RAG Knowledge Base: Deep Neural Networks (NN) - Single File

ID: RN-01 | Topic: Fundamentals | Question (P): What is an **Artificial Neuron** (Perceptron)? | **Answer (R):** It's the basic computational unit. It receives inputs (x), multiplies them by **weights (W)**, adds a **bias (b)**, and passes the result through an **activation function (g)** to produce an output.

ID: RN-02 | Topic: Fundamentals | Question (P): What is the difference between a *Shallow* Neural Network and a **Deep Neural Network (DNN)**? | **Answer (R):** A *shallow* network has only one or two hidden layers. A **DNN** has **multiple hidden layers** (generally three or more), which allows it to learn hierarchical representations and complex features from data.

ID: RN-03 | Topic: Architecture | Question (P): What are the **Weights (W)** and **Bias (b)** in an NN? | **Answer (R):** The **Weight** is the parameter that determines the importance of an input connection. The **Bias** is a constant value added to the weighted sum to shift the activation threshold. **These are the parameters the network learns.**

ID: RN-04 | Topic: Architecture | Question (P): What is the key purpose of the **Activation Function** (e.g., ReLU, Sigmoid)? | **Answer (R):** It introduces **non-linearity** to the network. Without it, the network would only calculate linear combinations of the inputs, limiting its capacity to model complex relationships.

ID: RN-05 | Topic: Architecture | Question (P): What is a **Hyperparameter** (e.g., Learning Rate, number of layers)? | **Answer (R):** It is a parameter whose value is **set manually** before the training process begins. It is distinct from weights and biases, which are parameters the network learns automatically.

ID: RN-06 | Topic: Architecture | Question (P): What does the **Softmax** function do in the output layer? | **Answer (R):** It is commonly used in **multiclass classification** problems. It converts the output of the final layer into a **probability distribution**, where the sum of all outputs is 1, indicating the likelihood that the input belongs to each class.

ID: RN-07 | Topic: Feedforward | Question (P): What is the **Feedforward** process (Forward Propagation)? | **Answer (R):** It is the **prediction calculation** process. Input data (X) travels through the network, layer by layer, using the current weights and biases to generate the **final output or prediction** (y^).

ID: RN-08 | Topic: Feedforward | Question (P): How is the **net activation (Z)** of a layer represented in terms of matrices? | **Answer (R):** It is represented as a matrix multiplication for the entire *minibatch* or layer: Z=W·Aprev+B. This allows all neurons to be processed efficiently, which is critical for GPU performance.

ID: RN-09 | Topic: Feedforward | Question (P): What is the mathematical formula for a neuron's activation (a)? | **Answer (R):** The neuron's output is calculated as: a=g(z), where z=(∑iwixi)+b. The term z is called the **weighted sum** or **net activation**.

ID: RN-10 | Topic: Functions | Question (P): Why is the **ReLU** function the most widely used in hidden layers? | **Answer (R):** The Rectified Linear Unit (ReLU) is simple to compute (max(0,z)) and, crucially, it helps **mitigate the vanishing gradient problem** compared to Sigmoid or Tanh, especially in deep networks.

ID: RN-11 | Topic: Learning | Question (P): What is the **Cost Function** (or Loss Function)? | **Answer (R):** It is a metric that quantifies the network's **error** by measuring the difference between the prediction (y^) and the actual value (y). The goal of training is to find the weights that **minimize** this function.

ID: RN-12 | Topic: Backpropagation | Question (P): What is the **Backpropagation** process? | **Answer (R):** It is the algorithm that efficiently computes the **gradients** (the partial derivative of the cost function with respect to every parameter, ∂W∂L) to determine how much each weight contributed to the network's total error.

ID: RN-13 | Topic: Backpropagation | Question (P): What is the role of the **Chain Rule** in *Backpropagation*? | **Answer (R):** The chain rule from calculus is essential. It allows the total gradient calculation to be **decomposed** into the product of local gradients for each layer and neuron, propagating the error **backward** through the network.

ID: RN-14 | Topic: Optimization | Question (P): What is **Gradient Descent** and what does it do? | **Answer (R):** It is the optimization algorithm that uses the **gradients** calculated by *Backpropagation* to **iteratively update the weights and biases** in the direction opposite to the gradient (the direction of **steepest descent**) to minimize the cost function.

ID: RN-15 | Topic: Optimization | Question (P): What is the **Learning Rate (α)**? | **Answer (R):** It is a key **hyperparameter** that defines the **step size** in each iteration of Gradient Descent. A high rate can "overshoot" the minimum, while a low rate makes training very slow.

ID: RN-16 | Topic: Matrices/Gradients | Question (P): How is the calculation of the **weight gradient (∂W∂L)** represented in matrices? | **Answer (R):** The gradient is calculated as a matrix product (typically AprevT·δ), combining the error vector of the current layer (δ) with the activation of the previous layer (Aprev) to obtain the weight updates for the entire layer.

ID: RN-17 | Topic: Optimization | Question (P): What is the difference between **Stochastic Gradient Descent (SGD)** and *Batch* Gradient Descent? | **Answer (R): SGD** updates weights after every **single data sample**, introducing more noise but potentially finding better minima. *Batch* Gradient Descent updates weights after processing the **entire training set**, which is slower but more stable.

ID: RN-18 | Topic: Optimization | Question (P): How does an optimizer like **ADAM** work? | **Answer (R):** ADAM (Adaptive Moment Estimation) is an optimizer that **adaptively adjusts the learning rate** for each weight individually, using an exponential moving average of both the gradients (momentum) and the squared gradients (velocity) to accelerate convergence.

ID: RN-19 | Topic: Regularization | Question (P): What are **Overfitting** and **Underfitting**? | **Answer (R): Overfitting** occurs when the network learns the noise in the training data too well, failing to generalize. **Underfitting** occurs when the network is too simple and fails to capture the essential relationships in the data.

ID: RN-20 | Topic: Regularization | Question (P): How does the **Dropout** technique help prevent *Overfitting*? | **Answer (R):** During training, *Dropout* **randomly deactivates** a percentage of neurons in the hidden layers. This forces the network to avoid relying on any specific neuron or feature, improving its **generalization ability**.

ID: RN-21 | Topic: RN Types | Question (P): What is a **Convolutional Neural Network (CNN)** and what is it used for? | **Answer (R):** It is a network specialized for processing grid-like data (primarily **images**). It uses **convolutional layers** and *pooling* to automatically extract spatial features from the data.

ID: RN-22 | Topic: RN Types | Question (P): What is a **Recurrent Neural Network (RNN)** and what is its main characteristic? | **Answer (R):** It is a network designed to process **sequence data** (text, audio, time series). Its main characteristic is the **recurrent connection** (loop) that allows information from previous time steps to persist and be used in the current time step.

ID: RN-23 | Topic: Problems | Question (P): What is the **Vanishing Gradient** problem? | **Answer (R):** It occurs in deep networks when the gradient becomes progressively **smaller** as it backpropagates, causing the weights in the initial layers to barely update, and the network to stop learning effectively.

ID: RN-24 | Topic: Problems | Question (P): What is the **Exploding Gradient** problem? | **Answer (R):** It's the opposite of vanishing gradient. It occurs when gradients become **too large** during *Backpropagation*, leading to massive and unstable weight updates, causing the model to diverge or produce NaN (Not a Number).

ID: RN-25 | Topic: Problems | Question (P): What is a common solution for the Exploding Gradient problem? | **Answer (R):** A common technique is **gradient clipping**, where gradients are limited to a predefined threshold if they exceed a certain value, preventing weight updates from becoming too large and unstable.