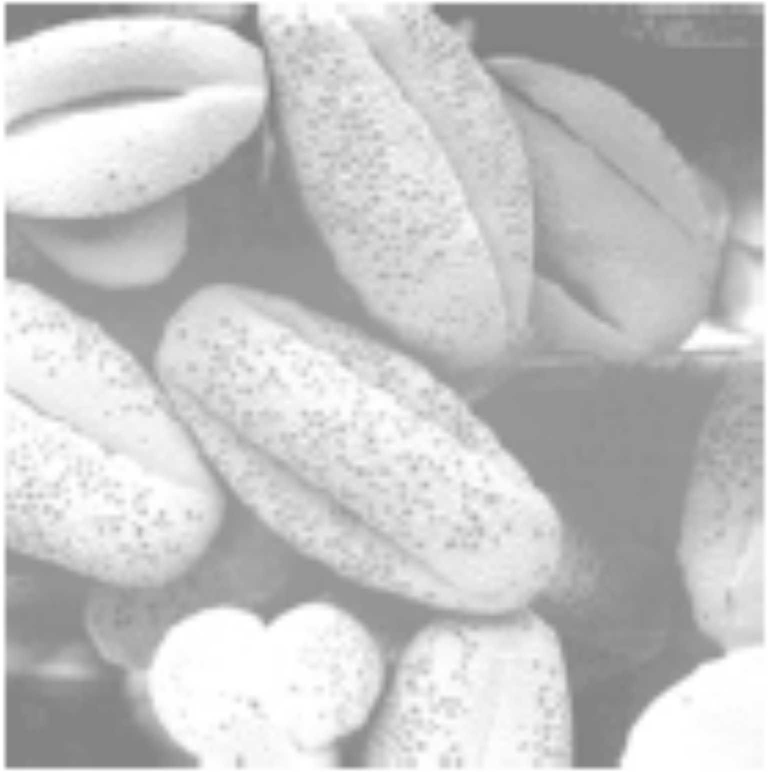
Histogram equalization

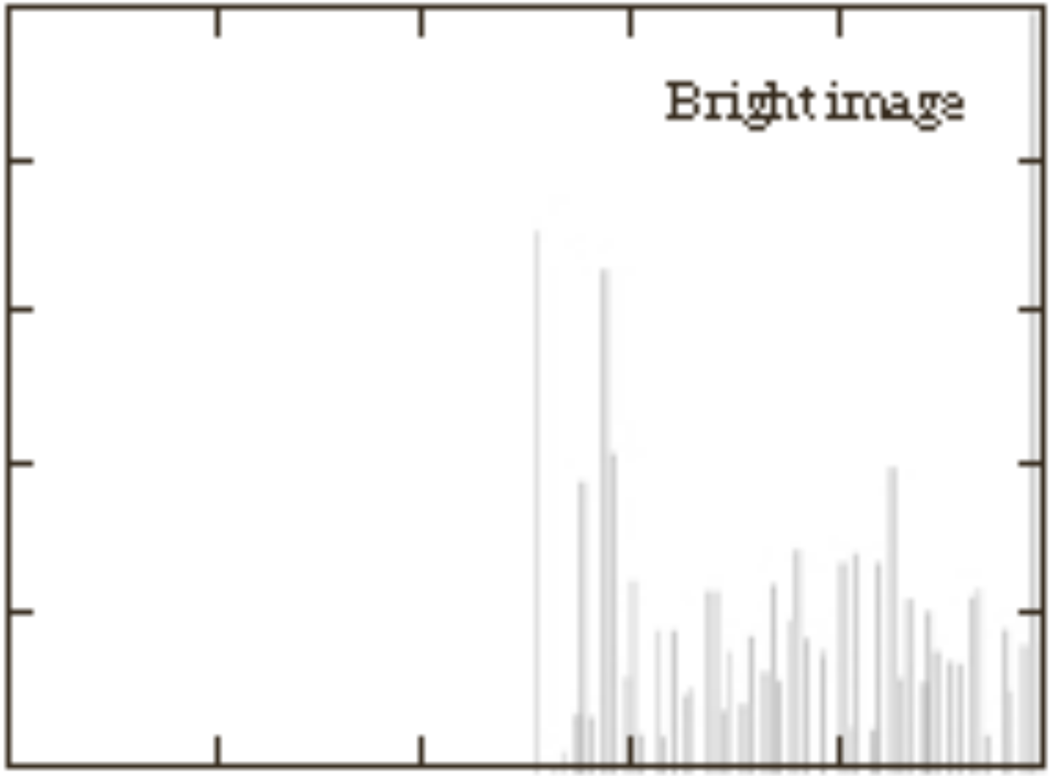
- Histogram of a bright image



Histogram equalization

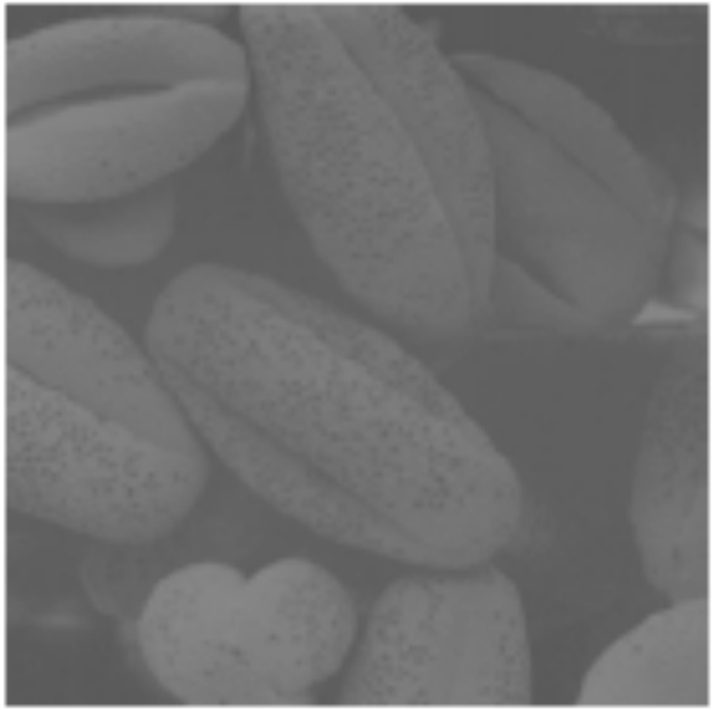
- Histogram of a bright image





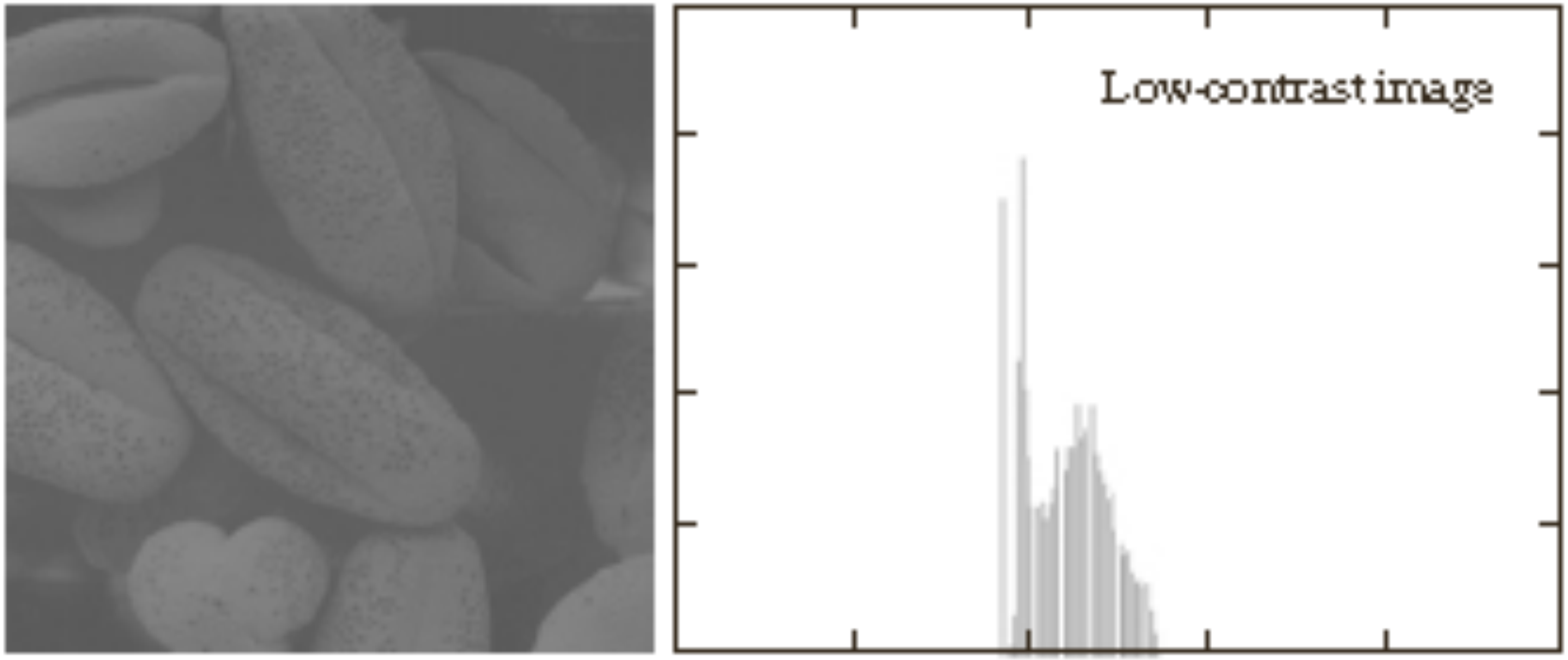
Histogram equalization

- Histogram of a low-contrast image



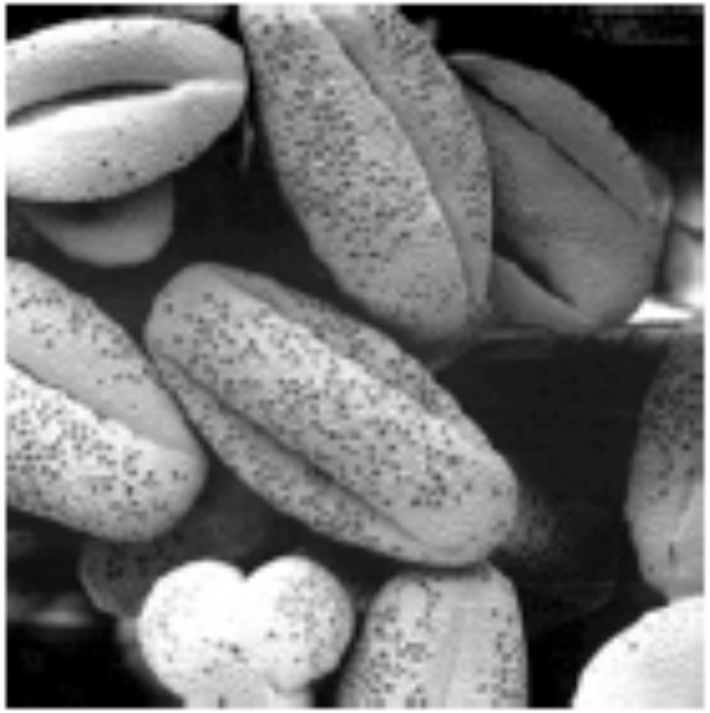
Histogram equalization

- Histogram of a low-contrast image



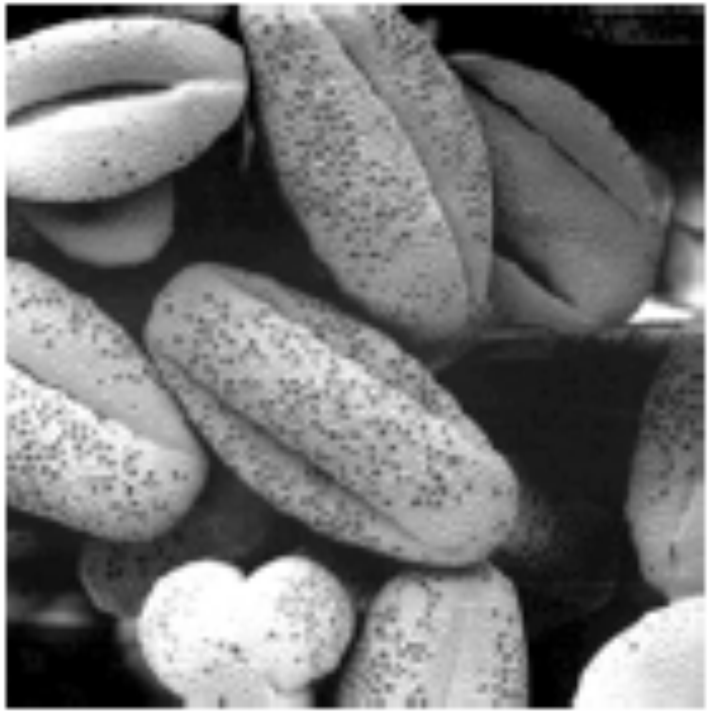
Histogram equalization

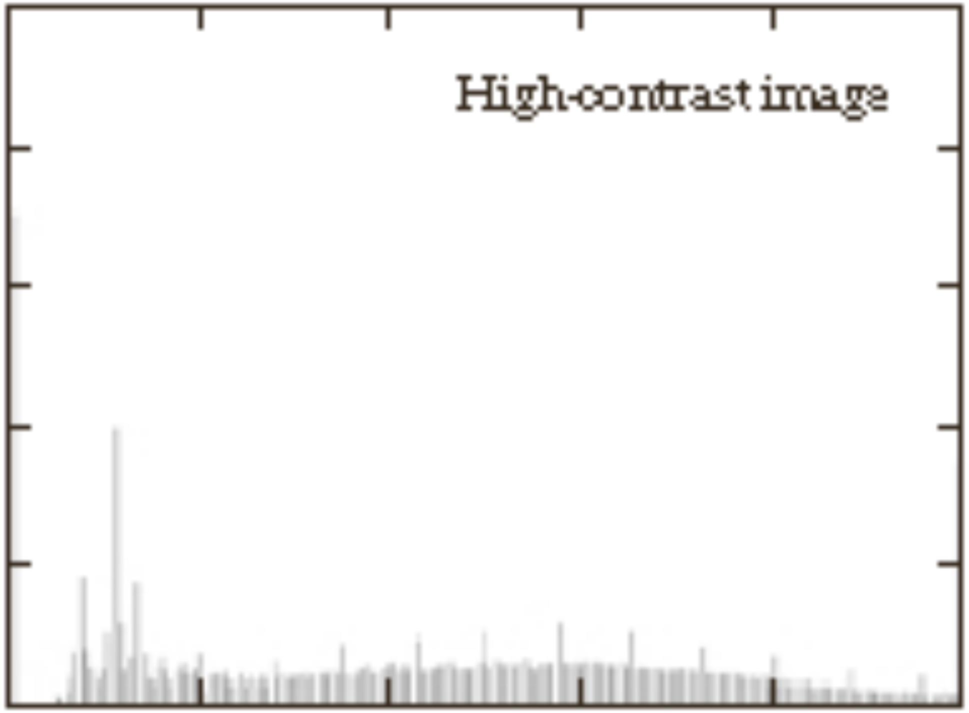
- Histogram of a high-contrast image



Histogram equalization

- Histogram of a high-contrast image





Cumulative distribution / density function (cdf)

- The cdf of a random variable X is given by

$$F_X(x) = P(X \leq x)$$

- If X is a continuous random variable, cdf is given by:

$$F_X(x) = \int_{-\infty}^{x} f_X(w)dw$$

- $f_X(x)$ is the probability density function, pdf

Cumulative distribution / density function (cdf)

- $f_X(x)$ is the probability density function, pdf

$$\int_{-\infty}^{\infty} f(x)\, dx = 1 \qquad\qquad f(x) \geq 0\,, \forall x$$

- A probability density is not the same as a probability

- The probability of a specific value as an outcome of continuous experiment is (generally) zero.

- To get meaningful numbers you must specify a range

$$P(a \leq X \leq b) = \int_a^b f(q)\, dq = F(b) - F(a)$$

Cumulative distribution function

- If X is a discrete r.v., cdf is given by

$$F_X(k) = \sum_{i=-\infty}^{k} p_i$$

(p_i is the probability mass of X at i)

Properties:

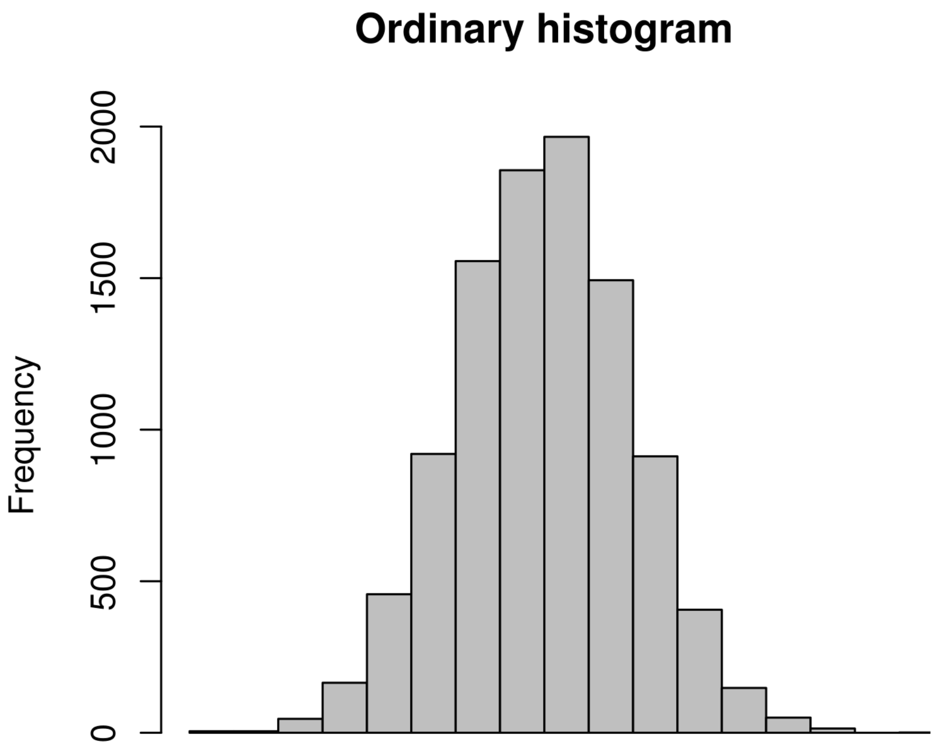
X is the random variable - > what does the R.V, represents here?

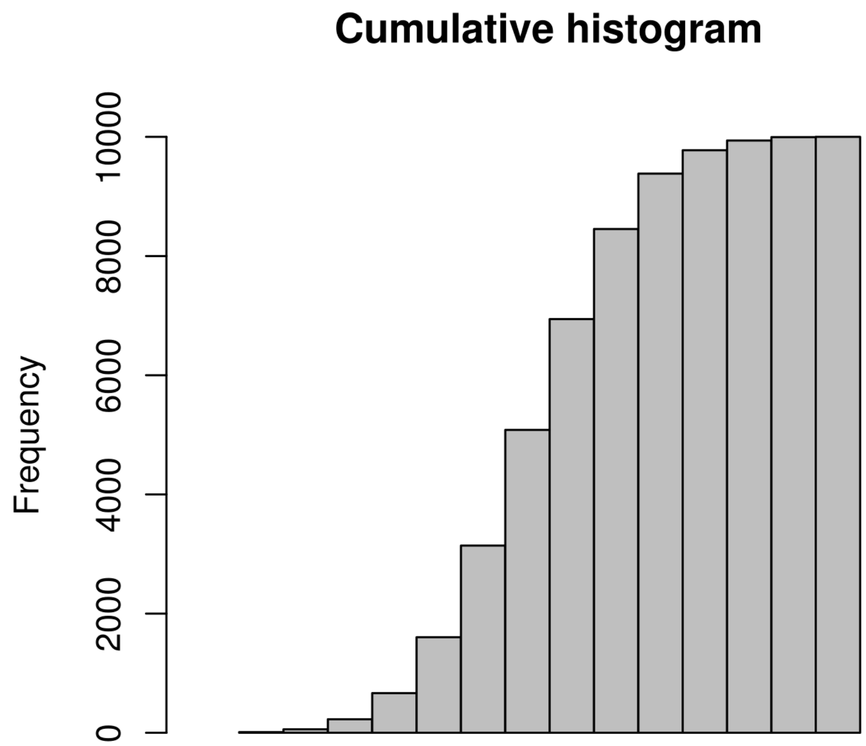
k is a value of the random variable

F(-inf) = 0

F(inf) = 1

Cumulative distribution function



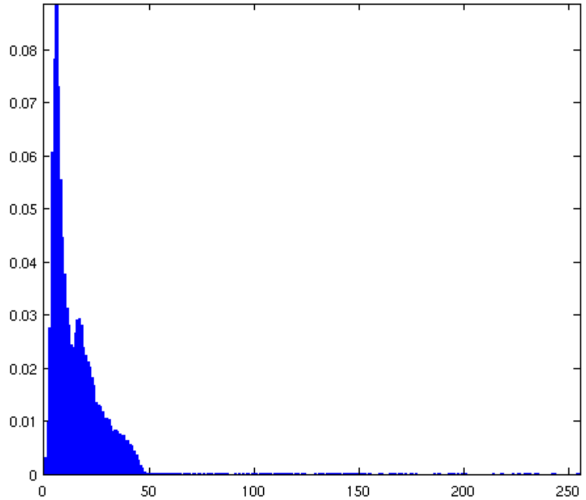


10,000 pixels (cartoon example)

cdf needs to be normalized

Cumulative distribution function

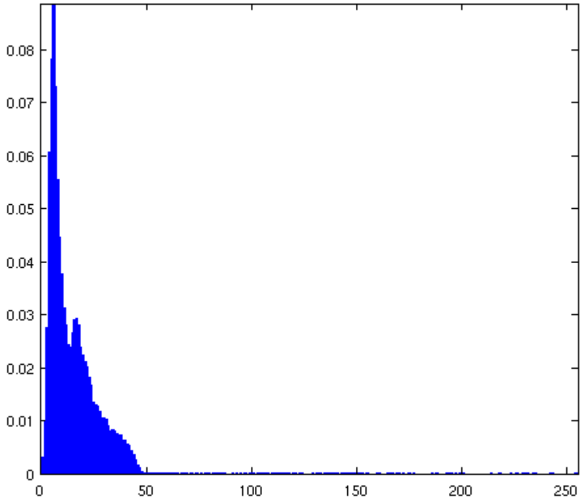
Normalized histogram



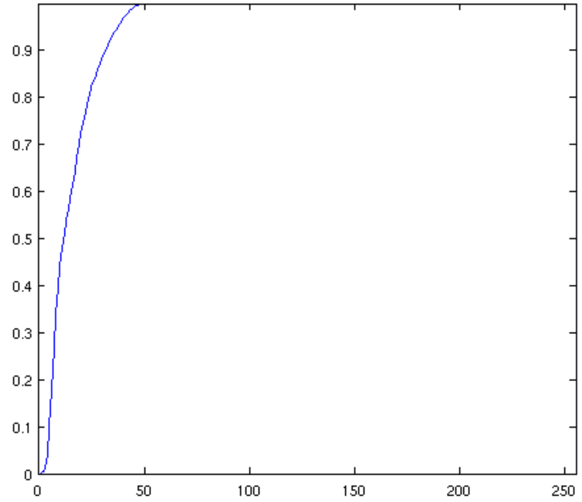
Shape of the cumulative distribution function ?

Cumulative distribution function

Normalized histogram

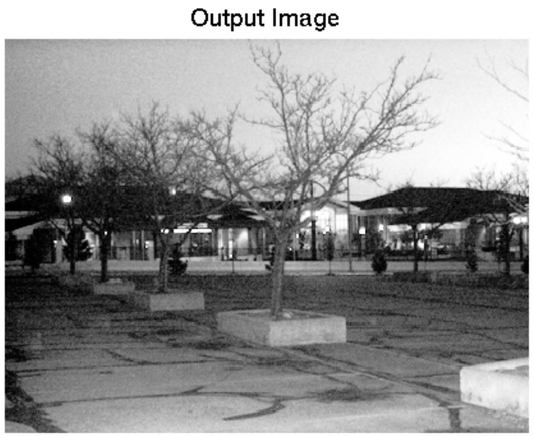


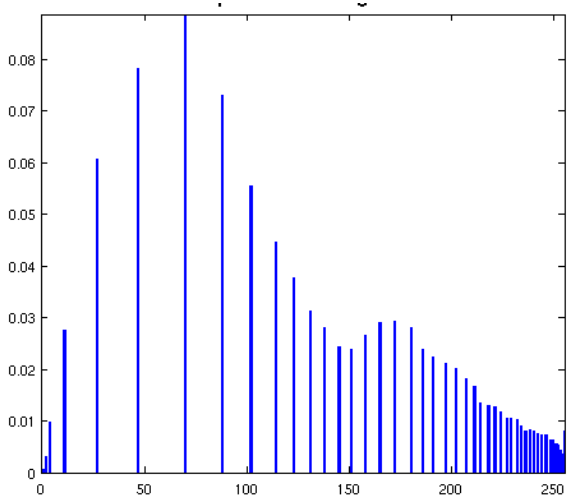
cdf



Cumulative distribution function

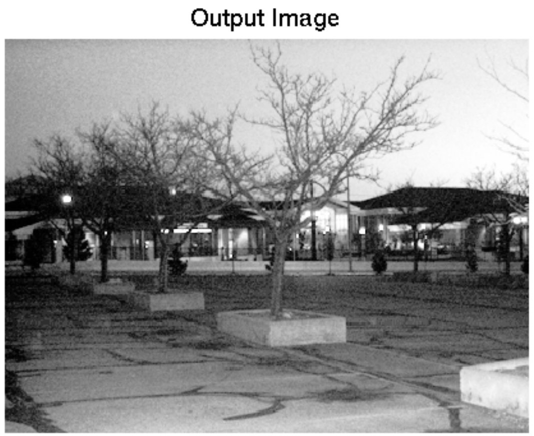
Normalized histogram



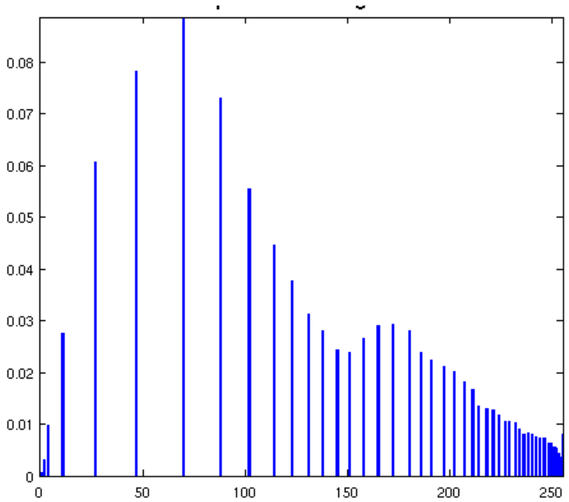


Shape of the cumulative distribution function ?

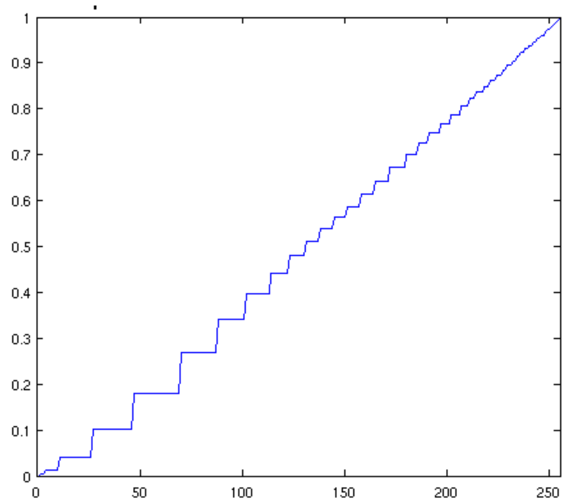
Cumulative distribution function



Normalized histogram

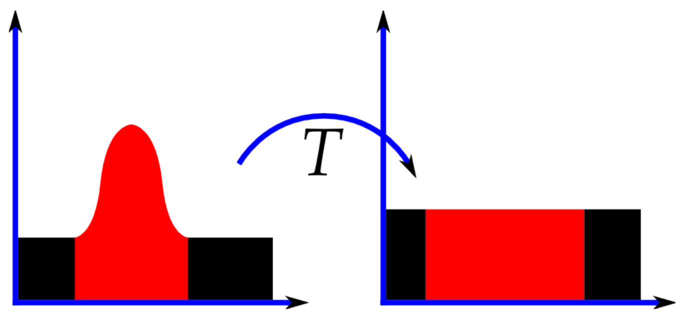


cdf



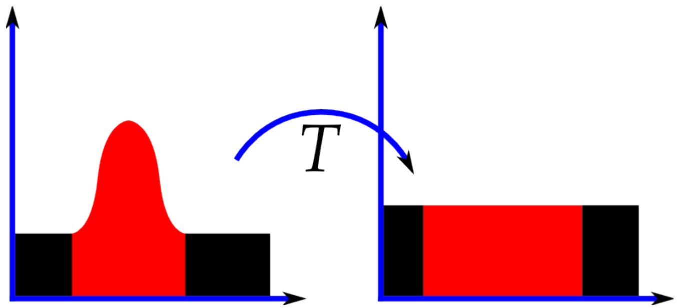
Histogram equalization

- We can transform image values to improve the contrast
- Want histogram of the image to be flat
- This will make full use of the entire display range
- This is called histogram equalization



Histogram equalization

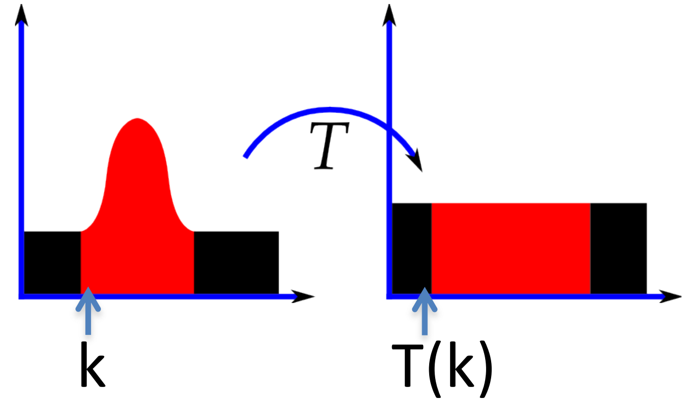
- Apply a transformation T to distribute intensities evenly over the range -> increase contrast
- A mapping of pixel value
- Note area (num. of pixels) in the histogram remains the same after transformation



Histogram equalization

- A mapping of pixel value
- For a pixel with intensity k, transform it using:

(L = number of level = 256)



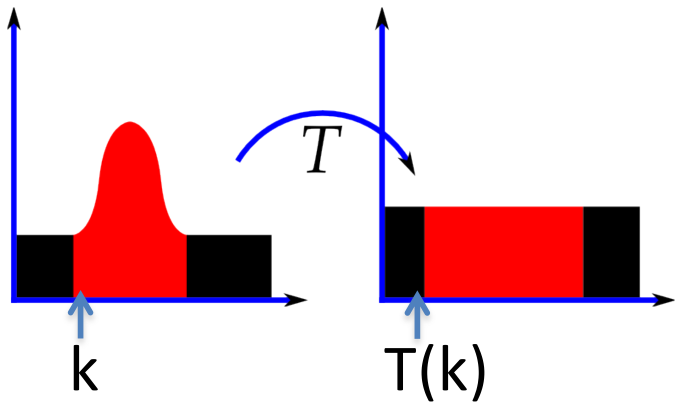
$$T(k) = \mathrm{floor}((L-1)\sum_{i=0}^{k} p_i) = \mathrm{floor}((L-1)F_X(k))$$

A pixel with value k

A pixel with value T(k)

Histogram equalization

- A mapping of pixel value
- For a pixel with intensity k, transform it using (L = number of level = 256)



$$T(k) = \text{floor}((L-1)\sum_{i=0}^{k} p_i) = \text{floor}((L-1)F_X(k))$$

- Algorithm:
  1. Compute cdf at k [*]
  2. Multiply by L-1, then floor(.)
  3. The result is the new intensity value

[*] normalize cdf to [0,1] by:

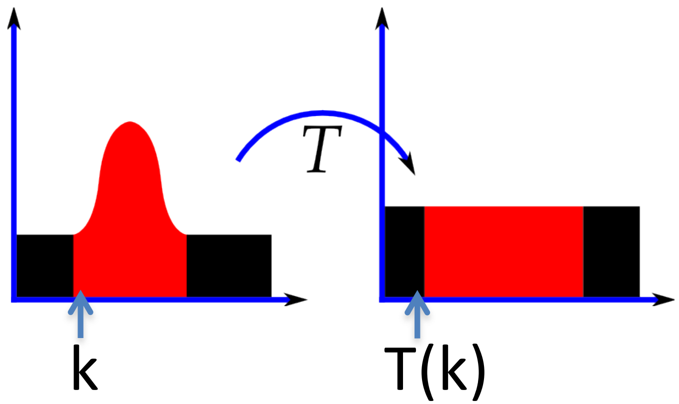
$$\frac{acc_k - acc_{min}}{acc_{max} - acc_{min}}$$

acc_{max}:

?

Histogram equalization

- A mapping of pixel value
- For a pixel with intensity k, transform it using (L = number of level = 256)



$$T(k) = \text{floor}((L-1)\sum_{i=0}^{k} p_i) = \text{floor}((L-1)F_X(k))$$

- Algorithm:
  1. Compute cdf at k [*]
  2. Multiply by L-1, then floor(.)
  3. The result is the new intensity value

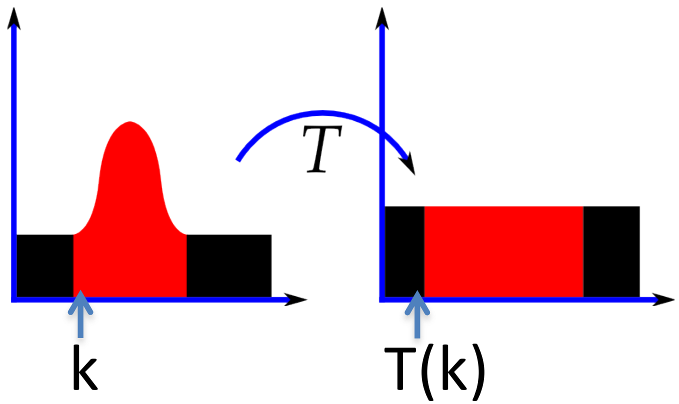
[*] normalize cdf to [0,1] by:

$$\frac{acc_k - acc_{min}}{acc_{max} - acc_{min}}$$

acc_{max}:
Number of pixels

Histogram equalization

Where does the formula come from?



$$T(k) = \text{floor}((L-1)\sum_{i=0}^{k} p_i) = \text{floor}((L-1)F_X(k))$$

- We want a transformation T(k) that will give an output image whose histogram is flat.
- The transformation should be a monotonically increasing function – this prevents artifacts created by reversals of intensity.

Histogram equalization

Where does the formula come from?

- The motivation for this transformation comes from thinking of the intensities of pixels before and after equalization as continuous random variables X, Y on [0, L − 1].

- Y defined by:

$$Y = T(X) = (L-1)\int_0^x f_X(x)dx$$

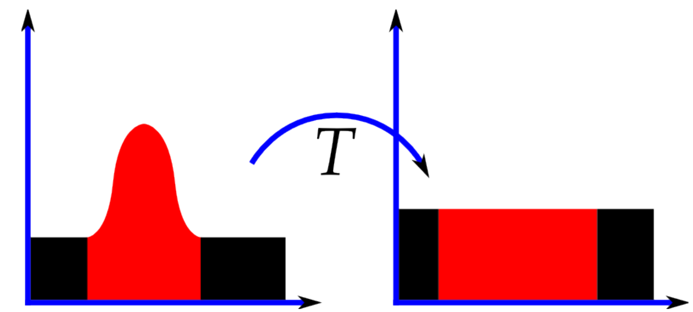
Histogram equalization

Where does the formula come from?

$$Y = T(X) = (L - 1) \int_0^x f_X(x)dx$$

- $f_X(x)$ is the probability density function of the original pixel values.
- T is the cumulative distributive function of X multiplied by (L − 1).
- Assume for simplicity that T is differentiable and invertible. It can then be shown that Y defined by T(X) is uniformly distributed on [0, L − 1], namely that $f_Y(y) = {}^{1}/_{L-1}$

Histogram equalization



Summary of justification:

- Discrete case:

$$T(k) = \text{floor}((L-1)\sum_{i=0}^{k} p_i) = \text{floor}((L-1)F_X(k))$$

- Continuous case: T(.) transforms a continuous r.v. X ~ $f_X(x)$ into Y ~ $f_Y(y)$, so that $f_Y(y)$ = U[0,L-1]

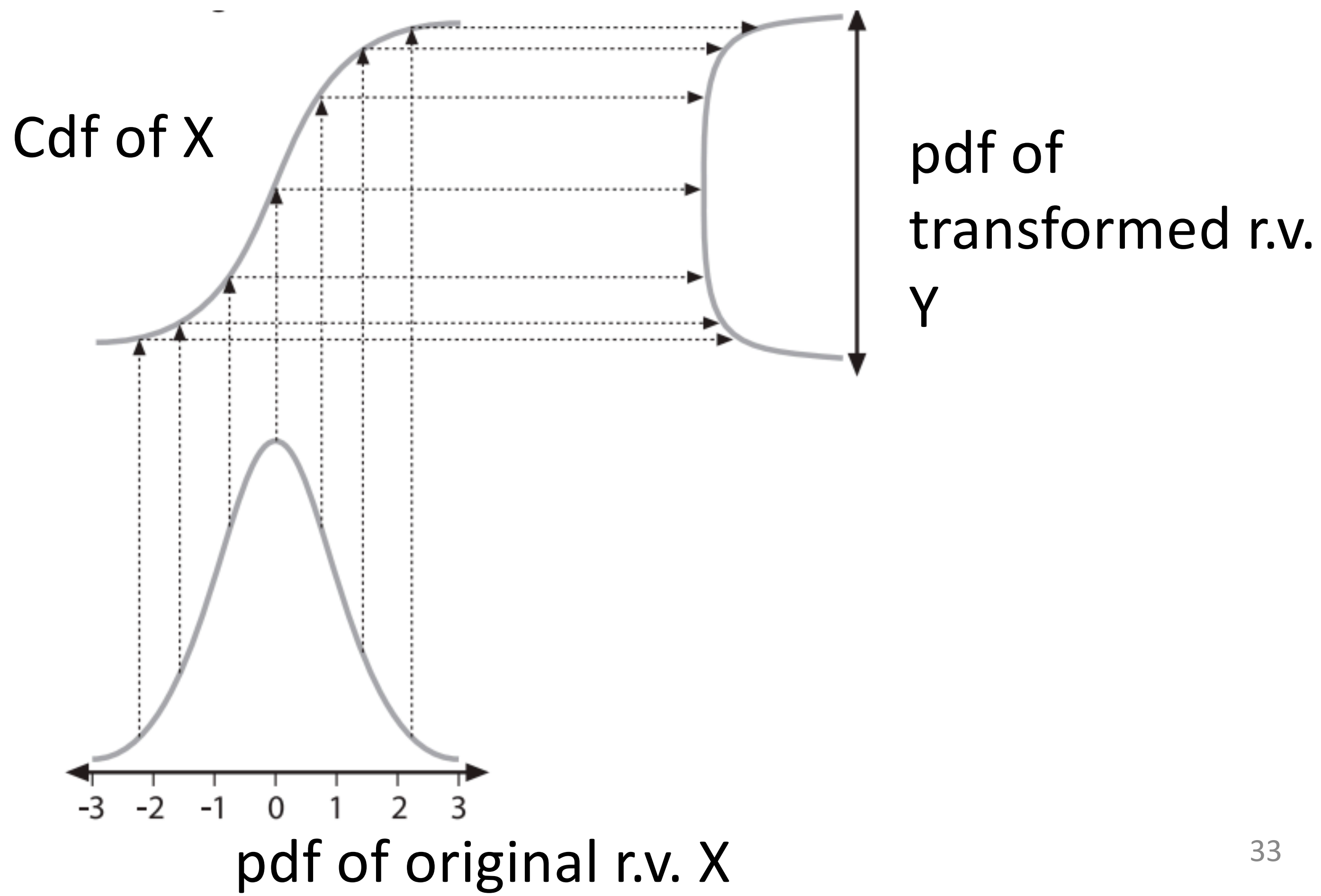
$$Y = T(X) = (L-1)F_X(X)$$

- Note that for any X ~ $f_X(x)$, Y is U[0,1] when the transformation is the cdf of X

$$Y = F_X(X)$$

Histogram equalization

- Cdf: transform a r.v. to a uniform one, ~U[0,1]



Histogram equalization

- Exercise

$$T(k) = \text{floor}((L-1)\sum_{i=0}^{k} p_i) = \text{floor}((L-1)F_X(k))$$

Input:
```
[[ 1  3  1  3]
 [ 2  3 10 11]
 [11 10  2  3]
 [ 1   2   3  3]]
```

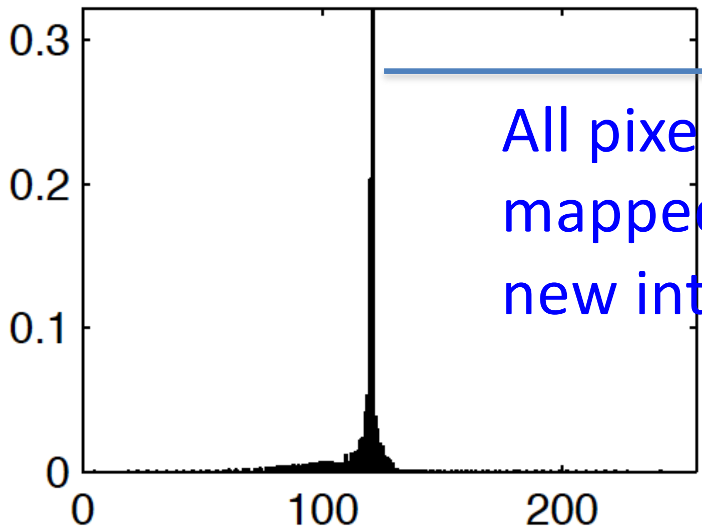
Output:
```
[[  0  176   0     176]
 [ 58 176   215 255]
 [255 215  58   176]
 [  0   58    176 176]]
```

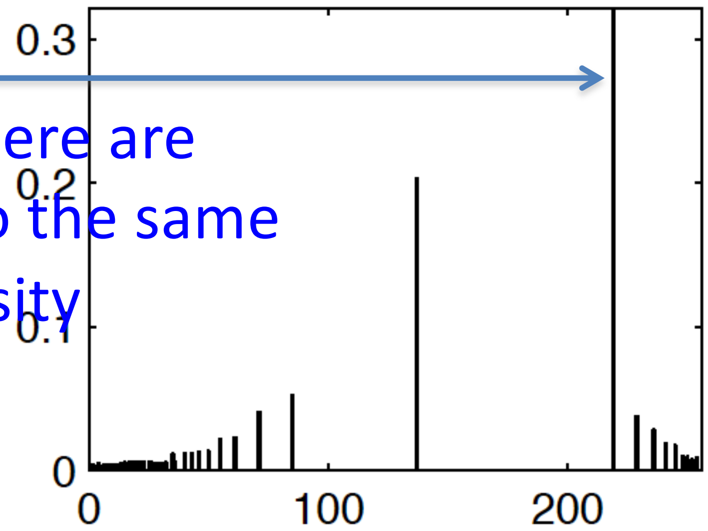
Input:
```
[[52  52  53  72]
  [72  72  53  53]
  [88  72  52  52]
  [88  88  53  53]]
```

Output:

Histogram equalization

- An example

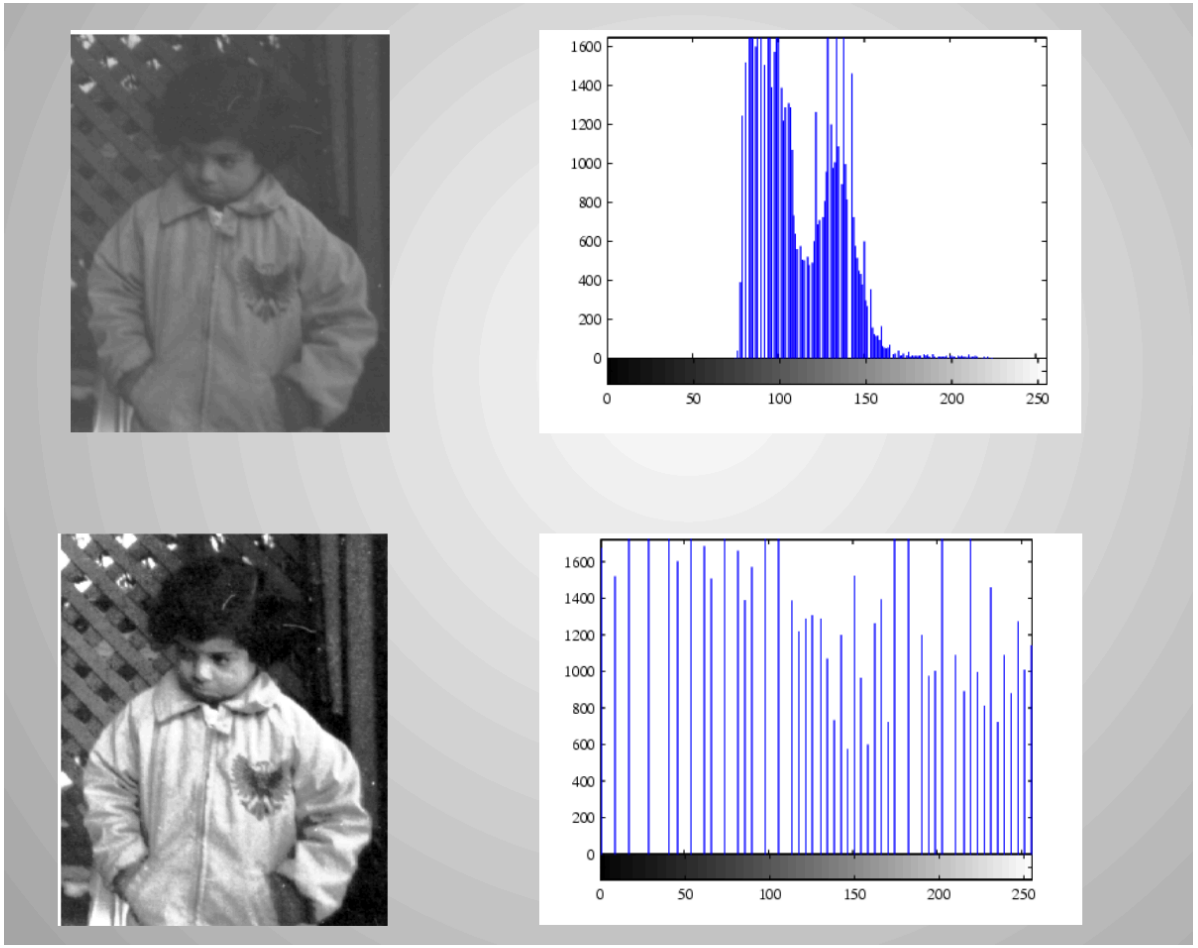




All pixels here are mapped to the same new intensity

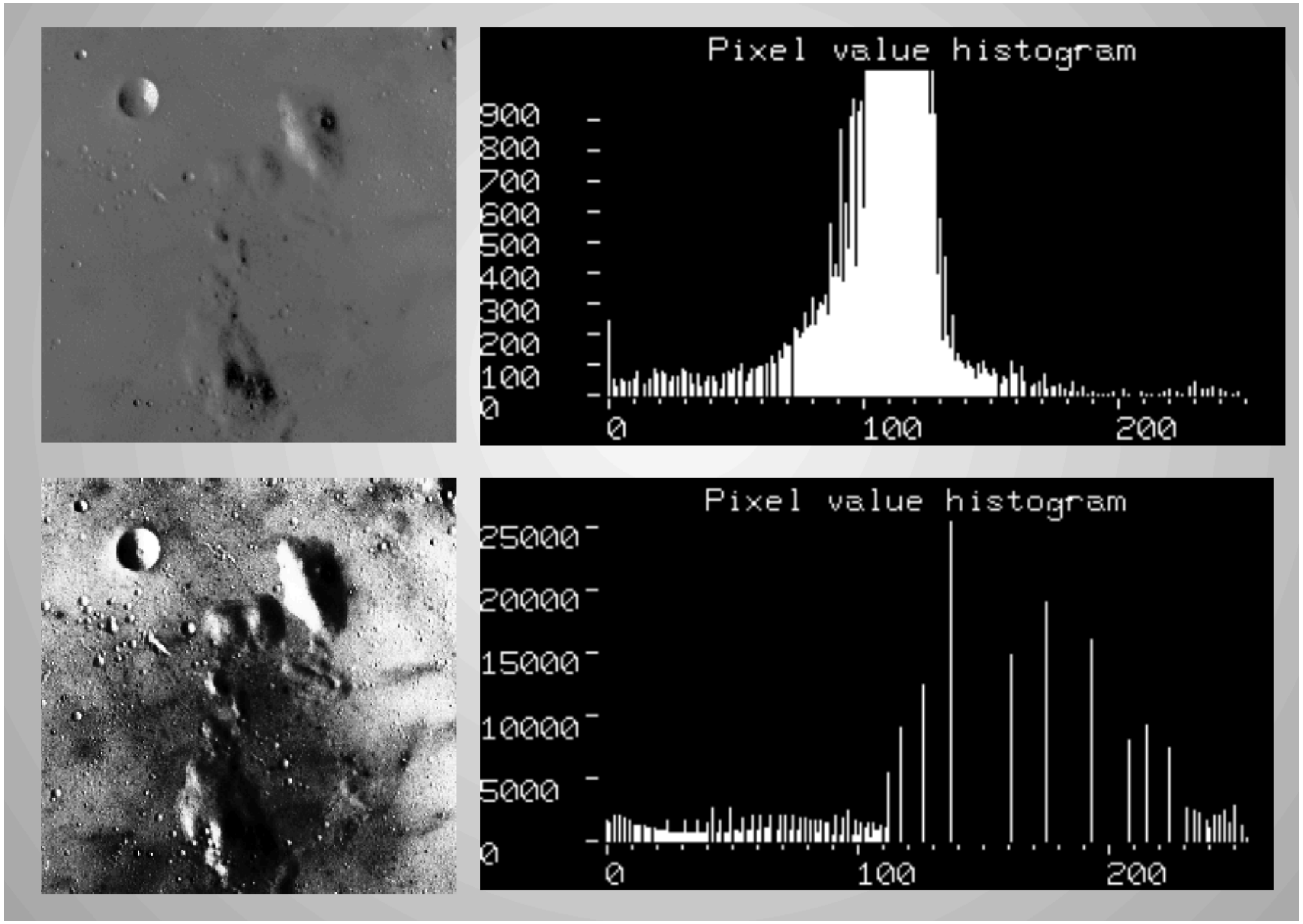
Histogram equalization

- An example



Histogram equalization

- An example



Histogram equalization

- Exercise solution:

$$T(k) = \text{floor}((L-1)\sum_{i=0}^{k} p_i) = \text{floor}((L-1)F_X(k))$$

Input:
```
[[ 1  3  1  3]
 [ 2  3 10 11]
 [11 10  2  3]
 [ 1   2   3  3]]
```

Output:
```
[[  0  176   0     176]
 [ 58 176   215 255]
 [255 215  58   176]
 [  0   58    176 176]]
```

Input:
```
[[52  52  53  72]
  [72  72  53  53]
  [88  72  52  52]
  [88  88  53  53]]
```

Output:
```
[[    0     0  106  191]
  [191  191  106  106]
  [255  191     0     0]
  [255  255  106  106]]
```

Image classification, data-driven approach, knn

Image classification/ object recognition

Which object is in the image?

Where is the object in the image?

Which pixels belong to the object in the image?

…

What is in the image?

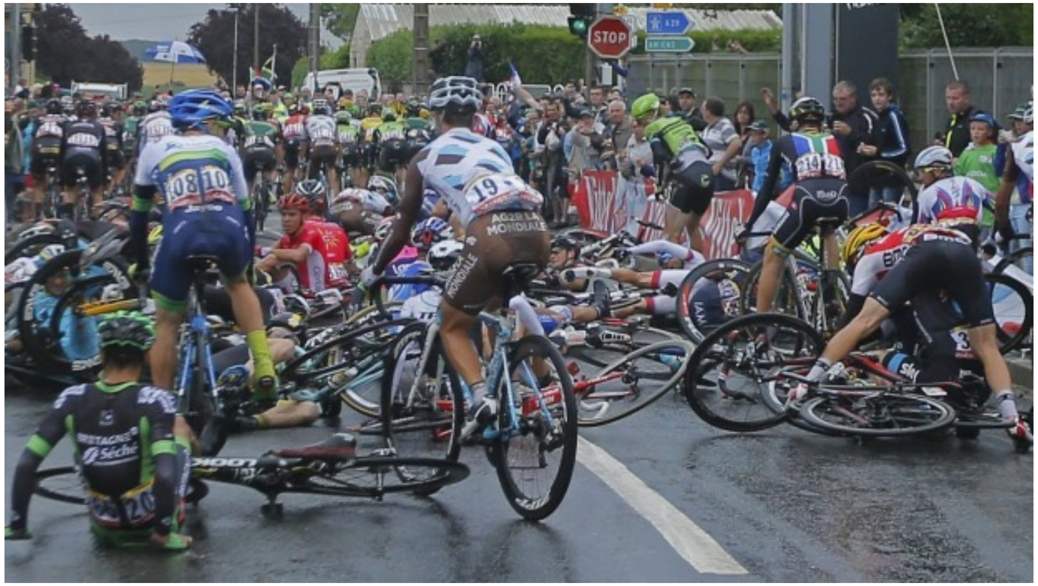


Image classification

- Given an input image, the algorithm produces one image label from a fixed set of classes (categories)

{fish, **soccer ball**, dog, boat}

- Image recognition (many classes)
  - 1000 categories in IMAGENET Large Scale Visual Recognition Challenge (ILSVRC): zebra, speedboat, lifeboat, …
  - 10,000+ categories in IMAGENET

zebra  speedboat  lifeboat 

Image classification

CIFAR-10

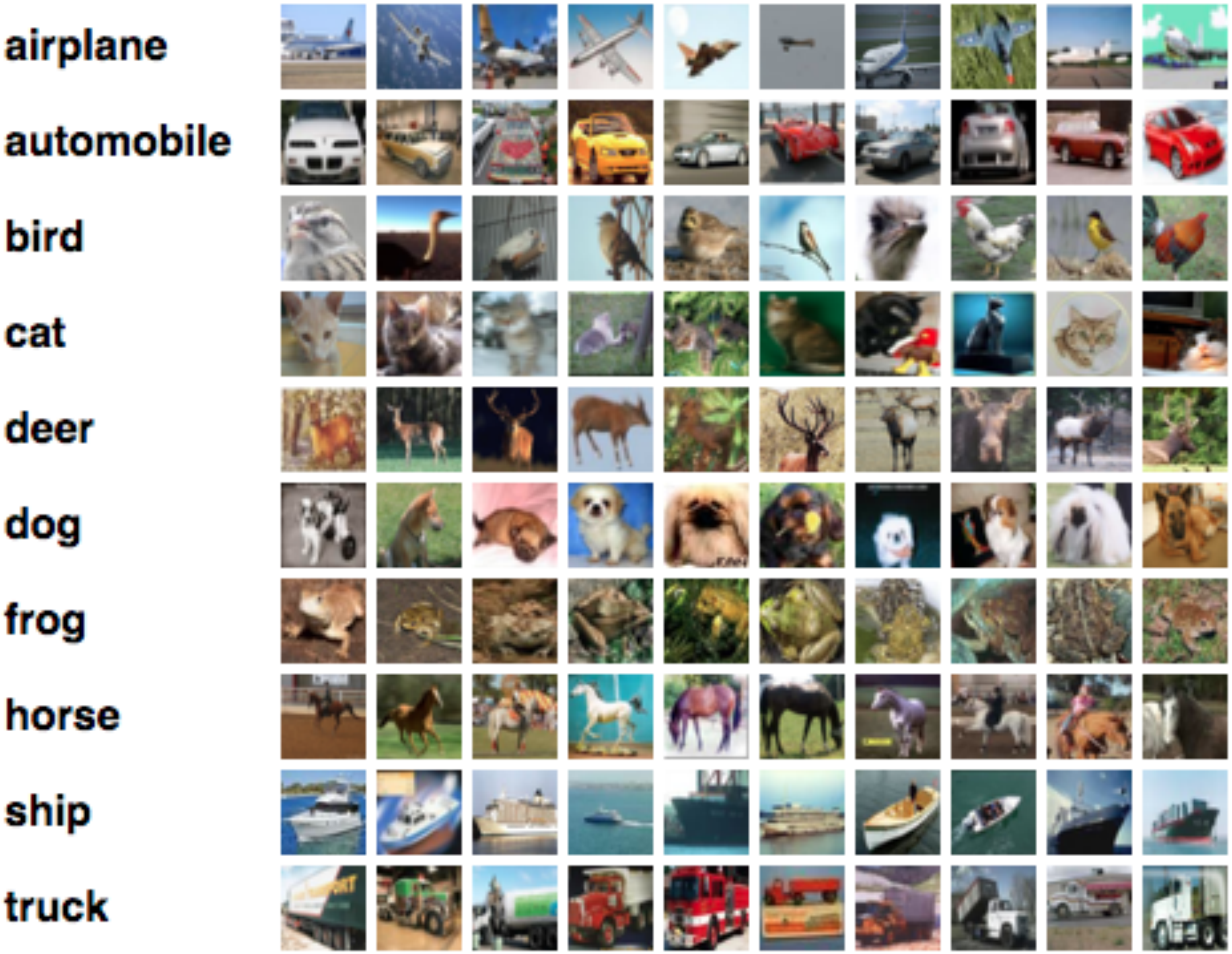
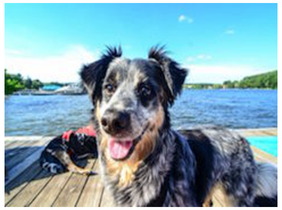


Image classification

- **Top-n accuracy**
- The algorithm outputs k confidence for each of the k classes

Test image: 

Algorithm outputs: {cat, dog, house, mouse} = {0.1, 0.2, 0.0, 0.7}

Top-1 class: {mouse}

Top-2 class: {mouse, dog}

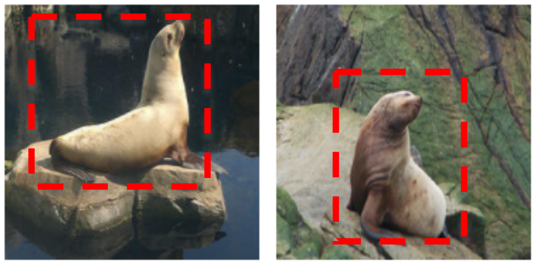
Incorrect for **top-1 accuracy**, correct for **top-2 accuracy** (ground truth is contained in the top-2 class)

- ILSVRC: Top-1, Top-5 accuracy

Number of correct / Number of test image

Image classification is fundamental to many computer vision tasks

- **Object localization**
- For a given image, the algorithm produces a **class label** and a **bounding box**
- Evaluation: label that best matches the ground truth label for the image, and bounding box that overlaps with the ground truth
- Error: if predicted label does not match the ground truth, or the predicted bounding box has less than 50% overlap



sea lion

Image classification is fundamental to many computer vision tasks

- **Object detection**
- Given an image, an algorithm produces a set of annotations (ci,si,bi): class label ci, bounding box bi and confidence score si
- Penalize: objects in the image not annotated by algorithm, more than 1 annotations for the same object in the image

- apple
- table
- bowl
- plate rack
- lamp
- chair

200 categories in ILSVRC2017

Image classification

- Challenges
  - Primitive data: Computer sees a 3d array of intensity values
  - Different variation for a certain class
    - Viewpoint variation
    - Scale variation
    - Deformation
    - Occlusion
    - Background clutter
    - Intra-class variation

Image classification

Challenges: Sources of Image Variation

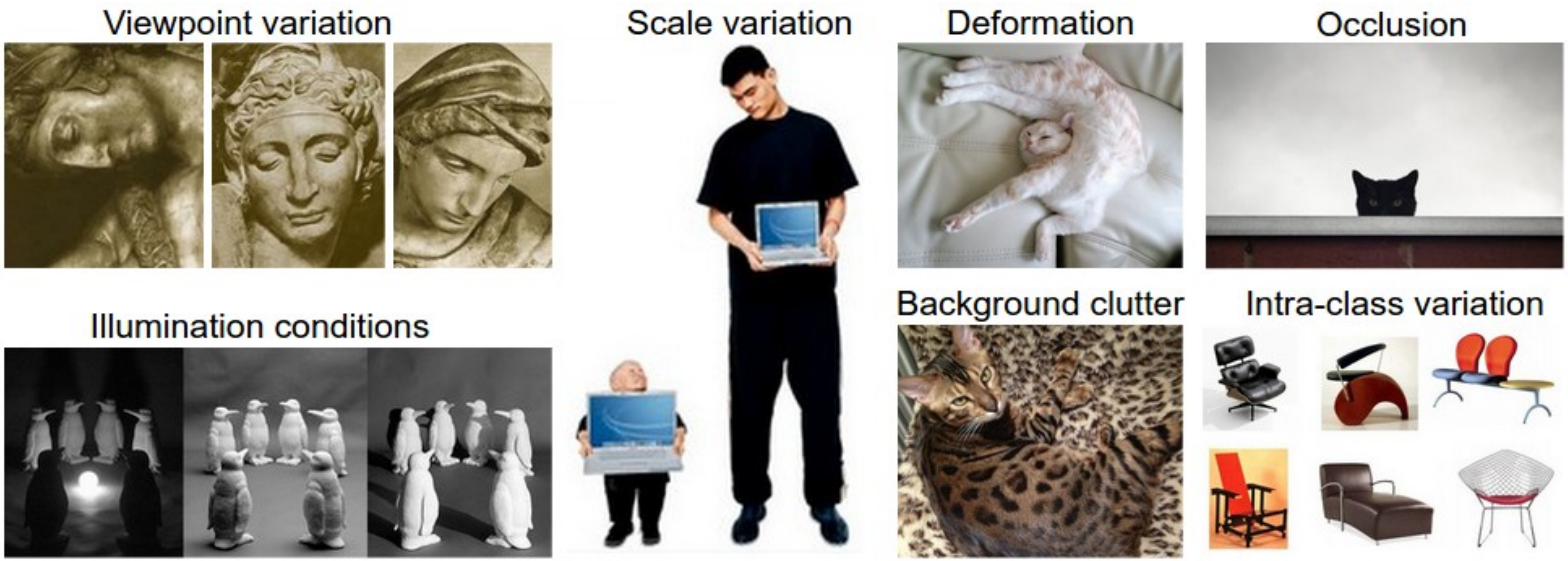


Image classification

- Challenges

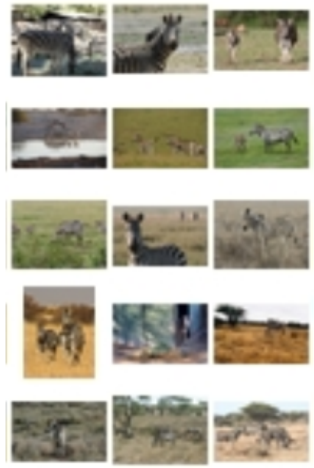
Deformation of non-rigid object

Background clutter

Position

Data driven approach

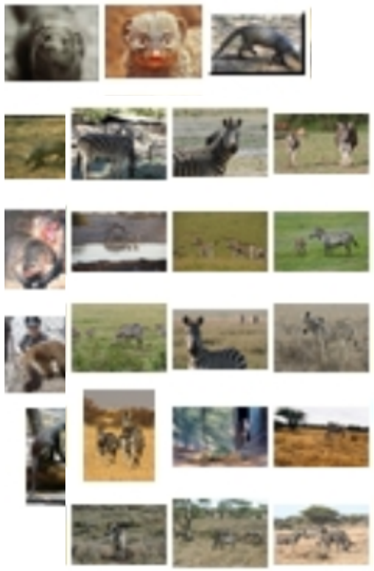
- Provide the computer with many examples of each class: **training data**
- Learn the visual appearance of each class: **learning algorithm**
- ILSVRC: 1.2 million images of 1000 categories
  - About 1k images per category



zebra

mongoose

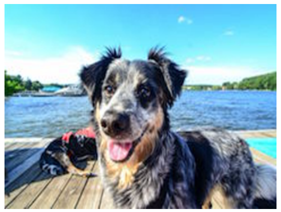
Data driven approach



Training/Learning (usually offline)

Training set: images with known class information → Learn a model using some algorithm → A model for this specific classification problem and classifier

Testing/Evaluation (usually online)



Testing set (with label during evaluation, without label in an application) → Classifier algorithm → Predicted class information for this new image (compare with ground-truth during evaluation)

Learning from examples

We want the algorithms to **learn** to do object recognition given examples of object categories

Training phase: examples images are shown to the algorithm

Testing phase: labelling of images never shown before

There are different modalities of supervision (fully supervised, unsupervised, semi-supervised, etc.)

Nearest Neighbor Classifier

- Given a test image, compare to every one of the training images

- Use the label of the 'closest training image' as the predicted label

Nearest Neighbor Classifier

- Consider an image as a vector (data point) in a very high dimensional vector space

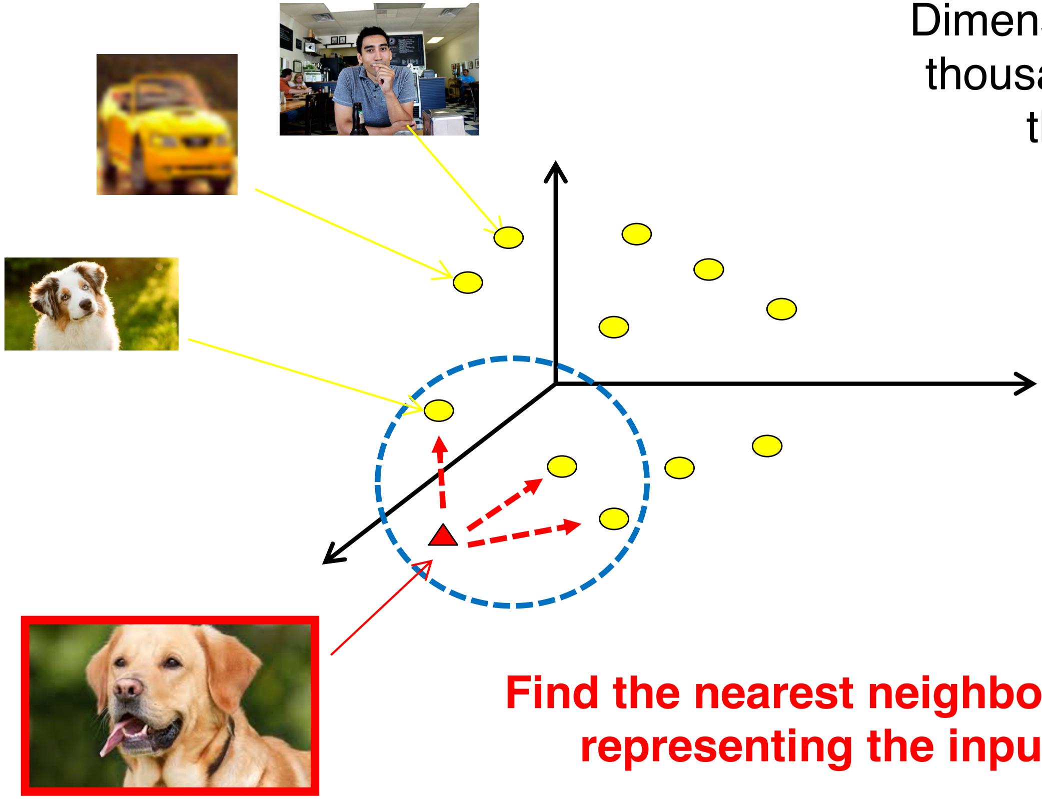
- 512x512x3 => a data point in the 786432-dim vector space

- Find the nearest neighbors of the vector representing the input test image

Nearest Neighbor Classifier

Each training image is represented by one high-dimensional vector (point)

Dimensionality can be thousands or tens of thousands



Find the nearest neighbors of the vector representing the input test image

Test image

Distance

- L2 distance (Euclidean distance)

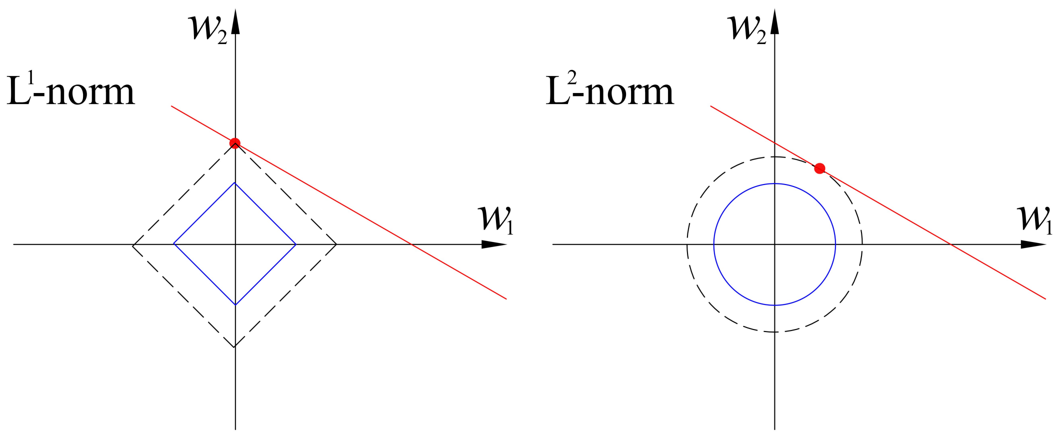
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

- L1 distance (Manhattan distance)
  - Sum of abs difference

$$d_1(I_1, I_2) = \sum_p \left|I_1^p - I_2^p\right|$$

Distance

- L1/L2 circle / ball
- A circle is a set of points with a fixed distance from a point (center)



L1 is more ‘restricted’, sensitive to rotation of coord system
L2 emphasizes dimensions with large differences

k-Nearest Neighbor Classifier (k-NN)

- Find the k closest images (nearest neighbors)

- Use them to vote on the label of the test image

k-Nearest Neighbor Classifier (k-NN)

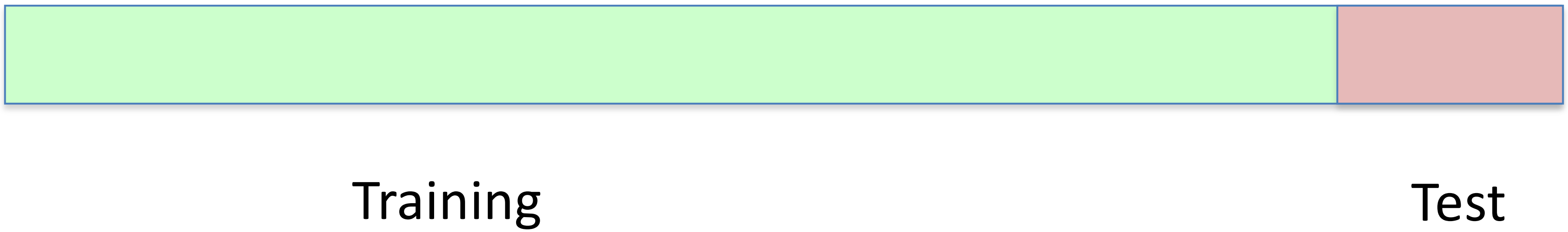
- How to determine k?

- k is a hyperparameter: related to the design of the machine learning algorithm

- Another hyperparameter: L1 norm or L2 norm

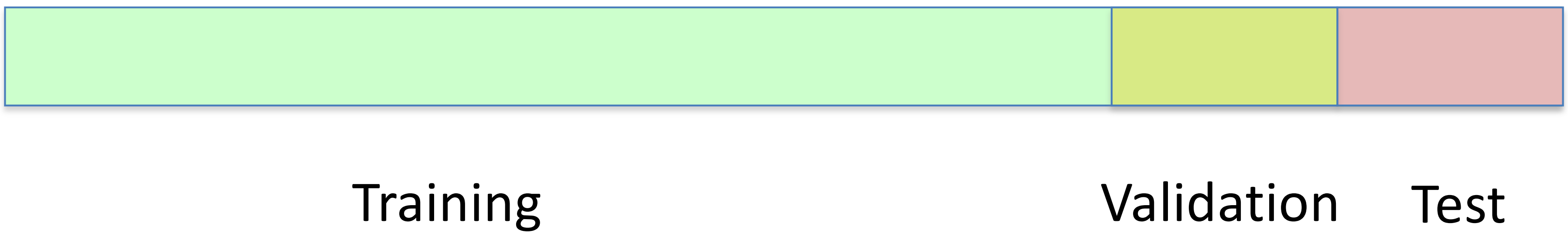
Validation set for hyperparameter tuning

- Use test set to tune the hyperparameter
- Not appropriate, as your model will *overfit* to the test data
- Poor generalization, significant degradation during deployment / testing for other datasets



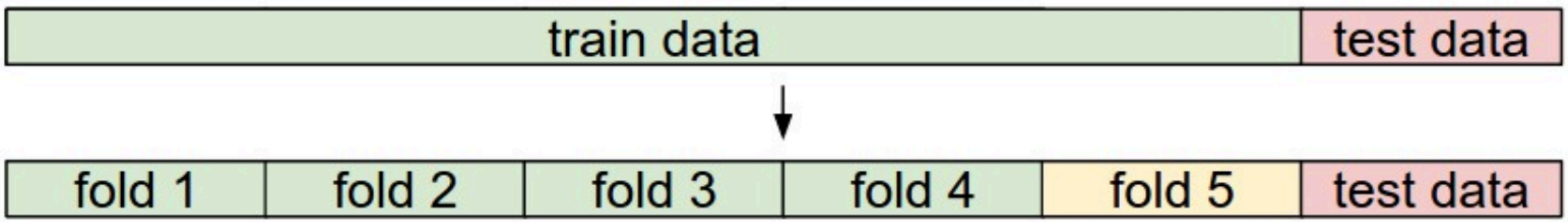
Validation set for hyperparameter tuning

- Partition the training set into a training set and a validation set
- Use validation set to *tune the hyperparameter*
- Use test set to evaluate the performance



Cross validation

- If the training dataset is small, can use cross validation
- 5-fold cross validation
  - For a given k (a certain setting of hyperparameters)
  - Divide the training dataset into 5 equal folds
  - Use 4 folds for training, 1 for validation
  - Repeat using another fold as the validation set
  - Average the performance



Issues of k-NN

- Memory expensive: need to remember all training data
- Computationally expensive during testing
  - Need to compare all training data
  - Not practical in an application

Original

Position shift

Intensity shift

Issues of k-NN

- Memory expensive: need to remember all training data
- Computationally expensive during testing
  - Need to compare all training data
  - Not practical in an application
- Approximate nearest neighbor (ANN) algorithms accelerate the search of the nearest neighbor
- Using image intensity value for distance comparison is not robust
  - Small position or intensity shift can result in large distance

Original

Position shift

Intensity shift

Today’s class

- Image histogram
- Image classification:
  - data-driven approach
  - K-nn

Next week's class

- Image classification:
  - Linear classifier
  - Gradient descent