Report: Incremental Learning with Elastic Weight Consolidation (EWC) on MNIST Dataset

Introduction

This report outlines the implementation and results of an incremental learning experiment using the Elastic Weight Consolidation (EWC) technique on the MNIST dataset. The goal of this experiment is to train a neural network on two sequential tasks (Task 1: digits 0-4, Task 2: digits 5-9) while mitigating catastrophic forgetting using EWC.

Methodology

1. Dataset Preparation
The MNIST dataset was split into two tasks:
- **Task 1**: Digits 0-4
- **Task 2**: Digits 5-9

The dataset was normalized and transformed using `torchvision.transforms` to prepare it for training.

2. Model Architecture
A simple feedforward neural network (`SimpleNN`) was used for this experiment. The architecture consists of:
- **Input Layer**: 784 neurons (28x28 pixels)
- **Hidden Layer**: 128 neurons with ReLU activation
- **Output Layer**: 10 neurons (for 10 classes)

3. Training Process
The training process was divided into two phases:
1. **Training on Task 1**: The model was trained on Task 1 (digits 0-4) for 5 epochs.
2. **Training on Task 2 with EWC**: The model was then trained on Task 2 (digits 5-9) while applying EWC to prevent forgetting of Task 1.

4. Elastic Weight Consolidation (EWC)
EWC was implemented to mitigate catastrophic forgetting. The key steps involved:

- **Computing Fisher Information**: The Fisher Information matrix was computed for Task 1 to identify the importance of each parameter.
- **EWC Loss**: During training on Task 2, an additional regularization term (EWC loss) was added to the loss function to penalize changes to important parameters learned in Task 1.

5. Evaluation

The model's performance was evaluated on both tasks after training on Task 2 with EWC.

Results

1. Training on Task 1

- The model achieved high accuracy on Task 1 after 5 epochs:
  - **Epoch 1**: Loss = 258.7517, Accuracy = 0.8380
  - **Epoch 5**: Loss = 42.1500, Accuracy = 0.9617

2. Training on Task 2 with EWC

- The model was trained on Task 2 with EWC for 5 epochs. The EWC loss was added to the task loss to prevent forgetting Task 1.
  - **Epoch 1**: Total Loss = 4028.0728, Task Loss = 2214.9615, EWC Loss = 0.0018, Accuracy = 0.0000
  - **Epoch 5**: Total Loss = 35411.5105, Task Loss = 471.7372, EWC Loss = 0.0349, Accuracy = 0.4102

3. Evaluation of Both Tasks

- **Task 1 Accuracy**: 0.7484
- **Task 2 Accuracy**: 0.4618

Discussion

1. Performance on Task 1

The model achieved a high accuracy of 96.17% on Task 1 after training. This indicates that the model successfully learned to classify digits 0-4.

2. Performance on Task 2 with EWC

The model's accuracy on Task 2 improved over the epochs, reaching 41.02% by the 5th epoch. However, the accuracy on Task 1 dropped to 74.84%, indicating some degree of forgetting despite the use of EWC.

3. Effectiveness of EWC

The EWC technique helped in retaining some knowledge of Task 1 while learning Task 2, as evidenced by the 74.84% accuracy on Task 1 after training on Task 2. However, the drop in accuracy suggests that the model still experienced some forgetting, possibly due to the high complexity of the tasks or the need for further tuning of the EWC hyperparameters (e.g., `ewc_lambda`).

4. Challenges and Limitations

- **Hyperparameter Tuning**: The choice of `ewc_lambda` (set to 1,000,000) may have been too high, leading to excessive regularization and slower learning on Task 2.
- **Model Capacity**: The simple architecture of the model may have limited its ability to retain knowledge from both tasks simultaneously.
- **Dataset Complexity**: The MNIST dataset, while simple, may not fully capture the challenges of catastrophic forgetting in more complex scenarios.

Conclusion
The experiment demonstrated the effectiveness of EWC in mitigating catastrophic forgetting to some extent. However, there is room for improvement in terms of hyperparameter tuning, model architecture, and possibly exploring more advanced techniques for incremental learning. Future work could involve experimenting with different values of `ewc_lambda`, using more complex models, or exploring other regularization techniques to further reduce forgetting.

Code Summary
The code provided in the notebook includes:
- Data preparation and splitting into Task 1 and Task 2.
- Definition of the `SimpleNN` model.
- Training loops for Task 1 and Task 2 with EWC.
- Computation of Fisher Information and implementation of EWC loss.

- Evaluation of the model on both tasks after training.

This experiment serves as a foundational step in understanding and implementing incremental learning techniques, with potential applications in real-world scenarios where models need to learn continuously from new data without forgetting previously learned knowledge.

Code work here :

https://colab.research.google.com/drive/1K3e7xQuFbl74pRKkllYtVQfHaknpz6na?usp=sharing