Question 4: Train the model for at least 10 epochs, and plot the performance. Define a function `train_model` that takes the training and validation dataloaders, optimizer, loss function, and the number of epochs to train.

The function should return the training losses and validation accuracies.

The training loop is as follows:

```
For each epoch:
   Set model to training mode
   Send model to device (GPU) using .to(device)
   For each batch in training dataloader:
     Get inputs and labels from batch. Send inputs to GPU device.

     Zero out gradients
     Forward pass through model
     Calculate loss
     Backward pass
     Update weights

   Set model to evaluation mode - remember, no backprop/gradient calculation!
   For each batch in validation dataloader:
     Get inputs and labels from batch
     Forward pass through model
     Calculate validation loss

   Store training and validation metrics. Print them if you'd like.
```

```
In [28]: # FOR FASTER TRAINING, USE GPU IF AVAILABLE
         device = torch.device('cuda' if torch.cuda.is_available(
         ) else 'mps' if torch.backends.mps.is_available() else 'cpu')

         print(f"Using device: {device}")
```

```
Using device: mps
```

```
In [59]: from tqdm import tqdm

         train_losses = []
         val_losses = []
         train_accuracies = []
         val_accuracies = []

         model.to(device)
```

```python
# Training loop
def train_model(model, training_loader, validation_loader, optimizer, loss_fn, device, EPOCHS)
    train_losses = []
    val_losses = []
    train_accuracies = []
    val_accuracies = []

    model.to(device)

    # Training loop
    for epoch in tqdm(range(EPOCHS), desc="Epochs"):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        # Training phase
        for inputs, labels in training_loader:
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = model(inputs)

            # YOUR CODE HERE
            loss = loss_fn(outputs, labels)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()

            running_loss += loss.item()

            predicted = torch.argmax(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        epoch_loss = running_loss / len(training_loader)
        epoch_acc = 100 * correct / total
        train_losses.append(epoch_loss)
        train_accuracies.append(epoch_acc)

        model.eval()
        val_loss = 0.0
        correct = 0
        total = 0

        with torch.no_grad():
            for inputs, labels in validation_loader:
                inputs, labels = inputs.to(device), labels.to(device)

                outputs = model(inputs)
                loss = loss_fn(outputs, labels)

                val_loss += loss.item()

                predicted = torch.argmax(outputs, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum().item()

        val_loss = val_loss / len(validation_loader)
        val_acc = 100 * correct / total
        val_losses.append(val_loss)
        val_accuracies.append(val_acc)

        print(f'Epoch [{epoch+1}/{EPOCHS}]')
        print(f'Train Loss: {epoch_loss:.4f}, Train Acc: {epoch_acc:.2f}%')
        print(f'Val Loss: {val_loss:.4f}, Val Acc: {val_acc:.2f}%')
        print('-' * 60)

    return train_losses, val_losses, train_accuracies, val_accuracies
```

```
In [61]: EPOCHS = 10

         train_losses, val_losses, train_accuracies, val_accuracies = train_model(model,
                                                                                  training_loader,
                                                                                  validation_loader,
                                                                                  optimizer,
                                                                                  loss_fn,
                                                                                  device,
                                                                                  EPOCHS)
```

```
Epochs:  10%|                                  | 1/10 [00:10<01:35, 10.63s/it]
```

```
Epoch [1/10]
Train Loss: 2.4113, Train Acc: 38.88%
Val Loss: 2.1730, Val Acc: 43.17%
------------------------------------------------------------
```

```
Epochs:  20%|                                | 2/10 [00:21<01:25, 10.64s/it]
```

```
Epoch [2/10]
Train Loss: 2.1143, Train Acc: 44.67%
Val Loss: 1.8392, Val Acc: 50.70%
------------------------------------------------------------
```

```
Epochs:  30%|                               | 3/10 [00:31<01:14, 10.61s/it]
```

```
Epoch [3/10]
Train Loss: 1.8640, Train Acc: 49.88%
```

```
Val Loss: 1.6074, Val Acc: 56.32%
------------------------------------------------------------



Epochs:  40%|                             | 4/10 [00:42<01:03, 10.60s/it]


Epoch [4/10]
Train Loss: 1.6404, Train Acc: 55.06%
Val Loss: 1.4162, Val Acc: 60.97%
------------------------------------------------------------



Epochs:  50%|                            | 5/10 [00:53<00:53, 10.60s/it]


Epoch [5/10]
Train Loss: 1.4179, Train Acc: 60.42%
Val Loss: 1.1173, Val Acc: 69.03%
------------------------------------------------------------



Epochs:  60%|                          | 6/10 [01:03<00:42, 10.59s/it]


Epoch [6/10]
Train Loss: 1.1933, Train Acc: 66.09%
Val Loss: 0.8991, Val Acc: 75.24%
------------------------------------------------------------



Epochs:  70%|                        | 7/10 [01:14<00:31, 10.58s/it]


Epoch [7/10]
Train Loss: 0.9638, Train Acc: 72.00%
Val Loss: 0.7352, Val Acc: 79.71%
------------------------------------------------------------



Epochs:  80%|                       | 8/10 [01:24<00:21, 10.58s/it]


Epoch [8/10]
Train Loss: 0.7708, Train Acc: 77.44%
```

```
Val Loss: 0.5429, Val Acc: 85.14%
------------------------------------------------------------



Epochs:  90%|                     | 9/10 [01:35<00:10, 10.57s/it]


Epoch [9/10]
Train Loss: 0.5873, Train Acc: 82.63%
Val Loss: 0.4511, Val Acc: 87.14%
------------------------------------------------------------



Epochs: 100%|                   | 10/10 [01:45<00:00, 10.59s/it]


Epoch [10/10]
Train Loss: 0.4399, Train Acc: 87.13%
Val Loss: 0.3454, Val Acc: 90.60%
------------------------------------------------------------
```

Question 5: Plotting model performance.

Create two plots: - Training loss against epochs - Validation accuracy epochs

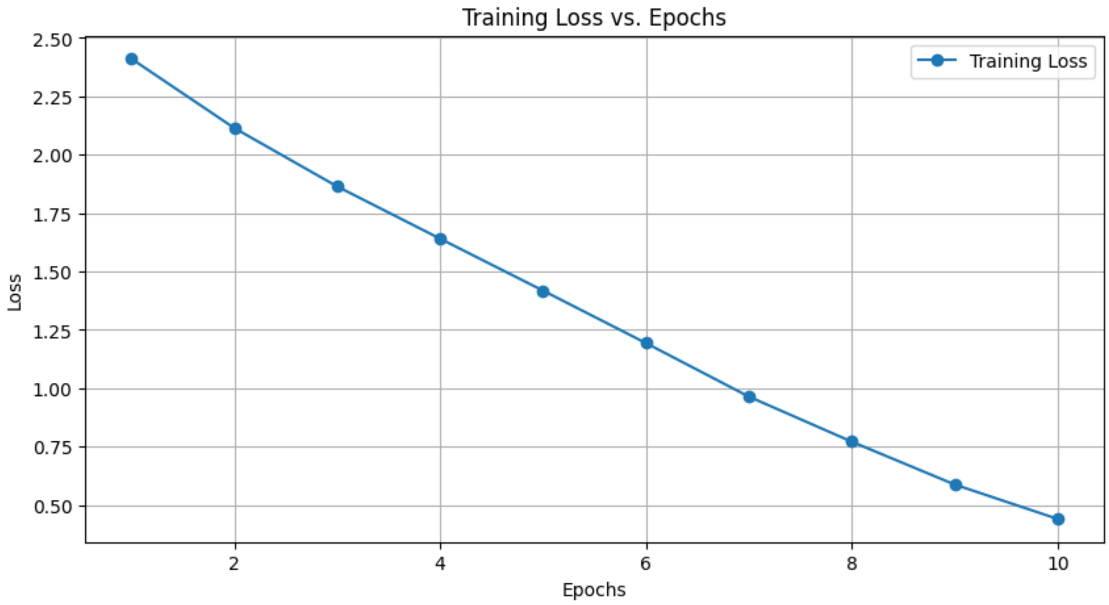
Make sure your plots are appropriately labeled.

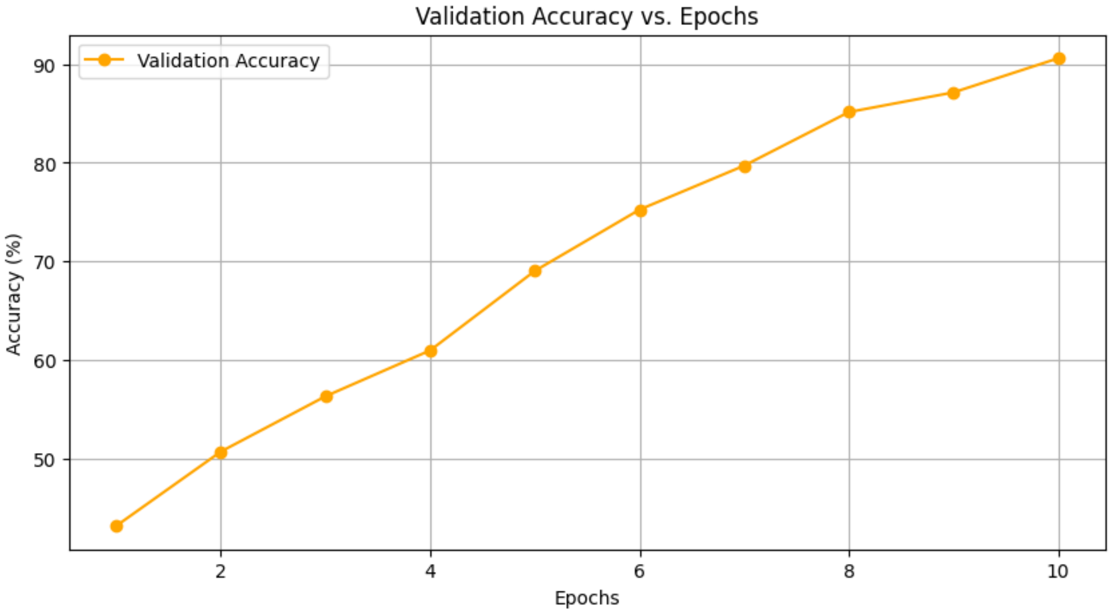
```
In [62]: print("Train Losses Length:", len(train_losses), "Values:", train_losses)
         print("Validation Losses Length:", len(val_losses), "Values:", val_losses)
         print("Train Accuracies Length:", len(train_accuracies), "Values:", train_accuracies)
         print("Validation Accuracies Length:", len(val_accuracies), "Values:", val_accuracies)

         if len(train_losses) == EPOCHS:
             plt.figure(figsize=(10, 5))
             plt.plot(range(1, EPOCHS + 1), train_losses, label='Training Loss', marker='o')
             plt.xlabel('Epochs')
             plt.ylabel('Loss')
             plt.title('Training Loss vs. Epochs')
             plt.legend()
             plt.grid(True)
             plt.show()
         else:
             print("Error: train_losses size mismatch with EPOCHS.")

         if len(val_accuracies) == EPOCHS:
             plt.figure(figsize=(10, 5))
             plt.plot(range(1, EPOCHS + 1), val_accuracies, label='Validation Accuracy', marker='o', col
             plt.xlabel('Epochs')
             plt.ylabel('Accuracy (%)')
             plt.title('Validation Accuracy vs. Epochs')
             plt.legend()
             plt.grid(True)
             plt.show()
         else:
             print("Error: val_accuracies size mismatch with EPOCHS.")
```

```
Train Losses Length: 10 Values: [2.4112659579957536, 2.1143187661000225, 1.8640483878457639, 1.64041913
Validation Losses Length: 10 Values: [2.1729505484366354, 1.839203498400081, 1.6074101385253166, 1.4161
Train Accuracies Length: 10 Values: [38.884, 44.672, 49.884, 55.064, 60.424, 66.092, 72.002, 77.442, 82
Validation Accuracies Length: 10 Values: [43.166, 50.704, 56.316, 60.972, 69.034, 75.244, 79.71, 85.142
```


Training Loss vs. Epochs
Training Loss
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
Loss
2
4
6
8
10
Epochs


Validation Accuracy vs. Epochs
Validation Accuracy
90
80
70
60
50
Accuracy (%)
2
4
6
8
10
Epochs

Compare the loss and accuracy plots for training versus validation. What do you notice?

(Replace this text with your answer.)