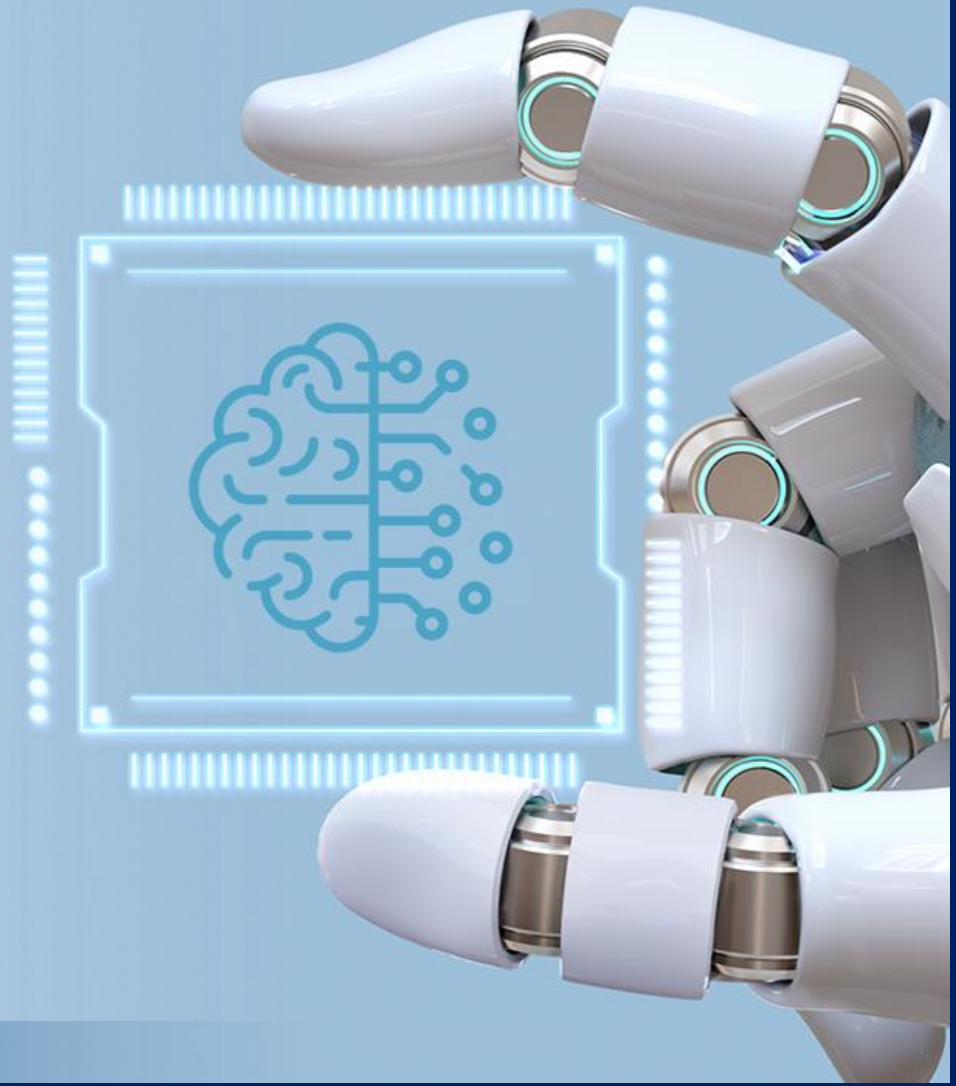
edunet
foundation

Common Deep Learning Architectures



Agenda

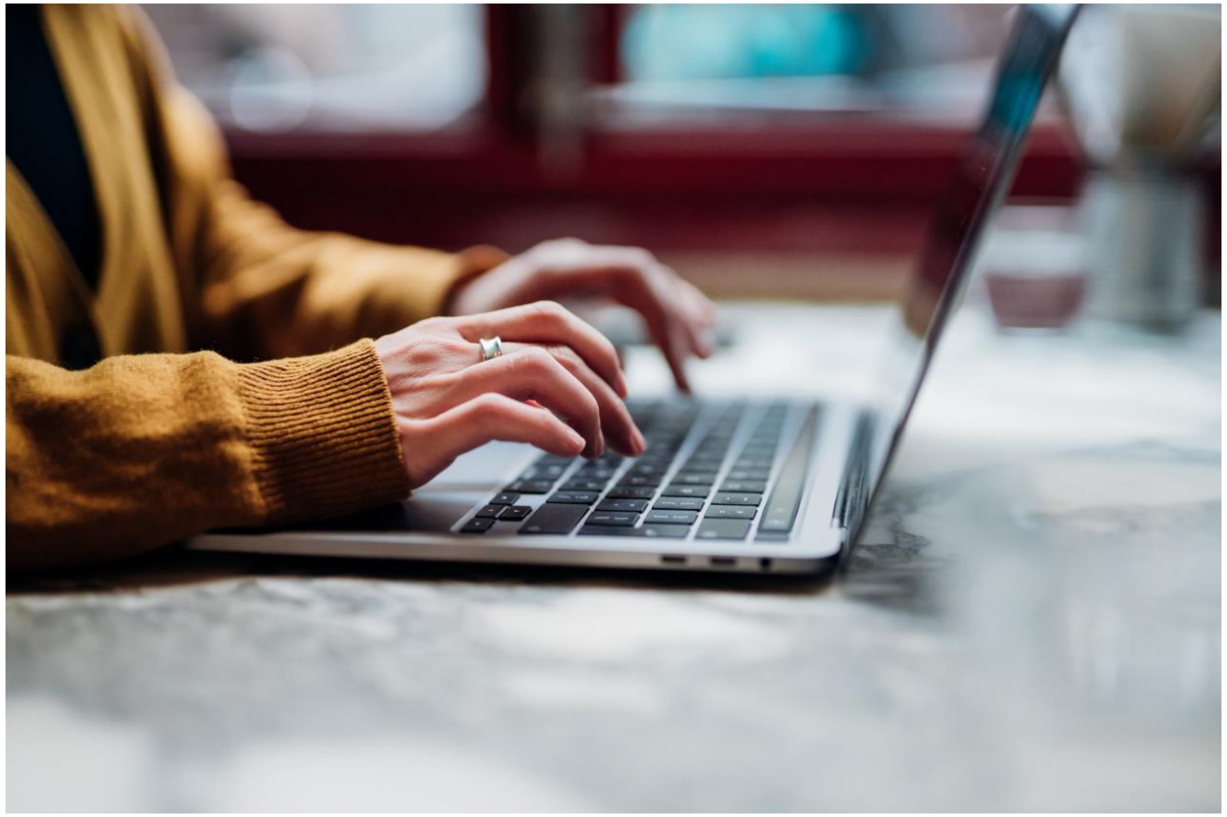
- Convolutional Neural Networks (CNNs)
- Convolutional layers, Pooling layers, Fully connected layers
- CNNs and their applications in computer vision
- Implementation of Basic CNN model

Convolutional Neural Networks (CNNs)

- Convolutional Neural Networks (CNNs) are a class of deep neural networks that are primarily used for analyzing visual imagery. They are designed to automatically and adaptively learn spatial hierarchies of features from the input data. CNNs have become the go-to model for various tasks in computer vision, such as image classification, object detection, segmentation, and more.
- Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

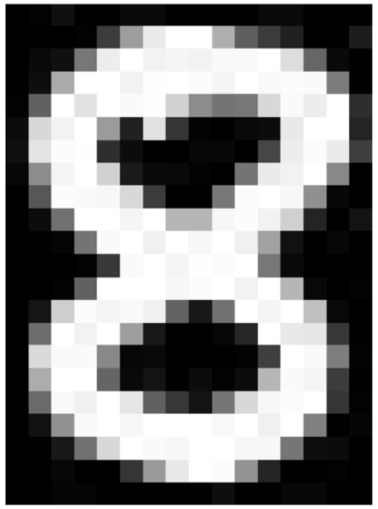
What is Computer Vision(CV)?

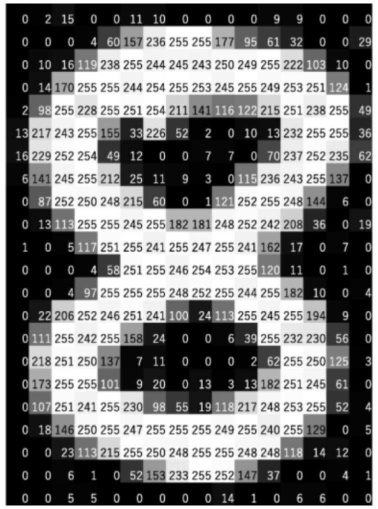
- Computer vision (CV) is an artificial intelligence (AI) subcategory that focuses on developing and deploying digital systems that process, analyze, and interpret visual input.
- The objective of computer vision is to allow computers to recognize an item or person in a digital image and take appropriate action.
- Convolutional neural networks (CNNs) are used in computer vision to analyze visual input at the pixel level.

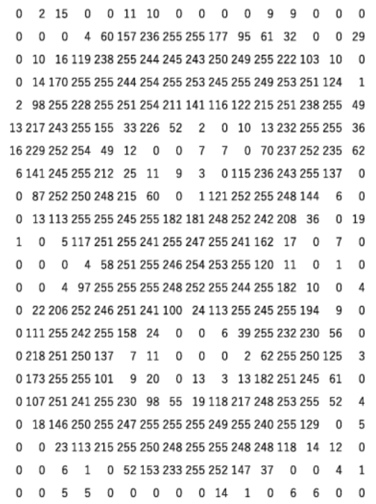


What is an Image?

- An image is a grid of tiny dots, where each dot, or pixel, represents a single point of color or brightness. These pixels are arranged in rows and columns, forming a mosaic that creates the visual representation of the image. The quality and detail of the image are determined by the number of pixels it contains, often referred to as the image's resolution.
- For example, a digital photograph taken with a smartphone might have a resolution of 3000 pixels wide by 2000 pixels high. This means the image is made up of 6 million pixels in total. Each pixel contains information about its color, usually represented as a combination of red, green, and blue (RGB) values.







0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Source: https://medium.com/lifeandtech/tagged/image-processing

Types of Images

There are three types of images RGB, Greyscale and Binary.

RGB Image

- RGB stands for Red, Green, and Blue, the primary colors of light.
- In an RGB image, each pixel is represented by three color channels: red, green, and blue.
- The intensity of each channel determines the color of the pixel, and these intensities are usually represented by values ranging from 0 to 255.
- When combined, these three channels create a wide range of colors, allowing for the representation of full-color images.
- Each pixel in an RGB image has a color depth of 24 bits, with 8 bits allocated to each color channel.

Source: https://www.pinterest.com/pin/333970128601973858/

Types of Images

Grayscale Image

- In a grayscale image, each pixel is represented by a single channel corresponding to its brightness.
- The intensity of this single channel varies from black to white, usually represented by values ranging from 0 to 255.
- Each pixel in a grayscale image has a color depth of 8 bits.

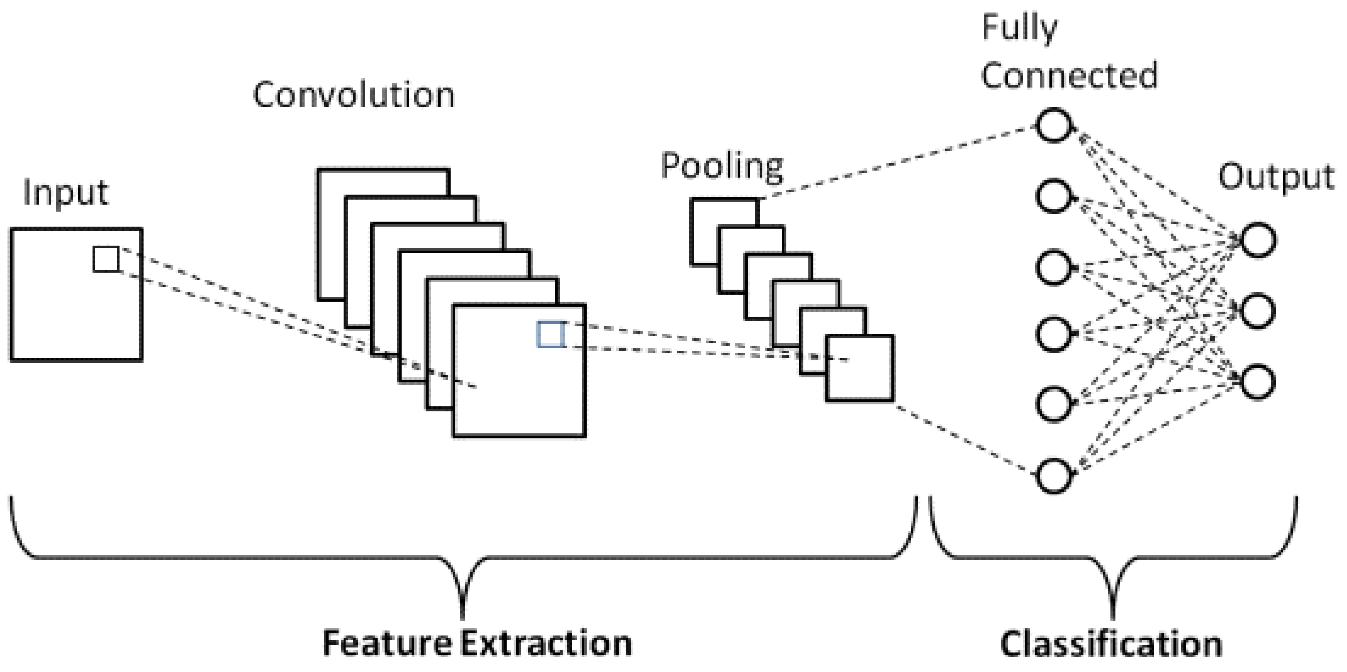
Binary Image

- In a binary image, each pixel is represented by only two values: black and white (or 0 and 1).
- Each pixel in a binary image requires only one bit to represent its value: 0 for black and 1 for white.

Source:

Components of CNN

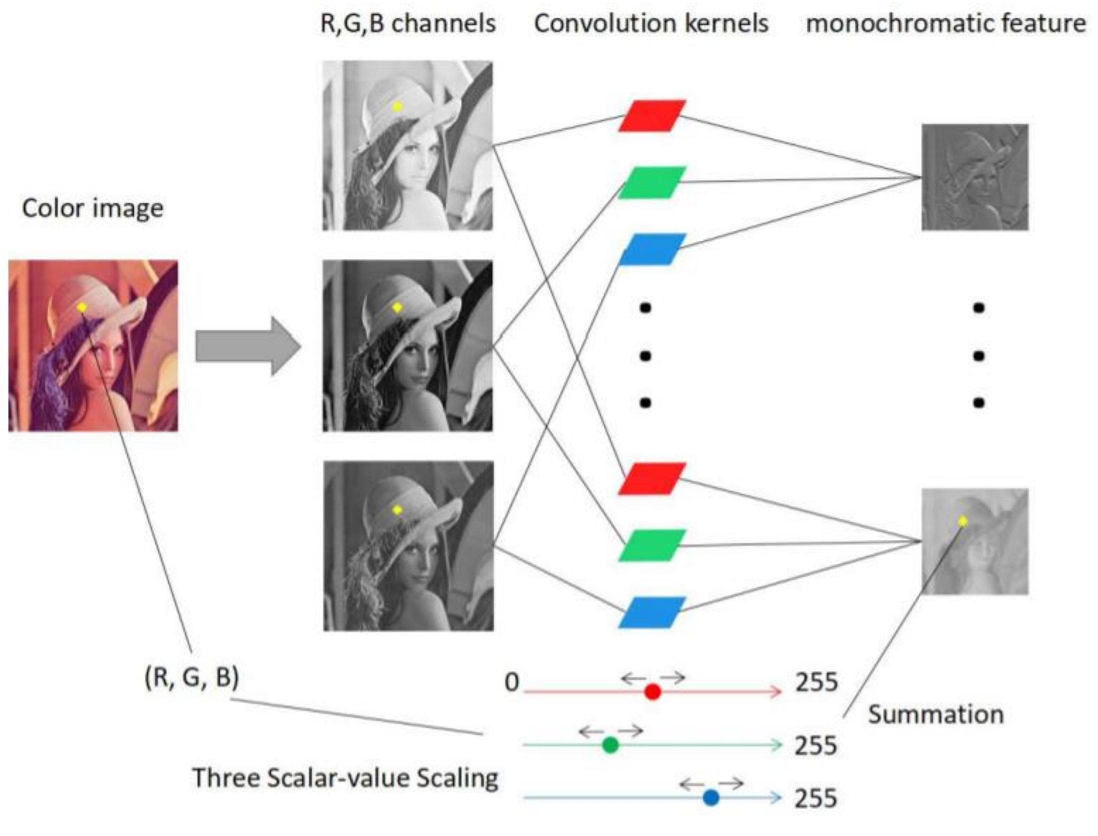
- Convolutional Layers:
- Activation Functions
- Pooling Layers
- Fully Connected Layers
- Flattening
- Dropout
- Batch Normalization
- Padding
- Strides
- Loss Functions
- Optimization Algorithms



Source: https://www.mdpi.com/2076-3417/13/1/131

Convolutional Layers

- These layers consist of a set of filters (also called kernels) that move across the input data (image) and perform convolution operations. Each filter extracts certain features from the input. Convolutional layers help the network learn hierarchical representations of the input data.
- Convolutional layers are the fundamental building blocks of Convolutional Neural Networks (CNNs). They play a crucial role in extracting features from the input data, typically images, by applying a series of learnable filters to the input.
- Each filter performs a convolution operation across the input data, producing feature maps that capture various patterns or features.



(a) Real-valued CNN

Source: https://stats.stackexchange.com/questions/534080/how-convolutional-layer-work-exaclty-in-rgb-image-processing

Components And Operations Within Convolutional Layers

- **Filters (Kernels):** A filter is a small matrix of weights that is convolved with the input data. Each filter detects specific patterns or features, such as edges, textures, or more complex structures. For example, one filter might detect horizontal edges, while another might detect diagonal edges.
- **Convolution Operation:** The convolution operation involves sliding the filter over the input data and computing the dot product between the filter and the overlapping region of the input at each position. This produces a single value in the output feature map, capturing the presence of the pattern corresponding to the filter at that location.
- **Feature Maps:** Each filter applied to the input produces a feature map, which is essentially a 2D matrix representing the activations of that filter across the spatial dimensions of the input. Multiple filters are used in a convolutional layer, generating multiple feature maps. These maps collectively capture different aspects of the input data.
- **Padding:** Padding is often applied to the input data before convolution to control the spatial dimensions of the output feature maps. Padding involves adding additional rows and columns of zeros around the input data. Padding can be "valid" (no padding) or "same" (pad to keep the output size the same as the input size).

Source:

Components And Operations Within Convolutional Layers

- **Stride:** The stride determines the step size with which the filter moves across the input data during convolution. A stride of 1 means the filter moves one pixel at a time, while a stride of 2 moves it two pixels at a time. Strides affect the spatial dimensions of the output feature maps.
- **ReLU Activation:** After the convolution operation, an activation function like ReLU (Rectified Linear Unit) is applied element-wise to introduce non-linearity. ReLU sets all negative values to zero and leaves positive values unchanged, promoting sparse activation and faster training.
- **Parameter Sharing:** One of the key advantages of convolutional layers is parameter sharing. Each filter is applied across the entire input data, and the same set of weights is used at every spatial position. This significantly reduces the number of parameters compared to fully connected layers, making the network more efficient and easier to train, especially when dealing with high-dimensional data like images.
- By stacking multiple convolutional layers, followed by pooling layers and fully connected layers, CNNs can learn increasingly complex and abstract features from the input data, leading to powerful representations that are crucial for tasks like image classification, object detection, and image segmentation.

Hands On: Code Implementation for Convolution

Hands On

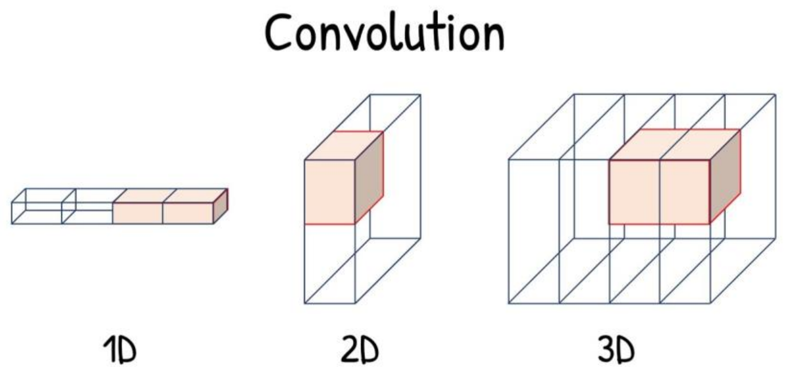
Refer: Lab 1

- Apply the convolution layer and activation layer operation to an image to extract the inside feature.

Source:

Convolution Operations

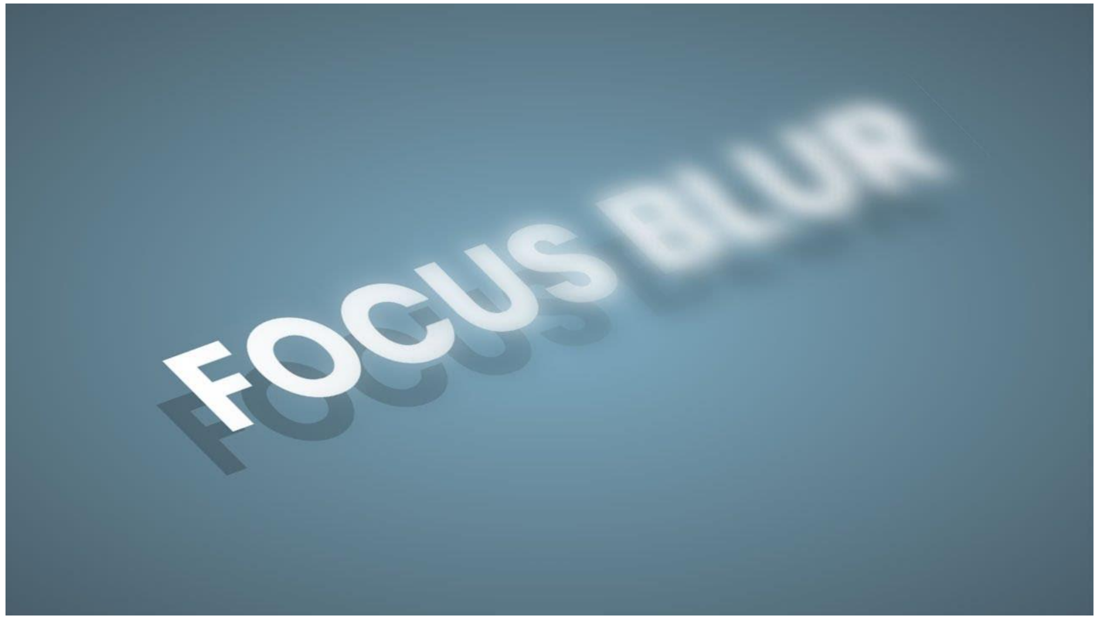
- Conv1D is used for input signals which are similar to the voice. By employing them you can find patterns across the signal. For instance, you have a voice signal and you have a convolutional layer. Each convolution traverses the voice to find meaningful patterns by employing a cos-function.
- Conv2D is used for images- This use case is very popular The convolution method used for this layer is so called convolution over volume. This means you have a two-dimensional image which contains multiple channels, RGB as an example- In this case, each convolutional filter should be a three-dimensional filter to be convolved, cross-correlated actually, with the image to find appropriate patterns across the image.
- Conv3D is usually used for videos where you have a frame for each time span. These layers usually have more parameters to be learnt than the previous layers- The reason we call them 3D is that other than images for each frame, there is another axis called time containing discrete values, and each of them corresponds to a particular frame.



Source: https://www.youtube.com/watch?v=MdNMzJ1qyOI

Image Kernel

- Blur filter



1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Source: https://www.youtube.com/watch?v=_nzc2dPRGQw

Pooling Layers

- Pooling layers are an essential component of Convolutional Neural Networks (CNNs) that help in reducing the spatial dimensions of the feature maps produced by the convolutional layers while retaining the most important information.
- Pooling is a form of down-sampling, which helps in reducing the computational complexity of the network and controlling overfitting by providing a form of translation invariance.

Types of Pooling:

- **Max Pooling:** Max pooling is the most common pooling technique. It divides the input feature map into non-overlapping rectangular regions and outputs the maximum value within each region. Max pooling helps in capturing the most important features while reducing spatial dimensions.
- **Average Pooling:** In average pooling, instead of taking the maximum value, the average value within each region is computed. While less commonly used, it has the effect of smoothing the feature maps.
- **Global Average Pooling:** This is a variation where the entire feature map is pooled into a single value by taking the average of all the values. It's often used in the final layers of the network.

Pooling Layers

- **Pooling Size:** Pooling layers have a hyperparameter called pooling size, which determines the size of the pooling window. Typical pooling sizes are 2x2 or 3x3, where the window moves across the feature map in strides of this size.
- **Stride:** Similar to convolutional layers, pooling layers also have a stride parameter that determines the step size with which the pooling window moves across the feature map. A stride of 1 means the window moves one unit at a time, while a stride of 2 moves it two units at a time.

Benefits of Pooling:

- **Dimensionality Reduction:** Pooling reduces the spatial dimensions of the feature maps, which reduces the computational complexity of the network and helps in controlling overfitting.
- **Translation Invariance:** Pooling helps in achieving translation invariance by considering local features. This means the network can recognize the same pattern regardless of its position in the image.
- **Feature Selection:** By selecting the maximum (or average) value within each region, pooling helps in retaining the most important features while discarding irrelevant ones.

Downsides of Pooling:

- **Information Loss:** Pooling discards some information from the input feature maps, which can lead to a loss of fine-grained details.
- **Reduced Spatial Resolution:** Pooling reduces the spatial dimensions of the feature maps, which might affect the ability of the network to precisely localize objects in the image.

Flattening

- The flattening layer is a specific layer used in Convolutional Neural Networks (CNNs) that converts the multi-dimensional feature maps produced by the convolutional and pooling layers into a one-dimensional array. This one-dimensional array can then be fed into fully connected layers for further processing.

Role:

- The primary purpose of the flattening layer is to reshape the multi-dimensional output of the previous convolutional or pooling layer into a format that can be understood by the fully connected layers.
- CNNs use convolutional and pooling layers to extract hierarchical features from the input data. These layers typically produce 3D or 4D output tensors (e.g., height x width x depth) where depth corresponds to the number of channels or feature maps.
- Before the output can be passed to fully connected layers, it needs to be flattened into a one-dimensional vector.

Flattening Operation:

- The flattening operation rearranges the 3D or 4D output tensor into a 1D array by simply concatenating all the elements.
- For example, if the output tensor from the previous layer is of shape (batch_size, height, width, depth), the flattening operation will reshape it into a 1D array of shape (batch_size, height * width * depth).

Fully Connected Layers

- Fully Connected Layers, also known as Dense Layers, are a crucial component of Convolutional Neural Networks (CNNs) that come after the convolutional and pooling layers. They are responsible for generating the final output of the network by performing classification, regression, or any other task-specific operation.

Some key aspect of Fully Connected Layers:

Architecture:

- Fully connected layers are composed of neurons arranged in a one-dimensional array, where each neuron is connected to every neuron in the previous layer, forming a fully connected graph.
- The input to each neuron in a fully connected layer is a weighted sum of the outputs of all neurons in the preceding layer, followed by an activation function.

Parameters:

- Each connection between neurons in adjacent layers is associated with a weight parameter, which the network learns during the training process.
- Additionally, each neuron has a bias term, which is added to the weighted sum before applying the activation function.
- The weights and biases of fully connected layers are learned through backpropagation during the training process, where the network adjusts these parameters to minimize a defined loss function.

Key Aspect of Fully Connected Layers

Role:

- Fully connected layers serve as a way to combine the features learned by the convolutional and pooling layers into high-level representations that are suitable for the final task.
- In classification tasks, the output of the last fully connected layer is often passed through a softmax function, which converts the raw scores into probabilities corresponding to each class.
- For regression tasks, the final output might be a single value or a set of continuous values, depending on the problem.

Number of Neurons:

- The number of neurons in the fully connected layers depends on the complexity of the problem and the desired capacity of the network.
- The number of neurons in the last fully connected layer corresponds to the number of classes in classification tasks or the dimensionality of the output in regression tasks.

Regularization:

- Fully connected layers are susceptible to overfitting, especially when dealing with high-dimensional feature representations.
- Techniques like dropout, L2 regularization, or batch normalization can be applied to prevent overfitting and improve generalization.

Dropout

- Dropout is a regularization technique used in neural networks, including Convolutional Neural Networks (CNNs), to prevent overfitting and improve the generalization ability of the model. It works by randomly deactivating (dropping out) a fraction of neurons in the network during training.

Here's how Dropout works:

Random Deactivation:

- During training, for each training example and each layer, Dropout randomly sets a fraction of the neurons' outputs to zero. This fraction is a hyperparameter typically set between 0.2 and 0.5.
- The neurons that are "dropped out" don't contribute to the forward pass or backward pass during that iteration of training.
- Importantly, Dropout is only applied during training, not during testing or inference.

Regularization Effect:

- By randomly dropping out neurons, Dropout prevents the network from relying too much on specific neurons and their combinations.
- This regularization effect forces the network to learn more robust features and prevents it from memorizing the training data, thus reducing overfitting.

Padding

- Padding is a technique used in Convolutional Neural Networks (CNNs) to control the spatial dimensions of the feature maps produced by convolutional layers.
- It involves adding additional border pixels to the input data before applying the convolution operation.
- Padding can help in preserving important information at the edges of the input data and in controlling the spatial dimensions of the output feature maps.

There are two common types of padding:

Valid Padding:

- In valid padding (also known as no padding), no extra pixels are added to the input data. The convolution operation is applied only to positions where the filter and the input fully overlap.

Same Padding:

- In same padding, the necessary amount of padding is added to the input data to ensure that the output feature maps have the same spatial dimensions as the input.

Role of Padding in CNNs

Preservation of Information:

- Padding helps in preserving the spatial dimensions of the input data, especially at the edges.
- Without padding, the information at the edges of the input might be underrepresented in the output feature maps.

Better Handling of Stride:

- Padding allows the use of larger stride values, which can help in reducing the spatial dimensions of the feature maps without losing too much information.
- With larger strides, the convolution operation can skip over certain areas of the input data, reducing computational costs.

Centrality of Pixels:

- Padding ensures that the central pixels of the input data have more influence on the output feature maps, as they are involved in more convolution operations.

Compatibility with Filter Sizes:

- Padding ensures that filter sizes can be chosen freely without worrying about the impact on the spatial dimensions of the output feature maps.

Strides

- Strides in Convolutional Neural Networks (CNNs) determine the step size with which the filter (or kernel) moves across the input data during convolution. The stride affects the spatial dimensions of the output feature maps produced by the convolutional layers.

Here's how strides work:

No Stride (Stride = 1):

- When the stride is set to 1, the filter moves one pixel at a time across the input data.
- This means that the filter is applied to every position of the input data, resulting in overlapping receptive fields.
- No additional padding is required to maintain the spatial dimensions of the output feature maps.

Strided Convolution (Stride > 1):

- When the stride is greater than 1 (e.g., Stride = 2), the filter moves multiple pixels at a time, skipping over certain positions of the input data.
- This results in a larger stride length between consecutive applications of the filter, effectively reducing the spatial dimensions of the output feature maps.
- Strided convolutions are commonly used to downsample the feature maps, reducing their size while retaining important features.

Key aspect of Stride in CNNs

Impact on Output Size:

- Using larger strides leads to a decrease in the spatial dimensions of the output feature maps. For example, a stride of 2 halves the spatial dimensions in each dimension.
- This reduction in size can help in reducing computational costs and memory usage in deeper networks.
- However, larger strides also result in a loss of spatial information, as the filter skips over certain regions of the input data.

Valid Padding with Strides:

- When using strides greater than 1, valid padding is typically used. Valid padding means no padding is added to the input data, and the filter is only applied to positions where it fully overlaps with the input.

Stride in Pooling Layers:

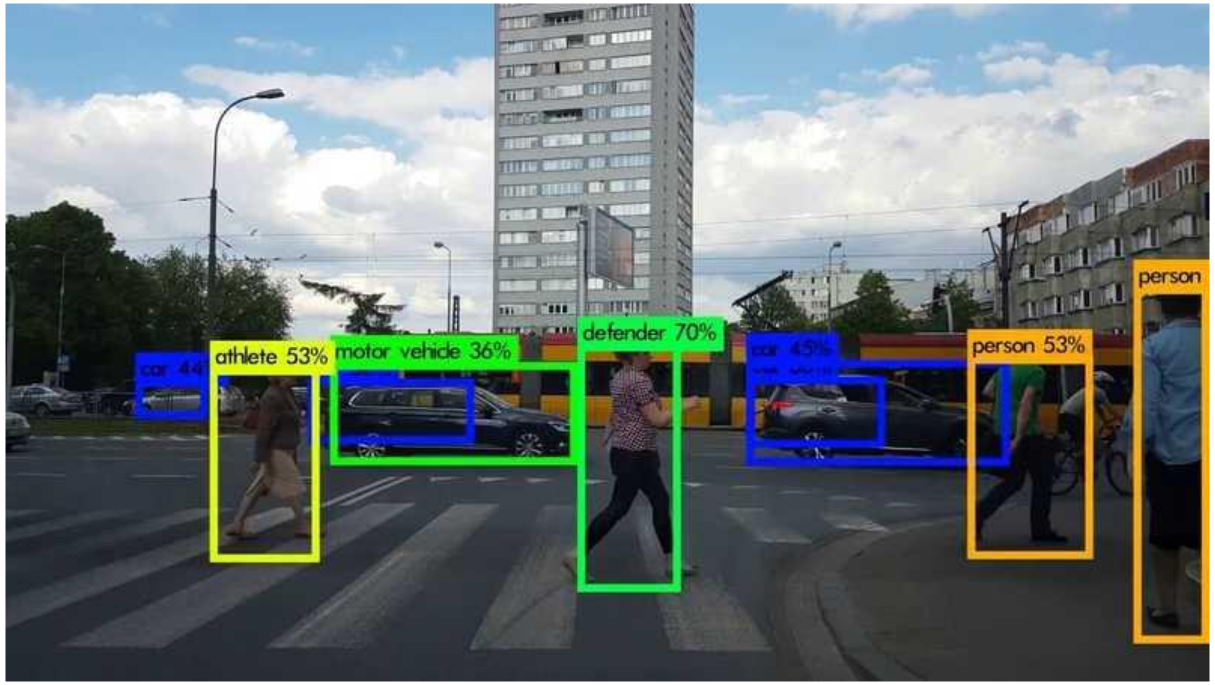
- Strides are also commonly used in pooling layers, such as max pooling or average pooling.
- Pooling layers downsample feature maps by considering only a subset of the input data, determined by the pooling window size and stride.
- Strided pooling helps in reducing the spatial dimensions of the feature maps more aggressively.

Applications of CNN in Computer Vision

- Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision with their ability to automatically learn hierarchical representations of visual data.

Here are some key applications of CNNs in computer vision:

- Image Classification
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Image Super-Resolution
- Image Generation
- Image Captioning
- Visual Question Answering (VQA)
- Facial Recognition



Source: https://theaisummer.com/YOLO/

Lab Exercise - Code Implementation for ANN

Hands On

Refer: Lab 2

- Build a CNN model for CIFAR10 dataset.

Conclusion

Congratulations! You have completed this course and gained a comprehensive understanding of Convolutional Neural Networks (CNNs). You have learned about the structure and functions of convolutional layers, pooling layers, and fully connected layers, as well as the applications of CNNs in computer vision. Additionally, you have implemented a basic CNN model, which has provided you with practical experience in building and training these powerful networks.

Quiz

1. **What is the primary purpose of Convolutional Neural Networks (CNNs)?**

A) To perform regression analysis

B) To process and analyze visual data

C) To cluster data points

D) To perform natural language processing

Answer: B
To process and analyze visual data

Quiz

2. What is the main function of a convolutional layer in a CNN?

A) To reduce the dimensionality of the input data

B) To detect local features in the input data

C) To randomly drop neurons during training

D) To connect every neuron to every other neuron

Answer: B
To detect local features in the input data

Quiz

3. What does a pooling layer do in a CNN?

A. It increases the size of the input data
B. It combines the outputs of multiple convolutional layers
C. It reduces the spatial dimensions of the input data
D. It normalizes the input data

Answer: C
It reduces the spatial dimensions of the input data

Quiz

4. What is the role of fully connected layers in a CNN?

A. To perform convolution operations
B. To down-sample the input data
C. To connect every neuron to every neuron in the previous layer
D. To normalize the input data

Answer: C
To connect every neuron to every neuron in the previous layer

Quiz

5. Which library is commonly used to implement CNN models in Python?

A) Pandas

B) Scikit-learn

C) TensorFlow/Keras

D) Matplotlib

Answer: C
TensorFlow/Keras

References

- https://www.ibm.com/topics/convolutional-neural-networks
- https://www.geeksforgeeks.org/introduction-convolution-neural-network/
- https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
- https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05#:~:text=Fully%20Connected%20Layer%20is%20simply,into%20the%20fully%20connected%20layer.
- https://www.analyticsvidhya.com/blog/2021/10/applications-of-convolutional-neural-networkscnn/

edunet foundation

Thank You!