1. Basic Components of a Digital Image and Representation in a Computer

A **digital image** is composed of a grid of **pixels** (picture elements), where each pixel represents a small portion of the image. The basic components of a digital image include:

- **Pixels**: The smallest unit of a digital image, each pixel contains color or intensity information.
- **Resolution**: The total number of pixels in an image, typically expressed as width × height (e.g., 1920 × 1080).
- **Channels**: The number of color components used to represent a pixel. For example:
  - **Grayscale images**: Have 1 channel, representing intensity (0 = black, 255 = white).
  - **Color images**: Typically have 3 channels (Red, Green, Blue, or RGB), each representing the intensity of a specific color.

Representation in a Computer:
- A digital image is stored as a matrix of pixel values. For grayscale images, this is a 2D matrix, while for color images, it is a 3D matrix (height × width × channels).
- Each pixel value is represented as a number. In an 8-bit image, pixel values range from 0 to 255.

Differences Between Grayscale and Color Images:
- **Grayscale images**: Use a single channel to represent intensity, resulting in shades of gray.
- **Color images**: Use multiple channels (e.g., RGB) to represent color, allowing for a wide range of colors.

---

2. Convolutional Neural Networks (CNNs) and Their Role in Image Processing

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed for processing grid-like data, such as images. They are particularly effective for image-related tasks due to their ability to automatically learn spatial hierarchies of features.

Role in Image Processing:
- CNNs extract features from images through convolutional layers, which detect patterns like edges, textures, and shapes.
- They are widely used in tasks such as image classification, object detection, segmentation, and more.

Advantages of CNNs Over Traditional Neural Networks:
1. **Local Feature Extraction**: CNNs use convolutional filters to focus on local regions of an image, capturing spatial patterns.
2. **Parameter Sharing**: Convolutional layers share weights across the image, reducing the number of parameters and computational complexity.

3. **Translation Invariance**: CNNs can recognize patterns regardless of their position in the image.
4. **Hierarchical Learning**: CNNs learn low-level features (e.g., edges) in early layers and high-level features (e.g., objects) in deeper layers.

---

3. Convolutional Layers, Filters, Padding, and Strides

Convolutional Layers:
- **Purpose**: Convolutional layers apply filters (kernels) to the input image to extract features. They are the core building blocks of CNNs.
- **Filters (Kernels)**: Small matrices (e.g., 3×3 or 5×5) that slide over the input image to perform element-wise multiplication and summation, producing a feature map.
- **Convolution Operation**: The filter moves across the image, computing dot products between the filter and local regions of the input.

Padding:
- **Purpose**: Padding adds extra pixels around the input image to control the size of the output feature map.
- **Types**:
  - **Valid Padding**: No padding is applied, resulting in a smaller output size.
  - **Same Padding**: Padding is added to ensure the output size matches the input size.

Strides:
- **Purpose**: Strides determine the step size by which the filter moves across the image.
- **Impact**: Larger strides reduce the output size, while smaller strides preserve more spatial information.

Output Size Calculation:

The output size of a convolutional layer can be calculated using:

$$\text{Output Size} = \frac{\text{Input Size} - \text{Filter Size} + 2 \times \text{Padding}}{\text{Stride}} + 1$$

---

4. Pooling Layers in CNNs

Purpose of Pooling Layers:
- Pooling layers reduce the spatial dimensions of the feature maps, decreasing computational complexity and controlling overfitting.

- They also introduce translation invariance, making the network less sensitive to small shifts in the input.

Types of Pooling:
1. **Max Pooling**:
   - Selects the maximum value from a region of the feature map.
   - Preserves the most prominent features, such as edges or textures.
   - Commonly used in CNNs.

2. **Average Pooling**:
   - Computes the average value from a region of the feature map.
   - Smooths out the feature map, reducing the impact of outliers.

Comparison:
- **Max Pooling**: More effective for preserving sharp features and edges.
- **Average Pooling**: Useful for smoothing and reducing noise.

Pooling layers typically use a 2×2 filter with a stride of 2, reducing the feature map size by half.