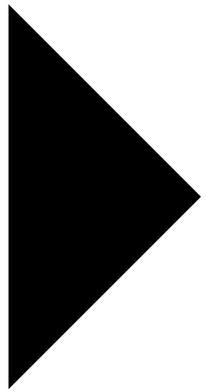
ADVANCED ARTIFICIAL INTELLIGENCE

This content is for revision purposes only.
Please refer to your official course materials for complete details.

Prepared by: Samah Al-hilale

OUTLINE

1. CONVOLUTIONAL NEURAL NETWORK
2. DATA HANDLING & AUGMENTATION
3. AUTOENCODERS
4. IMAGE SEGMENTATION
5. OBJECT DETECTION
6. ADVANCED TRAINING TECHNIQUES
7. SHORT QUIZ



CONVOLUTIONAL NEURAL NETWORK

STEPS:

1. **Data Preparation:**
   - Load and preprocess images
   - Resize all images to a fixed size
   - Apply data augmentation (if needed)
   - Convert images to tensors
   - Split into training and testing sets
   - Load data into a DataLoader
2. **Convolutional Layers:**
   - **Input Layer: Its Shape** should be: batch_size, channels, height, width]

3. Classifier:

- Flatten (convert feature maps to a vector): Its Shape should be: [batch_size, features]
- Fully Connected Layer
- BatchNorm (normalize activations)
- ReLU

5. Training:

- Loss: CrossEntropy
- Optimizer: Adam / SGD and so on....

6. Evaluation:

- Compute accuracy & loss Validate on test data

UNDERSTANDING THE IMAGES

A Deep Neural Network (DNN) treats an image as a 1D vector by flattening it, which leads to several problems:

- **Loss of spatial relationships:**

A cat's eye and nose are in specific positions, but DNNs treat them as justnumbers without spatial awareness.

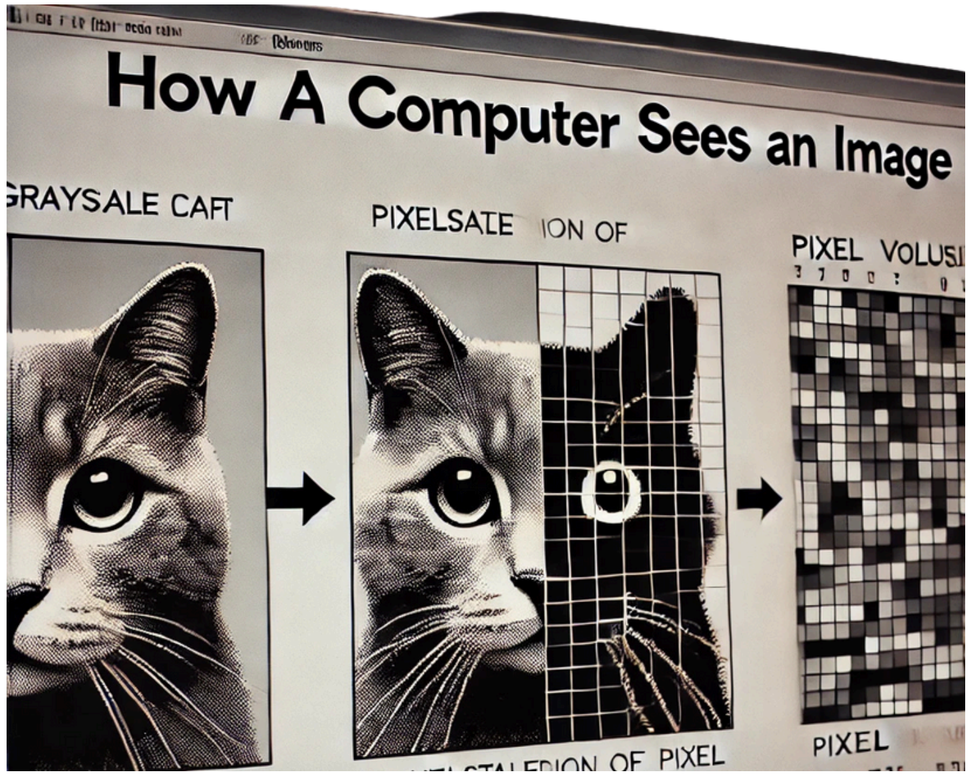
- **Too many parameters:**

A simple 100x100 image has 10,000 pixels → If fully connected to just one hidden layer of 256 neurons, that’s 2.56 million weights to train!

- **Inefficient feature detection:**

DNNs don’t detect edges, shapes, or textures well, since they learn from raw pixel intensities rather than meaningful patterns.

WHY CNNS (CONVOLUTIONAL NEURAL NETWORKS) ARE BETTER



A computer doesn’t see images like humans do. Instead, it perceives them as matrices of pixel values.

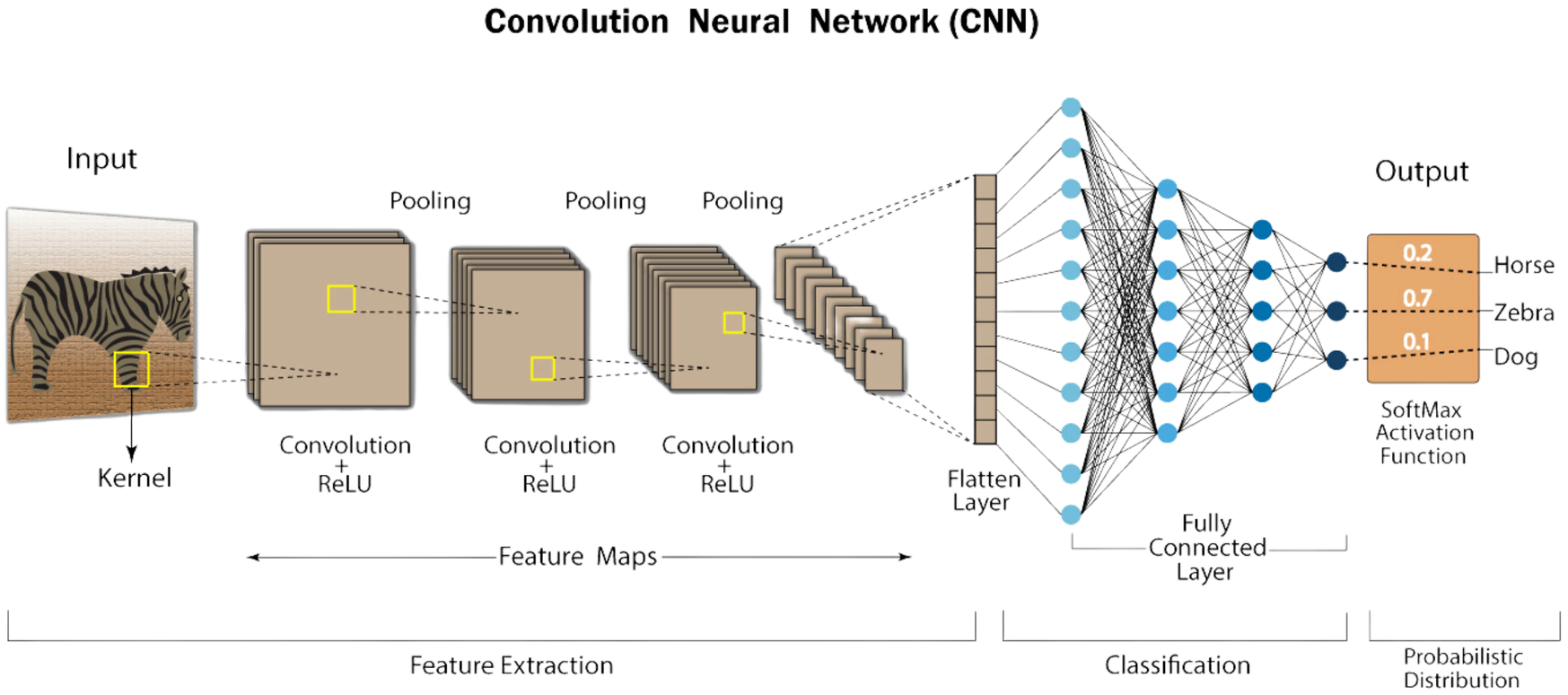
- **Grayscale images:** Each pixel has a single value (0-255) representing intensity.
- **Color images (RGB):** Each pixel has three values (Red, Green, and Blue), forming a 3D matrix.

WHY CNNS (CONVOLUTIONAL NEURAL NETWORKS) ARE BETTER

CNNs solve these problems by preserving spatial structure and reducing computational cost through local feature extraction:

- **Convolutional Layers:** Instead of connecting every pixel to every neuron, CNNs scan images using small filters (kernels) that detect edges, textures, and patterns.
- **Weight Sharing (Fewer Parameters):** Filters slide across the image and apply the same weights everywhere, greatly reducing the number of parameters.
- **Pooling (Downsampling):** CNNs use MaxPooling or AveragePooling to shrink images while keeping the important features, making computations faster.
- **Better Generalization:** Since CNNs focus on features instead of individual pixel values, they generalize better to different lighting, angles, and distortions.

WHAT IS A CNN?


Convolution Neural Network (CNN)
Input
Kernel
Pooling
Pooling
Pooling
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Feature Maps
Flatten Layer
Fully Connected Layer
Output
0.2
0.7
0.1
Horse
Zebra
Dog
SoftMax Activation Function
Feature Extraction
Classification
Probabilistic Distribution

WHAT IS A CNN?

Convolutional Layers:

1. Padding (Controls Image Size)

- **Valid Convolution** (No Padding, padding = 0)
- **Same Convolution** (With Padding)

2. Stride (Controls Movement of Filters)

- **Stride = 1 →** Moves 1 pixel at a time (detailed feature extraction).
- **Stride > 1 →** Moves more pixels (reduces output size, speeds up computation).

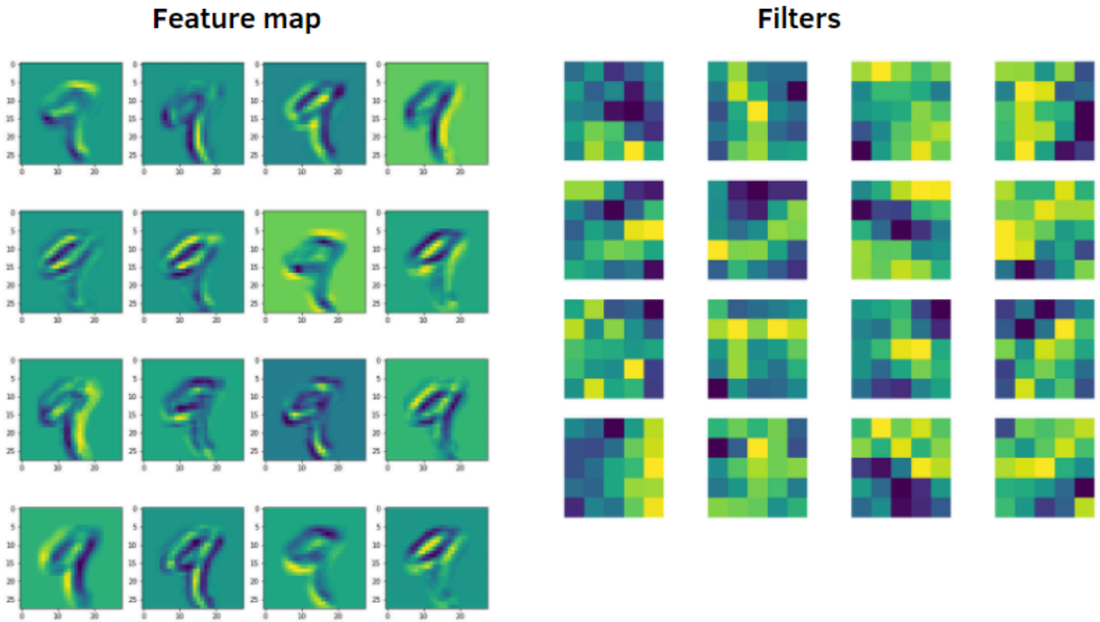
3. Kernel Size (Filter Size) (Controls Receptive Field

- Determines the area of the image the filter scans at each step.
- **Common sizes:** 3x3, 5x5, 7x7.
- Smaller filters capture fine details, larger filters capture broader patterns.
- **More filters →** More diverse feature extraction.

5. Pooling (Downsampling)

- **MaxPooling →** Keeps the most important feature.
- **AveragePooling →** Takes the average of pixel values.
- **Helps reduce size while keeping essential information.**

WHY ARE FILTERS USUALLY ODD-SIZED?



1. Avoids Asymmetric Padding
2. Has a Central Pixel
3. Consistent Striding & Pooling

WHAT IS A CNN?

Convolution Output Size:

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

Pooling Layer Output Size:

For MaxPooling or AveragePooling, the output size is given by:

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

- k = Pooling window size
- s = Stride
- No padding is applied in pooling layers.

the Number of Parameters in a Convolutional Layer:

TotaTotal Parameters = (Input Neurons × Output Neurons × number of filters)+Biases

- Number of filters (Number of filters = number of bias

Number of Parameters in a Fully Connected (Dense) Layer:

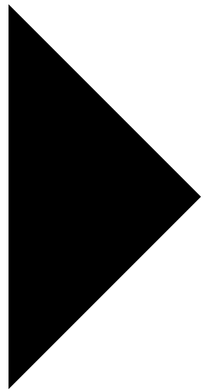
Total Parameters = (Input Neurons×Output Neurons)+Biases

TRY TO CACULATE THESE OUTPUTS USING THE PREVIOUS FOMRULAS

Layer name	Output shape	Parameter
Conv2D	50×48×32	2432
MaxPooling2D	25×24×32	0
Conv2D	25×24×64	18496
MaxPooling2D	12×12×64	0
Conv2D	12×12×96	55392
MaxPooling2D	6×6×96	0
Conv2D	6×6×96	83040
MaxPooling2D	3×3×96	0
Flatten	864	0
Dense	512	442880
Activition	512	0
Dense	15	7695
Classifier		softmax

POPULAR CNN ARCHITECTURES

- **AlexNet:** First deep CNN to show **ReLU** + **Dropout** + **GPU training**.
- **VGGNet:** Used **small 3x3 filters** with deep networks.
- **ResNet:** Introduced **skip connections** to combat vanishing gradients.
- **EfficientNet: Optimized scaling** of depth, width, and resolution.
- **InceptionNet:** Used **multiple kernel sizes** to capture different features in parallel.



DATA HANDLING & AUGMENTATION

STEPS:

1. Create a Custom Dataset Class:

- **Use __init__(): l**oad file paths and labels, Store the transformation pipeline.
- **__len__():** returns the total number of images in the dataset.
- **__getitem__():** Load the image based on the given index, then returns the image and corresponding label.

2 Define Training and Testing Transforms

- Training Transform (Includes Augmentation):
- Testing/Validation Transform (No Augmentation, Just Resizing & Normalization).

3. Split Dataset into Train, Validation, and Test Sets.

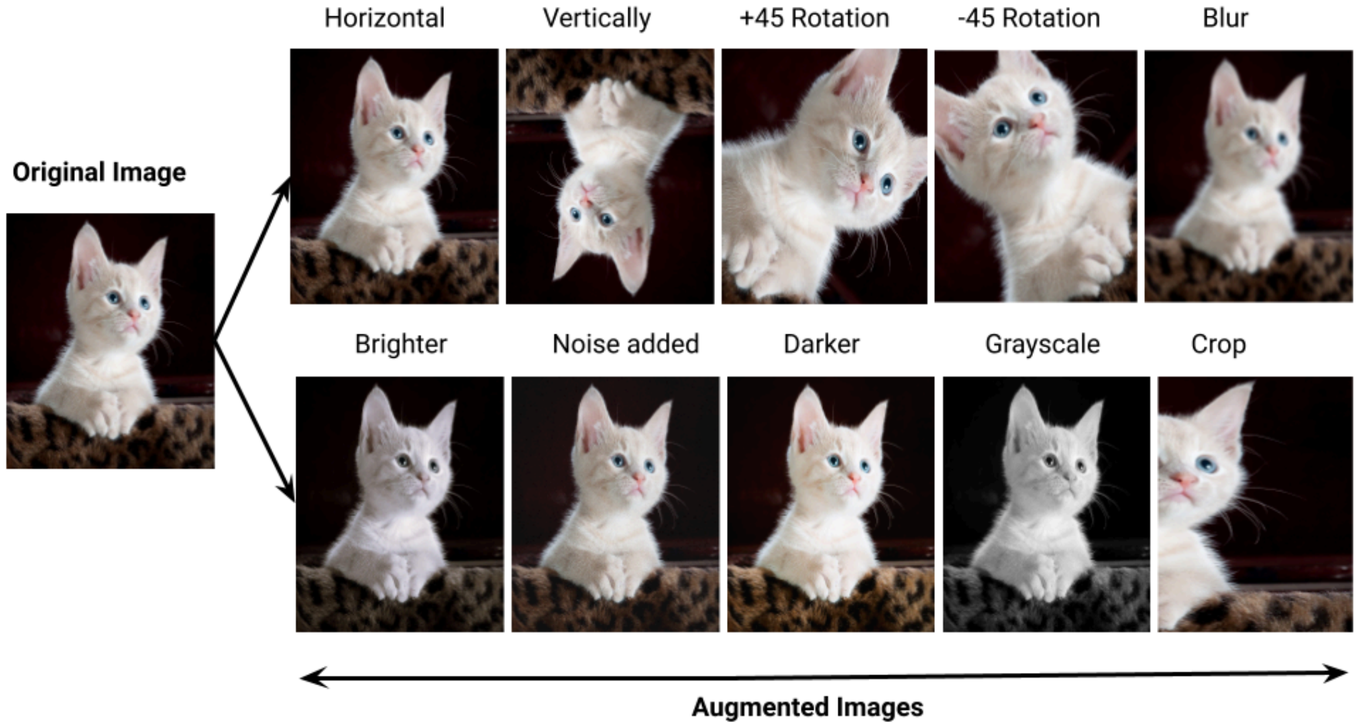
4. Load Data into a DataLoader.

DATALOADERS

Instead of feeding the entire dataset at once, we split it into smaller batches to make training efficient and manageable.

- **Efficient Data Loading:** Instead of manually loading images one by one, DataLoader automates the process.
- **Shuffling:** to ensure the model doesn't memorize sequences.
- **Batch Processing:** Automatically groups data into batches for training.
- **Parallel Loading (num_workers):** Uses multiple CPU threads to load data faster while training.

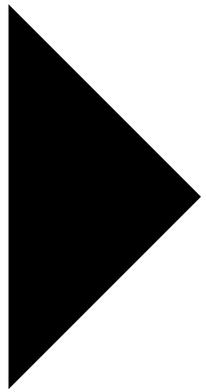
DATA AUGMENTATION



- Helps **improve model performance** and generalization by **creating modified versions of training images.**
- **Prevents** the model from **memorizing specific patterns.**

TRANSFER LEARNING

- **Allows us to use a pretrained model**, meaning the model already has previous knowledge from training on a large dataset (like ImageNet). Instead of training a model from scratch, **we use this existing knowledge for a new task.**
- **Fine tuning is one of the transfer learning methods**: Instead of using the pretrained model as-is, we **allow some layers to be updated during training.**



AUTOENCODERS

STEPS:

1. Data Preparation

2. Encoder (Feature Extraction & Compression):

- **Convolutional Layers**
- **Flatten (Convert to latent space): Shape →** [batch_size, latent_features].

3. Bottleneck (Latent Representation):

- The compressed representation (smallest dimensional feature map).
- Fully Connected Layer to learn a compact, meaningful feature space.

4. Decoder (Reconstruction of Image):

- **Fully Connected Layer:** Expands from latent space to feature maps.

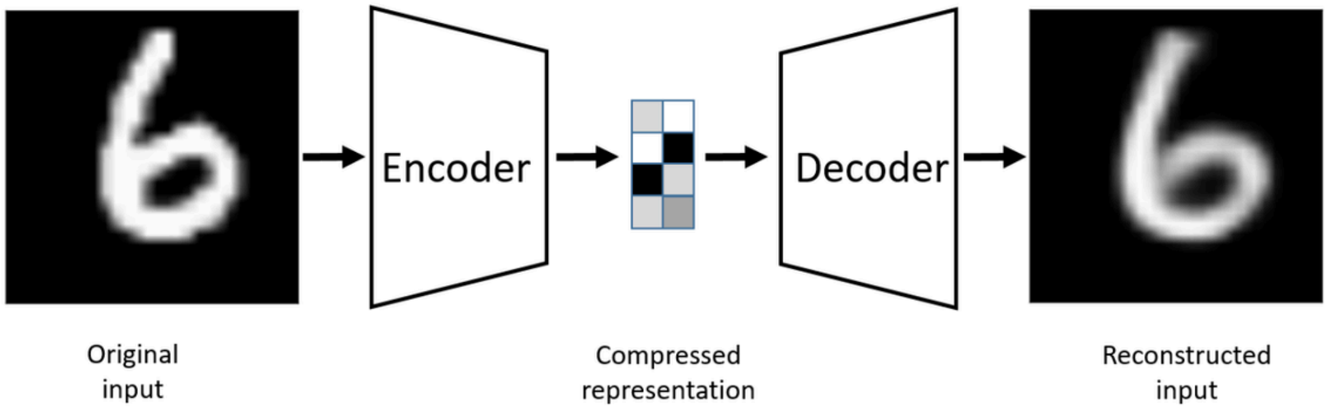
5. Training:

- Loss: Mean Squared Error (MSE) or Binary CrossEntropy (BCE)
- Optimizer: Adam / SGD and so on....

6. Evaluation

- Compute accuracy & loss Validate on test data

WHAT IS AN AUTOENCODER?



An Autoencoder (**AE**) is a type of neural network that learns to compress and reconstruct data **without needing labeled examples,** making it a **fully unsupervised** learning technique.

WHAT IS AN AUTOENCODER?

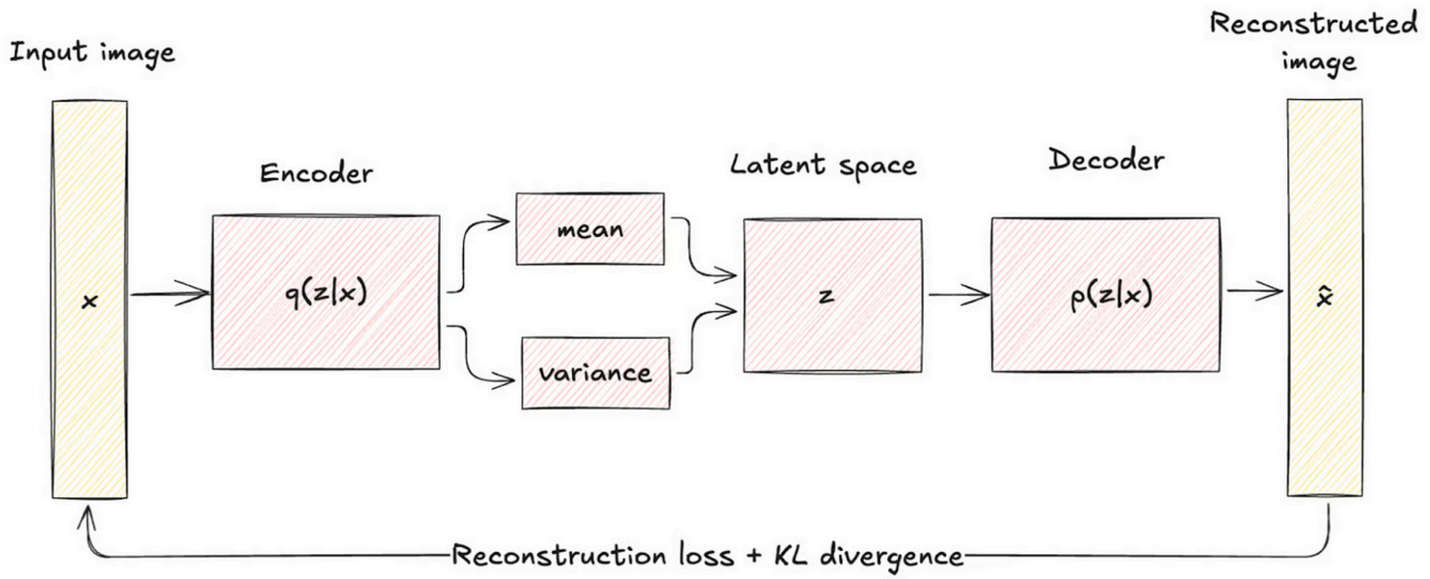
Main Components of an Autoencoder:

- **Encoder: compresses input** into a lower-dimensional representation.

- **Bottleneck: represents the essential features** of the data and acts as a bridge between encoding and decoding.

- **Decoder: reconstructs** the original input from the bottleneck representation, tries to make the output as close as possible to the input.

WHAT IS A VARIATIONAL AUTOENCODER?



A Variational Autoencoder (VAE) is an **extension of the standard autoencoder but with a key difference:** instead of learning a fixed latent representation, **it learns a probabilistic distribution over the latent space**.

WHAT IS A VARIATIONAL AUTOENCODER?

Main Components of a Variational Autoencoder:

- **Encoder (Latent Distribution Learning):** Unlike a regular autoencoder,

the encoder outputs two vectors:

1. M**ean (μ):** Represents the **center of the distribution.**
2. **Variance (σ2):** Represents the **spread of the distribution.**

- These values **define a Gaussian distribution in the latent space instead of a single point.**

- **Reparameterization Trick (Sampling from Latent Space):** Instead of directly using μ and σ2 , we sample a latent vector: $z=\mu+\sigma \cdot \epsilon$

- **Decoder (Reconstructing the Input):** Uses the sampled z to reconstruct the input, just like a standard autoencoder.

WHAT IS A VARIATIONAL AUTOENCODER?

Loss Function in Variational Autoencoder:

VAEs optimize a combination of two losses:

1. **Reconstruction Loss (Mean Squared Error / Binary Cross Entropy):**
   - Ensures the output is similar to the input.

2. **KL Divergence Loss (Kullback-Leibler Divergence):**
   - Ensures the latent space follows a Gaussian distribution.
   - Prevents the model from overfitting to specific data points.

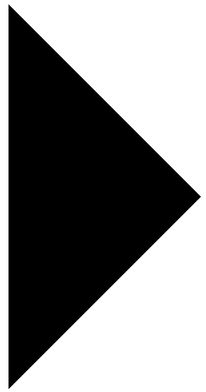


IMAGE SEGMENTATION

WHAT IS IMAGE SEGMENTATION ?

- It is the **process of dividing an image into meaningful parts.**
- Often used in **object detection, medical imaging, and autonomous driving**.
- It involves **classifying each pixel into different categories** rather than classifying the entire image as one object.

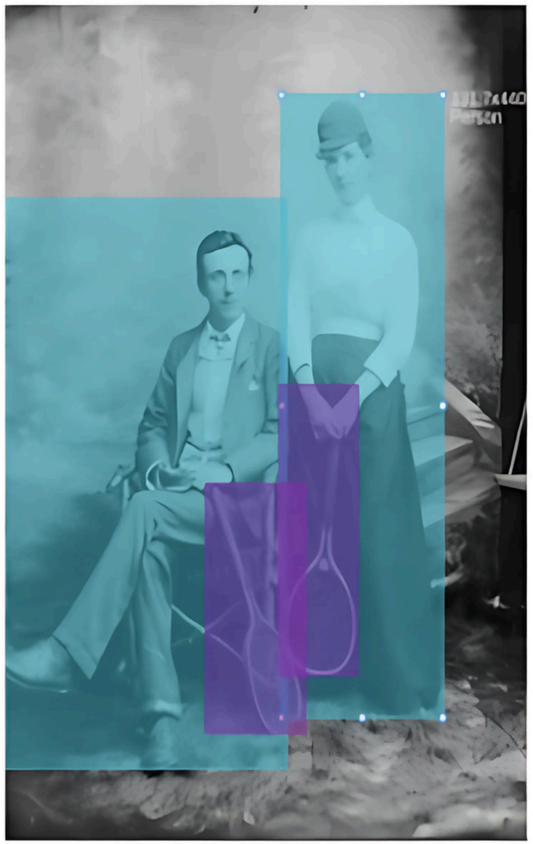
TYPES OF IMAGE SEGMENTATION

1. Semantic Segmentation: Classifies each pixel into **a category** (e.g., car, road, sky) without distinguishing different objects of the same class.

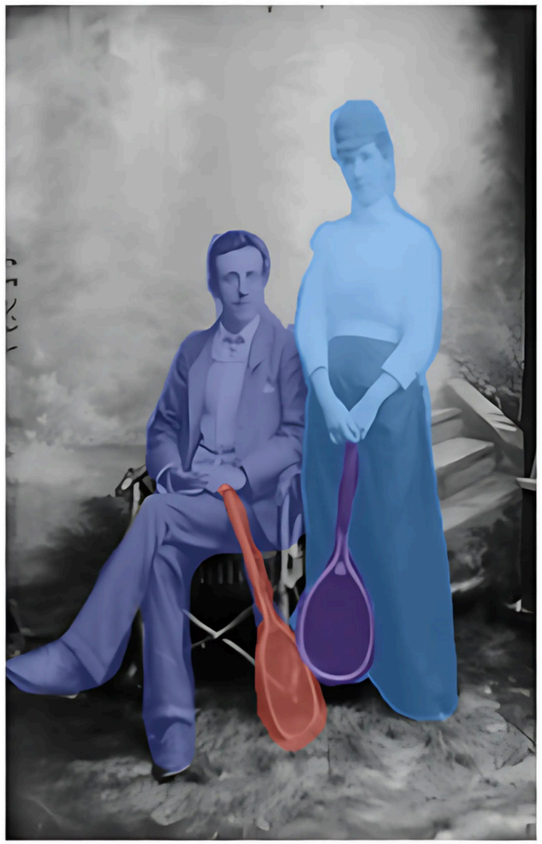
2. Instance Segmentation: Differentiates between objects of the same class (e.g., separates multiple cars in an image)

3. Panoptic Segmentation: A combination of semantic segmentation + instance segmentation. **It labels both stuff** (**backgrounds like roads, sky**) **and things** (**objects like cars, people**).

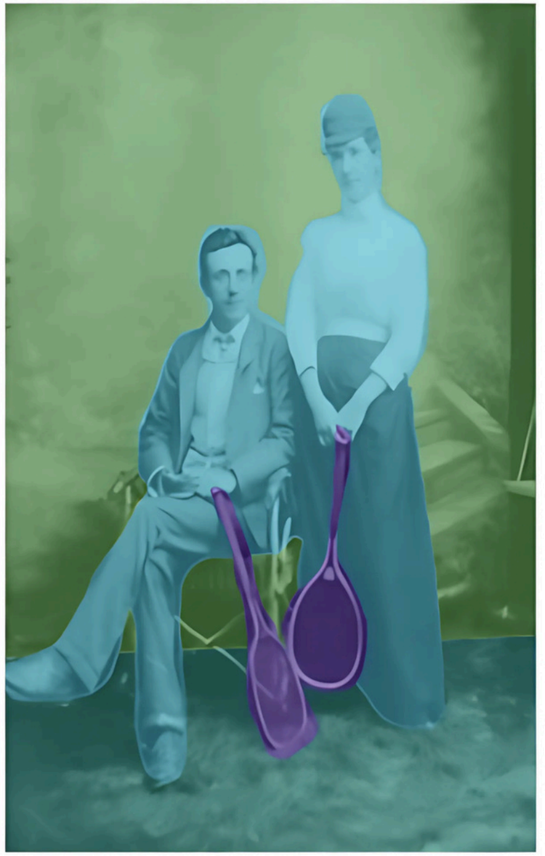
TYPES OF IMAGE SEGMENTATION



Detection
(Bounding Boxes)

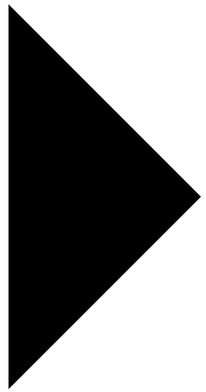


Instance Segmentation



Semantic Segmentation

Panoptic Segmentation

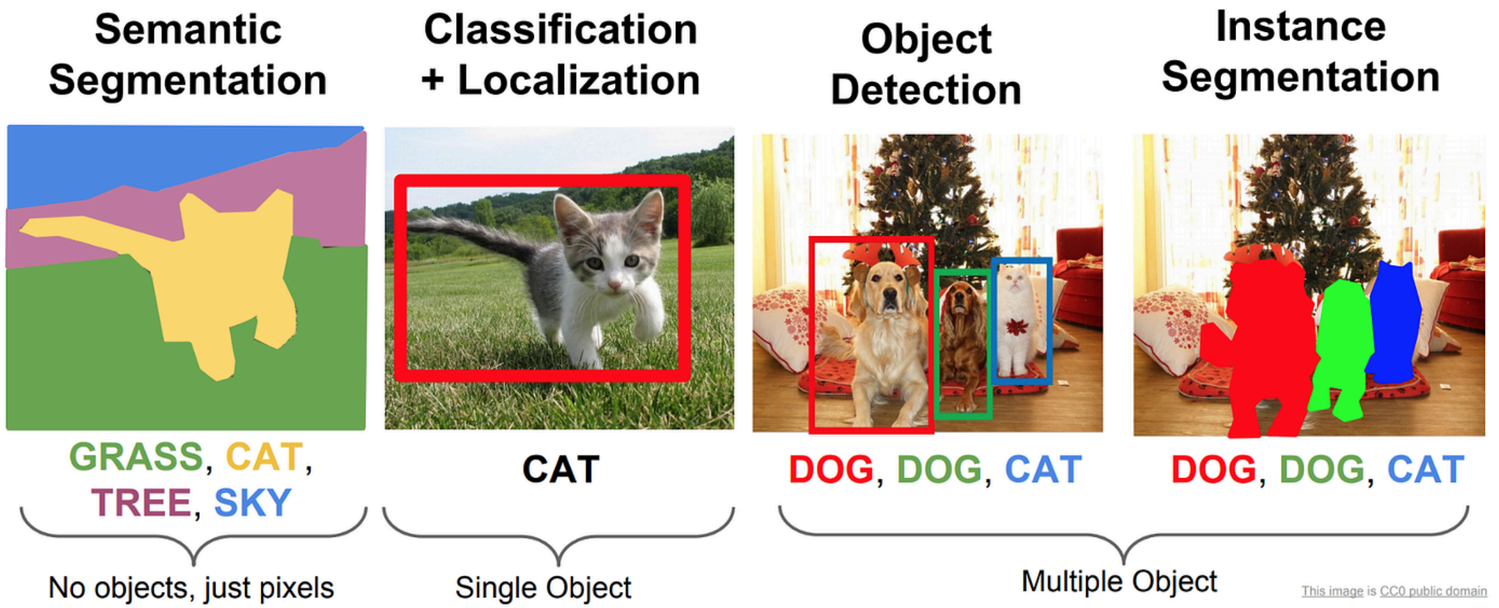


OBJECT DETECTION

STEPS:

1. **Load and preprocess data:**
   - **__init__():** Load image file paths, Load bounding box annotations (e.g., COCO, VOC, or custom format) , Store transformations.
   - **__len__():** Return the total number of images in the dataset.
   - **__getitem__():** Load an image and its corresponding bounding boxes, Apply transformations, Return the image, bounding boxes, and labels.
2. **Define Training and Testing Transforms, Split Dataset into Train, Validation, and Test Sets, Load Data into a DataLoader**
5. **Define Object Detection Model**
6. **Loss:**
   - Use a combination of classification loss and bounding box regression loss.
7. **Optimizer:**
   - Adam or SGD with weight decay.
8. **Train the Model**
9. **Evaluation**

WHAT IS OBJECT DETECTION?

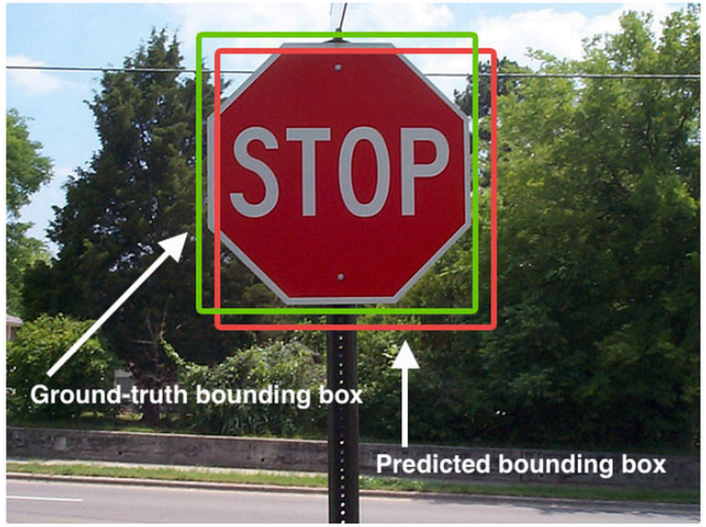


Unlike image classification, which assigns a single label to an entire image, object detection performs:

1. **Classification:** Identifies what the object is.
2. **Localization:** Determines where the object is in the image by drawing abounding box.

WHAT IS OBJECT DETECTION?

To evaluate object detection models, we use:



$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Intersection over Union (IoU): Measures how much the predicted bounding box overlaps with the ground truth box.

Mean Average Precision (mAP): Evaluates detection accuracy across different IoU thresholds.

Non-Maximum Suppression (NMS): Removes redundant bounding boxes and keeps only the highest-confidence detections.

TYPES OF OBJECT DETECTION MODELS

There are two main types of object detection models:

Two-Stage Detectors (More Accurate, Slower):

1. **R-CNN (Region-Based CNN):**
   - **Step 1:** Uses Region Proposal Networks (RPN) to generate potential bounding boxes.
   - **Step 2:** Extracts features from each region using a CNN.
   - **Step 3:** Classifies the objects and refines bounding boxes.

Limitation: Slow, requires processing each proposed region individually.

2. R-CNN (Region-Based CNN):

- **Step 1:** Extract features using a CNN.
- **Step 2: Use RPN to propose bounding boxes.**
- **Step 3:** Classify and refine bounding boxes.

Advantage: More efficient and accurate than R-CNN.

Limitation: Still slower than one-stage detectors like YOLO.

TYPES OF OBJECT DETECTION MODELS

There are two main types of object detection models:

One-Stage Detectors (Faster, Real-Time): These models skip the region proposal step and directly predict bounding boxes and object classes in a single pass.

1. **YOLO (You Only Look Once)**

How It Works:

- **Step 1:** Splits the image into a grid.
- **Step 2:** Each grid cell predicts bounding boxes and class probabilities.
- **Step 3:** Merges overlapping detections using Non-Maximum Suppression (NMS).
- **Advantages:**
- Very fast, ideal for real-time applications (self-driving cars, surveillance).
- Single-step processing → Faster than R-CNN & Faster R-CNN.
- **Limitations:**
- Struggles with small objects (e.g., detecting birds in the sky).
- Struggles with overlapping objects (e.g., cars close together).

TYPES OF OBJECT DETECTION MODELS

There are two main types of object detection models:

One-Stage Detectors (Faster, Real-Time): These models skip the region proposal step and directly predict bounding boxes and object classes in a single pass.

2. **SSD (Single Shot MultiBox Detector)**

How It Works:

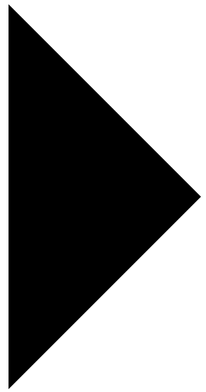
- **Step 1:** Predicts multiple bounding boxes at different scales directly from feature maps.
- **Step 2:** Uses anchor boxes (predefined sizes) to detect objects of various sizes.
- **Advantages:**
- Faster than Faster R-CNN but slightly slower than YOLO.
- Works better with small objects than YOLO.
- **Limitations:**
- Less accurate than Faster R-CNN for complex scenes.

TYPES OF OBJECT DETECTION MODELS

If you need **real-time detection** → Use **YOLO**

If you want **the highest accuracy** → Use **Faster R-CNN**

If you need a **balance of speed and accuracy** → Use **SSD**



ADVANCED TRAINING TECHNIQUES

REGULARIZATION METHODS

- **Dropout:** Prevents Overfitting by Randomly Deactivating Neurons.

When to Use:

- In fully connected (dense) layers of deep networks.
- When overfitting is detected (train accuracy high, validation accuracy low).

- **Batch Normalization:** Normalizes Layer Inputs to Stabilize Training

When to Use:

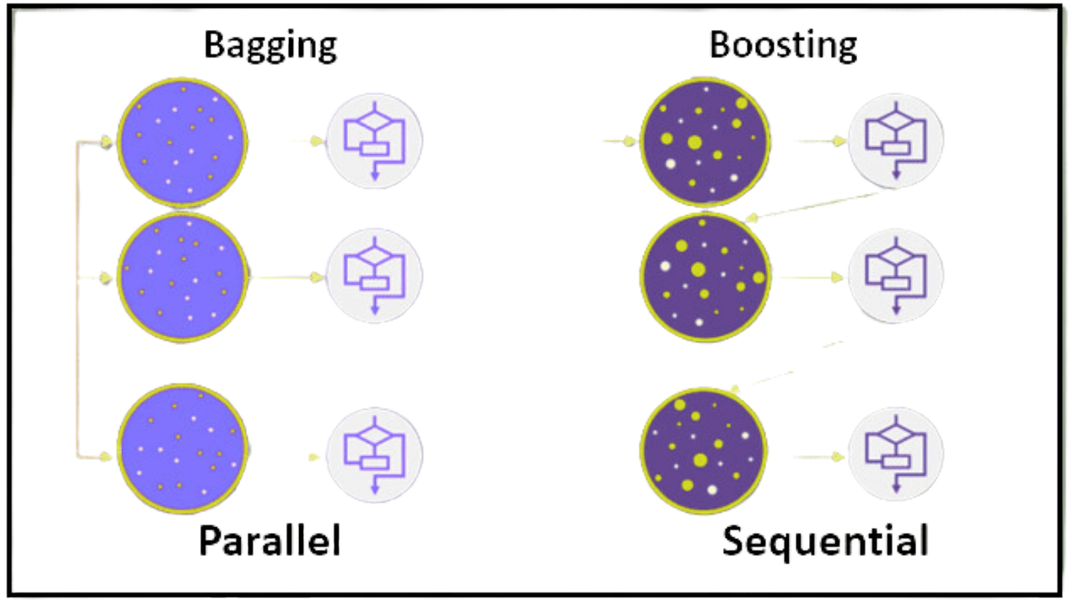
- Applies normalization before activation functions (e.g., ReLU).
- When experiencing unstable training or slow convergence.
- Works well with high learning rates.

- **Early Stopping:** Stops Training When the Model Starts Overfitting.

When to Use:

- When working with limited data to avoid excessive training.
- When validation loss starts increasing while training loss continues decreasing.

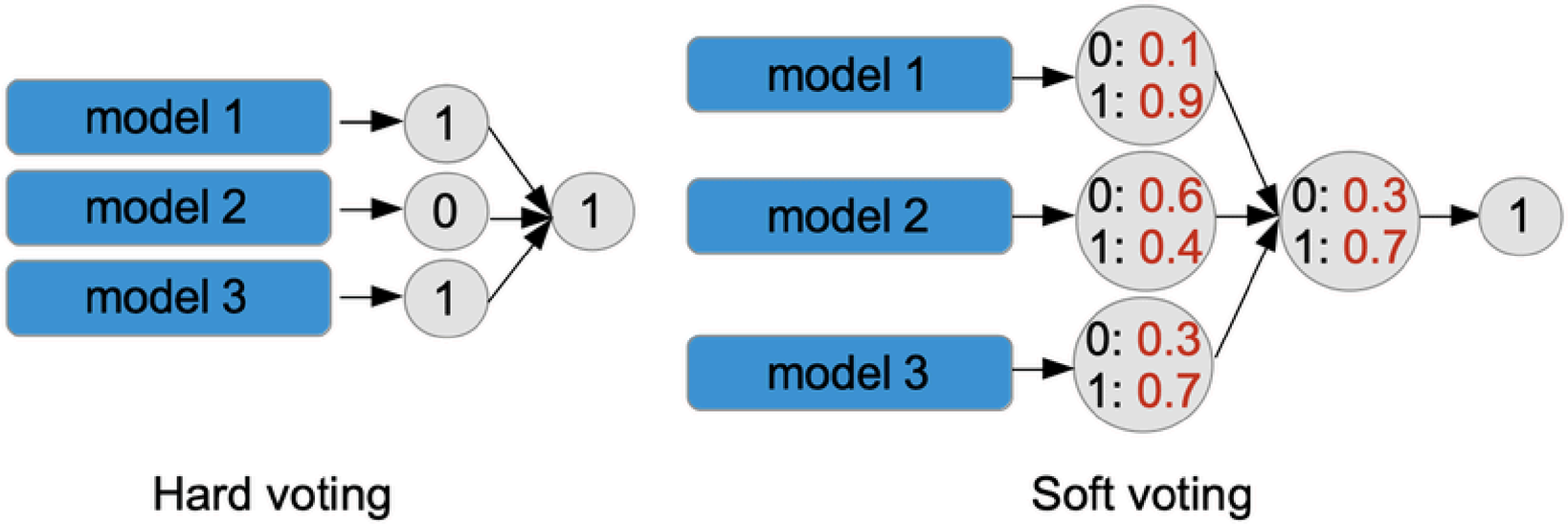
ENSEMBLING METHODS



Ensembling Methods: Combining Multiple Models for Better Accuracy

- **Bagging:** Trains multiple models on different data subsets and averages their predictions.
- **Boosting:** Sequentially trains models where each model focuses on errors made by the previous ones.

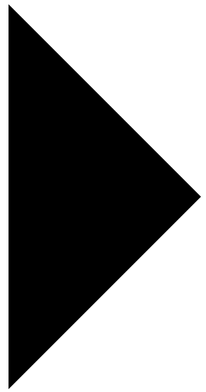
ENSEMBLING METHODS



- **Hard Voting:** Each model predicts a class, and the final decision is based on majority vote.
- **Soft Voting:** Averages the predicted probabilities of all models and picks the highest probability.

ENSEMBLING METHODS

- Use **Bagging** if the model is **overfitting** (reduces variance).
- Use **Boosting** if the model is **underfitting** (reduces bias, improves accuracy).
- Use **Voting** if you want to combine different models for a **stronger final prediction.**



SHORT QUIZ

QUESTION 1:

Given an input volume of 64 × 64 × 3, you apply a convolutional layer with 64 filters, each of size 3 × 3, a stride of 2, and padding of 1. What will be the dimensions of the resulting output volume?

a. 31 x 31 x 64

b. 33 x 33 x 64

c. 32 x 32 x 64

d. none of the above

QUESTION 2:

What is the total number of parameters in this following CNN:

Input 28 x 28 x 1, Conv layer with 16 filters including bias, each filter with size 3 x 3 and a stride of 1 along with a max pool layer with size 2x2 and stride of 2 ?

a. 144

b. 160

c. 10

d. none of the above

QUESTION 3:

what is number of pixels in a grayscale image of size 32 x 32 ?

a. 32

b. 784

c. 1024

d. none of the above

QUESTION 4:

what is the primary purpose of an autoencoder in machine learning?

a. Regression

b. Dimensionality reduction

c. Classification

d. none of the above

QUESTION 5:

Given an input size of 2 × 2, you apply a 2D transposed convolutional layer with:

- Filter size: 3 × 3
- Stride: 2
- Padding: 1

What will be the shape of the output?

a. 5x5

b. 6x6

c. 3x3

d. none of the above

WE WISH YOU THE BEST OF LUCK!