LeNet-5 Architecture and Bias Analysis

1. Introduction

LeNet-5, introduced by Yann LeCun et al. in 1998, is one of the earliest Convolutional Neural Networks (CNNs) and has been foundational in advancing the field of deep learning for image recognition. Despite its simplicity and small number of parameters, it has achieved significant success in tasks like handwritten digit classification. In this report, we examine the architecture of LeNet-5, its performance on different datasets, potential sources of bias, and ways to mitigate these biases.

2. LeNet-5 Architecture

2.1 Layers Overview

LeNet-5 consists of eight layers, including convolutional layers, subsampling layers, and fully connected layers. Below is a breakdown of each layer:

Input Layer

- **Dimensions:** 28×28 Grayscale Image
- **Description:** The input layer takes an image of size 28×28 pixels as input, where each pixel is a grayscale value.

Convolutional Layer 1 (C1)

- **Filters:** 6 filters of size 5×5
- **Stride:** 1
- **Activation Function:** Tanh (commonly replaced by ReLU in modern implementations)
- **Output Size:** 28×28×6
- **Description:** The first convolutional layer applies six filters to the input image to extract basic features such as edges and textures.

Subsampling (Average Pooling) Layer 1 (S2)

- **Pooling Size:** 2×2
- **Stride:** 2
- **Activation Function:** Tanh
- **Output Size:** 14×14×6
- **Description:** This pooling layer reduces the spatial dimensions by a factor of 2, keeping the most important features while discarding less relevant details.

Convolutional Layer 2 (C3)

- **Filters:** 16 filters of size 5×5
- **Stride:** 1
- **Activation Function:** Tanh
- **Output Size:** 10×10×16
- **Description:** The second convolutional layer extracts higher-level features from the output of the previous layer.

Subsampling (Average Pooling) Layer 2 (S4)

- **Pooling Size:** 2×2
- **Stride:** 2
- **Activation Function:** Tanh
- **Output Size:** 5×5×16
- **Description:** The second pooling layer further reduces the spatial dimensions.

Fully Connected Layer (F5)

- **Neurons:** 120
- **Activation Function:** Tanh
- **Description:** This fully connected layer consolidates information from the previous layers and prepares it for the final output.

Fully Connected Layer (F6)

- **Neurons:** 84
- **Activation Function:** Tanh
- **Description:** The second fully connected layer processes the features from the previous layer.

Output Layer

- **Neurons:** 10 (one for each digit class)
- **Activation Function:** Softmax
- **Description:** The output layer produces the final classification results, where each neuron corresponds to a digit from 0 to 9.

2.2 Key Features of LeNet-5

- **Convolutional Layers for Feature Extraction:** LeNet-5 utilizes convolutional layers to automatically extract hierarchical features from the input images.

- **Average Pooling:** Unlike modern CNNs that use max pooling, LeNet-5 uses average pooling to downsample the feature maps.
- **Small Number of Parameters:** With approximately 60,000 parameters, LeNet-5 is considered lightweight compared to more modern CNNs.
- **Optimized for Small Datasets:** LeNet-5 is well-suited for small datasets like MNIST, where it performs efficiently despite its relatively small architecture.

3. Bias in LeNet-5

Bias in machine learning models can stem from various sources. Below are some of the potential biases that may affect LeNet-5's performance:

3.1 Dataset Bias

- **Sampling Bias:** If trained on an imbalanced dataset (e.g., more samples of certain digits than others), the model might perform better on the more frequent classes while struggling with rare ones.
- **Handwriting Style Bias:** If the model is trained on a dataset with a limited diversity of handwriting styles, it may struggle with generalization when presented with new, unseen handwriting styles.

3.2 Model Architectural Bias

- **Limited Representation Power:** With only around 60,000 parameters, LeNet-5 may not be expressive enough to capture complex patterns in large or diverse datasets.
- **Use of Average Pooling:** LeNet-5 uses average pooling, which retains background noise and can reduce sensitivity to important, high-contrast features.

3.3 Training Data Bias

- **Class Distribution Bias:** If the training data has unequal class distributions, the model may favor the more common classes, leading to poor performance on underrepresented classes.
- **Handwriting Style Bias:** If the training data is not diverse enough in terms of handwriting styles, LeNet-5 may perform poorly on images from other sources or demographic groups.

3.4 Robustness and Generalization Bias

- **Rotation, Scaling, and Perspective Bias:** LeNet-5 was designed to work on clean, centered images. It may not perform well when images are rotated, resized, or distorted.
- **Adversarial Bias:** LeNet-5, like most neural networks, is vulnerable to adversarial attacks. Small perturbations in the input images can lead to misclassifications.

4. Mitigating Bias in LeNet-5

4.1 Data Augmentation

- **Description:** Data augmentation techniques, such as rotating, scaling, and shifting the images, can help improve the model's robustness and reduce its sensitivity to variations in the data.

4.2 Train on Diverse Data

- **Description:** To improve the model's generalization, it can be trained on datasets beyond MNIST, such as EMNIST, USPS, or other datasets that introduce more variability in terms of handwriting styles or object categories.

4.3 Regularization & Normalization

- **Description:** Techniques like batch normalization and dropout can help prevent overfitting and reduce bias caused by noisy or imbalanced data.

4.4 Use of More Advanced Models

- **Description:** Modern CNN architectures, such as ResNet or VGG, are better suited for handling complex datasets and mitigating biases due to their larger capacity and more sophisticated structures.

5. Datasets Used for Implementation

Architecture:-

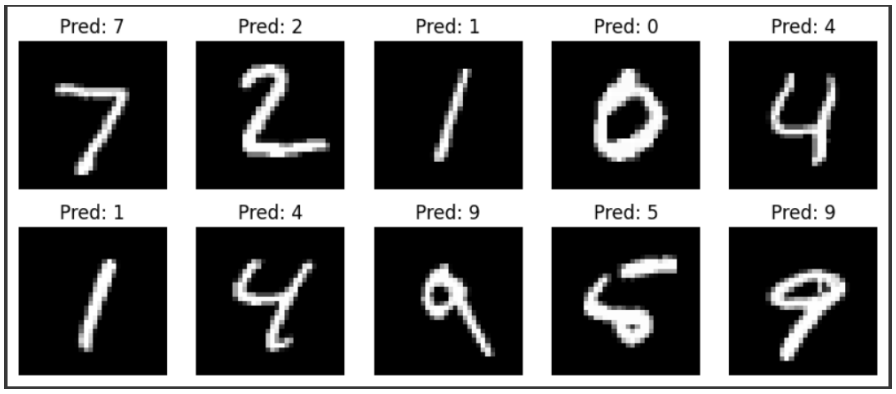
```
def LeNet5():
    model = models.Sequential([
        layers.Conv2D(6, (5,5), activation='tanh', padding='same', input_shape=(32,32,1)),
        layers.AvgPool2D((2,2), strides=2),
        layers.Conv2D(16, (5,5), activation='tanh'),
        layers.AvgPool2D((2,2), strides=2),
        layers.Conv2D(120, (5,5), activation='tanh'),
        layers.Flatten(),
        layers.Dense(84, activation='tanh'),
        layers.Dense(10, activation='softmax')  #output layer(0-9 digits)
    ])
```

5.1 Dataset 1: MNIST - Handwritten Digits

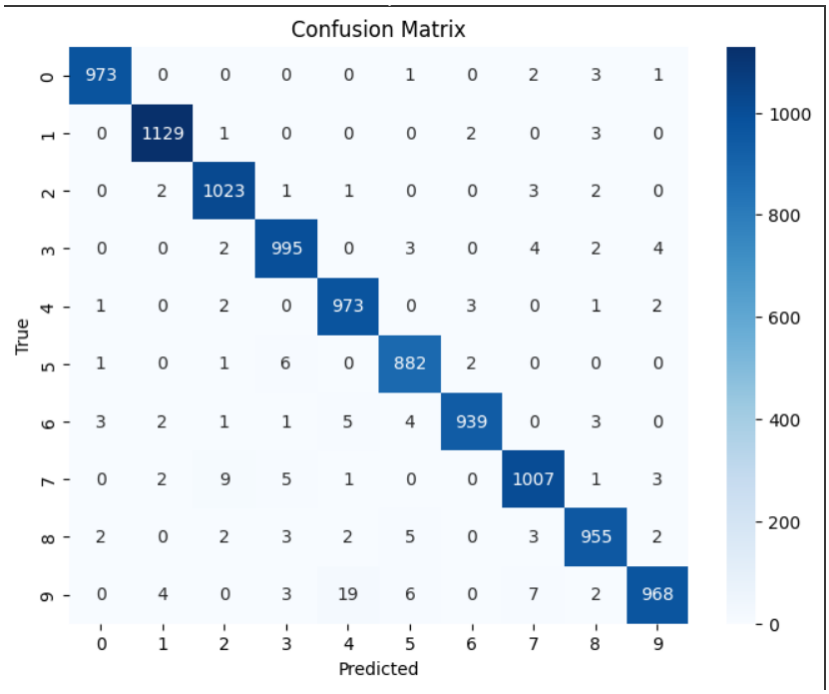
- **Description:** The MNIST dataset is a widely used benchmark for handwritten digit classification. It contains 60,000 training and 10,000 test grayscale images of handwritten digits, each of size 28×28 pixels.

- **Observation:** LeNet-5 performs well on MNIST due to its simplicity and the well-centered, clean nature of the dataset.

Test Prediction:-



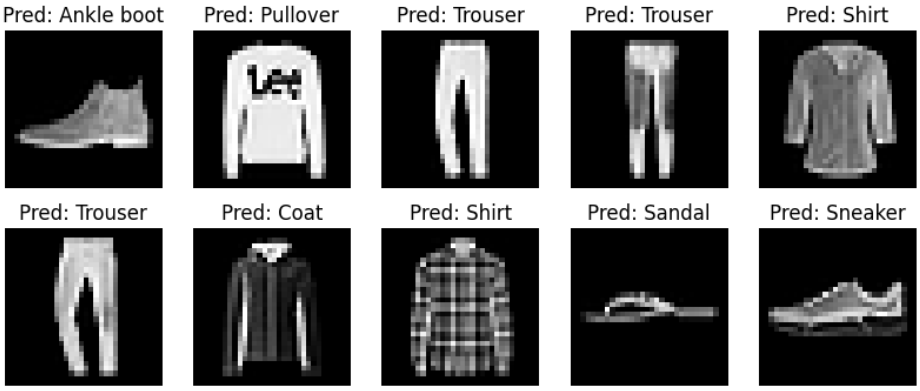
Confusion Matrix:-



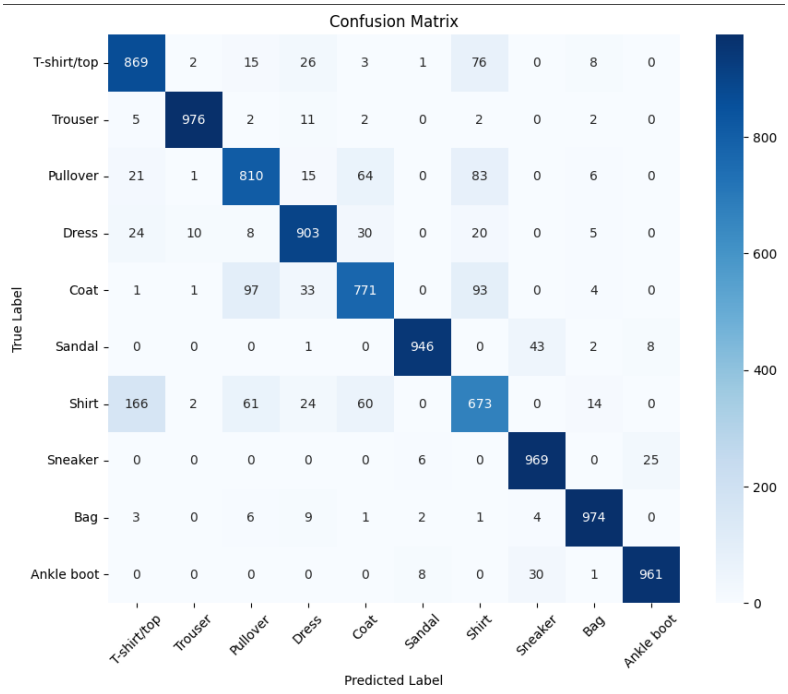
5.2 Dataset 2: Fashion MNIST

- **Description:** The Fashion MNIST dataset is a more challenging dataset than MNIST, containing 60,000 training and 10,000 test grayscale images of 10 fashion categories, such as T-shirts, trousers, and shoes.
- **Observation:** LeNet-5 performs slightly worse on Fashion MNIST due to the more complex visual patterns compared to handwritten digits.

Test Prediction:-



Confusion Matrix:-

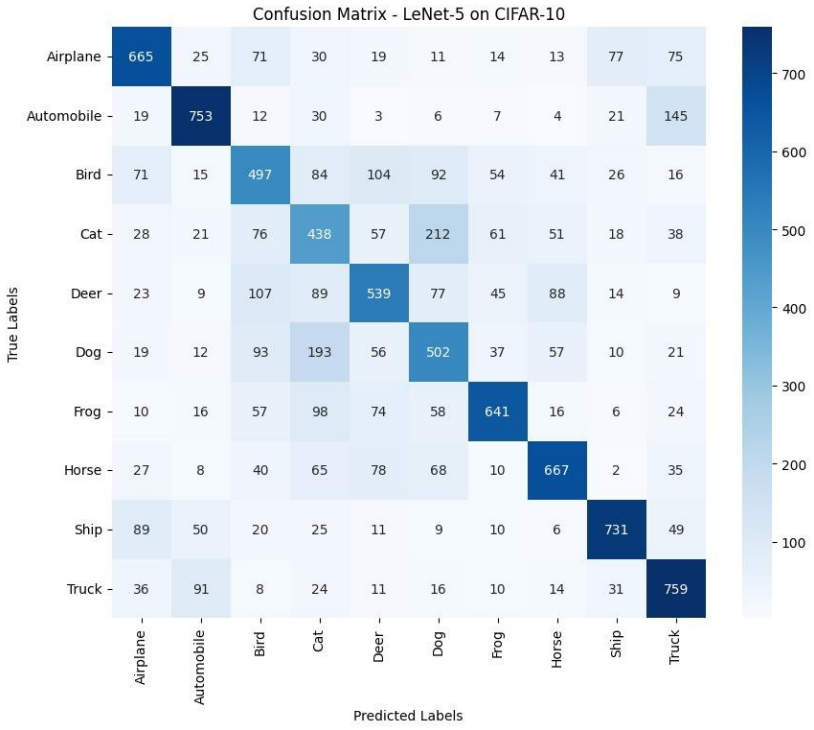


-

5.3 Dataset 3: CIFAR-10

- **Description:** The CIFAR-10 dataset consists of 60,000 color images categorized into 10 classes, such as airplanes, automobiles, and cats, each with a resolution of 32×32 pixels.
- **Observation:** LeNet-5 struggles more with CIFAR-10 due to its limited architecture, and it is less suited for handling color images and the variety of classes in the dataset.

Confusion Matrix:-



6. Test Predictions and Confusion Matrix

For each dataset, test predictions were made, and confusion matrices were generated to evaluate the model's performance. These matrices help identify misclassified examples and analyze the distribution of correct and incorrect predictions across different classes.

7. Conclusion

While LeNet-5 remains a foundational model in the history of deep learning, its simplicity limits its ability to handle more complex datasets and generalize well across diverse scenarios. Bias in LeNet-5 can arise from various sources, including data, architecture, and robustness issues. However, through techniques like data augmentation, training on diverse datasets, and the use of regularization methods, these biases can be mitigated to some extent. For more challenging tasks, modern CNN architectures with greater capacity are recommended for better performance and bias handling.