Neural Network — A Simple Perceptron

Q1. What is deep learning, and how is it connected to artificial intelligence?

Deep learning is a subset of machine learning (which in turn is a subset of AI) that uses multi-layered neural networks to learn hierarchical representations of data. It enables machines to learn complex patterns directly from raw data with minimal feature engineering.

Q2. What is a neural network, and what are the different types of neural networks?

A neural network is a set of layers of interconnected nodes (neurons) that transform input data through weighted connections and activation functions to produce outputs. Common types include: feedforward (MLP), convolutional (CNN), recurrent (RNN, LSTM/GRU), autoencoders, and transformers.

Q3. What is the mathematical structure of a neural network?

A neural network is composed of layers where each layer performs an affine transformation followed by a non-linear activation: $z = W x + b$, $a = \phi(z)$. Training adjusts weights W and biases b to minimize a loss function over data.

Q4. What is an activation function, and why is it essential in neural networks?

An activation function introduces non-linearity to allow the network to learn complex relationships. Without activation functions, the network would be equivalent to a single linear transformation regardless of depth.

Q5. Could you list some common activation functions used in neural networks?

Common activations: Sigmoid, Tanh, ReLU (Rectified Linear Unit), Leaky ReLU, ELU, Softmax (for multiclass output).

Q6. What is a multilayer neural network?

A multilayer neural network (also called a multilayer perceptron) has one or more hidden layers between input and output. Each hidden layer allows the model to learn higher-level features.

Q7. What is a loss function, and why is it crucial for neural network training?

A loss function measures the difference between predicted outputs and true targets. Training aims to minimize this loss using optimization algorithms; the choice of loss depends on the task (e.g., cross-entropy for classification, MSE for regression).

Q8. What are some common types of loss functions?

Common losses: Mean Squared Error (MSE), Mean Absolute Error (MAE), Binary Cross-Entropy, Categorical Cross-Entropy, Hinge loss.

Q9. How does a neural network learn?

It learns through iterative optimization: forward pass computes predictions and loss; backward pass (backpropagation) computes gradients of loss w.r.t parameters; an optimizer updates parameters using gradients.

Q10. What is an optimizer in neural networks, and why is it necessary?

An optimizer updates network parameters to minimize loss using gradients. It controls step sizes and can include momentum or adaptive learning rates to improve convergence.

Q11. Could you briefly describe some common optimizers?

SGD (Stochastic Gradient Descent), SGD with momentum, Adam (adaptive moment estimation), RMSprop, Adagrad. Adam is popular due to adaptive learning rates and good default performance.

Q12. Can you explain forward and backward propagation in a neural network?

Forward propagation passes input through layers to compute output. Backward propagation (backprop) computes gradients of loss wrt parameters using chain rule, propagating errors from output back to earlier layers.

Q13. What is weight initialization, and how does it impact training?

Weight initialization sets initial parameter values. Proper initialization (e.g., Xavier/Glorot, He initialization) helps avoid vanishing/exploding gradients and leads to faster, more stable training.

Q14. What is the vanishing gradient problem in deep learning?

When gradients become extremely small while backpropagating through many layers, early layers learn very slowly. This can make training deep networks difficult, particularly with sigmoid/tanh activations.

Q15. What is the exploding gradient problem?

When gradients grow very large during backpropagation, causing unstable updates and possible divergence. Techniques like gradient clipping and careful initialization help mitigate this.

Practical — Notes & Short Examples

Practical 1: Simple perceptron example using Keras Sequential; small OR dataset.

Practical 2: MLP with one hidden layer (Dense with ReLU) trained on toy data.

Practical 3: Use Glorot (Xavier) initializer via kernel_initializer=GlorotUniform()

Practical 4: Apply activations like relu, tanh, sigmoid on Dense layers.

Practical 5: Add Dropout layer to reduce overfitting.

Practical 6: Manual forward propagation example with numpy (showing math).

Practical 7: Use BatchNormalization layer between Dense layers.

Practical 8: Plot training loss and accuracy using matplotlib from history object.

Practical 9: Use optimizer arguments clipnorm or clipvalue for gradient clipping.

Practical 10: Define custom loss function using Keras backend.

Practical 11: Use model.summary() or plot_model for visualization.