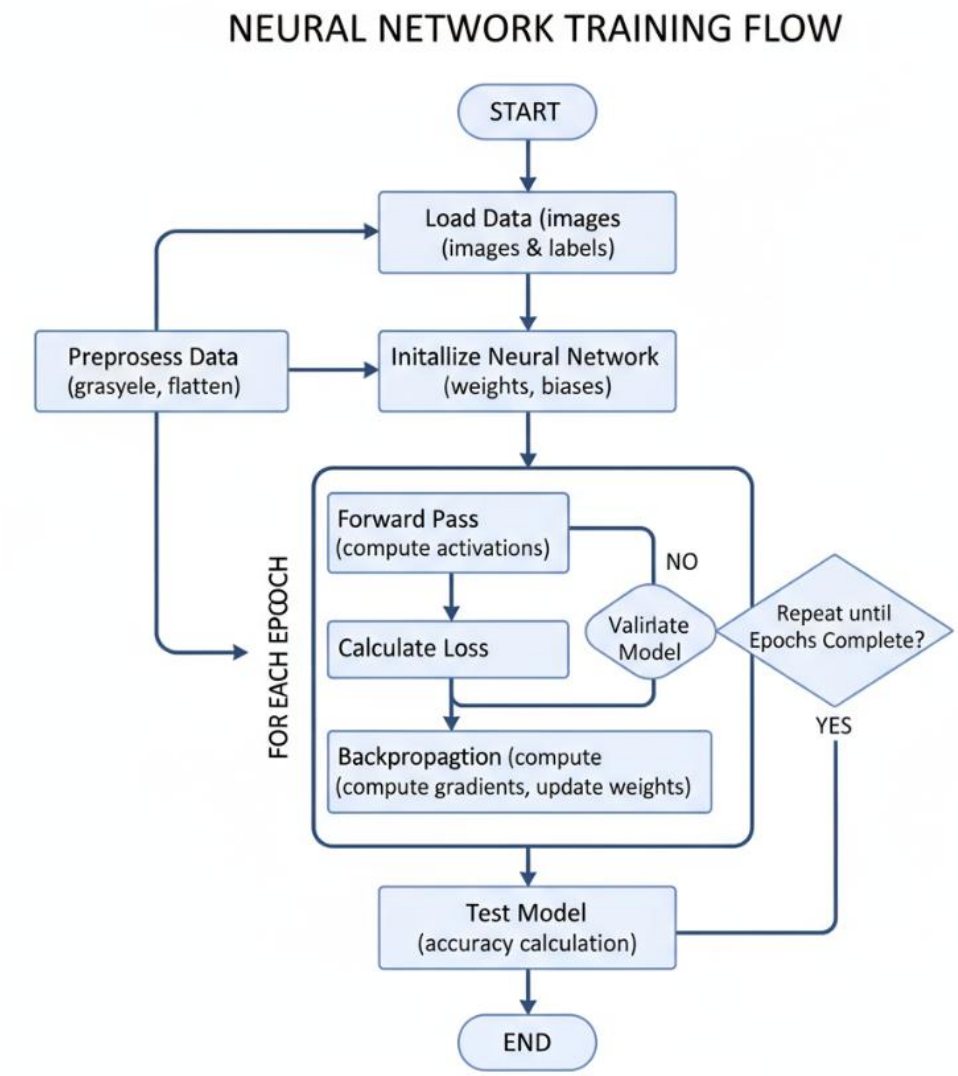

(Program Flowchart)

QUESTION ANSWERS

Q1: Why do we use a one-hot encoded label array instead of an integer for the class label?

Answer: One-hot encoding represents the class as a vector where only the correct class index

is 1, and others are 0. This format is essential for classification tasks because it enables the neural network to output probabilities for each class, which are then compared to these vectors using cross-entropy loss. Unlike integers, one-hot vectors don't assume any order or relationship between classes, preventing the model from mistakenly thinking some classes are "closer" than others.

Q2: Why convert image and label data into NumPy arrays?

Answer: NumPy arrays provide efficient storage and fast mathematical operations. Neural network computations involve heavy matrix and vector calculations, which NumPy handles optimally using compiled C code. Using arrays instead of Python lists speeds up training drastically and allows vectorized operations that are much more efficient than Python loops.

Q3: Why convert input images to grayscale by extracting only the first color channel?

Answer: The Fashion MNIST dataset is inherently grayscale, meaning images carry information only in intensity, not color. Extracting one channel simplifies the data, reduces computational load, and maintains meaningful image content. This also standardizes the input shape to 28×28 single-channel images needed for the network.

Q4: Why reshape 28×28 pixel images into 784-element vectors?

Answer: Neural networks with fully connected layers expect 1D feature input vectors. Reshaping the 2D images into 1D vectors creates a straight sequence of pixel data, suitable for matrix multiplication with the weight matrices. It transforms the spatial image data into a format the network can process.

Q5: What is the learning rate and why is it important?

Answer: The learning rate controls how much the weights update during each iteration. A high learning rate speeds learning but risks overshooting minima and missing the best solution. A too-low learning rate makes training slow and might get stuck in suboptimal points. It's a key hyperparameter that balances speed and stability in training.

Q6: Why must weight matrices be shaped as (hidden_neurons, input_neurons) and (output_neurons, hidden_neurons)?

Answer: The matrix shapes ensure that the weight matrices correctly multiply with input or hidden outputs per layer during forward propagation. This shape compatibility is crucial for

proper mathematical matrix operations—allowing the input to flow correctly through hidden layers to outputs.

Q7: What is broadcasting, and why must biases be broadcast?

Answer: Broadcasting is a feature where arrays of different shapes can still participate in arithmetic by "stretching" smaller arrays along dimensions to match larger arrays without copying data. Bias terms often have shape (neurons,1) and must be automatically expanded to be added to batch activations with shape (neurons, batch_size).

Q8: What does np.random.randn do, and why use He initialization?

Answer: np.random.randn generates random numbers from a standard normal distribution (mean 0, variance 1), used to randomly initialize weights. He initialization scales these weights depending on the number of inputs to a neuron, improving training stability for ReLU networks by preventing gradients from vanishing or exploding.

Q9: What is the benefit of the ReLU activation function?

Answer: ReLU introduces non-linearity, which allows networks to learn complex patterns. It outputs zero for negative inputs and passes positive inputs unchanged, helping mitigate the vanishing gradient problem by keeping gradients alive and speeding up training compared to sigmoid or tanh activations.

Q10: What is the softmax function and why is it used in the output layer?

Answer: Softmax converts raw output scores (logits) into probabilities that sum to 1 across classes. This makes the outputs interpretable as likelihoods for multi-class classification and works synergistically with cross-entropy loss to optimize the network.

Q11: What are loss functions, and why use cross-entropy loss for classification?

Answer: A loss function measures how far predictions are from true labels. Cross-entropy loss is ideal for classification because it quantifies the distance between the predicted probability distribution and the true distribution (one-hot), penalizing wrong predictions more heavily and providing clear gradients for learning.

Q12: In forward propagation, why does the network output an array of 10 elements per image?

Answer: Each element corresponds to the predicted probability that the input image belongs to one of the 10 fashion classes. This vector represents a probability distribution over all classes, allowing the model to express uncertainty for every class.

Q13: Why subtract the mean from the inputs in the forward pass?

Answer: Mean subtraction normalizes the inputs, centering the data distribution around zero. This stabilizes gradient updates, reduces bias during training, and often accelerates convergence by making the data more uniform.

Q14: Why is softmax specifically used in the output layer?

Answer: Softmax ensures outputs are valid probabilities (all positive and sum to 1), making it possible to interpret predictions probabilistically. Using softmax also pairs well mathematically with cross-entropy loss, simplifying gradient calculations.

Q15: Why perform a forward pass again within backpropagation instead of reusing stored outputs?

Answer: Backpropagation requires intermediate activations and pre-activation values to compute gradients correctly. While storing these during the forward pass is efficient, rerunning the forward pass makes the code clearer and ensures fresh values for gradient calculations.

Q16: What is validation data, and why is it important?

Answer: Validation data is a separate subset of data not used for training but used to evaluate model performance during training. It helps detect overfitting—when the model performs well on training data but poorly on new data—and guides hyperparameter tuning for better generalization.

Q17: Explain the parameters input_neurons, hidden_neurons, output_neurons, learning_rate, and epochs.

Answer:

- input_neurons: Number of features fed into the network (784 pixels per image).
- hidden_neurons: Size of the hidden layer determining model capacity and complexity.
- output_neurons: Number of classes to predict (10 for Fashion-MNIST).

- learning_rate: Step size for each update during training, balancing speed and stability.
- epochs: Number of full passes over the entire training dataset.

Q18: Why use argmax on the model output for predictions?

Answer: Argmax selects the class with the highest predicted probability, converting the output probability vector into a single class label. This final discrete prediction allows evaluation by comparing to true labels.