

DL – Unit 1 (Foundations of Deep Learning) – IN-SEM PYQ Answers

Q1. What is Deep Learning? Enlist and Explain Challenges of Deep Learning? [5]

Deep Learning is a subfield of machine learning that uses artificial neural networks with **multiple layers** (deep architectures) to automatically learn **hierarchical representations** of data for tasks such as **classification, regression, and generation**. It models complex patterns by transforming raw input through successive non-linear layers to extract features at increasing levels of abstraction.

1. **Overfitting and Underfitting:** Balancing model complexity to ensure it generalizes well to new data is challenging. Overfitting occurs when a model is too complex and captures noise in the training data. Underfitting happens when a model is too simple and fails to capture the underlying patterns.
2. **Data Quality and Quantity:** Deep learning models require large, high-quality datasets for training. Insufficient or poor-quality data can lead to inaccurate predictions and model failures. Acquiring and annotating large datasets is often time-consuming and expensive.
3. **Computational Resources:** Training deep learning models demands significant computational power and resources. This can be expensive and inaccessible for many organizations. High-performance hardware like GPUs and TPUs are often necessary to handle the intensive computations.
4. **Interpretability:** Deep learning models often function as "black boxes," making it difficult to understand how they make decisions. This lack of transparency can be problematic, especially in critical applications. Understanding the decision-making process is crucial for trust and accountability.
5. **Hyperparameter Tuning:** Finding the optimal settings for a model's hyperparameters requires expertise. This process can be time-consuming and computationally intensive. Hyperparameters significantly impact the model's performance, and tuning them effectively is essential for achieving high accuracy.
6. **Adversarial Attacks:** Deep learning models are susceptible to adversarial attacks, where subtle perturbations to input data can cause misclassification. Robustness against such attacks remains a significant concern in safety-critical applications.

Q2. Explain pros and cons of using Deep Learning.[5]

1. Pros (Advantages)

- **Automatic Feature Learning:** Deep learning models can **automatically extract relevant features** from raw data, eliminating the need for manual feature engineering.
- **Handling Large & Complex Data:** They excel at processing **large and intricate datasets**, including unstructured data like images, text, and audio.
- **High Performance:** Deep networks often achieve **high accuracy and superior performance** on complex tasks such as image and speech recognition, natural language processing, and pattern discovery.
- **Non-Linear Relationship Modeling:** They can learn **complex non-linear relationships** that traditional models might miss.
- **Flexible Applications:** Deep learning is adaptable across diverse domains (vision, speech, NLP, healthcare, autonomous systems) due to its hierarchical learning capabilities.

2. Cons (Disadvantages)

- **Large Data Requirement:** Deep learning models generally require **very large amounts of training data** to perform well, which can be costly and time-consuming to collect.
- **High Computational Cost:** They need **significant computational resources** (e.g., GPUs/TPUs) for training and inference, increasing cost and infrastructure needs.
- **Overfitting Risk:** Due to high model capacity, deep networks are **prone to overfitting**, especially on limited or noisy datasets.
- **Interpretability Issues:** Deep learning models often behave as **“black boxes,”** making it difficult to interpret how decisions are made.
- **Complex Design and Tuning:** They require **expertise in architecture design, hyperparameter tuning, and optimization** which adds to development complexity.

Q3. Enlist and Explain any four real life applications of Deep Learning. [5]

1. Computer Vision – Image Classification & Object Detection

- Deep learning models like **Convolutional Neural Networks (CNNs)** automatically learn hierarchical visual features from raw pixels.
- Used in **medical imaging** (e.g., detecting tumors in X-rays, MRI scans), **autonomous vehicles** (detecting pedestrians, traffic signs), and **security systems** (face recognition).
- **Key impact:** High accuracy on complex visual tasks without manual feature extraction.

2. Natural Language Processing (NLP) – Machine Translation

- Deep architectures such as **Transformer models** (e.g., **BERT, GPT**) learn contextual representations of text.
- Enables systems like **Google Translate** to convert text between languages by capturing semantic and syntactic patterns.
- **Key impact:** Produces translations that approach human quality for major language pairs.

3. Speech Recognition and Voice Assistants

- Recurrent Neural Networks (RNNs), **Long Short-Term Memory (LSTM)**, and more recently **Transformer-based models** convert spoken language into text.
- Used in applications like **virtual assistants** (e.g., Siri, Alexa), dictation tools, and voice-controlled systems.
- **Key impact:** Significantly reduced error rates in real-world noisy environments.

4. Recommendation Systems

- Deep learning models analyze large user interaction data to predict preferences.
- Used by platforms like **Netflix, YouTube** and **Amazon** to recommend movies, videos, and products.
- Models learn **latent user and item features** to personalize suggestions.
- **Key impact:** Improved engagement and user satisfaction.

Q4. Enlist and Explain any five popular industrial tools used for Deep Learning.[5]

1. TensorFlow

- **TensorFlow** is an open-source deep learning framework developed by Google Brain that provides a flexible and scalable platform for building, training, validating, and deploying complex neural networks.
- Supports multiple languages such as Python, C++, and JavaScript, and is widely used for production-level models in industry for applications like computer vision, NLP, and time-series prediction.

2. PyTorch

- **PyTorch** is an open-source deep learning library developed by Meta (Facebook) that emphasizes dynamic computation graphs and ease of experimentation.
- It is widely used in both research and industry for natural language processing, computer vision, and reinforcement learning due to its intuitive Python interface and flexibility.

3. Keras

- **Keras** is a high-level neural network API designed for fast experimentation and simplicity, originally independent and now integrated into TensorFlow.
- It provides modular and user-friendly interfaces to build and train deep learning models, making it popular in industry for rapid prototyping and development.

4. Caffe (Convolutional Architecture for Fast Feature Embedding):

- An open-source deep learning framework developed by Berkeley Vision and Learning Center (**BVLC**) with a focus on speed, modularity, and expressiveness for building, training, and deploying neural networks efficiently.
- It supports **CNN**, **RCNN**, **LSTM** and fully-connected networks, with both CPU and GPU acceleration, making it suitable for large-scale tasks like image classification, object detection and multimedia analysis in industrial applications.

5. Shogun

- Shogun is a free and open-source machine learning software library written in C++ that provides a broad range of algorithms including Support Vector Machines (**SVMs**), **dimensionality reduction**, **clustering**, and kernel methods, with interfaces for languages like Python, Octave, R, Java and others.
- Although not primarily designed as a deep learning framework, it includes support for neural network components and is used in research and production for scalable kernel-based and statistical learning tasks across various domains.

Q5. Explain how Deep Learning works in three figures with one example. Also Explain Common architectural principles of Deep Network. [5]

Q6. How Deep Learning works in three figures explained with an example? Also explain common architectural principles of Deep Network? [5]

3 figures:

Figure 1: Neural Network Architecture (Layers)

Input Layer Hidden Layers Output Layer
 $[x_1, x_2, x_3] \rightarrow [H_1 \rightarrow H_2 \rightarrow \dots \rightarrow H_n] \rightarrow [\text{Prediction/Classification}]$

- **Input Layer:** Receives **raw data** (e.g., pixel values of an image, audio waveform, text embeddings).
- **Hidden Layers:** Series of neurons where **feature transformation and extraction** occurs progressively.
 - Early layers learn **simple patterns**.
 - Deeper layers learn **complex abstractions** by combining earlier features.

- **Output Layer:** Produces **final prediction** (e.g., class label, regression value).
- **Key Idea:** Each layer transforms input into a more **meaningful representation** for solving the task.

Figure 2: Forward Propagation & Feature Extraction

Input → Layer 1 → Layer 2 → Layer 3 → Output

↓ ↓ ↓ ↓ ↓

Raw → Edges → Shapes → Objects → Prediction

- **Forward Propagation:** Data flows **forward** through the network from input to output.
- **Weight Multiplication & Activation:** Each neuron applies **weights** and an **activation function** to transform input signals.
- **Hierarchical Feature Extraction:**
 - Layer 1 detects **basic features** (e.g., edges).
 - Layer 2 detects **intermediate features** (e.g., shapes).
 - Deeper layers detect **complex patterns** (e.g., objects, context).
- The network's ability to **automatically extract features** eliminates manual feature design.

Figure 3: Backpropagation & Training (Weight Adjustment)

Forward Output

Input → ... → Prediction

|

Loss/Error

↓

Backpropagation

↓

Adjust Weights (↓Error)

- **Prediction vs True Value:** After forward pass, the network compares its output with the **true label** to calculate **error (loss)**.
- **Backpropagation:** Error is sent **backwards** through the network to determine how each weight contributed to the error.
- **Gradient Descent:** Weights are adjusted using gradient descent to **minimize error** over training iterations.
- Continuous repetition **improves accuracy** as weights converge to values that produce correct outputs.

One Example (Image Classification)

Scenario: Classifying images of handwritten digits (0–9).

- **Input Layer:** Pixel values of a grayscale image (28×28).
- **Hidden Layers:**
 - Early layers detect **edges and strokes**.
 - Mid layers detect **loops and line intersections**.
 - Deep layers detect **digit shapes** (0–9).
- **Forward Propagation:** The image flows through layers to produce a **predicted digit**.

- **Backpropagation:** If the predicted digit differs from the true label, **error is calculated** and **weights adjusted** so the network learns correct representations.

Common architectural principles of Deep Network:

1. Layered and Hierarchical Structure

- Deep networks are composed of multiple layers (input, hidden, output) stacked hierarchically.
- Depth (number of hidden layers) enables **progressive feature learning**, from simple to complex patterns.
- Increasing depth increases representational capacity but also challenges like optimization and overfitting.

2. Non-Linearity and Activation Functions

- Each neuron applies a **non-linear activation function** (e.g., ReLU, sigmoid) to introduce non-linearity.
- Non-linear transformations allow networks to model **complex non-linear relationships** in data beyond linear separability.

3. Connectivity Patterns

- Different deep architectures (e.g., fully connected, convolutional, recurrent) define how neurons connect.
- **Fully connected layers** connect every neuron to all inputs, while **CNNs use local receptive fields** and **weight sharing** for spatial data.
- These patterns shape how features are aggregated and processed.

4. Feature Abstraction and Representation Learning

- Layers successively transform raw input into **higher-level representations**.
- Early layers capture basic patterns; deeper layers capture abstract concepts.
- Hierarchical representation learning is fundamental to deep learning's success.

5. Modular Structural Motifs (Shortcuts and Normalization)

- Deep architectures often incorporate **residual/skip connections** and normalization modules to improve training stability and convergence.
- These motifs help **mitigate vanishing gradients** and facilitate deeper models.

Q7. Give Co-relation between Artificial Intelligence, Machine Learning, and Deep Learning. [5]

Q8. Compare Deep Learning with Machine Learning. [5]

Aspect	Artificial Intelligence (AI)	Machine Learning (ML)	Deep Learning (DL)
Definition	Broadest field of	Subset of AI that enables	Subset of ML that uses

	computing to create systems that mimic human intelligence to perform tasks.	systems to learn from data and improve automatically.	multi-layer neural networks to learn complex patterns.
Approach	Uses rules, logic, and search to mimic human reasoning.	Uses statistical models and algorithms to learn patterns from data.	Uses multi-layer neural networks for representation learning.
Goal	Enable machines to reason, plan, and act intelligently.	Learn patterns from data for prediction/decision making.	Automatic hierarchical feature learning from data.
Data Requirement	Can use rules or data; not always data-centric.	Requires moderate datasets .	Requires very large datasets for best performance.
Feature Engineering	Often needs manual features and rules .	May require manual feature engineering .	Automatic feature extraction ; minimal manual intervention.
Model Complexity	Can be simple (rule-based) or complex.	Moderate complexity models (SVM, trees).	High complexity (deep neural networks).
Compute Needs	Varies with approach; can be low.	Moderate computing.	High computational resources needed (GPUs/TPUs).
Performance on Tasks	Limited by rule design.	Better with pattern learning.	Superior performance on complex pattern and high-dimensional data.
Examples	Expert systems, autonomous decision systems.	Regression, classification, clustering.	CNN for vision, RNN/Transformers for NLP.

Q9. Compare Supervised and Unsupervised Learning. [5]

Aspect	Supervised	Unsupervised
Input Data	Uses labeled data (input features + corresponding outputs)	Uses unlabeled data (only input features, no outputs).
Goal	Predicts outcomes or classifies data based on known labels.	Discovers hidden patterns, structures, or groupings in data.
Computational Complexity	Less complex, as the model learns from labeled data with clear	More complex, as the model must find patterns without any guidance.

	guidance.	
Types	Two types : Classification (for discrete outputs) or regression (for continuous outputs).	Clustering and association
Testing the Model	Model can be tested and evaluated using labeled test data.	Cannot be tested in the traditional sense, as there are no labels.

Q10. Explain various limitations of Machine Learning.[5]

- 1. Data Dependency:** The performance of ML models heavily depends on the quality and quantity of training data. Poor-quality or biased data can lead to inaccurate predictions or unfair outcomes. For example, biased datasets in hiring algorithms can perpetuate discrimination.
- 2. High Computational Costs:** Developing and deploying machine learning models requires significant investment in infrastructure, computational resources, and skilled professionals. Small businesses may find these costs prohibitive.
- 3. Complexity and Interpretability:** Many machine learning models especially deep learning systems, operate as "black boxes." Their decision-making processes are difficult to interpret or explain, raising ethical concerns in critical fields like healthcare or finance.
- 4. Risk of Bias:** If the training data contains biases, the ML model may perpetuate or amplify these biases, leading to unfair or unethical outcomes, especially in sensitive applications like hiring or lending.
- 5. Job Displacement:** Automation through machine learning can lead to job losses in certain sectors. Roles involving repetitive tasks, such as data entry or assembly line work, are particularly vulnerable. While ML creates new opportunities, reskilling the workforce remains a significant challenge.

Q11. Write Short Note on: Under and Over fitting regularization.[5]

Q12. Short Note on LSTM networks & GRU networks. [5]

Long Short-Term Memory (LSTM):

- Specialized type of **Recurrent Neural Network (RNN)** designed to **learn and retain long-term dependencies** in sequential data.
- Unlike traditional RNNs, which struggle with remembering information over long sequences due to the **vanishing/exploding gradient problem**, LSTMs introduce a **memory-cell structure with gating mechanisms** that allow information to be **selectively remembered or forgotten** over time.

LSTM Networks: Key Characteristics

- Memory Cell and Gates:** An LSTM network contains a **memory cell** and three key gates: **input gate**, **forget gate**, and **output gate** — that regulate the **flow of information** into, within, and out of the cell.
 - Input Gate:** Decides what **new information** to store in the memory cell.
 - Forget Gate:** Determines which **past information** should be discarded.

- **Output Gate:** Controls which information from the cell is **passed forward** to the next state or output.
This design allows LSTM networks to **retain relevant context** over long sequences while filtering out irrelevant details.
- **Addressing Long-Term Dependencies:**
 - Traditional RNNs struggle with **long-range dependencies** because gradients can shrink or explode during training, preventing the model from effectively learning relationships that span many time steps.
 - LSTM networks mitigate this problem by **preserving gradients** through the memory cell and gating mechanisms, enabling the network to **capture long-term patterns** in sequential data.
- **Sequential Data Handling:**
 - LSTM networks are especially effective for **sequence prediction and processing tasks** such as **language modeling, speech recognition, time-series forecasting, and machine translation**, because they can remember important features from earlier steps in the sequence and use that information for current predictions.
- **Example of LSTM Application: speech recognition:** processes audio signals over time and retains information about earlier parts of the speech to improve accuracy in recognizing words or phonemes. In this task, LSTM's ability to remember long-term dependencies helps it understand context and continuity in spoken language, outperforming traditional RNNs.

Gated Recurrent Unit (GRU):

- **Recurrent Neural Network (RNN) architecture** designed to process **sequential data** by capturing dependencies over time steps.
- It was introduced to overcome limitations of standard RNNs such as the **vanishing gradient problem**, enabling the model to **retain relevant information across longer sequences**.
- Compared to traditional RNNs, GRUs use a **gating mechanism** that regulates the flow of information, making them **more effective for sequence modeling** tasks like language processing, time-series prediction, and speech recognition.

GRU Networks: Key Characteristics

- **Gating Mechanism:** A GRU network primarily uses **two gates**, the **update gate** and the **reset gate**: which control how much of the previous state should be retained or forgotten at each time step.
 - **Update Gate:** Decides how much of the past information should be passed forward to the current hidden state.
 - **Reset Gate:** Determines how much of the previous hidden state should be **forgotten** when calculating the candidate new state.
- **Simplified Structure:** Unlike Long Short-Term Memory (LSTM) networks, GRUs do **not have a separate memory cell**; instead, they integrate memory and hidden states into a single vector, resulting in **fewer parameters** and **simpler architecture**.
- **Mitigating Vanishing Gradients:** The gating mechanisms in GRU help preserve gradients during training, enabling the network to **capture long-term dependencies** in sequential patterns more effectively than simple RNNs.
- **Computational Efficiency:** With **fewer gates and parameters** than LSTM, GRUs tend to **train faster and be more computationally efficient**, making them suitable for applications with limited resources.

- **Example Application of GRU Network: natural language processing (NLP)** tasks such as **language modeling** or **text generation**. In this example, the GRU processes a sequence of words one step at a time, using its gates to decide which information from earlier words to retain for predicting the next word in the sequence.

Q13. Write a Short note on Autoencoders and Restricted Boltzmann Machines. [5]

Autoencoders:

- An **autoencoder** is a type of **unsupervised artificial neural network** that learns **efficient encodings of unlabeled data** by compressing input into a lower-dimensional representation and then reconstructing it back to the original form.
- It consists of two main parts:
 - **Encoder:** Transforms the input into a **compressed latent representation**.
 - **Decoder:** Reconstructs the original input from the **latent representation**.
- The network is trained to **minimize reconstruction error**, learning features that capture the essential structure of the data without supervision.
- Autoencoders are widely used for **dimensionality reduction, feature learning, denoising**, and as initial pre-training for deeper networks.

Restricted Boltzmann Machines (RBMs)

- A **Restricted Boltzmann Machine (RBM)** is a **generative stochastic neural network** used in unsupervised learning to **model the probability distribution** of input data.
- It has a **visible layer** (input units) and a **hidden layer** (feature detectors) with **no connections within the same layer** (only between layers), forming a **bipartite structure** that simplifies training.
- RBMs learn latent features by adjusting weights to approximate the data distribution, often using **Contrastive Divergence** for efficient learning.
- They are used for **feature learning, dimensionality reduction, collaborative filtering**, and act as building blocks of **deeper architectures** like Deep Belief Networks (DBNs).

Q14. Write Short note on: Bias, Variance and Tradeoff. [5]

Q15. Explain Bias-Variance Tradeoff. [5]

Bias: difference between the prediction of the values by the ML model and the correct value

- A high bias model gives large error on both training and testing data.
- It is recommended that an algorithm should always be low-biased to avoid underfitting.
- With high bias, the predicted data fits in a straight line format, not capturing the true pattern of the dataset.
- Such a situation where the model cannot fit the data accurately is known as Underfitting of data.
- This happens when the hypothesis is too simple or linear in nature.

Variance: is the variability of model prediction for a given data point, indicating the spread of the data.

- A model with high variance fits the training data too closely with a very complex fit.
- As a result, such models perform very well on training data but have high error rates on unseen test data.
- When a model has high variance, it is said to be Overfitting of data.

- Overfitting occurs when the model fits the training set accurately via a complex curve but does not generalize well to new data.
- While training a data model, variance should be kept low.

Bias–Variance Tradeoff:

- If an algorithm is too simple (e.g., linear hypothesis), then it may be in a state of high bias and low variance, making it error-prone.
- If an algorithm fits too complex (e.g., high degree hypothesis), it may have high variance and low bias, performing poorly on new entries.
- There is a condition between these extremes known as the Bias–Variance Trade-off.
- This trade-off exists because an algorithm cannot be overly complex and overly simple at the same time.
- We try to optimize the total error of the model by balancing bias and variance.
- The total error can be expressed as:
Total Error = Bias² + Variance + Irreducible Error

Underfitting: happens when the model fails to learn important patterns. It performs poorly on both training and testing data. Underfitting happens due to:

- Model is too simple
- Very high regularization
- Features are weak or missing
- Not enough training
- High bias = model makes strong assumptions, Ignores patterns, Learns an overly simple representation & Variance is low because the model gives similar outputs even if the data changes

Techniques to Reduce Underfitting:

- Use a more complex model
- Add new features and perform feature engineering
- Reduce regularization
- Train for more epochs
- Scale features properly

Overfitting happens when the model learns too much from the training data, including noise and outliers. It performs very well on training data but poorly on test data. Overfitting happens due to:

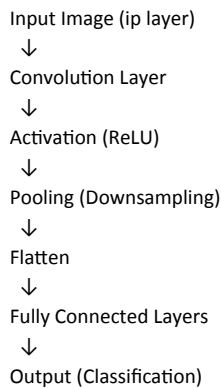
- Model too complex
- Too many features
- Very little data
- No regularization
- High variance = model reacts too strongly to training data, Learns noise as patterns & Low bias because the model is extremely flexible

Techniques to Reduce Overfitting:

- Collect more training data
- Reduce model complexity
- Use regularization (L1/L2)
- Apply dropout (for neural networks)
- Use early stopping
- Clean noisy data

Q16. Explain Convolution Neural Networks with diagrams. [5]

A **Convolutional Neural Network (CNN)** is a specialized type of **feed-forward deep learning network** designed to process data with a **grid-like topology**, such as images and videos. CNNs automatically **extract hierarchical features** from raw input through convolution and pooling operations, making them highly effective for **image classification, object detection and visual pattern recognition**.

Basic CNN Architecture

- **Input Layer:** Accepts grid-structured input (e.g., an image).
- **Convolution Layer:** Applies **filters/kernels** that slide over the input to extract **local features** like edges or textures.
- **Activation (ReLU):** Adds **non-linearity** to enable complex pattern learning.
- **Pooling Layer:** Reduces spatial dimensions, **retaining important features** and lowering computation.
- **Flatten:** Converts multi-dimensional feature maps into a vector.
- **Fully Connected Layer:** Performs final **classification or regression**.

Convolution Operation

Image Patch (3×3) × Filter (3×3) → Feature Map

[pixel values] [kernel weights]

- Convolution uses a **kernel/filter** that slides across the input image.
- At each position it computes a **dot product** between filter weights and image values to produce a **feature map** that highlights the presence of patterns.
- Multiple filters learn different features such as lines, curves and textures

Pooling Operation

Feature Map

↓ Max Pooling (2×2)

Reduced Feature Map

- **Pooling** (e.g., **max pooling**) reduces the **spatial size** of feature maps.
- It takes the maximum (or average) value from each small region of the input.
- This **reduces parameters** and helps the network learn **invariant features** regardless of location.