# Neural Network Lab Problem Summaries

## Feedforward Neural Network for MNIST Digit Classification

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 = -∑(yᵢ \* log(ŷᵢ))  
where yᵢ is the true label and ŷᵢ is the predicted probability.

|  |  |
| --- | --- |
| 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. |

## Convolutional Neural Network (CNN) for Image Classification

CNNs apply convolutional filters that capture spatial hierarchies in images.  
Convolution operation:  
 f(i, j) = ∑∑ x(i+m, j+n) \* k(m, n)  
where x is input and k is kernel.

|  |  |
| --- | --- |
| 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. |

## Recurrent Neural Network (RNN) for Text Classification

RNNs process sequences by maintaining hidden states:  
 h\_t = tanh(Wx\_t + Uh\_{t-1} + b)  
Used for sentiment classification, topic classification etc.

|  |  |
| --- | --- |
| 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. |

## Transformer Model for Text Classification

Transformers use self-attention:  
 Attention(Q, K, V) = softmax((QKᵀ) / √d\_k) \* V  
Highly parallel and effective for large datasets.

|  |  |
| --- | --- |
| 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. |

## GAN for Image Generation

GANs involve a min-max optimization:  
 min\_G max\_D E[log D(x)] + E[log(1 - D(G(z)))]  
Generator improves by learning to fool the Discriminator.

|  |  |
| --- | --- |
| 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 |
| 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). |

## Perceptron for AND Function with Bipolar Inputs

A perceptron network uses bipolar inputs and targets to learn the AND logic function.  
Decision function:  
 y = sign(w·x + b)

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Evaluate AND function using bipolar perceptron. |
| Input | Bipolar vectors [-1, 1]. |
| Model | Single-layer perceptron. |
| Layers | Input layer → Output layer. |
| Activation Function | Step function or sign function. |
| Loss Function | Perceptron criterion. |
| Optimizer | Perceptron learning rule. |
| Training Method | Supervised training until convergence. |
| Output | Binary classification output. |
| Evaluation | Convergence curve, decision boundary plot. |

## XOR Function Using McCulloch-Pitts Neuron

The XOR function is not linearly separable; combining multiple McCulloch-Pitts neurons creates the desired output.  
Basic neuron rule:  
 y = θ(∑xᵢwᵢ - T)

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Generate XOR using McCulloch-Pitts neuron. |
| Input | Binary input vectors. |
| Model | Multi-layer McCulloch-Pitts network. |
| Layers | Two-layer structure (hidden + output). |
| Activation Function | Threshold function. |
| Loss Function | None (rule-based). |
| Optimizer | Manual weight configuration. |
| Training Method | Rule-defined behavior. |
| Output | Binary result of XOR. |
| Evaluation | Truth table verification, decision boundary plot. |

## SGD with Delta Rule for Logical Function

Stochastic Gradient Descent with delta rule minimizes error by updating weights:  
 Δw = η(d - y)x

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Implement delta rule with SGD. |
| Input | Logical inputs: [0 0 1; 0 1 1; 1 0 1; 1 1 1]. |
| Model | Single-layer perceptron. |
| Layers | Input → Output. |
| Activation Function | Linear or sign function. |
| Loss Function | Mean Squared Error (MSE). |
| Optimizer | Stochastic Gradient Descent. |
| Training Method | Delta learning rule with SGD. |
| Output | Predicted binary targets. |
| Evaluation | Convergence curve, loss graph. |

## Compare SGD and Batch Methods with Delta Rule

Both SGD and batch apply delta rule, but with different update strategies.  
Batch:  
 Δw = η∑(d - y)x

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Compare learning strategies of SGD vs Batch. |
| Input | Logical datasets with targets. |
| Model | Perceptron network. |
| Layers | Input → Output. |
| Activation Function | Linear or sign. |
| Loss Function | MSE. |
| Optimizer | SGD vs Batch gradient descent. |
| Training Method | Delta learning rule. |
| Output | Predicted values using both methods. |
| Evaluation | Compare convergence speed and accuracy. |

## Digit Recognition from 5x5 Pixel Images

Recognize digits using simple neural network on 5x5 pixel inputs.  
Binary pixels form input vector:  
 x ∈ {0,1}^{25}

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Recognize digits 1–5 from 5x5 pixel images. |
| Input | Flattened 25-element binary vectors. |
| Model | Feedforward neural network. |
| Layers | Input (25) → Hidden → Output (5). |
| Activation Function | ReLU or sigmoid. |
| Loss Function | Cross-entropy. |
| Optimizer | SGD or Adam. |
| Training Method | Supervised backpropagation. |
| Output | Digit class prediction. |
| Evaluation | Accuracy, loss curves. |

## CNN for Fruit/Bird/Face Classification

CNN-based image classifier for identifying objects like fruits, birds, or faces.  
 Conv2D → Pool → Flatten → Dense

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Classify objects (face/fruit/bird) in images. |
| Input | Color or grayscale images. |
| Model | Convolutional Neural Network. |
| Layers | Conv → ReLU → Pool → Flatten → Dense. |
| Activation Function | ReLU, Softmax. |
| Loss Function | Cross-entropy. |
| Optimizer | Adam. |
| Training Method | Supervised learning. |
| Output | Object classification. |
| Evaluation | Accuracy, loss/precision curves. |

## 3-Layer ANN Using Backpropagation

Standard 3-layer ANN trained using backpropagation to minimize output error.  
 ∇E = ∂E/∂w = δ \* x

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Train 3-layer ANN using backprop. |
| Input | Numeric input features. |
| Model | Three-layer fully connected ANN. |
| Layers | Input → Hidden → Output. |
| Activation Function | ReLU, sigmoid, or tanh. |
| Loss Function | MSE or cross-entropy. |
| Optimizer | SGD or Adam. |
| Training Method | Backpropagation algorithm. |
| Output | Target classification or regression. |
| Evaluation | Error convergence, accuracy. |

## Speech Recognition for Digits 1–4 Using ANN

Speech features extracted (MFCCs) and classified using ANN.  
Input: Feature vector from audio signal.

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Recognize spoken digits (1–4). |
| Input | Speech features (e.g., MFCC). |
| Model | ANN with dense layers. |
| Layers | Input → Hidden → Output. |
| Activation Function | ReLU and Softmax. |
| Loss Function | Cross-entropy. |
| Optimizer | Adam or RMSProp. |
| Training Method | Supervised backpropagation. |
| Output | Predicted spoken digit. |
| Evaluation | Accuracy, confusion matrix. |

## SVM for Purchase Prediction

Support Vector Machine (SVM) finds a separating hyperplane:  
 w·x + b = 0

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Predict purchase decisions using SVM. |
| Input | Structured numerical or categorical data. |
| Model | Support Vector Machine (SVM). |
| Layers | Not applicable (non-neural). |
| Activation Function | Hinge loss (margin maximization). |
| Loss Function | Hinge loss. |
| Optimizer | Quadratic programming (solver-based). |
| Training Method | SVM fitting on labeled data. |
| Output | Purchase prediction (binary). |
| Evaluation | Accuracy, confusion matrix, ROC. |

## PCA for Dimensionality Reduction

Principal Component Analysis reduces features while retaining variance.  
 Covariance matrix C = (XᵀX)/n

|  |  |
| --- | --- |
| Aspect | Description |
| Problem | Reduce data dimensions using PCA. |
| Input | High-dimensional numeric dataset. |
| Model | Linear transformation via PCA. |
| Layers | Not applicable. |
| Activation Function | None. |
| Loss Function | Reconstruction error minimization. |
| Optimizer | Eigen decomposition. |
| Training Method | Covariance and eigenvalue computation. |
| Output | Projected low-dimensional data. |
| Evaluation | Explained variance, scatter plots. |