



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

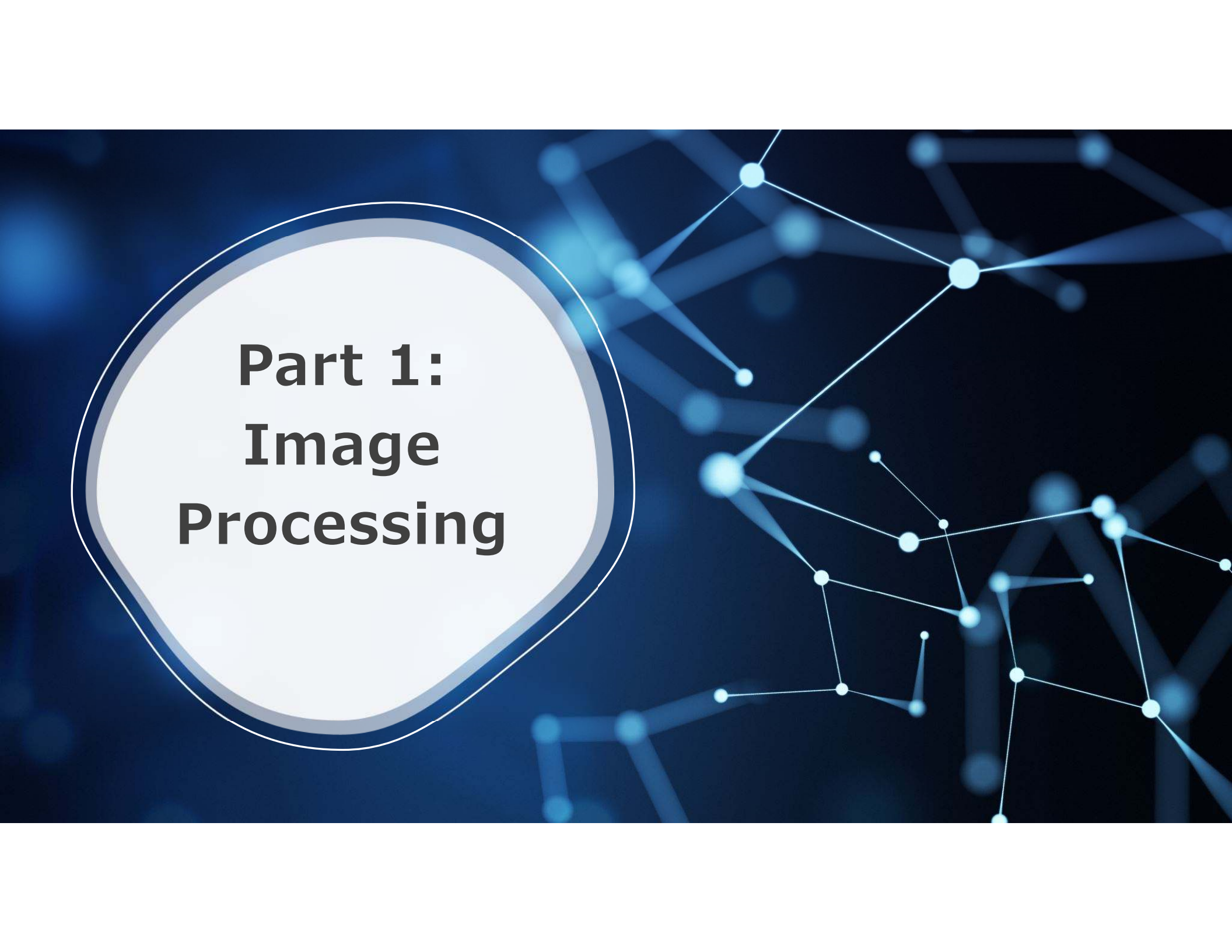
HCIA-AI Huawei

Lecture 8

Image Processing and CNN

Prepared by
Eng. Eman Elgallad

www.midocean.ae



Part 1: Image Processing

Lecture

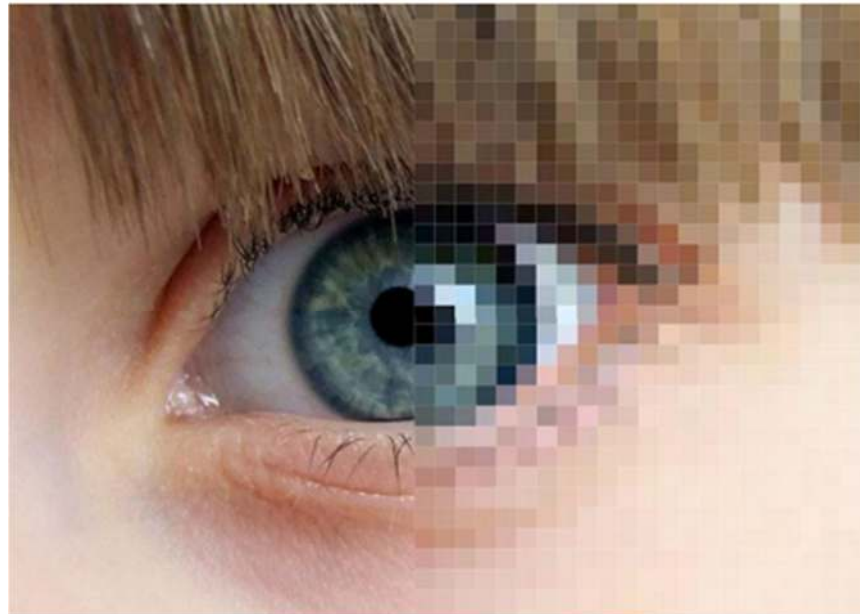
08

Image Processing

Image processing in Python is a rapidly growing field with a wide range of applications. It is used in a variety of industries, including Computer vision, medical imaging, security, etc. Image processing is the field of study and application that deals with modifying and analyzing digital images using computer algorithms.

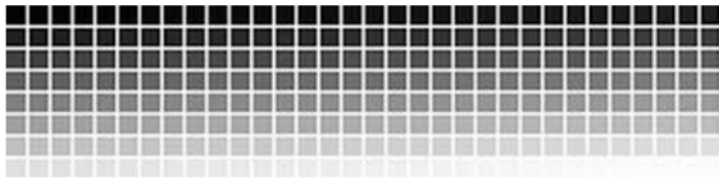
What is an image

An arrays of smaller cells (pixels) that, altogether, convey a visual information
An image is made of a set of pixels, Pixels are the building blocks of an image
➤ A pixel demonstrates the “intensity” of light or “color”

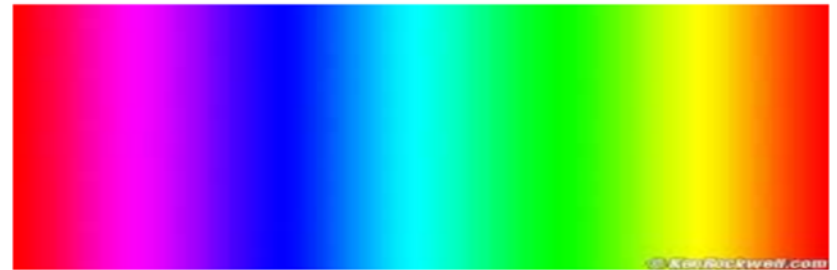


Pixel Representation:

Grey Scale:



Colored :

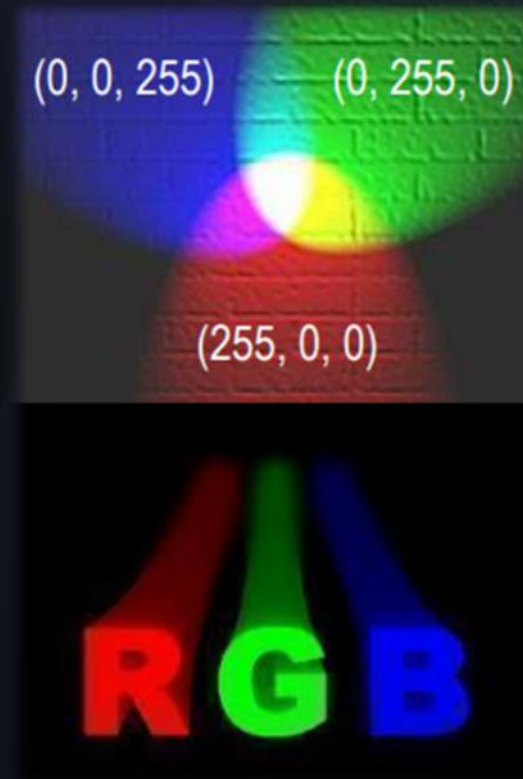


Lecture
08

Colored Images:

The amount of Red, Green, and Blue contained in a single pixel.

- A color is ranked (from 0 to 255) to carry its intensity.
- We can use 8-bit unsigned integers to represent the intensity
- A tuple can represent a color that is a mixed of three main colors: Red, Green, and Blue.
- To create a specific color, a bucket of (R, G, B) should be filled.



Colored Images Cont.

A color image is just three functions pasted together. We can write this as a “vector valued” function:

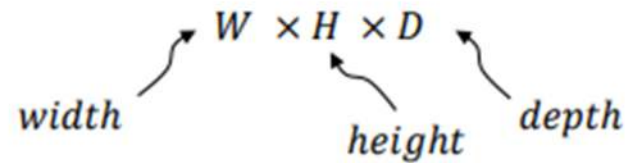
$$I(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

a color is ranked (from 0 to 255) to carry its intensity.



Lecture
08

Each channel is represented by three values;



an independent matrix, representing a major color
(Red, Green, or Blue)

Combination of matrices, forms an image

```
$ python load_display.py  
(248, 300, 3)
```

$H \times W \times D$

height width depth

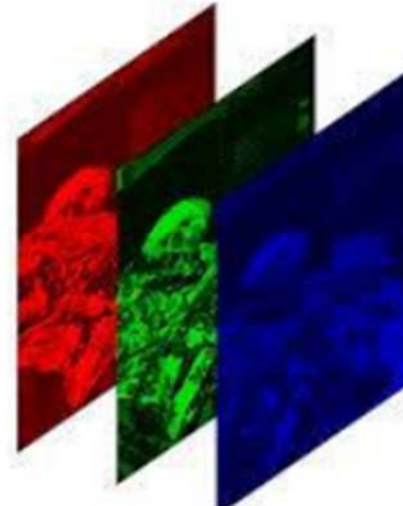


Image Formatting



Open Cv:

While taking photographs is as simple as pressing a button, processing and improving those images sometimes takes more than a few lines of code. That's where image processing libraries like OpenCV come into play. OpenCV is a popular open-source package that covers a wide range of image processing and computer vision capabilities and methods.

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

To install it in your colab type this line: `!pip install opencv-python`

Open Cv Cont.

- OpenCV stores RGB channels in reverse order:
Blue, Green, Red order, as it uses a different convention for the order of the different color channels.
- Most other tools and formats are in RGB order.
- When saving from OpenCV to standard image formats, OpenCV does the conversion automatically. However, for some other tools, you need to convert from BGR to RGB when going from OpenCV to those tools (and vice versa)

Digital Image Processing:

Low-level operations

- Levels, contrast, enhancement via point operations
- Filters e.g., smoothing, sharpening
- Output is a new image



www.midocean.ae

Digital Image Processing Cont. :

Mid-level operations – starting to understand

- Lines, regions, objects, motion
- Visual feature geometry
- Output is image attributes (e.g., edges)



Digital Image Processing Cont. :

- High-level operations – greater understanding
 - Image understanding, captioning
 - Facial recognition, object detection and classification
 - Output is complex image attributes



Most commonly uses for data augmentation task which training the model in computer Vision:

- Image Resizing
- Image Rotation
- Image Translation
- Image Shearing
- Image Normalization
- Edge detection in an Image
- Image Blurring

Basic Augmentation Transforms:

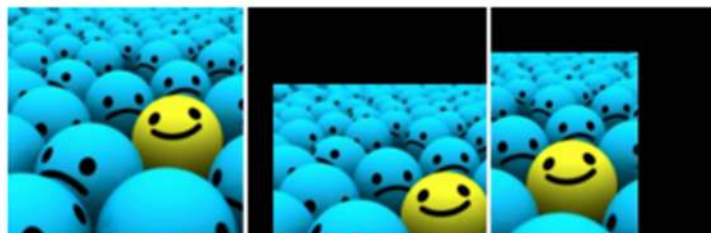
Flip

$$J(x, y) = I(-x, y)$$



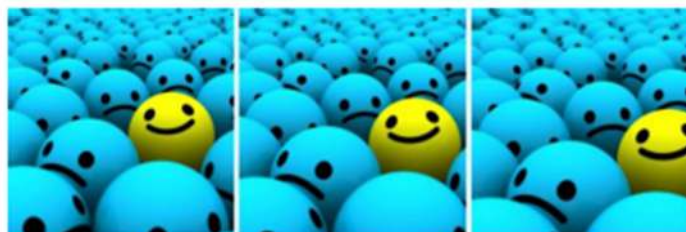
Translation

$$J(x, y) = I(x + u, y + v)$$



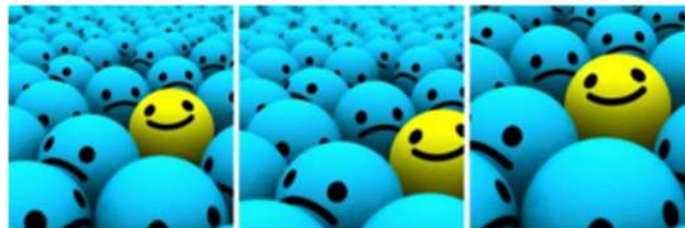
Scale

$$J(x, y) = I(ux, vx)$$

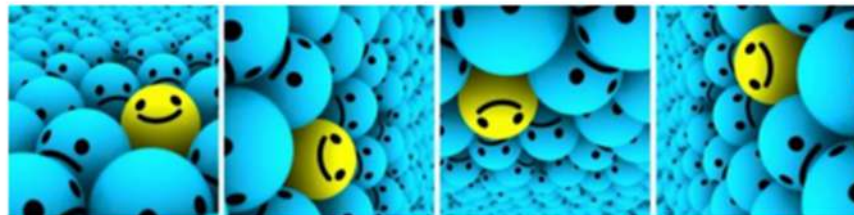


Basic Augmentation Transforms Cont.:

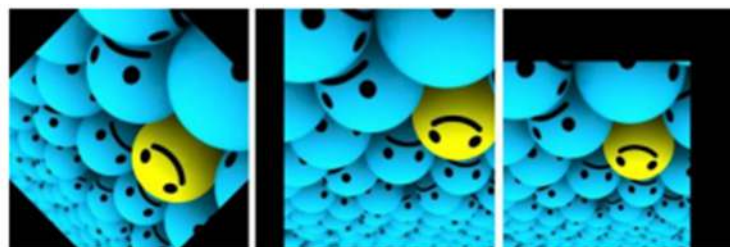
Crop (+ resize)



Rotation



Arbitrary



Practical Part for Image Processing

Lecture

08

https://colab.research.google.com/drive/1cUsOz_iSZjquKLm966KmkEQoPJltDzlg?usp=sharing

Part 2: CNN



CNN:

Definition of CNNs: Convolutional Neural Networks (CNNs) are a class of deep learning models designed for processing structured grid data, such as images and videos. They are particularly powerful in computer vision tasks due to their ability to automatically learn and extract features from visual data.

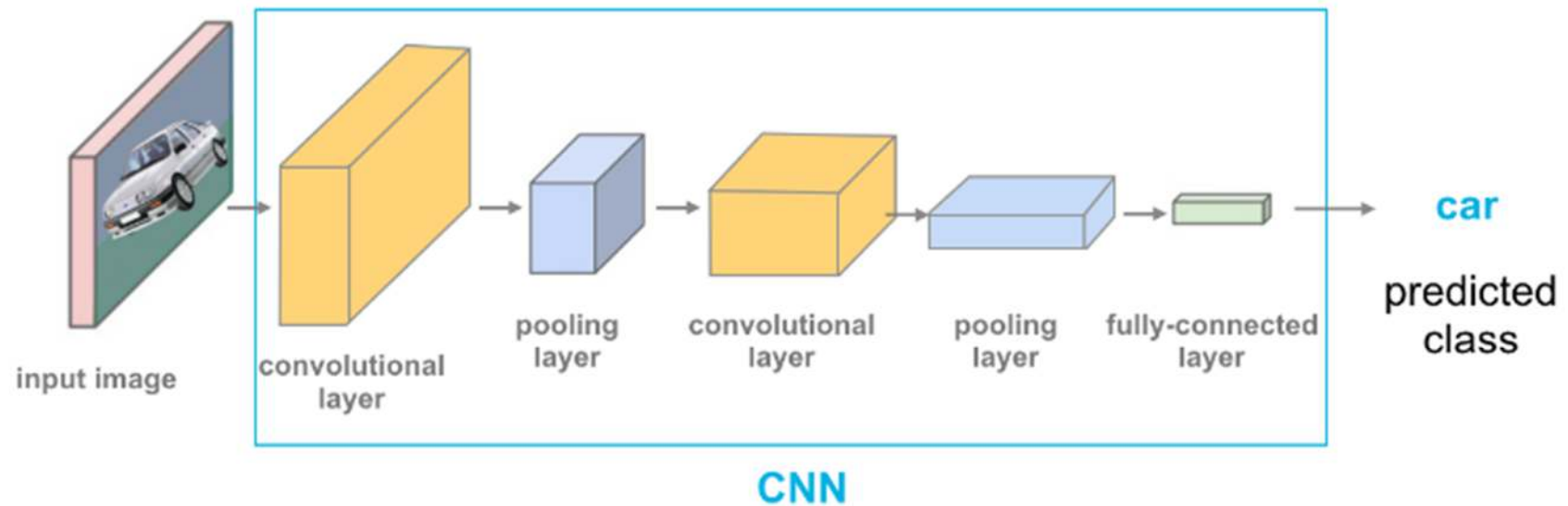
Historical Background: CNNs have a rich history. Mention seminal models like LeNet, introduced by Yann LeCun in the early 1990s, which was one of the first successful CNNs for handwritten digit recognition. Highlight the breakthrough made by AlexNet in 2012, which significantly boosted the popularity of CNNs by winning the ImageNet competition.

Motivation: Explain why CNNs are essential for computer vision tasks. Emphasize their ability to automatically learn hierarchical features, making them suitable for tasks like image classification, object detection, and more.

Basic Components of CNN:

- **Neurons and Layers:** CNNs consist of layers of interconnected neurons. Neurons in one layer are connected to those in adjacent layers, forming a network. Layers can be categorized into input, hidden, and output layers.
- **Convolutional Layers:** These layers are the core building blocks of CNNs. They apply convolution operations to input data using learnable **filters** or **kernels**. Convolutional layers capture local patterns and features.
- **Activation Functions (ReLU):** Activation functions introduce non-linearity to the model. Rectified Linear Unit (ReLU) is a commonly used activation function in CNNs, which helps the network model complex relationships in data.
- **Pooling Layers (Max Pooling):** Pooling layers reduce the spatial dimensions of feature maps, helping control the computational complexity of the network. Max pooling selects the maximum value within a region.
- **Fully Connected Layers:** After convolution and pooling, fully connected layers are used for classification. These layers flatten the data and connect every neuron in one layer to every neuron in the next layer.

Basic Components of CNN:



Filters and Kernels:

- Filters or kernels are small matrices that play a crucial role in convolution. These filters are the heart of feature extraction in CNNs.
- Filters are typically initialized with random values and then updated during the training process through backpropagation.
- Each filter specializes in recognizing a particular feature or pattern. For example, one filter might be designed to detect horizontal edges, while another might focus on diagonal edges or textures.
- CNNs use multiple filters in each convolutional layer to capture various features simultaneously, allowing the network to learn complex patterns.

We Can say that: A filter or kernel is a small matrix (e.g., 3x3 or 5x5) that contains learnable parameters. These parameters are adjusted during training to capture different features in the input data. Each filter is responsible for detecting a specific pattern or feature, such as edges, textures, or shapes.

Strides and Padding:

- Stride:** Stride refers to the step size at which the filter moves during convolution. A stride of 1 means the filter moves one pixel at a time, while a larger stride skips pixels. Smaller strides result in larger feature maps with more detailed information, while larger strides reduce the size of the feature map. **In summary** A stride of 1 moves the filter one pixel at a time, resulting in dense feature maps with minimal information loss. A larger stride (e.g., 2) skips pixels, reducing the size of the feature map.

- Padding:** Padding involves adding extra pixels to the input data before convolution. Padding can be "valid" (no padding), "same" (padding to preserve the spatial dimensions), or custom. Padding helps control the size of the output feature map.

Padding is essential to maintain spatial information and ensure that information at the edges of the input is considered during convolution. Without padding, the output feature map is smaller than the input, potentially losing valuable information.

Steps for Convolution layer

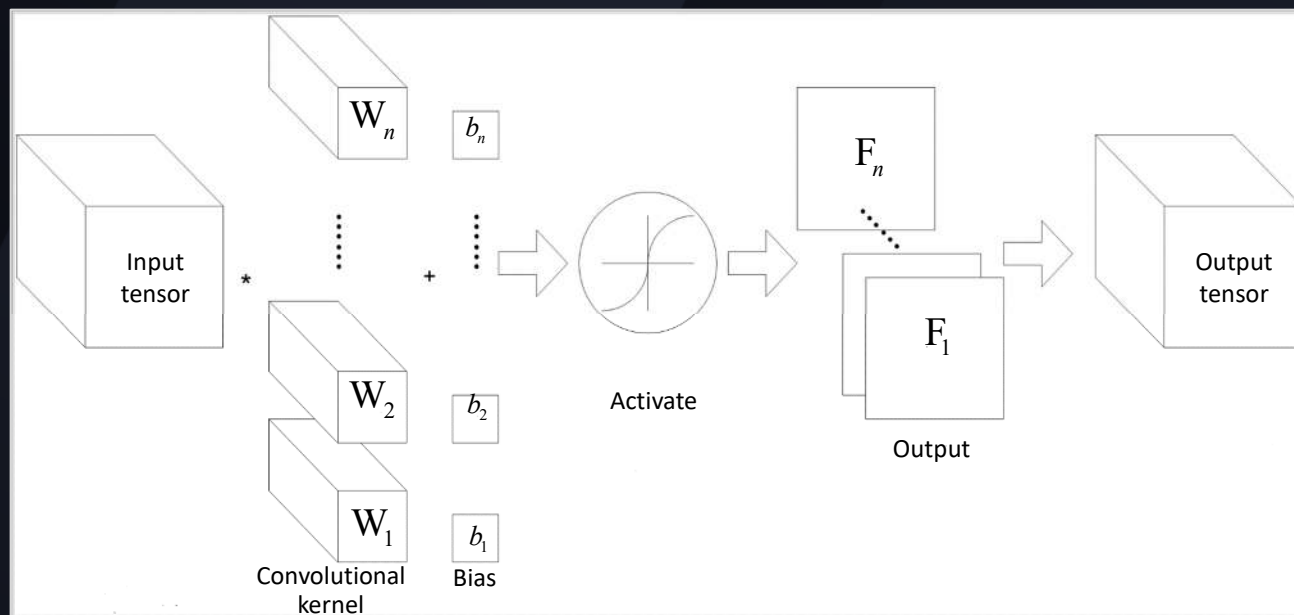
1.Sliding Window: The filter is systematically moved (convolved) across the input data. It starts from the top-left corner and slides horizontally and vertically, one step at a time. At each position, a dot product operation is performed between the filter and the overlapping region of the input data.

1.Dot Product: At each position of the sliding window, a dot product is computed between the elements of the filter and the corresponding elements of the input data. This dot product results in a single value, which is placed in the corresponding position of the feature map.

1.Feature Map: As the filter moves across the entire input data, it produces a feature map. The feature map highlights regions where specific patterns detected by the filter are present. Each value in the feature map represents the response of the filter to a local pattern in the input data.

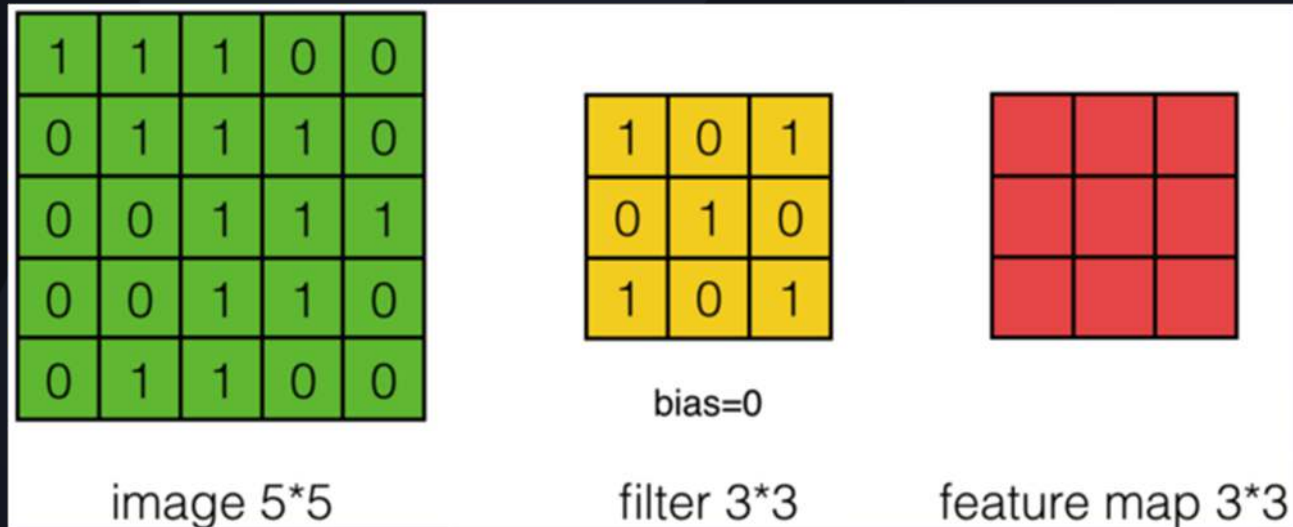
Convolutional Layer

- The basic architecture of a CNN is multi-channel convolution consisting of multiple single convolutions. The output of the previous layer (or the original image of the first layer) is used as the input of the current layer. It is then convolved with the filter in the layer and serves as the output of this layer. The convolution kernel of each layer is the weight to be learned. Similar to FCN, after the convolution is complete, the result should be biased and activated through activation functions before being input to the next layer.



Single-Filter Calculation (1)

- Description of convolution calculation



Single-Filter Calculation (2)

- Demonstration of the convolution calculation

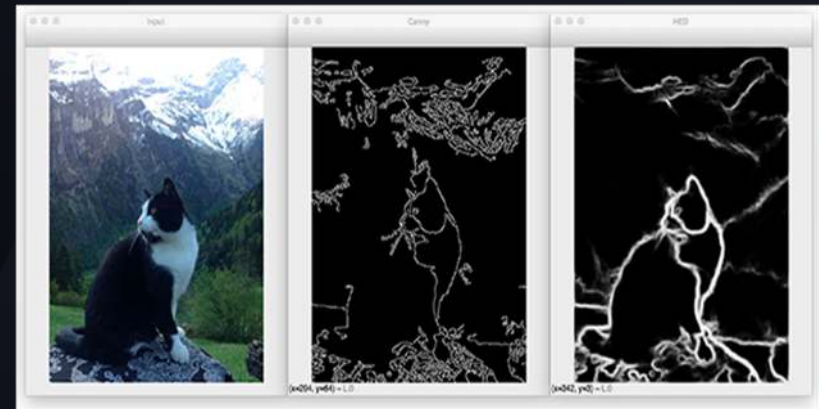
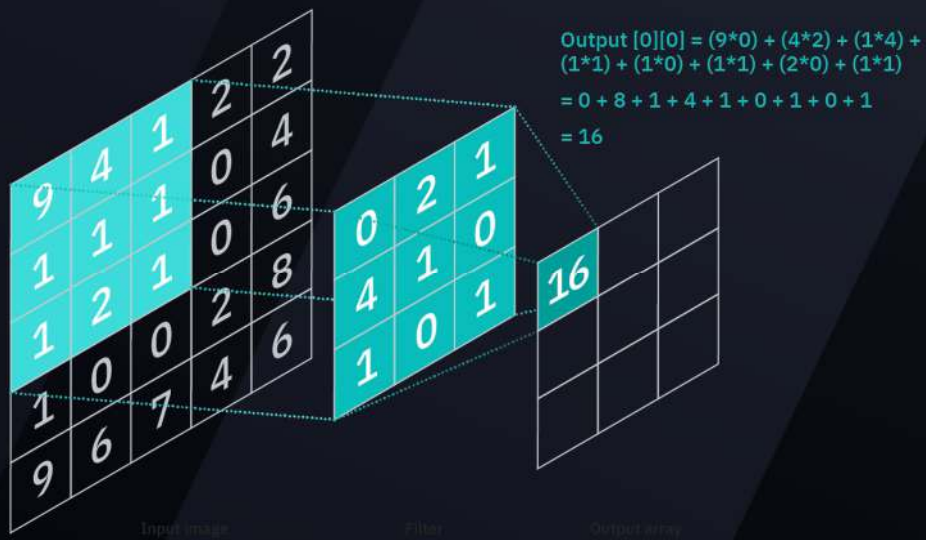
1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

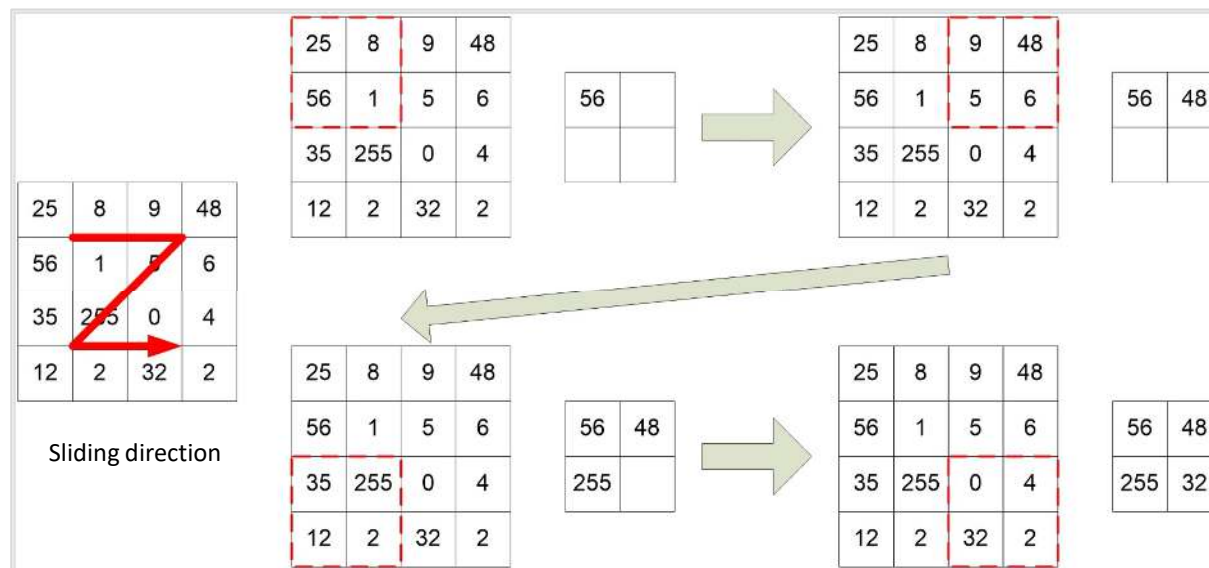
Convolved
Feature

CNN LAYERS – Convolutional Layer



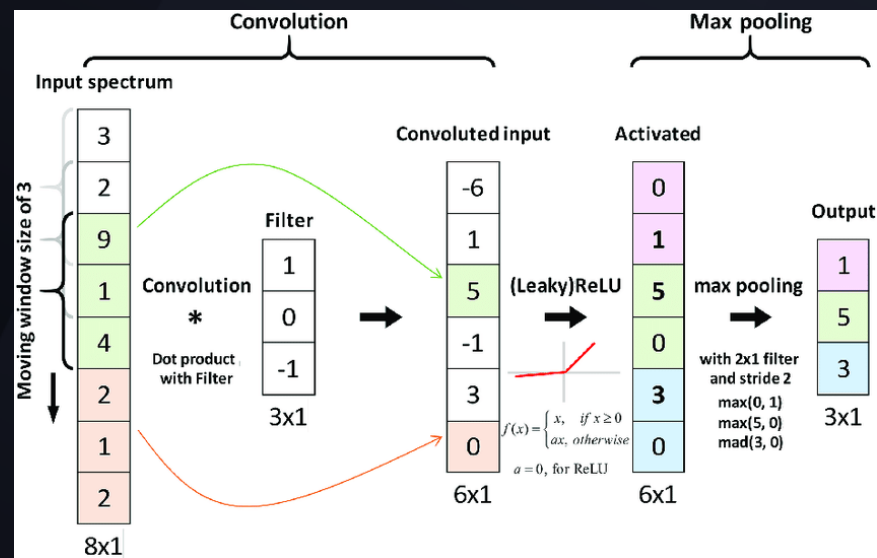
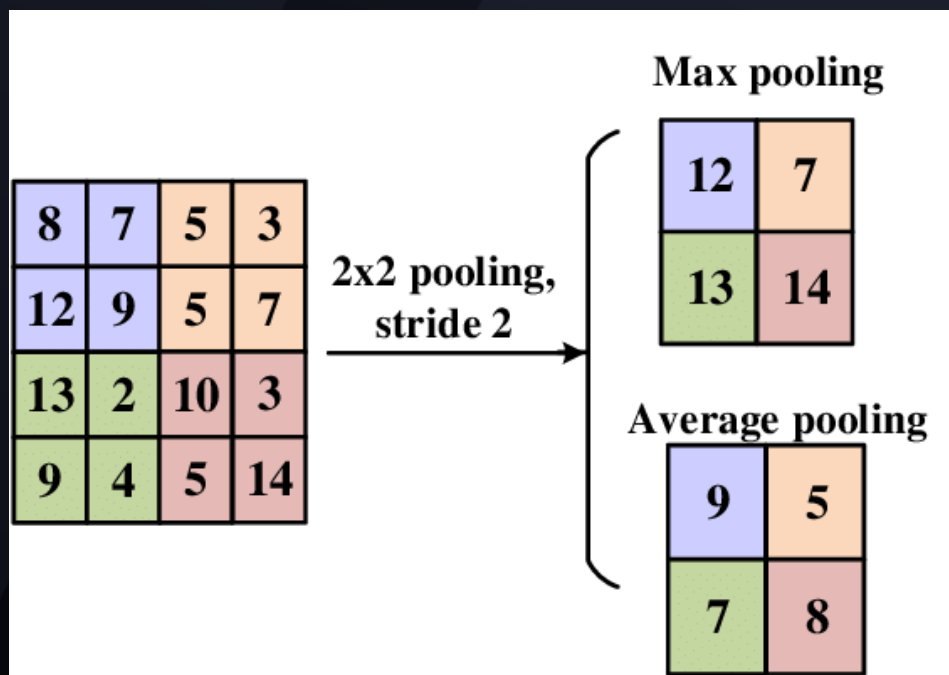
Pooling Layer

- Pooling combines nearby units to reduce the size of the input on the next layer, reducing dimensions. Common pooling includes max pooling and average pooling. When max pooling is used, the maximum value in a small square area is selected as the representative of this area, while the mean value is selected as the representative when average pooling is used. The side of this small area is the pool window size. The following figure shows the max pooling operation whose pooling window size is 2.



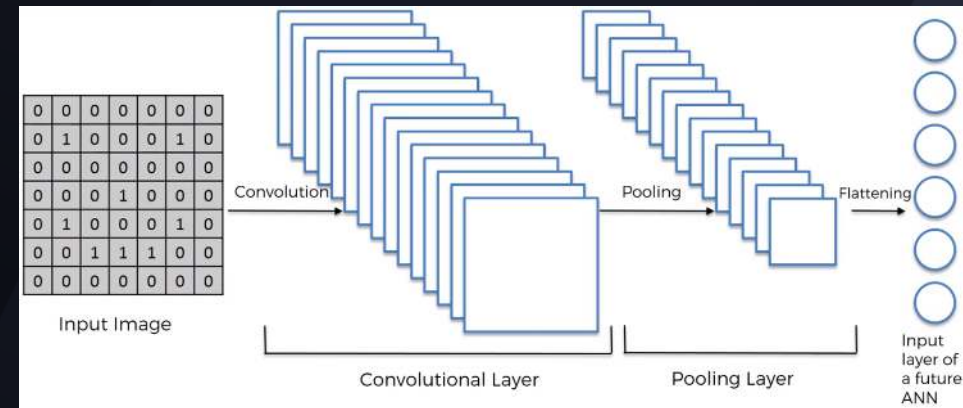
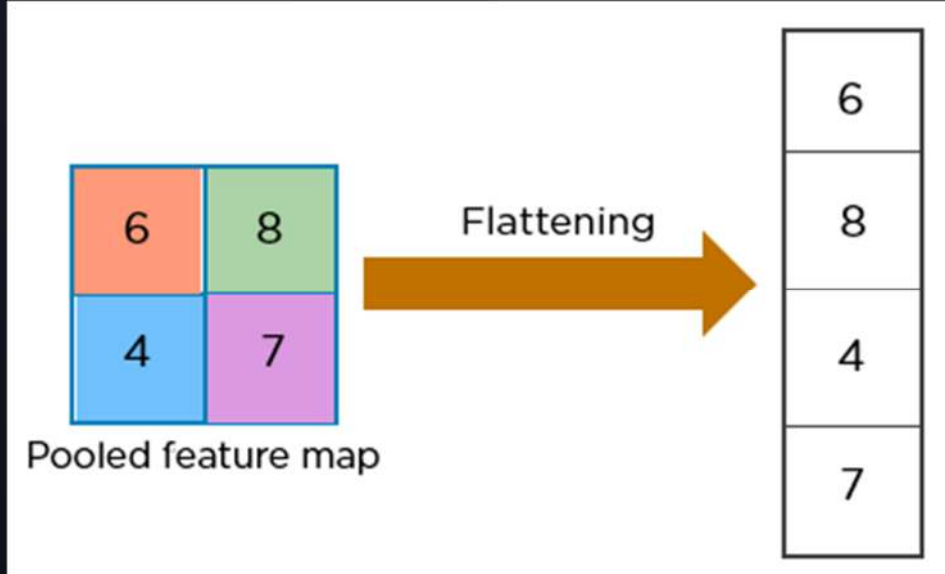
Lecture
08

CNN LAYERS – Pool Layer



Lecture
08

CNN LAYERS – Flattening and fully Connected Layer



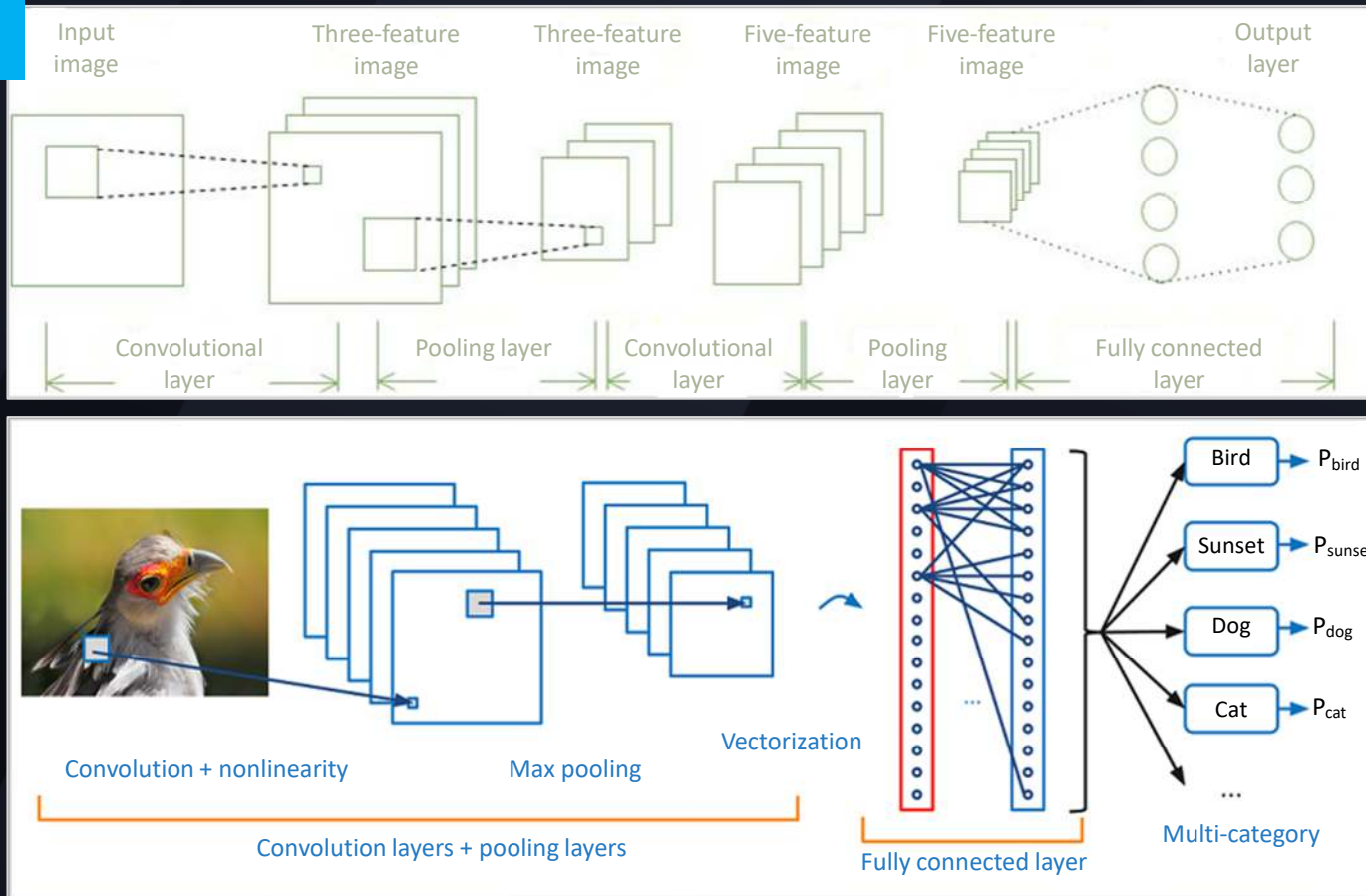
Fully Connected Layer

- The fully connected layer is essentially a classifier. The features extracted on the convolutional layer and pooling layer are straightened and placed at the fully connected layer to output and classify results.
- Generally, the SoftMax function is used as the activation function of the final fully connected output layer to combine all local features into global features and calculate the score of each type.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Lecture 08

Full Architecture of Convolutional Neural Network



https://adamharley.com/nn_vis/cnn/3d.html

Challenges and Future Trends:

- **Overfitting and Ways to Combat It.**
- **Computational Resources**
- **Current and Future Trends in CNN Research Such as
attention mechanisms and Capsule Networks.**

Overfitting in CNN:

• **Overfitting:** Overfitting occurs when a CNN learns to perform exceptionally well on the training data but fails to generalize to unseen data. It happens when the model captures noise or random variations in the training data instead of true patterns.

• **Techniques to Combat Overfitting:**

- **Dropout:** Dropout is a regularization technique where random neurons are temporarily dropped out (ignored) during training. This prevents the network from relying too heavily on any single neuron and encourages it to learn more robust features.
- **Regularization:** Techniques like L1 and L2 regularization add penalty terms to the loss function, discouraging large weight values and promoting simpler models.
- **Data Augmentation:** Generating additional training data through techniques like rotation, flipping, and scaling can help reduce overfitting by increasing the diversity of the training set.
- **Early Stopping:** Monitoring the model's performance on a validation dataset and stopping training when performance starts to degrade can prevent overfitting.

Computational Resources:

- **Training Deep CNNs:** Training deep CNNs with many layers and parameters can be computationally intensive and time-consuming. This is especially challenging for individuals or organizations with limited computational resources.
- **Solutions:**
 - **GPU Acceleration:** Graphics Processing Units (GPUs) significantly speed up the training of CNNs by parallelizing operations. Many deep learning frameworks, including TensorFlow and PyTorch, support GPU acceleration.
 - **Cloud Computing:** Cloud platforms like AWS, Google Cloud, and Microsoft Azure offer GPU instances for deep learning tasks, making it accessible to a broader audience.
 - **Model Compression:** Techniques like model pruning, quantization, and knowledge distillation reduce the size and computational requirements of deep CNNs without sacrificing too much accuracy.

Future Trends :

- **Attention Mechanisms:** Attention mechanisms, initially popularized in natural language processing, are gaining traction in CNNs. They enable models to focus on relevant parts of the input, improving both performance and interpretability.
- **Capsule Networks:** Capsule Networks, or CapsNets, are an emerging architecture that aims to overcome limitations in traditional CNNs, particularly in recognizing spatial hierarchies of features.
- **Self-Supervised Learning:** Self-supervised learning techniques, where models learn from unlabeled data, are becoming more prevalent and have the potential to reduce the need for large labeled datasets.
- **Transfer Learning Advancements:** Researchers are continually improving transfer learning techniques, making it easier to adapt pre-trained models to specific tasks with limited labeled data.

Transfer Learning and CNN:

Transfer learning is a powerful concept in the realm of Convolutional Neural Networks (CNNs) and deep learning. It involves taking a pre-trained CNN model and fine-tuning it for a specific task or dataset. Transfer learning is particularly valuable because it reduces the need for extensive training data and allows us to leverage the knowledge learned from one domain to improve performance in another. Here's a detailed explanation of transfer learning:

1. Pre-trained Models:

Transfer learning begins with a pre-trained CNN model that has been trained on a massive and diverse dataset, typically for a generic task like image classification. Popular pre-trained models include VGG, ResNet, Inception, and MobileNet, among others.

2. Leveraging Generalization:

These pre-trained models have learned to recognize a wide range of features, patterns, and object categories from the vast dataset they were trained on. They've developed generalization capabilities, meaning they can identify generic features like edges, textures, and shapes.

Transfer Learning and CNN Cont.:

3. Specific Fine-tuning:

When you have a specific task or dataset for which you want to train a CNN, you don't necessarily need to start from scratch. Instead, you can leverage the knowledge encoded in the pre-trained model.

4. Fine-tuning Steps:

The process of transfer learning typically involves a few key steps:

- **Initial Layers:** You may freeze (not update) the initial layers of the pre-trained model. These layers capture low-level features that are universally useful and don't need significant changes.
- **Custom Layers:** You add one or more custom layers on top of the pre-trained model. These custom layers are often used for task-specific feature extraction and decision making.
- **Fine-tuning:** You then train the entire network, including the custom layers, using your specific dataset. During this process, the model learns to adapt to the unique features and patterns present in your data while retaining the general knowledge from the pre-trained layers.

Transfer Learning and CNN Cont.:

5. Advantages of Transfer Learning:

- **Reduced Data Requirement:** One of the most significant advantages of transfer learning is that it reduces the amount of labeled training data needed for your specific task. You can achieve good results with a smaller dataset.
- **Faster Training:** Training from scratch can be computationally expensive and time-consuming. Transfer learning is often faster because you're building upon an already well-optimized model.
- **Improved Generalization:** Transfer learning helps the model generalize better to new, unseen data because it starts with a foundation of learned features.

6. Choosing the Right Pre-trained Model:

Selecting the appropriate pre-trained model depends on your task and dataset. Some models may be better suited for certain tasks due to their architectures or the types of data they were originally trained on.

Applications of CNNs:

- **Image Classification:** Explain how CNNs are used for classifying images into predefined categories.
- **Object Detection:** Discuss how CNNs can identify and locate objects within an image, a critical component in tasks like autonomous driving and surveillance.
- **Image Segmentation:** Describe image segmentation, where CNNs partition an image into meaningful segments or regions.

Applications of CNNs Cont.:

Classification



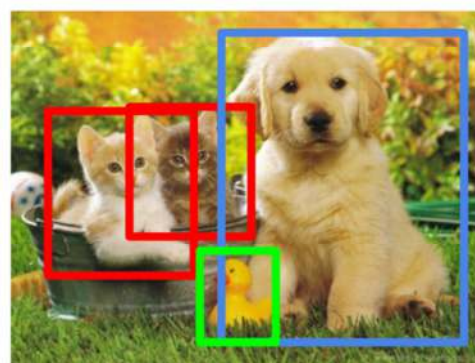
CAT

**Classification
+ Localization**



CAT

Object Detection



CAT, DOG, DUCK

**Instance
Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

Applications of CNNs Cont.:

- **Face Recognition:** Mention CNN's role in facial recognition applications, including security and authentication.
- **Autonomous Vehicles:** Explain how CNNs enable perception tasks for self-driving cars, such as detecting pedestrians and traffic signs.
- **Medical Imaging:** Discuss the use of CNNs in medical image analysis for diagnosis and treatment planning.



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY



Thank You

www.midocean.ae