🧠 Problem 05: Feedforward Neural Network for MNIST Digit Classification

Aspect	Description
Problem	Classify handwritten digits using a simple feedforward neural network.
Input	28×28 grayscale images (flattened to 784-dimensional vectors) from MNIST dataset.
Model	Fully connected feedforward neural network.
Layers	Input layer (784) → Hidden layer(s) → Output layer (10 classes).
Activation Function	ReLU in hidden layers, Softmax in output layer.
Loss Function	Cross-entropy loss.
Optimizer	Stochastic Gradient Descent (SGD) or Adam.
Training Method	Supervised learning via backpropagation and mini-batch gradient descent.
Output	Probability scores for each digit (0–9).
Evaluation	Accuracy, confusion matrix, and loss/accuracy learning curves.

Description:
A simple feedforward neural network (FNN) processes flattened image vectors and learns class-wise distinctions using supervised learning.
Cross-entropy loss is defined as:

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability.

📷 Problem 06: Convolutional Neural Network (CNN) for Image Classification

Aspect	Description
Problem	Classify images using a CNN architecture.
Input	RGB or grayscale images.
Model	Convolutional Neural Network with pooling and dense layers.
Layers	Conv → ReLU → Pool → Conv → Pool → Flatten → Dense → Softmax.
Activation Function	ReLU for hidden layers, Softmax for output.
Loss Function	Cross-entropy loss.
Optimizer	Adam or SGD.
Training Method	Supervised training using backpropagation and mini-batches.
Output	Class probabilities.
Evaluation	Accuracy, loss curves, and class-wise precision-recall.

Description:
CNNs apply convolutional filters that capture spatial hierarchies in images.
Convolution operation:

$$f(i,j) = \sum_m \sum_n x(i+m, j+n) \cdot k(m,n)$$

where x is input and k is kernel.

✍️ Problem 07: Recurrent Neural Network (RNN) for Text Classification

Aspect	Description
Problem	Classify text sequences using an RNN-based architecture.
Input	Tokenized text sequences.
Model	RNN (vanilla, LSTM or GRU).
Layers	Embedding → RNN → Dense → Softmax.
Activation Function	tanh, ReLU (hidden); Softmax (output).
Loss Function	Cross-entropy loss.
Optimizer	Adam optimizer.
Training Method	Sequence input processed through time steps.
Output	Class probabilities.
Evaluation	Accuracy, precision, recall, and loss visualization.

Description:
RNNs process sequences by maintaining hidden states:

$$h_t = \tanh(Wx_t + Uh_{t-1} + b)$$

Used for sentiment classification, topic classification etc.

🧠 Problem 08: Transformer Model for Text Classification

Aspect	Description
Problem	Classify text using a Transformer-based model.
Input	Tokenized text with special tokens and positional encodings.
Model	Transformer encoder with multi-head self-attention.
Layers	Embedding + Positional Encoding → Encoder Blocks → Classification Head.
Activation Function	Softmax in the final classification layer.
Loss Function	Cross-entropy.
Optimizer	AdamW.
Training Method	Fine-tuning pretrained transformer (e.g., BERT, RoBERTa).
Output	Predicted class probabilities.
Evaluation	Accuracy, precision, recall, F1-score.

Description:
Transformers use self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Highly parallel and effective for large datasets.

🎨 Problem 09: GAN for Image Generation

Aspect	Description
Problem	Generate new images using Generative Adversarial Network (GAN).
Input	Random noise vectors.
Model	Generator + Discriminator networks.
Layers	Generator: Dense → Reshape → ConvTranspose → Tanh; Discriminator: Conv → LeakyReLU → Sigmoid

Aspect	Description
Activation Function	Tanh (Generator), Sigmoid (Discriminator).
Loss Function	Binary cross-entropy for adversarial loss.
Optimizer	Adam (used for both Generator and Discriminator).
Training Method	Minimax game: Generator tries to fool Discriminator.
Output	Synthetic image samples.
Evaluation	Visual inspection, Inception Score (IS), Fréchet Inception Distance (FID).

Description:

GANs involve a min-max optimization:

$$\min_G \max_D \mathrm{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathrm{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

Generator improves by learning to fool the Discriminator.