NAME _ PRADEEP KESHARI

ROLL.NO _ 2202161520073

ASSIGNMENT _ 3

Que 1. Explain and compare PCA and LDA for linear dimensionality reduction.

ANS _ Explanation of PCA (Principal Component Analysis)

PCA is a linear, unsupervised dimensionality reduction technique.

How PCA works

- It does not use class labels.
- It finds new axes (called principal components) such that:
  - 1st principal component captures the maximum variance in the data
  - 2nd component captures the next maximum variance
- PCA keeps only those components that carry most information.
- It helps in data compression, noise removal, and speeding up ML models.

Explanation of LDA (Linear Discriminant Analysis)

LDA is a linear, supervised dimensionality reduction technique.

How LDA works

- It uses class labels.
- It tries to find axes (discriminant directions) that:

- Maximize distance between class means (between-class scatter)
- Minimize spread within each class (within-class scatter)
- So LDA finds directions that best separate different classes.
- It tries to find axes (discriminant directions) that:
    - Maximize distance between class means (between-class scatter)
    - Minimize spread within each class (within-class scatter)
- So LDA finds directions that best separate different classes.
- # Comparison: PCA vs LDA

Property	PCA	LDA
Type	Unsupervised	Supervised
Uses class labels?	❌ No	✅ Yes
Objective	Maximize variance	Maximize class separation
Basis	Covariance of data	Scatter of classes
Reduces noise?	Yes	Not main purpose
When to use?	No labels / feature compression	Classification problems
Max no. of components	≤ Number of features	≤ Number of classes – 1

Que 2. What is manifold learning? Describe any two manifold learning algorithms.

ANS _ Manifold Learning

Manifold learning is a non-linear dimensionality reduction technique based on the idea that high-dimensional data actually lies on a low-dimensional manifold (a smooth, curved surface) embedded within the high-dimensional space.

The goal is to discover this lower-dimensional structure while preserving important geometric relationships among data points.

Examples: images, speech signals, handwritten digits, and sensor data — all have high dimensions but follow an underlying low-dimensional manifold.

Two Manifold Learning Algorithms

1. Isomap (Isometric Mapping)

Concept

Isomap extends classical MDS (Multidimensional Scaling) to non-linear data.
Instead of preserving simple Euclidean distances (straight-line distances), Isomap preserves geodesic distances (distances measured along the curved manifold).

Steps

1. Construct a neighborhood graph
    - Connect each point to its *k* nearest neighbors.
2. Compute geodesic distances
    - Use Dijkstra's or Floyd–Warshall algorithm to calculate shortest paths on the graph.
3. Apply MDS
    - Use the geodesic distance matrix to compute a low-dimensional embedding.

- Captures global non-linear structure.

- Works well on curved datasets (e.g., the “Swiss roll”).

---

2. Locally Linear Embedding (LLE)

Concept

LLE assumes that each data point and its nearest neighbors lie on a locally linear patch of the manifold.
Each point can be represented as a linear combination of its neighbors.
LLE finds a low-dimensional embedding that preserves these local linear relationships.

Steps

1. Find k-nearest neighbors for every data point.
2. Compute reconstruction weights
    - Represent each point as a weighted sum of its neighbors.
3. Compute the embedding
    - Find low-dimensional coordinates that best preserve the same reconstruction weights.

Que 3. Explain different weight initialization methods used while training ConvNets**.**

ANS _ Weight Initialization Methods in ConvNets (Short Answer)

1. Random Initialization

- Small random values to break symmetry.
- Simple but may still cause vanishing/exploding gradients.

2. Xavier (Glorot) Initialization
    - Keeps variance of activations stable for sigmoid/tanh.
    - Useful for shallow and some deep ConvNets.
3. He Initialization (Kaiming)
    - Designed for ReLU activations.
    - Prevents vanishing gradients; most commonly used in ConvNets.
4. Orthogonal Initialization
    - Uses orthogonal matrices to maintain gradient flow.
    - Helps stabilize training in deep networks.
5. Lecun Initialization
    - Best for SELU activations.
    - Maintains self-normalization.
6. Pretrained Initialization
    - Uses weights from models like VGG/ResNet.
    - Speeds up training, especially with small datasets.

Que 4. Discuss the importance of batch normalization and hyperparameter optimization in ConvNet training.

ANS _ Importance of Batch Normalization and Hyperparameter Optimization in ConvNet Training

1. Batch Normalization (BN)

Batch Normalization is a technique that normalizes activations of each layer during training.

Importance

- Reduces Internal Covariate Shift
  Keeps input distributions stable across layers, making training smoother.
- Faster Training
  Allows the network to use higher learning rates without divergence.
- Improves Gradient Flow
  Reduces vanishing/exploding gradients.
- Acts as Regularization
  Reduces overfitting by adding noise through mini-batch statistics.
- Enables Deeper Networks
  Helps very deep ConvNets like ResNet train effectively.

---

2. Hyperparameter Optimization

Hyperparameters include learning rate, batch size, number of layers, filters, kernel size, dropout rate, etc.

Importance

- Improves Model Accuracy
  Proper selection of hyperparameters can significantly boost ConvNet performance.
- Stabilizes Training
  Right learning rate and momentum prevent instability or slow convergence.
- Controls Overfitting
  Using optimized dropout, weight decay, and batch size improves generalization.
- Enhances Training Efficiency
  Finding optimal hyperparameters reduces training time and computational cost.
- Essential for Model Tuning
  Techniques like Grid Search, Random Search, Bayesian optimization help identify the best configuration.

Que 5. Explain how autoencoders perform dimensionality reduction in neural networks.

ANS _ How Autoencoders Perform Dimensionality Reduction

An autoencoder is a neural network designed to learn a compressed (lower-dimensional) representation of input data.

It has two main parts:

1. **Encoder** – Compresses the input into a smaller, latent vector (bottleneck).

2. **Decoder** – Reconstructs the original input from the compressed representation.

How Dimensionality Reduction Happens

1. **Bottleneck Layer Forces Compression**

- The autoencoder contains a hidden layer with fewer neurons than the input layer.
- When the encoder maps high-dimensional data → low-dimensional latent space, it is forced to retain only the most important features, discarding noise and redundancies.

2**. Learns Non-Linear Transformations**

- Unlike PCA (linear), autoencoders learn non-linear mappings using activation functions.
- This allows them to capture complex structures in data.

3. **Reconstruction Training**

- The network is trained to minimize reconstruction error between input and output.
- To reconstruct accurately from fewer dimensions, it must learn meaningful patterns, not memorize data.

4. **Latent Vector = Reduced Dimension**

- After training, the latent representation (code) becomes the reduced-dimensional form of the data.
- This can be used for visualization, clustering, and further processing.