Machine Learning and Data Mining 2

Comparative Analysis of Image Classification Algorithms on MNIST Dataset

Le Tuan Huy (BI12-195)
Le Minh Hoang (BI12-167)
Nguyen Minh Hoang (BI12-172)
Nguyen The Hoang (BI12-171)
Nguyen Vu Viet Hoang (BI12-173)
Le Bui Huy Hoang (BI12-174)

University of Science and Technology
of Hanoi

Jan 2024

Contents

1 Introduction **2**

1.1 The purpose of our study . . . 2

1.2 What is MNIST dataset? . . . 3

2 Methodology **4**

2.1 Convolutional Neural Network (CNN) . . . 4

2.1.1 What is CNN? . . . 4

2.1.2 Key Components of CNN . . . 4

2.2 LeNet-5 . . . 6

2.2.1 What is LeNet? . . . 6

2.2.2 LeNet-5 Architecture . . . 7

2.3 AlexNet . . . 8

2.3.1 What is AlexNet? . . . 8

2.3.2 AlexNet Architecture . . . 8

2.4 Training and Testing . . . 9

2.4.1 Preparing Dataset & Parameters . . . 9

2.4.2 Training Process . . . 10

2.4.3 Testing Process . . . 11

3 Conclusion **17**

Appendices **18**

.1 Appendix . . . 18

.1.1 Code Implementation . . . 18

.1.2 Additional Material . . . 18

References **19**

Chapter 1

Introduction

1.1 The purpose of our study

Our study focuses on studying and evaluating two widely recognized convolutional neural networks(CNN) LeNet-5 and AlexNet in the context of image classification using the MNIST dataset. The motivation behind this study stems from the need to understand and implement these two architectures. LeNet-5 and AlexNet represent milestones in the evolution of CNN. We aim to draw insights into their respective strengths and weaknesses in the realm of digit recognition. We have simplified some complex concepts to enhance accessibility, making this report valuable not only to experts in the field but also to those seeking an introduction to the application of deep learning in image classification.

1.2 What is MNIST dataset?

The MNIST dataset was introduced in the late 1990s by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. Originally designed for training and testing algorithms related to image processing and pattern recognition, MNIST quickly gained popularity as a standard dataset for evaluating the performance of machine learning models, particularly in the context of digit recognition.

- MNIST Dataset Specifications:
  - Number of Classes: 10 (Digits 0 through 9)
  - Image Size: 28x28 pixels
  - Training/Testing Set: 60,000 samples / 10,000 samples
  - Training samples/class (0: 5923), (1: 6742), (2: 5958), (3: 6131), (4: 5842), (5: 5421), (6: 5918), (7: 6265), (8: 5851), (9: 5949)
  - Testing samples/class: (0: 980), (1: 1135), (2: 1032), (3: 1010), (4: 982), (5: 892), (6: 958), (7: 1028), (8: 974), (9: 1009)

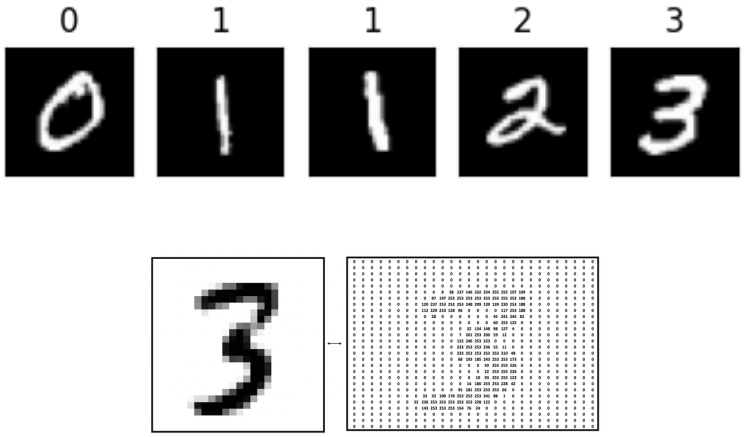
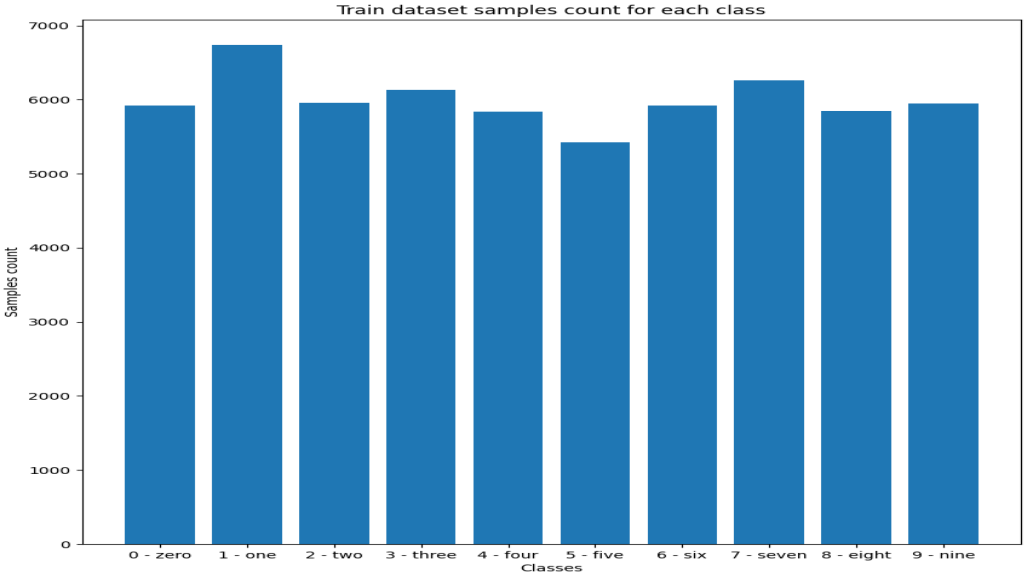


Figure 1.1: Image sample from MNIST converted to pixel value grid



Chapter 2

Methodology

2.1 Convolutional Neural Network (CNN)

2.1.1 What is CNN?

Convolutional Neural Networks (CNN) are a specialized type of neural network architecture designed for processing and analyzing grid-like data, particularly images and video. They have proven highly effective in various computer vision tasks, such as image recognition, object detection, and image segmentation.

2.1.2 Key Components of CNN

- Convolutional Layers: CNN leverage convolutional layers to automatically learn hierarchical representations of features within the input data. Convolutional operations involve the application of filters or kernels to the input, enabling the network to capture local patterns.

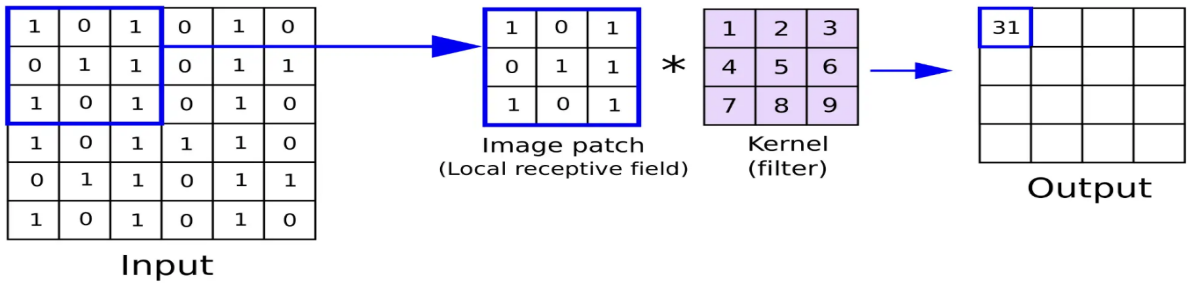


Figure 2.1: simple filter operation

  - The output size of a convolutional layer is given by:

$$\text{Output size(H or W)} = \left\lfloor \frac{\text{Input size(H or W)} - \text{Kernel size(f)} + 2 \times \text{Padding(p)}}{\text{Stride(s)}} \right\rfloor + 1$$

- Activation Functions: Activation functions introduce non-linearity to the model, allowing it to learn complex relationships. These functions have total weighted sum of all nodes from the previous layer as input and the output aare the nodes of the next layer. Common activation functions include:

- Sigmoid Function: $\sigma(x) = \frac{1}{1+e^{-x}}$ where x is the input to the function and e is the mathematical constant of 2.718
- ReLU(Rectified Linear Unit): $\text{ReLU}(x) = \max(0,\, x)$ where x is the input to the function and max is the mathematical function to filter node with value < 0
- Softmax Function (for multi-class classification):

$$\text{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}, \quad i = 1, 2, \ldots, n$$

where z is the final layer node, i and j is the node index number, n is the number of nodes at the final layer.

- Pooling Layers: Pooling layers reduce the spatial dimensions of the input, decreasing the computational load. Common pooling operations include:
  - Average Pooling:

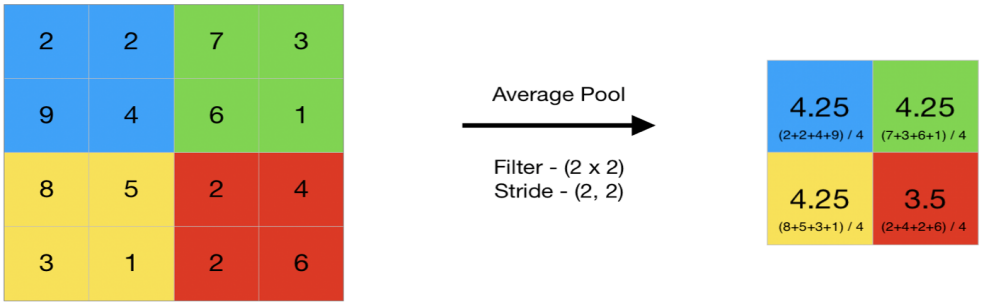


Figure 2.2: simple avg pooling operation

  - Max Pooling:

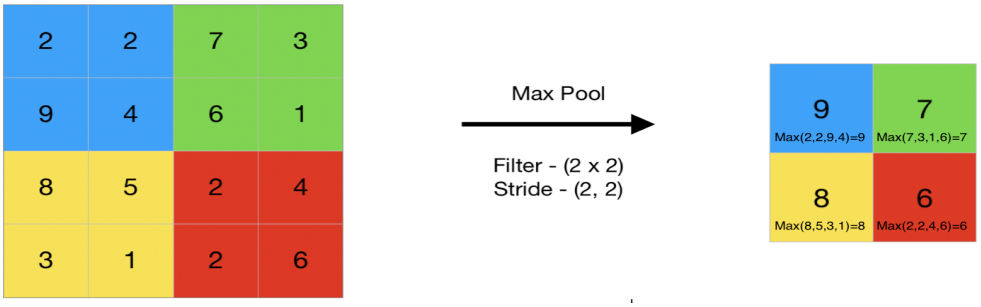


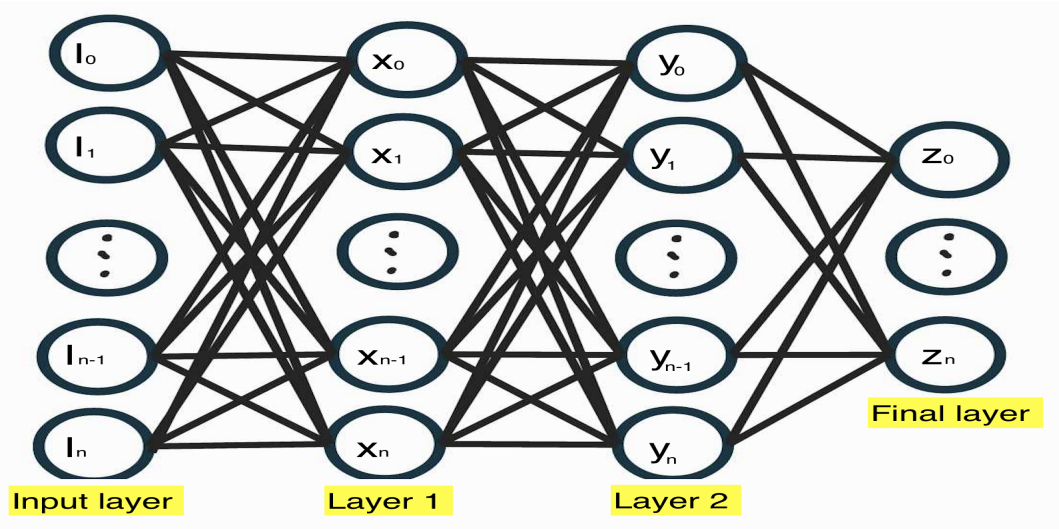
Figure 2.3: simple max pooling operation

  - The output size of an max/average pooling layer is given by:

$$\text{Output size(H or W)} = \left\lfloor \frac{\text{Input size} - \text{Pooling size(f)}}{\text{Stride(s)}} \right\rfloor + 1$$

- Flattening: Flattening transforms the multi-dimensional output of the convolutional and pooling layers into a one-dimensional vector. This vector serves as the input for the fully connected layers.

- Fully Connected Layers: Fully connected layers process the flattened input, learning complex relationships and making final predictions. The output size of these layers depends on the task. The final layer(output layer) consist of nodes corresponding to classes, each node of this layer will go through softmax function convert vector of raw scores (also known as logits) into a probability distribution over multiple classes.



2.2 LeNet-5

2.2.1 What is LeNet?

The LeNet-5 architecture, proposed by Yann LeCun and his collaborators in 1998 in the research paper Gradient-Based Learning Applied to Document Recognition, is one of the earliest models and a pioneering convolutional neural network (CNN) designed for handwritten digit recognition. It played a crucial role in the development of deep learning and convolutional networks.

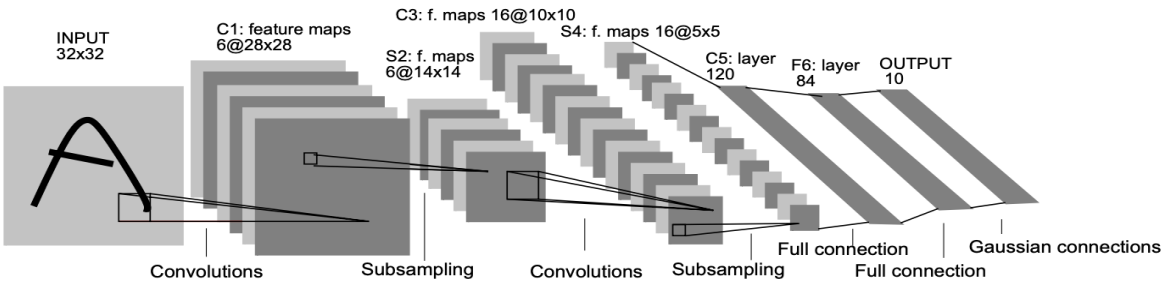


Figure 2.4: Original LeNet-5 architecture
[1]

2.2.2 LeNet-5 Architecture

- The Figure bellow is the architecture of LeNet-5:

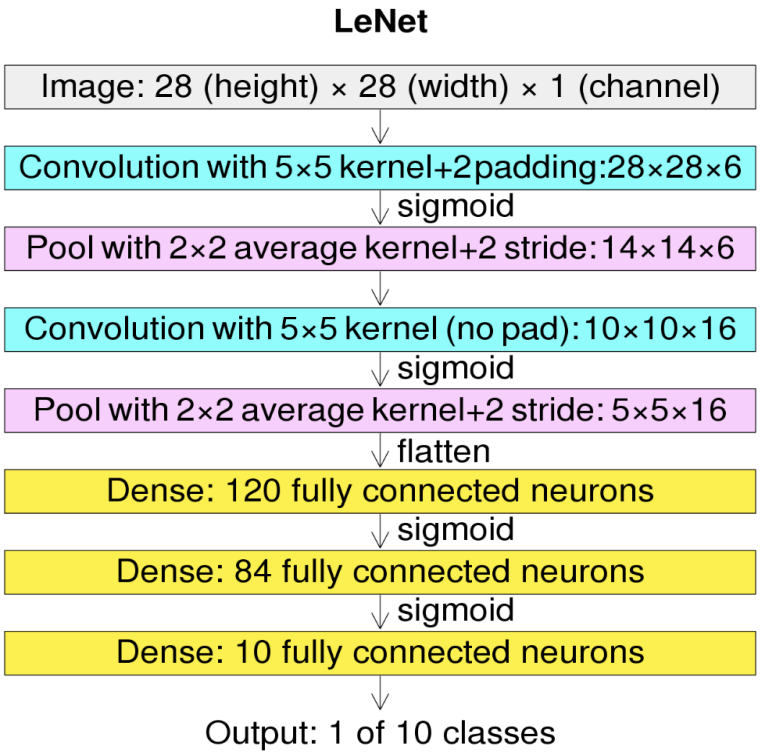


Figure 2.5: Detail LeNet-5 architecture

- The Figure bellow is the Data flow of LeNet-5:

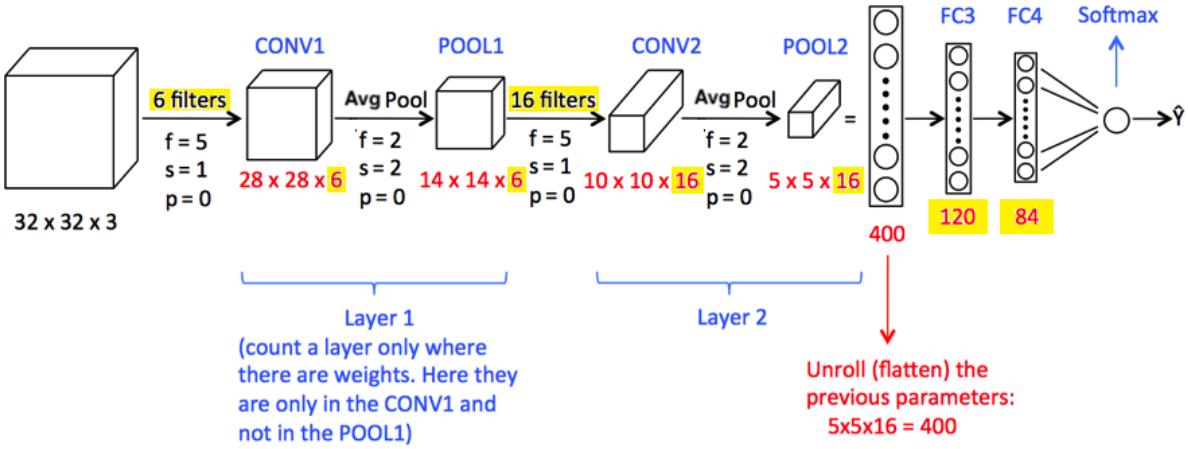


Figure 2.6: LeNet-5 Data flow(source: Medium)

- The input consists of grayscale images with a single channel and spatial dimensions of 28x28 pixels (H x W). As illustrated in the architecture above, it is noteworthy that padding of 2 is applied to the images before they are fed into the model. Consequently, the effective input size becomes 32x32 pixels (H + padding2) x (W + padding2)].

- It is important to note that, at this point in time, the Rectified Linear Unit (ReLU) function $\text{ReLU}(x) = \max(0, x)$, which is commonly used in modern neural network architectures, had not yet been discovered.

2.3 AlexNet

2.3.1 What is AlexNet?

The AlexNet architecture, introduced by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012 in the research paper ImageNet Classification with Deep Convolutional Neural Networks, marked a significant milestone in the field of deep learning. AlexNet is a pioneering convolutional neural network (CNN) that gained widespread attention for its success in the ImageNet Large Scale Visual Recognition Challenge. It consists of 5 convolutional layers followed by 3 fully connected layers, utilizing techniques such as ReLU activation, local response normalization, and dropout.

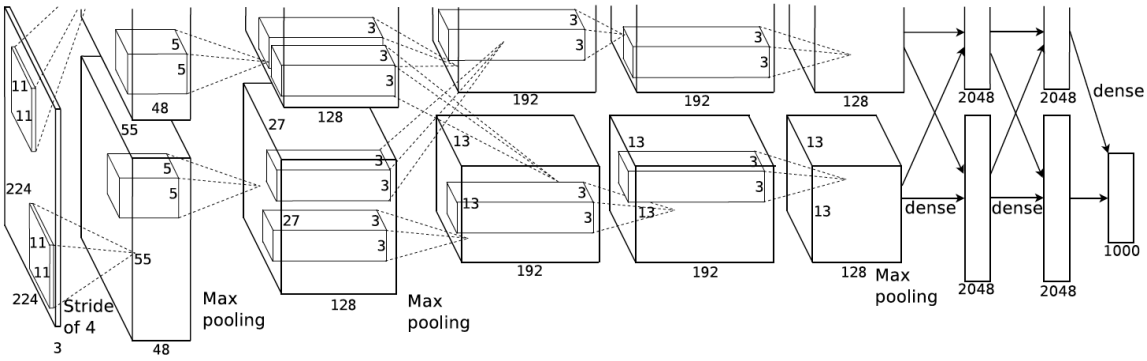


Figure 2.7: Original AlexNet architecture
[2]

2.3.2 AlexNet Architecture

- The Figure bellow is the architecture of AlexNet:

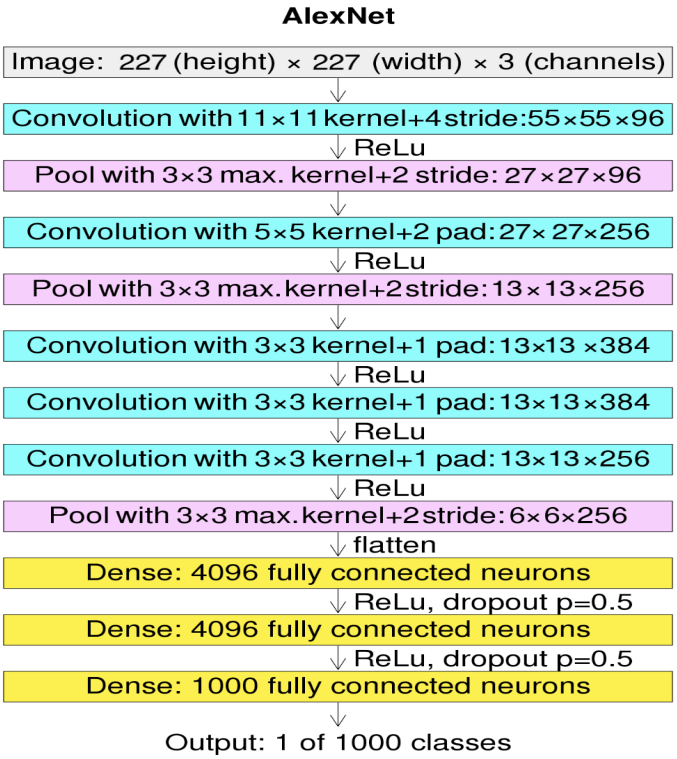


Figure 2.8: Detail AlexNet architecture

- The Figure bellow is the Data flow of AlexNet:

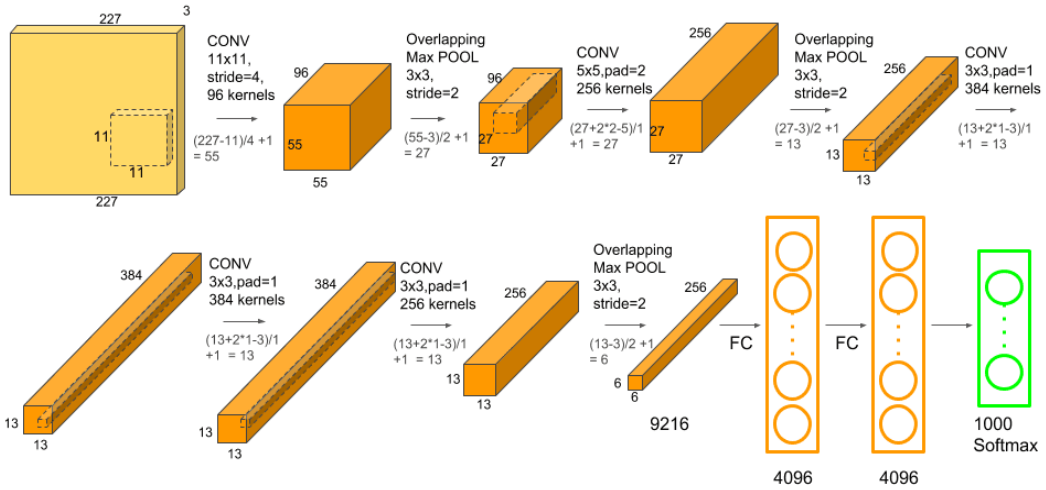


Figure 2.9: AlexNet Data flow(source: kaggle)

- The input to the AlexNet architecture comprises color (RGB) images with 3 channels and spatial dimensions of 227 × 227 pixels (H × W).
- In Alex Krizhevsky's paper, the specified input size is 224 × 224. This could has been a mistake from the authors since the minimum input size of AlexNet is 224 × 224 so the actual input images must be 227 × 227 for the math in the paper.

2.4 Training and Testing

2.4.1 Preparing Dataset & Parameters

- Dataset Preprocessing:
  - LeNet-5: 2-pixel padding for images size 28x28 to 32x32.
  - AlexNet: Resizing images from 28x28 to 227x227 pixels and converting to 3-channel grayscale.
  - Standardizing pixel values to [-1, 1] through mean(0.1307) subtraction and standard deviation(0.3081) scaling.
  - Split into training and validation subsets, with ratio 80/20.
  - Tensor conversion with batch size of 128 for GPU-accelerated.
- Parameters for training:
  - Platform tools: Python 3.11, Pytorch 2.0+, Torchvision 0.15+
  - Learning rate: 0.001
  - Optimizer: Adam
  - Loss function: Cross-Entropy Loss
  - Number of Epochs: [10, 20, 20, 40]

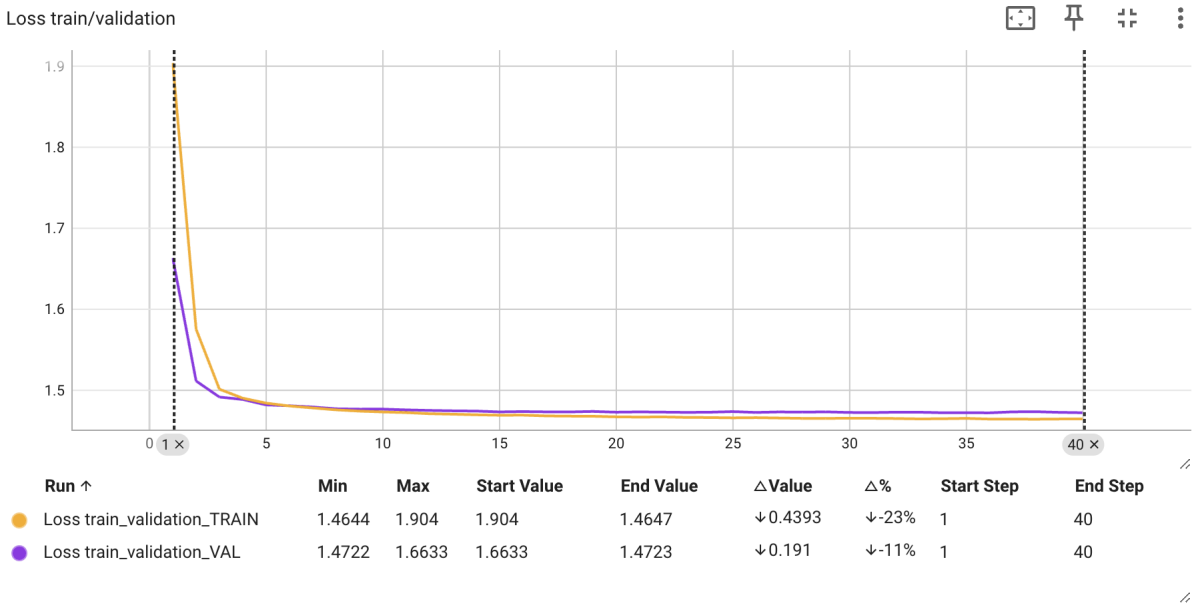
2.4.2 Training Process

We execute identical training procedures for both LeNet-5 and AlexNet:

- The training Phase iterates over 10, 20, 30, and 40 epochs, with each epoch incorporating 48,000 samples for training (divided into 375 batches with a batch size of 128).
- For each batch, the loss function calculates the error, triggering the backpropagation process. Subsequently, the Adam optimization algorithm updates the model parameters (weights) using a learning rate of 0.001 in PyTorch to minimize the loss.
- After completing the training phase for an epoch, the model undergoes the Validation Phase. The 12,000 validation samples are processed through 94 batches, mirroring the training setup. The validation metrics, including loss and accuracy, are logged for each epoch.

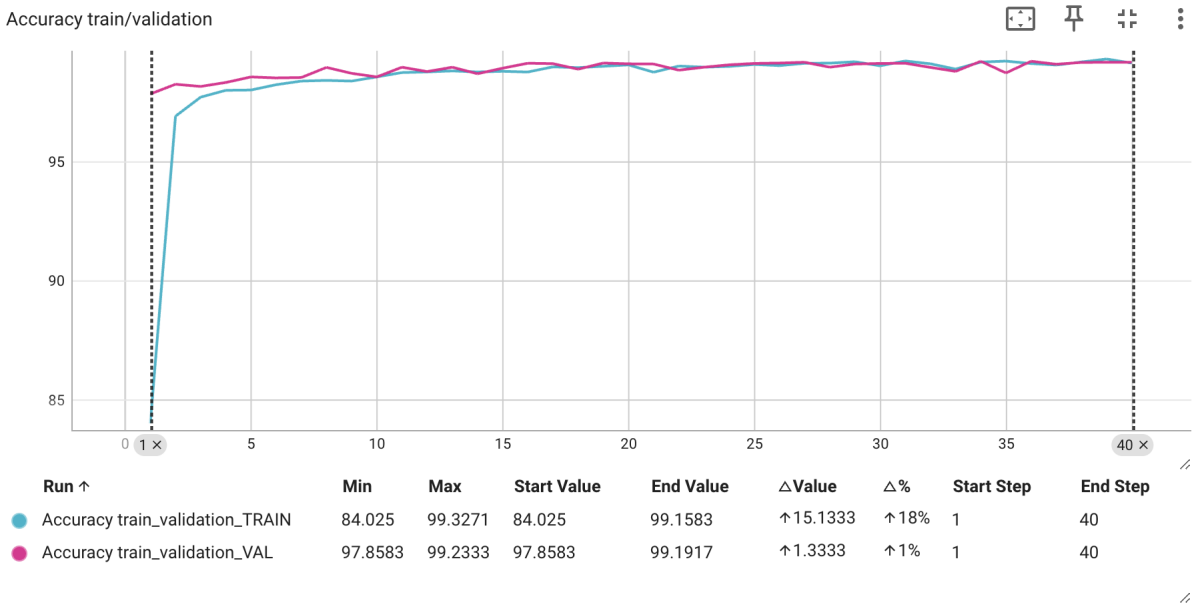
Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
Accuracy train_validation_TRAIN	59.2958	99.7125	59.2958	99.6646	↑40.3687	↑68%	1	40
Accuracy train_validation_VAL	83.0583	98.9333	83.0583	98.8917	↑15.8333	↑19%	1	40

Figure 2.10: LeNet-5 training results Accuracy/Epoch



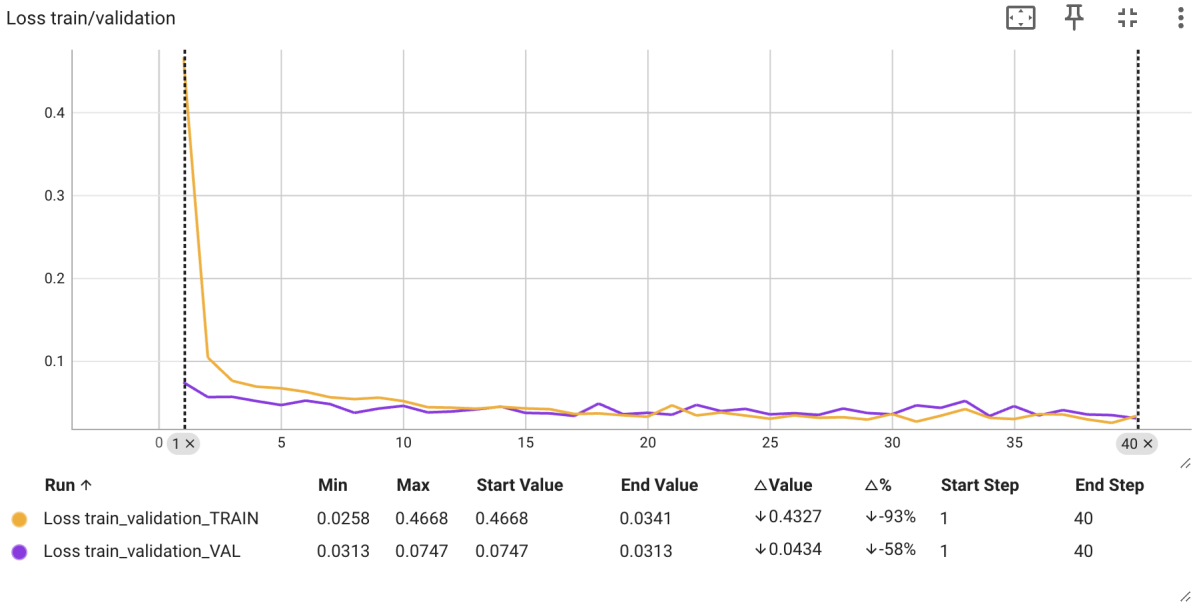
Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
Loss train_validation_TRAIN	1.4644	1.904	1.904	1.4647	↓0.4393	↓-23%	1	40
Loss train_validation_VAL	1.4722	1.6633	1.6633	1.4723	↓0.191	↓-11%	1	40

Figure 2.11: LeNet-5 training results Accuracy/Epoch



Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
Accuracy train_validation_TRAIN	84.025	99.3271	84.025	99.1583	↑15.1333	↑18%	1	40
Accuracy train_validation_VAL	97.8583	99.2333	97.8583	99.1917	↑1.3333	↑1%	1	40

Figure 2.12: AlexNet training results Accuracy/Epoch



Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
Loss train_validation_TRAIN	0.0258	0.4668	0.4668	0.0341	↓0.4327	↓-93%	1	40
Loss train_validation_VAL	0.0313	0.0747	0.0747	0.0313	↓0.0434	↓-58%	1	40

Figure 2.13: AlexNet training results Loss/Epoch

2.4.3 Testing Process

After completing the training phase and validation phase, the model undergoes the Testing Phase. The 10,000 test samples are processed through 100 batches, each batch containing 100 samples, with the testing process mirroring the validation setup.

1. The metrics we use for testing are:

   - Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

   - Accuracy is a straightforward metric that measures the ratio of correctly classified samples to the total number of samples. It provides a general indication of the model's overall correctness.

- Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Precision measures the accuracy of the positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives.

- Recall (Sensitivity):

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

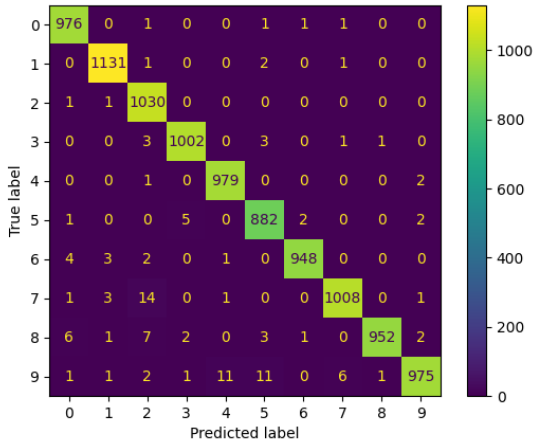
- Recall, or sensitivity, measures the ability of the model to correctly identify all relevant instances (true positives) among the total actual positives.

- F1 Score:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, which is useful when there is an imbalance between the classes.

- Confusion Matrix: A confusion matrix provides a detailed breakdown of correct and incorrect classifications for each class. It includes metrics such as true positives, true negatives, false positives, and false negatives.



2. LeNet-5 Testing Results:

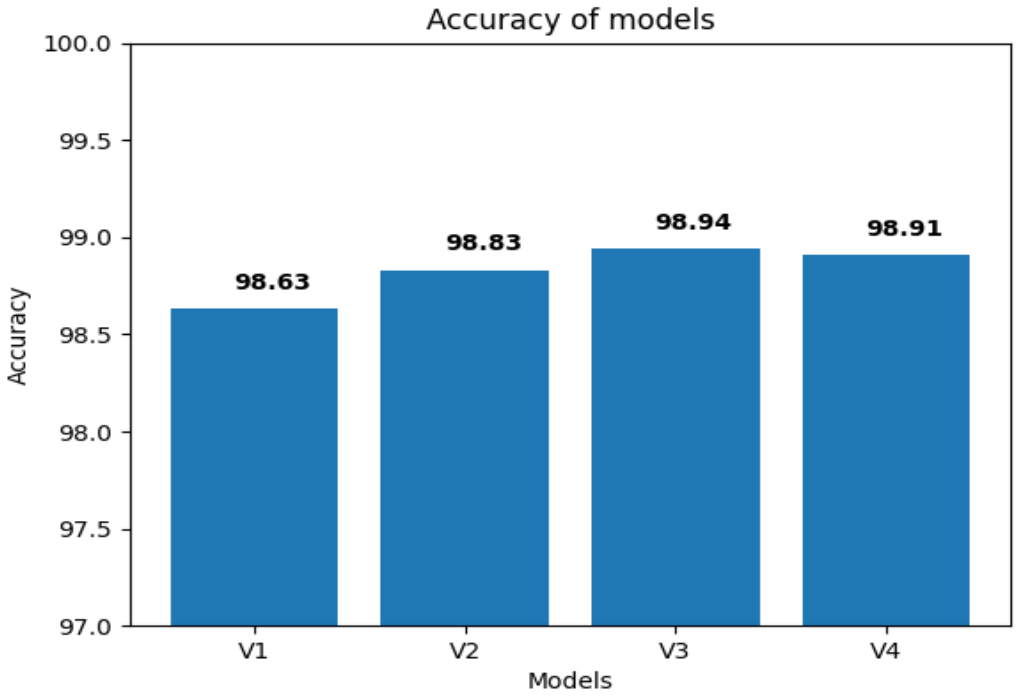


Figure 2.14: LeNet-5 Average Accuracy

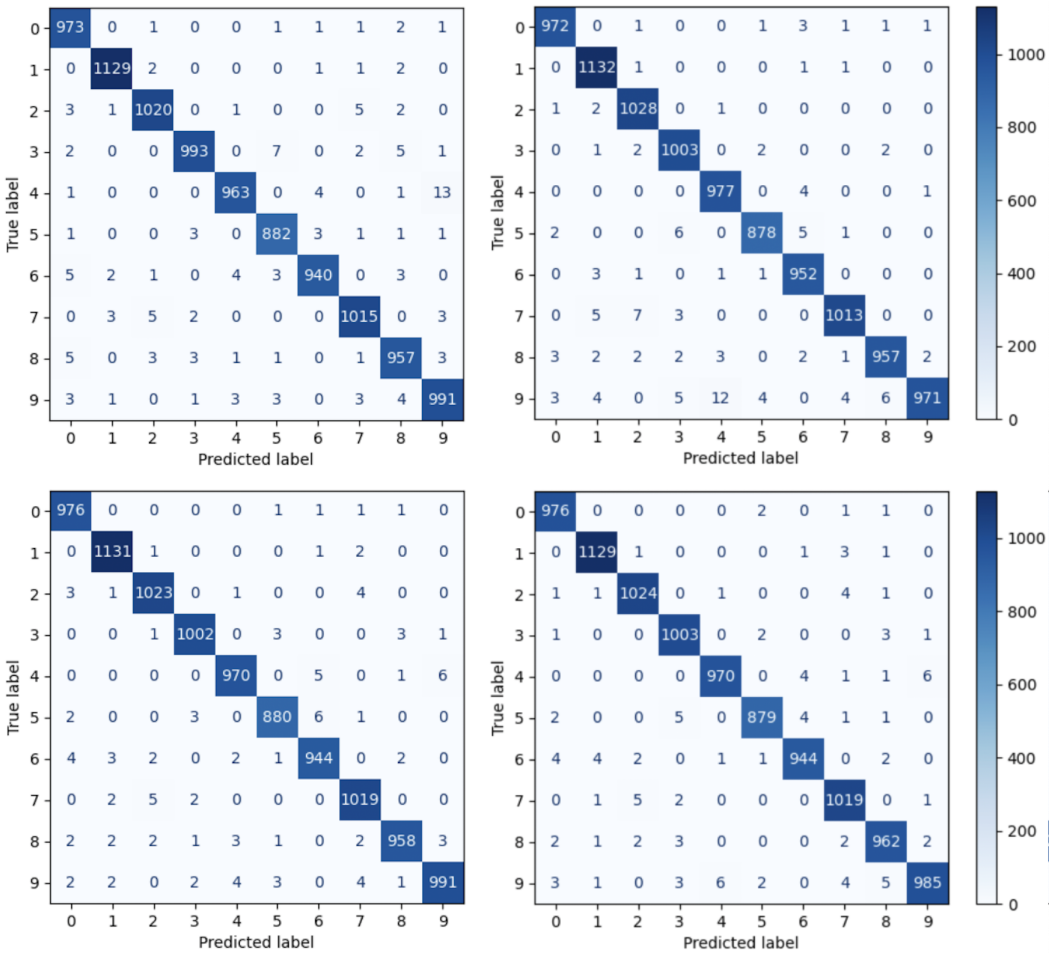


Figure 2.15: LeNet-5 Confusion Matrix V1 to 4

LeNet-5 V1					LeNet-5 V2				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.98	0.99	0.99	980	0	0.99	0.99	0.99	980
1	0.99	0.99	0.99	1135	1	0.99	1.00	0.99	1135
2	0.99	0.99	0.99	1032	2	0.99	1.00	0.99	1032
3	0.99	0.98	0.99	1010	3	0.98	0.99	0.99	1010
4	0.99	0.98	0.99	982	4	0.98	0.99	0.99	982
5	0.98	0.99	0.99	892	5	0.99	0.98	0.99	892
6	0.99	0.98	0.99	958	6	0.98	0.99	0.99	958
7	0.99	0.99	0.99	1028	7	0.99	0.99	0.99	1028
8	0.98	0.98	0.98	974	8	0.99	0.98	0.99	974
9	0.98	0.98	0.98	1009	9	1.00	0.96	0.98	1009
acc			0.99	10000	acc			0.99	10000
LeNet-5 V3					LeNet-5 V4				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.99	1.00	0.99	980	0	0.99	1.00	0.99	980
1	0.99	1.00	0.99	1135	1	0.99	0.99	0.99	1135
2	0.99	0.99	0.99	1032	2	0.99	0.99	0.99	1032
3	0.99	0.99	0.99	1010	3	0.99	0.99	0.99	1010
4	0.99	0.99	0.99	982	4	0.99	0.99	0.99	982
5	0.99	0.99	0.99	892	5	0.99	0.99	0.99	892
6	0.99	0.99	0.99	958	6	0.99	0.99	0.99	958
7	0.99	0.99	0.99	1028	7	0.98	0.99	0.99	1028
8	0.99	0.98	0.99	974	8	0.98	0.99	0.99	974
9	0.99	0.98	0.99	1009	9	0.99	0.98	0.98	1009
acc			0.99	10000	acc			0.99	10000

- The LeNet-5 model exhibits high accuracy across all versions, ranging from 98.63% to 98.91%.
- Classes 1, 4, and 8 consistently exhibit high scores, indicating reliable predictions for these digits.
- The model maintains high accuracy and consistent precision, recall, and F1 scores over multiple epochs. This suggests that the model is robust and capable of generalizing well to the MNIST test dataset.
- Generally, as the number of training epochs increases, the model's performance improves. This is evident in the increasing accuracy and generally improving precision, recall, and F1 score metrics over different versions.
- The efficient utilization of a GPU-accelerated environment significantly enhances the training speed of LeNet-5. The training times, spanning 1.058 minutes for 10 epochs, 2.233 minutes for 20 epochs, 3.399 minutes for 30 epochs, and 5.057 minutes for 40 epochs, highlight the accelerated learning capabilities facilitated by GPU processing.
- In summary, the LeNet-5 model exhibits impressive performance in digit classification on the MNIST dataset. It demonstrates consistent improvement with additional training epochs, achieving high accuracy and reliable precision, recall, and F1 scores across multiple classes. Fine-tuning opportunities could be explored to further optimize performance, especially for specific digits where fluctuations are observed.

3. AlexNet Testing Results:

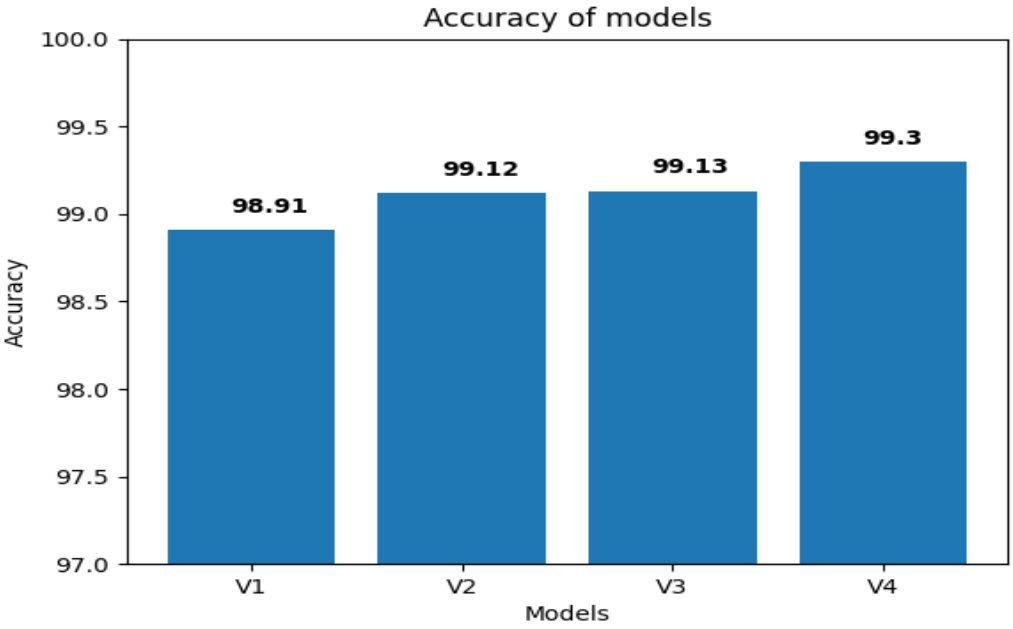


Figure 2.16: AlexNet Average Accuracy

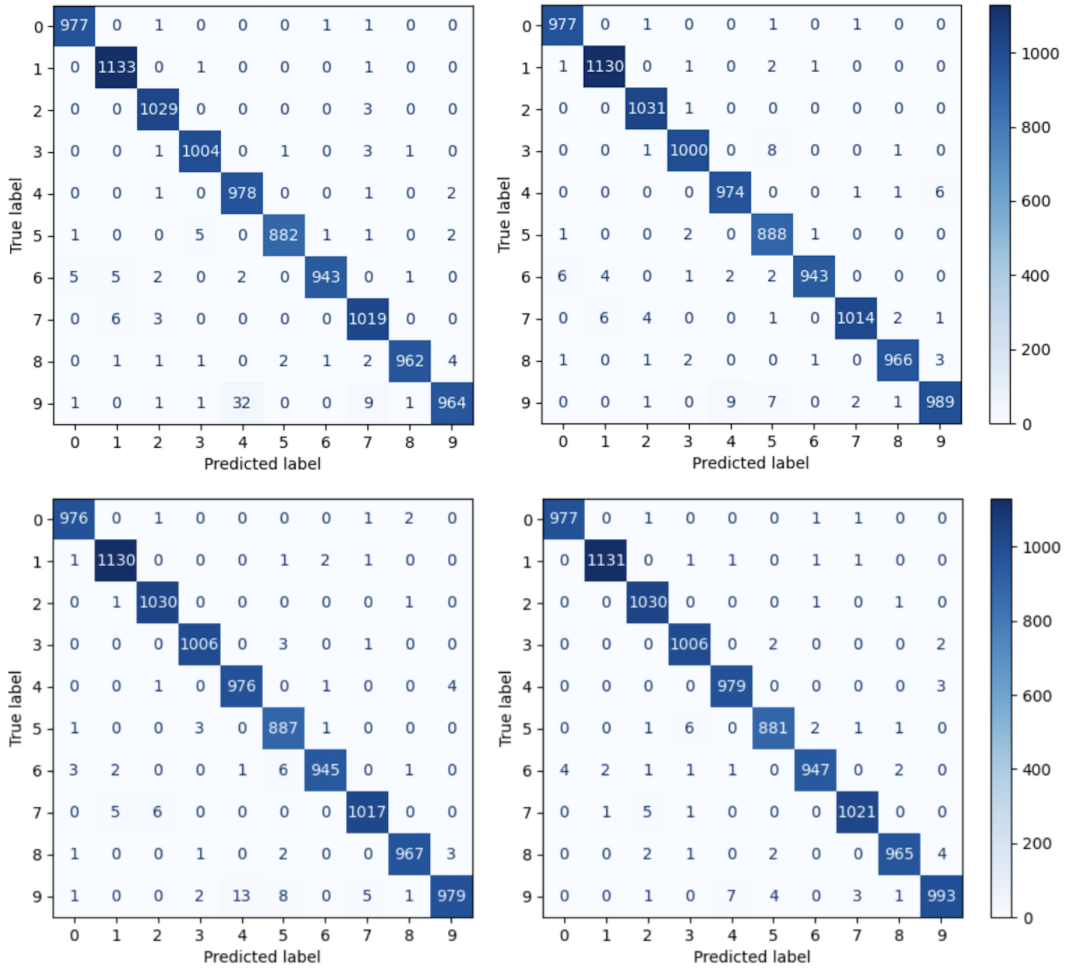


Figure 2.17: AlexNet Confusion Matrix V1 to 4

AlexNet V1					AlexNet V2				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.99	1.00	0.99	980	0	0.99	1.00	0.99	980
1	0.99	1.00	0.99	1135	1	0.99	1.00	0.99	1135
2	0.99	1.00	0.99	1032	2	0.99	1.00	1.00	1032
3	0.99	0.99	0.99	1010	3	0.99	0.99	0.99	1010
4	0.97	1.00	0.98	982	4	0.99	0.99	0.99	982
5	1.00	0.99	0.99	892	5	0.98	1.00	0.99	892
6	1.00	0.98	0.99	958	6	1.00	0.98	0.99	958
7	0.98	0.99	0.99	1028	7	1.00	0.99	0.99	1028
8	1.00	0.99	0.99	974	8	0.99	0.99	0.99	974
9	0.99	0.96	0.97	1009	9	0.99	0.98	0.99	1009
acc			0.99	10000	acc			0.99	10000
AlexNet V3					AlexNet V4				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.99	1.00	0.99	980	0	1.00	1.00	1.00	980
1	0.99	1.00	0.99	1135	1	1.00	1.00	1.00	1135
2	0.99	1.00	1.00	1032	2	0.99	1.00	0.99	1032
3	0.99	1.00	1.00	1010	3	0.99	1.00	0.99	1010
4	0.99	0.99	0.99	982	4	0.99	1.00	0.99	982
5	0.98	0.99	0.99	892	5	0.99	0.99	0.99	892
6	1.00	0.99	0.99	958	6	0.99	0.99	0.99	958
7	0.99	0.99	0.99	1028	7	0.99	0.99	0.99	1028
8	0.99	0.99	0.99	974	8	0.99	0.99	0.99	974
9	0.99	0.97	0.98	1009	9	0.99	0.98	0.99	1009
acc			0.99	10000	acc			0.99	10000

- AlexNet demonstrates excellent overall accuracy, ranging from 98.91% to 99.3%, showcasing its effectiveness in classifying digits in the MNIST dataset.
- Across different versions and epochs, precision, recall, and F1 scores are consistently high for most classes, indicating robust and reliable predictions.
- The model consistently improves its accuracy and class-specific metrics with additional training epochs, showcasing its ability to capture intricate patterns in the data.
- The model handles the imbalanced nature of the MNIST dataset well, providing reliable predictions even for classes with fewer samples.
- AlexNet generally achieves higher accuracy compared to LeNet-5 across all evaluated epochs, slightly better precision, recall, and F1 scores in some instances compare to LeNet-5.
- In comparison to training LeNet-5, AlexNet exhibits relatively longer training times on a GPU-accelerated environment. The training durations are 15.19 minutes for 10 epochs, 32.11 minutes for 20 epochs, 48.92 minutes for 30 epochs, and 1.3 hours for 40 epochs. While AlexNet requires more time per epoch compared to LeNet-5, it's essential to consider the complexity of the model. The longer training times highlight the trade-off between model sophistication and computational efficiency.
- In summary, AlexNet exhibits strong performance in digit classification on the MNIST dataset. Its high accuracy and consistent precision, recall, and F1 scores across various classes and epochs underscore its effectiveness in capturing digit patterns. Fine-tuning opportunities could be explored for further optimization, but overall, AlexNet demonstrates robust performance.

Chapter 3

Conclusion

- In conclusion, LeNet-5 and AlexNet mark pivotal moments in the evolution of convolutional neural networks. LeNet-5's historical significance lies in its simplicity and efficiency, while AlexNet's depth and innovations have paved the way for the development of more intricate architectures.
- LeNet-5 is well-suited for straightforward applications such as digit recognition, emphasizing lightweight and efficient feature extraction. However, its limited depth, combined with outdated activation functions, constrains its capacity to capture highly complex features. As a result, its applicability is restricted to less sophisticated tasks.
- In contrast, AlexNet offers a more robust solution tailored for demanding computer vision tasks. Its deep architecture, coupled with newer ReLU activation functions and dropout regularization, enhances the network's generalization capabilities. This facilitates improved training and reduces overfitting, enabling AlexNet to capture intricate features and achieve superior performance on diverse and challenging datasets. Nevertheless, it comes with the trade-off of increased computational requirements and sensitivity to hyperparameters.

.1 Appendix

.1.1 Code Implementation

You can find the code implementation of our models on GitHub:

`https://github.com/LTH3ar/MLDM_2_Group_Project.git`

.1.2 Additional Material

List of tools we use in our study

1. Python 3.11.6: `https://www.python.org/downloads/release/python-3116/`
2. Pytorch: `https://pytorch.org/`
3. Tensorboard: `https://www.tensorflow.org/tensorboard`
4. Jupyter Notebook: `https://jupyter.org/`

Bibliography

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.