Implementation Strategy

This project implements a binary classification task on the MNIST dataset using an entropy-gradient based approach. The key steps in the implementation are:

1. **Data Preprocessing:** The MNIST dataset is loaded, normalized, and transformed into a binary classification task where even numbers are labeled as 0 and odd numbers as 1.
2. **Neural Network Model:** A deep neural network is built using TensorFlow/Keras with multiple layers, batch normalization, and dropout for better generalization.
3. **Entropy-Gradient Updates:** A custom function modifies the weights using entropy-based constraints after training.
4. **Z-Mapping and Constraints:** The model integrates a Z-mapping structure and applies the constraint zi+1−zi<δz_{i+1} - z_i < \delta as required.
5. **Optimization and Regularization:** The Adam optimizer with weight decay is used for stability, and dropout layers help in preventing overfitting.

Incorporation of Z-Mapping, Dual-Weight Structure, and Constraints

- **Z-Mapping:** The transformation of weights is performed through entropy-gradient updates.
- **Dual-Weight Structure:** The model retains weight transformations before applying entropy constraints.
- **Constraint:** zi+1−zi<δz_{i+1} - z_i < \delta**:** This constraint ensures weight updates remain within a defined range, limiting drastic changes in adjacent layers.

Differences Between Classical Loss-Based and Entropy-Gradient Updates

Classical loss-based updates focus on minimizing cross-entropy loss using standard backpropagation, while entropy-gradient updates modify weights based on entropy constraints rather than direct loss minimization. In classical updates, weight modifications occur through gradient descent on loss functions, whereas in entropy-gradient updates, weights are adjusted to maintain entropy balance. The classical approach can sometimes lead to instability in deep networks, while entropy-based updates introduce a stabilizing effect by enforcing structured weight adjustments.

Observations on Training Dynamics and Classification Accuracy

- The addition of batch normalization and dropout has helped stabilize training and prevent overfitting.
- Entropy-gradient updates introduce a regularization-like effect, smoothing weight updates.
- The final test accuracy is improved over naive training but can still be fine-tuned with further constraints.

Execution Instructions

Requirements

Ensure you have the following dependencies installed:

```
pip install tensorflow numpy
```

Running the Code

To execute the training script, run the following command:

```
python mnist_entropy_gradient.py
```

This will train the model, apply entropy constraints, and evaluate its performance on the test set.