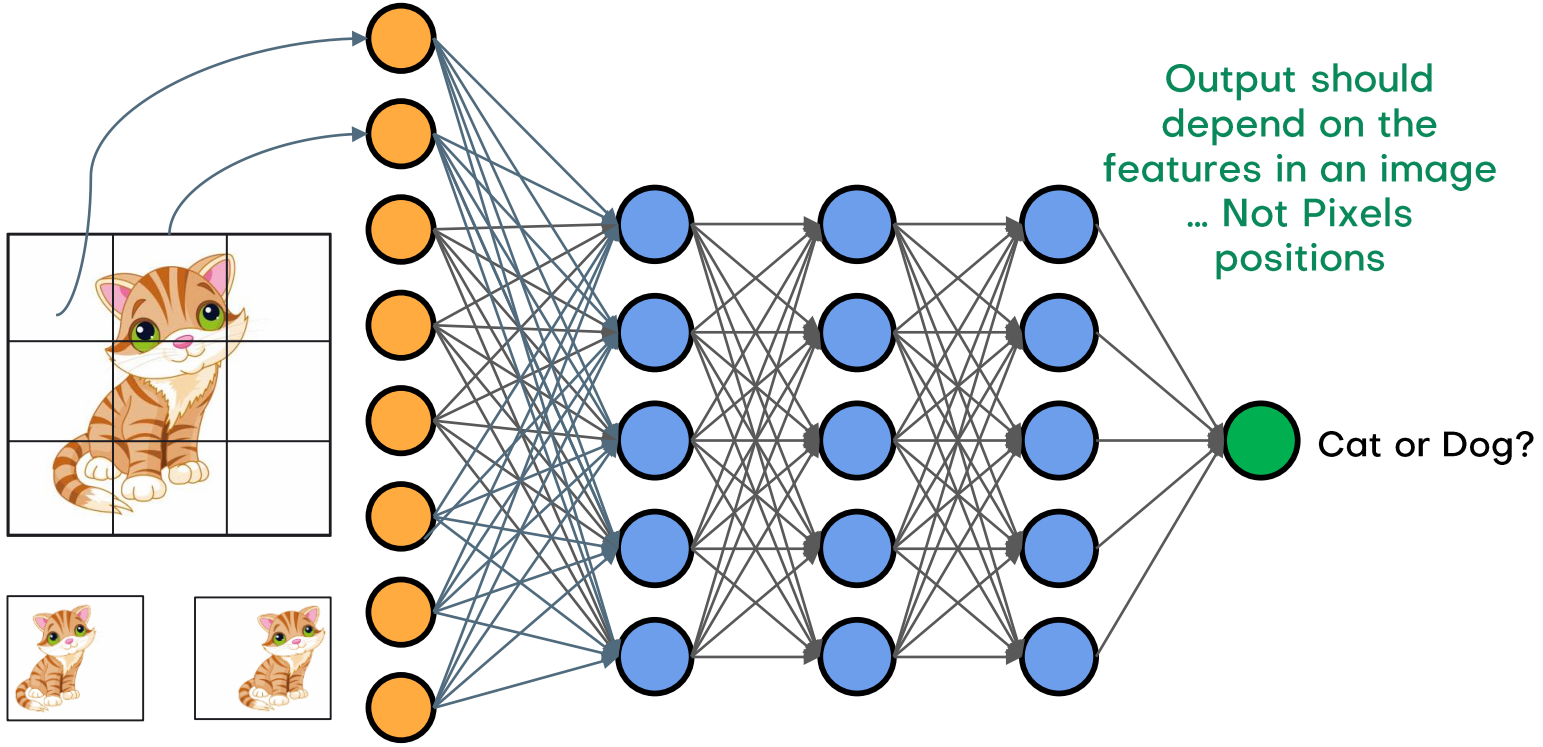
Deep Learning for Image Data

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

OUTLINE

- What is an image?
- Image types
- Image Operations
- Computer Vision
- DL for Computer Vision
- Convolutional Neural Networks CNNs
- Data Preparation for CNNs
- Transfer Learning

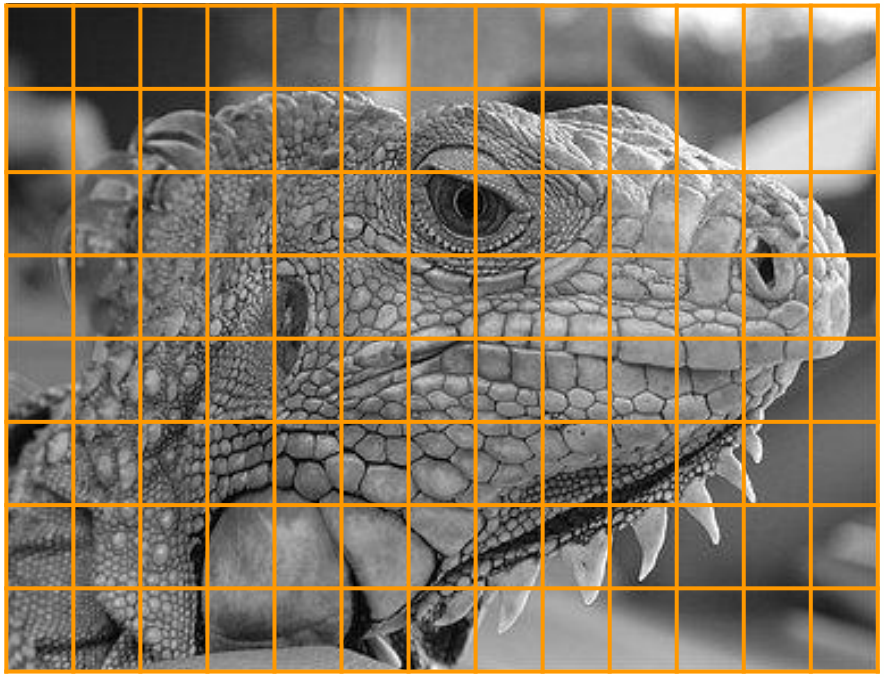
DL Architectures


Output should depend on the features in an image … Not Pixels positions
Cat or Dog?

What is an Image?

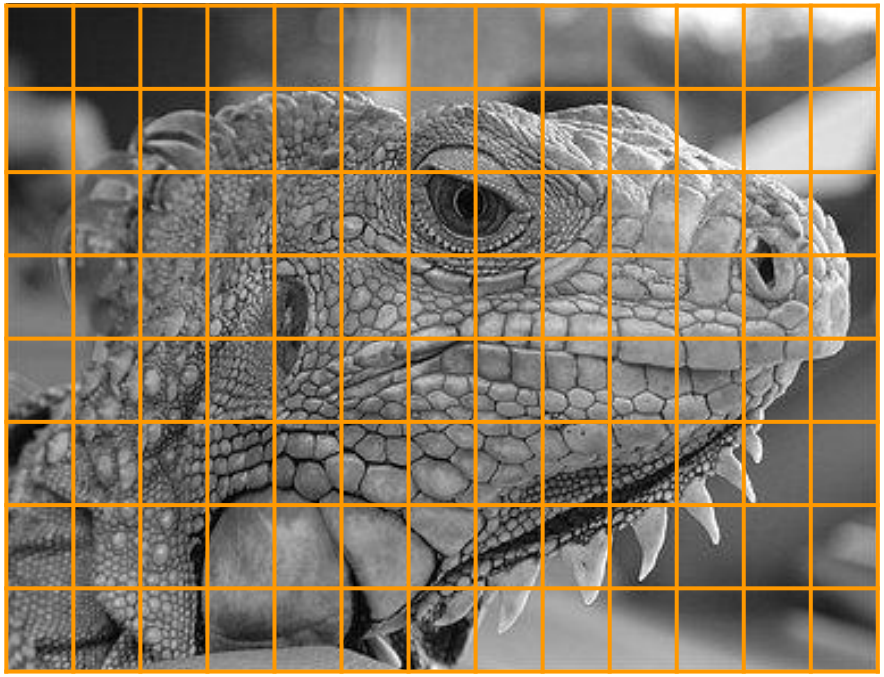
What is an Image?

- An image is like a puzzle with a huge number of pieces fitted together, each piece has the same size but can have different colourings or intensity.
- These small pieces are what we call pixels.
- So an image is a matrix of pixels



What is an Image?

- So an image is simply a matrix of pixels.
- Pixels are simply numbers that reflects the color intensity at a specific location.
- The higher the value of the pixel is, the more intensity of a certain color it represents.



What is an Image?

- So an image is simply a matrix of pixels.
- Pixels are simply numbers that reflects the color intensity at a specific location.
- The higher the value of the pixel is, the more intensity of a certain color it represents.
- But how do we assign a value for a pixel?

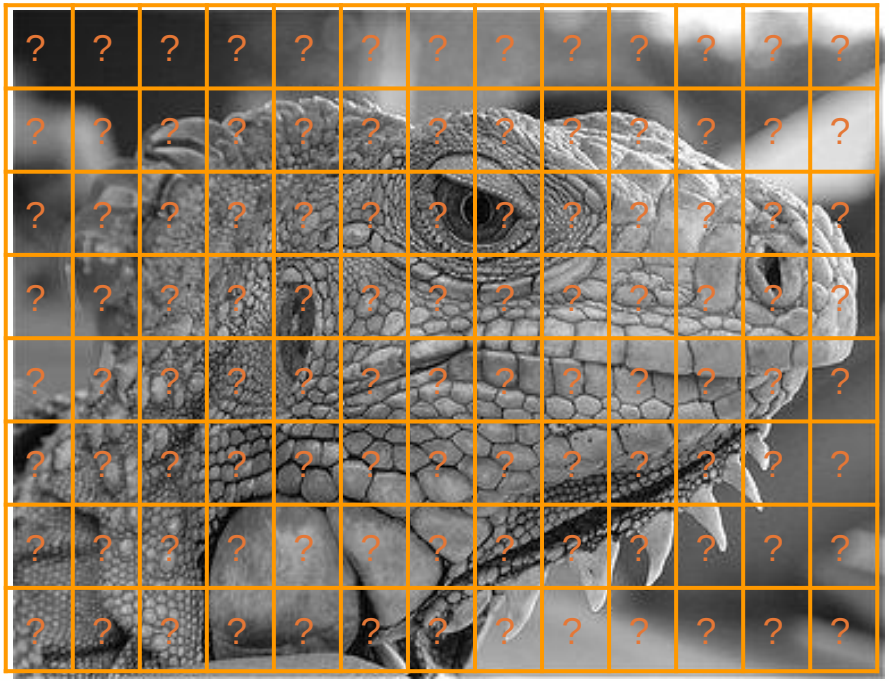
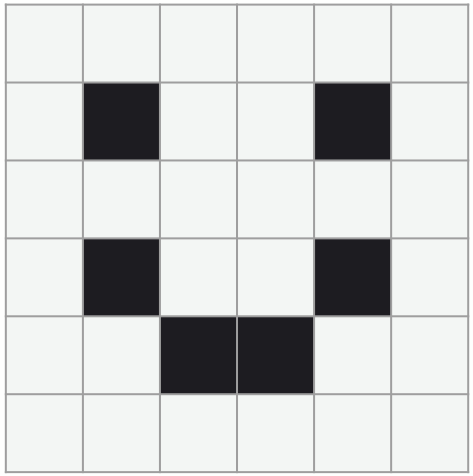


Image Types

Black & White Images

- We start with the basic type of images which is black and white.
- The value of a pixel in such images is either 0 (black) or 1 (white)

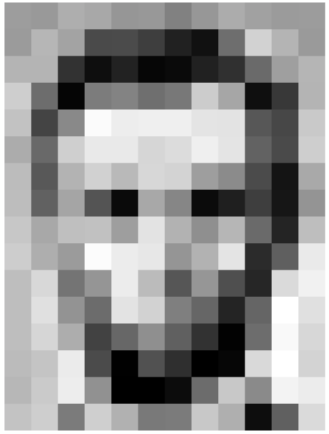


1	1	1	1	1	1
1	0	1	1	0	1
1	1	1	1	1	1
1	0	1	1	0	1
1	1	0	0	1	1
0	1	1	1	1	1

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

Gray Level Image

- The two extreme colors are black (0)… and white (255).
- The numbers in between show different level of Grey color.
- Store the intensity of pixels in a matrix, and there you have it!



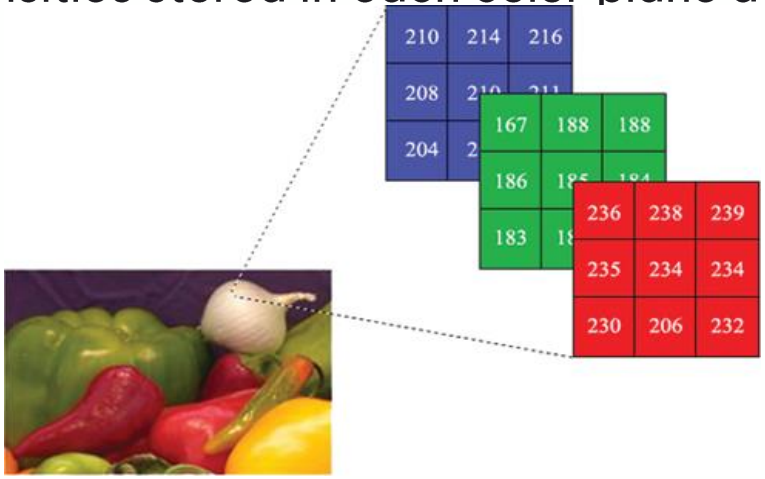
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	84	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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
196	206	123	207	177	121	123	200	175	13	96	218

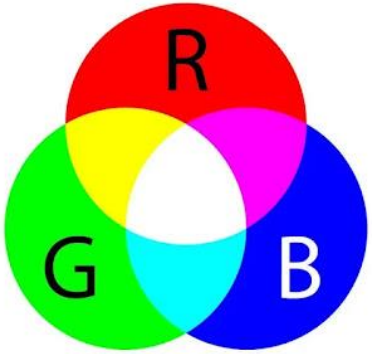
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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
196	206	123	207	177	121	123	200	175	13	96	218

Source: https://ai.stanford.edu/~syyeung/cvweb/Pictures1/imagematrix.png

RGB

The color of each pixel is determined by the combination of red, green, and blue intensities stored in each color plane at the pixel's location.





https://api.intechopen.com/media/chapter/51312/media/fig3.png

RGB

Color of the pixel at (0,1):

RGB triplet stored in (0,1,0:2).

- ➔ (0,1,0) = 144
- ➔ (0,1,1) = 125
- ➔ (0.1,2) = 187

The color for the pixel at (0,1) is

144, 125, 187

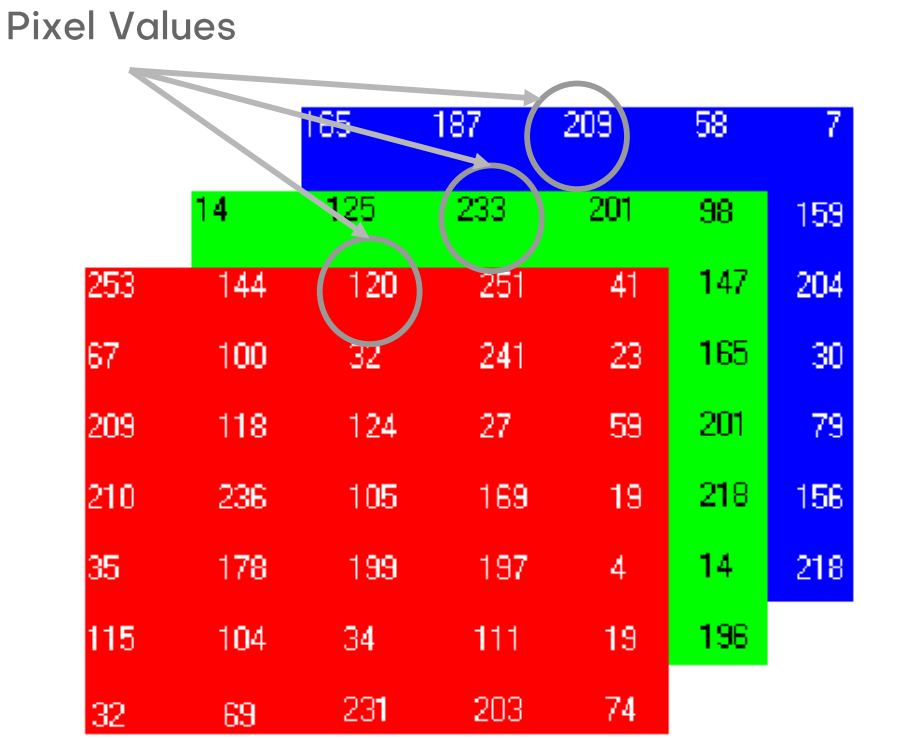


Image Operations

Image Operations

Since Images are simply matrices of numbers, we can perform many operations on them, just like what we do with normal matrices.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

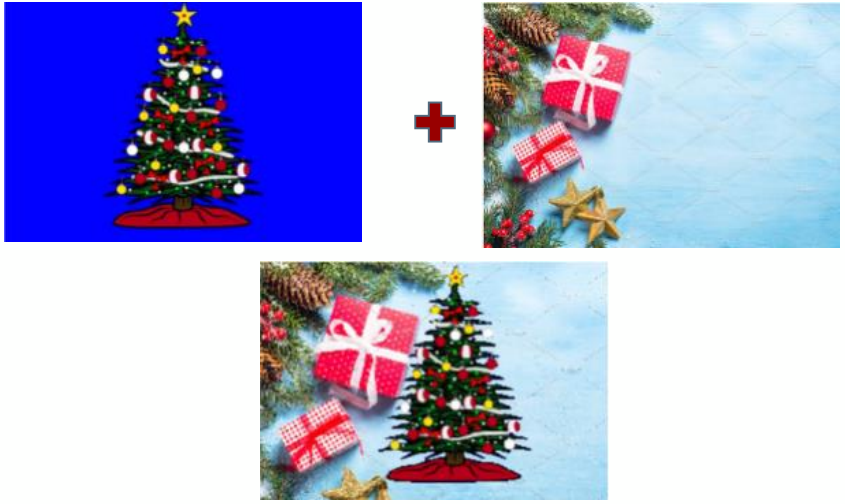


Image Operations

A lot of operations can be applied on images just because of the fact that they're a bunch of numbers.

Splitting and Merging Image Channels

Image Blending

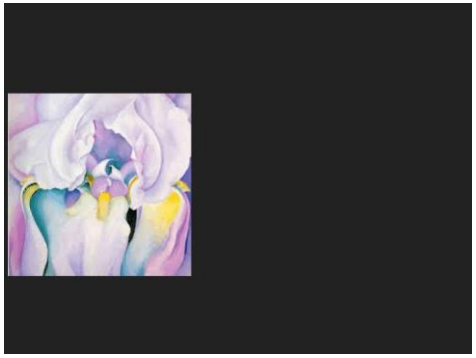


Image Rotation

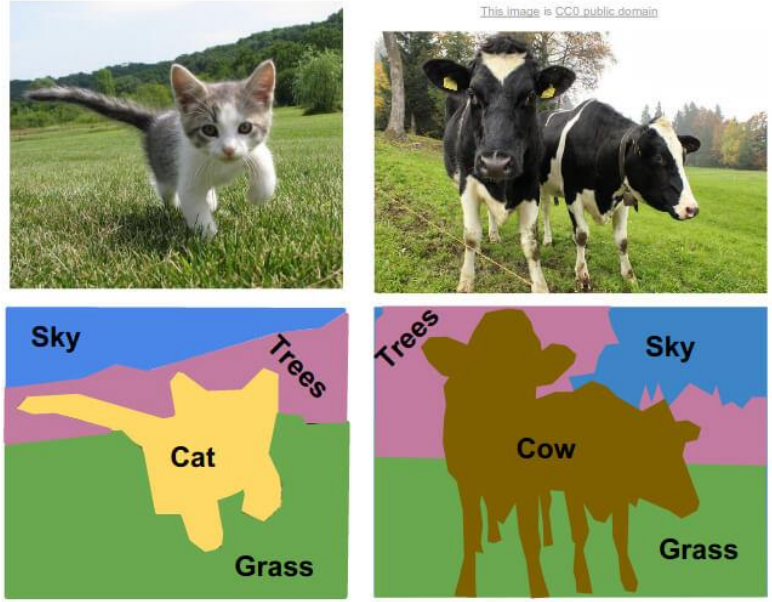
Image Operations

- To do these transformations, and even more, you can use the OpenCV library.
- The OpenCV Library has become a standard for dealing with image related tasks.

Computer Vision

Computer Vision

- Computer Vision (CV) is a field that deals with extracting information from images
- Common CV tasks: image classification, object detection, semantic segmentation

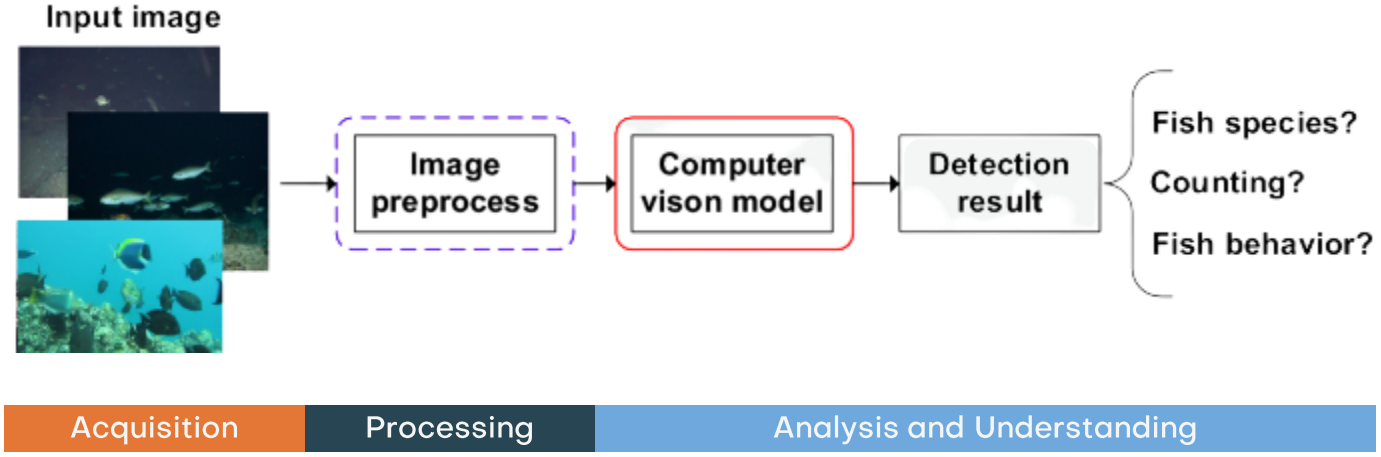


Computer Vision is not Image Processing

- Image Processing
  - Image → Image (process of creating a new image from an existing image)
  - Examples: de-noising, edge detections, etc.
- Computer Vision
  - Image → Content (observations)
  - Examples: face detection, object tracking, etc.
- A Computer Vision system may require image processing to be applied to raw input, e.g. pre-processing images.

Computer Vision Framework

1. Image acquisition
2. Image processing
3. Image analysis and understanding



Challenges of Computer Vision

After decades of research, computer vision remains unsolved, at least in terms of meeting the capabilities of human vision.

WHY?

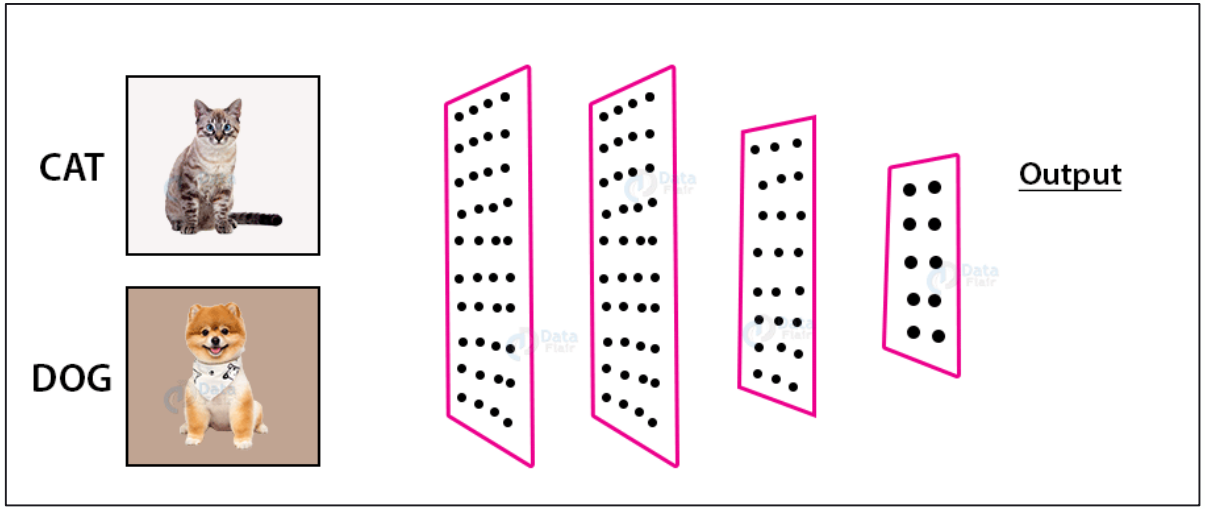
- We don't have a strong grasp of how human vision works
- The complexity inherent in the visual world (different orientation, in many lighting conditions, with any type of occlusion from other objects, and so on)

Challenges: Illumination

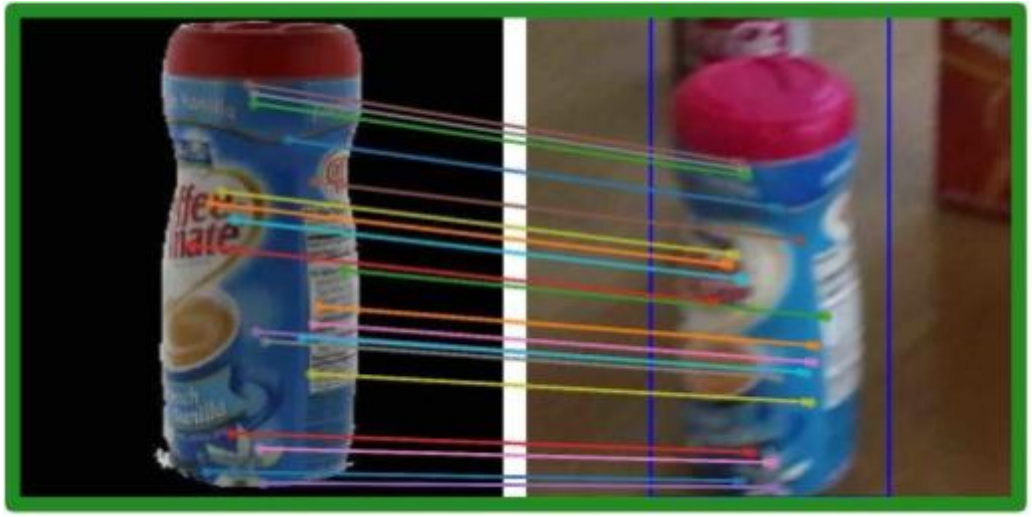
Challenges: Occlusion

Computer Vision Applications

Image Classification



Object Verification



Object Landmark Detection

What are the key points for the object in the image?

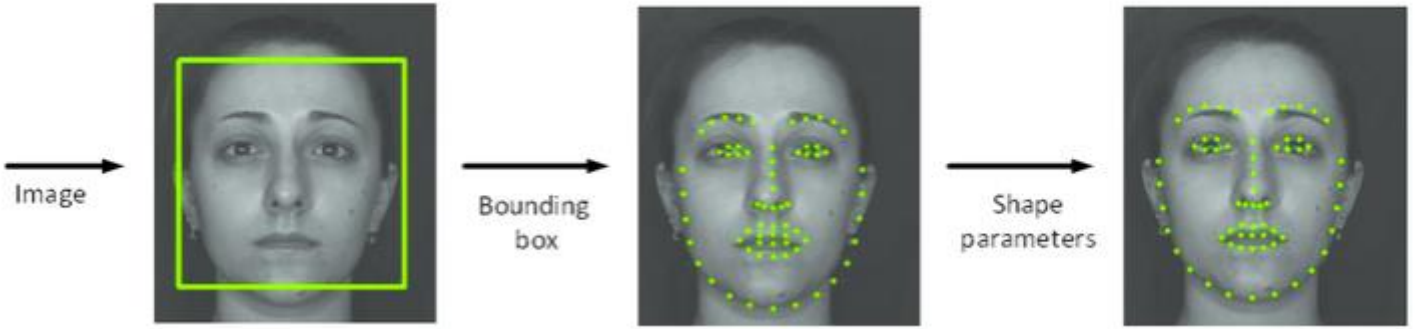
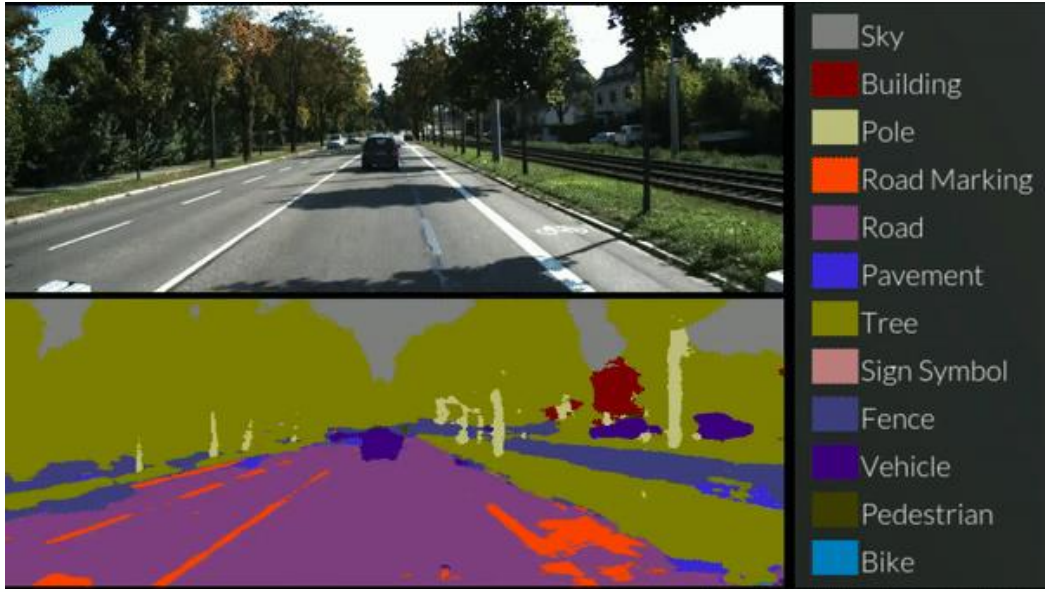


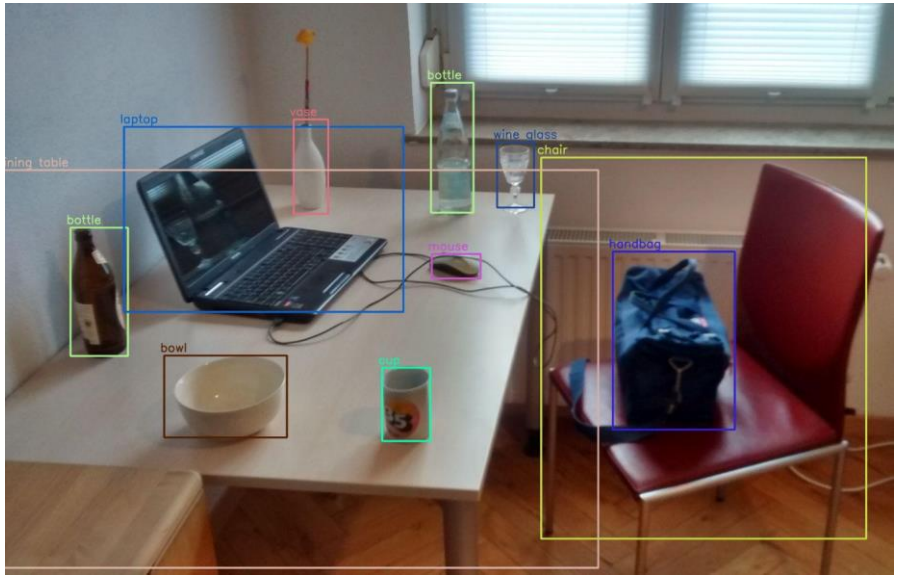
Image Segmentation

Pixels belonging to the object



Object Detection

What are the objects in the image?

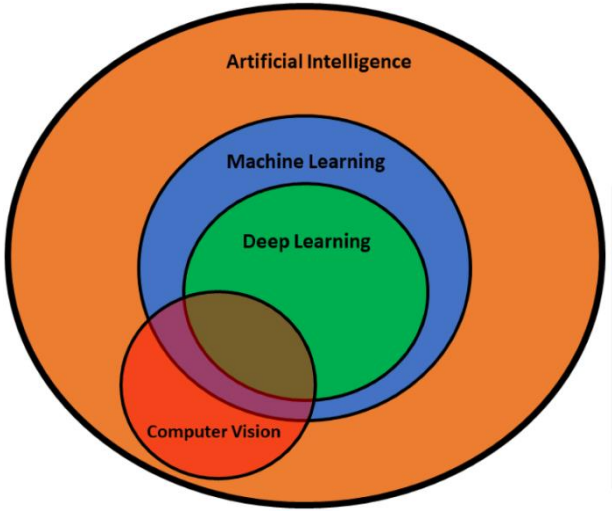


Deep Learning for Computer Vision

Deep Learning for Computer Vision

Since Computer Vision is about extracting information from images, Deep Learning can help to perform some Computer Vision tasks:

- Object Detection
- Image Classification
- Semantic Segmentation
- ...



Deep Learning for Computer Vision

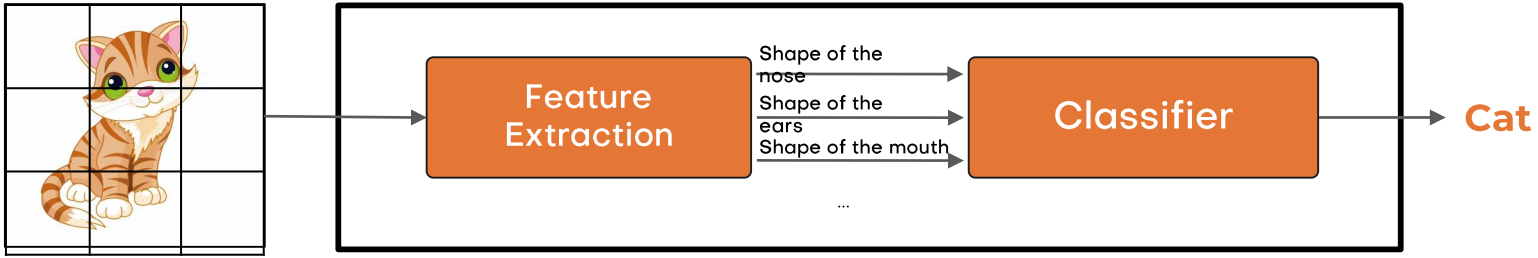
- The problem in that case would be formulated as a normal DL problem, the only difference is that the input is supposed to be an image instead of a tabular data.
- Example for Image Classification

Deep Learning for Computer Vision

- We want to dissect the deep learning model and see what it contains and how it deals with an image.
- At the very end of it, we should have a normal classifier that predicts Cat or Dog.

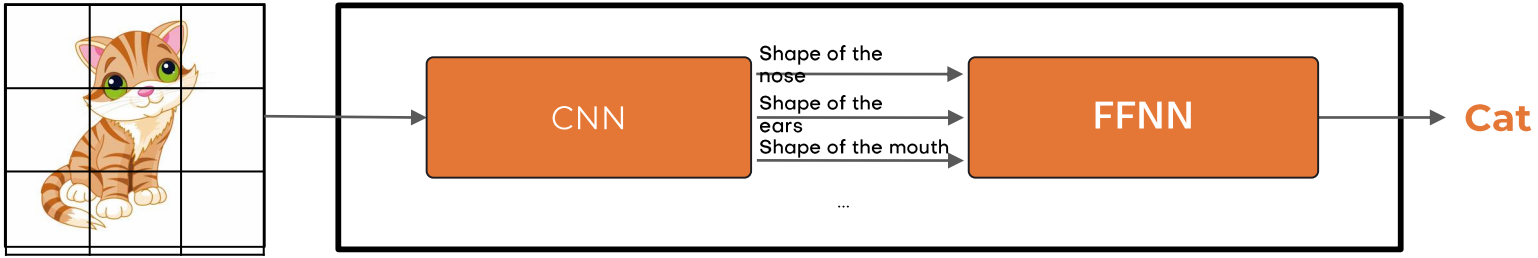
Deep Learning for Computer Vision

- The classifier ideally would predict according to several features from the image like the shape of the nose, shape of the ears, etc.
- These features should be extracted from the input image.



Deep Learning for Computer Vision

- To do the feature extraction in an automatic way, we use what we call Convolutional Neural Networks (CNN), and the classifier is a simple Feed-Forward Neural Network (FFNN)



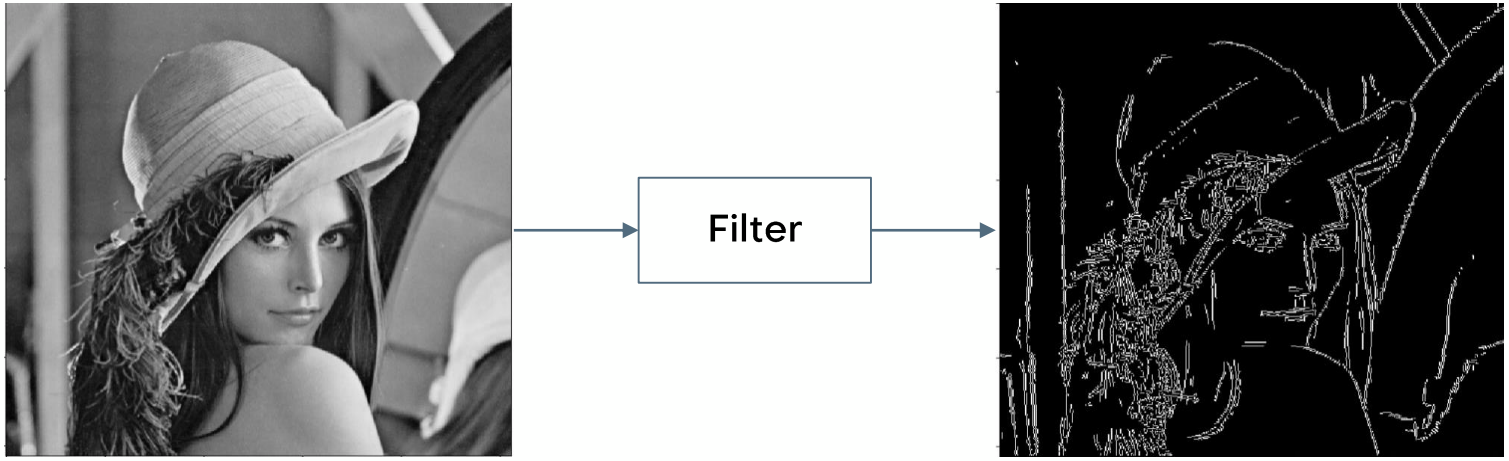
Convolutional Neural Networks (CNNs)

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs) are Neural Networks that operate on images, and extract features from them.
- To understand CNNs, we have to understand their basic building block which is a Convolution.

Convolution Operation

The convolution is about applying a filter to an input image to extract specific information from it.



Convolution Operation

- Filters are simply matrices that help us perform transformations on an original image to specific information from it depending on the values of these filters.

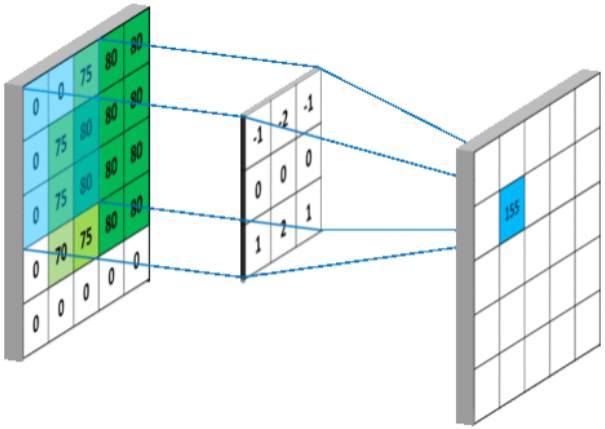
Input image

Original	Gaussian Blur	Sharpen	Edge Detection
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

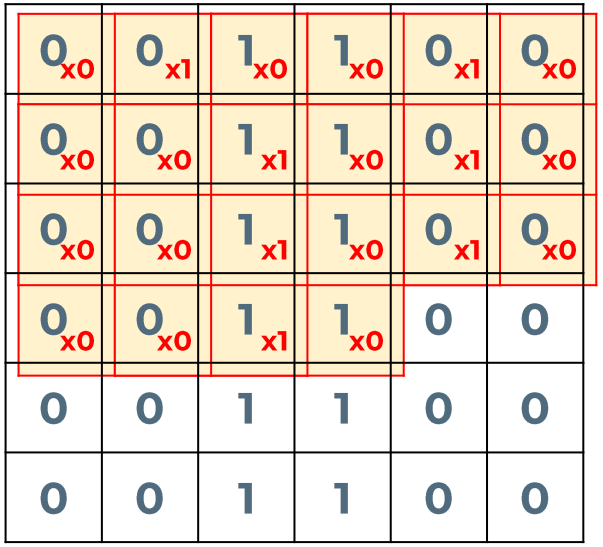
https://miro.medium.com/max/1206/1*ZPXWZDIHFbTxs-6KVPS5gg.png

Convolution Operation

- The convolution “mathematically” is simply about sliding a filter on an image and performing the the sum of the element wise multiplication between the filter and the region of the image that it covers



Convolution Example



= 0x0 + 1x0 + 0x1
+ 0x0 + 1x0 + 0x1
+ 0x0 + 1x0 + 0x1

0	1	0
0	1	0
0	1	0

Kernel

0	3	3	0
0	3	3	0
0	3	3	0
0	3	3	0

Result

Convolution Parameters

Filter Size

You can vary the size of your filter

1	2	1
2	4	2
1	2	1

3x3

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

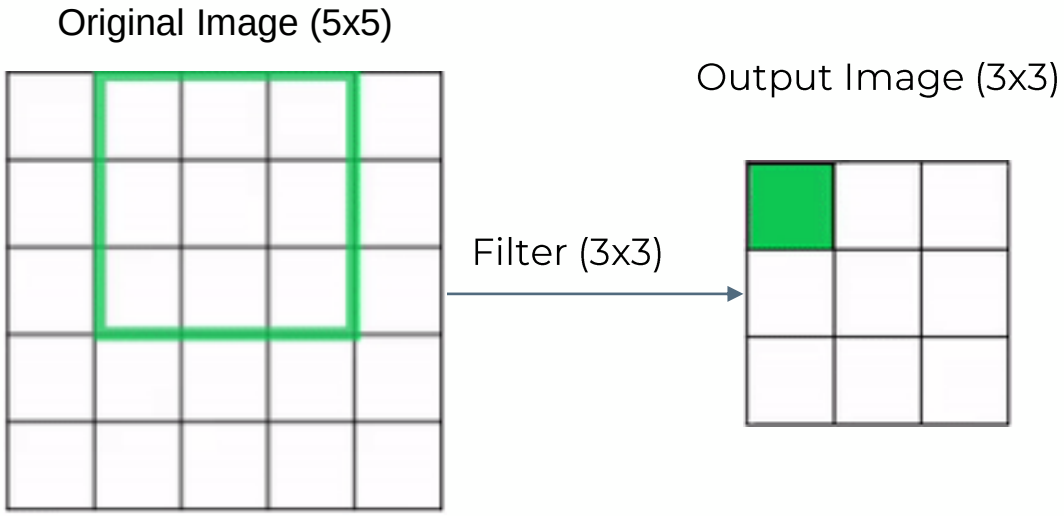
5x5

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

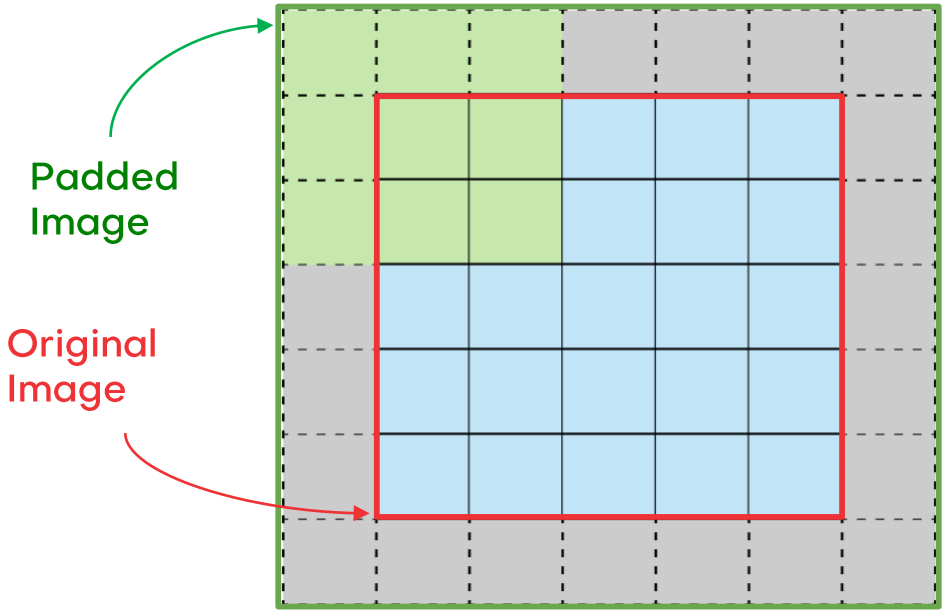
7x7

The Border Effect Problem

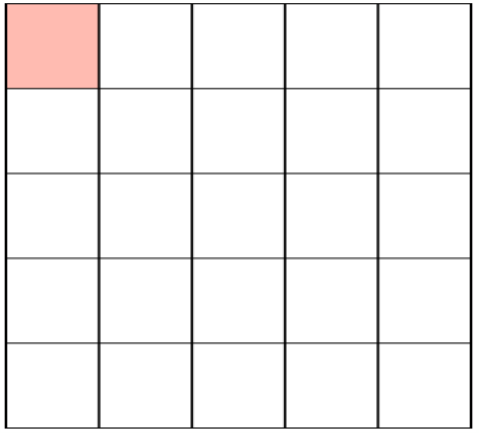
- An image of size 5*5 gets reduced to 3*3 after applying a 3*3 filter on it. Information on the border gets lost.



Padding to solve the Border Effect Problem



Output Image



Padding Example

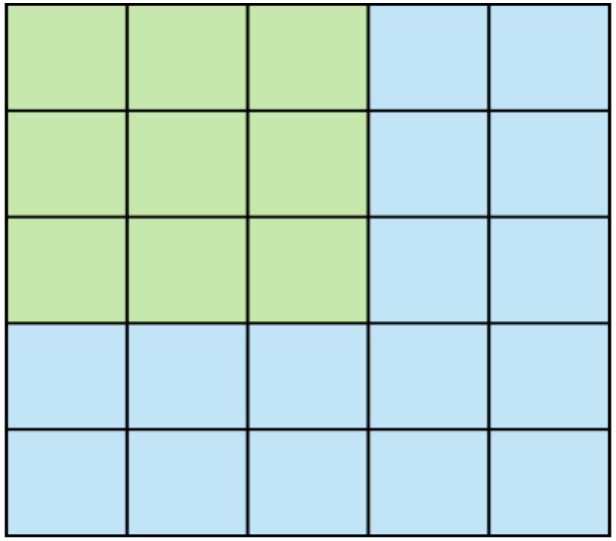
0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

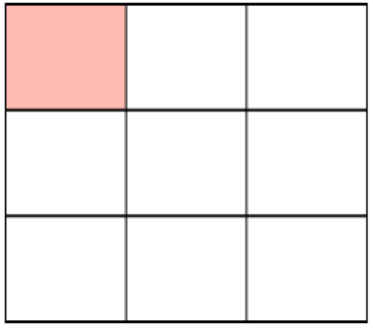
0	-1	0
-1	5	-1
0	-1	0

114				

Downsample Input with Stride

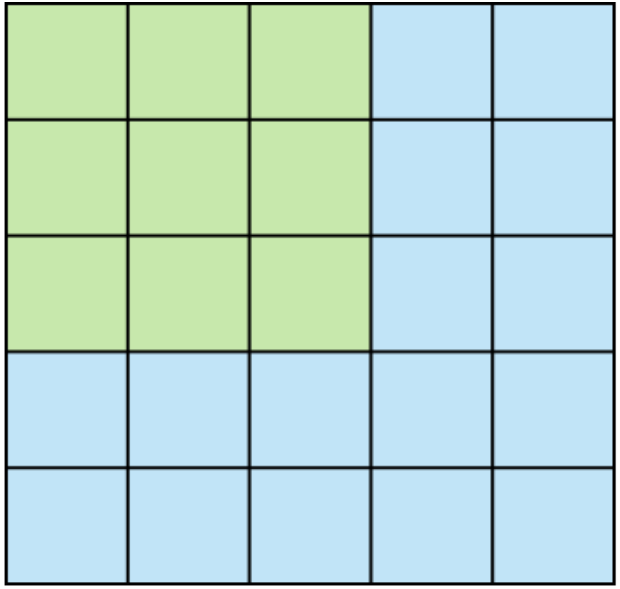


Stride = 1

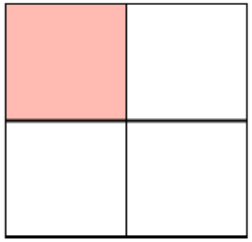


Output
Feature Map

Downsample Input with Stride



Stride = 2

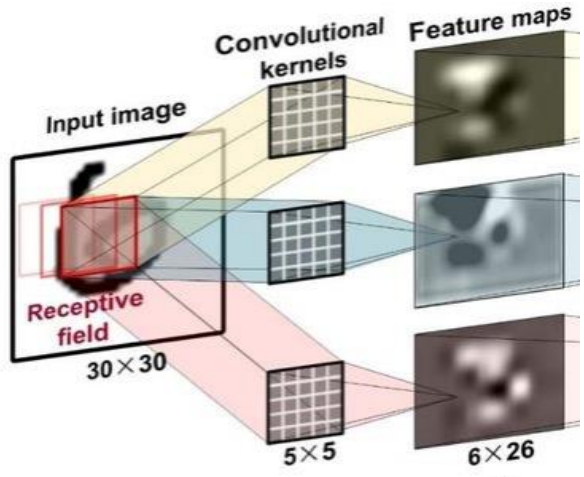


Output
Feature Map

Convolutional Layer

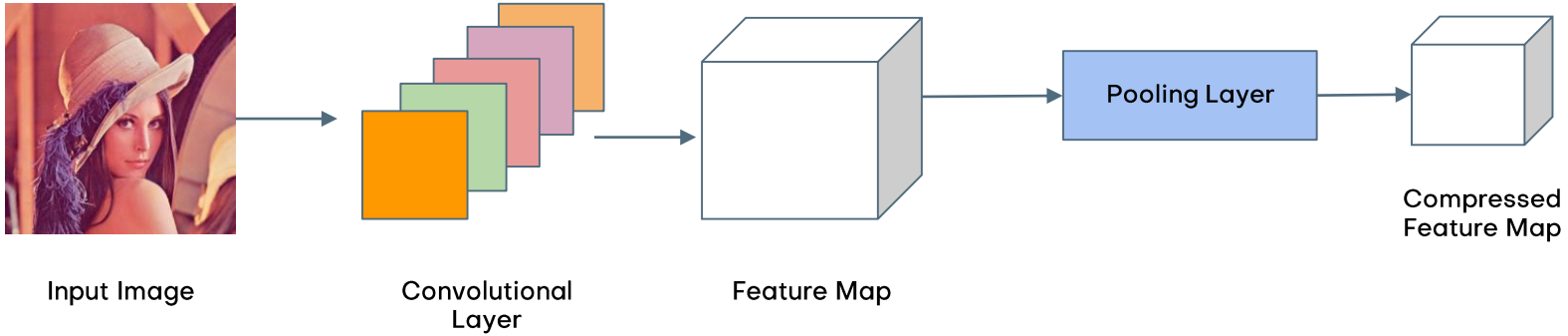
Convolutional Layers

Depending on the kernel values, the output will differ, this is why to one input, we can apply many filters to obtain multiple information from the input image → This is what we refer to as a convolutional layer



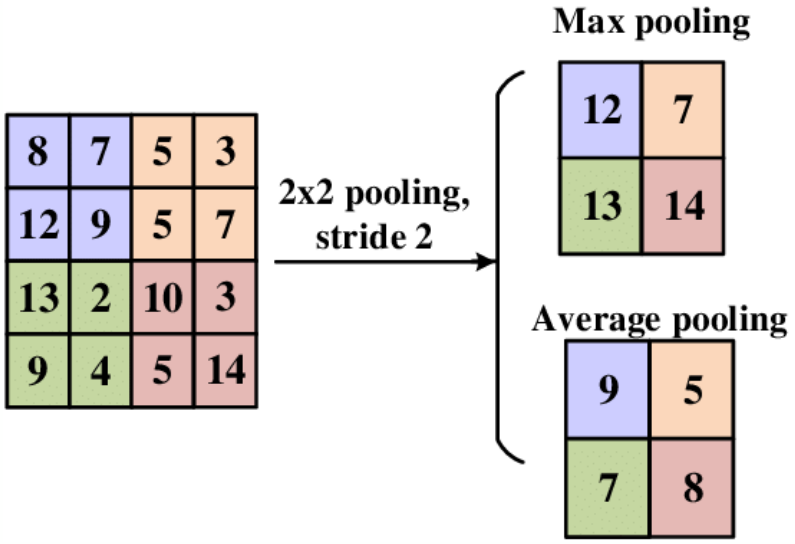
Pooling Layers

- After extracting the feature map, we need to compress this output so that the final classifier can use it to predict in an easier way.
- To do that we use what we call Pooling layers



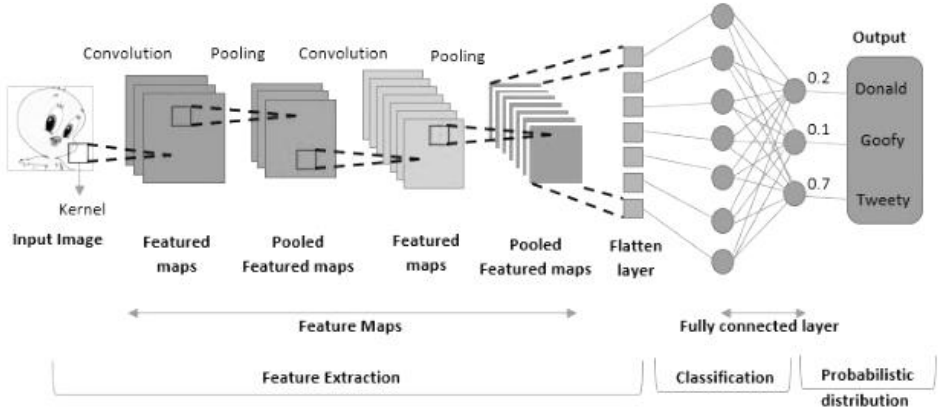
Pooling Layers

- There are 2 major forms of pooling: Max Pooling and Average Pooling
- The Max Pooling takes the maximum value of an image region
- The Average Pooling takes the average value of an image region



Convolutional Neural Network

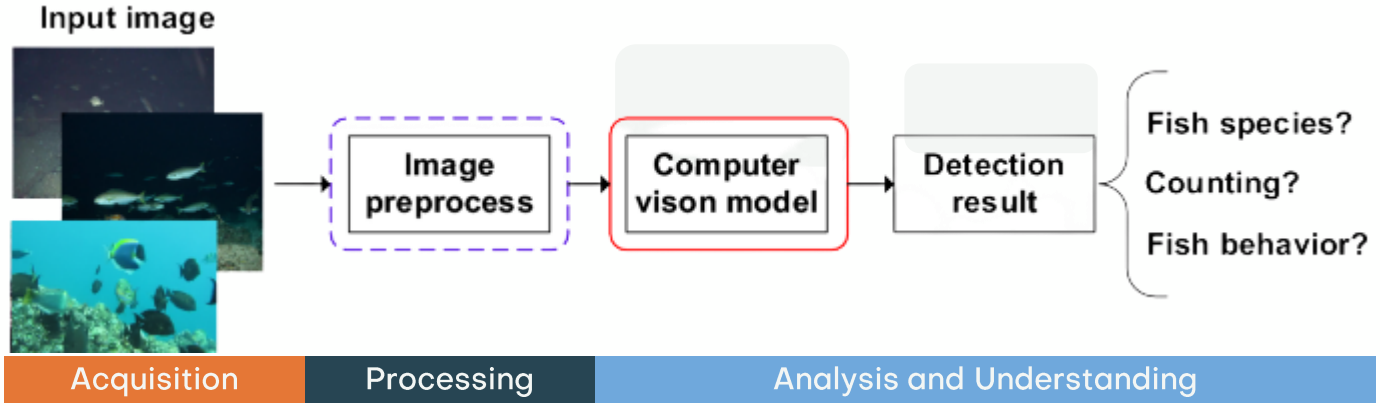
- When we stack all these layers we obtain the convolutional neural network (CNN)
- The goal of this network is to learn the optimal set of filter values along with the weights in the fully connected part.



Data Preparation for CNNs

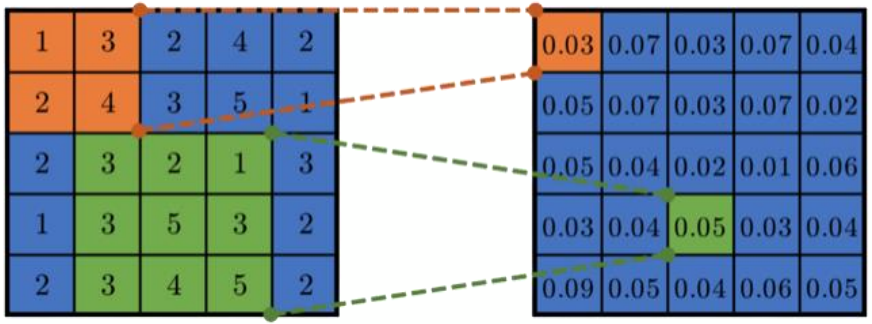
Data Preparation

- Just like any other DL problem, once we define the structure of our model, we have to prepare the data in a way to be able to train the model on it



Data Preparation

- Normalize Pixel Values
  As such it is good practice to normalize the pixel values so that each pixel has a value between 0 and 1.
  Since pixel values are between 0 and 255, the normalization can happen by dividing the values by 255.



Data Preparation

- Center Pixel Values
  Centering requires that a mean pixel value be calculated prior to subtracting it from the pixel values.

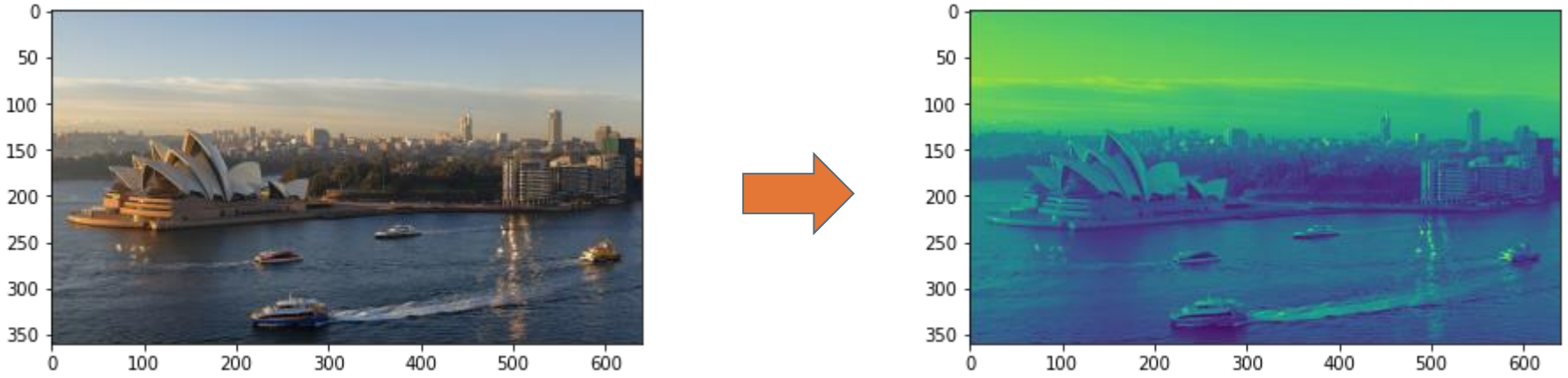


Image Data Generator

This Keras class provides a suite of techniques for scaling pixel values in your image dataset prior to modeling:

- Wrap your image dataset
- Then (when requested) return images in batches to the algorithm during training, validation, or evaluation
- Apply the scaling operations just-in-time

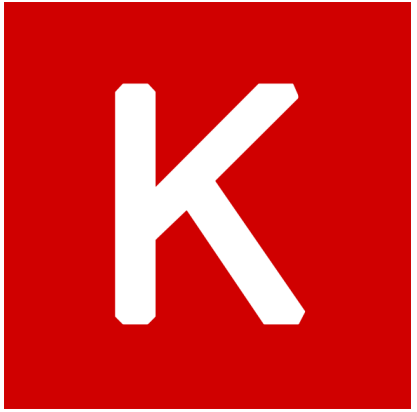


Image Data Generator

The usage of the ImageDataGenerator class is as follows:

1. Load your dataset
2. Configure the ImageDataGenerator class
3. Calculate image statistics where applicable (by calling the fit() function)
4. Create data iterators for each data split (train, test, validation) and point them to the datasets
5. Use the data iterators to fit the model (pass the instance to the **model.fit()** function)
6. Use the data iterators to evaluate the model (pass the instance to the **model.evaluate()** function)

Image Data Generator

```
# create data generator with appropriate arguments
datagen = ImageDataGenerator(...)

# calculate scaling statistics on the training dataset
datagen.fit(trainX)
# create training data iterator
train_it = datagen.flow(trainX, trainY, batch_size=64)
# create test data iterator
test_it = datagen.flow(testX, testY, batch_size=64)
# train model
model.fit(train_it, validation_data=test_it, ...)
```

Load Large Dataset from Directory

Dataset Directory Structure

```
data/
data/train/
data/test/
data/validation/
```

→

```
data/
data/train/
data/train/red/
data/train/blue/
data/test/
data/test/red/
data/test/blue/
data/validation/
data/validation/red/
data/validation/blue/
```

→

```
data/train/red/car01.jpg
data/train/red/car02.jpg
data/train/red/car03.jpg
...
data/train/blue/car01.jpg
data/train/blue/car02.jpg
data/train/blue/car03.jpg
```

Load Images Progressively

```
# create data generator with appropriate arguments
datagen = ImageDataGenerator(...)

# create training data iterator
train_it = datagen.flow_from_directory("dataset/train",
class_mode="binary", batch_size=64)

# create test data iterator
test_it = datagen.flow_from_directory("dataset/test",
class_mode="binary", batch_size=64)
# train model
model.fit(train_it, validation_data=test_it, ...)
```

Image Augmentation

Image augmentation with ImageDataGenerator!

- Image shifts
- Image flips
- Image rotations
- Image brightness
- Image zoom

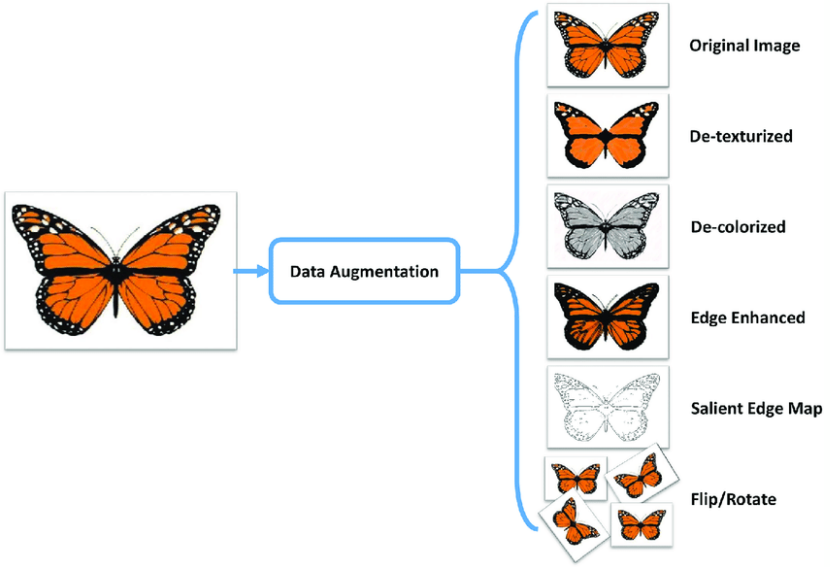
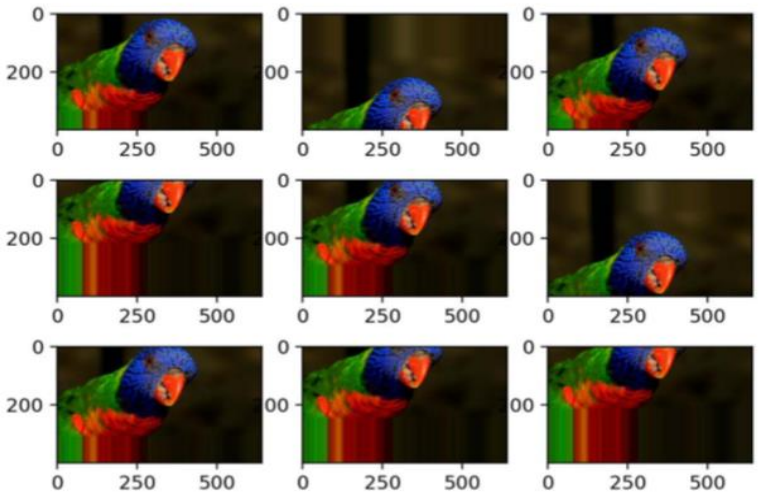


Image Shifting

Height_shift_range =[*min*, *max*]



Width_shift_range = x

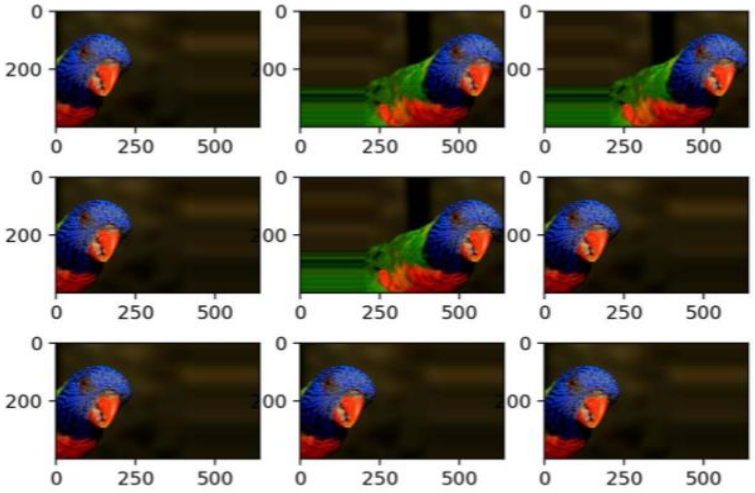
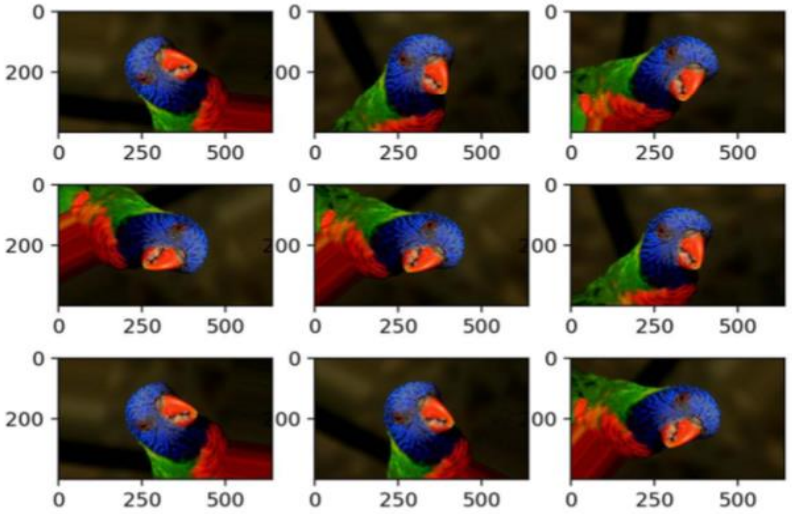


Image Flipping

horizontal_flip=True



vertical_flip=True

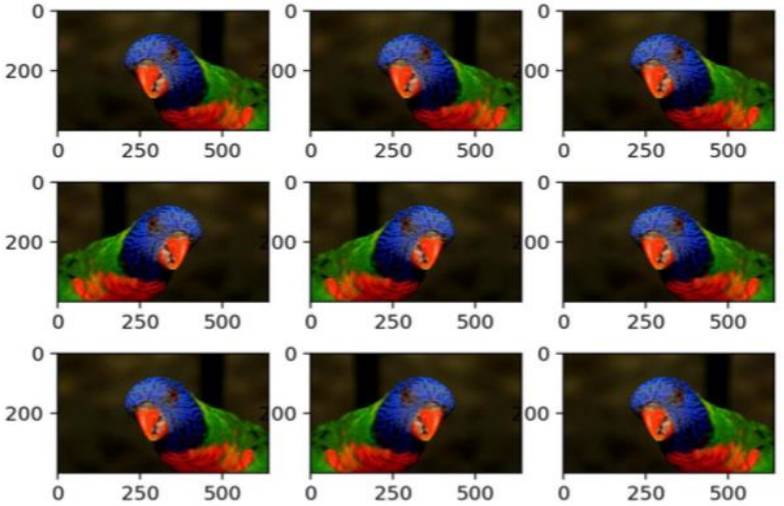


Image Rotation

rotation_range=*angle*

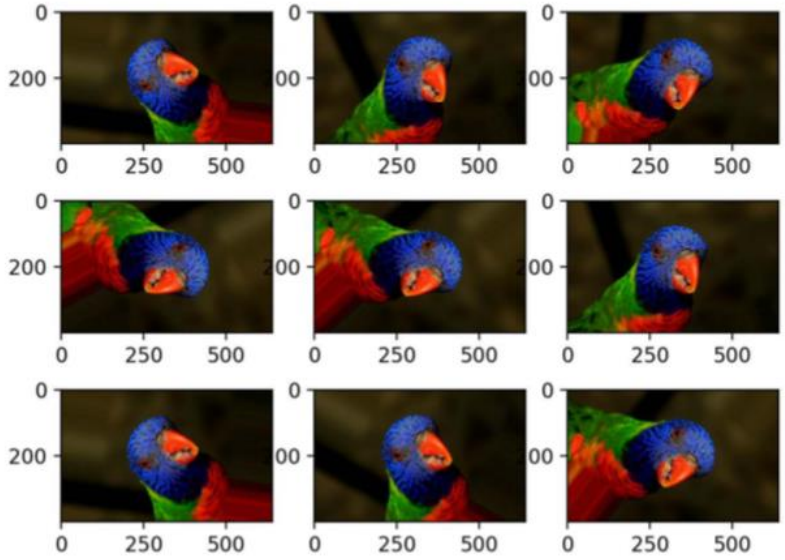
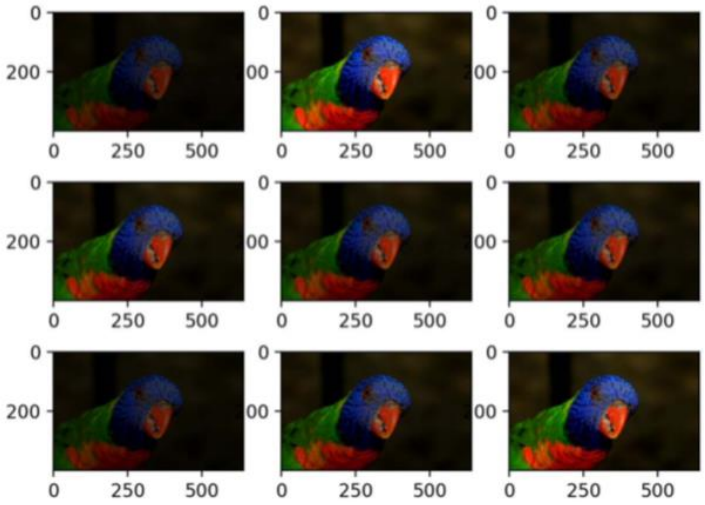


Image Brightness

brightness_range=[*min*,*max*]

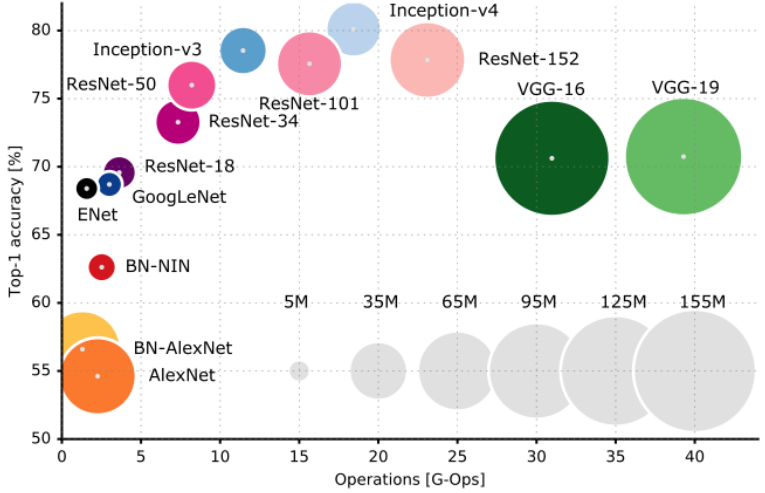


Hands-on: Mastering CNN Architectures

Advanced CNN Architectures

Advanced CNN Architectures

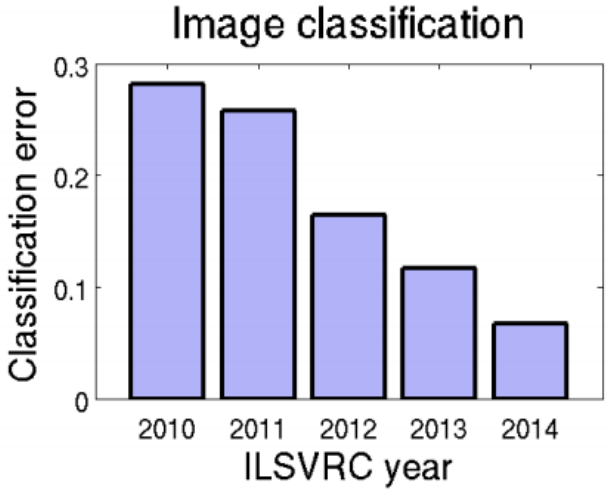
CNNs have been a booming topic, which is why, many advanced CNN architectures have been proposed with state of the art performance.

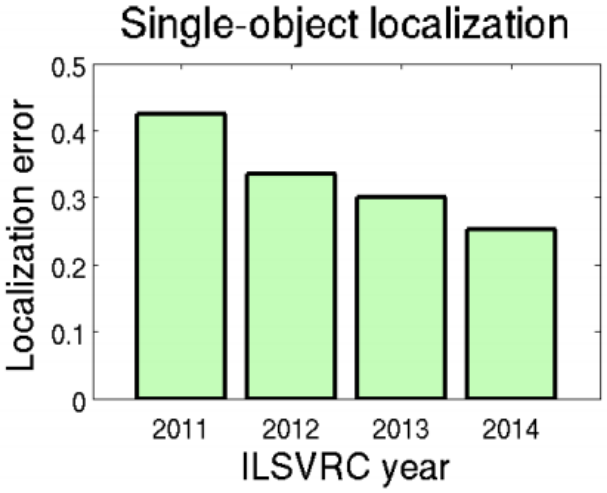


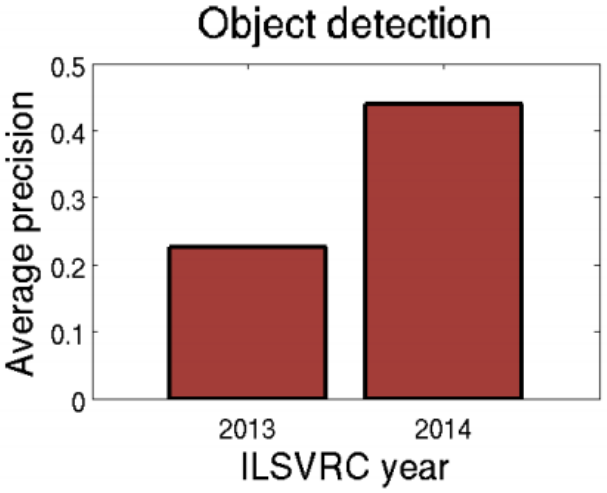
ImageNet Dataset

- Advanced architectures require a large dataset for training.
- ImageNet is a vast collection of human-annotated photographs used to develop computer vision algorithms.
- The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition using subsets of ImageNet to promote the development and benchmarking of advanced algorithms.

ImageNet Large Scale Visual Recognition Challenge



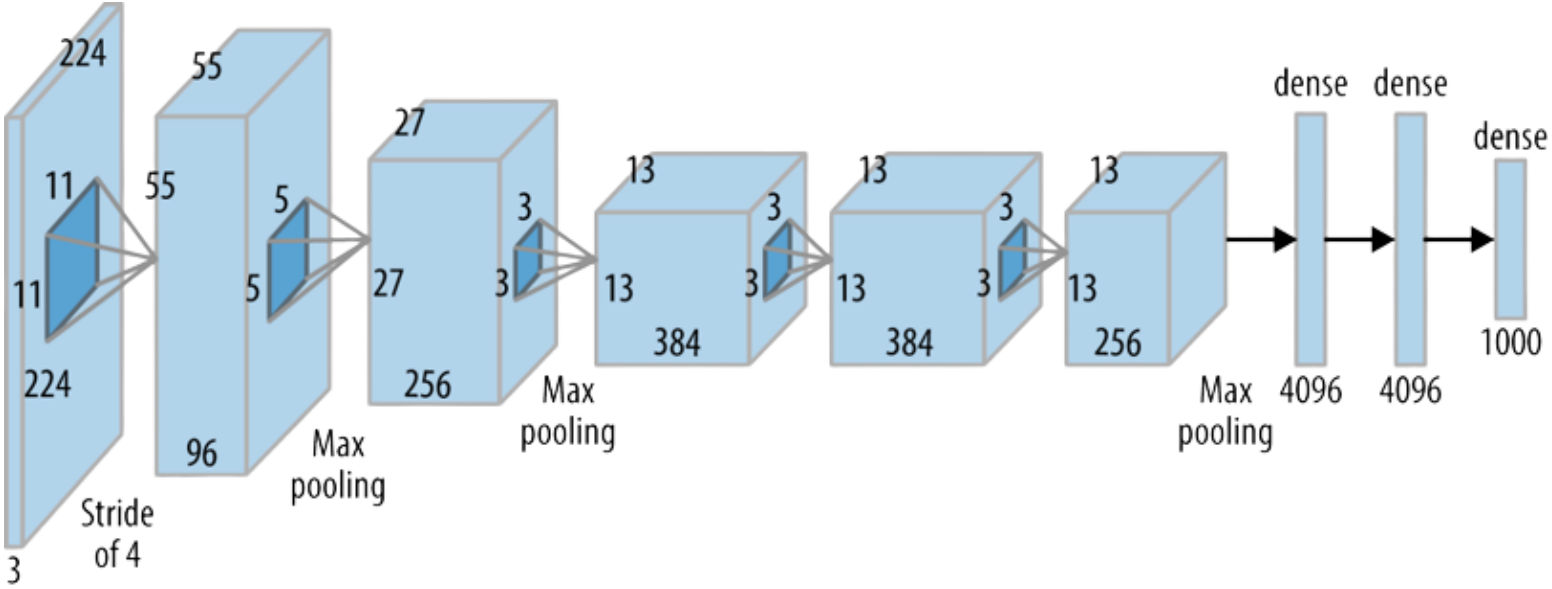




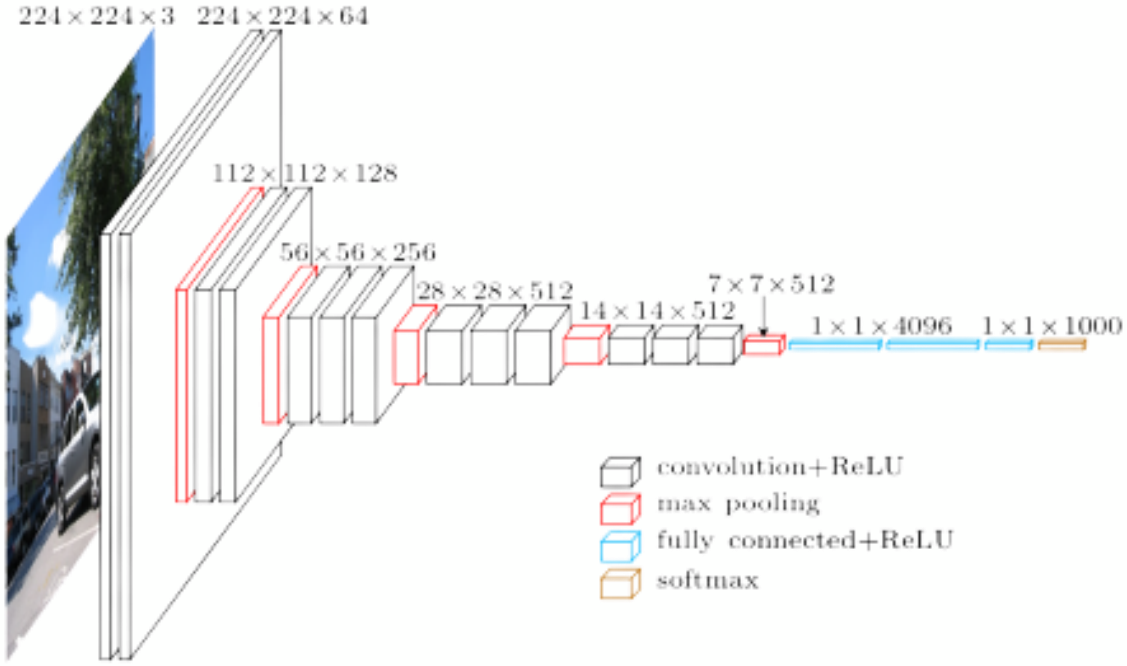
ILSVRC Results

- 2012: AlexNet (SuperVision)
- 2013: ZFNet (Clarifai)
- 2014 (object detection): Inception (GoogLeNet)
- 2014 (image classification): VGG
- 2015: ResNet

AlexNet

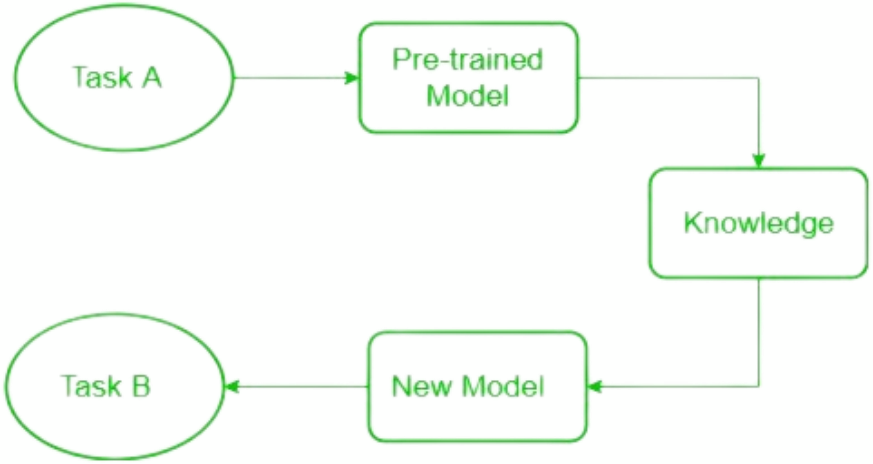


VGG 19


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

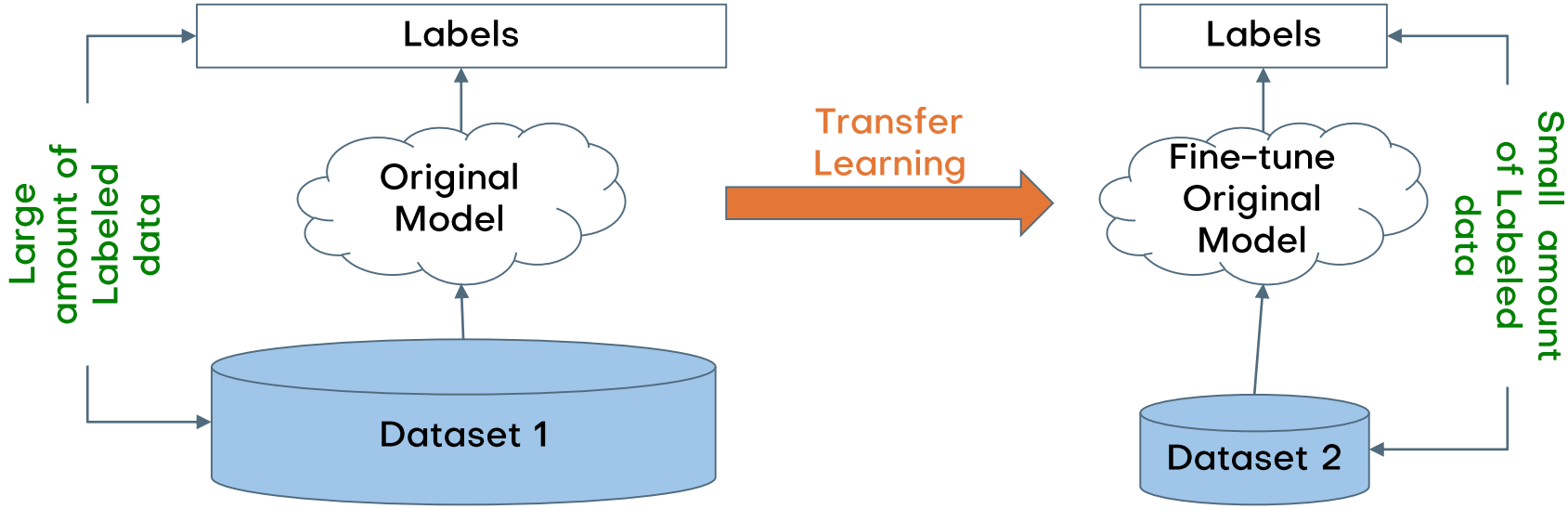
Why Advanced Architectures?

- These models have been trained on huge dataset, which means they have acquired a large knowledge. We can use this knowledge and build on top of it to build models instead of training things from scratch
- This is what we call transfer learning



Transfer Learning for Limited Data

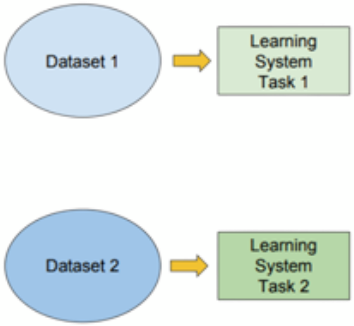
Transfer Learning is a powerful technique to use when dealing with limited amount of data.



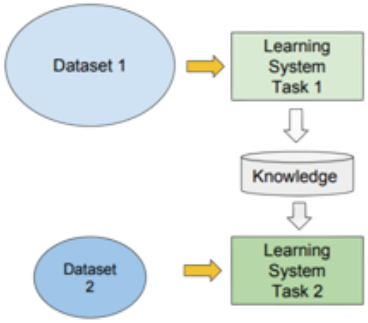
Transfer Learning for Limited Data

Traditional ML vs Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



شكراً لكم

Thank you

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority