Savitribai Phule Pune University
Final Year of Information Technology (2019 Course)
(With effect from Academic Year 2022-23)

Semester VII

Course Code	Course Name	Teaching Scheme(Hours/week)			Examination Scheme and Marks						Credit Scheme			
		Lecture	Practical	Tutorial	Mid-Sem	End-Sem	Termwork	Practical	Oral	Total	Lecture	Practical	Tutorial	Total
414441	Information and Storage Retrieval	03	-	-	30	70	-	-	-	100	3	-	-	3
414442	Software Project Management	03	-	-	30	70	-	-	-	100	3	-	-	3
414443	Deep Learning	03	-	-	30	70	-	-	-	100	3	-	-	3
414444	Elective III	03	-	-	30	70	-	-	-	100	3	-	-	3
414445	Elective IV	03	-	-	30	70	-	-	-	100	3	-	-	3
414446	Lab Practice III	-	04	-	-	-	25	-	25	50	-	2	-	2
414447	Lab Practice IV	-	02	-	-	-	25	25	-	50	-	1	-	1
414448	Project Stage-I	-	-	02	-	-	50	-	-	50	-	-	2	2
414449	Audit Course7													
										Total Credit	15	03	02	20
	Total	15	06	02	150	350	100	25	25	650	15	03	02	20

Elective III:

- Mobile Computing
- High Performance Computing
- Multimedia Technology
- Smart Computing

Elective IV:

- Bioinformatics
- Introduction to DevOps
- Computer Vision
- Wireless Communications

Lab Practice-III:

It is based on subjects:

- Information and Storage Retrieval

Lab Practice-IV:

It is based on subjects:

- Deep Learning

B.E. (INFORMATION TECHNOLOGY)

SEMESTER-VII

Time– 2.00 p.m. to 3.00 p.m.

Day& Date	SUBJECT (2019 Course)	SUBJECT CODE
Tuesday 19/08/2025	Information and Storage Retrieval	414441
Wednesday 20/08/2025	Software Project Management	414442
Thursday 21/08/2025	Deep Learning	414443
Friday 22/08/2025	(ELECTIVE-III) Mobile Computing	414444A
	(ELECTIVE-III) High Performance Computing	414444B
	(ELECTIVE-III) Multimedia Technology	414444C
	(ELECTIVE-III) Smart Computing	414444D
Saturday 23/08/2025	(ELECTIVE-IV)Bioinformatics	414445A
	(ELECTIVE-IV) Introduction to DevOps	414445B
	(ELECTIVE-IV) Computer Vision	414445C
	(ELECTIVE-IV) Wireless Communications	414445D

COURSE CONTENTS		
Unit I	**Fundamentals of Deep Learning**	**(06 hrs)**
What is Deep Learning?, Multilayer Perceptron ,Feed forward neural, Back propagation, Gradient descent, Vanishing gradient problem, **Activation Functions:** RELU, LRELU, ERELU, Optimization Algorithms, **Hyper parameters:** Layer size, Magnitude (momentum, learning rate),Regularization (dropout, drop connect, L1, L2)		
Mapping of Course Outcomes for Unit I	**CO1**	
Unit II	**Convolutional Neural Network:**	**(06 hrs)**
Introduction to CNN, Convolution Operation, Parameter Sharing, Equivariant Representation, Pooling, Variants of the Basic Convolution Function, The basic Architecture of CNN, Popular CNN Architecture – AlexNet.		

Unit II : **"Convolutional Neural Network"**

- **Introduction to CNN**
- **Convolution Operation**
- **Parameter Sharing**
- **Equivariant Representation**
- **Pooling**
- **Variants of the Basic Convolution Function**
- **The basic Architecture of CNN**
- **Popular CNN Architecture – AlexNet**

c) What is regularization? Explain the need for regularization. [5]

Q3) a) Explain the following hyper parameters for the convolutional layer. **[8]**

i) Filter size
ii) Output depth
iii) Stride
iv) Zero-padding

b) Explain convolution operation in CNN with a suitable example. Take 5*5 input data, 3*3 kernel data and calculate convoluted features. **[7]**

OR

Q4) a) Draw and explain architecture of AlexNet. **[7]**

b) Explain any four applications of CNNs with suitable diagrams. **[8]**

Q3)	a)	Illustrate Convolution operation in CNN with an example.	**[5]**
	b)	Explain the use of padding and strides in pooling layers.	**[5]**
	c)	What is the advantage of weight sharing in CNN.	**[5]**

OR

Q4)	a)	What are pooling layers in CNN? Illustrate Max pooling with an example.	**[5]**
	b)	Discuss applications of CNN.	**[5]**
	c)	Write short note on AlexNet.	**[5]**

Q3)	a)	Explain Alexnet architecture as per its layer.	**[5]**
	b)	Explain types of pooling with example.	**[5]**
	c)	Enlist and explain layers of CNN.	**[5]**

OR

Q4)	a)	Explain CNN architecture also explain functions of Hidden Layers.	**[7]**
	b)	What is convolution operation ? Explain circular and discrete convolution operation?	**[8]**

1. CNN kya hai?

CNN ek deep learning neural network architecture hai jo specially image data ke liye design kiya gaya hai.

•Traditional neural network (MLP) me har neuron har input pixel se connected hota hai — isse parameters bahut zyada hote hain.

•CNN me Convolution Operation use hota hai jo images ke features (edges, textures, shapes) automatically nikal leta hai, aur parameters ka number kaafi kam ho jata hai.

Short idea: CNN ka kaam hai *"features extract karna + classification karna"*.

2. CNN ka Motivation

Human vision system (Visual Cortex) images ko samajhne ke liye **receptive fields** ka use karta hai — matlab har neuron sirf ek chhote hissa (region) ko dekhta hai, pure image ko nahi. CNN isi concept par based hai.

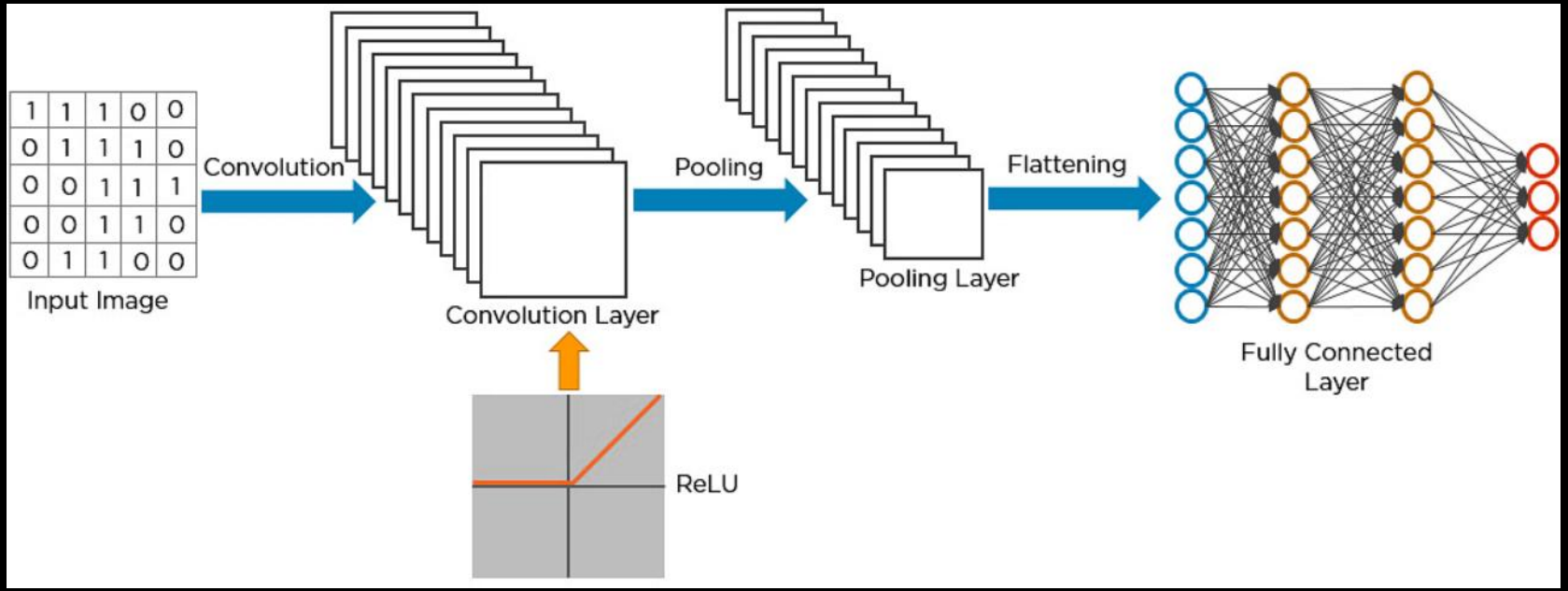
CNN (Convolutional Neural Network):

Convolutional Neural Networks (CNNs) specially design kiye gaye hote hain images ke saath kaam karne ke liye.
Ek image pixels se milkar banti hai. Deep learning me images ko pixel values ke arrays ke form me represent kiya jata hai.

CNN me 4 main types ke layers hote hain:

- Convolutional layers
- ReLU
- Pooling layers
- Fully connected (dense) layers

CNN me 4 main types ke operations hote hain:
Convolution operation, Pooling operation, Flatten operation aur Classification (ya koi other relevant) operation.


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
Input Image
Convolution
Convolution Layer
ReLU
Pooling
Pooling Layer
Flattening
Fully Connected Layer

Step	Process	Explanation	Real-Life Example
1	Input Image	Image ek matrix of pixels hoti hai. Har pixel ka value brightness ya color dikhata hai (grayscale: 0–255, RGB: 3 values).	Tum phone se ek dog ka photo lete ho. Ye photo pixels ka grid hai jo CNN ka input banega.
2	Convolution	Small filter (e.g., 3×3 kernel) image par slide hota hai. Har filter ek pattern (jaise edge, texture) detect karta hai. Kai filters lagake multiple feature maps ban jaate hain.	Ek filter dog ki ear edges detect karega, dusra fur ka texture, teesra eyes ke around shape.
3	ReLU Activation	Negative values ko 0 banata hai, positive values same rakhta hai. Ye network ko non-linear banata hai taki complex patterns seekh sake.	Agar ear edge ka value negative aya to ignore karega, sirf positive strong edges rakhega.
4	Pooling	Feature maps ka size kam karta hai (e.g., Max Pooling 2×2). Important features retain hote hain, computation kam hota hai, aur small shifts ka effect kam hota hai.	Agar dog thoda left/right ho to pooling ensure karega ki ear detection still valid rahe.
5	Flattening	Pooling ke baad ka 3D tensor ek 1D vector me convert hota hai taaki Fully Connected layer me feed ho sake.	Ear, eye, fur ke features ek lambi list ban ke final decision ke liye ready ho jaate hain.
6	Fully Connected Layer	Har neuron sab neurons se connect hota hai. Ye features ko combine karke final classification karta hai.	"Ear + Eyes + Fur Texture" ka combination network ko batata hai ki ye shayad dog hai.
7	Output Layer	Final probabilities nikalta hai (Softmax multi-class ke liye, Sigmoid binary ke liye).	Output: Dog = 0.92, Cat = 0.05, Horse = 0.03 → Highest probability = Dog.

Explain the following hyper parameters for the convolutional layer. **[8]**

i) Filter size
ii) Output depth
iii) Stride
iv) Zero-padding

Hyperparameter	Explanation (Easy Words)	Example
i) Filter Size (Kernel Size)	Convolution me jo chhota square matrix lagate hain (filter/kernel) uska size hota hai filter size. Ye decide karta hai ki ek time me kitne pixels ka area dekh rahe ho. Common values: 3×3, 5×5.	Agar filter size 3×3 hai, iska matlab ek chhote 3×3 pixels ka block image me slide karega aur feature detect karega (e.g., edge detection).
ii) Output Depth	Output depth = number of filters. Har filter ek different feature detect karta hai, to jitne zyada filters, utni zyada feature maps.	Agar tum 32 filters lagate ho, to output me 32 different feature maps milenge (ek edges detect karega, ek texture, ek corners, etc.).
iii) Stride	Stride ka matlab hai filter ko slide karte waqt kitne pixels ka jump lena. Stride = 1 matlab har ek pixel move hoga, Stride = 2 matlab 2 pixels ka jump.	Agar 3×3 filter stride = 1 ke saath use karo to smooth detail milegi, stride = 2 se image ka size jaldi reduce ho jayega.
iv) Zero-Padding	Image ke borders me extra 0 values add karna taaki convolution ke baad size kam na ho ya edge features bhi detect ho sake.	Agar tum ek 5×5 image pe 3×3 filter stride 1 use karte ho bina padding, to output 3×3 banega. Agar zero-padding 1 use karo to output 5×5 banega aur border features bhi detect honge.

📌 Step-by-Step Explanation

1. Input Image

•Yahan pe ek **5×5 grayscale image** dikhaya gaya hai:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

•1 means bright pixel, 0 means dark pixel.
•Example: Yeh kisi digit “1” ka chhota photo ho sakta hai.

•Convolution Operation

b) Explain convolution operation in CNN with a suitable example. Take 5*5 input data, 3*3 kernel data and calculate convoluted features. [7]

a) Illustrate Convolution operation in CNN with an example.

b) What is convolution operation ? Explain circular and discrete convolution operation? [8]

Definition
Convolution operation ek mathematical process hai jisme ek small matrix (kernel/filter) ko input image ke upar slide kiya jaata hai. Har position par **element-wise multiplication** kiya jaata hai aur result ka **sum** nikala jaata hai. Ye process CNN me feature extraction ke liye hota hai.

Purpose in CNN
•Automatically **features detect** karna (edges, corners, textures).
•Reduce **manual feature engineering**.
•Preserve **spatial relationships** between pixels.

Formula for Output Size

$$\text{Output size} = \frac{N - K + 2P}{S} + 1$$

Where:

- N = input size
- K = kernel size
- P = padding
- S = stride

Given Data

Input Image (5×5):

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

Kernel (3×3):

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

Stride (S) = 1, Padding (P) = 0

Step-1

Output size:

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

So, output = **3×3**

Position (0,0) → output[0,0]

Patch (top-left 3×3):

```
1 1 1
0 1 1
0 0 1
```

Kernel:

```
1 0 1
0 1 0
1 0 1
```

Element-wise multiply (patch * kernel):

```
1*1  1*0  1*1   =>  1 0 1
0*0  1*1  1*0   =>  0 1 0
0*1  0*0  1*1   =>  0 0 1
```

Sum = 1+0+1 + 0+1+0 + 0+0+1 = **4**
→ **output[0,0] = 4**

Position (0,1) → output[0,1]

Patch:

```
1 1 0
1 1 1
0 1 1
```

Kernel:

```
1 0 1
0 1 0
1 0 1
```

```
1*1  1*0  0*1   =>  1 0 0
1*0  1*1  1*0   =>  0 1 0
0*1  1*0  1*1   =>  0 0 1
```

Sum = 1+0+0 + 0+1+0 + 0+0+1 = **3**
→ **output[0,1] = 3**

Position (0,2) → output[0,2]
Patch:

1 0 0
1 1 0
1 1 1

1*1 0*0 0*1 => 1 0 0
1*0 1*1 0*0 => 0 1 0
1*1 1*0 1*1 => 1 0 1

Sum = 1+0+0 + 0+1+0 + 1+0+1 = **4**
→ output[0,2] = 4

Position (1,0) → output[1,0]
Patch:

```
0 1 1
0 0 1
0 0 1
```

```
0*1  1*0  1*1   =>  0 0 1
0*0  0*1  1*0   =>  0 0 0
0*1  0*0  1*1   =>  0 0 1
```

Sum = 0+0+1 + 0+0+0 + 0+0+1 = **2**
→ output[1,0] = 2

Position (1,1) → output[1,1]
Patch:

1 1 1
0 1 1
0 1 1

1*1 1*0 1*1 => 1 0 1
0*0 1*1 1*0 => 0 1 0
0*1 1*0 1*1 => 0 0 1

Sum = 1+0+1 + 0+1+0 + 0+0+1 = **4**
→ **output[1,1] = 4**

Position (1,2) → output[1,2]
Patch:

```
1 1 0
1 1 1
1 1 0
```

```
1*1  1*0  0*1   =>  1 0 0
1*0  1*1  1*0   =>  0 1 0
1*1  1*0  0*1   =>  1 0 0
```

Sum = 1+0+0 + 0+1+0 + 1+0+0 = **3**
→ output[1,2] = 3

Position (2,0) → output[2,0]
Patch:

0 0 1
0 0 1
0 1 1

0*1 0*0 1*1 => 0 0 1
0*0 0*1 1*0 => 0 0 0
0*1 1*0 1*1 => 0 0 1

Sum = 0+0+1 + 0+0+0 + 0+0+1 = **2**
→ output[2,0] = 2

Position (2,1) → output[2,1]
Patch:

0 1 1
0 1 1
1 1 0

0*1 1*0 1*1 => 0 0 1
0*0 1*1 1*0 => 0 1 0
1*1 1*0 0*1 => 1 0 0

Sum = 0+0+1 + 0+1+0 + 1+0+0 = **3**
→ **output[2,1] = 3**

Position (2,2) → output[2,2]
Patch:

1 1 1
1 1 0
1 0 0

1*1 1*0 1*1 => 1 0 1
1*0 1*1 0*0 => 0 1 0
1*1 0*0 0*1 => 1 0 0

Sum = 1+0+1 + 0+1+0 + 1+0+0 = **4**
→ **output[2,2] = 4**

Final Convolved Feature Map (3×3)

4	3	4
2	4	3
2	3	4

b) What is convolution operation ? Explain circular and discrete convolution operation? [8]

1. Convolution Operation – Definition

Convolution operation ek mathematical process hai jo **do signals** (ya functions) ko combine karke ek naya signal banata hai, jo batata hai ki ek signal dusre par kaise overlap hota hai.

•**Signal Processing me:** Filter apply karna, smooth karna, edge detection, etc.

•**CNN me:** Image ke upar kernel/filter slide karke **features detect** karna.

Mathematical form (Discrete):

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

yahan:

- $x[k]$ = input signal
- $h[k]$ = filter/kernel
- $y[n]$ = output (convolved signal)

2. Discrete Convolution

•Input aur filter **discrete points** (jaise pixels) par defined hote hain.

•Kernel ko flip karke, har position par multiply-sum kiya jata hai.

Example:

Let input x=[1,2,3], kernel h=[0,1,0.5]

Step-by-step:

n=0: (1*0) + (0*1) + (0*0.5) = 0

n=1: (2*0) + (1*1) + (0*0.5) = 1

n=2: (3*0) + (2*1) + (1*0.5) = 2 + 0.5 = 2.5

...

Ye method image convolution ka base hai.

3. Circular Convolution

•Isme convolution karte waqt signals ko **wrap-around** karke treat kiya jata hai.

•Matlab index agar last point se aage badh jaye, to start se values le li jati hain.

•Mostly **periodic signals** ya **DFT (Discrete Fourier Transform)** ke saath use hota hai.

Formula:

$$y[n] = \sum_{m=0}^{N-1} x[m] \cdot h[(n-m) \bmod N]$$

yahan mod operation wrap-around karta hai.

Example:

Let $x = [1, 2, 3]$, $h = [4, 5, 6]$, $N = 3$

y[0] = 1*4 + 2*6 + 3*5 = 4 + 12 + 15 = 31
y[1] = 1*5 + 2*4 + 3*6 = 5 + 8 + 18 = 31
y[2] = 1*6 + 2*5 + 3*4 = 6 + 10 + 12 = 28

Feature	Discrete Convolution	Circular Convolution
Signal type	Finite length or infinite	Periodic / wrapped
Index handling	Zero padding outside limits	Wrap-around (mod operation)
Applications	Filtering, CNN, edge detection	Fourier analysis, cyclic signals

•Parameter Sharing

Definition
Parameter sharing ka matlab hota hai ki **ek hi set of filter weights (kernel)** ko image ke har position par apply kiya jata hai, instead of har pixel ke liye alag weights rakhne ke.
Isse:
•**Parameters kam hote hain** → model chhota aur fast hota hai.
•**Generalization improve hoti hai** → kyunki ek hi feature (e.g., edge) kahi bhi detect ho sakta hai.

Why needed?

Agar ek image $32 \times 32 \times 3$ (RGB) ka hai aur fully connected layer use karein:

- Parameters = $32 \times 32 \times 3 \times \text{no. of neurons}$
- Bahut zyada weights ban jaate → training slow, overfitting risk.

CNN me:

- Same filter (e.g., edge detector) ko **poori image par slide** karke apply karte hain → weights **share** ho jaate hain.

Example
Maan lo ek filter (kernel) size 3×3 hai:

```
K = [[1, 0, -1],
     [1, 0, -1],
     [1, 0, -1]]
```

Ye filter vertical edges detect karta hai.

CNN me:
•Yehi filter poore image me har 3×3 patch par apply hoga.
•Har jagah **same weights** ka use hoga, bas input patch change hota rahega.

Benefits

1. **Less parameters** → kam memory + fast computation.
2. **Translation invariance** → ek pattern kahin bhi aaye, detect ho jayega.
3. **Better generalization** → model specific pixel location pe dependent nahi hota.

c) What is the advantage of weight sharing in CNN.

Advantages of Weight Sharing in CNN

1. **Fewer Parameters**
   - Har pixel ke liye alag weight store karne ki zarurat nahi hoti.
   - Memory requirement bahut kam ho jata hai.
   - Example: Ek 100×100 image ke liye agar 3×3 filter use karein, toh sirf 9 weights store hote hain, na ki 10,000 alag weights.
2. **Faster Computation**
   - Kam parameters = kam multiplications and additions = faster training & inference.
3. **Avoids Overfitting**
   - Kam parameters hone se model zyada generalize karta hai, overfitting ka risk kam hota hai.
4. **Translation Invariance**
   - Ek pattern (e.g., edge, corner) chahe image me kahin bhi ho, same filter usko detect karega.
5. **Better Feature Extraction**
   - Same filter poore image me slide hota hai, toh consistent feature detection hoti hai.
6. **Scalability**
   - Large images par bhi efficiently kaam karta hai, kyunki parameters image size ke saath exponentially nahi badte.

b) Explain the use of padding and strides in pooling layers.

a) What are pooling layers in CNN? Illustrate Max pooling with an example. **[5]**

b) Explain types of pooling with example.

Pooling in CNN

Definition:
Pooling ek operation hai jo **feature map ka size reduce** karta hai, taaki:
•Computation fast ho
•Memory kam use ho
•Features more robust ho (position changes, noise ke against)
Pooling me hum ek **small window** (jaise 2×2) ko feature map pe slide karte hain aur har window ka ek single value nikalte hain (jaise max ya average).

Purpose of Pooling:

1.Dimensionality Reduction – feature maps chhote hote hain.

2.Noise Reduction – random variations ka effect kam hota hai.

3.Translation Invariance – object thoda shift ho toh bhi detect ho sakta hai.

b) Explain types of pooling with example.

1. **Max Pooling**

Definition:
Max Pooling me ek small window (jaise 2×2) ko feature map ke upar slide karte hain, aur har window ka **sabse bada (maximum)** value output me lete hain.

Purpose:
•Sabse strong feature ko preserve karta hai.
•Unimportant (chhote) pixel values ignore ho jaati hain.
•Noise ka effect kam hota hai.

Example:

Feature map:

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

2×2 Max Pooling (stride = 2):

- Window 1: $\begin{bmatrix} 1 & 3 \\ 4 & 6 \end{bmatrix} \rightarrow 6$
- Window 2: $\begin{bmatrix} 2 & 0 \\ 5 & 2 \end{bmatrix} \rightarrow 5$
- Window 3: $\begin{bmatrix} 7 & 3 \\ 0 & 2 \end{bmatrix} \rightarrow 7$
- Window 4: $\begin{bmatrix} 8 & 1 \\ 3 & 4 \end{bmatrix} \rightarrow 8$

Output:

$$\begin{bmatrix} 6 & 5 \\ 7 & 8 \end{bmatrix}$$

2. Average Pooling

Definition:
Average Pooling me har window ka **average value** output me liya jaata hai.

Purpose:
•Feature map smooth banata hai.
•Sabhi pixels ka contribution hota hai.
•Thoda detail loose ho sakta hai.

Example:

Same feature map:

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

2×2 Average Pooling (stride = 2):

- Window 1: Avg = $\frac{1+3+4+6}{4} = 3.5$
- Window 2: Avg = $\frac{2+0+5+2}{4} = 2.25$
- Window 3: Avg = $\frac{7+3+0+2}{4} = 3$
- Window 4: Avg = $\frac{8+1+3+4}{4} = 4$

Output:

$$\begin{bmatrix} 3.5 & 2.25 \\ 3 & 4 \end{bmatrix}$$

3. Global Pooling

Definition:
Pura feature map se ek single value nikalta hai — ya to max ya average.
Mostly **Global Average Pooling (GAP)** classification me use hota hai.

Purpose:
•Feature map ko directly ek vector me convert karta hai.
•Fully Connected layer ka size reduce karta hai.

Example:

Feature map:

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 6 & 5 \\ 7 & 3 & 8 \end{bmatrix}$$

- **Global Max Pooling** → max = 8
- **Global Average Pooling** → avg = $\frac{1+3+2+4+6+5+7+3+8}{9} = 4.33$

1. Padding in Pooling

Definition:
Padding ka matlab hota hai feature map ke border me extra pixels (usually 0) add karna before pooling.

Why used in pooling:

1.Border features preserve karna – bina padding ke, border ke pixels kam use hote hain pooling me.

2.Exact output size control – agar tum chahte ho ki pooling ke baad size exactly half ya specific ho, padding se adjust hota hai.

3."Same" pooling implement karna – output size input size ke proportional rakhta hai.

Types:
- Valid Padding (No Padding) → border ke pixels partially use hote hain, output smaller hota hai.
- Same Padding (Zero Padding) → enough zeros add kiye jate hain taaki output size controlled rahe.

2. Strides in Pooling

Definition:
Stride ka matlab hota hai window shift ka step size jab pooling window slide hoti hai.

Why used in pooling:

1.Output size control – stride zyada ho toh output chhota hota hai.
2.Down sampling – large stride se feature map ka size jaldi reduce hota hai.
3.Overlapping ya Non-overlapping pooling decide karta hai.

Example:

Input Feature Map size: 4×4, Pool size: 2×2

Case 1: Stride = 2, No Padding

- Window har baar **2 steps** move karega → Non-overlapping pooling
- Output size: $\frac{4-2}{2} + 1 = 2$

Case 2: Stride = 1, No Padding

- Window **1 step** move karega → Overlapping pooling
- Output size: $\frac{4-2}{1} + 1 = 3$

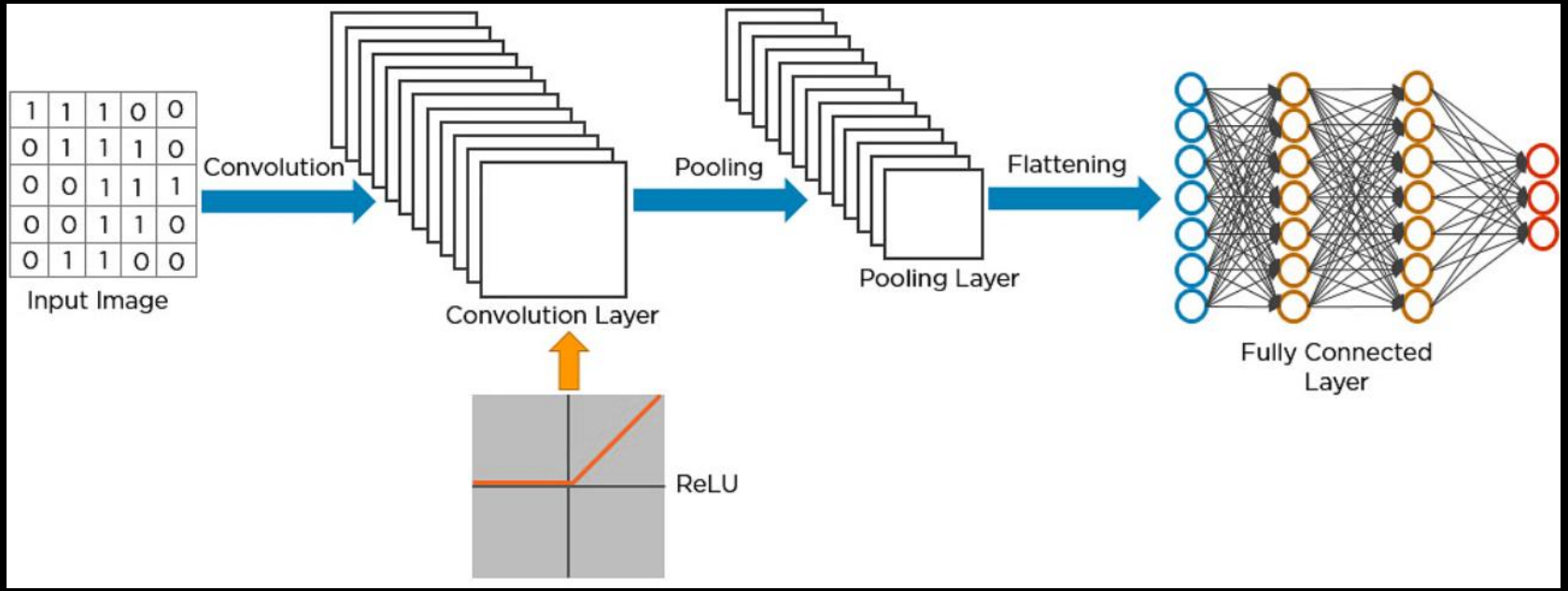
Case 3: Stride = 2, Same Padding

- Border me zeros add karke ensure ki output size zyada ho.
- Output size formula (Same Padding):

$$\text{Output size} = \left\lceil \frac{\text{Input size}}{\text{Stride}} \right\rceil$$

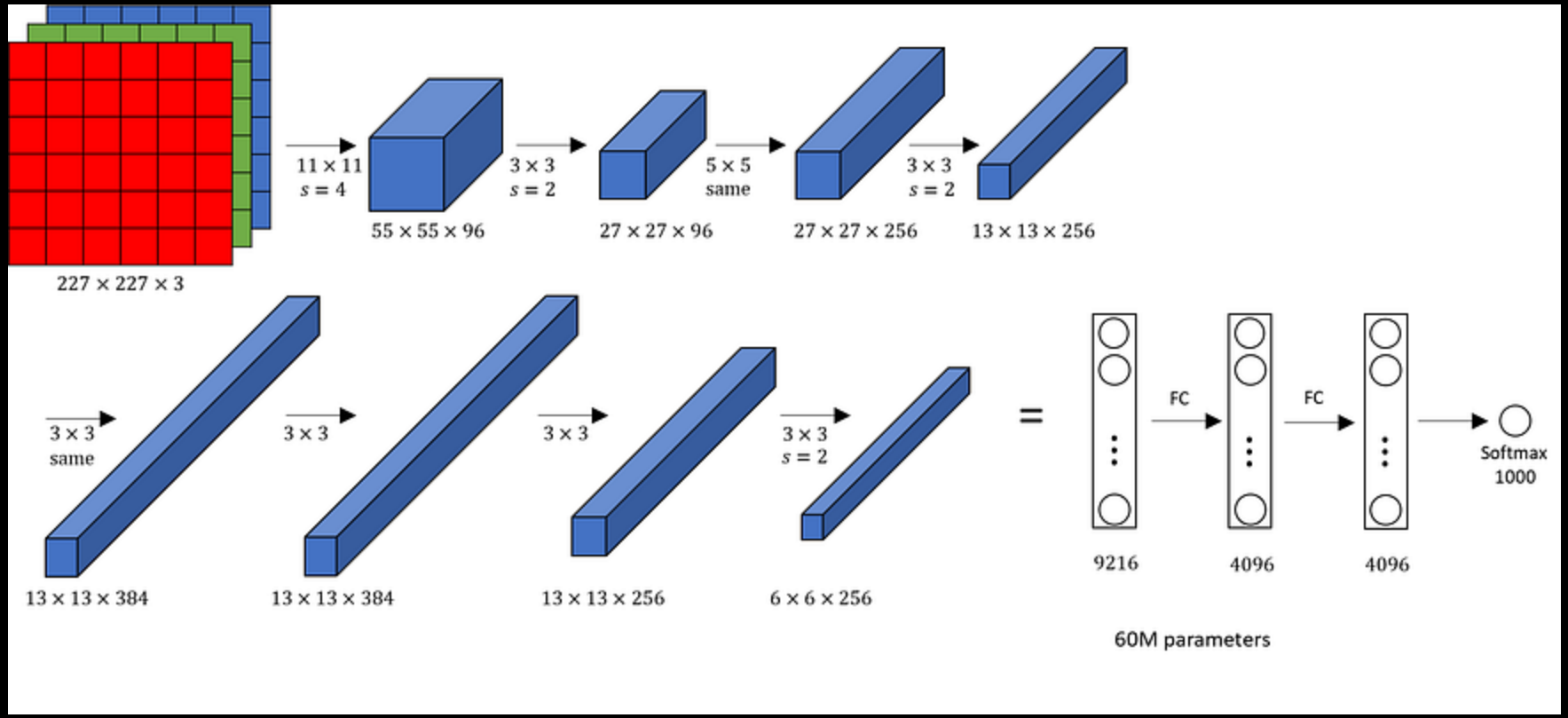
a) Explain CNN architecture also explain functions of Hidden Layers. [7]

CNN me **hidden layers** ka matlab hota hai wo layers jo **input layer** (raw image) aur **output layer** (final prediction) ke beech me hoti hain. Ye layers image ke raw pixels ko step-by-step transform karke useful features me convert karti hain.


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
Input Image
Convolution
Convolution Layer
ReLU
Pooling
Pooling Layer
Flattening
Fully Connected Layer

Layer Type	Kaam (Function)	Example / Note
1. Convolutional Layer	Feature extraction — filters (kernels) use karke patterns detect karta hai (edges, corners, textures, shapes).	3×3 filter detect karega vertical edge, 5×5 filter detect karega complex texture.
2. Activation Layer (ReLU)	Non-linearity introduce karta hai taaki complex patterns seekhe jaa saken.	max(0, x) negative values ko 0 kar deta hai.
3. Pooling Layer	Feature maps ka size reduce karta hai, important information retain karta hai, computation kam karta hai.	Max pooling 2×2 me 4 pixels me se largest pixel value rakhta hai.
4. Batch Normalization Layer (optional)	Training fast karta hai aur network stable banata hai.	Har layer ka output normalize karta hai (mean=0, variance=1 ke around).
5. Dropout Layer (optional)	Overfitting kam karta hai by randomly neurons ko deactivate karna during training.	0.5 dropout = half neurons randomly ignore.
6. Flatten Layer	Multi-dimensional feature maps ko 1D vector me convert karta hai.	7×7×64 → 3136 vector.
7. Fully Connected (Dense) Layer	Extracted features ka combination banata hai aur final decision karta hai.	Agar dog vs cat classification hai, last dense layer output [Dog=0.92, Cat=0.08].

Q4)	a)	Draw and explain architecture of AlexNet.	**[7]**
	b)	Explain any four applications of CNNs with suitable diagrams.	**[8]**


227 × 227 × 3
11 × 11
s = 4
55 × 55 × 96
3 × 3
s = 2
27 × 27 × 96
5 × 5
same
27 × 27 × 256
3 × 3
s = 2
13 × 13 × 256
3 × 3
same
13 × 13 × 384
3 × 3
13 × 13 × 384
3 × 3
13 × 13 × 256
3 × 3
s = 2
6 × 6 × 256
=
9216
FC
4096
FC
4096
Softmax
1000
60M parameters

Output size formula (har convolution/pooling ke liye):

> `Out = floor((W - F + 2P) / S) + 1`
>
> jahaan W = input width, F = filter size, P = padding, S = stride.

1) Input

- **Image:** `227 × 227 × 3` (RGB image).

  So width=227, height=227, channels=3.

2) Conv1

- **Filter:** `11×11`, **stride** = 4, **filters** = 96, **padding** = 0 (no pad).
- **Calculation:** `Out = floor((227 - 11 + 0)/4) + 1 = floor(216/4)+1 = 54+1 = 55`.
- **Output:** `55 × 55 × 96`.
- **Then:** ReLU → *Local Response Normalization (LRN)* → **Max-Pooling** `3×3`, stride 2 (overlapping pooling).
- **After pooling:** `Out = floor((55 - 3)/2) + 1 = floor(52/2)+1 = 26+1 = 27` → `27 × 27 × 96`.

Simple idea: Pehli layer badi filter se image ko chhota kar deti hai (spatially compress).

3) Conv2

- **Filter:** `5x5`, stride 1, **padding** = 2 (`same`), **filters** = 256.
- **Because padding =2**, spatial size stays same: `27 x 27 x 256`.
- **Then:** ReLU → LRN → Max-Pooling `3x3`, stride 2.
- **After pooling:** `Out = floor((27 - 3)/2) + 1 = floor(24/2)+1 = 12+1 = 13` → `13 x 13 x 256`.

Idea: Ab features jyada channels (256) mein represent ho rahe hain — network complex patterns seekh raha hai.

4) Conv3

- **Filter:** `3×3`, stride 1, padding = 1 (`same`), **filters** = 384.
- **Output:** `13 × 13 × 384`.
- **No pooling immediately** — continue extracting patterns.

---

5) Conv4

- `3×3`, stride 1, padding 1, **filters** = 384 → **Output:** `13 × 13 × 384`.

---

6) Conv5

- `3×3`, stride 1, padding 1, **filters** = 256 → **Output:** `13 × 13 × 256`.
- **Then pooling:** `3×3`, stride 2 → `Out = floor((13 - 3)/2) + 1 = floor(10/2)+1 = 5+1 = 6` → `6 × 6 × 256`.

7) Flatten → Fully Connected (FC)

- **Flatten:** `6 × 6 × 256 = 9216` values (yeh ek long vector ban gaya).
- **FC1:** 9216 → **4096** neurons → ReLU → Dropout.
- **FC2:** 4096 → **4096** neurons → ReLU → Dropout.
- **FC3 (final):** 4096 → **1000** neurons → **Softmax** (1000 class probabilities).

---

8) Parameters & Extras

- **Total parameters ≈ 60 million** (bahut saare weights, mostly FC layers mein).
- **Activation:** ReLU (fast training).
- **LRN:** original AlexNet mein use hua — aap modern nets mein zyaada nahi dekhoge.
- **Overlapping pooling:** pool size 3, stride 2 (isse slightly better performance milta tha).
- **Note:** Original AlexNet GPU-splitting and data augmentation tricks use karta tha.

1️⃣ Input Image

`227 × 227 × 3` → RGB photo

⬇️

2️⃣ Conv1

- Filter: 11×11, stride 4
- Output: `55 × 55 × 96`
- ReLU → LRN → Max Pool (3×3, s=2)
- After Pool: `27 × 27 × 96`

⬇️

3️⃣ Conv2

- Filter: 5×5, stride 1, same padding
- Output: `27 × 27 × 256`
- ReLU → LRN → Max Pool (3×3, s=2)
- After Pool: `13 × 13 × 256`

⬇️

Bas yaad rakho —
11 → 5 → 3 → 3 → 3 → FC → FC → FC
Aur pooling hamesha Conv1, Conv2, Conv5 ke baad hoti hai.

⬇️

4️⃣ Conv3

- Filter: 3×3, stride 1, same padding
- Output: `13 × 13 × 384`

⬇️

5️⃣ Conv4

- Filter: 3×3, stride 1, same padding
- Output: `13 × 13 × 384`

⬇️

6️⃣ Conv5

- Filter: 3×3, stride 1, same padding
- Output: `13 × 13 × 256`
- Max Pool (3×3, s=2) → `6 × 6 × 256`

⬇️

7️⃣ Flatten

`6 × 6 × 256 = 9216` values

⬇️

8️⃣ FC1

9216 → 4096 neurons → ReLU → Dropout

⬇️

9️⃣ FC2

4096 → 4096 neurons → ReLU → Dropout

⬇️

🔟 FC3 (Output Layer)

4096 → 1000 neurons → Softmax → Class probabilities

b) Explain any four applications of CNNs with suitable diagrams. [8]

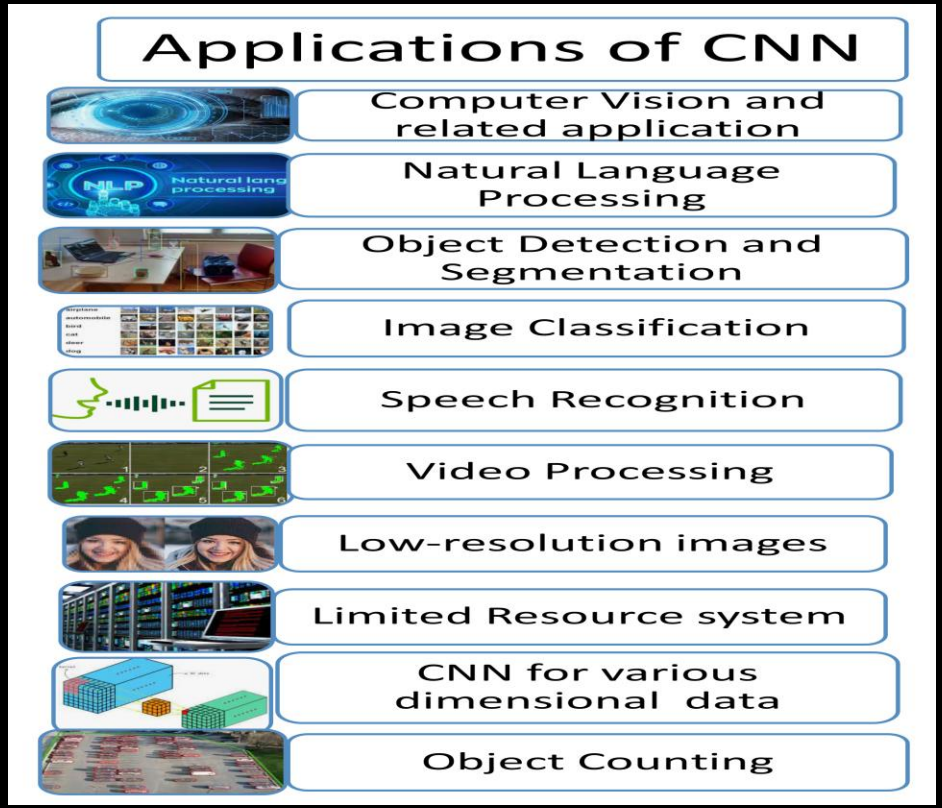
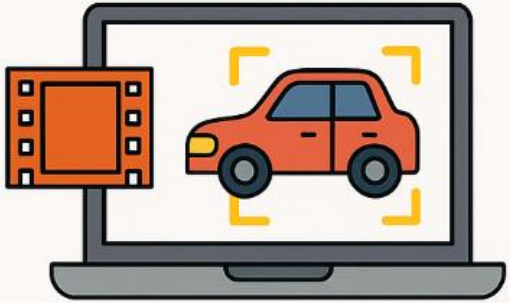
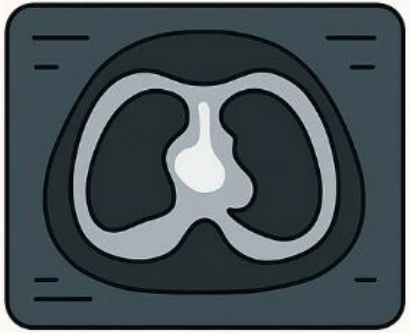


IMAGE & VIDEO RECOGNITION



MEDICAL IMAGING



AUTONOMOUS VEHICLES

FACIAL RECOGNITION & EMOTION DETECTION