# Artificial Neural Networks and TensorFlow - Comprehensive Question Bank

## Question 1: Linear Threshold Unit and Perceptron Fundamentals (8 marks)

### Part A (4 marks)

Given a Linear Threshold Unit (LTU) with inputs $x_1 = 0.8$, $x_2 = -0.3$, $x_3 = 0.6$ and weights $w_1 = 0.5$, $w_2 = -0.7$, $w_3 = 0.4$:

1. Calculate the weighted sum $z = w^T x$ (show all steps)
2. Apply the step function to determine the output $\hat{y}$
3. Explain why the step function creates a linear decision boundary

### Part B (4 marks)

A perceptron is being trained for binary classification with the following data:

- Training instance: $x = [1.2, -0.5, 0.8]$, true label $y = 1$
- Current weights: $w = [0.3, -0.2, 0.6]$, bias $b = 0.1$
- Learning rate $\alpha = 0.1$

Using the sigmoid activation function $\sigma(z) = 1/(1 + e^{-z})$:

1. Calculate the predicted output $\hat{y}$ (show all calculation steps)
2. Compute the Mean Squared Error $J(w,b) = (y - \hat{y})^2$
3. Update the first weight $w_1$ using gradient descent: $w_1(new) = w_1 - \alpha \cdot \partial J/\partial w_1$
4. Explain why sigmoid is preferred over step function for training

---

## Question 2: Multi-Layer Perceptron Architecture Design (10 marks)

### Scenario

You are tasked with designing neural networks for two different problems:

1. **Problem A**: Predicting house prices (regression) with 15 input features
2. **Problem B**: Classifying emails into 5 categories (multi-class classification)

### Part A: Regression Network Design (5 marks)

For the house price prediction problem:

1. Specify the complete network architecture including:
   - Number of input neurons and justification

- Recommended number of hidden layers and neurons per layer
- Number of output neurons
- Activation functions for hidden and output layers
- Appropriate loss function

2. Explain why you chose ReLU activation for hidden layers
3. Justify your choice of output layer activation (or lack thereof)

### Part B: Classification Network Design (5 marks)

For the email classification problem:

1. Design the network architecture specifying:
   - Input layer configuration
   - Hidden layer recommendations
   - Output layer setup with activation function
   - Appropriate loss function
2. Explain why softmax is suitable for the output layer
3. Calculate what the softmax outputs would be for intermediate values z = [2.1, 1.0, 0.5, 1.8, 0.2] (show all steps)

---

## Question 3: Cost Functions and Training Process (12 marks)

### Part A: Mathematical Calculations (6 marks)

Given the following data for a 3-class classification problem:

- True labels (one-hot encoded): $y_1 = [1,0,0]$, $y_2 = [0,1,0]$, $y_3 = [0,0,1]$
- Predicted probabilities: $\hat{y}_1 = [0.7,0.2,0.1]$, $\hat{y}_2 = [0.1,0.8,0.1]$, $\hat{y}_3 = [0.2,0.3,0.5]$

1. Calculate the cross-entropy loss for each sample: $\text{cross\_entropy}(y_i, \hat{y}_i) = -\Sigma_j \, y_{j,i} \log(\hat{y}_{j,i})$
2. Compute the total cost function: $\text{cost}(y, \hat{y}) = (1/m) \, \Sigma_{i=1}^{m} \text{cross\_entropy}(y_i, \hat{y}_i)$
3. Compare this with using MSE - explain why cross-entropy is preferred for classification

### Part B: Backpropagation Process (6 marks)

For a simple 2-layer MLP with:

- Input layer: 2 neurons
- Hidden layer: 3 neurons with ReLU activation
- Output layer: 1 neuron with sigmoid activation
- Training instance: x = [0.5, -0.2], target y = 1

Describe the complete training process:

1. **Forward Pass**: Explain each step from input to final prediction
2. **Error Calculation**: How the cost function is computed
3. **Backward Pass**: Describe how error gradients flow backward through the network
4. **Weight Update**: Explain the gradient descent step
5. Explain the role of the chain rule in backpropagation

---

## Question 4: TensorFlow/Keras Implementation (11 marks)

### Part A: Complete Implementation (8 marks)

Write complete TensorFlow/Keras code to solve the Fashion-MNIST classification problem with the following specifications:

**Requirements:**

- Load and preprocess Fashion-MNIST dataset
- Create an MLP with:
  - Flatten layer for input
  - Two hidden layers (128 and 64 neurons) with ReLU activation
  - Output layer with appropriate activation for 10-class classification
- Use appropriate loss function and optimizer
- Train for 20 epochs with validation split
- Include early stopping when validation accuracy stops improving

**Your code should include:**

1. Data loading and preprocessing (normalize pixel values)
2. Model architecture definition using Sequential API
3. Model compilation with appropriate parameters
4. Training with validation data
5. Model evaluation and prediction example

### Part B: Parameter Analysis (3 marks)

For the model you created above:

1. Calculate the total number of trainable parameters (show calculations)
2. Explain why you chose the specific loss function
3. Discuss the trade-off between model complexity and overfitting risk

---

## Question 5: Hyperparameter Tuning and Regularization (9 marks)

### Part A: Hyperparameter Optimization (5 marks)

You are tasked with optimizing a neural network for a binary classification problem. Write a complete Keras Tuner implementation to find the best hyperparameters for:

- Number of hidden layers (1-4)
- Number of neurons per layer (32-256)
- Learning rate (1e-4 to 1e-1, log scale)
- Optimizer choice (SGD, Adam, RMSprop)

Your implementation should:

1. Define a `build_model(hp)` function that creates the tunable model
2. Set up the tuner with appropriate search strategy
3. Include proper search parameters and constraints
4. Show how to retrieve and use the best model

### Part B: Regularization Techniques (4 marks)

Given a neural network that is overfitting:

1. Implement L1 and L2 regularization in Keras with $\lambda_1 = 0.01$ and $\lambda_2 = 0.02$
2. Explain mathematically how L1 regularization differs from L2:
   - L1: $J(W) = J(W) + \lambda_1 \Sigma_i |w_i|$
   - L2: $J(W) = J(W) + \lambda_2 \Sigma_i w_i^2$
3. Describe the effect of each regularization type on the model weights
4. Implement early stopping with patience=5 and monitor validation loss

---

## Question 6: Advanced Concepts and Problem Solving (10 marks)

### Part A: Activation Functions and Initialization (4 marks)

1. Compare the mathematical properties of the following activation functions:
   - ReLU: f(x) = max(0, x)
   - Sigmoid: f(x) = 1/(1 + e^(-x))
   - Tanh: f(x) = 2σ(2x) - 1

   For each function, discuss:
   - Output range

- Derivative characteristics
- Suitable use cases

2. Explain why He initialization is recommended for ReLU networks and Glorot initialization for sigmoid/tanh networks

### Part B: Real-World Application (6 marks)

**Scenario**: A telecommunications company wants to predict customer churn (binary classification) using customer data with 25 features. They have 100,000 training samples.

Design a complete solution including:

1. **Data Preprocessing Strategy**:
   - How to handle different feature scales
   - Train/validation/test split recommendations
2. **Network Architecture**:
   - Justify your architecture choices
   - Include regularization strategies
   - Select appropriate optimizers and learning rates
3. **Training Strategy**:
   - Batch size selection and justification
   - Early stopping criteria
   - Performance metrics beyond accuracy
4. **Code Implementation**: Write the key parts of the TensorFlow/Keras implementation including model definition, compilation, and training setup

---

## Bonus Questions for Advanced Students

### Bonus Question 1: Mathematical Derivation (5 marks)

Derive the gradient of the cross-entropy loss with respect to the weights in the output layer of a neural network. Show all mathematical steps including the application of the chain rule.

### Bonus Question 2: Optimization Comparison (5 marks)

Compare SGD, Adam, and RMSprop optimizers:

1. Explain the mathematical formulation of each
2. Discuss convergence characteristics
3. Provide scenarios where each would be preferred
4. Implement a simple comparison study in code

## Practical Coding Challenges

### Challenge 1: From Scratch Implementation (8 marks)

Implement a simple 2-layer neural network from scratch in Python (without TensorFlow/Keras) for binary classification:

```python
class SimpleNeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        # Initialize weights and biases
        pass

    def sigmoid(self, x):
        # Implement sigmoid activation
        pass

    def forward(self, X):
        # Implement forward pass
        pass

    def backward(self, X, y, output):
        # Implement backpropagation
        pass

    def train(self, X, y, epochs, learning_rate):
        # Implement training loop
        pass
```

Complete this implementation with proper mathematical formulations.

### Challenge 2: Model Comparison (6 marks)

Given a dataset with 1000 samples and 20 features for regression:

1. Implement three different network architectures
2. Compare their performance using appropriate metrics
3. Plot learning curves for each model
4. Provide recommendations based on the results

---

## Answer Guidelines and Marking Rubric

### Theoretical Questions:

- **Mathematical Accuracy** (40%): Correct formulas and calculations
- **Conceptual Understanding** (30%): Clear explanations of concepts
- **Justification** (20%): Reasoning behind choices and recommendations
- **Presentation** (10%): Clear step-by-step solutions

### Practical Questions:

- **Code Correctness** (50%): Syntactically correct and functional code
- **Implementation Logic** (25%): Proper use of TensorFlow/Keras APIs
- **Code Documentation** (15%): Comments and clear structure
- **Testing/Validation** (10%): Proper evaluation methods

### Key Learning Objectives Assessed:

1. Understanding of neural network fundamentals (LTU, Perceptron, MLP)
2. Mathematical foundations (cost functions, backpropagation, optimization)
3. Practical implementation skills with TensorFlow/Keras
4. Hyperparameter tuning and regularization techniques
5. Real-world problem-solving and architecture design
6. Performance evaluation and model comparison

---

*Note: This question bank is designed to assess both theoretical understanding and practical implementation skills. Students should be prepared to show detailed mathematical calculations, write complete code implementations, and provide justified reasoning for their design choices.*