CNN

Convolutional Neural Networks

Short Notes By Shivani Bhatia

1. Basics of CNNs

- Purpose: CNNs are designed for processing structured grid data like images.
- Applications: Primarily in computer vision tasks—object recognition, segmentation, image classification, etc.
- Benefit: Ability to capture spatial features through convolutions.
- Structure: Convolutional layers, pooling layers, fully connected layers.

2. Architecture of CNNs

- Layers in CNNs:
  - Input Layer: Takes in image data (height x width x channels).
  - Convolutional Layer (Conv Layer): The core of a CNN where filters (kernels) slide across the input to produce feature maps. It detects local patterns.
  - Activation Layer: Often uses ReLU (Rectified Linear Unit) to introduce non-linearity, helping the network learn complex patterns.
  - Pooling Layer: Reduces dimensionality by summarizing feature maps, commonly using Max Pooling or Average Pooling.
  - Fully Connected (FC) Layer: The final layers, usually fully connected to classify extracted features.
  - Output Layer: For classification, often uses Softmax activation for probability output across classes.

3. Convolutional Layer

- Filters/Kernels: Small, learnable matrices that slide (convolve) across input data. Common sizes: 3x3, 5x5.
- Stride: The step size of the filter as it moves across the input. Higher stride reduces the spatial dimensions.
- Padding:
  - Valid Padding (No padding): Reduces the spatial dimension.
  - Same Padding (Zero padding): Maintains the spatial dimension.
- Output Shape: Determined by filter size, stride, and padding.

4. Pooling Layer

- Purpose: Down-sampling to reduce dimensionality and computation, while retaining important features.
- Types:
  - Max Pooling: Takes the maximum value from each patch of the feature map.
  - Average Pooling: Takes the average value from each patch.
- Pooling Size: Commonly 2x2 with a stride of 2.

5. Activation Functions

- ReLU (Rectified Linear Unit): f(x)=max(0,x)f(x) = /max(0, x)f(x)=max(0,x); helps prevent vanishing gradients and speeds up training.
- Sigmoid / Tanh: Less common in CNNs as they can suffer from vanishing gradients.
- Softmax: Often used in the final layer for multi-class classification.

6. Important Hyperparameters

- Filter Size: Determines the receptive field. Typical sizes are 3x3, 5x5.
- Number of Filters: More filters capture more complex patterns but increase computation.
- Stride and Padding: Affect the spatial dimensions of output.
- Learning Rate: Step size for gradient descent. Can vary using learning rate schedulers.

7. Training and Regularization Techniques

- Backpropagation: Used to adjust filter weights based on error gradients.
- Batch Normalization: Normalizes inputs to layers, stabilizing learning and accelerating convergence.
- Dropout: Randomly drops neurons during training to prevent overfitting.
- Data Augmentation: Enhances generalization by modifying training images (rotation, flipping, etc.).

8. Popular CNN Architectures

- LeNet-5: Early CNN model, mainly used for digit recognition.
- AlexNet: First deep CNN model that popularized deep learning in computer vision (ImageNet winner).
- VGG: Introduced deep networks with 3x3 filters. Known for simplicity but computationally expensive.
- ResNet: Introduced Residual Blocks to address vanishing gradient problems in very deep networks.
- Inception (GoogLeNet): Introduced Inception Modules, which use filters of multiple sizes in parallel.

9. Concepts of Transfer Learning and Fine-tuning

- Transfer Learning: Using a pre-trained model on a new but similar task.
- Fine-tuning: Adjusting the weights of a pre-trained model slightly to fit the new dataset.
- Feature Extraction: Using earlier layers of a pre-trained CNN as a fixed feature extractor.

10. Advanced Concepts

- Residual Blocks (ResNet): Uses skip connections to bypass one or more layers, helping mitigate vanishing gradients.
- Depthwise Separable Convolution (MobileNet): Reduces computation by breaking down convolution into two simpler operations.

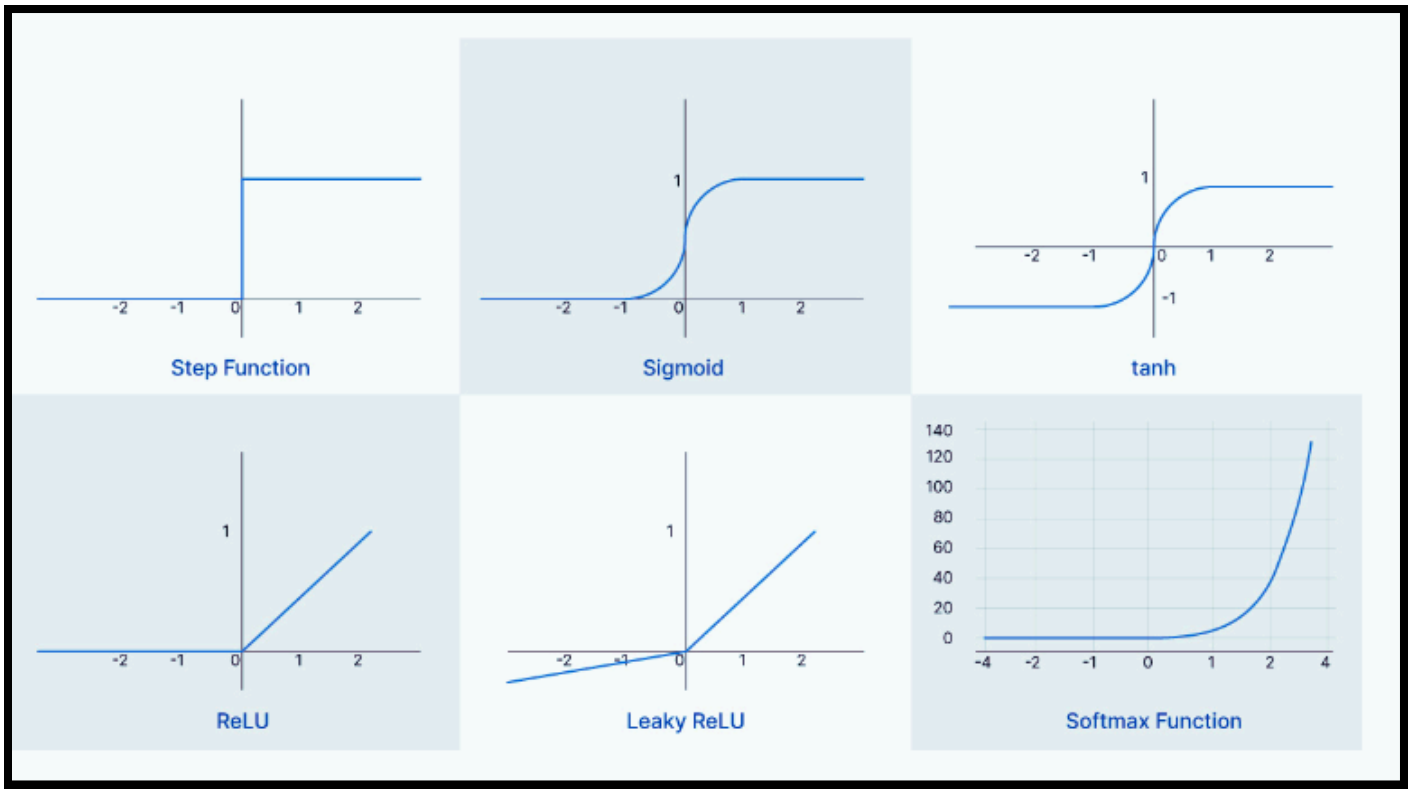
- Dilated Convolutions: Expands the receptive field without increasing computation, useful for tasks like segmentation.

11. Evaluation Metrics

- Accuracy: Common metric but less useful in imbalanced data.
- Precision, Recall, F1 Score: Important for classifying classes with imbalance.
- Confusion Matrix: Shows true/false positives and negatives, useful in classification tasks.
- AUC-ROC Curve: Evaluates classifier performance across different thresholds.

12. Other Major Concepts

- Vanishing Gradient
  - Vanishing Gradient Problem: Gradients become very small during backpropagation in deep CNNs.
  - Causes: Common with activation functions like sigmoid or tanh that can saturate.
  - Effects: Slow or stalled learning for earlier layers; difficulty in capturing long-range dependencies.
  - Mitigation Techniques:
  - Use ReLU (Rectified Linear Unit) activations.
  - Implement batch normalization.
  - Employ architectures like LSTMs for sequential data.
  - Use residual connections (e.g., in ResNet) to maintain gradient flow.
- Important Diagrams


Step Function
Sigmoid
tanh
ReLU
Leaky ReLU
Softmax Function

CNN Numerical Formulas Cheat Sheet

Key Formulas for Convolutional Neural Networks (CNN)

1. Output Shape Calculation for Convolutional Layers

Formula:

Output Height (H_out) = (H - F + 2P) / S + 1

Output Width (W_out) = (W - F + 2P) / S + 1

Output Depth = Number of Filters

Where:

- H, W: Height and Width of input
- F: Filter size
- P: Padding
- S: Stride

Example:

Input: 32x32x3, Filter: 3x3, Stride: 1, Padding: 1, Filters: 64

Output Shape = 32x32x64

2. Output Shape Calculation for Pooling Layers

Formula:

Output Height (H_out) = (H - F) / S + 1

Output Width (W_out) = (W - F) / S + 1

Depth remains unchanged.

Example:

Input: 32x32x64, Pooling Size: 2x2, Stride: 2

Output Shape = 16x16x64

3. Parameter Calculation for Convolutional Layers

Formula:

Total Parameters = (F * F * D + 1) * Number of Filters

Where:

- F: Filter size

- D: Input depth

- +1 for each filter's bias term

Example:

Filter: 3x3, Input Depth: 3, Number of Filters: 64

Total Parameters = (3 * 3 * 3 + 1) * 64 = 1792

4. Parameter Calculation for Fully Connected Layers

Formula:

Total Parameters = Input Size * Output Size + Output Size (for bias)

Example:

Input Size: 4096, Output Size: 1000

Total Parameters = 4096 * 1000 + 1000 = 4,097,000

5. Receptive Field Calculation

Formula:

Receptive Field = Previous Layer RF + (Kernel Size - 1) * Stride of Previous Layer

Example:

Layer 1: 3x3, Stride 1 -> RF = 3x3

Layer 2: 3x3, Stride 1 -> RF = 5x5

Layer 3: 3x3, Stride 2 -> RF = 11x11