DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBJECT CODE: 23CSX502 **SEM :VI**

SUBJECT NAME: DEEP LEARNING AND NEURAL NETWORKS **YEAR: III**

UNIT II- CONVOLUTION NEURAL NETWORKS

SOLVED

PART-A (2 MARKS)

1. What is CNN and why is it suitable for image data? **K1**

- CNN is a deep learning model mainly used for image processing tasks.
- It extracts features like edges, textures, and shapes automatically.
- It uses **local connections** and **weight sharing** to reduce parameters.
- It preserves spatial information (nearby pixels relationship).
- Used in classification, detection, and segmentation.

2. State the basic architecture of a CNN. **K2**

- CNN consists of **Convolution → ReLU → Pooling → Flatten → Fully Connected → Output**.
- Convolution layer extracts important features.
- Pooling reduces size and computation.
- Flatten converts feature maps into 1D vector.
- Output layer gives final classification probabilities.

3. What is the purpose of the convolution layer in CNN? **K1**

- **Applies filters over the input image**: Kernels slide over the image to process it.
- **Detects edges, textures, and shapes**: Finds important visual patterns.
- **Produces feature maps**: Generates outputs showing detected features.
- **Enables automatic feature extraction**: Learns features without manual design.
- **Helps learn hierarchical patterns**: Builds from simple to complex features.

4. Define local receptive field and weight sharing in CNN **K1**

In CNN, the **local receptive field** means each neuron is connected only to a small region of the input image, helping the network focus on nearby pixel patterns. **Weight sharing** means the same filter (kernel) is applied across the entire image, so the same feature can be detected in different areas. This approach **reduces the number of parameters**, making the model smaller and easier to train. It also **improves computational efficiency** and enables the CNN to **detect the same feature at multiple locations** in the image.

5. What is a feature map **K1**

A feature map is the output produced by a convolution layer in a CNN. It shows the presence and location of specific features such as edges, textures, or shapes in the input image. Each convolution filter (kernel) generates its own feature map by detecting a particular pattern. In the early layers, feature maps usually capture simple patterns like edges and corners. In deeper layers, feature maps represent complex features like object parts and complete shapes.

6. What is pooling and why is it used in CNN? K1

Pooling is a technique used in CNNs to **reduce the size of feature maps** by retaining only important information. It removes unnecessary details and noise while preserving key features. By reducing the spatial dimensions, pooling **lowers computation and memory requirements**. It also **helps prevent overfitting** by reducing the number of parameters. Additionally, pooling provides **translation invariance**, allowing the network to recognize features even when their position changes slightly.

7. **Differentiate between max pooling and average pooling with suitable examples. K2**

Aspect	Max Pooling	Average Pooling
Operation	Selects the maximum value from the pooling region	Computes the mean of all values in the region
Feature Preservation	Preserves strong features like edges	Produces smoother feature representation
Sensitivity	Focuses on most dominant activations	Considers all activations equally
Usage	Commonly used in CNN architectures	Less commonly used
Effect on Output	Highlights prominent features	Smoothens feature maps

8. **What is the role of ReLU in CNN? K2**

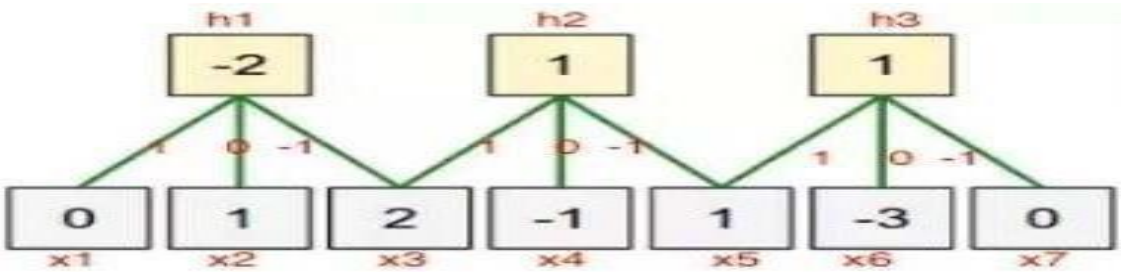
ReLU (Rectified Linear Unit) plays an important role in CNN by introducing non-linearity, which helps the network learn complex patterns from images. It works by converting all negative values to zero while keeping positive values unchanged. This reduces the chances of the vanishing gradient problem, especially in deep networks. ReLU also makes training faster because it is simple to compute and helps the model converge quickly. Due to these advantages, ReLU is widely preferred over sigmoid and tanh activation functions in CNNs.

9. **What is flattening in CNN? K1**

Flattening is the process of converting the 2D or 3D feature maps produced by convolution and pooling layers into a single **1D vector**. It is done before the **fully connected layer** so that the extracted features can be given as input for classification. Flattening helps prepare the data in the correct format for dense layers, which work only with one-dimensional inputs. It acts as a bridge between convolution layers and dense layers. This step is essential because it supports the final prediction and output generation of the CNN.

10. **Why parameter sharing is used in CNN? K2**

Parameter sharing is used in the convolutional layers to reduce the number of parameters in the network. For example, in the first convolutional layer let's say we have an output of 15x15x4 where 15 is the size of the output and 4 the number of filters used in this layer. For each output node in that layer we have the same filter, thus reducing dramatically the storage requirements of the model to the size of the filter.



The same filter (weights) (1, 0, -1) are used for that layer.

11. **List the limitations of traditional Artificial Neural Networks (ANN) in image processing** **K1**

- **Requires large number of parameters**: Fully connected layers need many weights for every pixel, so parameters become very high.
- **Does not preserve spatial structure**: ANN treats pixels independently and cannot understand nearby pixel relationships.
- **Computationally expensive**: More parameters increase training time, memory usage, and processing cost.
- **Prone to overfitting**: ANN may memorize training images instead of learning general features.
- **Not suitable for large images**: As image size increases, network size and computation increase greatly.

12. **Define Sparse Connectivity or Sparse interactions.** **K1**

A Convolution layer defines a window or filter or kernel by which they examine a subset of the data, and subsequently scans the data looking through this window. his is what we call **sparse connectivity** or **sparse interactions** or **sparse weights.** Actually, it limits the activated connections at each layer. In the example below an 5x5 input with a 2x2 filter produces a reduced 4x4 output. The first element of feature map is calculated by the convolution of the input area with the filter i.e.

Input Image

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

Feature Map

6			

Apply 2x2 filter to the input and get the first convolutional layer (a feature map)

1*0+2*1+2*1+1*2=6 First element of the feature map

13. **Differentiate between translation invariance and translation equivariance.** **K2**

Aspect	Translation Equivariance	Translation Invariance
Definition	Feature maps shift in the same way as input shifts	Output remains unchanged despite input shifts
Achieved by	Convolution layers	Pooling layers
Effect on Feature Map	Feature location changes correspondingly	Features recognized regardless of position
Purpose	Preserves spatial information	Provides robustness to object movement
Helps in	Tracking exact feature positions	Detecting objects despite their translation

14. **Why is padding important in CNN?** **K2**

Padding involves adding extra pixels, usually zeros, around the borders of an image before applying convolution. This process prevents the feature maps from shrinking too quickly after each convolution operation, which helps to preserve important edge information near the borders. By maintaining the spatial dimensions, padding enables the construction of deeper networks without losing critical data. It also helps to maintain the output size consistent with the input, allowing better control over the network architecture.

15. **Define Padding and Stride?** **K1**

Padding

Padding is the process of adding extra pixels (usually zeros) around the border of the input image. It helps **preserve the spatial size of the original image** and prevents loss of information at the edges during convolution.

Stride

Stride is the number of pixels by which the convolution filter moves (shifts) over the input image. A larger stride results in a **smaller output feature map**, while a smaller stride produces a **larger output**.

Output Image Dimension

- Input image size = **N × N**
- Filter (kernel) size = **F × F**
- Padding = **P**
- Stride = **S**

Then, the output feature map dimension is calculated as:

$$\text{Output size} = \left(\frac{N + 2P - F}{S} + 1\right)$$

This formula shows how **padding (P)** increases the effective input size, while **stride (S)** controls how many positions the filter can move across the image.

16. **What is AlexNet and its significance?** **K1**

AlexNet is a pioneering deep convolutional neural network introduced in 2012. It gained significant attention by winning the ImageNet competition that year with a large margin, marking a breakthrough in image classification. AlexNet helped popularize deep learning by demonstrating the power of deep architectures in computer vision. It leveraged GPU training to speed up the learning process, making it feasible to train large networks. Overall, AlexNet greatly improved image classification accuracy and set the foundation for modern deep learning models.

17. **State the input size and main layers of the AlexNet architecture.** **K2**

- AlexNet accepts an input image size of **227 × 227 × 3** (height × width × color channels).
- It has **five convolutional layers** that extract increasingly complex features from the input image.
- Each convolutional layer is followed by a **ReLU activation**, which introduces non-linearity and speeds up training.
- **Max pooling layers** are used after some convolutional layers to reduce spatial dimensions and computational complexity.
- The network includes **three fully connected layers** that learn to combine extracted features for classification.

18. **List the key innovations introduced by AlexNet.** **K1**

AlexNet introduced several key innovations that significantly advanced deep learning for image classification. It popularized the use of ReLU activation functions, which helped speed up training by addressing the vanishing gradient problem. The model also employed dropout as a regularization technique to reduce overfitting in fully connected layers. Additionally, AlexNet was one of the first to leverage GPU acceleration for faster training of deep networks. It also used data augmentation methods like image flipping and cropping to artificially expand the training dataset, improving generalization. These innovations together enabled AlexNet to achieve breakthrough performance in the ImageNet competition.

19. How dropout improves AlexNet performance **K2**

- Randomly disables neurons during training.
- Prevents over-dependence on features.
- Reduces overfitting.
- Improves generalization.
- Acts like ensemble learning.

20. What is GoogLeNet and its main idea? **K1**

- **GoogLeNet** is a deep convolutional neural network architecture introduced in 2014.

- It is based on **Inception modules** that apply multiple convolutions (1×1, 3×3, 5×5) in parallel within the same layer.
- Uses **1×1 convolutions** for dimensionality reduction, which helps reduce computational cost.
- The architecture is **very deep with 22 layers** but has relatively **few parameters (~4 million)** compared to other networks.
- Includes **auxiliary classifiers** during training to provide intermediate supervision and act as regularizers.
- Replaces traditional fully connected layers with **global average pooling**, reducing overfitting.
- Won the **ILSVRC 2014** competition with high accuracy and computational efficiency.
- The design is **scalable** and has inspired many subsequent CNN architectures

21. How does the Inception module work in GoogLeNet? K2

The Inception module in GoogLeNet uses parallel convolutional layers with different filter sizes 1×1, 3×3, and 5×5 to capture features at multiple scales simultaneously. It also incorporates a pooling operation within the module to reduce spatial dimensions and enhance feature robustness. The outputs from all these parallel operations are concatenated along the depth dimension to form a combined feature map. This multi-scale approach allows the network to extract both fine and coarse features effectively, leading to improved overall performance and accuracy in image recognition tasks.

22. Why is 1×1 convolution used in CNN? K2

- Reduces number of channels.
- Reduces computation cost.
- Acts as dimensionality reduction.
- Adds non-linearity.
- Improves efficiency.

23. What is global average pooling, and what are its benefits in CNNs? K1

Global average pooling works by averaging each feature map into a single value, effectively summarizing the presence of a feature across the entire spatial dimension. This process removes the need for fully connected layers, which are typically parameter-heavy. By doing so, it reduces the total number of parameters in the network, helping to lower the risk of overfitting. Additionally, global average pooling makes the model more lightweight and efficient, simplifying the architecture while maintaining strong performance.

24. What is ResNet and why was it introduced? K2

- ResNet stands for **Residual Network**.
- It enables the training of **very deep networks**.
- Solves the **degradation problem** where deeper networks perform worse.
- Uses **skip connections** to bypass layers and help gradient flow.

- Achieves **high accuracy** in complex image tasks.

25. Define skip connection in deep neural networks. **K2**

Skip connections in ResNet work by adding the input of a layer directly to its output, forming an equation like Output = F(x) + x, where F(x) is the layer's transformation. This mechanism helps improve gradient flow during backpropagation, preventing the vanishing gradient problem that often occurs in very deep networks. By allowing gradients to pass through these shortcuts, skip connections make it easier to train deep architectures and enable the network to learn more effectively.

26. **What is the degradation problem in deep neural networks and how does ResNet address it?** **K2**

The degradation problem occurs when adding more layers to a deep neural network causes the accuracy to decrease, not improve. This issue arises due to optimization difficulties during training, rather than overfitting. ResNet solves this problem by introducing residual learning through skip connections, which allow the network to learn identity mappings easily. This helps improve gradient flow and makes it easier to train very deep networks, thereby enhancing overall performance.

27. **How does ResNet solve the vanishing gradient problem?** **K1**

Skip connections in ResNet enable the direct flow of gradients from deeper layers to earlier ones during training, preventing the vanishing gradient problem where gradients become too small to effectively update weights. This allows early layers to learn efficiently, improving the overall training process and leading to faster, more stable convergence. Consequently, skip connections support the successful training of very deep networks without performance degradation.

28. **What is R-CNN and full form?** **K1**

- R-CNN means Region-based CNN.
- Used for object detection.
- Generates region proposals.
- Classifies each region.
- More accurate than sliding window.

29. How does selective search work in R-CNN? **K2**

Selective Search is a technique used in object detection to generate **region proposals** candidate areas in an image that might contain objects. It works by **grouping similar pixels** together based on features like **colour, texture, and size**, creating regions that are likely to represent meaningful parts of the image. This process produces approximately **2000 region proposals**, which significantly narrows down the areas the detector needs to analyse. By focusing only on these promising regions rather than the entire image, Selective Search **reduces computational complexity** and improves the efficiency of object detection algorithms like R-CNN.

30. **What is Non-Maximum Suppression (NMS)?** **K1**

Non-Maximum Suppression (NMS) is a post-processing technique used in object detection to **remove duplicate bounding boxes** that predict the same object. It works by **keeping only the bounding box with the highest confidence score** among overlapping boxes and **eliminating others that overlap significantly**. This process helps to reduce multiple detections of the same object, leading to a **cleaner and more accurate detection output**. By filtering out redundant boxes, NMS improves the overall **detection accuracy** and clarity of the final results.

PART-A (16 MARKS)

1. **Analyze why Convolutional Neural Networks (CNNs) outperform traditional Artificial Neural Networks (ANNs) for image data, with reference to parameter count, spatial information loss, overfitting, and how CNN architectures overcome these limitations (16)** **K4**

Why ANN is not suitable for Images

Images are **structured data** arranged as **Height × Width × Channels** (2D/3D). A traditional **ANN (Fully Connected Network)** treats each input value as **independent**, but in images, **spatial arrangement of pixels is important**.

Example: In a face image, pixels near the **eyes, nose, mouth** must remain close to learn correct patterns. So, **CNN is designed for images** using **convolution + pooling + feature maps**.

Limitations of Traditional ANN for Image Data: Flattening causes Spatial Information Loss:

In ANN, an image must be converted into a **1D vector** using flattening.

Example: A 4×4 image matrix becomes:

2D Image

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Flattened Vector:

$$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]$$

Neighbour pixels in 2D become **far apart in 1D**. ANN cannot learn **local patterns** Important image features like: edges, corners, textures, shapes are not captured properly So, ANN fails in understanding **object structure**.

Traditional ANNs and fully connected networks treat every input neuron as independent and connect it to every neuron in the next layer. When applied to images, this creates several problems:

Huge Number of Parameters: Even a small colour image (e.g., 200 × 200 × 3) has 120,000 input values. Connecting each of these to neurons results in millions of parameters, making the model:

- Very slow
- Memory intensive
- Hard to train

Loss of Spatial Information: Fully connected networks do not preserve spatial structure. A pixel next to another pixel is treated the same as a pixel far away. This makes it difficult to learn local patterns such as edges, corners, and textures.

Overfitting Problem: Because of too many parameters, the network tends to memorize training data instead of learning meaningful features, especially when training data is limited.

Manual Feature Extraction: Earlier systems required hand-crafted features (edges, shapes, textures).

- This process is time-consuming
- Task-specific
- Poor at generalization

Poor Scalability: Fully connected networks work for small images like MNIST (28 × 28), but they fail for real-world large, high-resolution images.

How CNN solves ANN problem: CNN solves these issues using: local connectivity, weight sharing, pooling + hierarchical learning

Local Receptive Field (Local Connectivity)

CNN neurons connect only to a small region of image (not entire image), called receptive field.

Examples: 3×3, 5×5 filters Helps learn local features like: edges, lines, corners

Weight Sharing reduces parameters

In CNN, the same filter (kernel) is applied across the entire image.

Example: For one 3×3 filter: Weights = 3 × 3 = 9 weights only (instead of millions!)

Benefits:

- Huge reduction in parameters
- Less memory and faster training
- Better generalization
- Reduces overfitting

Pooling reduces computation and overfitting:

Pooling reduces the size of feature maps using **Max pooling / Avg pooling**.

Example: 2×2 max pooling selects the maximum value.

Advantages:

- Reduces dimensions
- Reduces computation
- Adds translation invariance (small shift won't affect output much)
- Helps prevent overfitting

CNN Feature Learning Advantage (Hierarchical Learning)

- CNN learns features automatically in layers:
- **Early layers:** edges, lines
- **Middle layers:** textures, patterns
- **Deep layers:** object parts (eye, wheel)
- **Final layer:** complete objects (face, car)

CNN performs **automatic feature extraction**, unlike ANN.

2. **Examine the fundamental architecture of a Convolutional Neural Network CNN and explain how its five core layers Convolution Activation Pooling Fully Connected and Output work sequentially to transform an input image into final class probabilities (16)** **K4**

Five Core CNN Layers in Architecture

A basic architecture of convolutional neural network works in five clear steps. Each layer in CNN has a simple job. Together, they turn raw images into class scores. These layers of CNN appear in almost every CNN you study, no matter how small or deep.

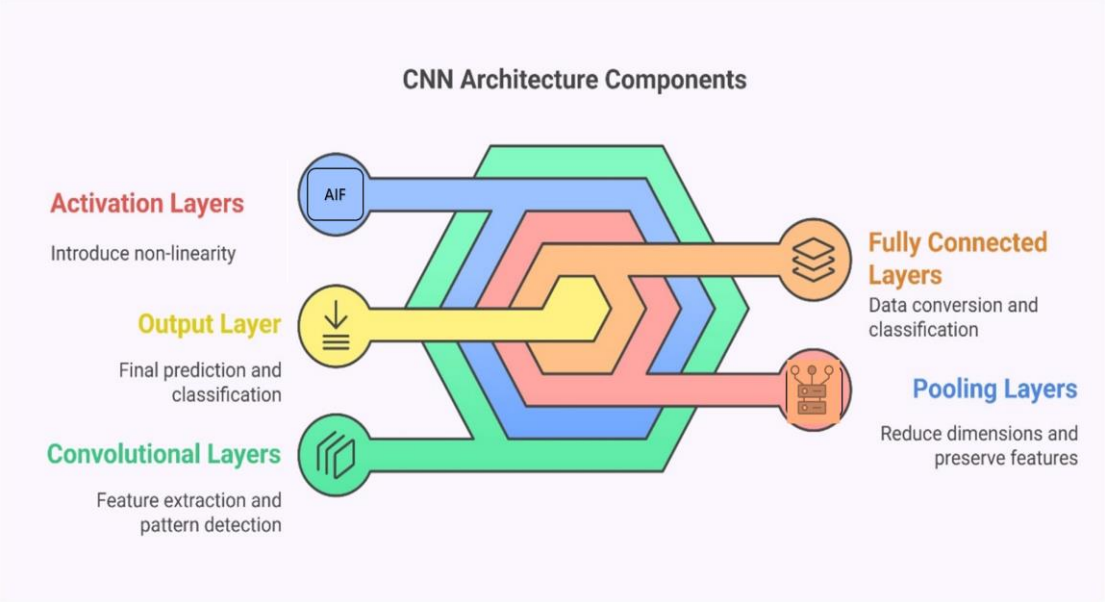


Figure 2.2: CNN Architecture Components

A) Convolution Layer

The convolution layer is the entry point of the simple CNN architecture. Instead of looking at the whole image at once, the model focuses on small blocks of pixels. This helps it notice local patterns before building a full understanding of the image. It uses multiple filters, and each one learns a different visual pattern such as an edge, corner, or texture.

How It Works:

- A filter slides across the image one small step at a time. At each step, it multiplies its values with the pixel values under it.
- The results are added and stored in a new grid called a feature map. This feature map shows where the learned pattern appears in the image.
- During training, the CNN adjusts the values inside each filter. Over time, the filters become good at detecting useful patterns. Early filters catch simple shapes. Deeper filters catch more complex textures and visual details.

Key actions in this layer

- Scanning small regions to understand local structure
- Extracting edges, corners, textures, and simple shapes
- Producing feature maps that highlight important visual details

B) Activation Layer

After the convolution layer picks up basic patterns, the model needs a way to understand more complex shapes and relationships. The activation layer makes this possible by adding nonlinearity. Without this step, an architecture of CNN in deep learning would behave like a simple linear model and would miss many details found in real images.

ReLU is the most widely used activation in convolutional neural network architecture. It keeps positive values and removes negative ones. This helps the model focus on strong signals and train faster. Sigmoid and Tanh are also used, mainly when the model needs smoother transitions between values.

How It Works

- The activation function is applied to every value in the feature map.
- If you use ReLU, negative values become zero while positive values stay as they are.
- This simple step helps the model learn curves, textures, and layered patterns.

Common activation functions

- ReLU
- Sigmoid
- Tanh

C). Pooling Layer

The pooling layer reduces the size of the feature maps created by the previous convolutional neural network layers. It selects the most important values from small regions and leaves out the rest. This keeps essential details while removing noise and extra information. The result is a lighter and faster model that still retains the core features needed for learning.

Pooling also helps the CNN stay stable when objects shift slightly within an image. A feature that appears a little to the left or right will still be captured after pooling. This makes the model more reliable and improves its ability to generalize.

How It Works

- The layer divides each feature map into small blocks.
- Depending on the pooling type, it either picks the strongest value or takes the average of the block.
- This reduces the spatial size but keeps the important signal intact.

Two common types

- Max pooling: Strongest value in the region
- Average pooling: Average value in the region

D) Fully Connected Layer

After the earlier CNN layers extract and refine features, the CNN needs a way to combine everything into a final understanding. This is where the fully connected layer comes in. The model first flattens all feature maps into one long vector. This vector represents every detail learned so far.

The flattened vector is then passed into dense units. Each unit connects to every value in the vector. These connections help the model understand how different features relate to each other. Early layers of CNN focus on edges and shapes. This layer focuses on the big picture, such as identifying whether the image shows a digit, an object, or a face.

How It Works

- Each dense unit receives all input values and multiplies them with learned weights.
- The results are added and then passed through an activation function.
- This helps the model capture high-level patterns that simpler layers cannot detect.

What this layer handles

- Combining all learned features
- Understanding high-level patterns
- Passing values to the final stage

E) Output Layer

The output layer is the final step in the architecture of Convolutional Neural Network. This is where the model makes its decision. After the fully connected layer processes all features, the output layer converts those values into clear probabilities. These probabilities tell you which class the model believes the image belongs to.

For multi-class problems, the model uses Softmax. It assigns a probability to every class and ensures the values add up to one. For binary problems, the model uses Sigmoid. It produces a single probability that represents a yes or no outcome.

How It Works

- The final values from the dense units are passed into the chosen output function.
- The function transforms raw numbers into interpretable scores.
- The class with the highest score becomes the predicted label.

Role of this layer

- Producing class scores
- Returning the final prediction

These five layers in CNN complete the full flow of a simple CNN architecture. The model begins by detecting simple shapes and ends by delivering a clear prediction.

3. a) Analyze the variations and extensions of the basic CNN architecture by comparing their strengths and limitations and examining their suitability for specific real-world applications. (8) **K4**

Variations and Extensions of Basic CNN Architecture: A basic architecture gives a strong starting point, but many tasks need deeper or more efficient designs. Over the years, several variations have been introduced to improve feature learning, speed, and accuracy.

A. Deep CNNs: Deep CNNs add more convolution and pooling layers. More convolutional neural network layers let the model learn detailed and complex patterns. Early layers capture edges, while deeper layers understand shapes and object parts. This design is common in large image classification tasks.

B. Residual Networks: Residual networks help when models grow very deep. They include skip connections that pass information forward without losing it. This makes training stable and prevents the model from forgetting earlier patterns.

3. Dilated Convolutions: Dilated convolutions widen the filter's field of view. They help the model capture broader context without increasing parameters. This works well for tasks like segmentation and depth estimation.

4. Depth wise Separable Convolutions: This variation breaks the convolution process into two smaller steps. It reduces computation and keeps the model fast. Lightweight CNNs and mobile models often use this approach.

Comparison Table

Variation	Main Benefit
Deep CNNs	Better feature depth
Residual networks	Stable training for deep models
Dilated convolutions	Wider context understanding
Depthwise separable convolutions	Faster and lighter models

CNN Variants with Strengths, Limitations, and Applications:

Variation	Strength	Limitation	Best-Suited Applications
Deep CNNs	Learns complex features	High computation and overfitting	Large-scale image classification
Residual Networks	Stable deep training	Increased memory usage	Medical imaging, object detection
Dilated Convolutions	Captures wide context	May miss fine details	Image segmentation
Depthwise Separable Conv	Fast and lightweight	Slight accuracy trade-off	Mobile vision, IoT

Real-World Applications of CNN Architecture

Application Area	How CNNs Are Used
Image classification	Classifies images into labels like animals, objects, or scenes by learning visual patterns.
Object detection	Locates multiple objects in an image and draws bounding boxes to identify each one.
Medical imaging	Helps detect tumors, fractures, and diseases by analyzing X-rays, MRIs, and CT scans.
Face recognition	Identifies and verifies faces in security systems, phones, and attendance tools.
Autonomous systems	Reads roads, signs, pedestrians, and obstacles for safe navigation.
Text extraction	Converts handwritten or printed text into digital text in OCR systems.
Quality inspection	Finds product defects on manufacturing lines by spotting texture or shape irregularities.

3. b) Analyze the role of feature extraction and pooling in Convolutional Neural Networks by examining their advantages and disadvantages in modern AI applications. (8) K5

Feature Extraction in CNNs:

- CNNs learn features hierarchically.
- Early layers detect simple patterns like edges.
- Middle layers capture complex shapes.
- Deeper layers recognize entire objects (e.g., faces, cars).

Pooling Layer:

- Reduces the size of feature maps, which decreases computation and helps prevent overfitting.
- Max pooling selects the strongest activations, while average pooling smooths the features.
- Helps retain important features while lowering complexity.

Advantages of CNNs:

- Automatically learn features without manual feature engineering.
- Use shared weights to reduce the number of parameters and improve efficiency.
- Recognize patterns irrespective of their position in the input.
- Capture both low-level and high-level features effectively.
- Applicable to various data types: images, audio, video, text.
- Automatic hierarchical feature extraction reduces the need for manual feature design.
- Pooling reduces spatial dimensions, lowering computation and improving generalization.
- Feature extraction enables CNNs to learn invariant representations of patterns

Disadvantages of CNNs (1 mark):

- Require high computational power and large labeled datasets.
- Difficult to interpret decision-making processes.
- Can overfit on small datasets.
- Require fixed input sizes, limiting flexibility.
- Pooling may cause loss of fine-grained spatial information.
- Deep feature extraction requires large labeled datasets.
- Aggressive pooling can reduce localization accuracy in tasks like object detection

4. K5

R=

7	3	5	6	8
2	5	6	7	9
10	11	18	19	10
3	6	7	4	2
3	8	7	4	5

G=

6	4	5	7	6
3	6	7	8	7
9	10	16	17	9
2	5	6	3	1
4	7	6	3	4

B=

5	2	4	5	7
1	4	5	6	8
8	9	14	15	8
1	4	5	2	1
2	6	5	2	3

Kernel=

0	-1	0
-1	5	-1
0	-1	0

Given a 5×5 input image and a 3×3 kernel, evaluate the process of convolution, pooling, fully connected layer output calculation, and activation in a CNN by performing the following:

a) Calculate the size of the output feature map and the convolution output after applying a 3×3 kernel on the 5×5 input with valid padding and stride 1. Explain the significance of these parameters.

b) Analyse the max pooling operation and determine the size of the pooled feature map when a 2×2 max pooling with stride 1 is applied on the output feature map. Discuss how pooling impacts the feature map.

c) Construct the formula for the output of a fully connected (dense) layer given weights $w = [1, 1, 1, 1]$and bias $b = 1$. Express the output $y$in terms of inputs x_1, x_2, x_3, x_4.

d) Evaluate the sigmoid activation function by writing its formula and explain its role and importance in neural networks. (16)

STEP 1: Output feature map size

We calculate **how many valid positions** the 3 × 3 kernel can slide over a 5 × 5 image.

Formula

$$\text{Output size} = \frac{W - F + 2P}{S} + 1$$

Where

- $W = 5$(input width/height)
- $F = 3$(kernel size)
- $P = 0$(padding)
- $S = 1$(stride)

Substitute values

$$\frac{5 - 3 + 2 * 0}{1} + 1 = 2 + 1 = 3$$

Output feature map size = 5 × 5

This means the kernel fits in **3 valid positions horizontally and vertically**.

Step 2: Formula for convolution at position (i, j):

$$\text{output}[i,j] = \sum_{m=0}^{2} \sum_{n=0}^{2} \text{input}[i+m, j+n] \times \text{kernel}[m,n]$$

To calculate Output [0,0]:

7	3	5	6	8	6	4	5	7	6	5	2	4	5	7
2	5	6	7	9	3	6	7	8	7	1	4	5	6	8
10	11	18	19	10	9	10	16	17	9	8	9	14	15	8
3	6	7	4	2	2	5	6	3	1	1	4	5	2	1
3	8	7	4	5	4	7	6	3	4	2	6	5	2	3

R = (7×0)+(3×−1)+(5×0)+(2×−1)+(5×5)+(6×−1)+(10×0)+(11×−1)+(18×0)=3

G = (6×0)+(4×−1)+(5×0)+(3×−1)+(6×5)+(7×−1)+(9×0)+(10×−1)+(16×0)=6

B = (5×0)+(2×−1)+(4×0)+(1×−1)+(4×5)+(5×−1)+(8×0)+(9×−1)+(14×0)=3

output(0,0)=R(0,0)+G(0,0)+B(0,0)+bias=3+6+3+1=13

To calculate Output [0,1]:

R= (3×0)+(5×−1)+(6×0)+(5×−1)+(6×5)+(7×−1)+(11×0)+(18×−1)+(19×0)=−5

G= (4×0)+(5×−1)+(7×0)+(6×−1)+(7×5)+(8×−1)+(10×0)+(16×−1)+(17×0)= 0

B= (2×0)+(4×−1)+(5×0)+(4×−1)+(5×5)+(6×−1)+(9×0)+(14×−1)+(15×0)= −3

output(0,1)=R(0,1)+G(0,1)+B(0,1)+bias= −5+0+(−3)+1=-7

To calculate Output [0,2]:

R= (5×0)+(6×−1)+(8×0)+(6×−1)+(7×5)+(9×−1)+(18×0)+(19×−1)+(10×0) =−5

G= (5×0)+(7×−1)+(6×0)+(7×−1)+(8×5)+(7×−1)+(16×0)+(17×−1)+(9×0) = 2

B= (4×0)+(5×−1)+(7×0)+(5×−1)+(6×5)+(8×−1)+(14×0)+(15×−1)+(8×0) = −3

output(0,2)=R(0,2)+G(0,2)+B(0,2)+bias= −5+2+(−3)+1=-5

To calculate Output [1,0]:

R sum = 15; G sum = 15; B sum = 15

$$\text{output}(1,0) = 15 + 15 + 15 + 1 = 46$$

To calculate Output [1,1]:

R sum = 41; G sum = 41; B sum = 41

$$\text{output}(1,1) = 41 + 41 + 41 + 1 = 124$$

To calculate Output [1,2]:

R sum = 50; G sum = 50; B sum = 50

$$\text{output}(1,2) = 50 + 50 + 50 + 1 = 151$$

To calculate Output [2,0]:

R sum = 0; G sum = 0; B sum = 0

$$\text{output}(2,0) = 0 + 0 + 0 + 1 = 1$$

To calculate Output [2,1]:

R sum = 0; G sum = 0; B sum = 0

$$\text{output}(2,1) = 0 + 0 + 0 + 1 = 1$$

To calculate Output [2,2]:

R sum = −12; G sum = −12; B sum = −13

$$\text{output}(2,2) = (-12) + (-12) + (-13) + 1 = -36$$

Final Output Matrix

$$\begin{bmatrix} 13 & -7 & -5 \\ 46 & 124 & 151 \\ 1 & 1 & -36 \end{bmatrix}$$

Step 3: Pooling layer

$$\text{Output size} = \frac{W - F + 2P}{S} + 1$$

- Pooling size = 2 × 2
- Stride = 1
- No padding

Output size of pooling

$$(3 - 2 + 1) \times (3 - 2 + 1) = 2 \times 2$$

Max pooling calculations

Pooling window (0,0)

$$\begin{bmatrix} 13 & -7 \\ 46 & 124 \end{bmatrix} = 124$$

Pooling window (0,1)

$$\begin{bmatrix} -7 & -5 \\ 124 & 151 \end{bmatrix} = 151$$

Pooling window (1,0)

$$\begin{bmatrix} 46 & 124 \\ 1 & 1 \end{bmatrix} = 124$$

Pooling window (1,1)

$$\begin{bmatrix} 124 & 151 \\ 1 & -36 \end{bmatrix} = 151$$

Final Max-Pooled Output

$$\begin{bmatrix} 124 & 151 \\ 124 & 151 \end{bmatrix}$$

Flattening: Flattened size $= h \times w \times c$

$$\begin{bmatrix} 124 & 151 \\ 124 & 151 \end{bmatrix} \Rightarrow [124, 151, 124, 151]$$

Step 4: Fully Connected Layer:

Number of neurons = **1**

Weight vector: $w_i = [1\ 1\ 1\ 1]$

$x_i = [124, 151, 124, 151]$; Bias = 1; N=4

Fully Connected Layer Formula

$$y = \sum_{i=1}^{n} w_i\ x_i + b$$

y=(1×124)+(1×151)+(1×124)+(1×151)+1=550+1=551

Sigmoid Activation Function:

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

$$\sigma(551) = \frac{1}{1 + 2.7^{-551}}$$

Since e^{-551} is extremely close to 0,

$$\sigma(551) \approx 1$$

5. a) Differentiate between AlexNet, GoogLeNet, ResNet, and Region-Based CNNs. (8) K4

Feature	AlexNet	GoogLeNet (Inception)	ResNet	Region-Based CNNs
Year Introduced	2012	2014	2015	2014 onwards
Primary Task	Image classification	Image classification	Image classification	Object detection
Key Innovation	First deep CNN to win ImageNet	Inception modules	Residual (skip) connections	Region proposal mechanism
Architecture Idea	Deep CNN with ReLU and pooling	Parallel convolutions	Skip connections between layers	CNN + region proposals
Convolution Filters	Large filters (11×11, 5×5)	Parallel filters (1×1, 3×3, 5×5)	Mostly small filters (3×3)	Depends on backbone CNN
1×1 Convolutions	Not used	Used for dimensionality reduction	Used in bottleneck blocks	Used in backbone networks
Depth	8 layers	22 layers	50, 101, 152+ layers	Varies by model
Number of Parameters	Very high	Much fewer than AlexNet	Efficient despite depth	High (especially early R-CNN)
Main Advantage	Proved deep CNNs are effective	Efficient and computationally optimized	Solves vanishing gradient problem	Detects and classifies objects
Limitation	Heavy computation, overfitting	Complex architecture	High training cost	Slow in early versions
Special Capability	Classification only	Classification only	Classification only	Localization + classification
Accuracy	Good (for its time)	Higher than AlexNet	Very high	High for detection tasks
Typical Use Case	Basic image classification	Efficient large-scale image classification	Very deep image recognition	Detecting multiple objects

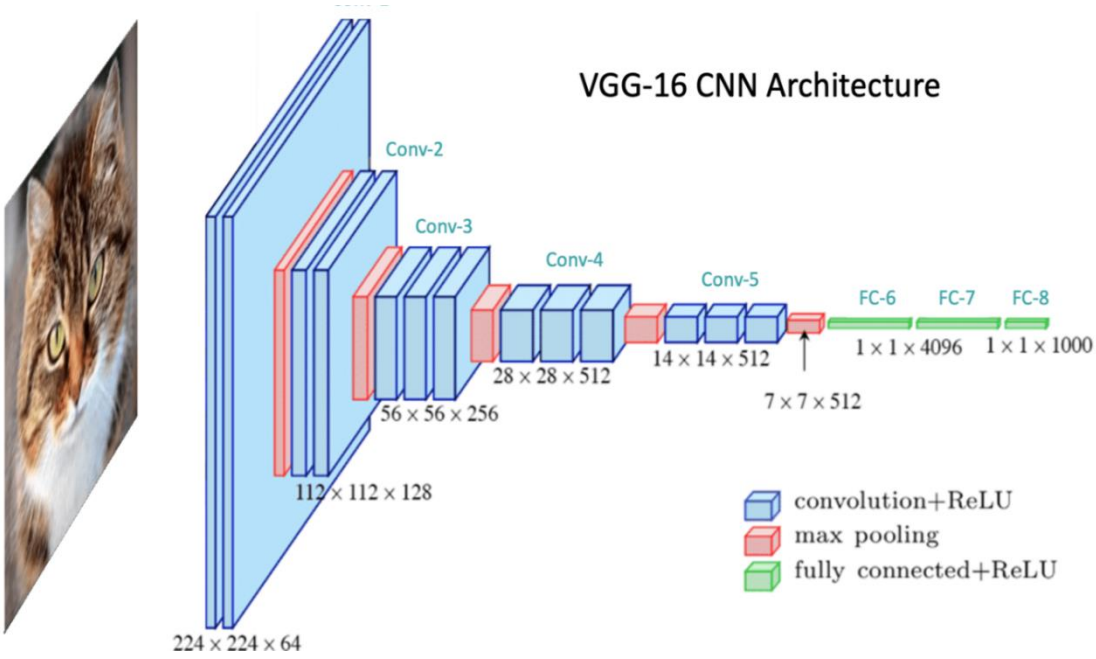
5. b) Analyse the sequence of operations in one CNN stage and how these operations contribute to processing input images for classification in a Deep CNN like VGG-16. (8) **K4**

Deep Convolutional Neural Networks (Deep CNNs)

A Deep Convolutional Neural Network (Deep CNN) consists of many layers (stages) connected in sequence to learn hierarchical features from images. Each stage has a similar structure, though the number of stages may vary across architectures.

VGG-16 Architecture: VGG-16 is a well-known Deep CNN architecture.

- Uses repeated Convolution + ReLU + Pooling blocks
- Input size reduces from 224×224 to 7×7
- Number of feature maps increases from 64 to 512
- Ends with Fully Connected layers for classification



Structure of One CNN Stage: Each stage of a Deep CNN consists of:

1. Input Maps Volume
    - Receives feature maps from the previous stage
    - For RGB image: 3 maps (R, G, B)
2. Feature Maps Volume
    - Generated after convolution, bias addition, and activation
    - Each map detects specific patterns such as edges or textures
3. Pooled Maps Volume (Optional)
    - Obtained using pooling
    - Reduces spatial size and computation

Volume Convolution

- Convolution in Deep CNNs is 3D (volume convolution)
- Kernel depth equals input volume depth
- Convolution is applied spatially on each map
- Outputs are summed to produce one feature map value

$$z = \sum(w \cdot v) + b$$

Activation and Pooling

- Output $z$is passed through an activation function:

$$a = h(z)$$

- Activation (ReLU) introduces non-linearity
- Pooling reduces feature map size while retaining important information

Flow of One CNN Stage: Input Maps → Convolution → Bias → Activation → Feature Maps → Pooling → Next Stage

Fully Connected Layers: After all CNN stages:

- Feature maps are flattened
- Passed to fully connected layers
- Final layer uses Softmax (multi-class) or Sigmoid (binary) for classification

6. Analyse the architecture and layer-wise functioning of AlexNet. Illustrate your answer with a neat diagram. (16) **K4**

Introduction and Need for AlexNet

AlexNet is a deep convolutional neural network introduced in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. It won the ILSVRC-2012 ImageNet competition, proving that deep CNNs trained using GPUs can outperform traditional machine learning methods. Earlier image recognition relied on handcrafted features, which were difficult to design and could not scale to large datasets like ImageNet (1M+ images, 1000 classes). AlexNet addressed this by enabling automatic feature learning using deep CNNs, ReLU activation, dropout, and GPU acceleration, marking the start of modern deep learning in computer vision.

AlexNet Architecture: AlexNet consists of 8 learnable layers:

- 5 Convolutional layers
- 3 Fully Connected layers

Key techniques: ReLU, overlapping max pooling, dropout, and softmax output.

Layer-wise Working

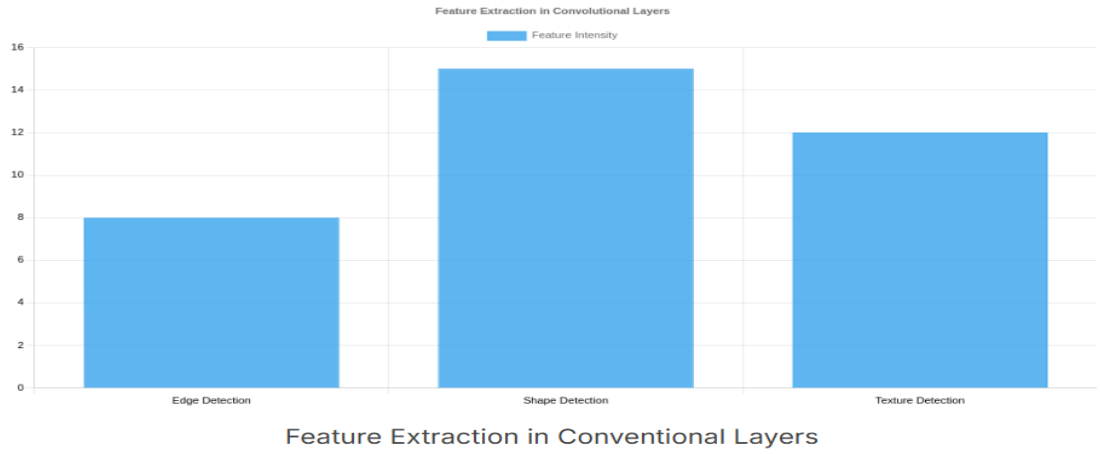
1. Input Layer

- Input size: 227 × 227 × 3 (RGB)
- Images are normalized before processing.

2. Convolutional Layers

- Use filters to extract features.
- Early layers detect edges and colors.
- Deeper layers detect shapes, object parts, and objects.

- First layer uses 11 × 11 filters.



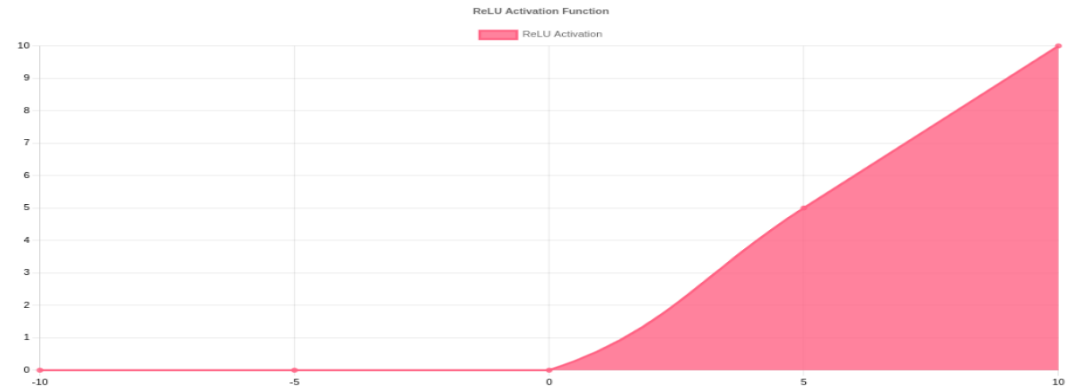
Feature Extraction in Conventional Layers

3. ReLU Activation

- Function:

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

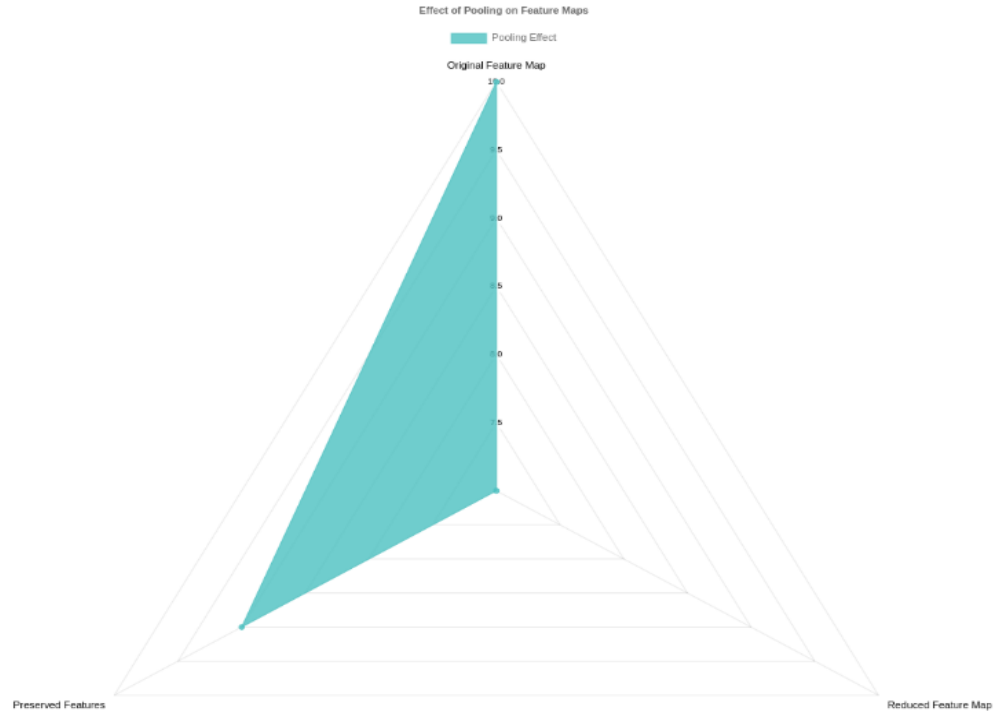
- Removes negative values.
- Introduces non-linearity.
- Faster training and reduced vanishing gradient problem.



ReLU Activation Function

4. Pooling Layers

- Uses overlapping max pooling.
- Reduces spatial size.
- Preserves important features.
- Reduces overfitting and computation.



pooled results vs. the original image

5. Fully Connected Layers

- Combine all extracted features.
- Perform final decision-making for classification.

6. Dropout Layer

- Applied after the fully connected layers during training
- Randomly disables some neurons
- Typical dropout rate: **0.5**
- Prevents overfitting
- Improves generalization
- Makes the network robust

6. Output Layer

- Uses Softmax activation.
- 1000 neurons (ImageNet classes).
- Outputs class probabilities.

7. Analyze the design innovations in GoogLeNet's architecture, such as the Inception module and auxiliary classifiers, by examining how they contribute to improved accuracy and computational efficiency. Illustrate your answer with a neat diagram (8) K5

Introduction: GoogLeNet (Inception V1) is a deep convolutional neural network developed by Google for efficient large-scale image classification. It introduced the Inception module, which performs parallel convolutions (1×1, 3×3, 5×5) and max pooling, followed by concatenation. GoogLeNet focuses on high accuracy with low computational cost and won the ILSVRC-2014 competition.

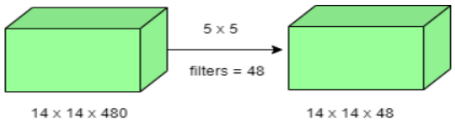
Key Features of GoogLeNet

A. 1×1 Convolutions

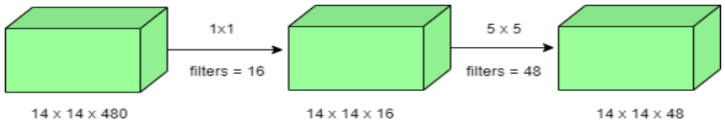
- Used for dimensionality reduction
- Reduces number of parameters and computation
- Adds non-linearity and improves efficiency

Example Comparison:

- **Without 1×1 Convolution:**(14×14×48)×(5×5×480)=112.9M operations



- **With 1×1 Convolution:**(14×14×16)×(1×1×480)+(14×14×48)×(5×5×16)=5.3M operations



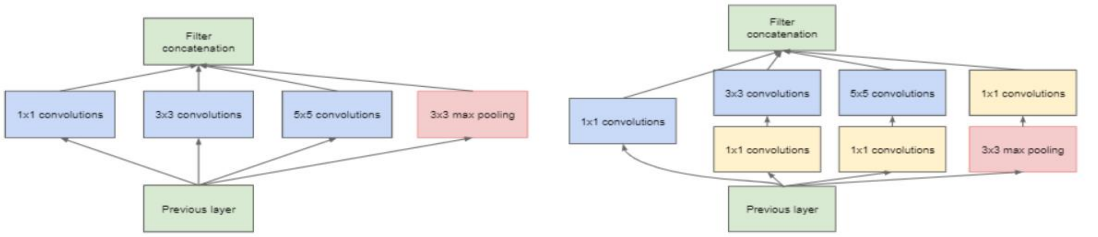
This results in a massive reduction in computation without compromising performance.

B. Inception Module

- Core building block of GoogLeNet
- Processes input in parallel using:
  - 1×1 convolution
  - 3×3 convolution
  - 5×5 convolution
  - 3×3 max pooling
- Outputs are concatenated depth-wise
- Captures multi-scale features efficiently

Optimized Inception Module

- Uses 1×1 convolutions before 3×3 and 5×5 filters
- Reduces computation and memory
- Maintains representational power



(a) Inception module, naïve version (b) Inception module with dimension reductions

C. Global Average Pooling

- Replaces fully connected layers
- Converts feature maps into single values
- Reduces parameters and overfitting
- No additional trainable weights

D. Auxiliary Classifiers

- Added at intermediate layers during training
- Help reduce vanishing gradient problem
- Act as regularizers
- Removed during testing

GoogLeNet Architecture

- 22 layers deep (excluding pooling)
- Input: 224 × 224 × 3 RGB image
- Initial convolution and pooling layers extract low-level features
- Followed by multiple Inception modules
- Ends with global average pooling and softmax classifier

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Working of GoogLeNet

Input → Conv + Pool → Inception Modules → Auxiliary Classifiers → Global Average Pooling → Softmax Output

8. **Analyse the architecture and working of Residual Networks (ResNet) with emphasis on residual learning, the residual block, and skip connections. Using ResNet-34 as an example, describe how these components enable effective training of deep networks. Support your answer with suitable diagrams (16)** K4

Residual Networks (ResNet) were introduced to overcome the difficulties in training very deep neural networks. As network depth increases, problems such as vanishing gradients and degradation occur. ResNet solves these issues using skip (shortcut) connections, allowing the network to learn residual mappings instead of direct mappings, making deep networks easier to train. Traditional deep networks attempt to learn a direct mapping $H(x)$.

ResNet instead learns a residual function $F(x)$and adds the input directly to the output.

$$H(x) = F(x) + x$$

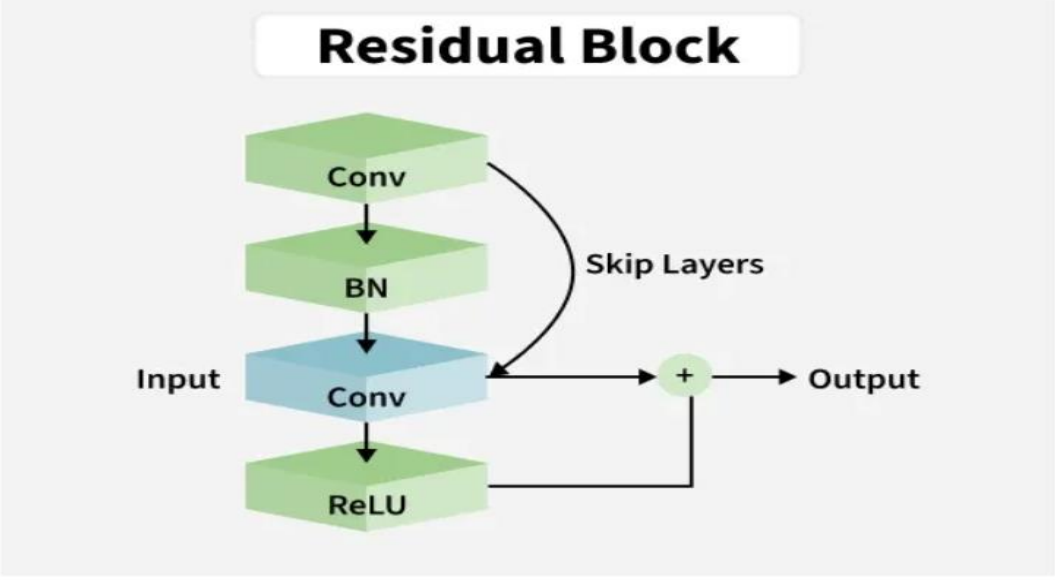
Where:

- x: Input to the block
- $F(x)$: Residual function
- $H(x)$: Output of the block

This formulation simplifies learning and improves gradient flow.

Residual Block

A residual block is the basic building unit of ResNet. It contains convolution layers and a skip connection that bypasses these layers and adds the input directly to the output.



Residual Block

Skip (Shortcut) Connections

Skip connections allow information to flow directly across layers by bypassing one or more layers. This improves parameter updates and stabilizes training.

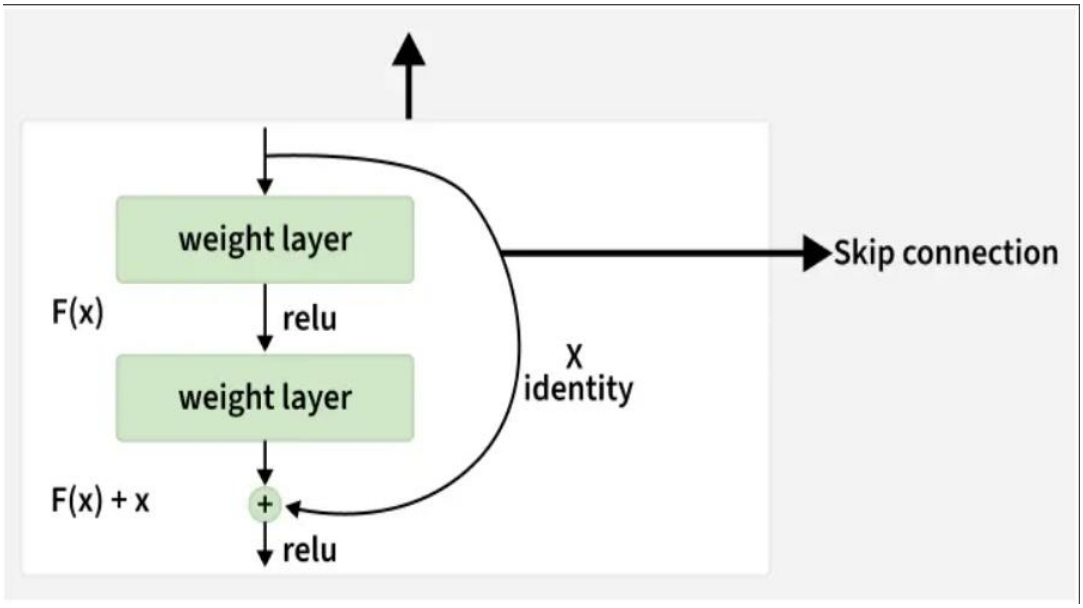
Handling Dimension Mismatch

When input and output dimensions differ:

- Zero Padding: Pads extra zeros to match dimensions
- Linear Projection: Uses 1×1 convolution to match dimensions

How ResNet Works

Instead of learning full transformations, ResNet learns residual functions and adds them to the input using shortcut connections. This allows deeper networks to be trained effectively.



ResNet Architecture

ResNet architectures are created by stacking multiple residual blocks. Popular variants include ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152.

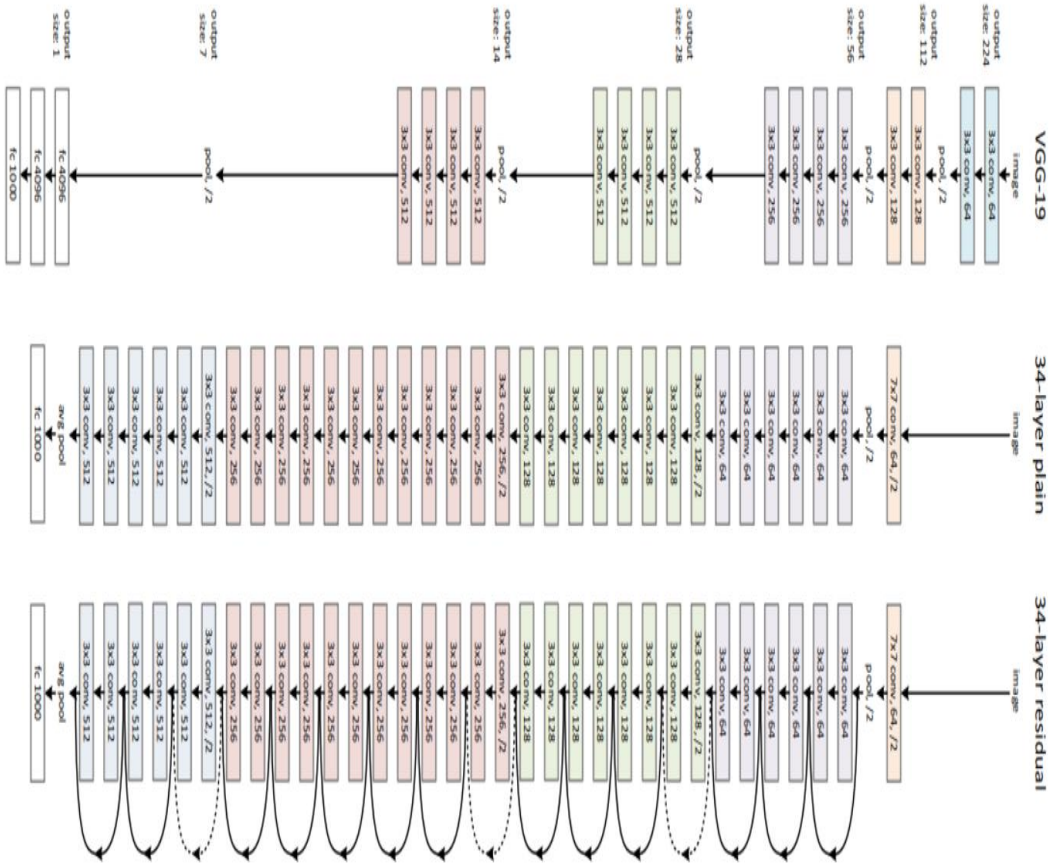
ResNet-34 Architecture

ResNet-34 is a deep residual network consisting of 34 layers and 16 residual blocks.

Architecture Stages:

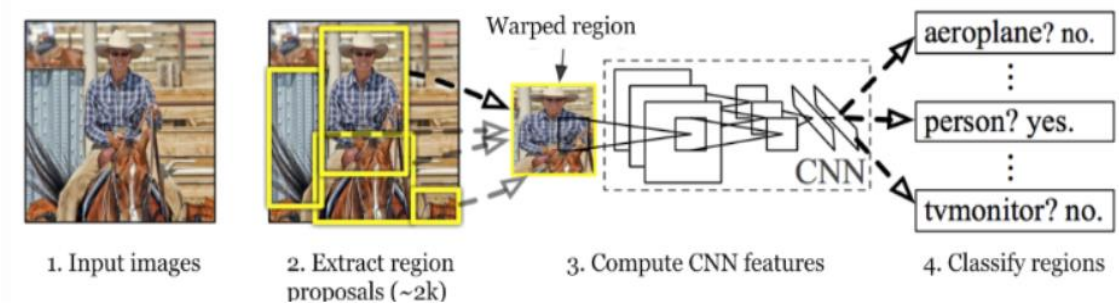
- First Stage: 3 residual blocks with 64 filters
- Second Stage: 4 residual blocks with 128 filters
- Third Stage: 6 residual blocks with 256 filters
- Fourth Stage: 3 residual blocks with 512 filters

The output is passed through Global Average Pooling, followed by a fully connected layer with 1000 neurons and softmax for classification.



9. Analyze the design and working of R-CNN for object detection by examining the roles of selective search, feature extraction, SVM classification, and bounding box regression, and by analyzing the limitations that affect its performance (8) K5

Traditional Convolutional Neural Networks (CNNs) are effective for image classification but perform poorly for object detection tasks, especially when multiple objects of varying sizes and locations appear in an image. Sliding window approaches are computationally expensive and inefficient. To overcome these limitations, R-CNN (Regions with CNN features) was introduced. R-CNN uses a region proposal approach to detect objects efficiently and laid the foundation for modern object detection frameworks.

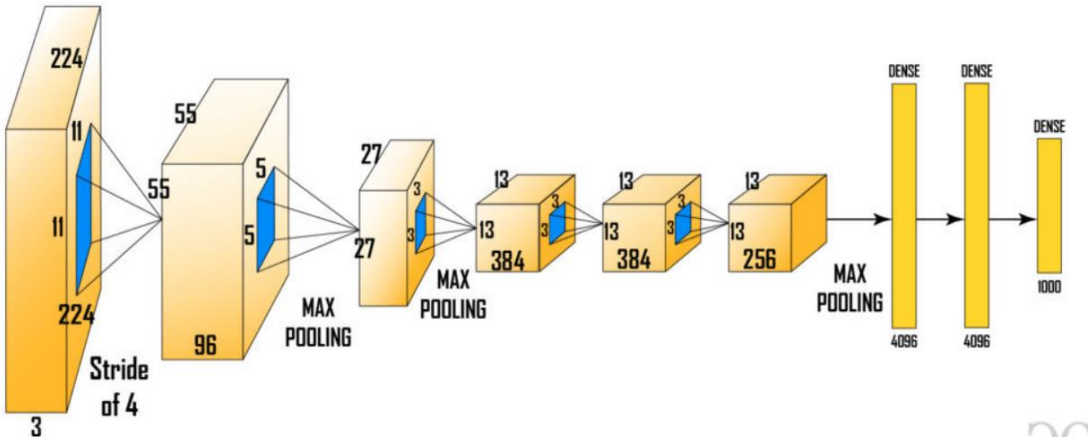


1. Input Image: A single input image containing one or more objects is provided to the system.

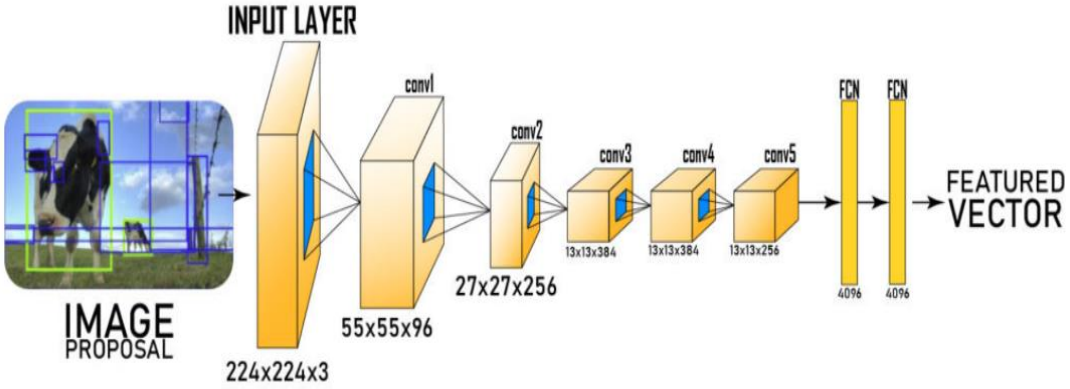
2. Region Proposal Generation: R-CNN uses Selective Search to generate approximately 2,000 region proposals per image. These proposals represent potential object locations.

3. Warping and Feature Extraction

- Cropped from the image
- Resized to a fixed size
- Passed through a CNN (AlexNet) to extract features

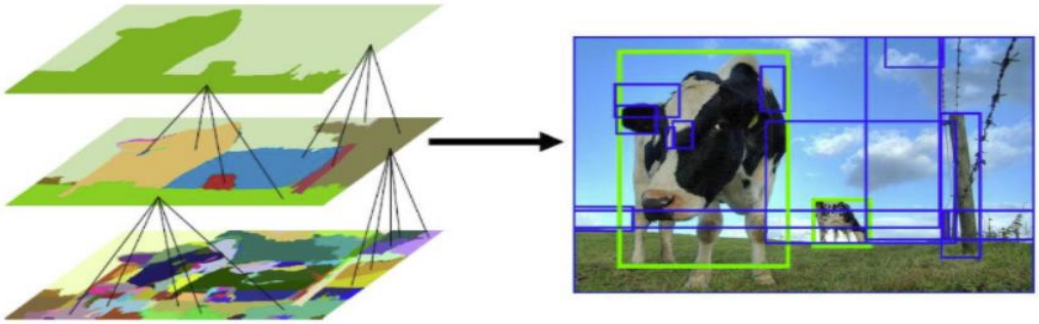


CNN Feature Extraction: The final softmax layer of AlexNet is removed, producing a 4096-dimensional feature vector for each region.



Key Components of R-CNN

A. Selective Search: Selective Search is a greedy algorithm used to generate region proposals.



Algorithm Steps:

1. Perform initial image segmentation
2. Combine similar regions based on color, texture, and size
3. Generate bounding box proposals

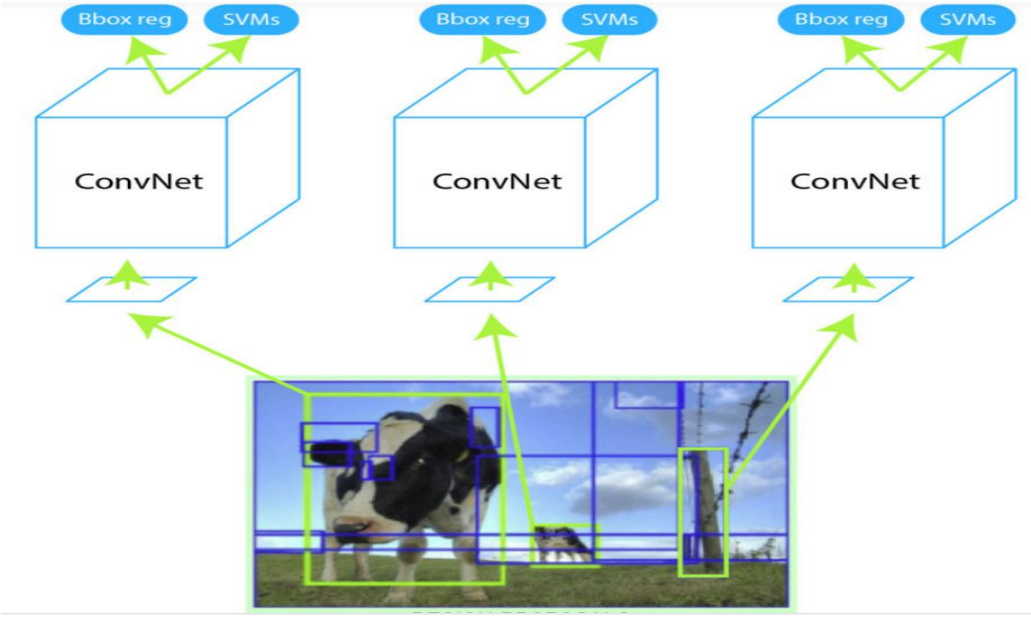
This method balances efficiency and high object recall

B. SVM Classification

Each region's feature vector is classified using a **binary Support Vector Machine (SVM)** trained separately for each object class. The SVM outputs a confidence score for object presence.

Limitation:
CNN and SVM cannot be trained jointly, increasing training complexity.

C. Bounding Box Regression



To improve localization accuracy, a **bounding box regressor** refines predicted boxes using:

$$(x_m, y_m, w_m, h_m)$$

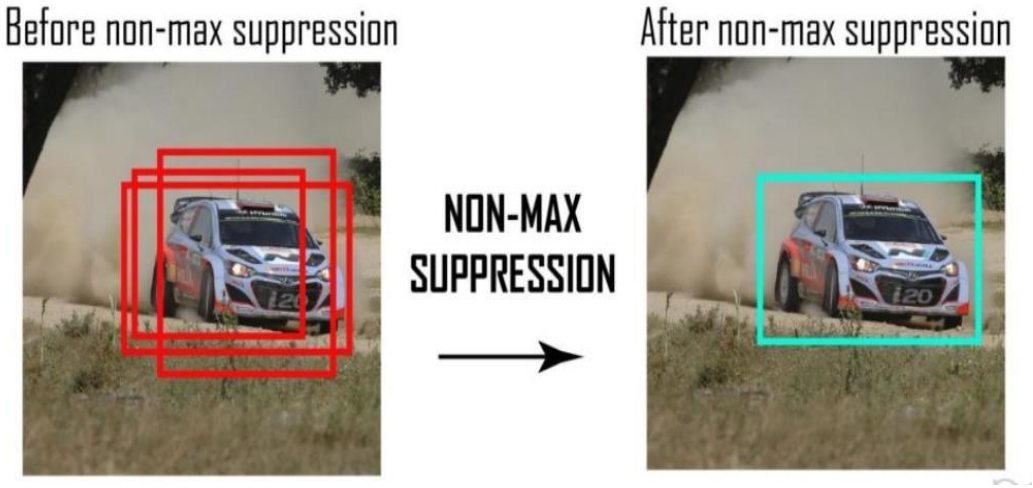
Where:

- x_m, y_m: center coordinates
- w_m, h_m: width and height

This improves **mAP by 3–4%**.

Non-Maximum Suppression (NMS): To eliminate duplicate detections:

- Remove boxes below a confidence threshold
- Retain the highest-scoring box
- Suppress overlapping boxes with IoU > 0.5
- $IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$



Performance of R-CNN

- VOC 2010: mAP = 53.7%
- ILSVRC 2013: mAP = 31.4%
- Test time ≈ 49 seconds per image, making it unsuitable for real-time applications.