Machine Learning for Data Science

M. H. Rahman

Associate Professor
Department of Statistics and Data Science
Jahangirnagar University
Bangladesh
E-mail: habib.drj@juniv.edu

Twenty-Fifty

Image Classification

Book References

Texts

1. Dengsheng. Zhang, 2019. Fundamentals of Image Data Mining: Analysis, Features, Classification and Retrieval. SPRINGER NATURE.
2. Toennies, K.D., 2024. An Introduction to Image Classification: From Designed Models to End-to-End Learning (pp. 1-290). Springer.

References

1. Umbaugh, S.E., 2023. Computer Vision and Image Analysis: Digital Image Processing and Analysis. CRC Press.
2. Kumar, L.A. and Renuka, D.K., 2023. Deep learning approach for natural language processing, speech, and computer vision: techniques and use cases. CRC Press.

Image Classification

Image Classification Outline

- Computer vision
- Image classification
- Convolution, Padding, ReLU, Max Pooling, Flattening
- Kernels
- Classification with CNN
- Training CNNs
- Applications of CNNs

Image Classification

Computer vision

Computer vision is an interdisciplinary scientific field that focuses on addressing how computers can understand the contents of images or videos to automate tasks like the human visual system. Computer vision has a wide range of applications such as video surveillance, face recognition, face annotation, image retrieval, biometrics, traffic monitoring, visibility restoration, object detection, etc. The four main tasks of computer vision are:

- Image classification
- Object detection
- Semantic segmentation
- Instance segmentation

Image Classification

Computer vision...

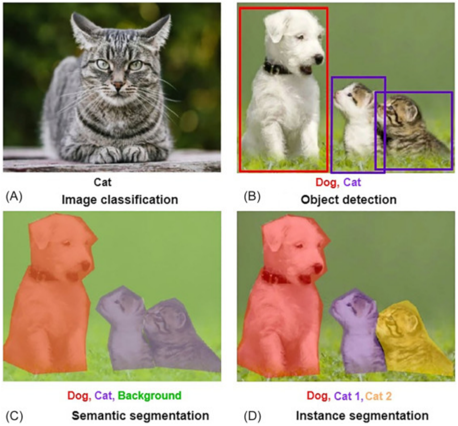


Figure: Four main tasks in computer vision

Image Classification

Computer vision...

- Image classification: To identify the category of the image. The common network models used for image classification include GoogLeNet, ResNet, and so on.
- Object detection: To find out the category of objects and their location in a given image. The common network models used for object detection include Faster R-CNN, YOLO-v3, and so on.
- Semantic segmentation: To assign a label to every pixel in the image. The common network models used for semantic segmentation include FCN, DeepLab-v3, and so on.
- Instance segmentation: To identify the boundary of each object at the detailed pixel level in the image. The common network models used for instance segmentation include FCIS, Mask R-CNN, and so on.

Image Classification

Image classification

Image classification is a fundamental task in computer vision. This task aims at categorizing images into one or several predefined classes (Matias et al.; 2021).

Image classification can be considered the basis for other computer vision tasks, e.g., localization, detection, and segmentation (Rawat and Wang, 2017).

Even though it is a trivial task for human beings, it is challenging for an automated system (Cireșan et al., 2011). In the past few years, the developers usually adopted a two-step image processing pipeline approach to solve this problem.

First, a set of image type-specific features were extracted from images using a sequence of one or more handcrafted feature descriptors. Afterwards, these features were used to train a classifier (Lecun et al., 1998).

Convolution

Convolution

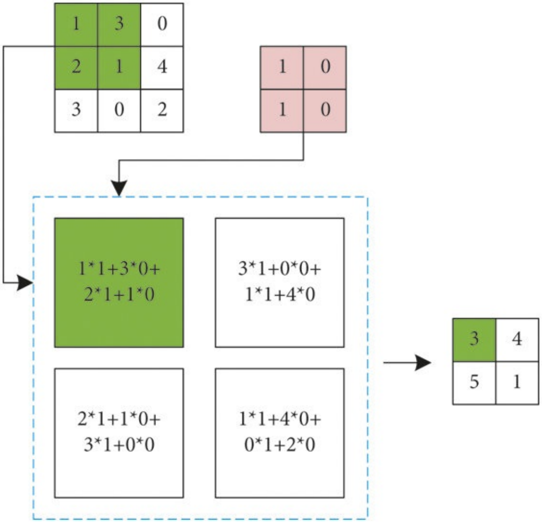
Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel. This is related to a form of mathematical convolution. The matrix operation being performed—convolution—is not traditional matrix multiplication

Convolution is a mathematical operation that describes a rule of how to combine two functions or pieces of information to form a third function. The feature map (or input data) and the kernel are combined to form a transformed feature map.

Convolution is a mathematical operation that allows the merging of two sets of information. In the case of CNN, convolution is applied to the input data to filter the information and produce a feature map.

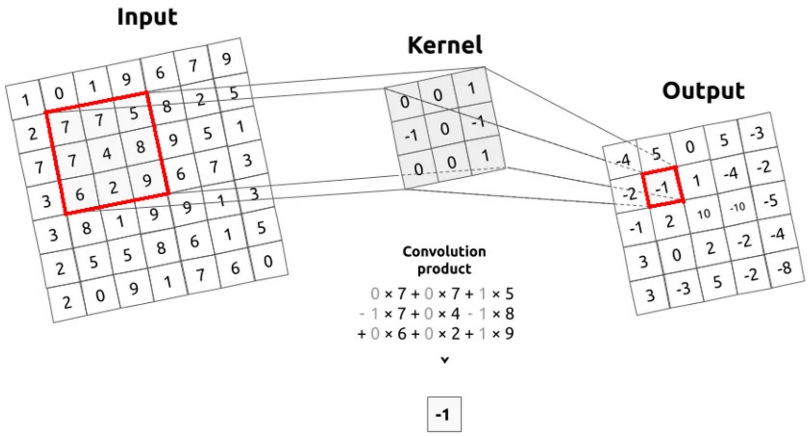
Convolution

Convolution



Convolution

Convolution



Convolution Example in CNN

Convolution Example in CNN

Consider a grayscale image represented as follows:

$$\text{Image} = \begin{bmatrix} 1 & 2 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 0 & 1 \end{bmatrix}$$

And a convolutional filter defined as:

$$\text{Filter} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Convolution Example in CNN

The convolution operation involves sliding the filter over the image and performing element-wise multiplication followed by summation.
Calculating for position (0,0):

$$(1*1)+(0*2)+(-1*3)+(1*0)+(0*1)+(-1*2)+(1*1)+(0*2)+(-1*3) = -6$$

Calculating for position (0,1):

$$(1*2)+(0*3)+(-1*0)+(1*1)+(0*2)+(-1*3)+(1*2)+(0*3)+(-1*0) = 2$$

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -6 & 2 & 6 \\ -6 & -2 & 6 \\ -6 & 2 & 6 \end{bmatrix}$$

$$(n_1 \times n_2) * (f \times f) = \left[\frac{n_1 - f}{s} + 1 \right] \times \left[\frac{n_2 - f}{s} + 1 \right]$$

Convolution Example in CNN

The output feature map is:

$$\text{Output Feature Map} = \begin{bmatrix} -6 & 2 & 6 \\ -6 & -2 & 6 \\ -6 & 2 & 6 \end{bmatrix}$$

To calculate the output dimensions, we use:

$$\text{Output Size} = \left\lfloor \frac{(n_1 - f)}{s} + 1 \right\rfloor \times \left\lfloor \frac{(n_2 - f)}{s} + 1 \right\rfloor$$

Where:

$$n_1 = 5, n_2 = 5, f = 3, s = 1$$

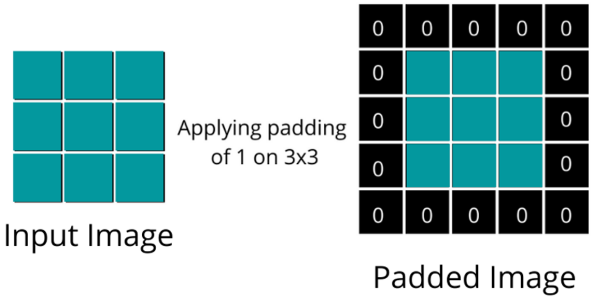
Thus,

$$\text{Output Size} = \left\lfloor \frac{(5 - 3)}{1} + 1 \right\rfloor \times \left\lfloor \frac{(5 - 3)}{1} + 1 \right\rfloor = 3 \times 3$$

Therefore, the output dimensions are 3×3.

Padding

Padding
Padding in Convolutional Neural Networks (CNNs) is a crucial technique that involves adding extra pixels around the input image or feature map before performing convolution operations. This practice helps maintain spatial dimensions, preserves important information at the edges, and enhances the overall performance of the network.



Padding

To calculate the output dimensions:

$$\text{Output Size} = \left\lfloor \frac{(n_1 + 2p - f)}{s} + 1 \right\rfloor \times \left\lfloor \frac{(n_2 + 2p - f)}{s} + 1 \right\rfloor$$

ReLU

CNN Architecture with ReLU Consider a simple CNN architecture designed for image classification:

- Input Layer: Accepts images (e.g., 32x32 pixels).
- Convolutional Layer: Applies several filters to extract features from the images.
- ReLU Layer: Applies the ReLU activation function to the output of the convolutional layer.
- Pooling Layer: Reduces dimensionality while retaining important features (e.g., max pooling).
- Fully Connected Layer: Combines features for classification.

Implementation using Python

Step-by-Step Implementation using Python with TensorFlow/Keras

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Define a simple CNN model
model = models.Sequential()

# Input layer
model.add(layers.InputLayer(input_shape=(32, 32, 3)))

# Convolutional layer
model.add(layers.Conv2D(32, (3, 3), activation='relu'))

# ReLU layer (implicitly included in Conv2D)
# If needed separately:
# model.add(layers.Activation('relu'))
```

Implementation using Python

Step-by-Step Implementation using Python with TensorFlow/Keras...

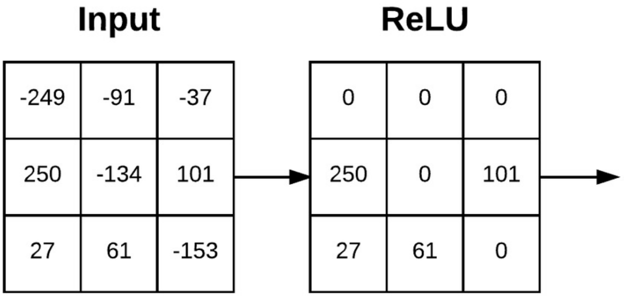
```
# Pooling layer
model.add(layers.MaxPooling2D((2, 2)))

# Flattening and Fully Connected Layer
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
 # Assuming 10 classes for classification
model.add(layers.Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
```

ReLU

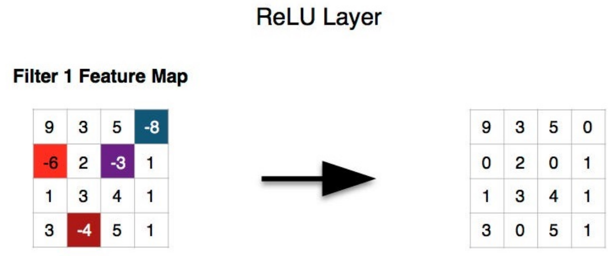
ReLU

The Rectified Linear Unit (ReLU) is a widely used activation function in Convolutional Neural Networks (CNNs) due to its simplicity and effectiveness in introducing non-linearity into the model. ReLU is defined mathematically as:

$$f(x) = max(0, x)$$

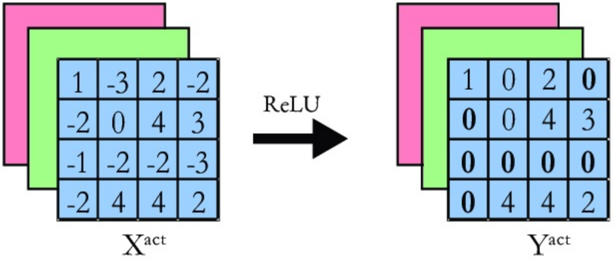


ReLU



9	3	5	-8
-6	2	-3	1
1	3	4	1
3	-4	5	1

9	3	5	0
0	2	0	1
1	3	4	1
3	0	5	1



1	-3	2	-2
-2	0	4	3
-1	-2	-2	-3
-2	4	4	2

1	0	2	0
0	0	4	3
0	0	0	0
0	4	4	2

ReLU

ReLU
After convolution, applying ReLU helps in introducing non-linearity and speeding up training by allowing only positive activation.

ReLU is a fundamental component in modern CNN architectures due to its efficiency and effectiveness at enabling complex representations. Its ability to mitigate issues like vanishing gradients makes it a preferred choice over traditional activation functions. When constructing CNNs for tasks such as image classification, incorporating ReLU can significantly enhance performance and training speed

Max Pooling

Max Pooling

Max pooling is a crucial operation in Convolutional Neural Networks (CNNs) that helps reduce the spatial dimensions of feature maps while retaining the most significant features. Here's a detailed example of how max pooling works, including calculations for each position.

Max Pooling

Max Pooling Example

Let's consider a 4×4 feature map obtained from a Convolutional layer. Consider the 4×4 feature map:

$$\text{Feature Map} = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 6 & 2 & 8 \\ 9 & 7 & 3 & 1 \\ 4 & 2 & 0 & 6 \end{bmatrix}$$

We will apply max pooling with a window size of 2×2 and a stride of 2.

Max Pooling Calculations

1. Position (0,0): Window:

$$\begin{bmatrix} 1 & 3 \\ 5 & 6 \end{bmatrix}$$

- Maximum Value: 6

Max Pooling

2. Position (0,1): Window:

$$\begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix}$$

- Maximum Value: 8

3. Position (1,0): Window:

$$\begin{bmatrix} 9 & 7 \\ 4 & 2 \end{bmatrix}$$

- Maximum Value: 9

4. Position (1,1): Window:

$$\begin{bmatrix} 3 & 1 \\ 0 & 6 \end{bmatrix}$$

- Maximum Value: 6

Max Pooling

Feature Map $\rightarrow$ Pooled Feature Map

$$\left[\begin{array}{cc|cc} 1 & 3 & 2 & 4 \\ 5 & 6 & 2 & 8 \\ \hline 9 & 7 & 3 & 1 \\ 4 & 2 & 0 & 6 \end{array}\right] \rightarrow \begin{bmatrix} 6 & 8 \\ 9 & 6 \end{bmatrix}$$

The resulting pooled feature map is as follows:

$$\text{Pooled Feature Map} = \begin{bmatrix} 6 & 8 \\ 9 & 6 \end{bmatrix}$$

This demonstrates how max pooling reduces the spatial dimensions while retaining important features from the input feature map.

Convolution and Pooling

Example: Consider a grayscale image represented as follows:

$$\text{Image} = \begin{bmatrix} 9 & 2 & 9 & 2 & 1 & 1 \\ 8 & 4 & 8 & 3 & 5 & 0 \\ 7 & 5 & 5 & 5 & 8 & 1 \\ 9 & 5 & 7 & 3 & 7 & 1 \\ 5 & 6 & 8 & 5 & 2 & 1 \\ 7 & 5 & 4 & 4 & 5 & 5 \end{bmatrix}$$

And, a convolutional filter defined as

$$\text{Filter} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Find Max Pooling and Average Pooling Feature Map using 2×2 window after convolution with stride 1.

Convolution and Pooling

Solution Here, Image dimension: $n_1 = n_2 = 6$, Kernel dimension: $f = 3$, Stride: $s = 1$

$$\begin{bmatrix} 9 & 2 & 9 & 2 & 1 & 1 \\ 8 & 4 & 8 & 3 & 5 & 0 \\ 7 & 5 & 5 & 5 & 8 & 1 \\ 9 & 5 & 7 & 3 & 7 & 1 \\ 5 & 6 & 8 & 5 & 2 & 1 \\ 7 & 5 & 4 & 4 & 5 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 8 & 8 \\ 4 & 3 & 0 & 9 \\ 1 & 3 & 3 & 10 \\ 2 & 4 & 5 & 5 \end{bmatrix}$$

$$(n_1 \times n_2) * (f \times f) = [(n_1 - f)/s + 1] \times [(n_2 - f)/s + 1]$$

ReLU function: $f(x) = \max(0, x)$

$$\begin{bmatrix} 2 & 1 & 8 & 8 \\ 4 & 3 & 0 & 9 \\ 1 & 3 & 3 & 10 \\ 2 & 4 & 5 & 5 \end{bmatrix} \xrightarrow{ReLU} \begin{bmatrix} 2 & 1 & 8 & 8 \\ 4 & 3 & 0 & 9 \\ 1 & 3 & 3 & 10 \\ 2 & 4 & 5 & 5 \end{bmatrix}$$

Convolution and Pooling

Max Pooling

$$\left[\begin{array}{cc|cc} 2 & 1 & 8 & 8 \\ 4 & 3 & 0 & 9 \\ \hline 1 & 3 & 3 & 10 \\ 2 & 4 & 5 & 5 \end{array}\right] \to \begin{bmatrix} 4 & 9 \\ 4 & 10 \end{bmatrix}$$

Average pooling

$$\left[\begin{array}{cc|cc} 2 & 1 & 8 & 8 \\ 4 & 3 & 0 & 9 \\ \hline 1 & 3 & 3 & 10 \\ 2 & 4 & 5 & 5 \end{array}\right] \to \begin{bmatrix} 2.5 & 6.25 \\ 2.5 & 5.75 \end{bmatrix}$$

Convolution for Color Image

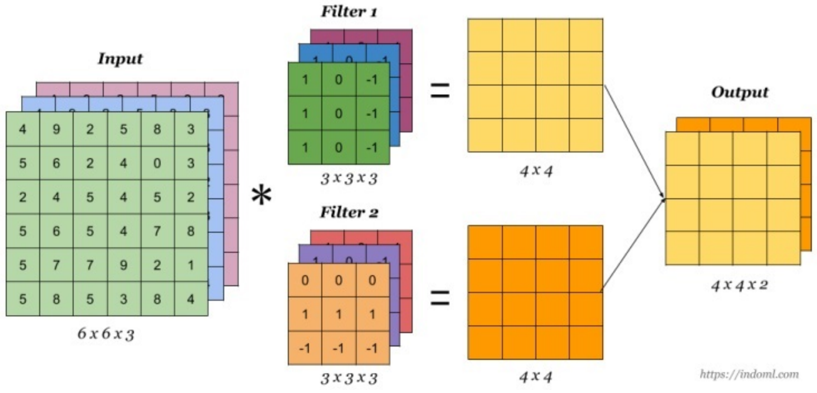


Figure: Convolution for Color Image.

Flattening

Flattening
Flattening is a crucial step in the architecture of Convolutional Neural Networks (CNNs), serving as the bridge between the convolutional layers, which extract features from input data, and the fully connected layers, which perform classification or regression tasks.

The primary purpose of the flatten layer is to convert the multidimensional output from preceding layers (typically convolutional and pooling layers) into a one-dimensional array. This transformation is necessary because fully connected layers require input in a flat format to process the data effectively. For instance, if the output tensor from a convolutional layer has dimensions $(batch - size, height, width, channels)$, the flatten layer reshapes it to $(batch - size, height \times width \times channels)$

Flattening

Flattening

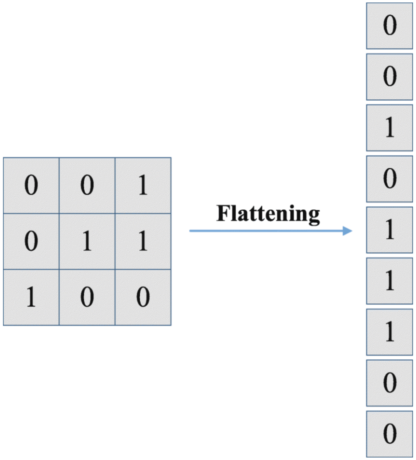


Figure: Flattening

Flattening

Flattening

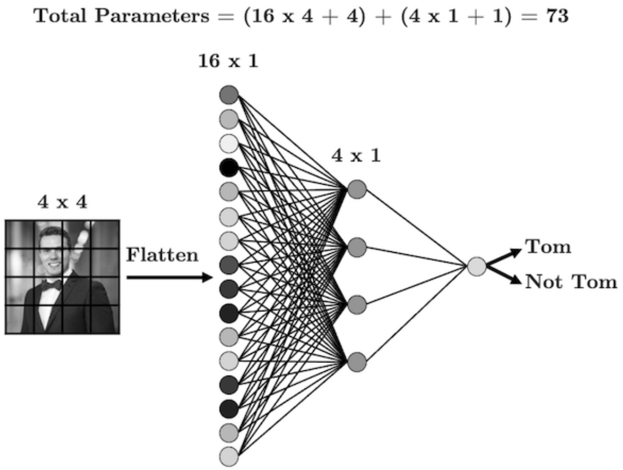


Figure: Flattening

Flattening

Consider a CNN designed for image classification. After several convolutional and pooling operations, you might have an output tensor with dimensions such as $(32, 7, 7, 64)$ (where 32 is the batch size). The flatten layer would convert this tensor into a one-dimensional array of length $32 \times 7 \times 7 \times 64 = 100352$. This flattened output can then be fed into dense layers for classification tasks.

While flattening has been a standard practice in CNNs, recent trends suggest alternatives like Global Average Pooling (GAP). GAP reduces feature maps to a single vector by averaging values instead of flattening them directly. This method can be more efficient and mitigate issues related to overfitting and computational load

Kernels

Kernels

Convolution kernels, or filters, are fundamental in image processing, serving as small matrices applied over images to perform specific operations.

These operations include enhancing details, detecting edges, and reducing noise. The process of using a convolution kernel involves sliding it across the image and applying a mathematical operation at each position to alter the pixel values.

This operation, known as convolution, helps in highlighting important features or smoothing out imperfections in the image.

Kernels

Basic Convolution Kernels

- 1. Identity Kernel $= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- 2. Edge Detection Kernels
  - Horizontal Edge Detection Kernel $= \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
  - Vertical Edge Detection Kernel $= \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
- 3. Sharpening Kernel $= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- 4. Box Blur Kernel $= \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$

Kernels

Advanced Convolution Kernels

- 1. Gaussian Blur Kernel $= \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
- 2. Sobel Kernels
  - Sobel Kernel (Horizontal) $= \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - Sobel Kernel (Vertical) $= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Kernels

Advanced Convolution Kernels...

- 3. Prewitt Kernels
  - Prewitt Kernel (Horizontal) $= \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
  - Prewitt Kernel (Vertical) $= \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- 4. Laplacian Kernel $= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Kernels

Specialized Convolution Kernels

- 1. Gabor Kernels

$$G(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left[-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right] \cos\left[2\pi \frac{x'}{\lambda} + \psi\right]$$

$x' = x\cos(\theta) + y\sin(\theta)$ and $y' = x\sin(\theta) + y\cos(\theta)$

λ is the wavelength of the sinusoidal factor. θ is the orientation of the normal to the parallel stripes of a Gabor function. ψ is the phase offset. σ is the standard deviation of the Gaussian envelope. γ is the spatial aspect ratio.

- 2. Scharr Kernels
  - Scharr Kernel (Horizontal) = $\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$
  - Scharr Kernel (Vertical) = $\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$

Kernels

Convolution Kernels
A convolution kernel is a small matrix used to apply convolution operations on input data, extracting features like edges, textures, and patterns.

Gabor kernels
Gabor kernels are used for texture analysis and feature extraction, particularly effective in edge detection and texture discrimination.

Example of Convolution, ReLU, and Pooling for CNN

Example Consider a grayscale image represented as follows:

$$\text{Image} = \begin{bmatrix} 1 & 0 & 9 & 2 & 1 & 5 \\ 2 & 1 & 8 & 3 & 5 & 8 \\ 3 & 6 & 5 & 5 & 8 & 9 \\ 4 & 2 & 7 & 3 & 7 & 6 \\ 5 & 3 & 8 & 5 & 2 & 5 \\ 6 & 4 & 4 & 4 & 5 & 5 \end{bmatrix}$$

And, a convolutional filter defined as

$$\text{Filter} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Find Max Pooling and Average Pooling Feature Map using 2x2 window with stride 2 after convolution with stride 1 and ReLU.

Example of Convolution, ReLU, and Pooling for CNN

Solution

$$\textbf{Image} * \textbf{Kernel} = \textbf{Convolution Feature Map}$$

$$\begin{bmatrix} 1 & 0 & 9 & 2 & 1 & 5 \\ 2 & 1 & 8 & 3 & 5 & 8 \\ 3 & 6 & 5 & 5 & 8 & 9 \\ 4 & 2 & 7 & 3 & 7 & 6 \\ 5 & 3 & 8 & 5 & 2 & 5 \\ 6 & 4 & 4 & 4 & 5 & 5 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 22 & 5 & -11 & 17 \\ 13 & 1 & 3 & 16 \\ 11 & 3 & -3 & 10 \\ 7 & 5 & -11 & 4 \end{bmatrix}$$

Applying ReLU to the Convolution Feature Map

$$f(x) = \max(0, x)$$

$$\begin{bmatrix} 22 & 5 & -11 & 17 \\ 13 & 1 & 3 & 16 \\ 11 & 3 & -3 & 10 \\ 7 & 5 & -11 & 4 \end{bmatrix} \xrightarrow{ReLU} \begin{bmatrix} 22 & 5 & 0 & 17 \\ 13 & 1 & 3 & 16 \\ 11 & 3 & 0 & 10 \\ 7 & 5 & 0 & 4 \end{bmatrix}$$

Example of Convolution, ReLU, and Pooling for CNN

$$\text{ReLU Map} = \left[\begin{array}{cc|cc} 22 & 5 & 0 & 17 \\ 13 & 1 & 3 & 16 \\ \hline 11 & 3 & 0 & 10 \\ 7 & 5 & 0 & 4 \end{array}\right]$$

$$\text{Max Pooling} = \begin{bmatrix} 22 & 17 \\ 11 & 10 \end{bmatrix}$$

$$\text{Average Pooling} = \begin{bmatrix} 10.25 & 9.00 \\ 6.50 & 3.50 \end{bmatrix}$$

Overview of Image Classification with CNN

Overview of Image Classification with Convolutional Neural Networks (CNNs)
Image classification is a fundamental task in computer vision, where the goal is to assign a label to an image based on its content. Convolutional Neural Networks (CNNs) are the most widely used architecture for this purpose due to their ability to automatically extract and learn features from images.

Overview of Image Classification with CNN

How CNNs Work

- **1. Input Layer**
  The process begins with the input layer, where an image is represented as a matrix of pixel values. Each pixel's intensity can be in grayscale or color (RGB), creating a multi-dimensional array.
- **2. Convolutional Layer**
  The convolutional layer is the core component of a CNN. It applies various filters (also known as kernels) to the input image to create feature maps. The convolution operation involves sliding the filter across the image and performing element-wise multiplication followed by summation, resulting in a new matrix that highlights specific features such as edges or textures. This process helps reduce spatial dimensions while retaining essential information.

Overview of Image Classification with CNN

How CNNs Work..

- **3. Activation Function (ReLU)**
  After convolution, an activation function is applied, commonly the Rectified Linear Unit (ReLU). This function introduces non-linearity into the model by converting all negative values in the feature map to zero, allowing the network to learn complex patterns.
- **4. Pooling Layer**
  Pooling layers are used to downsample the feature maps, reducing their size and computational load while preserving important features. Max pooling is a popular method where a filter slides over the feature map and selects the maximum value from each region. This step not only speeds up computation but also helps in making the detection of features invariant to small translations in the input image.

Overview of Image Classification with CNN

How CNNs Work..

- **5. Fully Connected Layer**
  The final stage of a CNN involves one or more fully connected layers. These layers take the flattened output from the previous pooling layer and compute class scores using weights learned during training. The final output is typically processed through a softmax function, which converts raw scores into probabilities for each class. For example, after processing an image, a CNN might output a 90% probability that it depicts a dog and 10% for a cat.

Image Classification

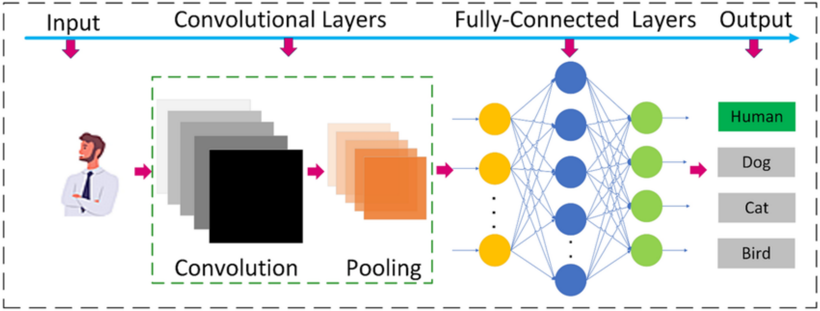


Image Classification

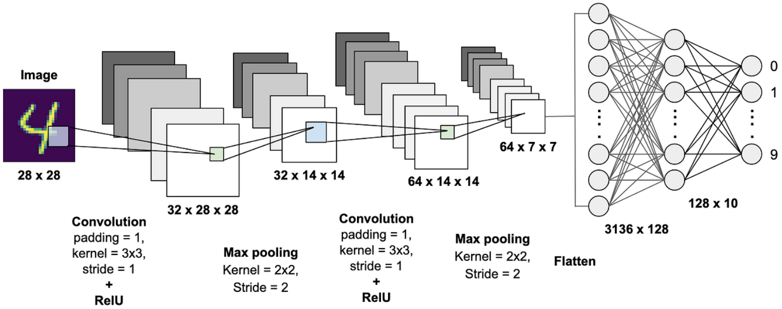


Image Classification

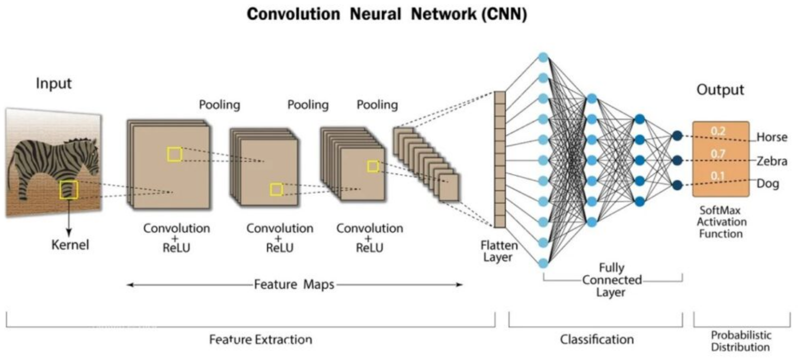


Image Classification

Training CNNs

Training a CNN involves feeding it labeled images and adjusting its weights based on how well it predicts the correct labels. This is done through backpropagation, where the model learns from its errors by updating weights to minimize loss functions such as categorical cross-entropy. Regularization techniques like dropout may also be employed to prevent overfitting by randomly ignoring some neurons during training.

Image Classification

Applications CNNs are utilized in various applications including:

- **Image Recognition**: Identifying objects within images.
- **Facial Recognition**: Classifying faces for security systems.
- **Medical Imaging**: Analyzing medical scans for diagnosis.
- **Autonomous Vehicles**: Recognizing road signs and pedestrians.

Image Classification

CNNs have revolutionized image classification by automating feature extraction and learning through layered architectures. Their ability to handle high-dimensional data efficiently makes them indispensable in modern computer vision tasks. As research progresses, CNN architectures continue to evolve, leading to improved accuracy and new applications across different fields.

Book

Chapter 8 (Kumar and Renuka; 2023),
Chapter 9 (Goodfellow et al.; 2016)

References I

Goodfellow, I. et al. (2016). Deep learning-ian goodfellow, yoshua bengio, aaron courville, *Adapt. Comput. Mach. Learn* .

Kumar, L. A. and Renuka, D. K. (2023). *Deep learning approach for natural language processing, speech, and computer vision: techniques and use cases*, CRC Press.

Matias, A. V., Amorim, J. G. A., Macarini, L. A. B., Cerentini, A., Onofre, A. S. C., Onofre, F. B. D. M., Daltoé, F. P., Stemmer, M. R. and von Wangenheim, A. (2021). What is the state of the art of computer vision-assisted cytology? a systematic literature review, *Computerized Medical Imaging and Graphics* **91**: 101934.

Machine Learning for Data Science

***** - /// - There are no End .. - /// - *****

****

****