Explicit Function Decomposition with MoEs

Objective

Standard neural networks are "monolithic"—one giant block of weights processes every input. In this assignment, you will build a **Mixture of Experts (MoE)** architecture. You will train a "Gating Network" to divide a complex problem into sub-tasks and delegate them to specialized "Expert Networks."

The Problem

You are approximating a system that applies one of 4 distinct transformations (Invert, Flip Vertical, Flip Horizontal, Scanlines) to an image based on a hidden rule.

- A single small MLP struggles to learn all 4 disjoint tasks simultaneously (it suffers from "catastrophic interference").
- Your goal is to build a modular network where different parts of the neural net specialize in different transformations.

The Architecture

You must implement a custom PyTorch Module `class ExplicitMoE(nn.Module)` that contains:

1. **The Gate:** A "Router" network that takes the input image and outputs a probability distribution (softmax) over the 4 experts.
2. **The Experts:** Four separate, identical sub-networks (small MLPs).
   - **Hint:** You can use `torch.stack` to combine the outputs of all experts into a single tensor of shape `(batch_size, num_experts, output_size)`.
3. **The Forward Pass:**
   - The input goes into the Gate `weights`.
   - The input goes into *all* Experts `expert_outputs`.
   - The final output is the weighted sum: .

The Task

1. **Data Generation:**

   - Use **Fashion-MNIST (or another of your choosing)**. Generate targets using the provided `black_box`.

2. **Training:**

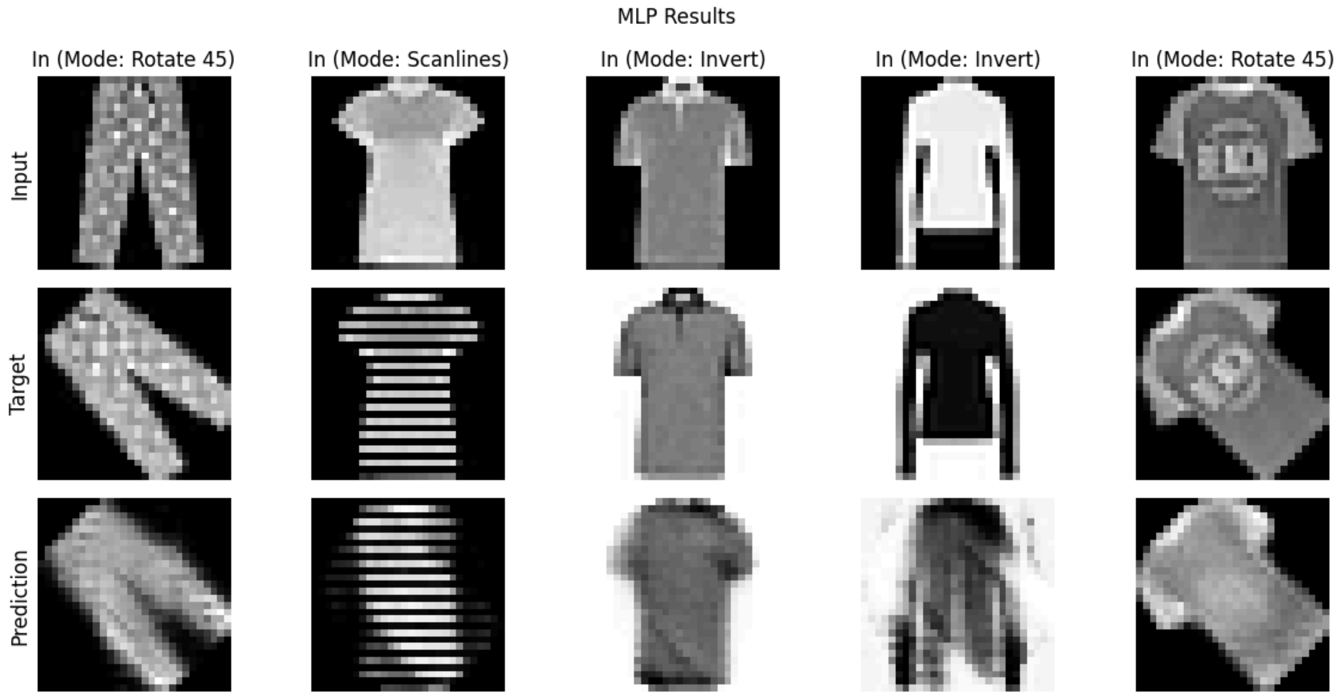
   - Train your MoE model using a loss function of your choice. Try experimenting with different ones to see what works best.
   - **Note:** The Gating network is *never told* which expert to use. It must learn to route traffic solely to minimize the global error.

3. **Interpretability Analysis:**
    - After training, extract the `gate_weights` for a batch of test images.
    - Determine if the experts actually specialized. (e.g., Does "Expert 1" handle all the inverted images, or is the load split randomly?)

Deliverables

1. **Python Code:** A clean script or Notebook containing your `Dataset`, `ExplicitMoE`, and `Training Loop`.
2. **Report (PDF):**
    - **Visualizations:** A 3x5 grid showing:
        - Top row = Original Input
        - Middle row = Oracle Ground Truth
        - Bottom row = Your Model's Prediction.
    - **Expert Specialization Plot:** A scatter plot or bar chart showing the average "Gating Confidence" for different types of images.
    - *Did your model experience "Mode Collapse" (where one expert does everything)? If so, discuss why.*
3. **Architecture Diagram:** A simple block diagram explaining how your Gate and Experts interact.

Visualization Example:


MLP Results
In (Mode: Rotate 45)
In (Mode: Scanlines)
In (Mode: Invert)
In (Mode: Invert)
In (Mode: Rotate 45)
Input
Target
Prediction