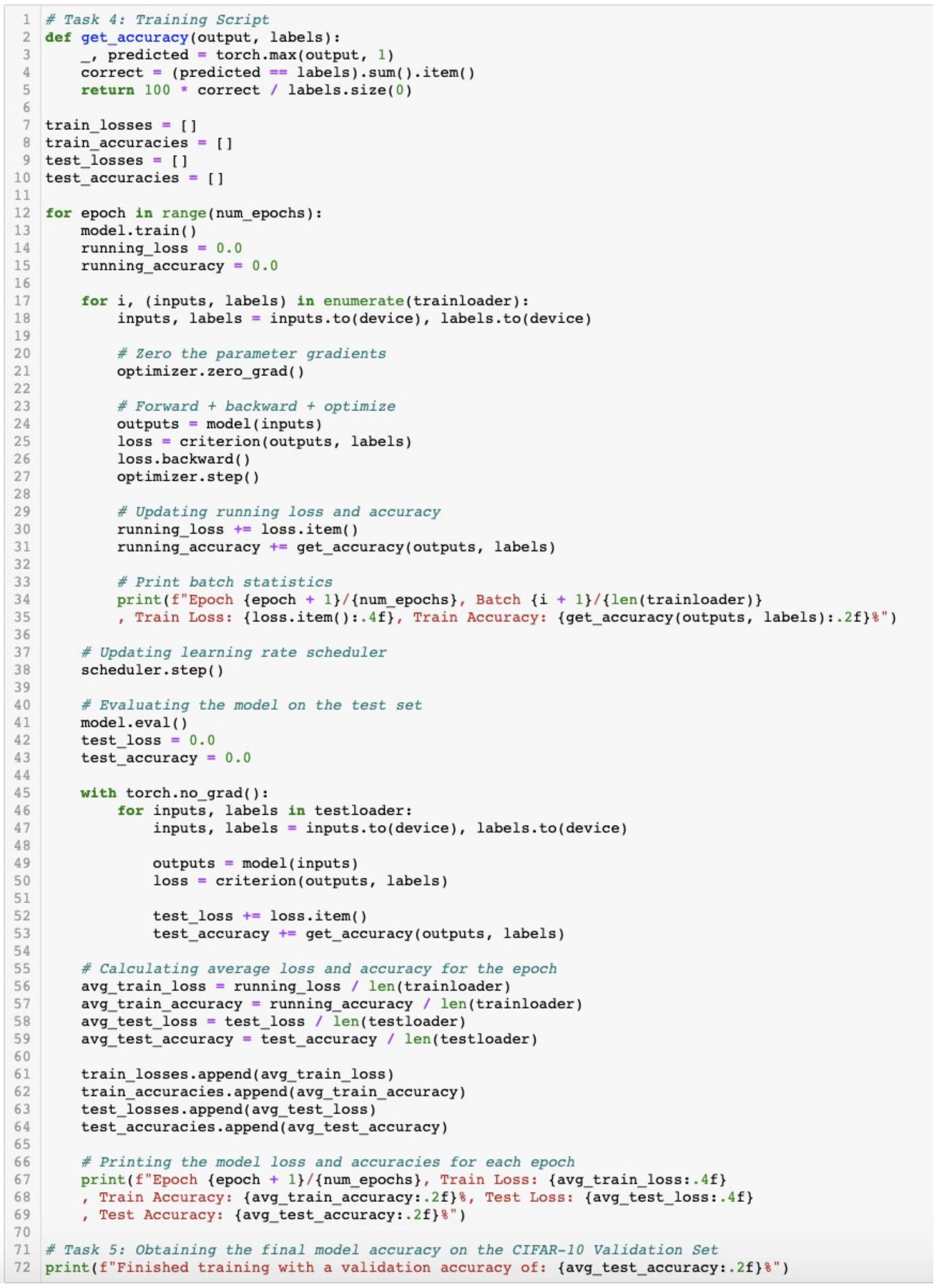
Neural Networks and Deep Learning Coursework Tasks 4 & 5

Task 4: Write the training script to train the model (30%)



```python
# Task 4: Training Script
def get_accuracy(output, labels):
    _, predicted = torch.max(output, 1)
    correct = (predicted == labels).sum().item()
    return 100 * correct / labels.size(0)

train_losses = []
train_accuracies = []
test_losses = []
test_accuracies = []

for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    running_accuracy = 0.0

    for i, (inputs, labels) in enumerate(trainloader):
        inputs, labels = inputs.to(device), labels.to(device)

        # Zero the parameter gradients
        optimizer.zero_grad()

        # Forward + backward + optimize
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # Updating running loss and accuracy
        running_loss += loss.item()
        running_accuracy += get_accuracy(outputs, labels)

        # Print batch statistics
        print(f"Epoch {epoch + 1}/{num_epochs}, Batch {i + 1}/{len(trainloader)}
        , Train Loss: {loss.item():.4f}, Train Accuracy: {get_accuracy(outputs, labels):.2f}%")

    # Updating learning rate scheduler
    scheduler.step()

    # Evaluating the model on the test set
    model.eval()
    test_loss = 0.0
    test_accuracy = 0.0

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

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

            test_loss += loss.item()
            test_accuracy += get_accuracy(outputs, labels)

    # Calculating average loss and accuracy for the epoch
    avg_train_loss = running_loss / len(trainloader)
    avg_train_accuracy = running_accuracy / len(trainloader)
    avg_test_loss = test_loss / len(testloader)
    avg_test_accuracy = test_accuracy / len(testloader)

    train_losses.append(avg_train_loss)
    train_accuracies.append(avg_train_accuracy)
    test_losses.append(avg_test_loss)
    test_accuracies.append(avg_test_accuracy)

    # Printing the model loss and accuracies for each epoch
    print(f"Epoch {epoch + 1}/{num_epochs}, Train Loss: {avg_train_loss:.4f}
    , Train Accuracy: {avg_train_accuracy:.2f}%, Test Loss: {avg_test_loss:.4f}
    , Test Accuracy: {avg_test_accuracy:.2f}%")

# Task 5: Obtaining the final model accuracy on the CIFAR-10 Validation Set
print(f"Finished training with a validation accuracy of: {avg_test_accuracy:.2f}%")
```

First, the function get_accuracy is defined to calculate the accuracy of the model's predictions. This function takes the model's output tensor and the ground truth label tensor as inputs and returns the percentage of correctly classified examples in the batch. Then, four lists are defined to store the training and test losses and accuracies throughout the training process.

Next, a loop is started that runs for num_epochs iterations. Within each epoch, the model is put in training mode (model.train()) and the running loss and accuracy are initialized to 0.0. Then, a nested loop is started that iterates over the training data loader. For each batch of data, the input and label tensors are moved to the GPU (if available), the optimizer's gradients are zeroed (optimizer.zero_grad()), the model's forward pass is calculated (model(inputs)), the loss is computed (criterion(outputs, labels)), the gradients are backpropagated through the model (loss.backward()), and the optimizer's parameters are updated (optimizer.step()).

The running loss and accuracy are updated for each batch, and the current batch statistics (epoch, batch number, train loss, and train accuracy) are printed to the console. After the training loop completes for the epoch, the learning rate scheduler is updated (scheduler.step()), and the model is put in evaluation mode (model.eval()).

Then, another loop is started that iterates over the test data loader. For each batch of data, the input and label tensors are moved to the GPU, the model's forward pass is calculated (model(inputs)), and the loss is computed (criterion(outputs, labels)).

The running test loss and accuracy are updated for each batch. After the test loop completes for the epoch, the average train loss and accuracy and the average test loss and accuracy are calculated for the epoch. These values are appended to the corresponding lists. Finally, the epoch's loss and accuracies are printed to the console. After the training loop completes, the final test accuracy is printed to the console.

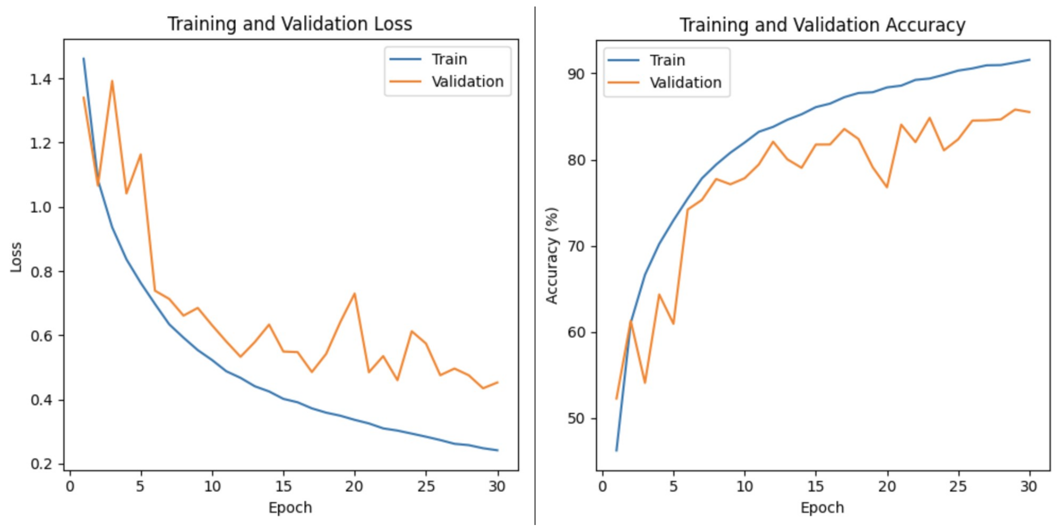
```python
from torch.optim.lr_scheduler import StepLR

# Hyperparameters
num_blocks = 3
K = 10
num_classes = 10
learning_rate = 0.001
batch_size = 100
num_epochs = 30
step_size = 30
gamma = 0.1
```

These are the hyperparameters that are used to train and evaluate the model:

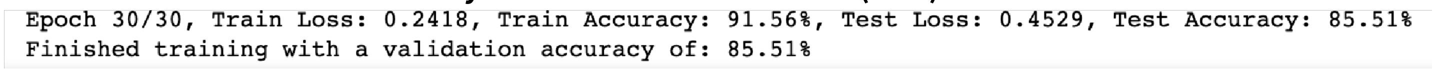
1. num_blocks: The number of residual blocks in the backbone of the model. Each residual block contains two convolutional layers (which is set to 3)
2. K: The number of output channels in the first convolutional layer of the backbone of the model (set to 10)

3. num_classes: The number of classes in the dataset (which is 10)
4. learning_rate: The learning rate used by the optimizer to update the model weights during training (set to 0.001)
5. batch_size: The number of examples in each batch during training (set to 100)
6. num_epochs: The number of times the entire training dataset is passed through the model during training (set to 30)
7. step_size: The number of epochs after which the learning rate is decreased (set to 30)
8. gamma: The factor by which the learning rate is decreased after each step (set to 0.1)



The training and validation accuracy/loss are recorded during the training process and stored in the train_accuracies, test_accuracies, train_losses, and test_losses lists. These lists are then used to plot the training and validation accuracy/loss graphs using the matplotlib library. The graph consists of two subplots: the left subplot shows the training and validation loss over epochs, while the right subplot shows the training and validation accuracy over epochs.

Task 5: Final model accuracy on CIFAR-10 Validation Set (15%)


```
Epoch 30/30, Train Loss: 0.2418, Train Accuracy: 91.56%, Test Loss: 0.4529, Test Accuracy: 85.51%
Finished training with a validation accuracy of: 85.51%
```

The final training accuracy for this model was > 90% and the final validation accuracy was > 85%