1. Problem Statement

Braille is a tactile writing system used by visually impaired individuals. This project aims to develop a deep learning model capable of recognizing Braille characters from images and translating them into readable text. The ultimate goal is to create an accessible tool that assists in the conversion of Braille into standard text using computer vision techniques.

2. Data Preprocessing

Dataset: The dataset consists of 26 Braille characters, each augmented with three variations (rotation, brightness, and shift). Each character has 20 different images for each augmentation type, making a total of 1,560 images (26 characters * 3 augmentations * 20 images).

Preprocessing Steps:

- Converted all images to grayscale to standardize the input.
- Resized all images to 28x28 pixels to ensure uniform input dimensions.
- Augmentation was applied to enhance the robustness of the model.
- Data was split into 80% training and 20% validation to ensure proper evaluation.

3. Machine Learning Model

The initial plan was to use a Convolutional Neural Network (CNN) for classification, and this choice remains unchanged due to its effectiveness in image recognition tasks.

- Convolutional Layers:
    - 3 convolutional layers (32, 64, 128 filters) with ReLU activation and 3x3 kernels.
    - MaxPooling (2x2) applied after each convolutional layer.
    - Output layer with 26 neurons (corresponding to Braille letters)
- Framework: PyTorch
- learning rate = 0.001
- Loss Function: Cross-Entropy Loss (suitable for multi-class classification)
- Batch Size: 32
- Number of Epochs: 20
- Performance monitored using accuracy, precision, and recall.

4. Preliminary Results

Evaluation Metrics & Performance

- Training Accuracy: ~92.5%
- Validation Accuracy: ~87.3%
- Loss: Decreased steadily over epochs, showing good convergence.
- Confusion Matrix Analysis: Some characters like ‘o’ and ‘q’ were misclassified, likely due to similar dot patterns.

5. Next Steps

Pros & Cons of Current Approach

Pros:

- The CNN model effectively extracts Braille features and achieves decent accuracy.
- Augmented dataset improved model robustness.
- Model architecture is simple and computationally efficient.

Cons:

- Some characters are difficult to distinguish, leading to misclassification.
- Model performance can still be improved with additional training data.

Future Work

- Fine-tune hyperparameters (e.g., learning rate, batch size) to optimize performance.
- Research pre-trained models like MobileNetV2 for improved feature extraction.
- Implement multi-character translation (whole words)
- Integrate Text-to-Speech (TTS) for enhanced accessibility.