CS2V13
Neural Networks and Deep learning

Dr. J. Benadict Raja
Associate Professor / CSE
PSNA College of Engineering and Technology
Dindigul, TN

Unit III : Convolutional networks and sequence modeling

Convolutional networks
- – convolution operation
- – motivation pooling
- – basic convolution function
- – algorithms

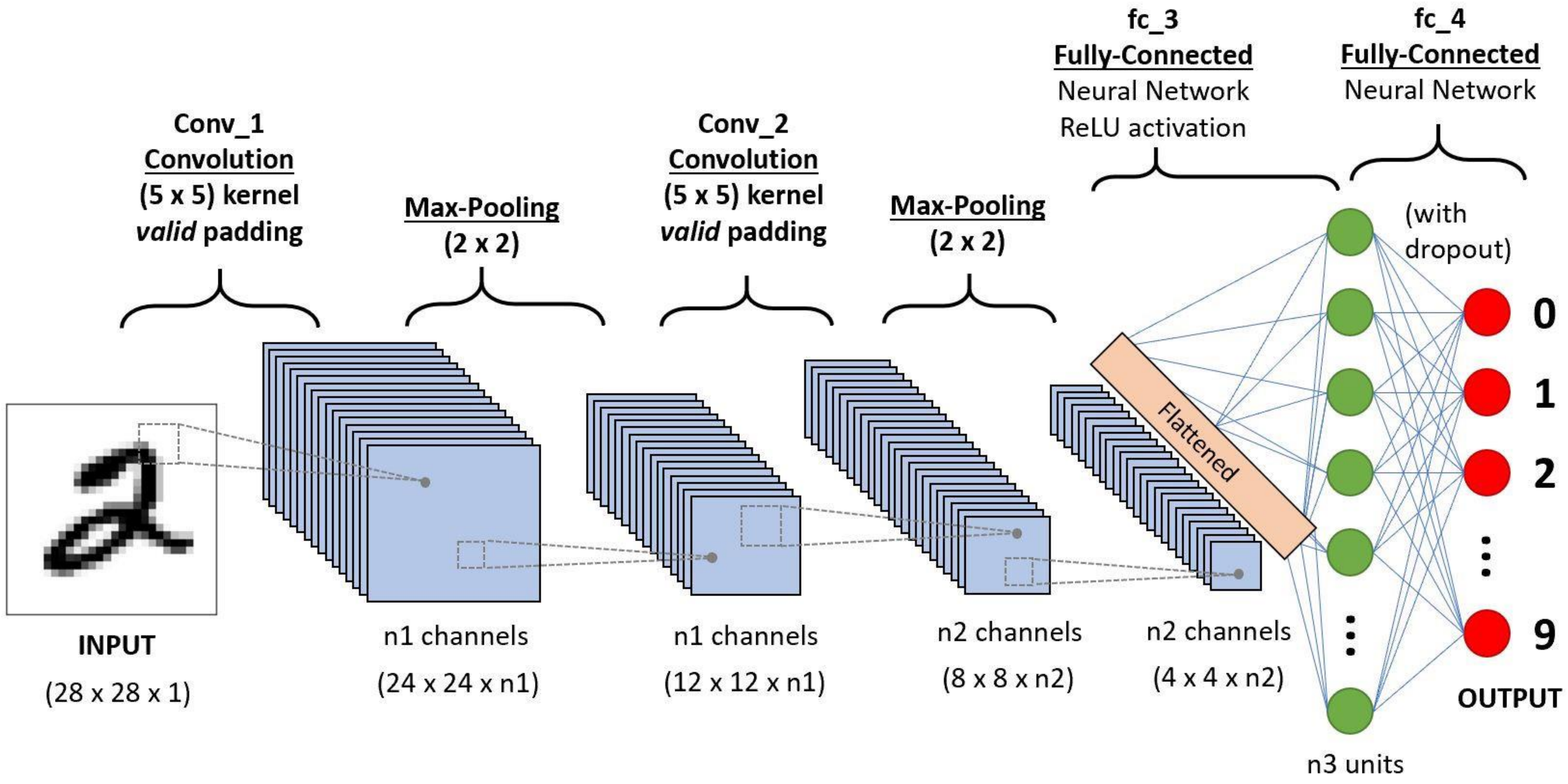
Recurrent and recursive nets : recurrent neural networks
- – bidirectional RNN
- – recursive neural network
- – auto regressive networks

Introduction to long–term dependencies and temporal dependencies

Application in object recognition.

Convolutional networks

also known as convolutional neural networks or CNNs.
A Convolutional Neural Network (CNN or ConvNet) is a type of deep learning model specifically designed to process structured grid data, like images, video, and speech. .
the network employs a mathematical operation called convolution



Convolutional networks

1. Convolutional Layers
Convolution Operation: The core operation in CNNs is the convolution, where a small matrix called a filter or kernel slides over the input data to produce a feature map. This operation captures local patterns like edges, textures, and shapes in the input data.
Feature Maps: As the filter convolves over the input, it generates feature maps that highlight specific features of the input, like edges or textures.

2. Activation Functions
ReLU (Rectified Linear Unit): After each convolution operation, an activation function like ReLU is applied to introduce non-linearity, helping the network learn more complex patterns. ReLU simply replaces all negative values in the feature map with zero.

3. Pooling Layers
Max Pooling/Average Pooling: Pooling layers reduce the spatial dimensions (width and height) of the feature maps while retaining the most important information. Max pooling, for example, takes the maximum value from a set of values in the feature map, which helps to reduce computational complexity and avoid overfitting.

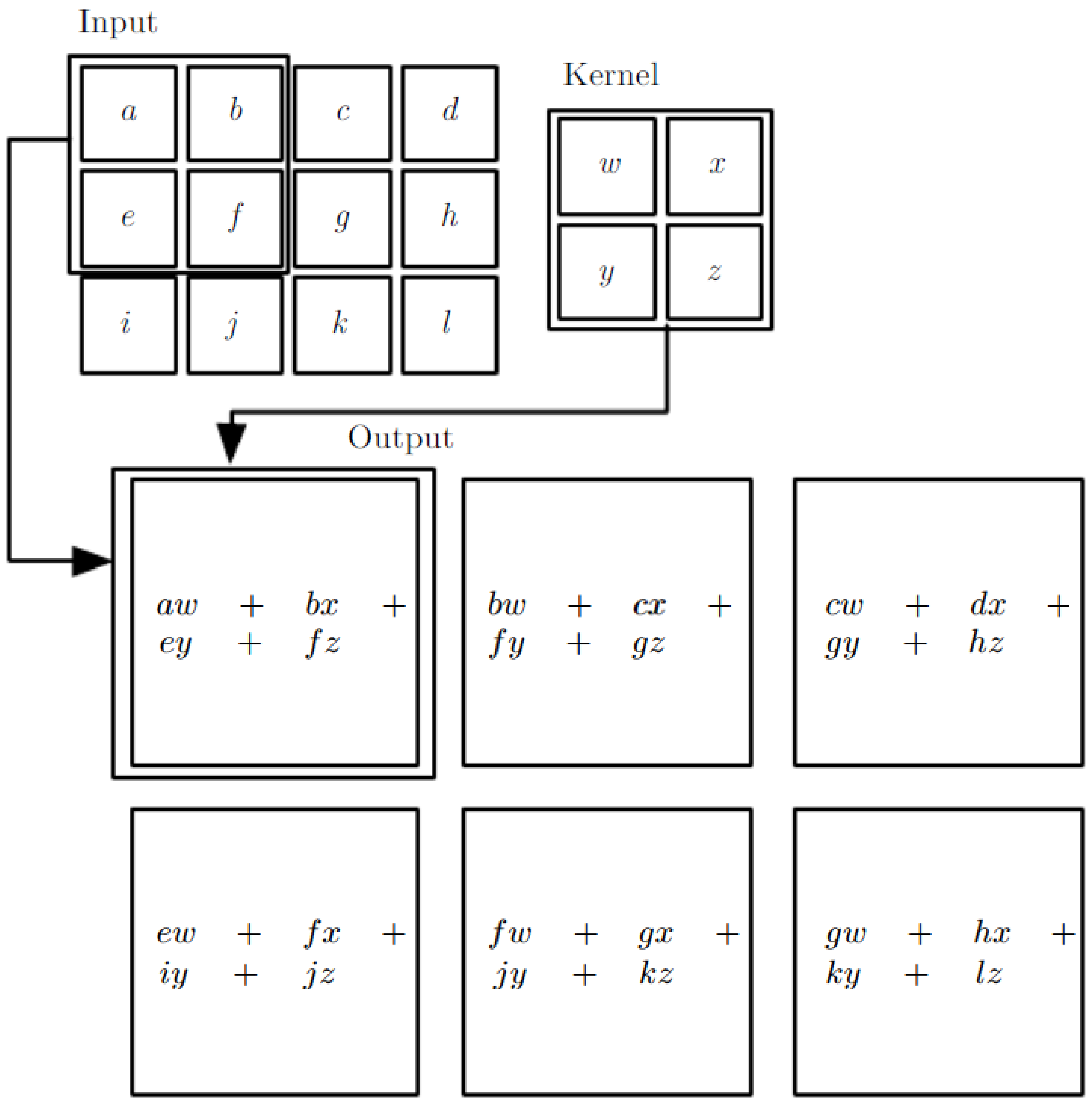
Convolutional networks

4. Fully Connected Layers
Dense Layers: After several convolutional and pooling layers, the network typically includes one or more fully connected (dense) layers. These layers take the high-level filtered information from previous layers and make predictions or classifications.
Softmax/Output Layer: In classification tasks, the final layer usually uses a softmax function to output probabilities for different classes.

5. Training CNNs
Backpropagation and Optimization: CNNs are trained using backpropagation, where the error is propagated back through the network to update the weights. Optimization algorithms like stochastic gradient descent (SGD) are used to minimize the loss function.
Data Augmentation: To improve the generalization of CNNs, techniques like data augmentation (e.g., rotating, flipping images) are used during training to artificially increase the size of the training dataset.

6. Applications
Image Classification: Recognizing objects or patterns within an image.
Object Detection: Identifying and localizing objects within an image.
Image Segmentation: Classifying each pixel in an image to understand the object boundaries and regions.
Facial Recognition: Identifying or verifying a person based on their facial features.

CNN - Convolution

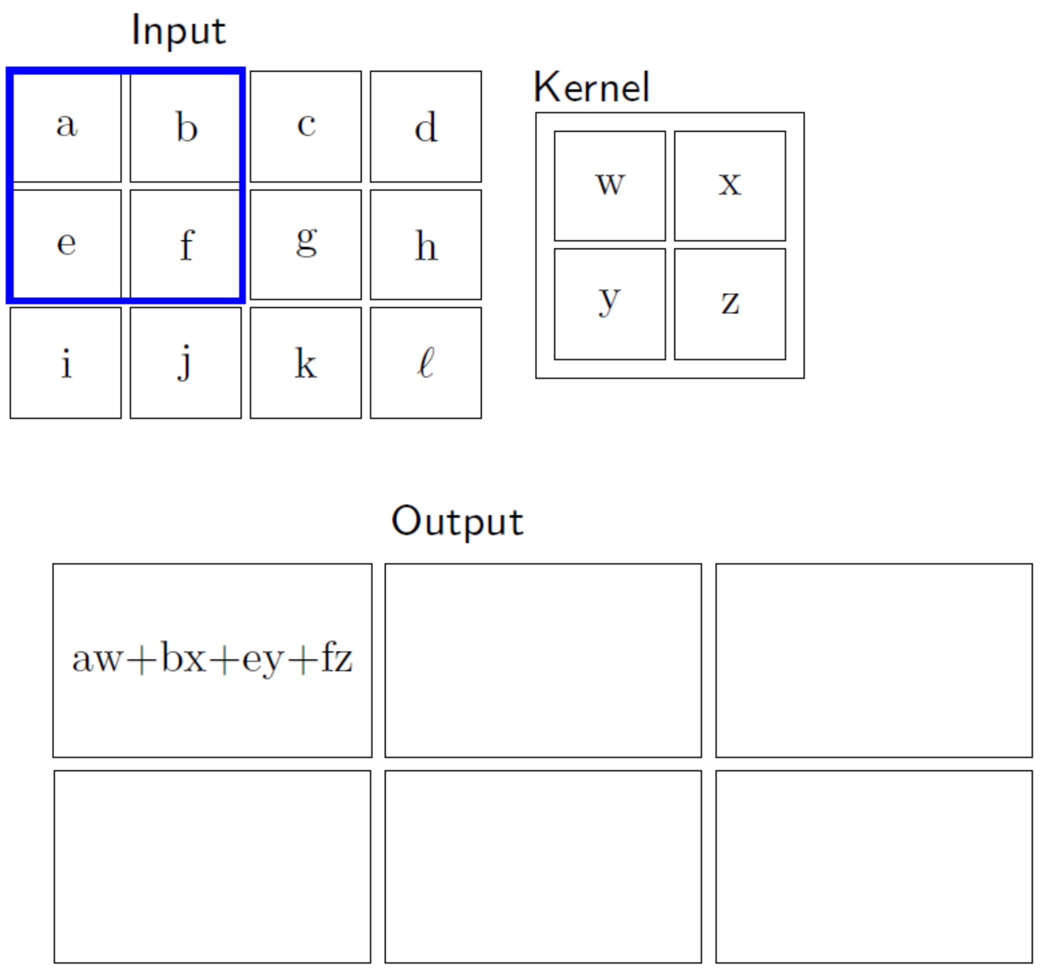


The convolution operation is a fundamental mathematical operation used in Convolutional Neural Networks (CNNs) to extract features from input data, typically images.

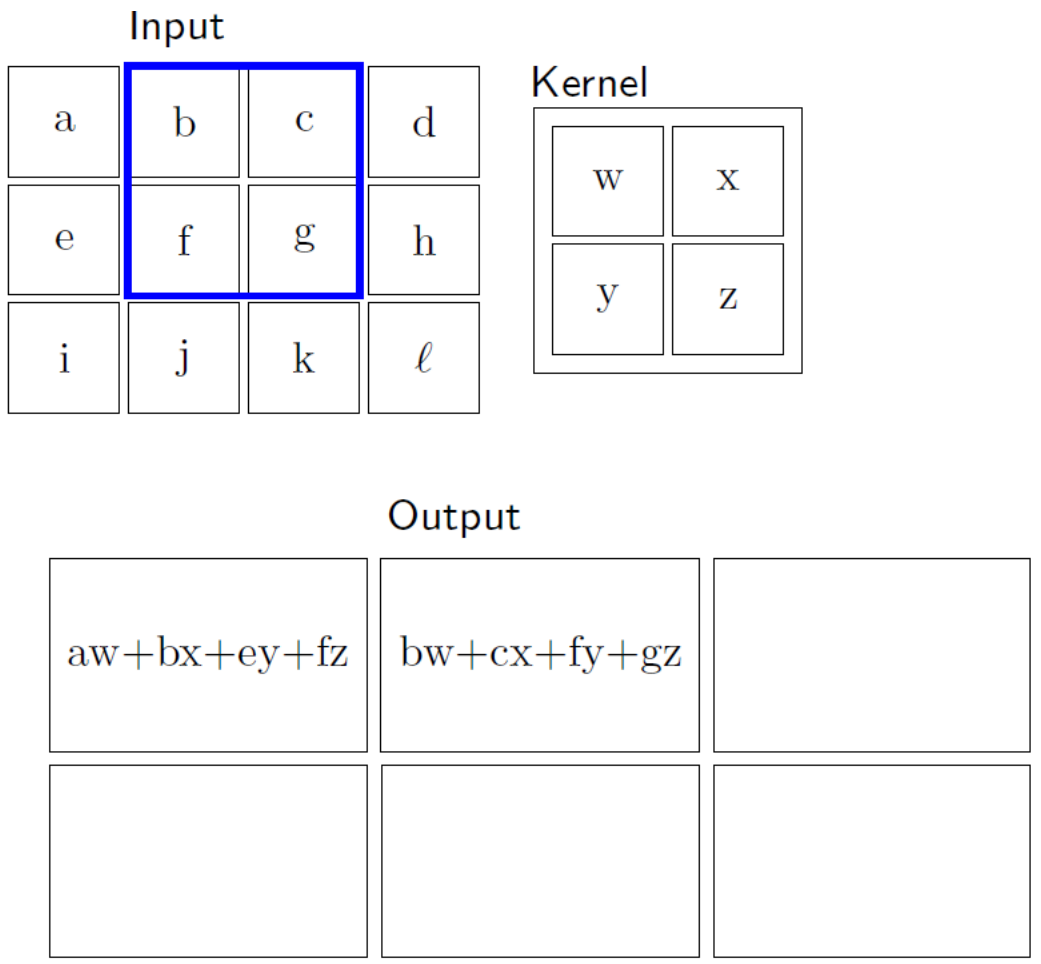
Here's a breakdown of what a convolution operation involves: An example of 2-D convolution without kernel-flipping.

In this case the output to only positions where the kernel lies entirely within the image, called “valid” convolution in some contexts.

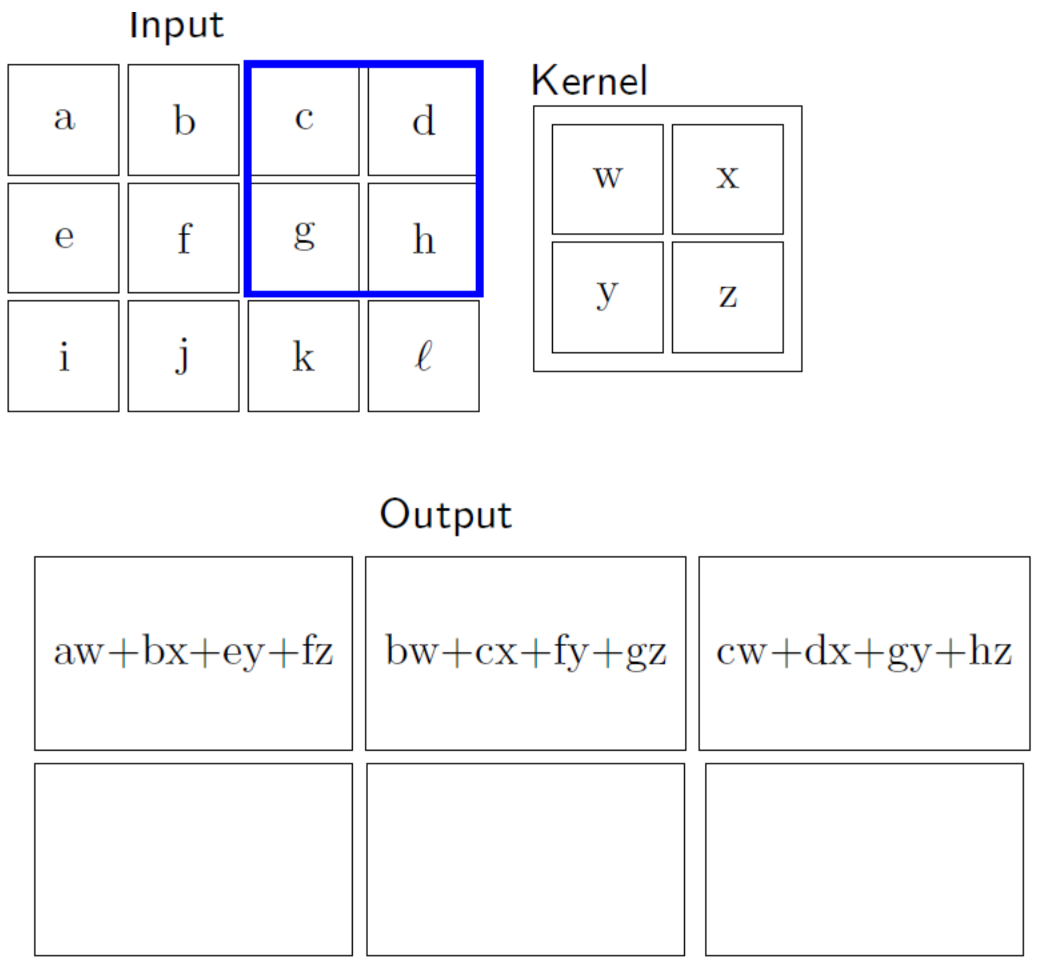
CNN - Convolution



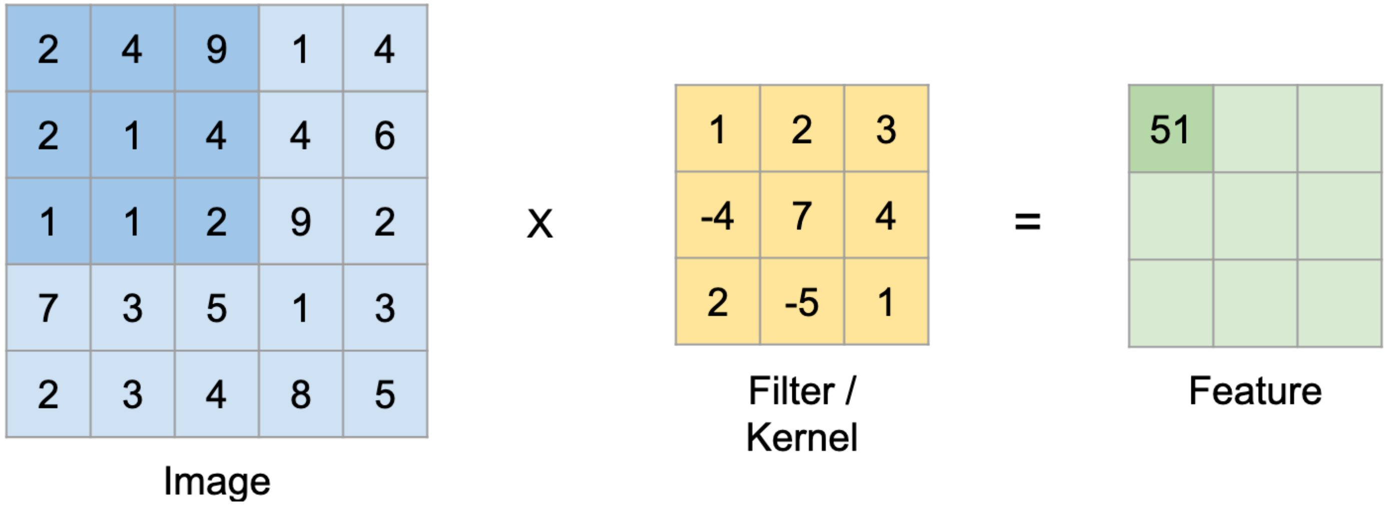
CNN - Convolution



CNN - Convolution



CNN - Convolution



1. Input Data (Image)

The input to the convolution operation is usually a 2D array of pixel values for grayscale images or a 3D array for colored images (with separate channels for Red, Green, and Blue).

2. Kernel (Filter)

A kernel, also known as a filter or feature detector, is a small matrix (usually 3x3, 5x5, etc.) of weights. The kernel slides (or convolves) over the input image to perform the convolution operation.

The values in the kernel are learned during the training process, allowing the network to identify various features like edges, textures, or specific patterns in the input data.

3. Stride

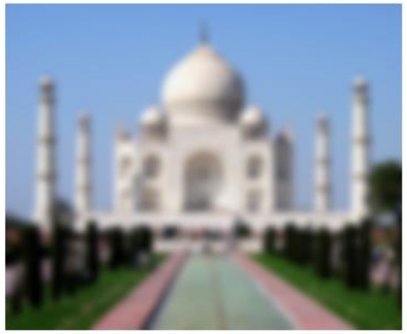
The stride is the step size with which the kernel moves across the input image. A stride of 1 means the kernel moves one pixel at a time, while a stride of 2 means it skips one pixel between each step. Larger strides reduce the spatial dimensions of the output.

CNN - Convolution

4. Padding
Padding involves adding extra pixels (usually zeros) around the border of the input image. This is done to control the spatial dimensions of the output feature map.
Valid Padding: No padding is added, resulting in a smaller output.
Same Padding: Padding is added to ensure the output has the same dimensions as the input.
5. Convolution Process
Sliding the Kernel: The kernel is placed on top of the input image and slides across it, performing element-wise multiplication between the kernel and the corresponding section of the image.
Summation: The results of the element-wise multiplications are summed up to produce a single value, which becomes a pixel in the output feature map.
Repeat: This process is repeated as the kernel slides across the entire image, producing a 2D feature map (or several feature maps if multiple kernels are used).
6. Output (Feature Map)
The result of the convolution operation is a 2D feature map (or 3D for multiple filters), which represents the presence of features detected by the kernel across the image.
The values in the feature map indicate how strongly a feature is present at a particular location in the input.

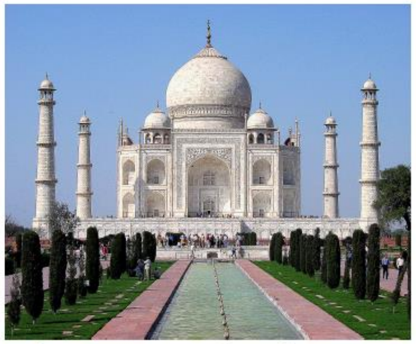
CNN - Convolution

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} =$$



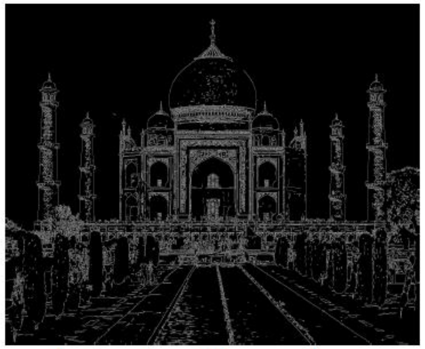
blurs the image

$$* \begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix} =$$



sharpens the image

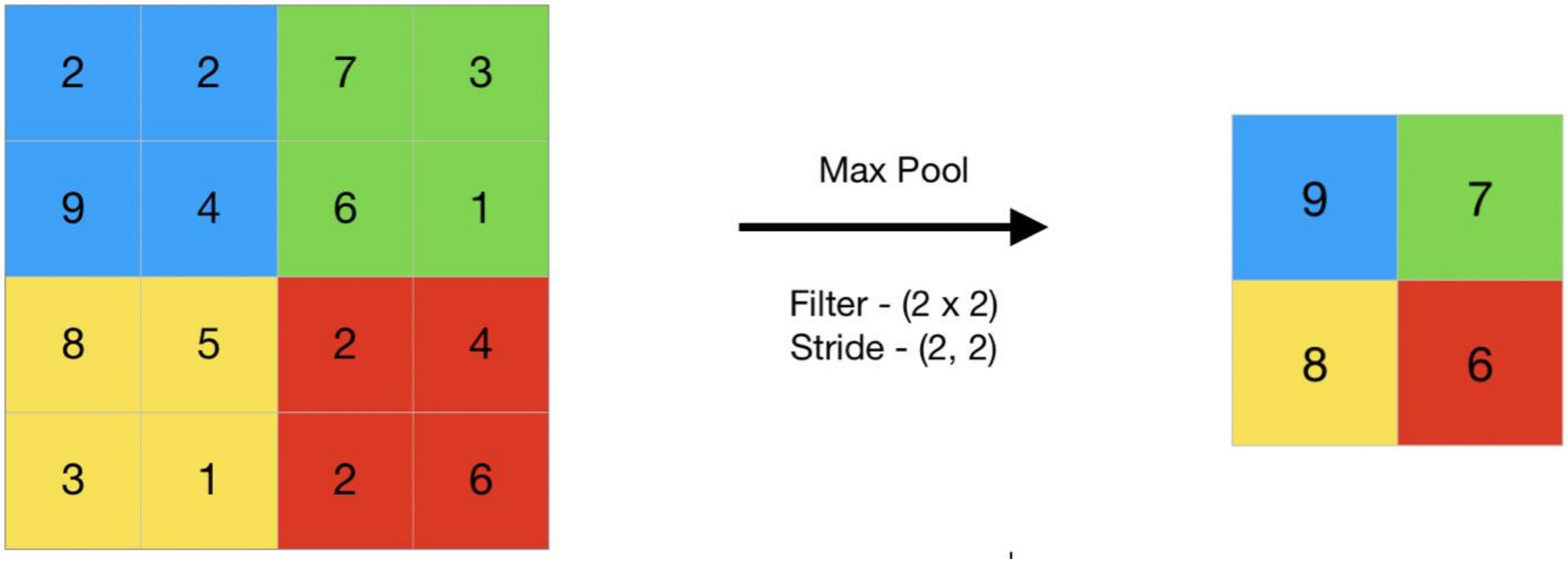
$$* \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} =$$



detects the edges

CNN - Pooling

also known as convolutional neural networks or CNNs.
A Convolutional Neural Network (CNN or ConvNet) is a type of deep learning model specifically designed to process structured grid data, like images, video, and speech. .
the network employs a mathematical operation called convolution



CNN - Pooling

The pooling operation is a crucial step in Convolutional Neural Networks (CNNs) that serves to reduce the spatial dimensions (width and height) of the feature maps generated by convolutional layers. This reduction helps in several ways, including lowering computational complexity, reducing the chances of overfitting, and making the model more invariant to small translations or distortions in the input image.

Key Types of Pooling Operations

Max Pooling

- **Operation**: Max pooling works by selecting the maximum value from each patch of the feature map covered by the pooling window. The window slides over the feature map, and only the largest value within each window is retained.
- **Purpose**: This operation helps to capture the most prominent features (like the strongest edges) in each region, making the model focus on the most important information.
- **Example**: For a 2x2 pooling window, if the input patch is [1, 3; 2, 4], max pooling will return 4.

Average Pooling

- **Operation**: In average pooling, the average of all values in the pooling window is computed and taken as the output for that region.
- **Purpose**: This operation is less aggressive than max pooling, as it retains more information about the region by averaging the values, rather than focusing solely on the maximum.
- **Example**: For a 2x2 pooling window, if the input patch is [1, 3; 2, 4], average pooling will return 2.5.

CNN - Pooling

Global Pooling

Operation: Global pooling reduces each feature map to a single value by taking the maximum or average across the entire map, rather than using a sliding window. This is typically used before the fully connected layers in a CNN.

Purpose: This operation helps to condense the information in each feature map into a single value, which is particularly useful for classification tasks.

Pooling Parameters

Window Size (Pooling Size)

The size of the pooling window (e.g., 2x2, 3x3) determines how much the feature map will be reduced. A 2x2 window reduces the size of the feature map by half in both dimensions.

Stride

The stride determines how far the pooling window moves across the feature map. A stride of 2, for example, moves the window by two units at a time, effectively skipping over some areas and further reducing the output size.

Padding

Padding in pooling operations is less common than in convolution, but it can be used if you want to maintain the spatial dimensions of the input and output.

Purpose and Benefits of Pooling

Dimensionality Reduction: Pooling reduces the spatial dimensions of the feature maps, which decreases the number of parameters and computations in the network.

Translation Invariance: Pooling makes the model less sensitive to small translations or distortions in the input image, as it captures the most dominant features in each region.

CNN - Algorithm

CNN Algorithm

Input Layer:

- **Input**: An image represented as a matrix of pixel values (e.g., 32x32x3 for a 32x32 RGB image).

Convolutional Layer (Conv Layer):

- **Operation**: Apply a set of convolutional filters (kernels) to the input image.
- **Result**: Produces a set of feature maps (also known as activation maps) by sliding the kernels over the input, computing the dot product at each position, and summing up the results.
- **Activation Function**: Apply a non-linear activation function like ReLU (Rectified Linear Unit) to introduce non-linearity.

Repeat:

- **Stride**: Move the filter by a specified number of pixels after each operation (common stride is 1 or 2).
- **Padding**: Optionally add padding to the input to control the spatial dimensions of the output.

Pooling Layer (Subsampling or Downsampling):

- **Operation**: Reduce the spatial dimensions of the feature maps, commonly using Max Pooling or Average Pooling.
- **Max Pooling**: For each sub-region in the feature map, take the maximum value.
- **Result**: A downsampled version of the feature map, reducing computational complexity and controlling overfitting.

CNN - Algorithm

Additional Conv & Pooling Layers (optional):

- **Purpose**: Further extract hierarchical features from the input by adding more convolutional and pooling layers.
- **Deeper Networks**: Typically, deeper CNNs with multiple layers can learn more abstract features.

Flattening:

- **Operation**: Convert the 2D feature maps into a 1D vector.
- **Result**: This vector serves as the input for the fully connected layers.

Fully Connected Layer (Dense Layer):

- **Operation**: Perform traditional neural network operations where each neuron is connected to every neuron in the previous layer.
- **Activation Function**: Often ReLU, Sigmoid, or Softmax (for classification tasks).

Output Layer:

- **Operation**: The final dense layer produces the output, typically using Softmax for multi-class classification.
- **Output**: A vector of probabilities representing the likelihood of each class.

CNN - Algorithm

Loss Function:

- **Common Choices**: Cross-entropy loss for classification tasks.
- **Purpose**: Measure the difference between the predicted output and the actual labels.

Backpropagation and Optimization:

- **Backpropagation**: Compute gradients of the loss with respect to all weights in the network and propagate them backward through the network.
- **Optimization**: Update the weights using an optimization algorithm like Stochastic Gradient Descent (SGD) or Adam to minimize the loss function.

Training:

- **Epochs**: Train the model over several epochs, repeatedly updating weights using the backpropagation algorithm.
- **Batch Size**: Process the input data in batches to manage memory usage and improve convergence.

Evaluation:

- **Validation**: After training, evaluate the model's performance on a separate validation set to fine-tune hyperparameters and prevent overfitting.
- **Testing**: Finally, assess the model on an unseen test set to determine its generalization ability.