

DL – Unit 3 (Convolution Neural Network) – END-SEM PYQ Answers

MAY-JUNE 2023

Q1a Explain Pooling Layer with its need and different types.

[6 Marks]

Pooling Layer — Overview

A Pooling Layer (also called a sub-sampling or down-sampling layer) is placed after a Convolution Layer to reduce the spatial dimensions (width × height) of the feature maps while retaining the most important information. This reduces the number of parameters, mitigates overfitting, and makes the network more computationally tractable.

Need for Pooling

- Reduces spatial size of feature maps, lowering computation cost.
- Provides spatial invariance — small translations of the input do not change the pooled output.
- Reduces the number of learnable parameters, helping prevent overfitting.
- Makes features more abstract and robust to noise.

Types of Pooling

- **Max Pooling:** Selects the maximum value from each pooling window. Retains the most prominent feature in a region. Most commonly used.
- **Average Pooling:** Computes the average of values in each pooling window. Gives a smoother representation; used in some modern architectures.
- **Global Average Pooling (GAP):** Reduces each feature map to a single number (its mean). Often replaces fully-connected layers at the end of CNNs (e.g., GoogLeNet).
- **Min Pooling:** Selects the minimum value — rarely used in practice.
- **Stochastic Pooling:** Samples a value from the pooling region according to a multinomial distribution proportional to activations; used for regularisation.

Max Pooling — Numerical Example

Input feature map (4×4): After 2×2 Max Pool (stride 2):

1	3	2	4	→	3	4
5	6	1	2		6	3
7	1	3	0			

Note: Pooling has NO learnable parameters. The window size (kernel) and stride are hyperparameters set by the designer. Typical choice: 2×2 kernel, stride 2 — halves each spatial dimension.

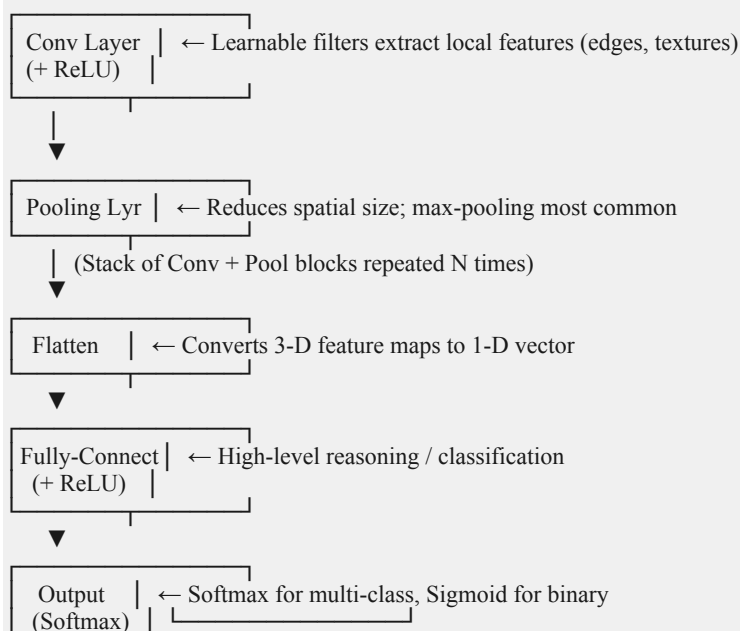
Q1b Draw and explain CNN (Convolution Neural Network) architecture in detail. [6 Marks]

CNN Architecture

A Convolutional Neural Network is a specialised deep neural network designed for grid-structured data such as images. Its architecture chains layers that progressively extract, refine, and classify features.

Architecture Flowchart

Input Image
↓
▼



Layer-by-Layer Explanation

- **Input Layer:** Raw pixel values of the image (e.g., 224×224×3 for RGB).
- **Convolutional Layer:** Applies a set of learnable filters (kernels) that slide over the input to produce feature maps. Each filter detects a specific local pattern. Operation: $\text{Output} = \text{ReLU}(\text{Input} * \text{Filter} + \text{bias})$.
- **Activation (ReLU):** Introduces non-linearity. Replaces negative values with 0: $f(x) = \max(0, x)$.
- **Pooling Layer:** Down-samples the feature maps (see Q1a). Reduces dimensions while preserving dominant features.
- **Flatten Layer:** Reshapes the multi-dimensional feature maps into a 1-D vector to feed into dense layers.
- **Fully Connected (Dense) Layers:** Perform classification or regression based on the extracted features.
- **Output Layer:** Uses Softmax for multi-class classification to produce a probability distribution over classes.

Applications of CNNs

- Image Classification (ImageNet, CIFAR-10)
- Object Detection (YOLO, Faster-RCNN)
- Face Recognition
- Medical Image Analysis (tumour detection, X-ray diagnosis)
- Self-driving car perception
- Natural Language Processing (1-D convolutions over text)

Note: Landmark CNN architectures: LeNet-5 (1998), AlexNet (2012), VGGNet, GoogLeNet (Inception), ResNet, EfficientNet. ResNet introduced residual/skip connections to allow training of very deep networks (100+ layers).

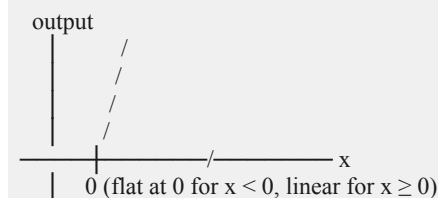
Q1c Explain ReLU Layer in detail. What are the advantages of ReLU over Sigmoid? [6 Marks]

ReLU (Rectified Linear Unit) — Activation Layer

ReLU is the most widely used activation function in deep learning. It is applied element-wise after a convolution (or any linear operation) to introduce non-linearity, enabling the network to learn complex decision boundaries.

Mathematical Definition

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



Advantages of ReLU over Sigmoid

Property	ReLU vs Sigmoid
Vanishing Gradient	ReLU does NOT saturate for $x > 0$, so gradients flow freely. Sigmoid saturates near 0/1 causing near-zero gradients.
Computation Speed	ReLU is simply $\max(0, x)$ — extremely fast. Sigmoid requires exponentiation.
Sparsity	ReLU outputs exactly 0 for negative inputs, producing sparse activations — more efficient and interpretable.
Convergence Speed	Networks with ReLU converge $\sim 6\times$ faster than with tanh or sigmoid.
No Output Scaling Issue	ReLU output range is $[0, \infty)$. Sigmoid squashes to $(0, 1)$ causing vanishing gradients in deep nets.

Disadvantages of ReLU (and Variants)

- **Dying ReLU:** If a neuron's input is always negative, it outputs 0 permanently and never updates — the neuron 'dies'. Solutions: Leaky ReLU, ELU, Parametric ReLU.
- **Leaky ReLU:** $f(x) = \max(0.01x, x)$ — allows small negative slope to keep neurons alive.
- **ELU (Exponential Linear Unit):** Smooth for negative values, reduces bias shift.

Note: In practice, ReLU or its variants (Leaky ReLU, ELU, SELU) are the default choice for hidden layers. Sigmoid is still used at output layers for binary classification; softmax for multi-class.

Q2a Explain all the features of pooling layer. **[6 Marks]**

[REPEATED] Pooling Layer is covered in Q1a above. Key additional features to emphasize:

- Pooling is a form of non-linear down-sampling.
- It has no learnable parameters — purely a fixed mathematical operation.
- Controls spatial hierarchy of features.
- Provides translation invariance within the pooling window.
- Reduces memory footprint and computational load for subsequent layers.

- Helps make representations approximately invariant to small translations, rotations, and scale changes.

Q2b Explain Dropout Layer in Convolutional Neural Network. [6 Marks]

Dropout Layer

Dropout is a regularisation technique introduced by Srivastava et al. (2014) to prevent overfitting in neural networks. During training, each neuron (along with its connections) is randomly 'dropped out' (set to zero) with a probability p (typically 0.5 for hidden layers, 0.2 for input layers).

How Dropout Works

Training Phase:

Full Layer: [N1][N2][N3][N4][N5]

After Drop: [N1][0][N3][0][N5] ← N2, N4 dropped ($p=0.4$)

Inference Phase:

All neurons active, but weights scaled by $(1-p)$ (or equivalently, training weights are scaled by $1/(1-p)$)

Why Dropout Works

- **Prevents co-adaptation:** Neurons can no longer rely on specific other neurons, forcing them to learn more robust independent features.
- **Ensemble effect:** With N neurons, 2^N possible subnetworks are trained, and at test time their predictions are implicitly averaged.
- **Reduces overfitting:** Acts as a strong regulariser, comparable in effect to L2 weight decay.

Dropout in CNNs vs Dense Layers

- In CNN feature layers, Spatial Dropout (dropping entire feature maps) is more effective than per-neuron dropout, preserving spatial structure.
- Standard per-neuron dropout is most effective in fully-connected layers.

Note: Dropout is only active during training. At inference, all neurons are active but outputs are scaled accordingly. Modern alternatives include DropBlock (drops contiguous regions in feature maps) and Stochastic Depth (drops entire residual blocks).

Q2c Explain working of Convolution Layer with its features. [6 Marks]

Convolution Layer

The Convolution Layer is the core building block of a CNN. It learns spatial hierarchies of features by applying learnable filters (kernels) to the input feature map. The mathematical operation is a discrete cross-correlation (though conventionally called convolution in the ML community).

Working of Convolution

Input (5×5) Filter (3×3) Feature Map (3×3)

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 1 \\ 4 & 5 & 6 & 1 & 2 \\ 7 & 8 & 9 & 2 & 3 \\ 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \Sigma(\text{element-wise product}) + \text{bias}$$

Filter slides left→right, top→bottom with given stride

Key Features of a Convolution Layer

- **Local Connectivity:** Each neuron in the output connects only to a small local region (receptive field) of the input — unlike fully connected layers.
- **Parameter Sharing:** The same filter is applied across the entire input, drastically reducing parameters. A 3×3 filter has only 9 weights regardless of input size.
- **Multiple Filters:** Multiple filters are applied in parallel, each learning a different feature. If there are K filters, the output has K channels (depth).
- **Stride:** Controls how many pixels the filter moves at each step. Stride 1 → dense sampling; Stride 2 → halves dimensions.
- **Padding:** Zero-padding preserves spatial dimensions after convolution.
- **Output Size Formula:** $\text{Output_size} = \lfloor (\text{Input_size} - \text{Filter_size} + 2 \times \text{Padding}) / \text{Stride} \rfloor + 1$

Note: Depth of each filter must match the depth of the input (e.g., 3 for RGB). The number of filters is a hyperparameter and sets the depth of the output feature map.

NOV-DEC 2023

Q1a Explain stride Convolution with example.

[6 Marks]

Stride in Convolution

Stride is the step size by which the convolution filter moves across the input. It directly controls the spatial dimensions of the output feature map.

Effect of Stride

Stride Value	Effect
Stride = 1	Filter moves 1 pixel at a time. Output is nearly the same size as input (with appropriate padding). Maximum information retention.
Stride = 2	Filter jumps 2 pixels at a time. Output is approximately halved in each dimension. Acts like pooling.
Stride > 2	Aggressive down-sampling; reduces computation but may lose fine detail.

Numerical Example — Stride 2

Input (5×5):
 1 2 3 4 5
 6 7 8 9 10
 11 12 13 14 15
 16 17 18 19 20
 21 22 23 24 25

Filter (3×3): Stride = 2

Filter applied at: Output (2×2)

top-left (0,0) → O[0,0]
 top-right(0,2) → O[0,1]
 bot-left (2,0) → O[1,0] Output size = $\lfloor (5-3)/2 \rfloor + 1 = 2$ bot-right(2,2) → O[1,1]

Advantages of Strided Convolution

- Replaces the need for a separate pooling layer, reducing depth of the network.
- The network can learn an optimal down-sampling rather than using a fixed pooling operation.
- Reduces computational cost and memory usage.

Note: Modern architectures (e.g., ResNet, DCGAN) prefer strided convolution over max-pooling for down-sampling, as it is learnable and thus more flexible.

Q1b Explain Padding and its types. [6 Marks]

Padding in CNN

Padding refers to adding extra values (typically zeros) around the border of an input before applying a convolution filter. Without padding, each convolution reduces the spatial dimensions of the feature map, and pixels at the edges of the image are processed far fewer times than central pixels, losing information.

Types of Padding

- **Valid Padding (No Padding):** No extra pixels are added. The filter only slides where it fully fits inside the input. Output size = $\lfloor (W - F)/S \rfloor + 1$, where W = input size, F = filter size, S = stride. Information at borders is lost.
- **Same Padding (Zero Padding):** Zeros are added around the input so the output has the same spatial size as the input (when stride = 1). Amount of padding $P = (F - 1) / 2$. Preserves spatial dimensions and edge information.
- **Full Padding:** Enough zeros are added so the filter can slide over every position where it has at least one input pixel overlapping. Output is larger than input. Rarely used in classification networks.
- **Reflect Padding:** Pads with mirrored input values instead of zeros. Reduces border artifacts for image generation tasks.

Visual Comparison — Same vs Valid Padding

Input: 4×4 Filter: 3×3 Stride: 1

Valid Padding → Output: 2×2 (shrinks) Same Padding → Output: 4×4 (preserved, P=1 zero-padding on each side)

Note: Same Padding is used extensively in ResNets and any network where consistent spatial dimensions between layers are important (e.g., skip connections require matching sizes).

Q1c Explain Local Response Normalization and need of it. [6 Marks]

Local Response Normalization (LRN)

Local Response Normalization is a normalization technique introduced in AlexNet (2012) that promotes competition among neurons activated by neighboring feature maps. It mimics the lateral inhibition observed in real neurons in the brain.

Mathematical Formula

$$b^i_{(x,y)} = a^i_{(x,y)} / (k + \alpha \cdot \sum (a^j_{(x,y)})^2)^{\beta}$$

where the sum is over j from $\max(0, i-n/2)$ to $\min(N-1, i+n/2)$

$a^i_{(x,y)}$ = activation at position (x,y) in feature map i

N = total number of feature maps

n = number of adjacent feature maps to normalize over k, α, β = hyperparameters ($k=2, n=5, \alpha=1e-4, \beta=0.75$ in AlexNet)

Need / Purpose of LRN

- Normalizes activations across adjacent feature maps at the same spatial position.
- Encourages feature maps with large activations to suppress neighboring maps — creates lateral inhibition.
- Provides a form of local contrast normalization, making highly activated neurons more pronounced relative to neighbors.
- Helps the network generalize better, especially for unnormalized (unbounded) activations like ReLU.

Limitations

- LRN has largely been replaced by Batch Normalization (BN), which is more principled, faster to train, and consistently more effective.
- LRN normalizes across feature maps (channel dimension), while BN normalizes across the batch and spatial dimensions.

Note: Batch Normalization (Ioffe & Szegedy, 2015) made LRN obsolete in practice. BN normalizes layer inputs using batch statistics (mean and variance) and introduces learnable scale (γ) and shift (β) parameters, allowing much faster training with higher learning rates.

Q2a Explain ReLU Layer and its advantages. [6 Marks]

[REPEATED] Covered comprehensively in Q1c of May-June 2023. Key advantages summary:

- Avoids vanishing gradient problem — gradient is either 0 or 1 for negative/positive inputs.
- Computationally trivial: $f(x) = \max(0, x)$ — no exponentiation.
- Produces sparse activations — efficient and more biologically plausible.
- Networks converge significantly faster compared to sigmoid/tanh.

Q2b Explain Pooling layers and its types with examples. [6 Marks]

[REPEATED] Pooling Layer covered in detail in Q1a of May-June 2023. See that section for full explanation and types.

Q2c What are the applications of Convolution with examples? [6 Marks]

Applications of Convolution in Deep Learning

- **Image Classification:** CNNs classify images into categories. Example: ResNet classifying 1000 ImageNet classes; VGG achieving human-level performance.
- **Object Detection:** Detect and localise multiple objects in a scene. Example: YOLO (You Only Look Once) for real-time detection; Faster-RCNN using region proposal networks.
- **Semantic Segmentation:** Assign a class label to every pixel. Example: FCN (Fully Convolutional Network) for scene parsing; DeepLab for medical imaging.
- **Face Recognition:** Identify individuals from facial images. Example: FaceNet, DeepFace (Facebook).
- **Medical Image Analysis:** Detect tumours, segment organs, analyse X-rays. Example: U-Net architecture for biomedical image segmentation.
- **Natural Language Processing:** 1-D convolutions over word embeddings for text classification. Example: TextCNN for sentiment analysis.
- **Autonomous Driving:** Road segmentation, pedestrian detection, lane detection. Example: CNNs in Tesla's Autopilot, Waymo.
- **Image Generation:** Transposed convolutions (deconvolution) in GANs to upsample and generate images. Example: DCGAN.

Note: The key insight enabling all these applications is that convolution exploits local spatial structure and parameter sharing, making it data-efficient and powerful for any data with grid-like topology (images, audio spectrograms, video frames, time-series).

MAY-JUNE 2024

Q1a Explain CNN architecture with its application. **[6 Marks]**

[REPEATED] CNN Architecture is covered comprehensively in Q1b of May-June 2023. Refer to that section for the detailed layered diagram and explanations. Applications are also listed there.

Q1b What is Padding? Enlist and explain types of padding. **[6 Marks]**

[REPEATED] Padding is covered in full in Q1b of Nov-Dec 2023. Refer to that section for all types (Valid, Same, Full, Reflect) with formulas.

Q1c Explain Dropout Layer in Convolutional Neural Network. **[6 Marks]**

[REPEATED] Dropout Layer is explained in detail in Q2b of May-June 2023. Refer to that section.

Q2a Define ReLU. Explain disadvantages of ReLU. **[6 Marks]**

ReLU — Definition

ReLU (Rectified Linear Unit) is a piecewise linear activation function defined as $f(x) = \max(0, x)$. It outputs the input directly if positive, otherwise outputs zero. It is the most widely used activation function in hidden layers of deep neural networks due to its simplicity and effectiveness.

Disadvantages of ReLU

- **Dying ReLU Problem:** If a neuron's weights update such that it always receives negative input, it permanently outputs 0 and its gradients are also 0 — the neuron 'dies' and stops learning. This is especially problematic with high learning rates.
- **Not Zero-Centred:** ReLU outputs are always ≥ 0 . When inputs to the next layer are all positive, gradients are always positive or always negative for a given weight, causing zig-zag updates (inefficient convergence).
- **Unbounded Output:** ReLU has no upper bound, which can cause very large activations if not managed (e.g., with batch normalisation).
- **Not Differentiable at 0:** The derivative is undefined at $x = 0$ (though in practice, it is set to 0 or 0.5 and does not cause issues).
- **No Negative Information:** Negative inputs are completely discarded, losing potentially useful information.

Variants that Address These Issues

Variant	How It Fixes the Issue
Leaky ReLU	$f(x) = \max(0.01x, x)$ — allows small negative gradient, preventing dying neurons.
Parametric ReLU (PReLU)	Learns the slope for negative part as a parameter.
ELU	Exponential for negatives; smooth, zero-centred-like outputs.
SELU	Self-normalising — maintains mean=0 and std=1 across layers.
GELU	Gaussian-weighted; used in transformers (BERT, GPT).

Note: Despite its disadvantages, ReLU remains the default activation in CNNs due to its simplicity and empirical effectiveness. The dying ReLU problem is mitigated by using appropriate weight initialisation (He initialisation), lower learning rates, and batch normalisation.

Q2b What is Strides in CNN? Explain in brief. **[6 Marks]** **[REPEATED]** Stride is covered in detail in Q1a of Nov-Dec 2023. Refer to that section for full explanation with numerical example and comparison table.

Q2c Explain Pooling Layer with its different types. **[6 Marks]** **[REPEATED]** Pooling Layer is covered exhaustively in Q1a of May-June 2023. Refer to that section.