1. Computer Vision – ILSVRC DATASET, RESNET

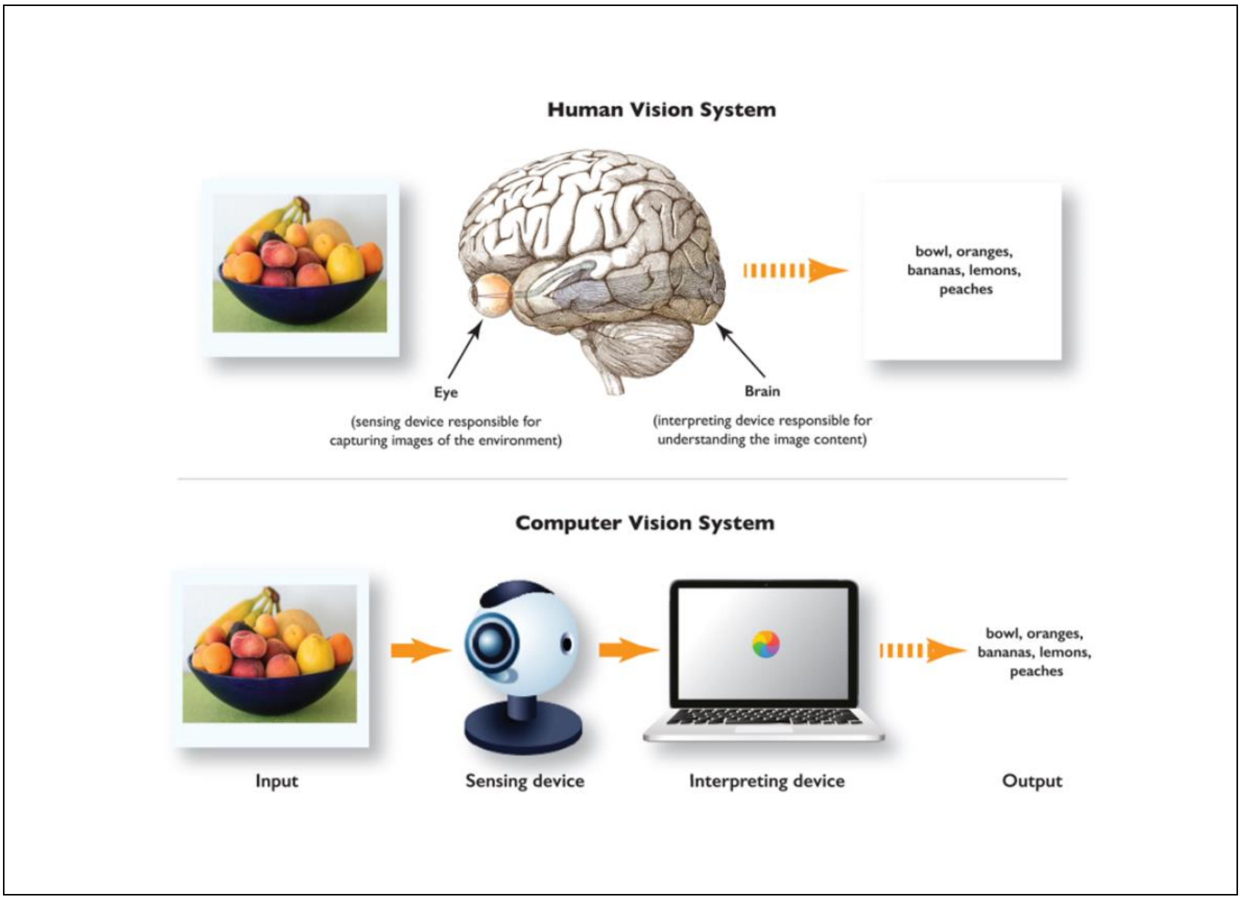
Contents

1. Computer Vision – ILSVRC DATASET, RESNET

1. Computer vision

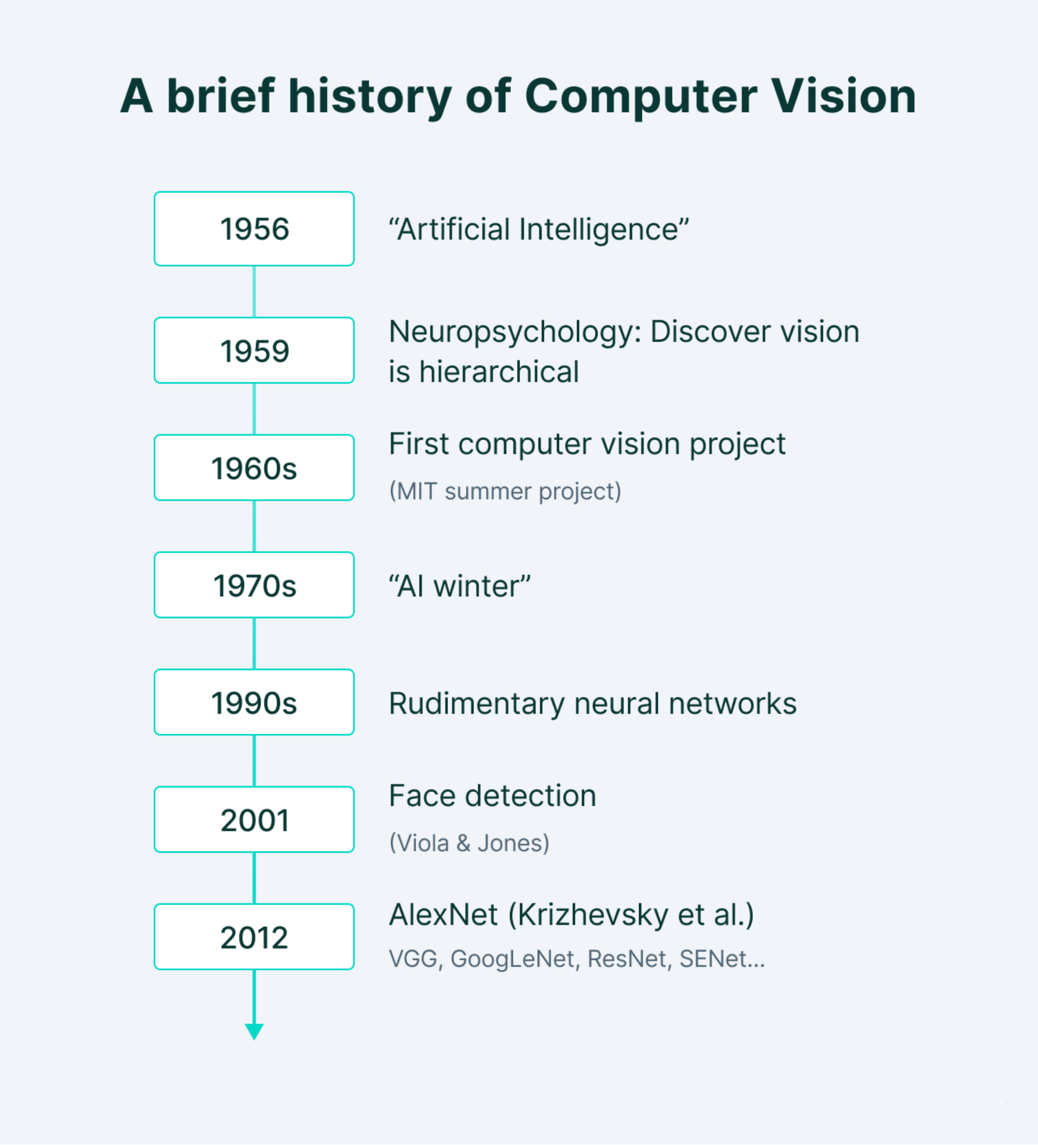
- ✓ Computer Vision is a subfield of Deep Learning and Artificial Intelligence.
- ✓ By using this human can teach computers to see and interpret the world around them.

2. Human & computer vision systems


Human Vision System
bowl, oranges, bananas, lemons, peaches
Eye
(sensing device responsible for capturing images of the environment)
Brain
(interpreting device responsible for understanding the image content)
Computer Vision System
bowl, oranges, bananas, lemons, peaches
Input
Sensing device
Interpreting device
Output

3. History of Computer Vision

- ✓ Machine vision, inspired by animal vision since 1959, evolved with advances in image technology.
- ✓ Key milestones: AI in the 1960s, OCR (Optical Character Recognition) in 1974, and complex topics by the 2000s.
  - Object identification
  - Facial recognition
  - Image Segmentation
  - Image Classification

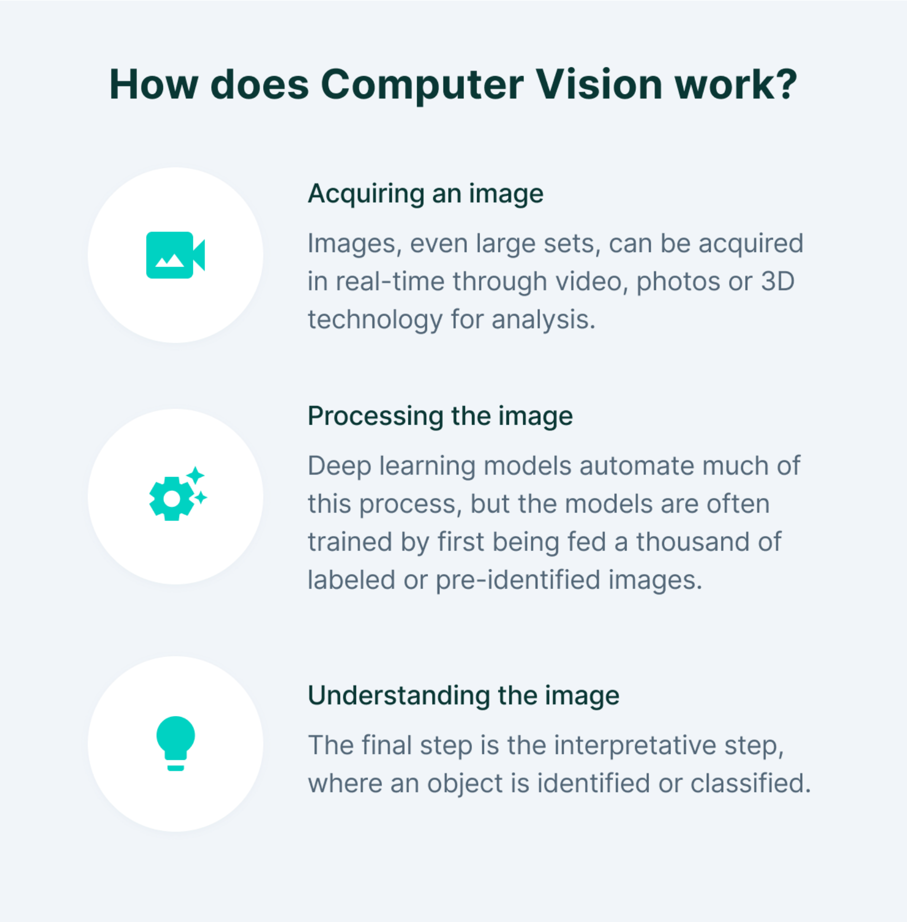


A brief history of Computer Vision

1956	"Artificial Intelligence"
1959	Neuropsychology: Discover vision is hierarchical
1960s	First computer vision project (MIT summer project)
1970s	"AI winter"
1990s	Rudimentary neural networks
2001	Face detection (Viola & Jones)
2012	AlexNet (Krizhevsky et al.) VGG, GoogLeNet, ResNet, SENet...

4. How Computer Vision works?

- ✓ There are mainly three steps,
  - Acquiring an image
  - Processing the image
  - Understanding the image



How does Computer Vision work?

Acquiring an image

Images, even large sets, can be acquired in real-time through video, photos or 3D technology for analysis.

Processing the image

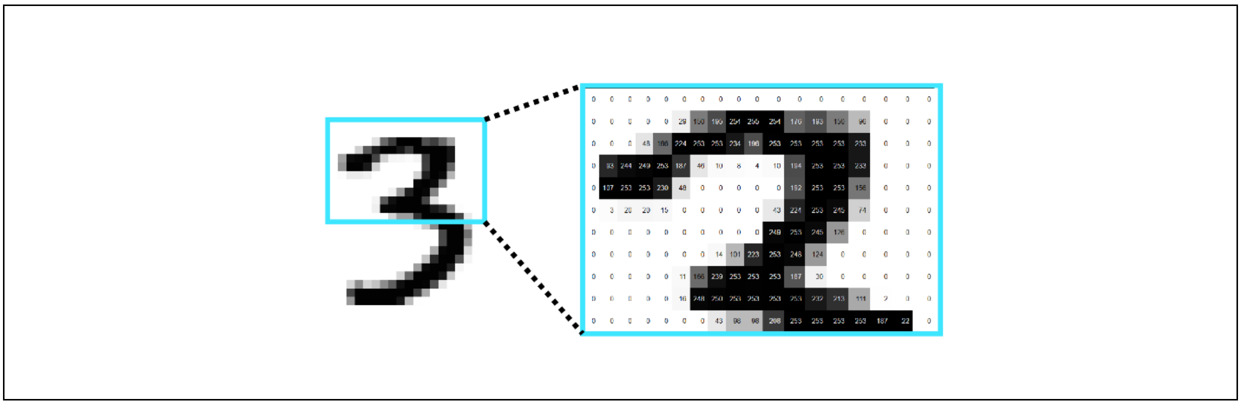
Deep learning models automate much of this process, but the models are often trained by first being fed a thousand of labeled or pre-identified images.

Understanding the image

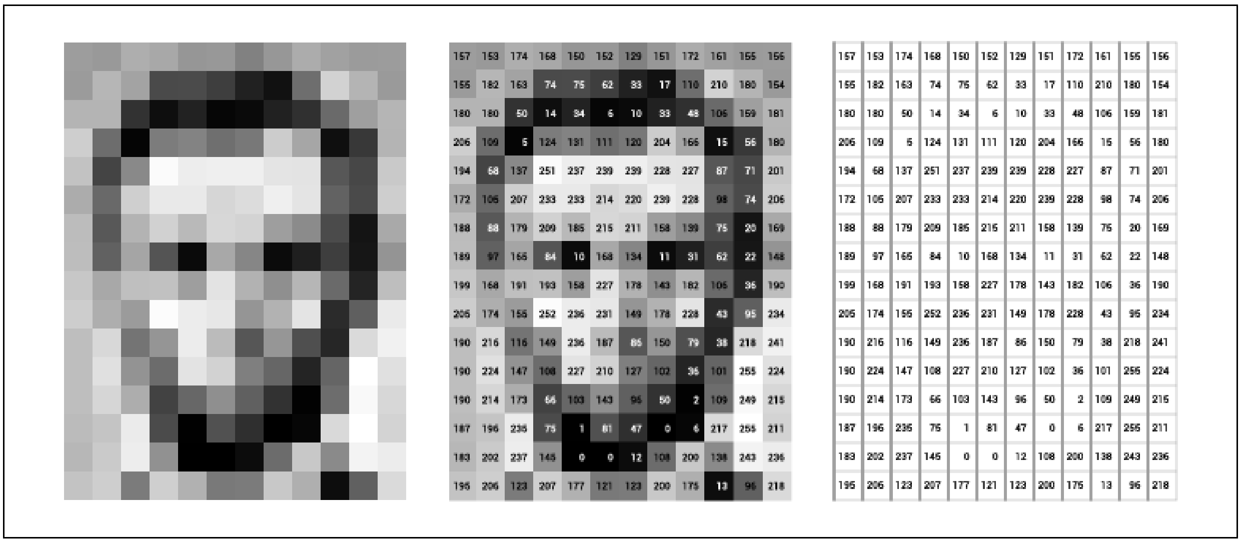
The final step is the interpretative step, where an object is identified or classified.

5. Image means pixel to computer

- ✓ An image is group of pixels.
- ✓ Computers process images as pixel arrays.
- ✓ Computer vision uses,
    - ○ Linear algebra for simple tasks and convolutions with pooling for complex tasks.



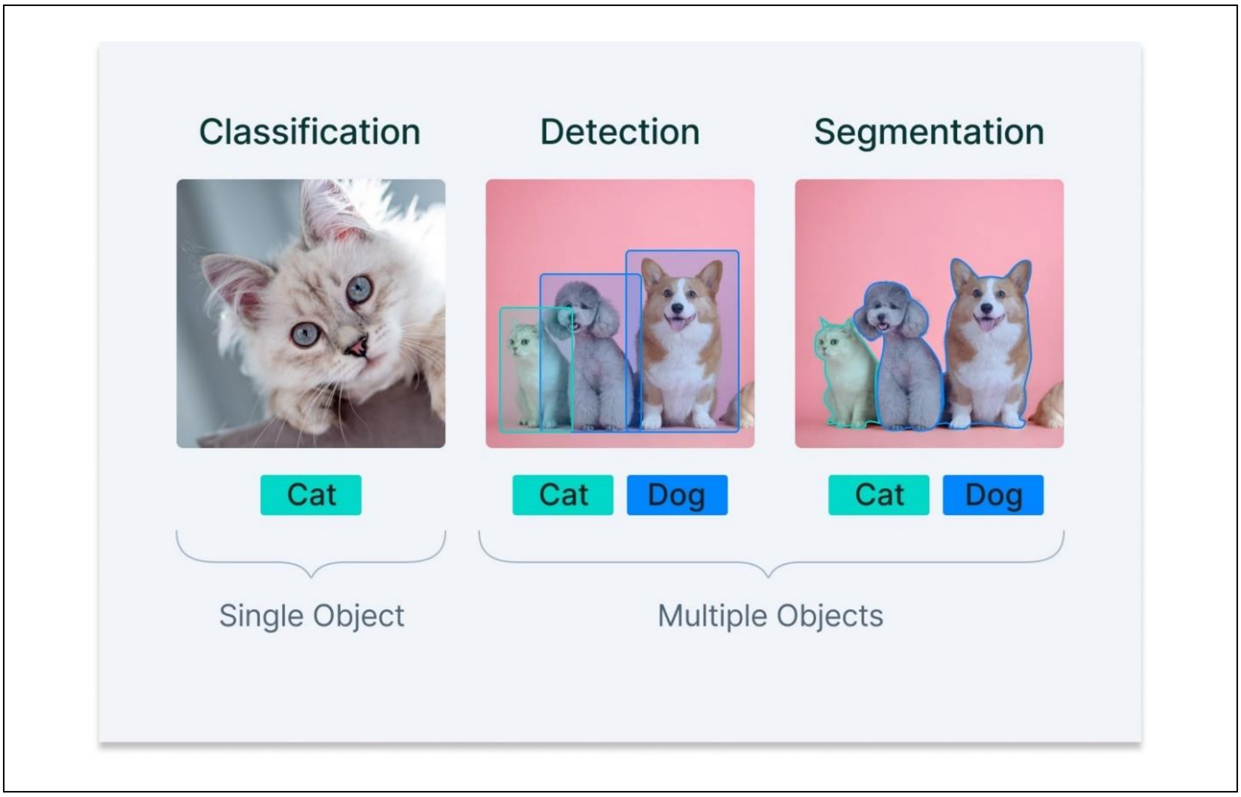
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	28	150	195	254	255	254	176	193	150	96	0	0	0
0	0	0	48	166	224	253	253	234	196	253	253	253	253	233	0	0	0
0	93	244	249	253	187	46	10	8	4	10	194	253	253	233	0	0	0
0	107	253	253	230	48	0	0	0	0	0	192	253	253	156	0	0	0
0	3	20	20	15	0	0	0	0	0	43	224	253	245	74	0	0	0
0	0	0	0	0	0	0	0	0	0	249	253	245	126	0	0	0	0
0	0	0	0	0	0	0	14	101	223	253	248	124	0	0	0	0	0
0	0	0	0	0	11	166	239	253	253	253	187	30	0	0	0	0	0
0	0	0	0	0	16	248	250	253	253	253	253	232	213	111	2	0	0
0	0	0	0	0	0	0	43	98	98	208	253	253	253	253	187	22	0



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

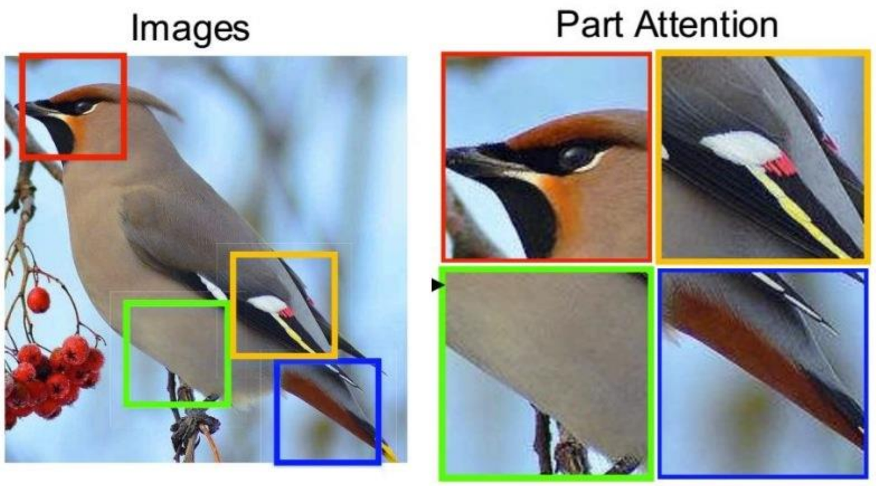
6. Common computer vision tasks

- ✓ Classification
- ✓ Detection
- ✓ Segmentation
- ✓ Face recognition
- ✓ Edge detection & etc



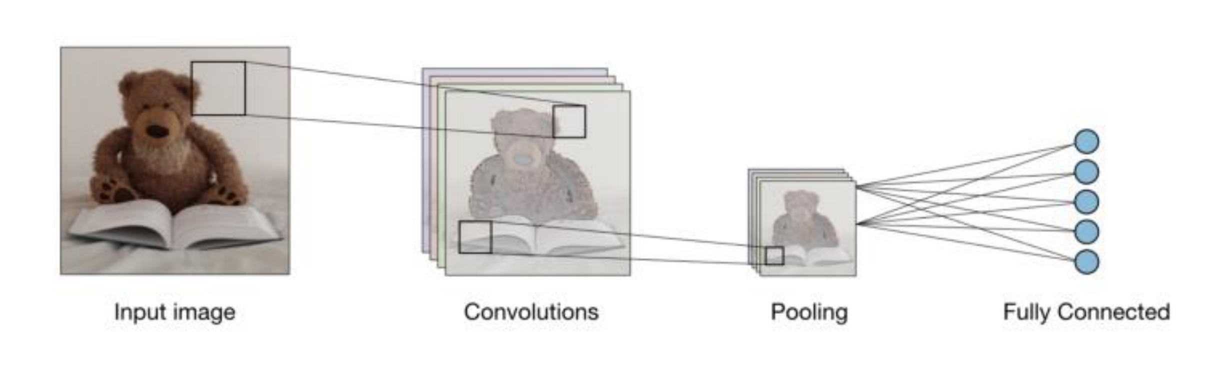
7. Human Image recognition

- ✓ Image recognition means, a computer or machine can see/observe the images.
- ✓ This is something like, **how our eye works.**
  - ○ If we see an image then automatically it split into parts of image.
  - ○ In next step our eyes can analyse sub-parts individually.
  - ○ By assembling these parts, our eyes process and interpret the image.
- ✓ The same process was implemented in CNN or Convolutional Neural Network.



8. Short Introduction: Convolutional Neural Network

- ✓ CNN is a type of deep learning model.
- ✓ This is commonly used for image recognition and classification tasks.
- ✓ The main parts in CNN,
  - ○ Convolution Layer
  - ○ Pooling Layer
  - ○ Fully-Connected Layer



8.1. Convolutional Layer

- ✓ In this layer, the input image is convolved (In maths: combine one function or series) with a set of learnable filters, also known as kernels.
- ✓ Every filter extracts different features from the input image by performing element-wise multiplication and summation operations.

8.2. Pooling layers

- ✓ Pooling layers reduce the size of the input by taking either the maximum or average value from small regions.
- ✓ This makes the network faster and helps prevent overfitting.

8.3. Fully Connected Layer

- ✓ Connects every neuron in the previous layer to every neuron in the next layer.
- ✓ This layer learns global features.

9. Image recognition task by CNN

- ✓ CNN having layers to recognize the image.
- ✓ Every layer can learn and delivery each task as,
    - The first layer can learn to detect basic edges and shapes.
    - The second layer can recognize these more detailed features.
    - The third layer can recognize parts of objects etc.
- ✓ This learning process helps the network to build a detailed understanding of the data.

10. ImageNet

10.1. The Birth of ImageNet

- ✓ Once upon a time, in the mid-2000s, a brilliant researcher named Fei-Fei Li.
  - https://profiles.stanford.edu/fei-fei-li
  - https://en.wikipedia.org/wiki/Fei-Fei_Li
- ✓ She had an idea like, just as humans learn by seeing countless objects and scenes throughout their lives.
- ✓ Her idea to make computers see and understand the world like humans.
- ✓ So, here needed a vast amount of data.
- ✓ She and her team aimed to create ImageNet.
- ✓ Started gathering images and created dataset
- ✓ The ImageNet is a very large collection of a dataset.
  - https://image-net.org/index.php

10.2. Dataset Information

- ✓ Gathering images, labelling them accurately was really a challenging task.
- ✓ Amazon Mechanical Turk thousands of team members manually verify and annotate the images.

10.3. Dataset Information

- ✓ This dataset contains,
  - More than **14** million images
  - More than **22** thousand groups or classes.
  - More than **1** million images that have bounding box annotations

10.4. The Birth of the Competition: ILSVRC

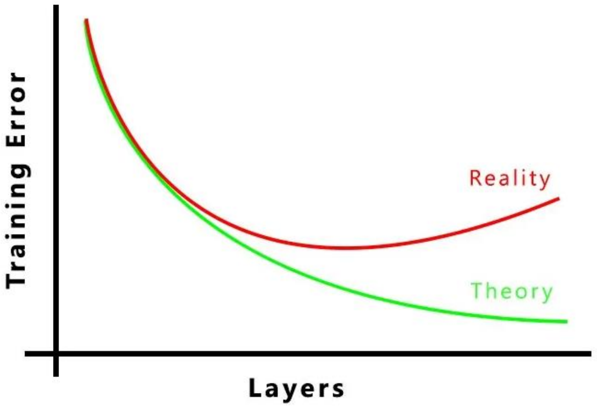
- ✓ ILSVRC the full form is,
    - ImageNet Large Scale Visual Recognition Challenge
- ✓ In 2010, Fei-Fei Li and her team introduced the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).
- ✓ This annual competition invited researchers from around the world to develop algorithms that could classify and detect objects within the ImageNet dataset.
- ✓ It quickly became the benchmark for measuring progress in computer vision.

11. The problem with deep neural networks

- ✓ To get the better results, the architecture used to add more and more layers, but this can lead to vanishing gradient issues.

12. If we increase more layers then…

- ✓ If we increase more layers (deep networks) then we may get the problem of vanishing/exploding gradients.



13. Vanishing / Exploding Gradients

- ✓ If we add more layers to the neural network then one of the problems is like, gradients can become too small (vanishing) or too large (exploding).
- ✓ Due to this Neural Network cannot learn effectively.

14. How to solve Vanishing / Exploding Gradients problem?

- ✓ We can solve this problem by using **ResNet**.

15. History of ResNet

- ✓ ResNet was implemented by Kaiming He and his team, in 2015.
- ✓ This research team achieved top results for object detection and object detection with localization tasks.

15.1. ResNet Paper

- ✓ This research paper written for Deep Residual Learning for Image Recognition.
- ✓ Find the below link for research paper,
    - ○ https://arxiv.org/abs/1512.03385

15.2. Winner of ILSVRC 2015

- ✓ ResNet won the champion for ILSVRC 2015 in,
    - ○ Image classification, detection, and localization

16. ResNet Introduction

- ✓ ResNet stands for Residual Network.
- ✓ It's a type of deep learning model.
- ✓ The main goal of ResNet (Residual Network) is to address the vanishing gradient problem.
- ✓ ResNet proves that, it is easy to train very deep neural networks.

17. ResNet Example

Program Name

ResNet Example
demo1.py

```
import numpy as np

from keras.preprocessing import image
from keras.applications import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras.applications.resnet50 import decode_predictions

model = ResNet50(weights = 'imagenet')

def load_and_preprocess_image(img_path):
        img = image.load_img(img_path, target_size = (224, 224))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis = 0)
        img_array = preprocess_input(img_array)

        return img_array


img_path = '1.jpeg'
img_array = load_and_preprocess_image(img_path)
predictions = model.predict(img_array)
decoded_predictions = decode_predictions(predictions, top =
1)[0]

for i, (imagenet_id, label, score) in
enumerate(decoded_predictions):
        print(f"{i + 1}: {label} ({score:.2f})")
```

Output

```
goldfish
```

2. Computer Vision – Features

Contents

2. Computer Vision – Features

1. Computer Vision

- ✓ Computer vision is the automated extraction of information from images.
- ✓ The goal of computer vision is to understand the content of digital images.
- ✓ Computer vision is a field of study focused on the problem of helping computers to see.

2. Computer Vision and Image Processing

- ✓ Computer vision is different from image processing.
- ✓ Image processing is the process of creating a new image from an existing image
- ✓ Computer vision system may require image processing.
- ✓ Examples of image processing
    - ○ Increase image brightness or change color.
    - ○ Cropping the images
    - ○ Removing digital noise from an image means low light levels.

3. Computer vision applications

- ✓ Object Classification
  - ○ Broad category of object in image
- ✓ Object Identification
  - ○ Type of a given object in image
- ✓ Object Verification
  - ○ Is the object existing in images?
- ✓ Object Detection
  - ○ Where are the objects in images?
- ✓ Object Landmark Detection
  - ○ Key points for the object in the image
- ✓ Object Segmentation
  - ○ What pixels belong to the object in the image
- ✓ Object Recognition
  - ○ What objects are in this photograph and where are they?

4. Automatic Feature Extraction

- ✓ Features can be automatically learned and extracted from raw image data.

5. Reuse the models

- ✓ We can reuse the existing models

6. Superior Performance

- ✓ Computer vision is very good to get the best results

3. Computer Vision – Load & work with images

Contents

3. Computer Vision – Load & work with images

1. Loading the images in different ways

- ✓ Pillow is open-source python library.
- ✓ By using this we can load and manipulate images.

```
pip install Pillow
```

Program Name

Loading image by using pillow
demo1.py

```
from PIL import Image

image = Image.open("opera_house.jpg")

image.show()
```

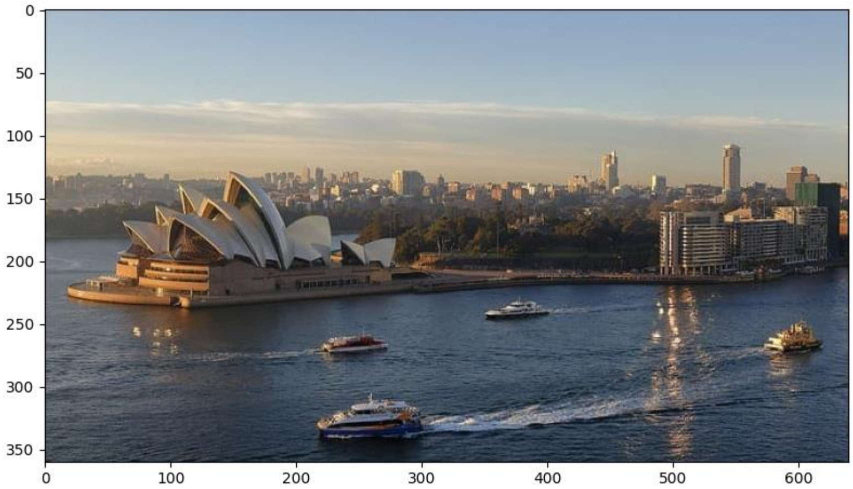
output

Program Name

Loading image by using matplotlib
demo2.py

```
from matplotlib import image
from matplotlib import pyplot

data = image.imread("opera_house.jpg")

pyplot.imshow(data)
pyplot.show()
```

output



Program Name

Converting normal images to grayscale
demo3.py

```
from PIL import Image

image = Image.open("opera_house.jpg")

gs_image = image.convert(mode = "L")

image.show()
gs_image.show()
```

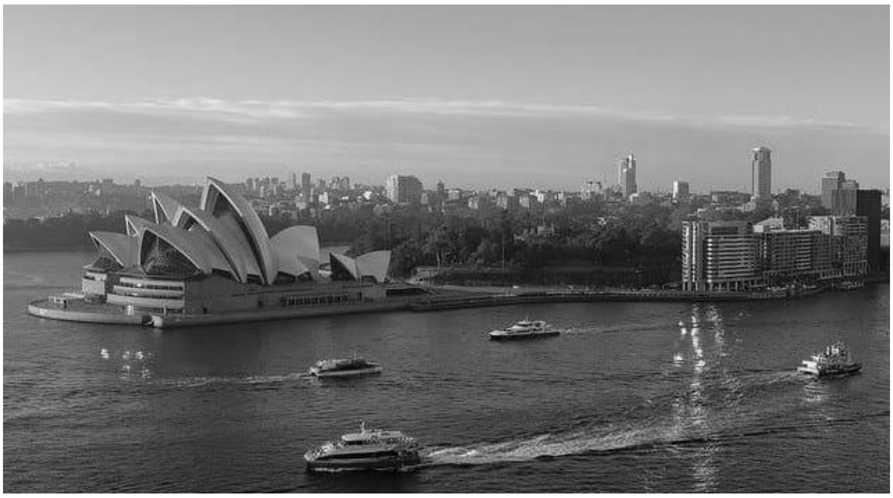
output

Program Name

Converting normal images to grayscale, save image
demo4.py

```
from PIL import Image

image = Image.open("opera_house.jpg")

gs_image = image.convert(mode = "L")

gs_image.save("opera_house_grayscale.jpg")

image2 = Image.open("opera_house_grayscale.jpg")

image2.show()
```

output



Program Name

Resize the images
demo5.py

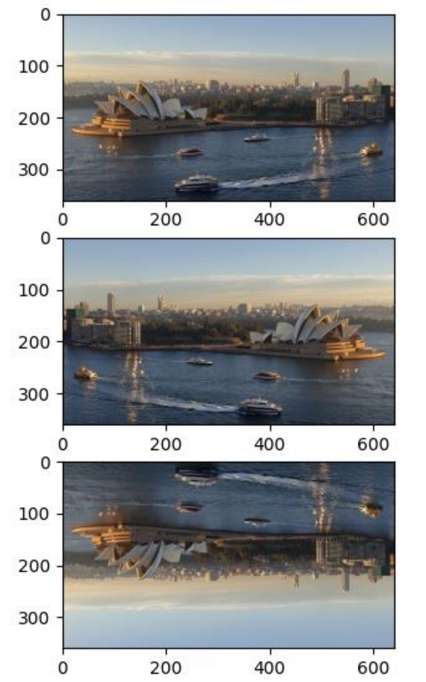
```
from PIL import Image

image = Image.open("opera_house.jpg")

image.thumbnail((100,100))

image.show()
```

output

Program	Flipping the image
Name	demo6.py

```
from PIL import Image
from matplotlib import pyplot

image = Image.open("opera_house.jpg")

hoz_flip = image.transpose(Image.FLIP_LEFT_RIGHT)
ver_flip = image.transpose(Image.FLIP_TOP_BOTTOM)

pyplot.subplot(311)
pyplot.imshow(image)

pyplot.subplot(312)
pyplot.imshow(hoz_flip)

pyplot.subplot(313)
pyplot.imshow(ver_flip)

pyplot.show()
```

output

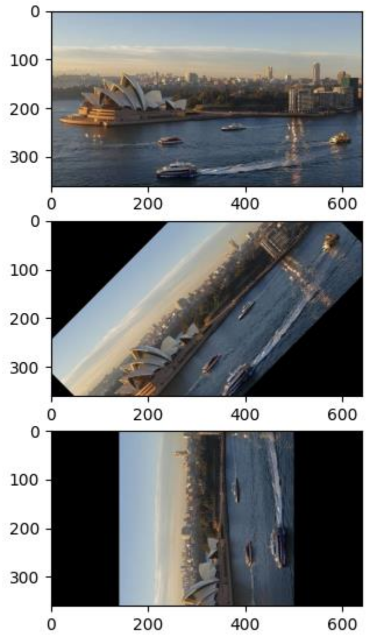

0
100
200
300
0
200
400
600
0
100
200
300
0
200
400
600
0
100
200
300
0
200
400
600

Program Name

Rotate the images
demo7.py

```
from PIL import Image
from matplotlib import pyplot

image = Image.open("opera_house.jpg")

pyplot.subplot(311)
pyplot.imshow(image)

pyplot.subplot(312)
pyplot.imshow(image.rotate(45))

pyplot.subplot(313)
pyplot.imshow(image.rotate(90))

pyplot.show()
```

output



4. Computer Vision – Load images with keras

Contents

4. Computer Vision – Load images with keras

1. Loading the images with keras

- ✓ We can load & save the images with keras package

Program Name

Loading image by using keras
demo1.py

```
from tensorflow.keras.utils import load_img

img = load_img("opera_house.jpg")

img.show()
```

output

Program Name

Loading & saving the image by using keras
demo2.py

```
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import save_img
from tensorflow.keras.utils import img_to_array

img = load_img("opera_house.jpg", color_mode= "grayscale")
img_array = img_to_array(img)

save_img("bondi_beach_grayscale.jpg", img_array)
img = load_img("bondi_beach_grayscale.jpg")

img.show()
```

output

5. Computer Vision – Image Data Augmentation in Keras

Contents

5. Computer Vision – Image Data Augmentation

1. Image Data Augmentation in Keras

- ✓ Image data augmentation is a technique, by using this we can create new images.
- ✓ It helps to increase the training dataset.
- ✓ If more data/images then model will gives good accuracy.

2. Different type of Image Data Augmentation in Keras

- ✓ Image Augmentation With ImageDataGenerator
- ✓ Horizontal and Vertical Shift Augmentation
- ✓ Horizontal and Vertical Flip Augmentation
- ✓ Random Rotation Augmentation
- ✓ Random Brightness Augmentation
- ✓ Random Zoom Augmentation

3. ImageDataGenerator class

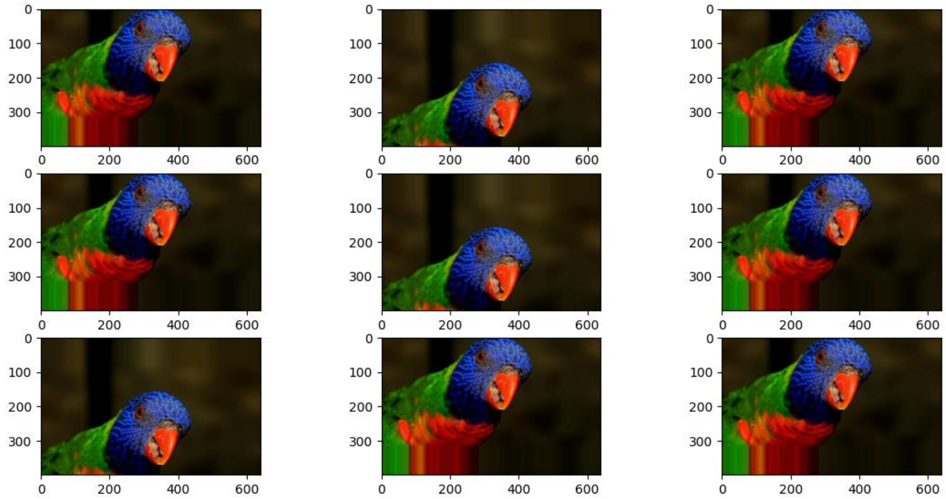
- ✓ **ImageDataGenerator** is a predefined class
    - ○ Here keyword arguments plays important role.
- ✓ **Keyword arguments.**
    - ○ width_shift_range
    - ○ height_shift_range
    - ○ horizontal_flip
    - ○ rotation_range

Program Name

Image data augmentation, **width_shift_range**
demo1.py

```
from numpy import expand_dims
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot

img = load_img("bird.jpg")
data = img_to_array(img)
samples = expand_dims(data, 0)

datagen = ImageDataGenerator(width_shift_range = [-80, 80])

it = datagen.flow(samples, batch_size = 1)

for i in range(9):
        pyplot.subplot(330 + 1 + i)
        batch = it.next()
        image = batch[0].astype("uint8")
        pyplot.imshow(image)

pyplot.show()
```

output

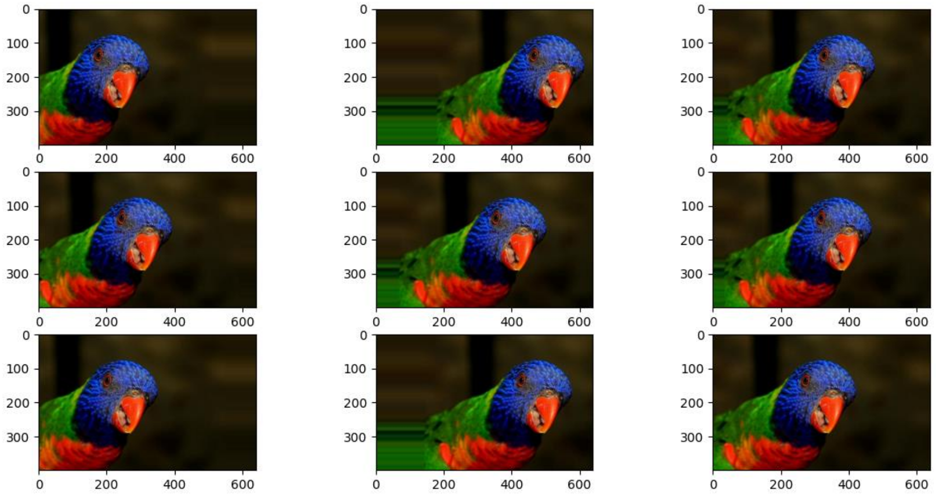


Program Name

Image data augmentation, **height_shift_range**
demo2.py

```
from numpy import expand_dims
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot

img = load_img("bird.jpg")
data = img_to_array(img)
samples = expand_dims(data, 0)

datagen = ImageDataGenerator(height_shift_range = 0.8)

it = datagen.flow(samples, batch_size = 1)

for i in range(9):
        pyplot.subplot(330 + 1 + i)
        batch = it.next()
        image = batch[0].astype("uint8")
        pyplot.imshow(image)

pyplot.show()
```

output

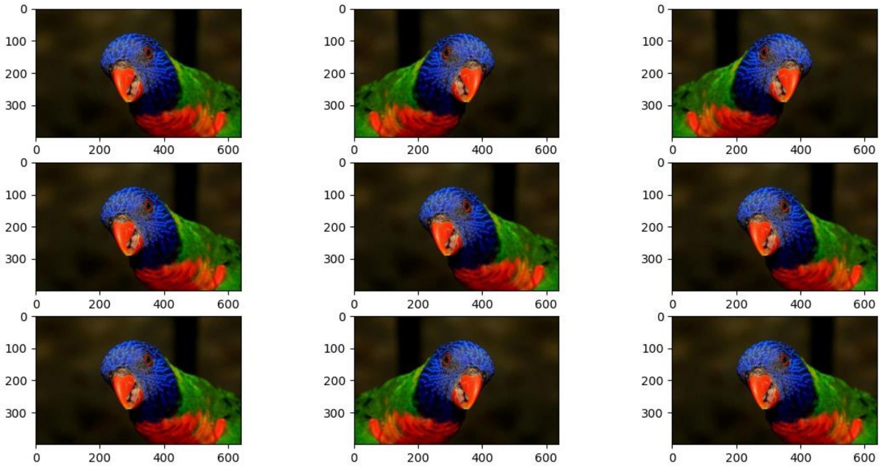


Program Name

Image data augmentation, **horizontal_flip**
demo3.py

```
from numpy import expand_dims
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot

img = load_img("bird.jpg")
data = img_to_array(img)
samples = expand_dims(data, 0)

datagen = ImageDataGenerator(horizontal_flip = True)

it = datagen.flow(samples, batch_size = 1)

for i in range(9):
        pyplot.subplot(330 + 1 + i)
        batch = it.next()
        image = batch[0].astype("uint8")
        pyplot.imshow(image)

pyplot.show()
```

output

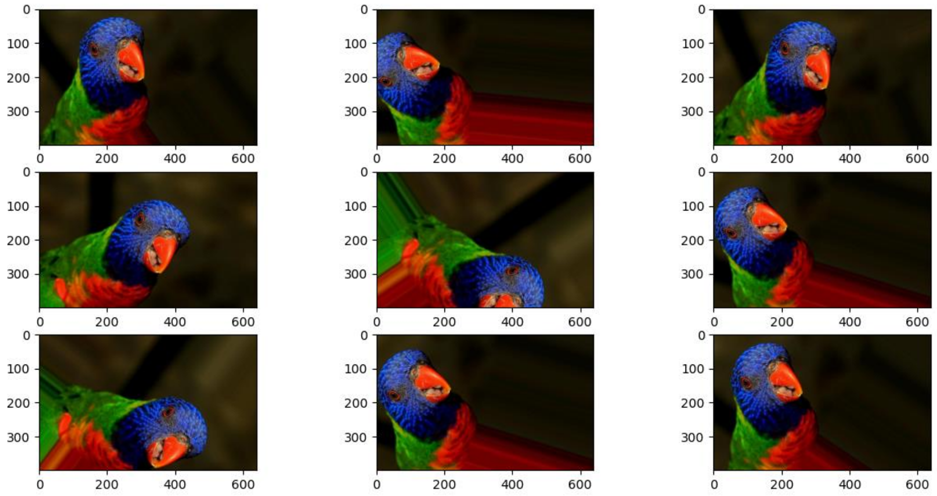


Program Name

Image data augmentation, **rotation_range**
demo4.py

```
from numpy import expand_dims
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot

img = load_img("bird.jpg")
data = img_to_array(img)
samples = expand_dims(data, 0)

datagen = ImageDataGenerator(rotation_range = 90)

it = datagen.flow(samples, batch_size = 1)

for i in range(9):
        pyplot.subplot(330 + 1 + i)
        batch = it.next()
        image = batch[0].astype("uint8")
        pyplot.imshow(image)

pyplot.show()
```

output



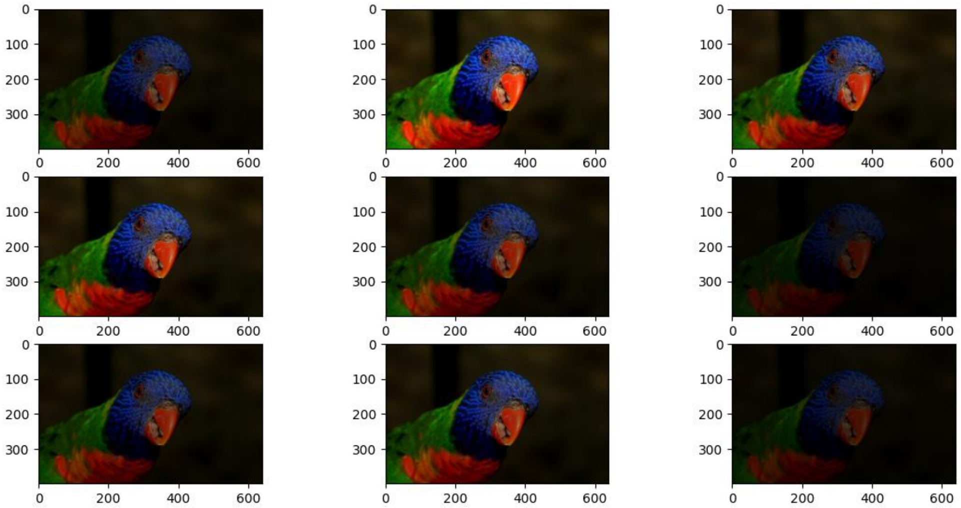
4. Random Brightness Augmentation

- ✓ The brightness of the image can be augmented by either randomly darkening images, brightening images, or both.
- ✓ **ImageDataGenerator** is a predefined class
    - Here keyword arguments plays important role.
- ✓ **Keyword arguments.**
    - brightness_range

Program Name	Image data augmentation, **brightness_range_range** demo5.py

```
from numpy import expand_dims
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot

img = load_img("bird.jpg")
data = img_to_array(img)
samples = expand_dims(data, 0)

datagen = ImageDataGenerator(brightness_range = [0.2, 1.0])

it = datagen.flow(samples, batch_size = 1)

for i in range(9):
        pyplot.subplot(330 + 1 + i)
        batch = it.next()
        image = batch[0].astype("uint8")
        pyplot.imshow(image)

pyplot.show()
```

output



5. Random Zoom Augmentation

- ✓ A zoom augmentation randomly zooms the image and either adds new pixel values around the image or interpolates pixel values respectively.
- ✓ **ImageDataGenerator** is a predefined class
    - Here keyword arguments plays important role.
- ✓ **Keyword arguments.**
    - zoom_range
        - For example, [0.7, 1.3] means 70% (zoom in) and 130% (zoom out).

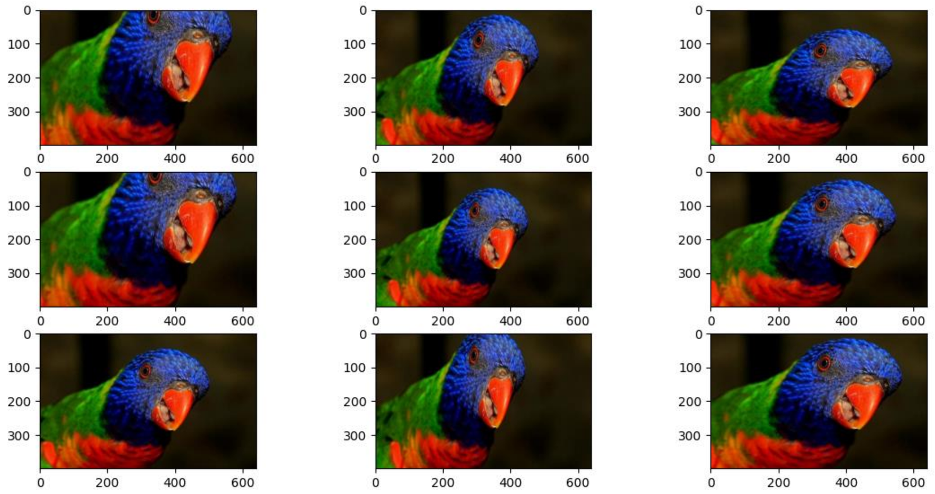
Program Name

Image data augmentation, **zoom_range**

demo6.py

```
from numpy import expand_dims
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot

img = load_img("bird.jpg")
data = img_to_array(img)
samples = expand_dims(data, 0)

datagen = ImageDataGenerator(zoom_range = [0.5,1.0])

it = datagen.flow(samples, batch_size = 1)

for i in range(9):
        pyplot.subplot(330 + 1 + i)
        batch = it.next()
        image = batch[0].astype("uint8")
        pyplot.imshow(image)

pyplot.show()
```

output



6. Computer Vision – Use Cases

Contents

6. Computer Vision – Use Cases

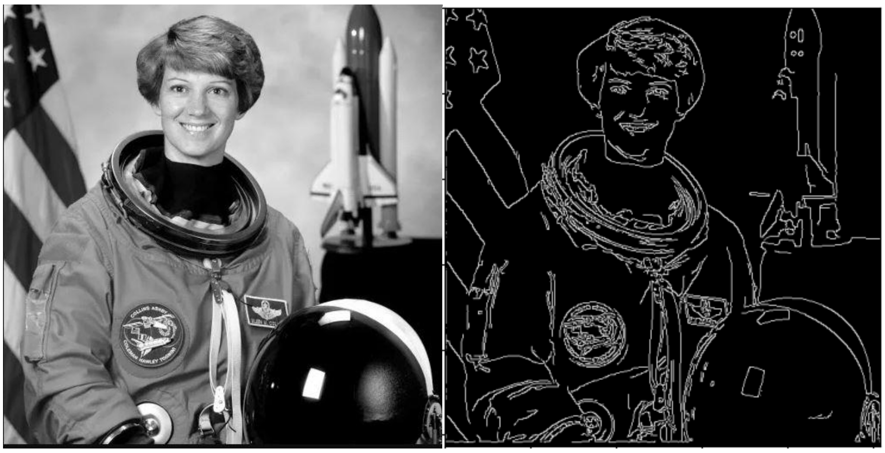
1. Edge & Contour Detection

- ✓ CV applications detect edges first and then collect other information.
- ✓ There are many edge detection algorithms, and the most popular is the Canny edge detector because it’s pretty effective compared to others.
- ✓ It’s also a complex edge-detection technique.
- ✓ Below are the steps for Canny edge detection:
  - ○ Reduce noise and smoothen image
  - ○ Calculate the gradient
  - ○ Non-maximum suppression
  - ○ Double the threshold
  - ○ Linking and edge detecting

Program Name	Canny edge detection demo1.py

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('1.jpg')
edges = cv2.Canny(img, 100, 200, 3, L2gradient = True)

plt.figure()
plt.title('Spider')
plt.imsave('dancing-spider-canny.png', edges, cmap = 'gray',
format='png')
plt.imshow(edges, cmap='gray')
plt.show()
```

Input



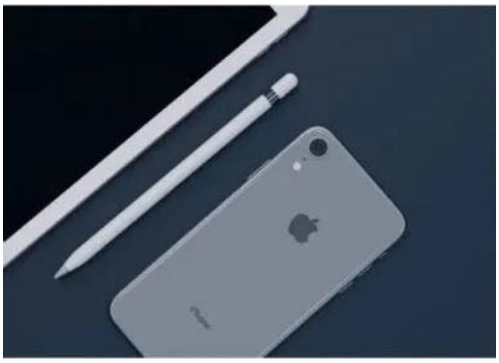
2. Contours

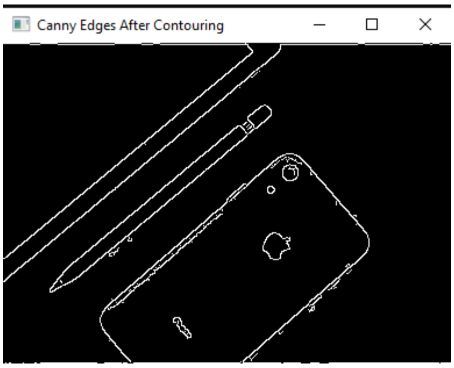
- ✓ Contours are lines joining all the continuous objects or points (along the boundary), having the same color or intensity.
- ✓ For example, it detects the shape of a leaf based on its parameters or border.
- ✓ Contours are an important tool for shape and object detection.
- ✓ The contours of an object are the boundary lines that make up the shape of an object as it is.
- ✓ Contours are also called outline, edges, or structure, for a very good reason.

Program Name	Contours demo2.py

```
import cv2
import numpy as np

image = cv2.imread('2.jpg')
cv2.waitKey(0)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

edged = cv2.Canny(gray, 30, 200)
cv2.waitKey(0)

contours, hierarchy = cv2.findContours(edged,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

cv2.imshow('Canny Edges After Contouring', edged)
cv2.waitKey(0)

cv2.drawContours(image, contours, -1, (0, 255, 0), 3)

cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input





Canny Edges After Contouring