**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |
| **Course/Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |

**CS-477 Computer Vision**

**Lab#9: Implementation of a simple CNN on Pytoroch**

Tutorial from pytorch site

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **PLO4-CLO4** | **PLO5-CLO5** | **PLO8-CLO6** | **PLO9-CLO7** |
| **Name** | **Reg. No** | **Investigation**  **(5 marks)** | **Modern Tool Usage**  **(5 marks)** | **Ethics**  **(5 marks)** | **Individual and Team Work**  **(5 marks)** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Lab#9: Implementation of a simple CNN on Pytoroch**

**Objectives**

* Understand PyTorch’s Tensor library and neural networks at a high level.
* Introduction CNN
* Training a CNN classifier

**Lab Instructions**

* This lab activity comprises of following parts: Lab Exercises, and Post-Lab Viva/Quiz session.
* The lab report shall be uploaded on LMS.
* Only those tasks that are completed during the allocated lab time will be credited to the students. Students are however encouraged to practice on their own in spare time for enhancing their skills.

**Lab Report Instructions**

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

* Lab objectives
* Python codes
* Results (graphs/tables) duly commented and discussed
* Conclusion

# Introduction to CNN

### Convolutional Neural Network (CNN): A CNN is a type of deep neural network designed to recognize and process visual data with a grid-like structure, such as images. CNNs are particularly effective in image recognition, object detection, and other computer vision tasks.

### Grid-like Topology: Images can be thought of as a grid of pixels, where each pixel represents the smallest unit of information. The arrangement of pixels creates a grid-like structure, and CNNs leverage this spatial organization for more effective feature extraction.

### Digital Image: In the context of CNNs, digital images are represented as a grid of pixels. Each pixel's position in the grid corresponds to a specific location in the image, and the pixel value represents the color and intensity at that point. The combination of pixel values across the grid forms the complete visual representation of the image.

### Binary Representation: While you mentioned a binary representation, it's important to note that digital images typically use a range of values to represent colors. In the RGB (Red, Green, Blue) color space, for example, each pixel is often represented by three values corresponding to the intensity of each color channel. The values are usually integers ranging from 0 to 255, or they could be normalized to the range [0, 1].

### Pixel Values: Pixel values indicate the brightness and color information of the image. In grayscale images, each pixel has a single value representing intensity. In color images, each pixel has multiple values corresponding to the intensities of different color channels.

### CNNs use convolutional layers to automatically and adaptively learn spatial hierarchies of features from input images. These layers contain filters that are convolved with the input image to extract features such as edges, textures, and more complex patterns. Pooling layers are often used to reduce the spatial dimensions of the data, and fully connected layers integrate the learned features for final classification or regression tasks.

### In summary, CNNs are a powerful class of neural networks designed for processing grid-like data, making them especially effective for tasks involving images and spatial relationships within those images.

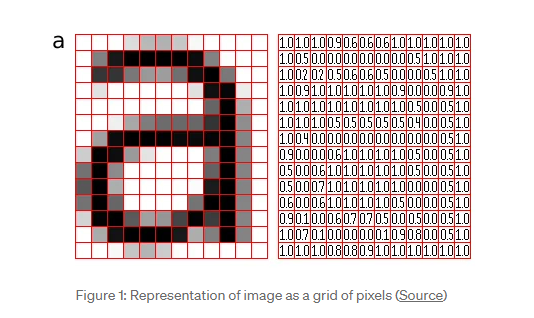


Figure 1: Source: https://pippin.gimp.org/image\_processing/images/sample\_grid\_a\_square.png

Convolutional Neural Network (CNN) typically consists of three fundamental layers: a convolutional layer, a pooling layer, and a fully connected layer.

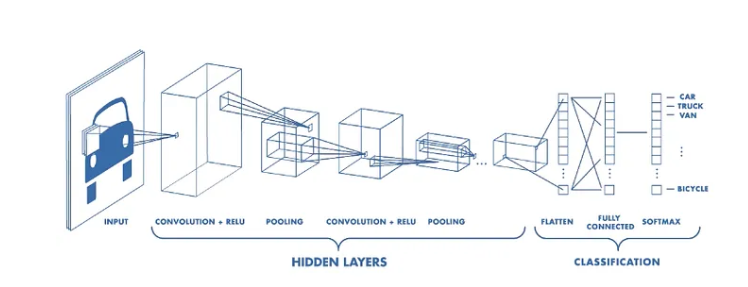


Figure 2: Source https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

Convolution Layer

At the core of the CNN architecture lies the convolutional layer, bearing the primary computational load of the network. This layer executes a dot product operation between two matrices – one matrix comprises learnable parameters known as a kernel, and the other matrix represents a confined section of the receptive field. While the kernel is spatially smaller than the image, it possesses greater depth. For instance, in an image with three (RGB) channels, the kernel's height and width are spatially small, yet its depth extends across all three channels.

**Illustration of Convolution Operation**

In the forward pass, the kernel traverses the height and width of the image, generating the image representation of the receptive region. This process yields a two-dimensional representation known as an activation map, showcasing the kernel's response at each spatial position within the image. The extent of movement of the kernel is termed as the "stride."

For an input of size W x W x D, with Dout being the number of kernels, a spatial size of F, a stride of S, and a specified amount of padding P, the size of the output volume is determined by the following formula:

A mathematical equation with black text

Description automatically generated

**Pooling layer**

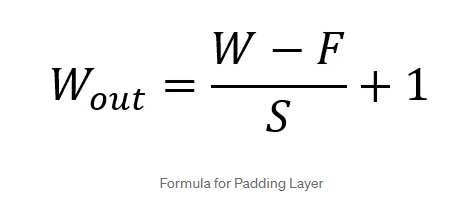
Following the convolutional layer, the pooling layer plays a crucial role in the Convolutional Neural Network (CNN) architecture. Its function involves replacing specific locations in the network's output by computing a summary statistic of the neighboring outputs. This strategic substitution aids in diminishing the spatial size of the representation, leading to a reduction in computational demands and the number of weights. Notably, the pooling operation is applied independently to each slice of the representation.

Various pooling functions exist, each offering distinct ways of summarizing information within a neighborhood. Options include the average of the rectangular neighborhood, the L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, among these, max pooling stands out as the most widely employed method. Max pooling entails selecting the maximum output value from the neighborhood, providing a robust approach to retaining essential features while reducing the dimensionality of the representation.

A diagram of a graph

Description automatically generated

If we have an activation map of size W x W x D, a pooling kernel of spatial size F, and stride S, then the size of output volume can be determined by the following formula:



This will yield an output volume of size Wout x Wout x D.In all cases, pooling provides some translation invariance which means that an object would be recognizable regardless of where it appears on the frame.

**Fully Connected Layer**

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect. The FC layer helps to map the representation between the input and the output.

**Non-Linearity Layers**

Given that convolution is inherently a linear operation and images exhibit significant non-linearity, non-linearity layers are frequently inserted immediately after the convolutional layer to impart non-linearity to the activation map.

Several types of non-linear operations are commonly employed, with notable examples including:

Sigmoid:

The sigmoid non-linearity is expressed mathematically as σ(κ) = 1/(1+e¯κ). It takes a real-valued number and compresses it into a range between 0 and 1. However, a drawback of the sigmoid function is its tendency to generate gradients close to zero, particularly at its tails. This property can hinder the backpropagation process, potentially causing the gradient to become too small and impede effective weight updates. Additionally, when the input data is consistently positive, the sigmoid may produce outputs that are either entirely positive or entirely negative, leading to a zig-zag dynamic in gradient updates for weights.

Tanh:

Tanh transforms a real-valued number to the range [-1, 1]. Similar to sigmoid, tanh activations can saturate, but unlike sigmoid, its output is zero-centered.

ReLU (Rectified Linear Unit):

ReLU has gained widespread popularity in recent years. It computes the function ƒ(κ) = max(0, κ), effectively thresholding the activation at zero. Compared to sigmoid and tanh, ReLU offers faster convergence, accelerating the learning process by a factor of six.

Despite its advantages, ReLU does have a potential drawback during training. If a large gradient flows through it, the neuron may undergo an update in such a way that further updates become unlikely. This issue can be mitigated by carefully selecting an appropriate learning rate

**Note:Watch following video to understand how CNN works**

**https://www.youtube.com/watch?v=HGwBXDKFk9I**

**lab task 1**

Build a simple convolutional neural network in PyTorch and train it to recognize handwritten digits using the MNIST dataset.

**Note: Training a classifier on the MNIST dataset can be regarded as the hello world of image recognition.**

***### TASK CODE STARTS HERE ###***

*### TASK CODE ENDS HERE ###*

***### TASK SCREENSHOT STARTS HERE ###***

*### TASK SCREENSHOT ENDS HERE ###*

*### TASK Description*

**Lab task 2**

Build a simple convolutional neural network in PyTorch and train it to recognize following fashion object using the fashion MNIST dataset.

* 10 classes (Tshirt, Trouser, Pullover, Dress, Coat, Sandal, Shift, Sneaker, Bag, Ankle boot)

***### TASK CODE STARTS HERE ###***

*### TASK CODE ENDS HERE ###*

***### TASK SCREENSHOT STARTS HERE ###***

*### TASK SCREENSHOT ENDS HERE ###*

*### TASK Description*

# Helpful links

**Convolutional Neural networks**

**https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939**

<https://www.youtube.com/watch?v=HGwBXDKFk9I>

* <https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html>
* https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/\_downloads/4e865243430a47a00d551ca0579a6f6c/cifar10\_tutorial.ipynb#scrollTo=PP9km88QkiZp