

Google Classroom Code: mhxgl24

Deep Learning for Computer Vision

Deep Learning (DS-5006)

Dr. Adeel Mumtaz

Lecture 7

Fall, 2022



National University
Of Computer and Emerging Sciences

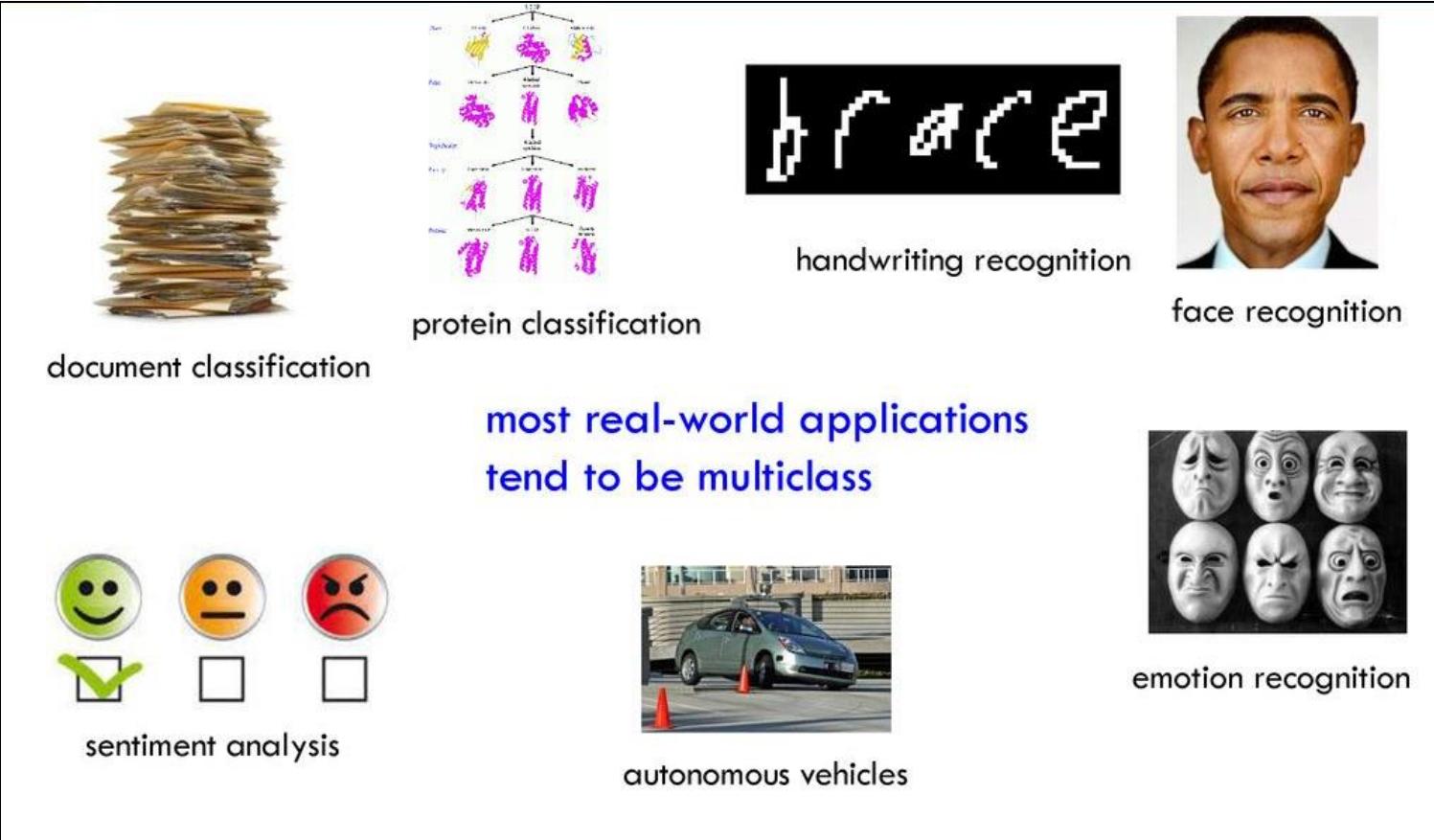
Contents

- Last Lecture
 - Multiclass Classification (Review)
 - Home Task 3
- Computer Vision
 - Introduction
 - History
 - Applications
- Basics of Image Processing
 - Types of Images
 - Pillow
 - OpenCV
- NCHW vs NHWC
- Torchvision
 - ImageFolder Dataset
 - Image transformations
 - Conversion transforms
 - Normalization Transforms
 - Data Augmentation
- Home Task-4

MULTICLASS CLASSIFICATION (REVIEW)

Multiclass Classification

- A problem is considered a multiclass classification problem if there are more than two classes



document classification

protein classification

handwriting recognition

face recognition

sentiment analysis

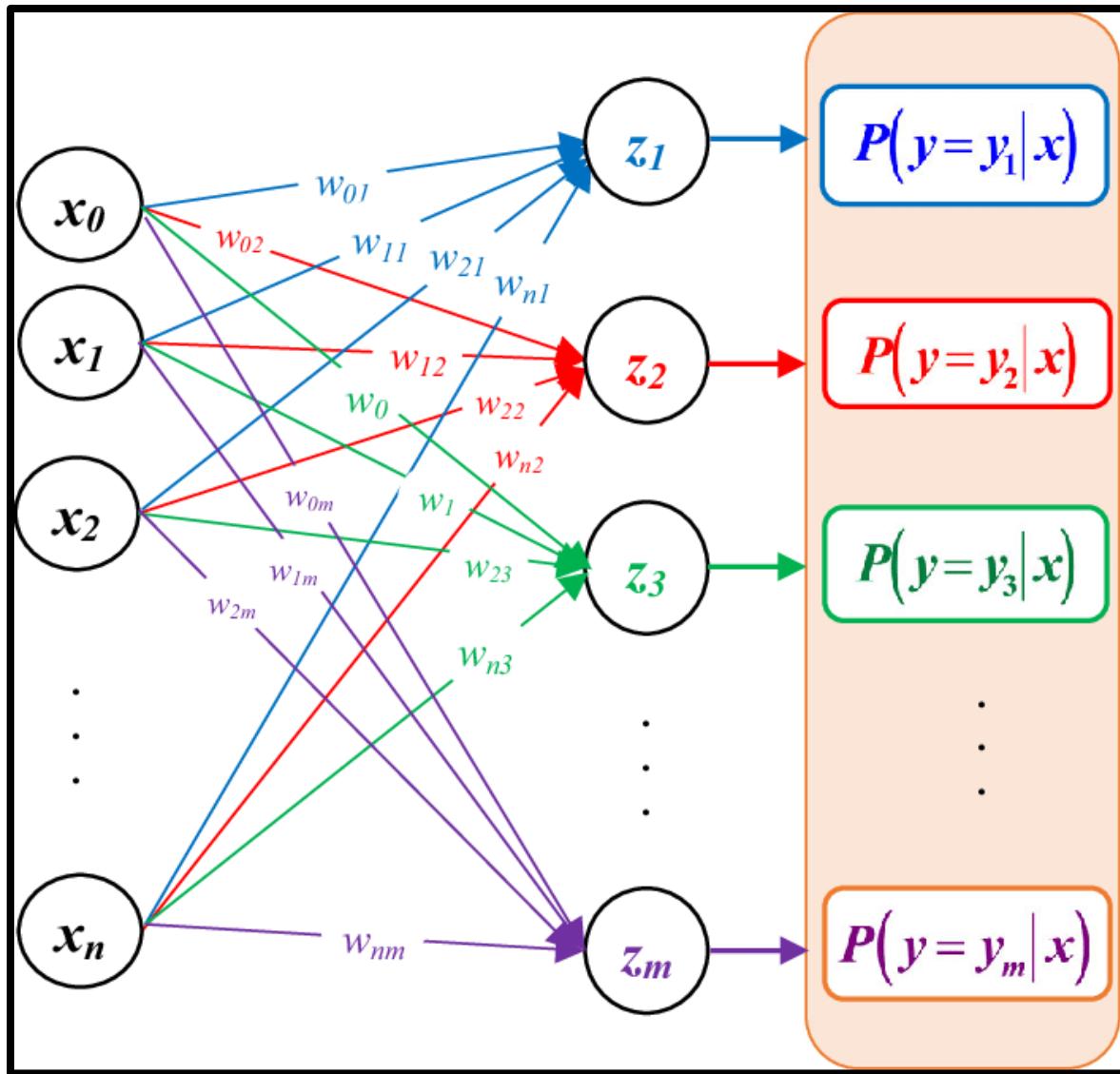
emotion recognition

autonomous vehicles

most real-world applications
tend to be multiclass

4

Architecture

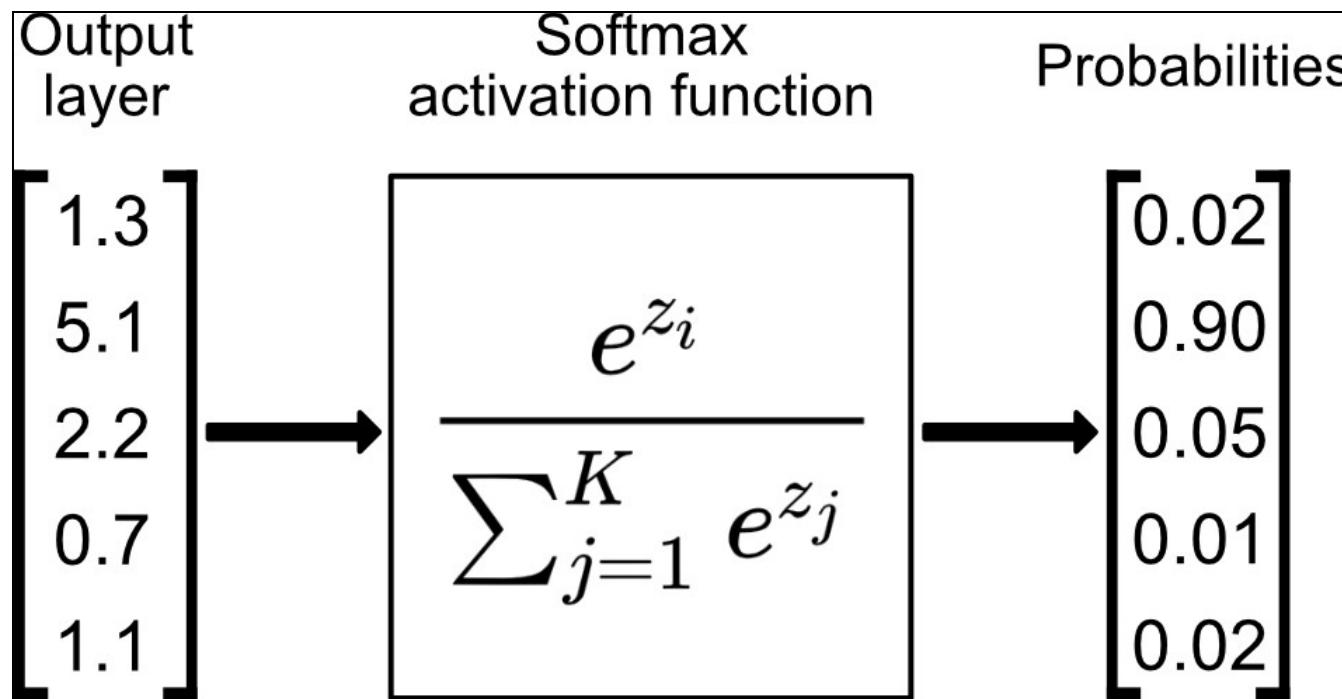


Differences with Binary Classification:

1. Output layer has **m neurons** each having its own logit
2. Output of network has **m probability values**
3. **Can we use Sigmoid?**
4. **Can we use BCELoss?**

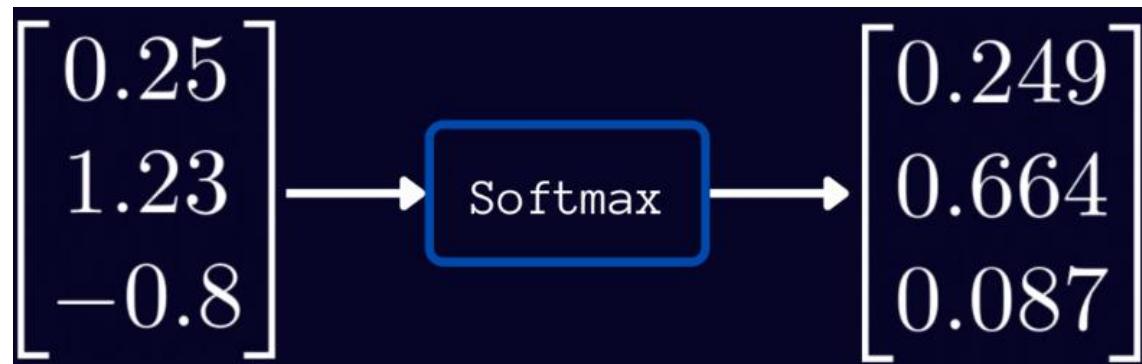
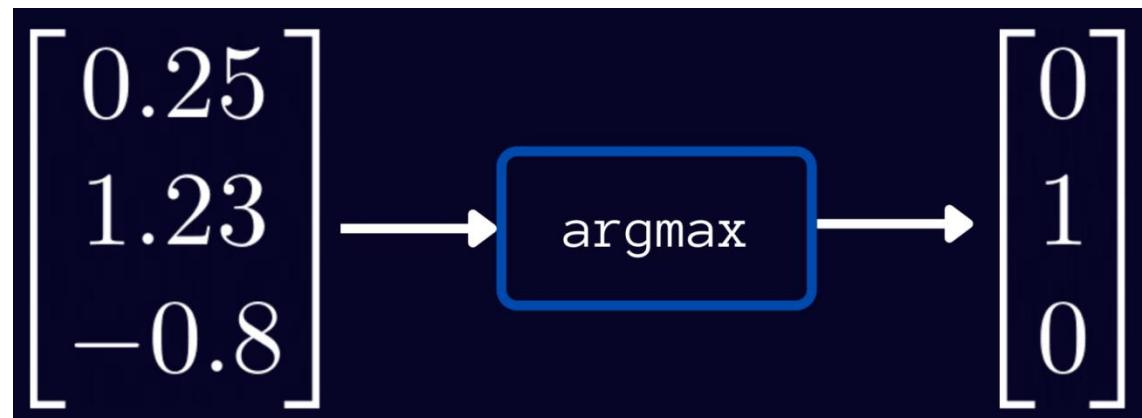
Softmax Activation

- Softmax is a mathematical function that converts a vector of numbers into a **vector of probabilities**

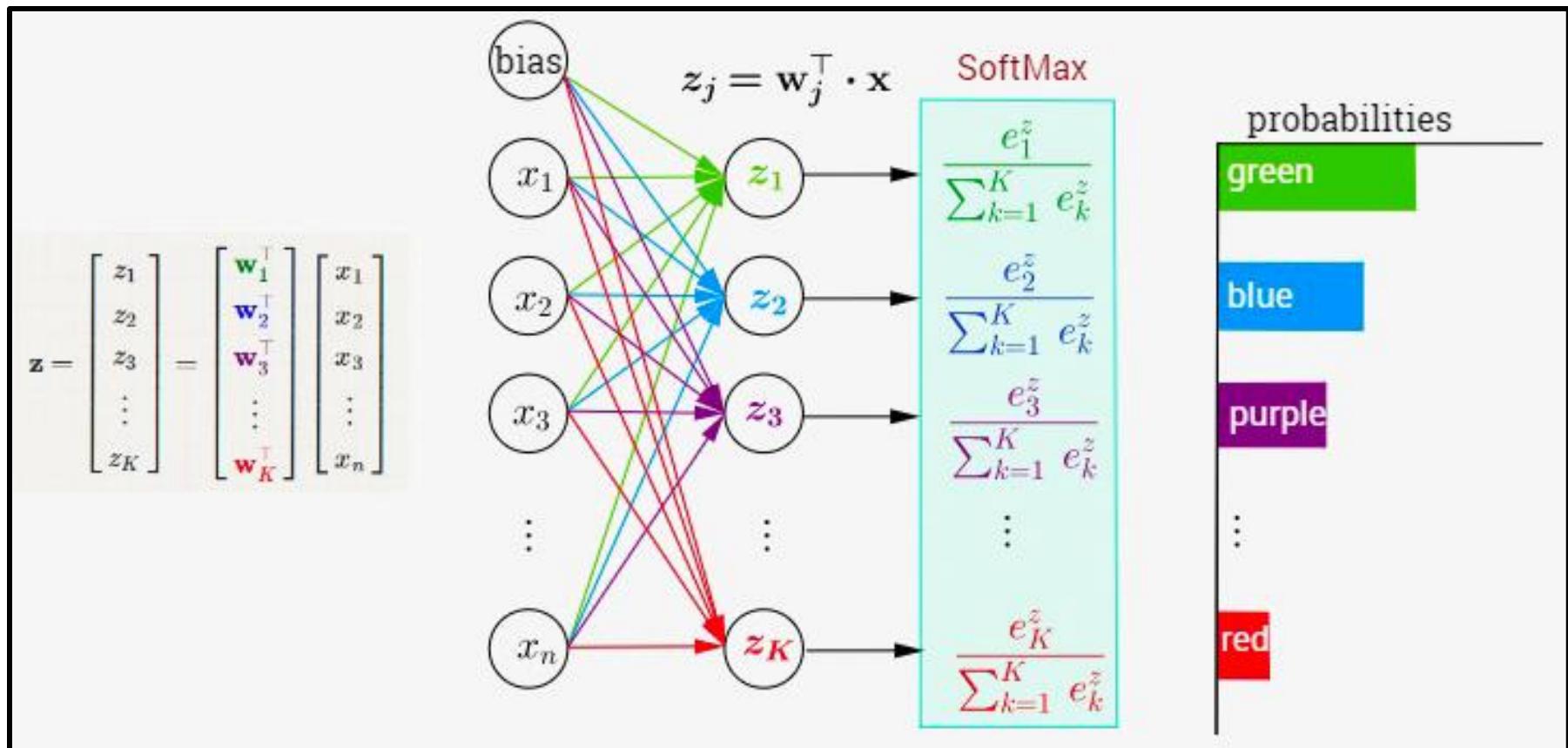


Softmax Vs Argmax

- Why argmax is not a good candidate
 - Recall gradients



Architecture with Softmax



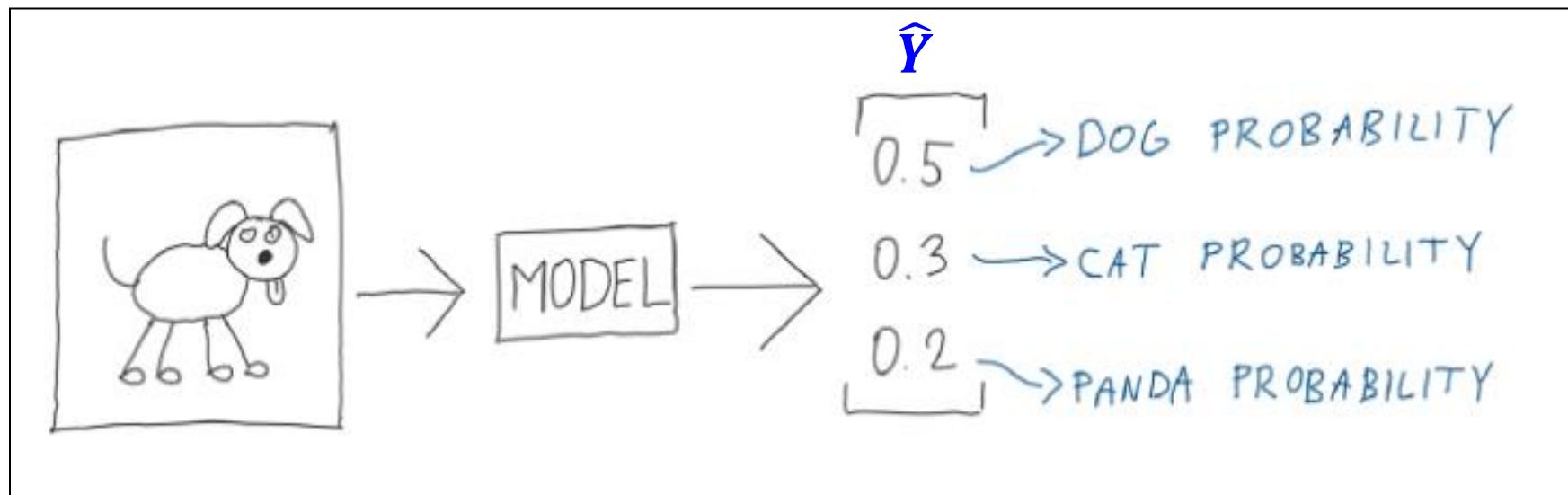
PyTorch Implementation of Softmax

- **torch.nn.Softmax** as a layer
- **torch.nn.functional.softmax** as a function

```
1 import torch
2 import torch.nn.functional as F
3 import torch.nn as nn
4
5 dummy_logits_for_batch = torch.tensor([[21,-5,0.5],[7,8,-10]]) #shape (2,3)
6
7 #Softmax as a layer
8 softmaxLayer = nn.Softmax(dim=-1)
9 probabilities_yhat = softmaxLayer(dummy_logits_for_batch) #shape (2,3)
10 print(probabilities_yhat)
11 test_sum_each_row = torch.sum(probabilities_yhat, dim=1) #shape (2)
12
13 #softmax as a function
14 probabilities_yhat=F.softmax(dummy_logits_for_batch, dim=-1) #shape (2,3)
15 test_sum_each_row = torch.sum(probabilities_yhat, dim=1) #shape (2)
16 print(probabilities_yhat)
```

Loss Function for Multiclass Classification

- How does the target (Y) and predicted (\hat{Y}) looks now?

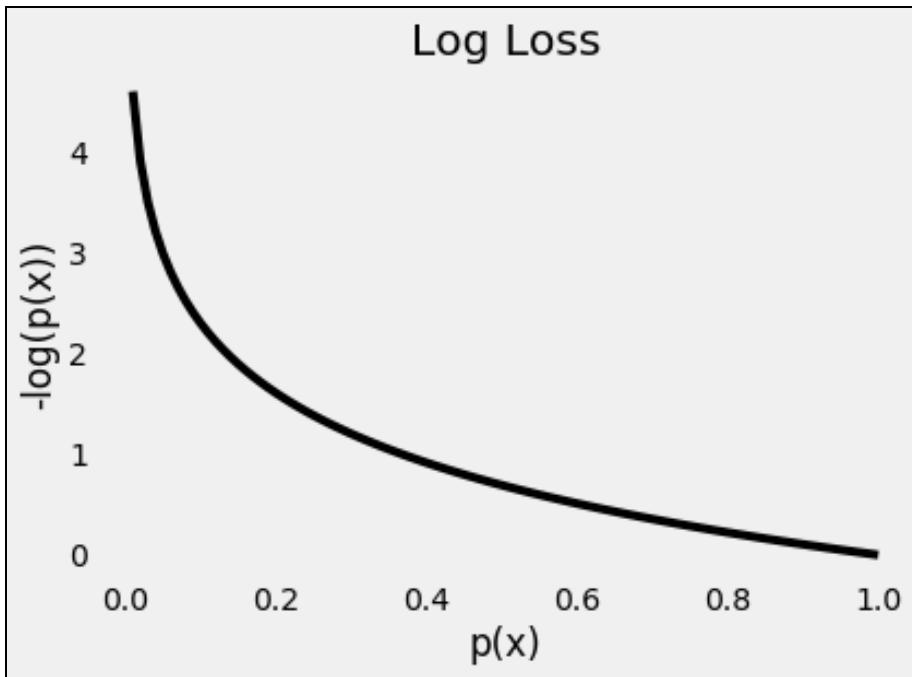


Loss Function for Multiclass Classification

- How does the target (Y) and predicted (\hat{Y}) looks now?

y^i	\hat{y}^i
TARGET	PREDICTION
1	0.5
0	0.3
0	0.2

Remember $-\log(\hat{Y})$



TARGET	PREDICTION
1	0.5
0	0.3
0	0.2

Loss for class X = $-Y \log(\hat{Y})$

Loss for class Dog = $-1 * \log(0.5) = 0.69$

Loss for class Cat = ?

Loss for class Panda = ?

Cross-Entropy (CE) or negative log-likelihood (NLL) Loss

Three Classes

$$NLLLoss(y) = -\frac{1}{(N_0 + N_1 + N_2)} \left[\sum_{i=1}^{N_0} \log(P(y_i = 0)) + \sum_{i=1}^{N_1} \log(P(y_i = 1)) + \sum_{i=1}^{N_2} \log(P(y_i = 2)) \right]$$

$$\text{cross-entropy} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^k t_{i,j} \log(p_{i,j})$$

Note:

$t_{i,j}$ is the target value (Y) from the one hot vector of ith training example and jth class
 $p_{i,j}$ is the predicted value (\hat{Y}) from the logit vector of ith training example and jth class

Cross-Entropy Loss in PyTorch

- Option-1: **nn.CrossEntropyLoss**
- Option-2: **nn.LogSoftmax + nn.NLLLoss**

```
import torch
import torch.nn.functional as F
import torch.nn as nn

torch.manual_seed(11)
dummy_batch_of_logits_Z = torch.randn((5, 3))
dummy_labels_Y = torch.tensor([0, 0, 1, 2, 1])

#option-1 logits to Loss (no need for softmax layer)
loss_fn = nn.CrossEntropyLoss()
loss=loss_fn(dummy_batch_of_logits_Z, dummy_labels_Y)
print(loss)

#option-2 log(Yhat) to Loss (last layer of network as LogSoftMax)
logSoftmaxLayer=nn.LogSoftmax(dim=-1)
log_batch_Yhat=logSoftmaxLayer(dummy_batch_of_logits_Z)
loss_fn = nn.NLLLoss()
loss=loss_fn(log_batch_Yhat, dummy_labels_Y)
print(loss)
```

Summary (Activation + Loss)

	Activation Function	Loss Function
Binary Classification	Sigmoid	BCELoss
	No activation (logits only)	BCEWithLogitsLoss
Multiclass Classification	LogSoftmax	NLLLoss
	No activation (logits only)	CrossEntropyLoss

Be careful and choose right activation for a selected Loss function

Graded Home Task 3

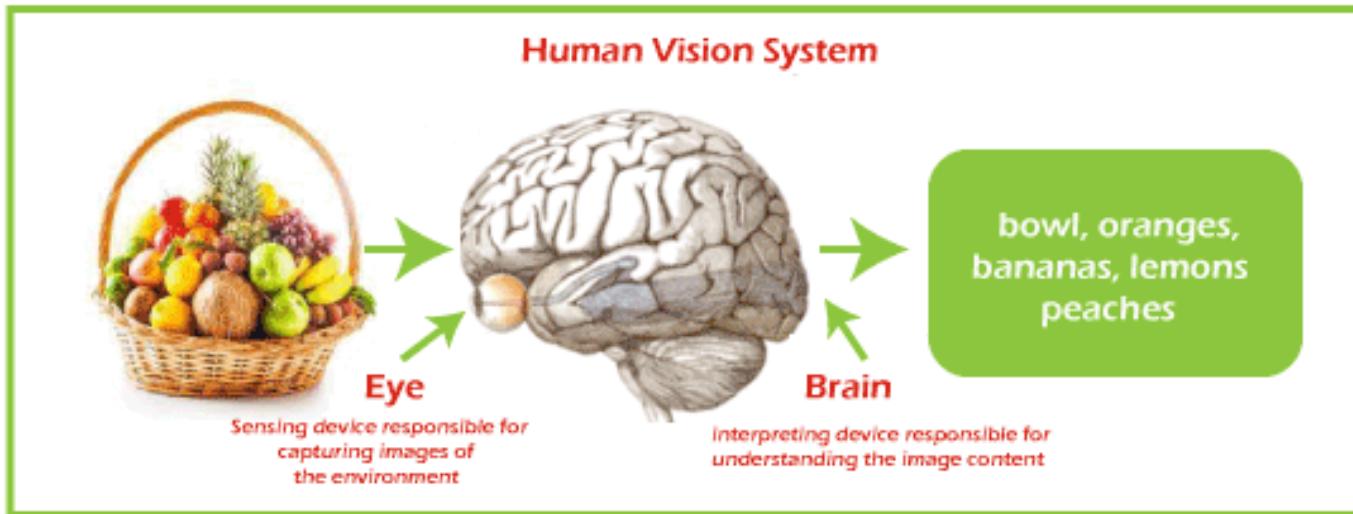
- Create a multiclass classification model for sklearn's digits dataset
- DataSet (X,y) with size (1797,64)
- Each 64 features are pixels of an 8×8 digit image
- Do training with different optimizers (SGD+ADAM) and show their comparison (Loss vs Iteration)
- Compute Confusion matrix for all 10 classes
- Check in your code and results

COMPUTER VISION

Computer Vision

- Computer vision is an area of artificial intelligence (AI) that enables systems to derive meaningful information from **images, videos and other visual inputs**
 - Take actions or make recommendations based on that information
- Computer vision tries to emulate human vision using computers and the power of AI

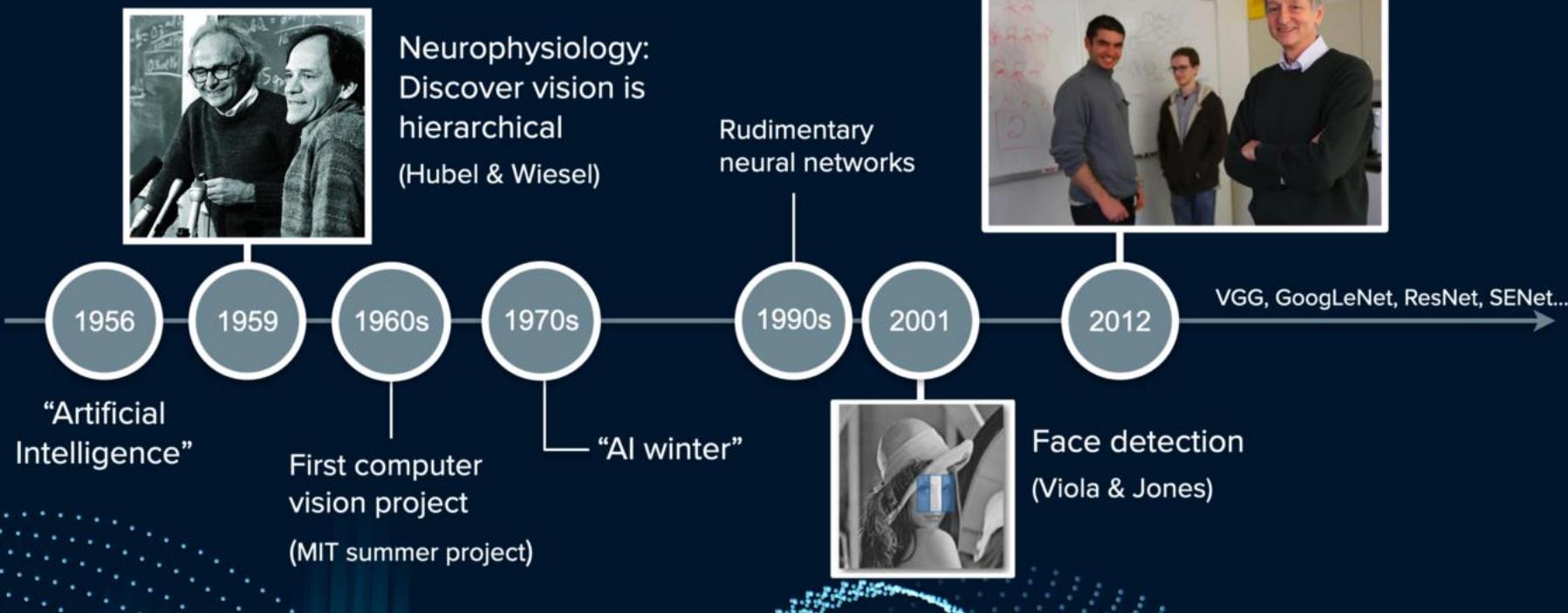
Computer Vision



Deep Learning for Computer Vision

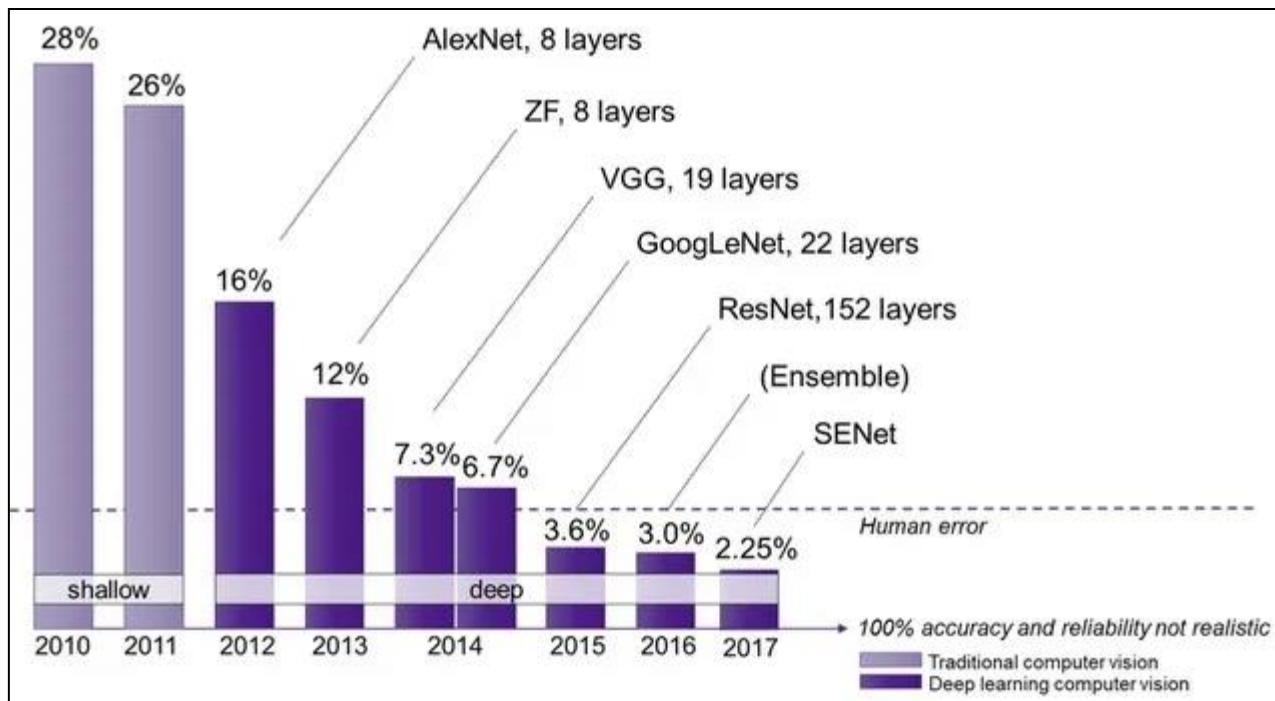
HISTORY

A brief history

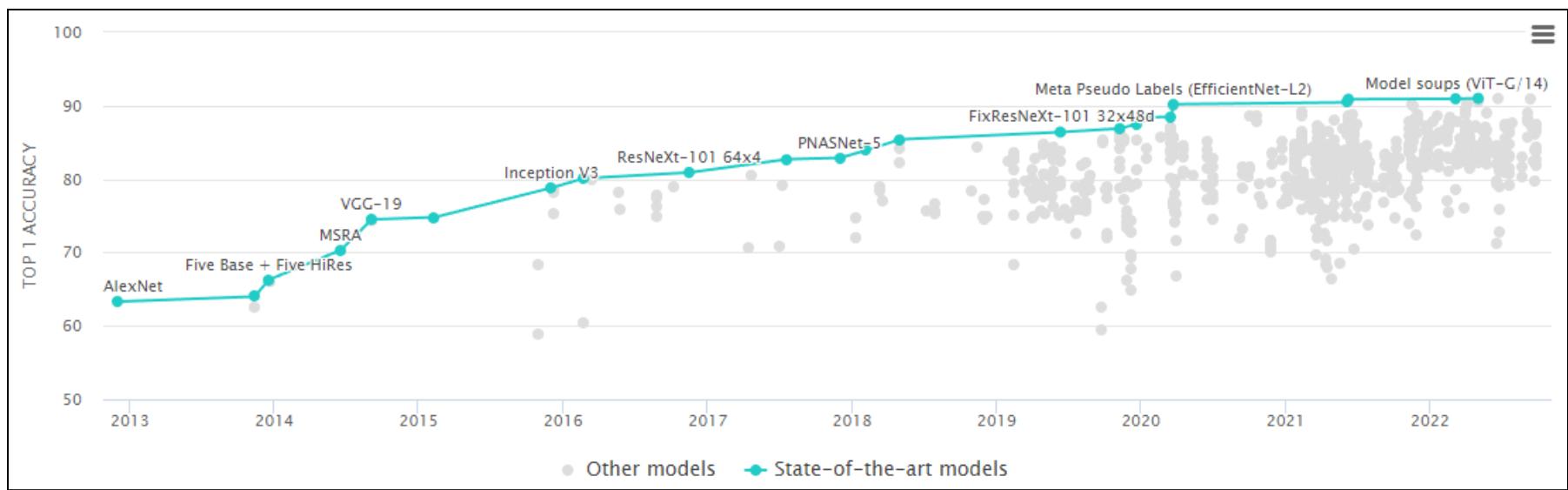


ImageNet Challenge (ILSVRC)

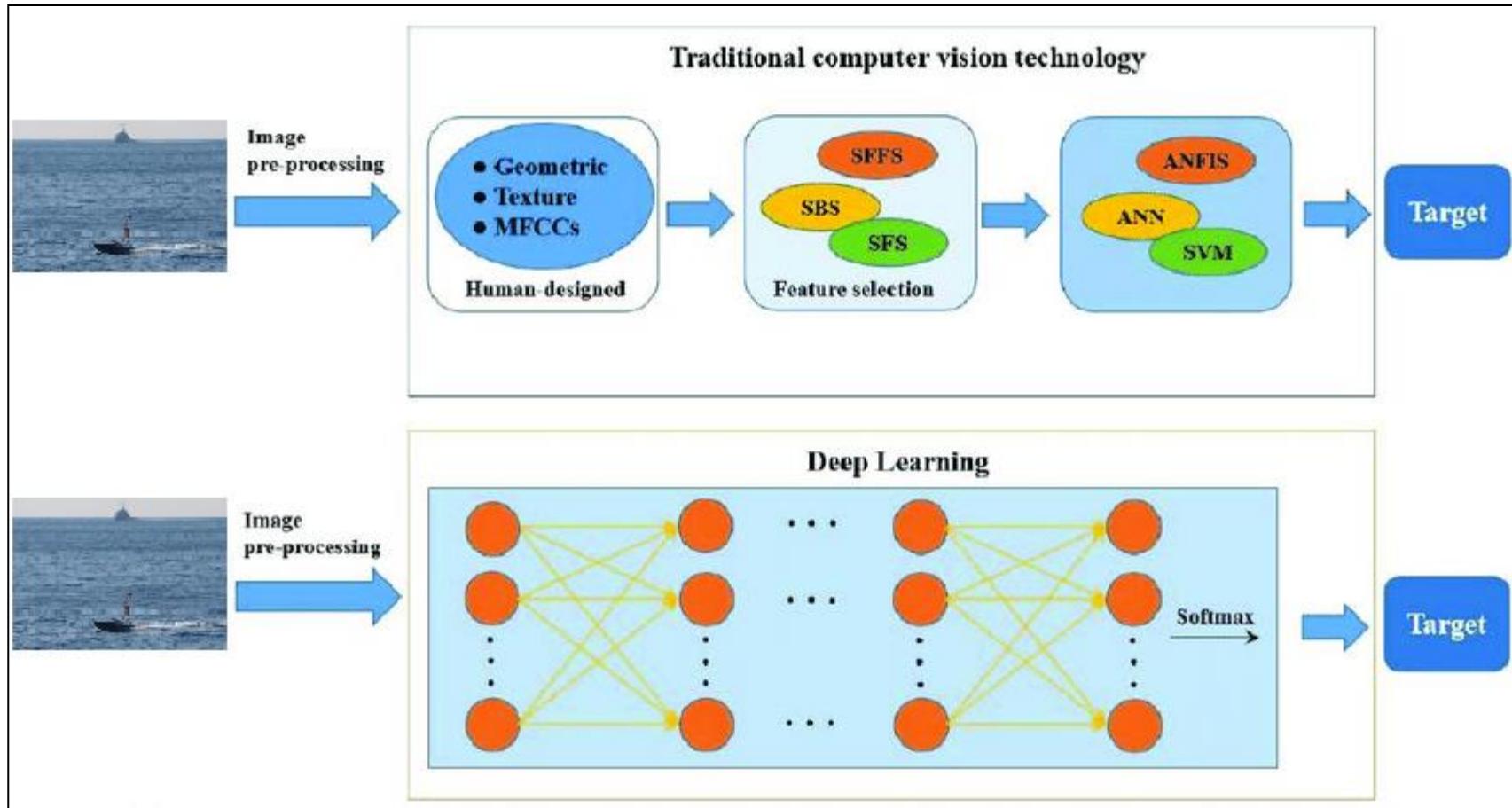
- ImageNet Large Scale Visual Recognition Challenge is an annual computer vision competition developed upon a subset of a publicly available computer vision dataset called **ImageNet**
- 14 million images have been hand-annotated
- **Image classification, Object detection**



ImageNet Challenge (ILSVRC)



Traditional vs Deep Vision



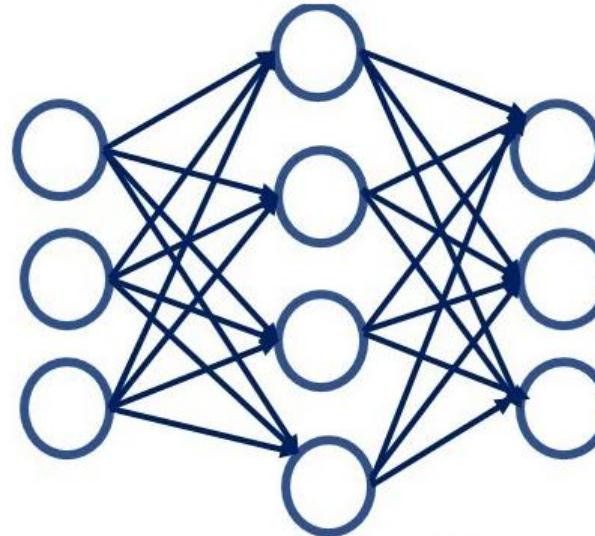
COMPUTER VISION APPLICATIONS

Image Classification

Dog



Fish



Output

Dog

Deep Learning architecture

MobileNet

AlexNet

GoogleNet

Image Classification

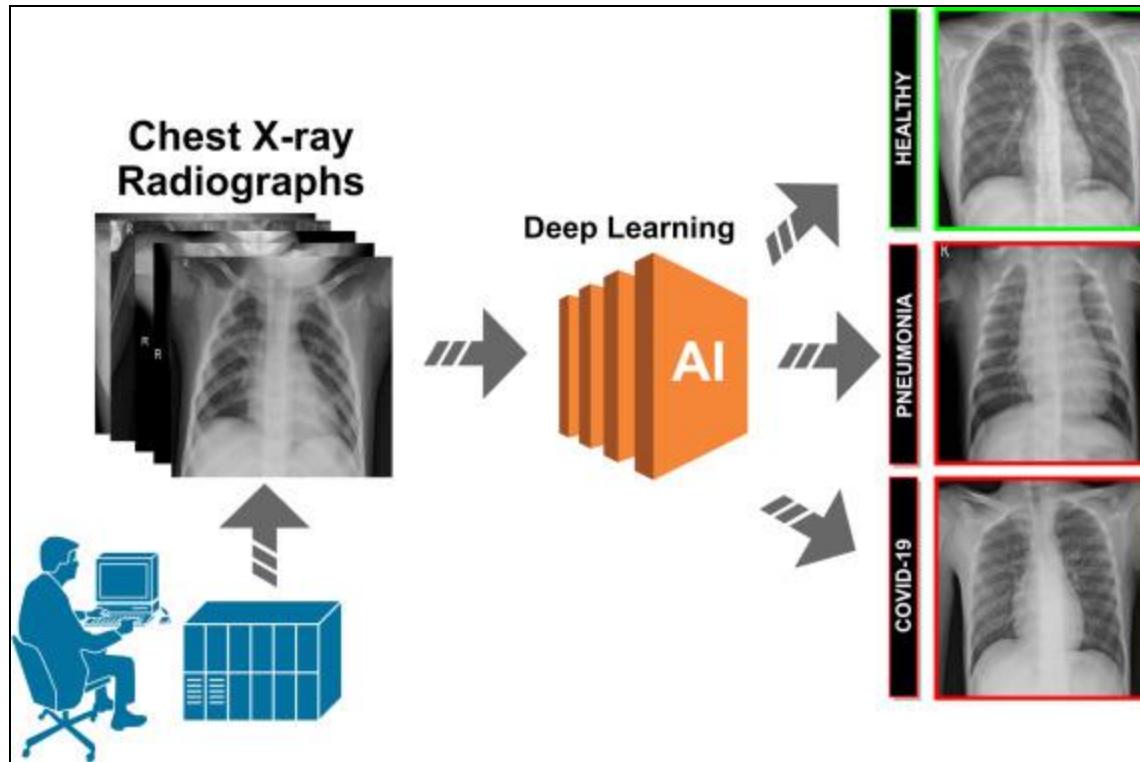
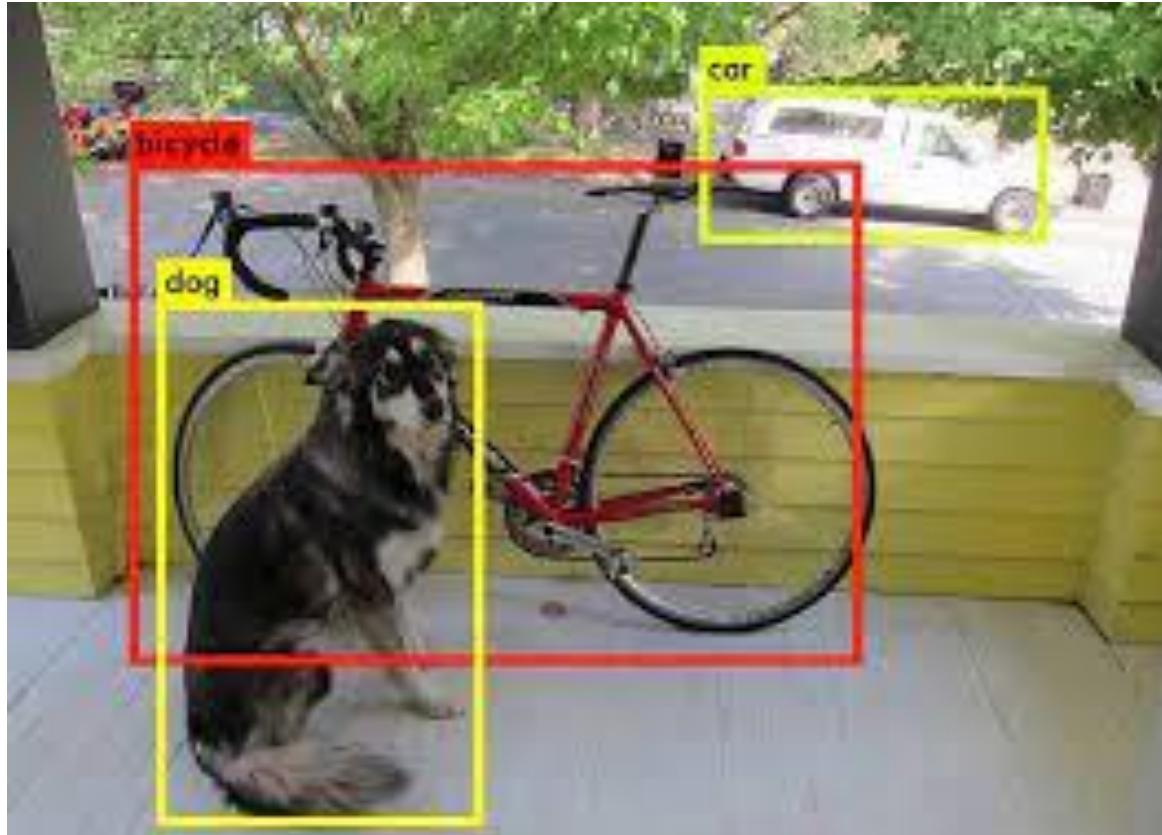


Image Classification Models

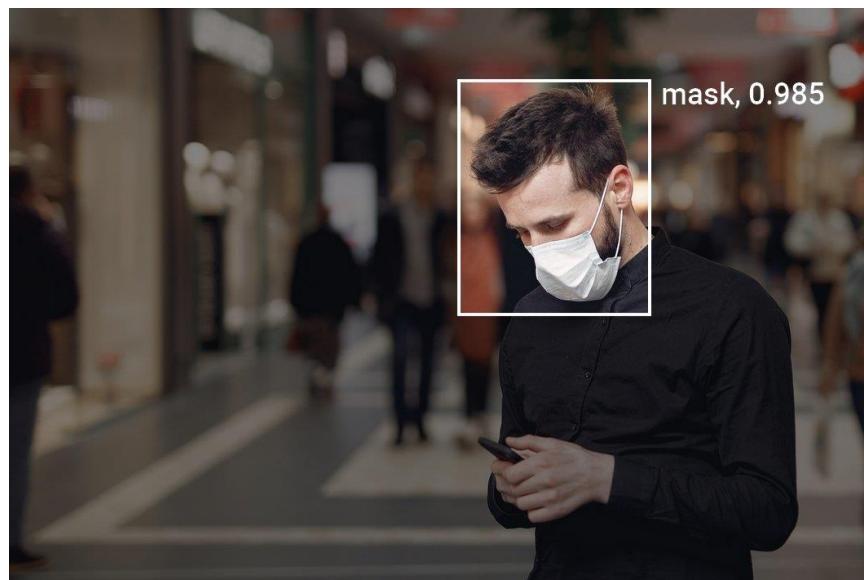
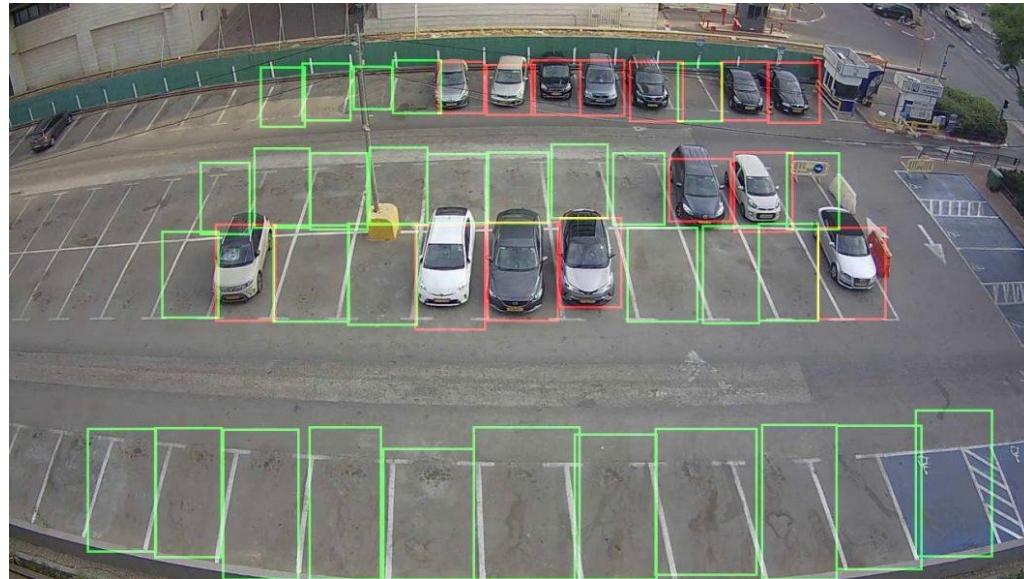
- AlexNet
- ConvNeXt
- DenseNet
- EfficientNet
- EfficientNetV2
- GoogLeNet
- Inception V3
- MNASNet
- MobileNet V2
- MobileNet V3

- RegNet
- ResNet
- ResNeXt
- ShuffleNet V2
- SqueezeNet
- SwinTransformer
- VGG
- VisionTransformer
- Wide ResNet

Object Detection



Object Detection



Object Detection Models

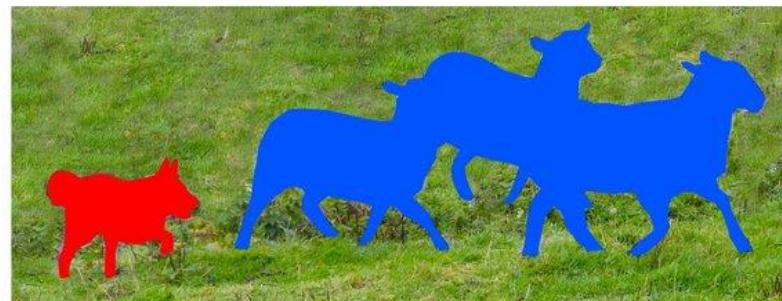
- Faster R-CNN
- FCOS
- RetinaNet
- SSD
- SSDlite



Semantic Segmentation

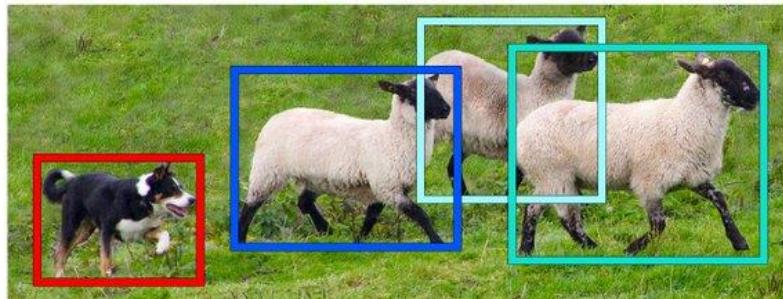


Image Recognition

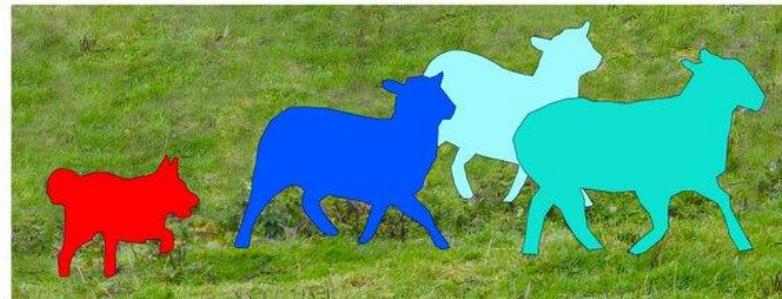


Semantic Segmentation

- DeepLabV3
- FCN
- LRASPP



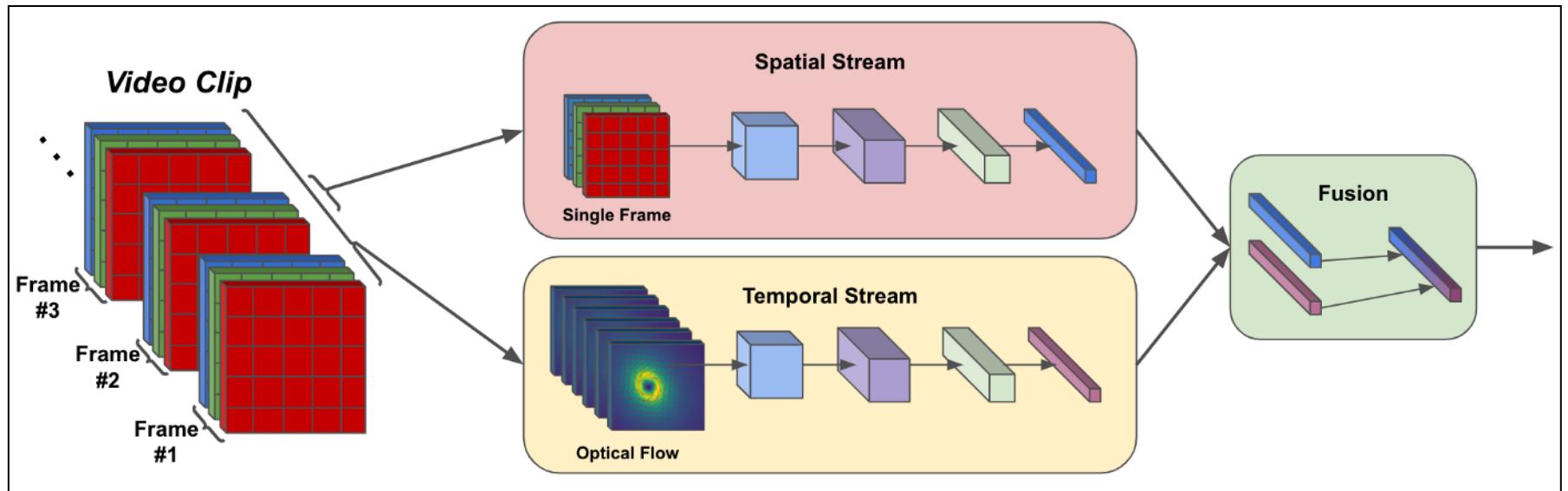
Object Detection



Instance Segmentation

- Mask R-CNN

Video Classification



No errors



A white teddy bear sitting in the grass



A man riding a wave on top of a surfboard

Minor errors



A man in a baseball uniform throwing a ball



A cat sitting on a suitcase on the floor

Somewhat related



A woman is holding a cat in her hand



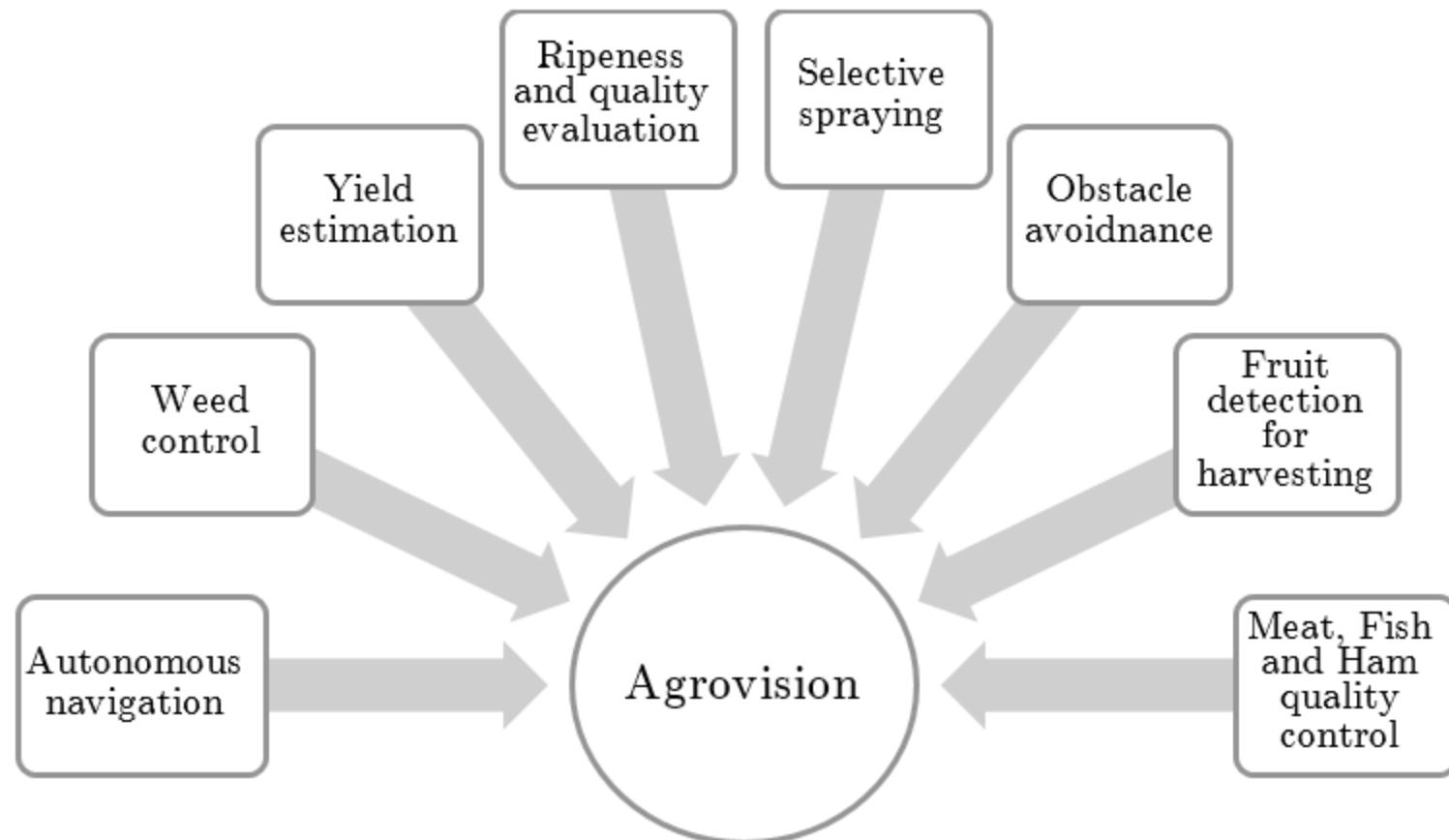
A woman standing on a beach holding a surfboard

Image Captioning

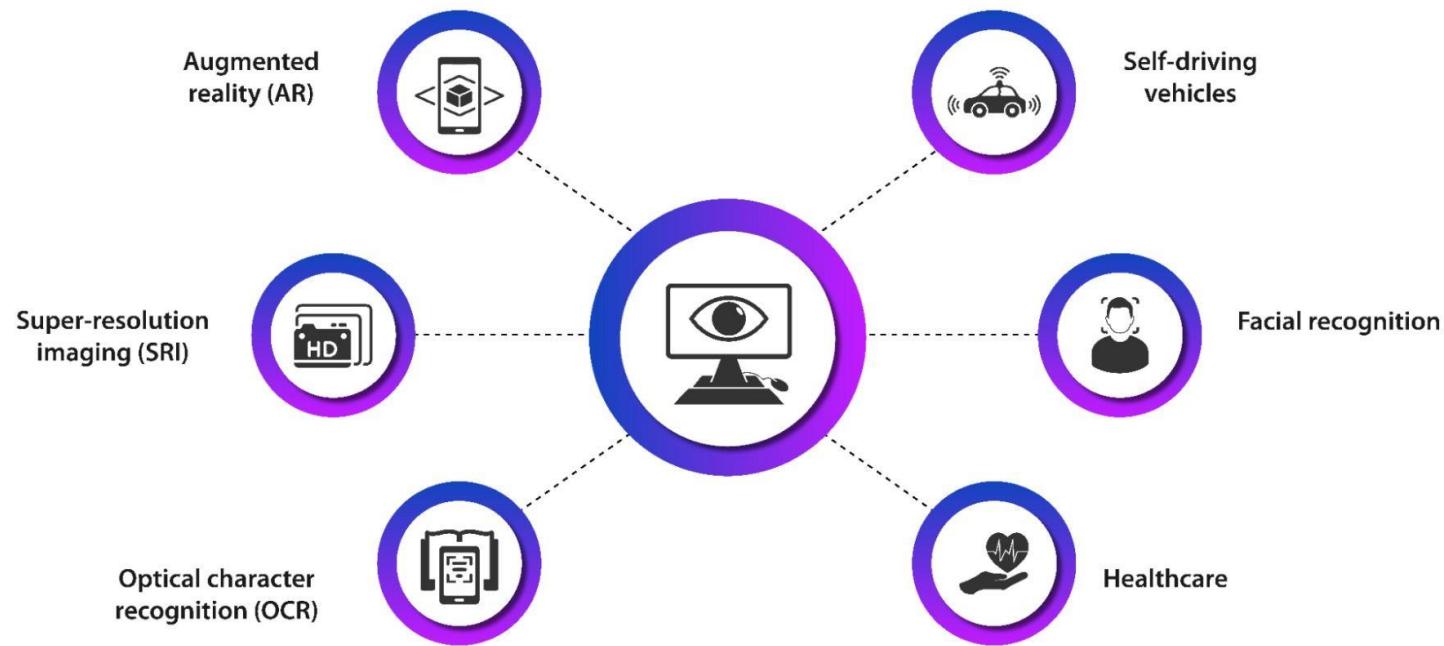
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

All images are CC0 Public domain:
<https://pixabay.com/en/lugasse-antique-cat-1643010/>
<https://pixabay.com/en/teddy-blush-bear-cute-teddy-bear-1623436/>
<https://pixabay.com/en/blue-wave-summer-snoot-literal-1668716/>
<https://pixabay.com/en/woman-female-model-portrait-adult-983967/>
<https://pixabay.com/en/hands-on-head-like-meditation-496008/>
<https://pixabay.com/en/baseball-player-spring-training-field-1045263/>

Captions generated by Justin Johnson using [Neuraltalk2](#)



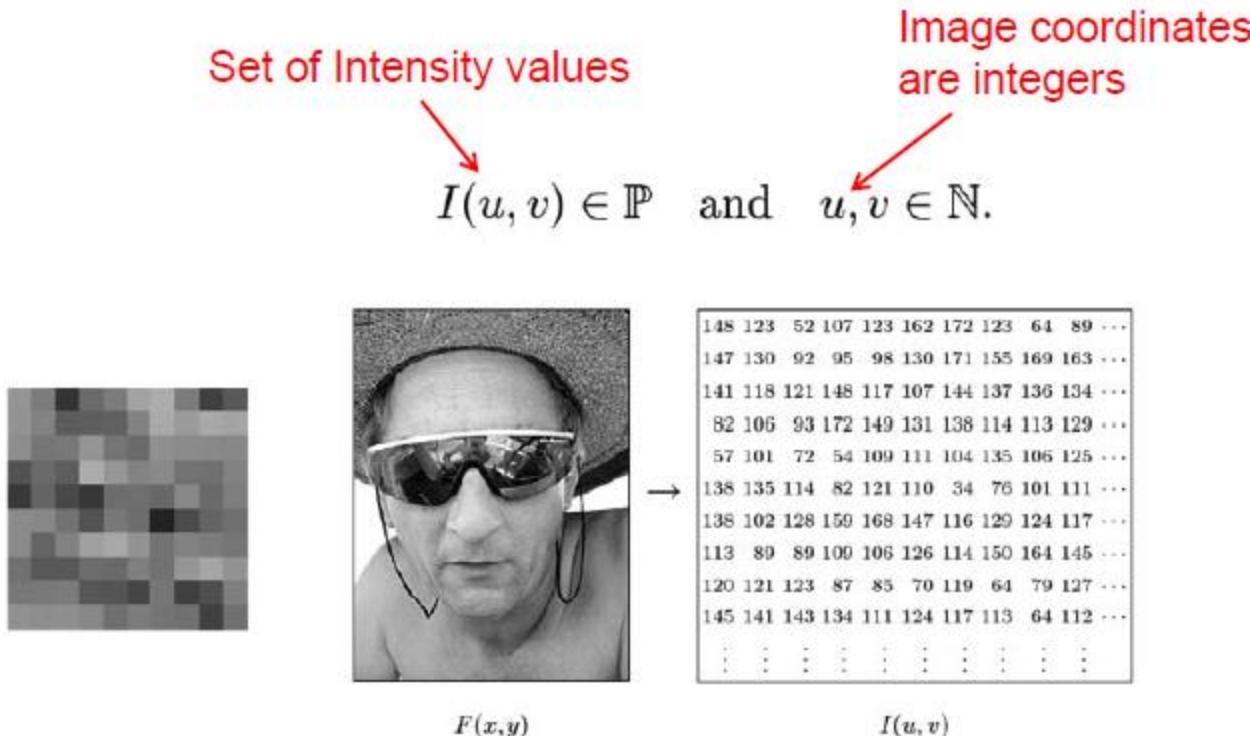
Computer Vision Applications



BASICS OF IMAGE PROCESSING

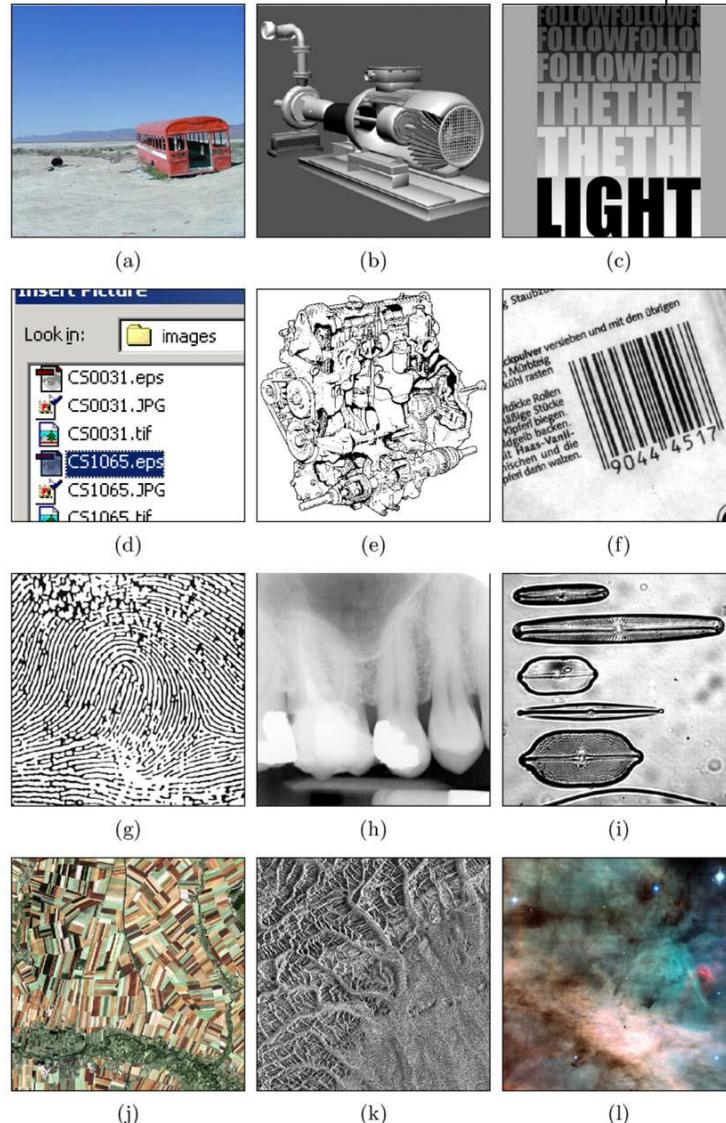
What is an Image?

- 2-dimensional matrix of Intensity (gray or color) values



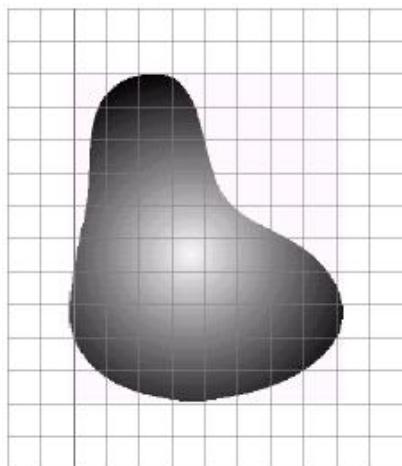
Examples of Digital Images

1. Natural landscape
2. Synthetically generated scene
3. Poster graphic
4. Computer screenshot
5. Black and white illustration
6. Barcode
7. Fingerprint
8. X-ray
9. Microscope slide
10. Satellite Image
11. Radar image
12. Astronomical object

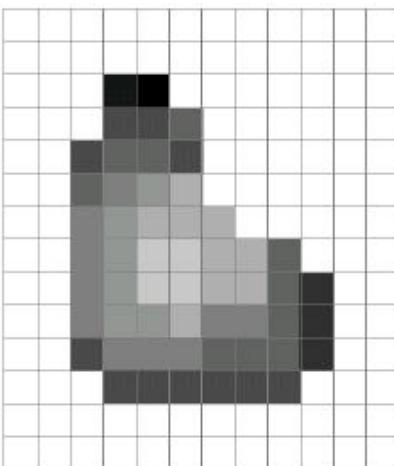


Digital Image?

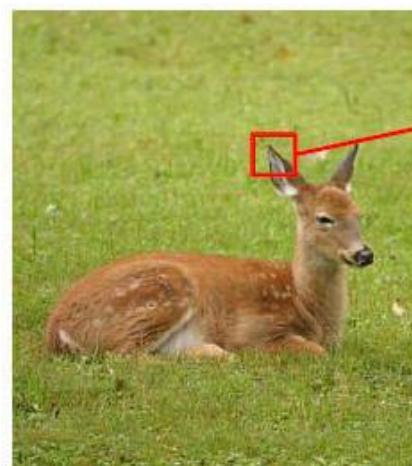
- Digitization causes a digital image to become an approximation of a real scene
- Spatial coordinates (u, v) and amplitude function (I) is **finite/discrete/integer**
- The elements of a image are usually referred to as **pixels**



Real image



Digital Image
(an approximation)

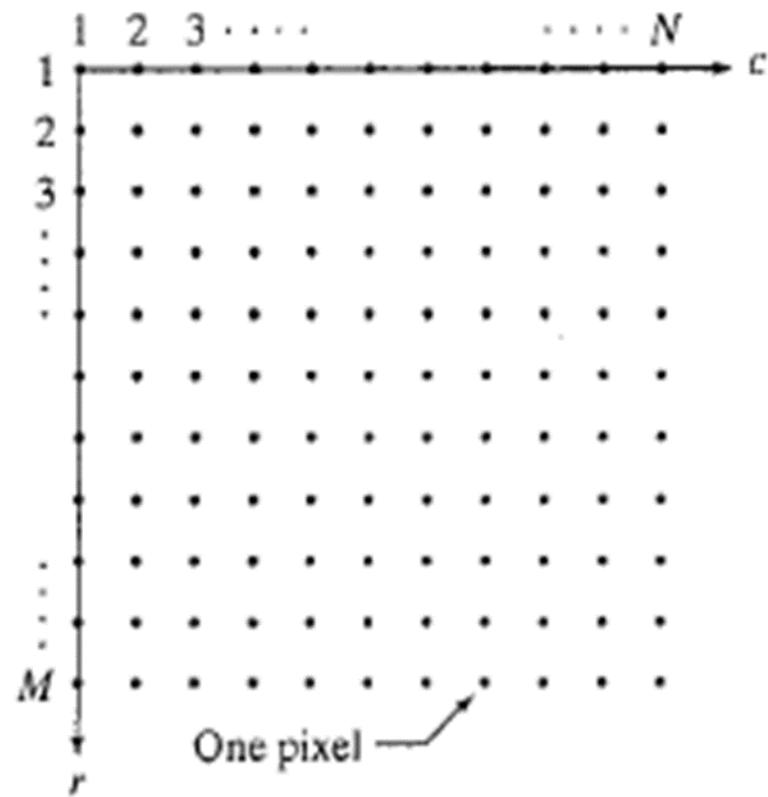
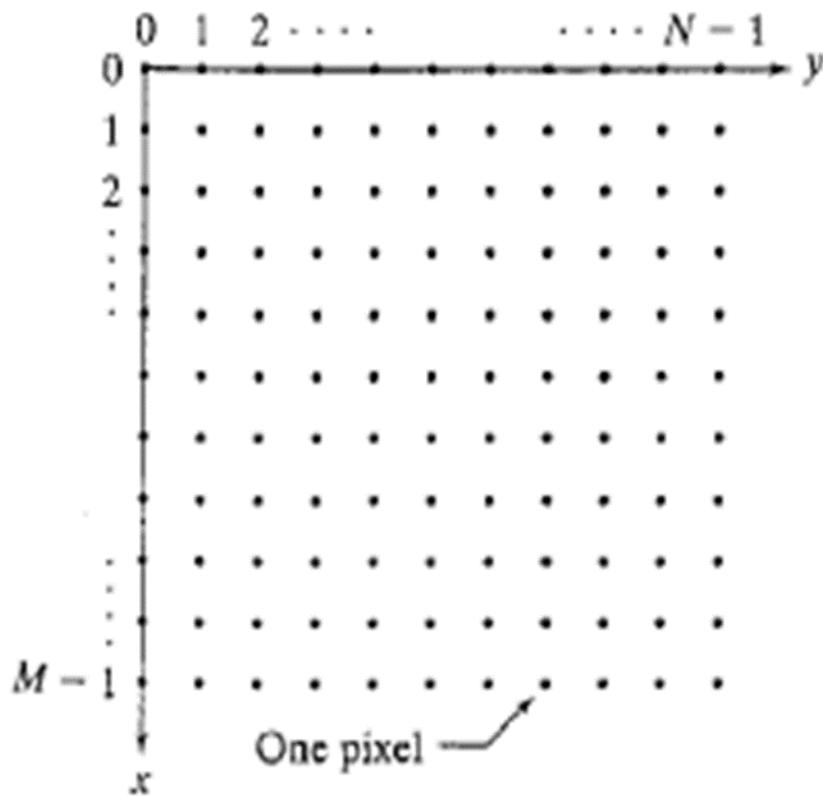


Real image

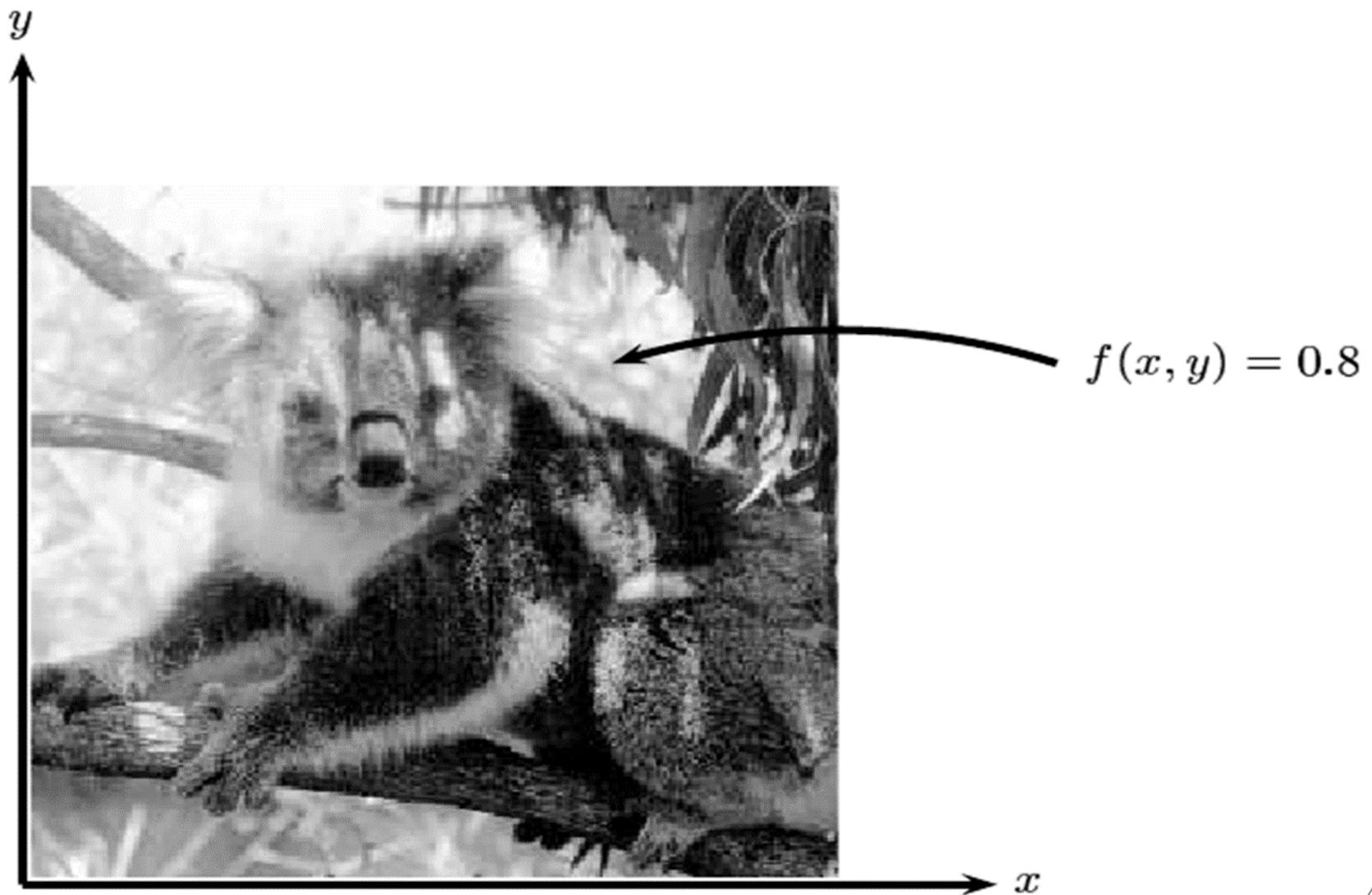


Digital Image
(an approximation) 40

An Image as 2D Function



An Image as 2D Function



An Image as 2D Function

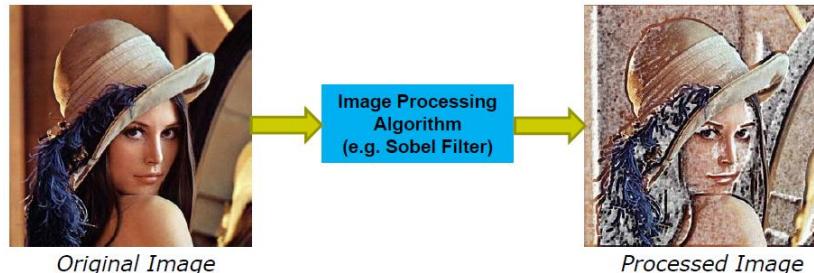
48	219	168	145	244	188	120	58
49	218	87	94	133	35	17	148
174	151	74	179	224	3	252	194
77	127	87	139	44	228	149	135
138	229	136	113	250	51	108	163
38	210	185	177	69	76	131	53
178	164	79	158	64	169	85	97
96	209	214	203	223	73	110	200

Current pixel

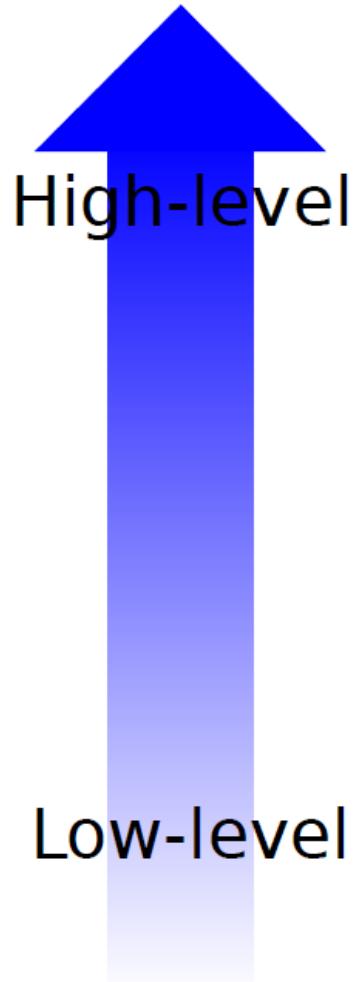
3×5 neighbourhood

What is Image Processing

- Algorithms that alter an input image to create new image
 - Input is image, output is image
- Improves an image for human interpretation
 - Image enhancement
 - Make result more suitable for a particular application.
 - Sharpening an out of focus image, highlighting edges, improving image contrast, or brightening an image, removing noise
 - Image restoration
 - Reversing the damage done to an image by a known cause.
 - Removing of blur caused by linear motion, removal of optical distortions
 - Image segmentation
 - Dividing an image into constituent parts, or isolating certain aspects of an image.
 - finding lines, circles, or particular shapes in an image, in an aerial photograph, identifying cars, trees, buildings, or roads.
- More examples, history and applications in lec-1



Relationship with other Fields



Computer Vision

Object detection, recognition, shape analysis, tracking
Use of Artificial Intelligence and Machine Learning

Image Analysis

Segmentation, image registration, matching

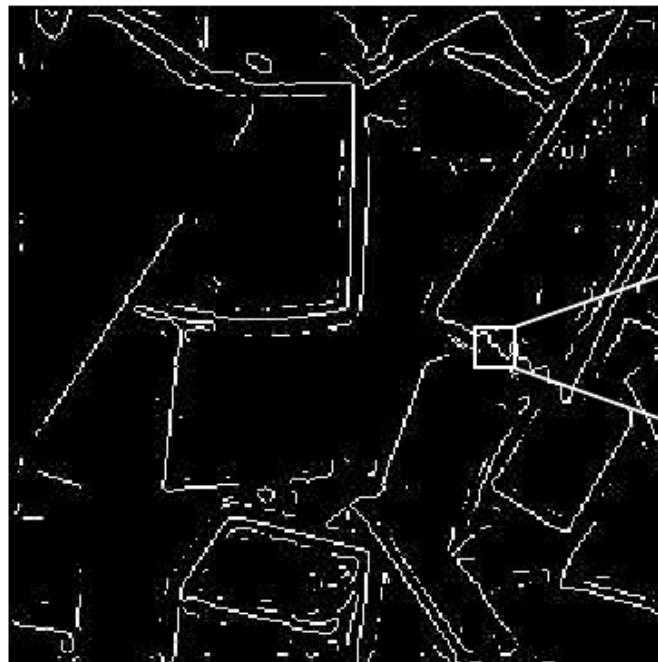
Image Processing

Image enhancement, noise removal, restoration,
feature detection, compression

Types of image

- **Binary**: Each pixel is black or white, so only one bit is needed per pixel.
- **Gray-Scale**: Each pixel is a shade of gray normally from black (0) to white (255) so 8 bits per pixel.
- **Color**: Each pixel is a color described by the mix of red, green or blue (RGB). Each channel has a range of 0-255, so 24bits in total.
- **Indexed**: Most color images only use a subset of the possible colors, for reduction of storage a color map is used.

Binary Image



1	1	0	0	0	0
0	0	1	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	0	0	0	1

Gray Scale Image



White = 255 ← Black = 0

RGB Image



49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

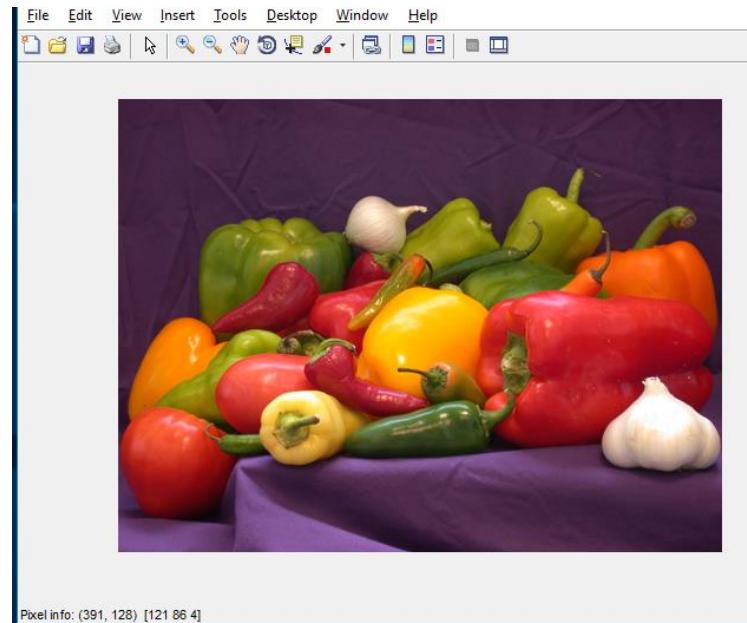
Red

64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Green

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Blue



Indexed Image

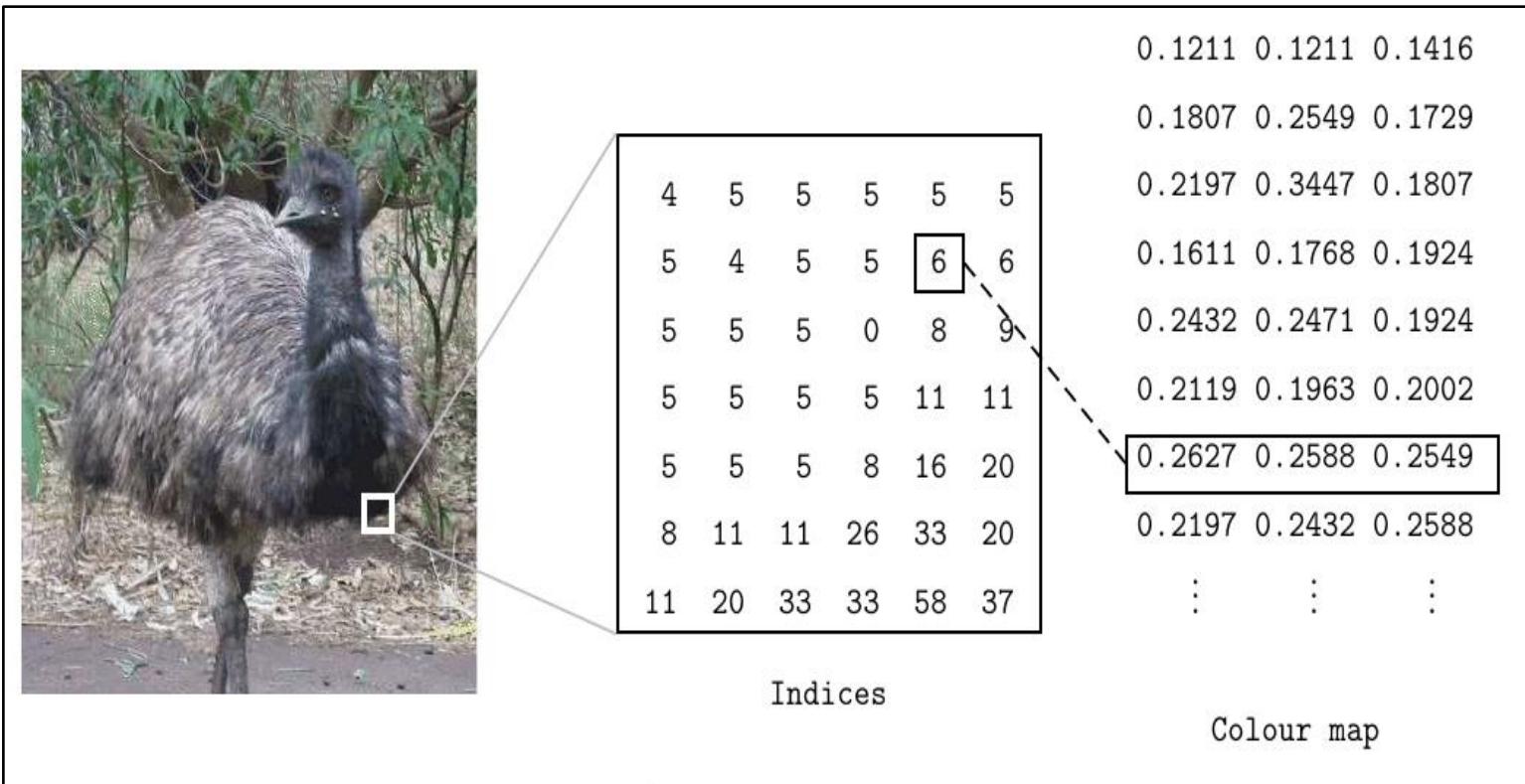


Image Size

- Image files tend to be large
 - e.g. 512 x 512 gray scale image requires 256 Kbytes = $512 \times 512 \times 8\text{bits}$.
 - For an RGB color image this will be 768 Kbytes= $256 \times 3\text{Kbytes} = 512 \times 512 \times 24\text{bits}$

Pillow Library

- Popular image processing libraries
 - OpenCV
 - scikit-image
 - Python Imaging Library
 - Pillow
- Pillow, a powerful library that provides a wide array of image processing features and is simple to use
 - pip install pillow
 - <https://pillow.readthedocs.io/en/stable/index.html>

Pillow Example

```
1  from PIL import Image, ImageDraw, ImageFont, ImageFilter
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  pillowIm=Image.open('building.jpg')
6
7  #pillow to numpy
8  npIm=np.array(pillowIm)
9  npIm[500:800,500:800,0:3]=0
10
11 #numpy to pil
12 pillowIm2=Image.fromarray(npIm)
13
14 #operations
15 rotated_img = pillowIm.rotate(45)
16 resized_img = pillowIm.resize((300,600))
17
18 #filters
19 img_gray=pillowIm.convert('L')
20 edge_image = img_gray.filter(ImageFilter.FIND_EDGES)
21
```

Pillow Example

```
22 #drwaing
23 pillowImDraw=pillowIm.copy()
24 font = ImageFont.truetype("arial.ttf", 200)
25 img_draw = ImageDraw.Draw(pillowImDraw)
26 img_draw.rectangle((70, 50, 270, 200), outline='red', fill='blue')
27 img_draw.text((70, 250), 'Hello World', fill='red',font=font)
28
29 rows=2
30 cols=3
31 plt.figure()
32 plt.subplot(rows,cols,1)
33 plt.imshow(pillowIm)
34 plt.axis('off')
35 plt.title('Size:' +str(pillowIm.size) + ',mode:' +pillowIm.mode+ ',fmt:' +pillowIm.format)
36
37 plt.subplot(rows,cols,2)
38 plt.imshow(npIm)
39 plt.axis('off')
40 plt.title('Shape:' +str(npIm.shape))
```

Pillow Example

```
plt.subplot(rows,cols,3)
plt.imshow(rotated_img)
plt.axis('off')
plt.title('Size:' + str(rotated_img.size) + ', mode:' + rotated_img.mode)

plt.subplot(rows,cols,4)
plt.imshow(resized_img)
plt.title('Size:' + str(resized_img.size) + ', mode:' + resized_img.mode)

plt.subplot(rows,cols,5)
plt.imshow(edge_image)
plt.gray()
plt.title('Size:' + str(edge_image.size) + ', mode:' + edge_image.mode)

plt.subplot(rows,cols,6)
plt.imshow(pillowImDraw)
plt.axis('off')
plt.title('Size:' + str(pillowImDraw.size) + ', mode:' + pillowImDraw.mode)

plt.show()
```

Pillow Example

Size:(1920, 1273),mode:RGB,fmt:JPEG



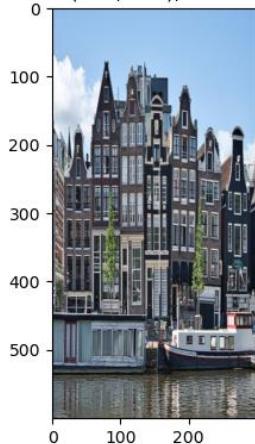
Shape:(1273, 1920, 3)



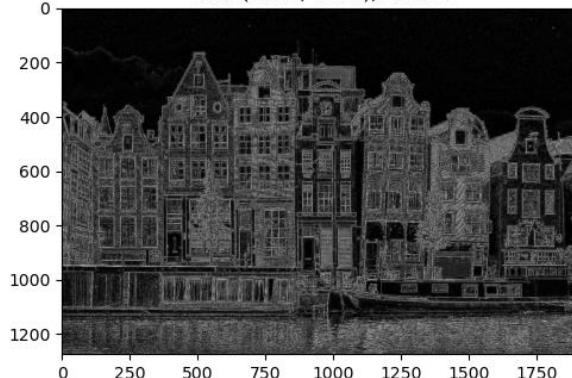
Size:(1920, 1273),mode:RGB



Size:(300, 600),mode:RGB



Size:(1920, 1273),mode:L



Size:(1920, 1273),mode:RGB



OPENCV Video Reading

```
1 import numpy as np
2 import cv2
3 cap = cv2.VideoCapture('intro.mp4')
4 while(cap.isOpened()):
5
6     ret, frame = cap.read()
7     #cv2.namedWindow("window", cv2.WND_PROP_FULLSCREEN)
8     #cv2.setWindowProperty("window",cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)
9
10    if ret:
11        cv2.imshow("Image", frame)
12    else:
13        print('no video')
14        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
15
16    if cv2.waitKey(1) & 0xFF == ord('q'):
17        break
18
19
20 cap.release()
21 cv2.destroyAllWindows()
```

NCHW vs NHWC

- **NCHW**
 - **N** stands for the **N**umber of images (in a mini-batch, for instance)
 - **C** stands for the number of **Ch**annels (or **f**ilters) in each image
 - **H** stands for each image's **He**ight
 - **W** stands for each image's **Wi**th
- acronyms indicate the expected **shape of the mini-batch**
- **NCHW**: (number of images, channels, height, width)
- **NHWC**: (number of images, height, width, channel)

NCHW vs NHWC

- *PyTorch* uses NCHW
- *TensorFlow* uses NHWC
- *PIL* images are HWC
- `numpy.transpose` for swapping axes

```
>>> x = np.ones((1, 2, 3))
>>> np.transpose(x, (1, 0, 2)).shape
(2, 1, 3)
```

```
>>> x = np.ones((2, 3, 4, 5))
>>> np.transpose(x).shape
(5, 4, 3, 2)
```

TORCHVISION

Introduction

- Package Containing Computer Vision Popular
 - Datasets
 - Like MNIST, ImageNet, CIFAR
 - Inherit from the original Dataset class
 - Can be naturally used with a DataLoader
 - Model architectures
 - Including its pretrained weights
 - AlexNet, VGG, ResNet (ResNet18, ResNet34, ResNet50, ResNet101, ResNet152), and Inception V3
 - Image transformations
 - Transformations based on images (either in PIL or PyTorch shapes)
 - Transformations based on Tensors
 - Conversion transforms to convert from tensor **ToPILImage** and from PIL image **ToTensor**

Image Transformations

- ToTensor()
 - Convert a PIL Image or numpy.ndarray to tensor
- ToPILImage()
 - Convert a tensor or an ndarray to PIL Image

Image Datasets

- **ImageFolder**
 - a generic dataset that you can use with your own images organized into sub-folders
 - Each sub-folder named after a class and containing the corresponding images
 - The "Rock-Paper-Scissors" dataset is organized like that:
 - Inside the rps folder there are three sub-folders named after the three classes (rock, paper, and scissors)

ImageFolder & Conversion Transforms

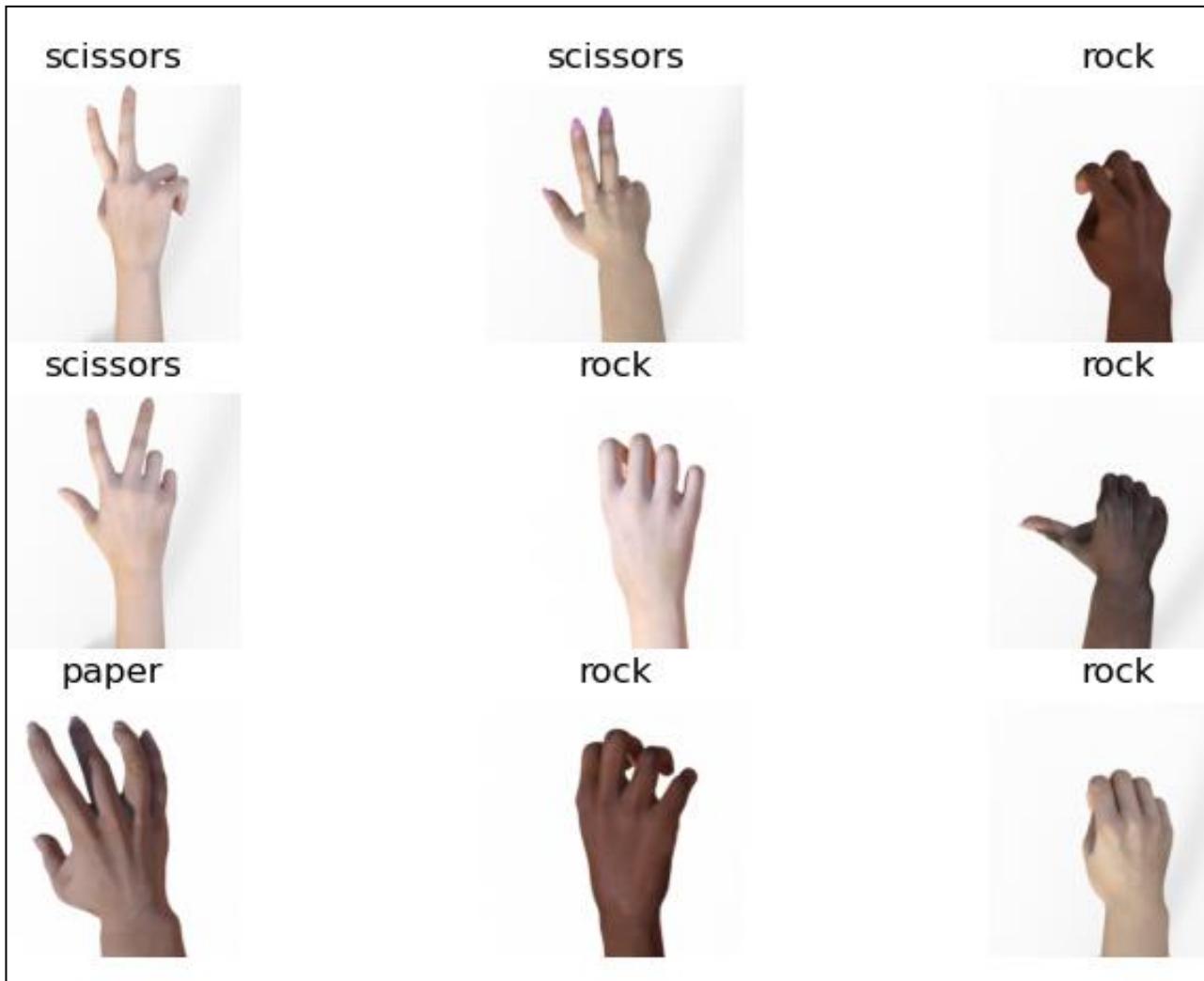
```
import numpy as np
from matplotlib import pyplot as plt
from PIL import Image
from torchvision.datasets import ImageFolder
from torchvision.transforms import Compose, Normalize, Resize, ToTensor, ToPILImage, RandomRotation
from torch.utils.data import DataLoader, Dataset, random_split
```

```
#Datasets & Loaders
transform1 = ToTensor()
imageDataSet=ImageFolder(root='dataset/rps/',transform=transform1)
imageLoader=DataLoader(imageDataSet, batch_size=9,shuffle=True)
image_batch_X, image_batch_Y=next(iter(imageLoader))
```

```
transform2 = ToPILImage()
plt.figure()
for i in range(9):
    pilImage=transform2(image_batch_X[i])
    classInd = image_batch_Y[i]

    plt.subplot(3,3,i+1)
    plt.imshow(pilImage)
    plt.axis('off')
    plt.title(imageDataSet.classes[classInd])
plt.show(block=True)
```

Output



Standardization Transform

- Similar to sklearn **standardScalar**
- Each Image Dataset has it's own mean and std for each image channel (for training set only)
- **Normalize**(mean=torch.Tensor([0.8502, 0.8215, 0.8116]),std=torch.Tensor([0.2089, 0.2512, 0.2659]))
- Remember always use the training set to compute statistics for standardization

Resize transforms

- Resize the input image to the given size
- `torchvision.transforms.Resize(size)`

Compose transformations

- from torchvision.transforms import **Compose**
- No need to apply transformations one by one
- Compose can combine several transformations into a single, big, composed, transformation.

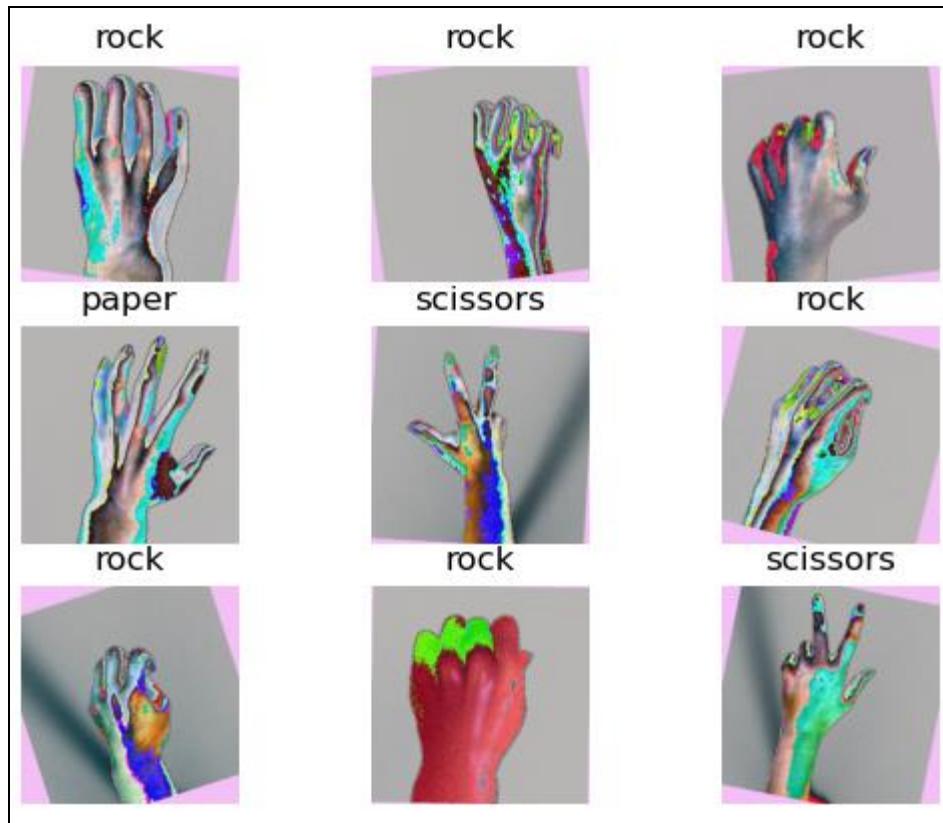
```
normTransform=Normalize(mean=torch.Tensor([0.8502, 0.8215, 0.8116]),std=torch.Tensor([0.2089, 0.2512, 0.2659]))  
transform=Compose([Resize(28),RandomRotation(90), ToTensor(), normTransform])
```

Data augmentation

- These transformations modify the training images in many different ways:
 - rotating, shifting, flipping, cropping, blurring, zooming in, adding noise to it, erasing parts of it...
- Why we need it?
 - clever technique to expand a dataset (augment it) without collecting more data
 - deep learning models are very data-hungry, requiring a massive amount of examples to perform well
- Not suited for every task
 - In **object detection**, you shouldn't do anything that changes its position, like flipping or shifting.

Example

```
normTransform=Normalize(mean=torch.Tensor([0.8502, 0.8215, 0.8116]),std=torch.Tensor([0.2089, 0.2512, 0.2659]))  
transform=Compose([Resize([500, 500]), RandomHorizontalFlip(p=.5), RandomAutocontrast(0.5),RandomRotation(20),ToTensor(),normTransform])  
  
imageDataSet=ImageFolder(root='dataset/rps/',transform=transform)  
imageLoader=DataLoader(imageDataSet, batch_size=9,shuffle=True)  
image_batch_X, image_batch_Y=next(iter(imageLoader))
```



Graded Home Task-4

- Build your own dataset class for RPS dataset
 - Don't use ImageFolder
 - Use Pillow to load images from folders of each class
 - Make a list of images & labels manually
- Add a compose transforms to your dataset class
 - Resize, RandomRotation, ToTensor, Normalize
- Create train & test loaders
 - Only do Resize & ToTensor transform for validation loader
 - You can create a flag in your dataset class
- Take a batch of images from your loader and display it