Assignment Title: Debugging an Artificial Neural Network with Gradio Front-End

---

Objective:

You are provided with a Python script implementing an Artificial Neural Network (ANN) for classifying handwritten digits from the MNIST dataset. The script includes a Gradio front-end that allows users to draw a digit and see the model's prediction. However, the script contains errors that prevent the ANN from functioning correctly.

Your task is to identify and fix the errors in the implementation to restore the correct functionality of the ANN. This assignment will test your understanding of neural networks, data processing, model training, and the integration of a front-end interface using Gradio.

Instructions:

1. **Setup:**
    - Ensure you have Python 3.x installed on your machine.
    - Install the necessary libraries:

      ```
      pip install torch torchvision gradio
      ```

    - Download the provided Python script: `mnist_ann_gradio.py`.
2. **Task:**
    - Run the script and observe its behavior.
    - Analyze the code to identify any errors or issues preventing the ANN from working correctly.
    - Fix the errors and ensure the model can accurately classify handwritten digits through the Gradio interface.
    - Document each error you found and explain how you resolved it.
3. **Deliverables:**
    - **Corrected Python Script:** Submit the fixed version of `mnist_ann_gradio.py`.
    - **Error Report:** A brief report detailing:
        - The errors you discovered.
        - The steps you took to identify them.
        - How you fixed each error.
        - Any challenges you faced during the debugging process.
4. **Time Allocation:**
    - You have **2 hours** to complete this assignment.
    - You may use the internet to research and assist in troubleshooting.
5. **Submission:**
    - Submit your corrected script and error report through the assignment portal by the end of the allocated time.

Grading Criteria:

- **Error Identification (40%):** Ability to accurately identify all the errors in the script.
- **Problem Resolution (40%):** Effectiveness of the solutions implemented to fix the errors.
- **Documentation (10%):** Clarity and thoroughness of your error report.
- **Code Quality (10%):** Cleanliness of code, use of best practices, and appropriate comments.

Provided Script: `mnist_ann_gradio.py`

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import gradio as gr
import numpy as np

# Define the neural network model
class MNISTModel(nn.Module):
    def __init__(self):
        super(MNISTModel, self).__init__()
        self.layer1 = nn.Linear(28 * 28, 128)
        self.relu = nn.ReLU()
        self.layer2 = nn.Linear(128, 10)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        out = self.layer1(x)
        out = self.relu(x)
        out = self.layer2(x)
        out = self.softmax(x)
        return out

# Load the dataset
transform = transforms.Compose([
    transforms.ToTensor()
```

```
])

train_dataset = datasets.MNIST(root='./data', train=True, download=True, transform=transform)

train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size=64, shuffle=True)

# Initialize the model, loss function and optimizer
model = MNISTModel()
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Train the model
for epoch in range(2):
    for images, labels in train_loader:
        images = images.reshape(-1, 28*28)
        outputs = model(images)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

# Define the Gradio interface function
def predict(image):
    image = transforms.functional.to_tensor(image).reshape(-1, 28*28)
    outputs = model(image)
    _, predicted = torch.max(outputs.data, 1)
    return predicted.item()

# Create Gradio interface
interface = gr.Interface(fn=predict, inputs="sketchpad", outputs="label", live=True, description="Draw a digit and the model will predict it.

# Launch the Gradio app
if __name__ == "__main__":
    interface.launch()
```

Additional Notes:

- Pay close attention to the data shapes and types flowing through the network.
- Consider the expected inputs and outputs at each stage of the network.
- Ensure that the loss function is appropriate for the task.
- Verify that the model's forward pass correctly applies each layer and activation function.
- Remember to test the corrected script to confirm that it works as intended before submission.

Good luck, and happy debugging!

Why This Assignment Requires Critical Thinking:

- **Understanding of Model Architecture:** Students need to comprehend how data flows through the network and be able to trace and correct the data transformations at each layer.
- **Knowledge of Appropriate Loss Functions:** Selecting the correct loss function for classification tasks is essential, and understanding the implications of using an incorrect one.
- **Data Processing Skills:** Properly handling and preprocessing input data is crucial, especially when integrating a front-end input like Gradio.
- **Debugging Skills:** Students must use their knowledge to interpret error messages (if any) and understand where the model's performance is deviating from expectations, which may not always produce explicit errors.
- **Integration of Multiple Components:** Combining PyTorch models with Gradio interfaces requires understanding of both back-end and front-end considerations.
