MNIST database classification

Data Preprocessing

- Data Loading
    - The dataset utilized for training the neural network model is the MNIST dataset, a widely-used resource for handwritten digit recognition tasks.
    - The MNIST dataset is directly loaded using torchvision's dataset utilities, ensuring a standardized and reliable source of data.
    - The dataset is split into training and validation sets, with the training set used to adjust the model's weights and the validation set used to evaluate the model's performance and prevent overfitting.
- Data Transformation and Augmentation
    - Data Augmentation: For the training data, random affine transformations and color jitter are applied. These augmentations introduce variability in the dataset, helping the model become more robust and less sensitive to small changes in input.
    - Normalization: Both the training and validation datasets are normalized using a mean and standard deviation of 0.5. This step brings the pixel values of the images into a range that is more suitable for training neural networks, aiding in faster convergence.
    - ToTensor: The images are converted to PyTorch tensors, making them compatible with the PyTorch framework.
- Data Loading and Batching
    - The transformed datasets are wrapped in DataLoaders, which automate the batching of the data and provide functionality for shuffling the training data.
    - A batch size of 96 is used, striking a balance between computational efficiency and the granularity of the gradient updates
- Exploratory Data Analysis (EDA)
    - While explicit EDA is not shown in the provided notebook, visualizing a batch of images from the training set can be considered a form of EDA. This helps in verifying that the data loading and transformations are working as expected.

- Handling Missing Values
    - The MNIST dataset provided by torchvision is a cleaned and well-maintained dataset, and as such, missing values are not a concern in this specific context.
    - However, in a more general setting, checking for and handling missing values would be a crucial step to ensure the integrity and reliability of the training process.

Model Architecture

- Layers
    - Convolutional Layers: There are three convolutional layers, each followed by batch normalization, a ReLU activation function, and max pooling.
        - First Convolutional Layer: 1 input channel, 4 output channels, 3x3 kernel, padding of 1, followed by 2x2 max pooling.
        - Second Convolutional Layer: 4 input channels, 16 output channels, 3x3 kernel, padding of 1, followed by 2x2 max pooling.
        - Third Convolutional Layer: 16 input channels, 64 output channels, 3x3 kernel, padding of 1, followed by 3x3 max pooling with a stride of 2.
    - Fully Connected Layers: A series of linear layers interspersed with dropout, batch normalization, and ReLU activation functions.
        - A dropout layer with a probability of 0.5.
        - A linear layer that flattens the output from the previous layer and reduces the dimension to 128.
        - A linear layer that reduces the dimension from 128 to 32.
        - A final linear layer that outputs 10 values corresponding to the 10 classes of the MNIST dataset.
- Activation Functions:
    - ReLU (Rectified Linear Unit) is used after each convolutional and fully connected layer, except for the final layer.
- Output:
    - The final output is passed through a log softmax function to convert the raw scores into probabilities.

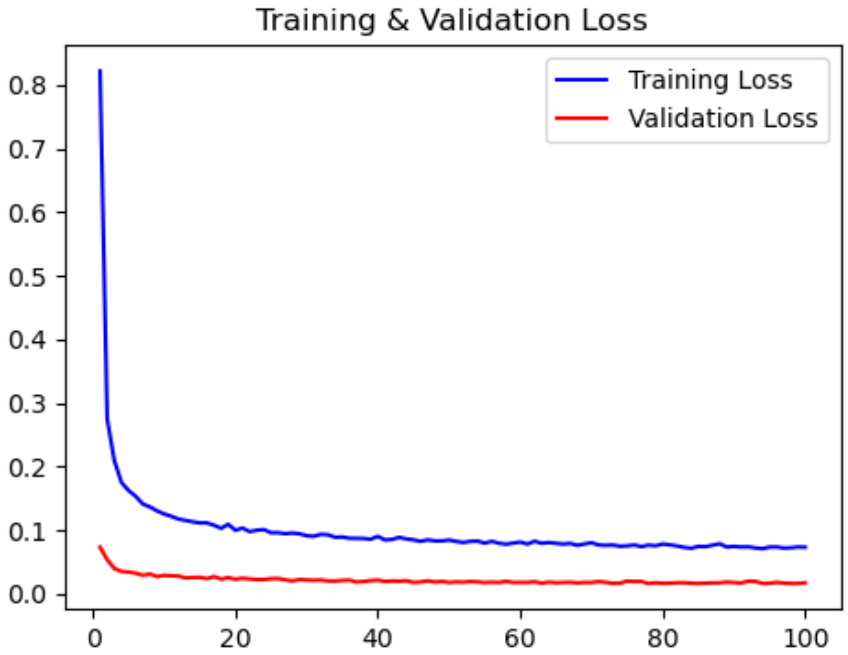
- Regularization and Batch Normalization:
    - Dropout with a probability of 0.5 is used to prevent overfitting.
    - Batch normalization is applied after each convolutional and fully connected layer to normalize the activations, which helps in speeding up training and achieving better performance.
- Explanation
    - Convolutional Layers: These layers are capable of capturing hierarchical patterns in the image data. The increasing number of filters in deeper layers allows the network to detect more complex features.
    - Pooling Layers: Max pooling is used to reduce the spatial dimensions, leading to a reduction in the number of parameters and computational requirements.
    - Fully Connected Layers: These layers are used to learn a non-linear mapping from the extracted features to the output classes.
    - Dropout: Applied to prevent overfitting by randomly dropping units during training.
    - Batch Normalization: Helps in stabilizing and accelerating the training process.
    - ReLU Activation: Introduces non-linearity, allowing the network to solve complex problems.
    - Log Softmax: Used to convert the final layer's output into probability distributions over the classes.

Choice of Hyperparameters

- Batch Size: A larger batch size can lead to faster training, but it can also require more memory, and the model might not generalize as well. A batch size of 96 is a balanced choice.
- Learning Rate: The default learning rate of the Adam optimizer is usually a good starting point, but it might be beneficial to experiment with different learning rates or learning rate scheduling to potentially achieve better performance.
- Optimizer: The Adam optimizer is known for being robust and efficient, making it a safe choice for many training tasks.
- Loss Function: Cross-Entropy Loss is suitable for multi-class classification problems like MNIST.

- Number of Epochs: Training for more epochs can lead to a better model, but it also increases the risk of overfitting, especially without proper regularization or early stopping.

Learning Curves



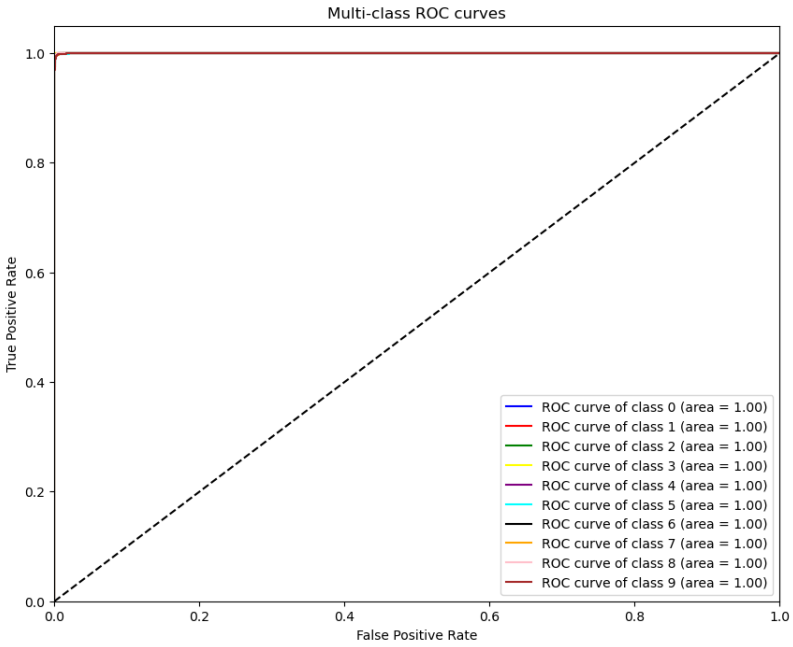
- X-Axis (Epochs): Represents the number of passes through the entire training dataset. As the number of epochs increases, the model has more opportunities to learn from the data.
- Y-Axis (Loss): Represents the model's error or loss. Lower values indicate better performance.
- Training Loss: Typically decreases over time as the model learns from the training data.
- Validation Loss: Shows how well the model is performing on unseen data.
- Convergence: Both training and validation loss decrease and stabilize over time.

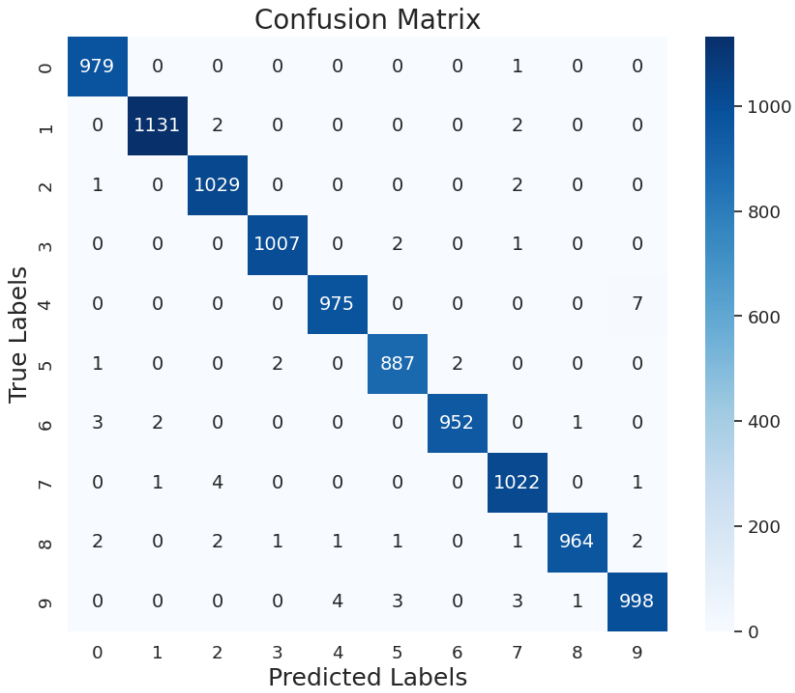
Evaluation metrics on the test set

- Precision, Recall, and F1 Score: These metrics provide a balanced view of the model's performance, especially in the context of imbalanced datasets.
- Confusion Matrix: Helps to identify the classes that are being misclassified and the types of misclassifications that are occurring.
- AUC-ROC: Useful for understanding the model's ability to distinguish between classes, regardless of the threshold.
- Log-Loss: Provides a measure of accuracy for a classifier, penalizing false classifications more harshly when they are made with high confidence.

Result

Precision: 0.9944
Recall: 0.9944
F1 Score: 0.9944
AUC-ROC: 1.0000
Log Loss: 0.0176





Summary

- Neural Network Architecture:
    - The model is a Convolutional Neural Network (CNN) with three convolutional layers followed by batch normalization, ReLU activations, and max pooling, and concluding with fully connected layers. Dropout and batch normalization are utilized for regularization.
- Hyperparameters:
    - Batch Size: 96, balancing computation efficiency and learning stability.
    - Optimizer: Adam, chosen for its adaptive learning rate properties.
    - Learning Rate: Default value of 0.001.
    - Loss Function: Cross-Entropy Loss, suitable for classification tasks.
    - Epochs: 100, providing sufficient time for the network to learn.
- Learning Curves :
    - Training Loss: Expected to decrease, indicating learning progress.
    - Validation Loss: Should decrease, but an increase may indicate overfitting
- The architecture and hyperparameters are well-chosen for the image classification task, and monitoring learning curves is crucial for ensuring

effective training and identifying areas for improvement. Future work could involve hyperparameter tuning and experimenting with different architectures.