COMP 472: Artificial Intelligence

Project: Image Classification using CIFAR10 dataset

Due date: Monday November 25th, 2024

Prepared by:

Andrew Chebli - 40125859

Qian Yi Wang - 40211303

Professor: Dr. Ali Ayub

Grader: Professor Amin Karimi

Table Of contents

1. Introduction:

The objective of this project is to apply various machine learning models for image classification on the CIFAR-10 dataset, which consists of 32x32 RGB images belonging to 10 different object classes. The following models were implemented and evaluated:

- Naive Bayes
- Decision Tree
- Multi-Layer Perceptron (MLP)
- Convolutional Neural Network (CNN)

The goal was to explore the effectiveness of each model and its variants by analyzing performance metrics such as accuracy, precision, recall, and F1-score. The project also includes comparison between different configurations for each model type.

2. Methodology:

2.1. Dataset Preprocessing:

The CIFAR-10 dataset consists of 32x32 pixel RGB images, which were preprocessed as follows:

- **Resizing**: The images were resized to 224x224 pixels to match the input size expected by the pre-trained **ResNet-18** model.
- **Feature Extraction**: Feature vectors of size **512x1** were extracted from the ResNet-18 model, pre-trained on ImageNet. These feature vectors were reduced to **50x1** using **Principal Component Analysis (PCA)** for dimensionality reduction.
- **Final Dataset**: The 50-dimensional feature vectors were used as input to train and test the various machine learning models.

2.2. Model Architectures and Training:

- **Naive Bayes:**
  The custom Naive Bayes Model was implemented using only basic Python and NumPy libraries from scratch. It assumes feature independence and calculates probabilities using Gaussian likelihood. The difference between this model and a neural network is that this model does not undergo iterative training, it computes statistical parameters like the “mean” and “standard deviation” for each label-feature combination. The SciKit-Learn Naive Bayes Model relies on GaussianNB as well. It uses the fit() method for training on preprocessed data.

- **Decision Tree:**

The Custom Decision tree model was designed from scratch, employing the Gini coefficient to determine the best split at each node. We experimented with different depths, 10,20 and 50 to analyze the trade-offs between underfitting and overfitting. The tree construction follows an iterative process, and will terminate when we reach the specified depth, or if a node contains a single class. No optimization techniques were used, we relied on deterministic splitting only. On the other hand, the Scikit-learn Decision tree model uses built in DecisionTreeClassifier with the criterion as the "gini" coefficient. The model was trained using the fit() method provided by the library, and experiments were also conducted on different depths: 10,20,50.

- **Multi-Layer Perceptron (MLP):**

  The MLP was implemented using PyTorch, it features a fully connected architecture, with configurable layers and unit. Our base model includes two hidden layers, each containing 512 perceptrons. Different experimental models were also created by changing the number and size of the hidden layers, including configurations with:
  - one hidden layer of 512 perceptrons,
  - Three hidden layers of 512 perceptrons each,
  - Two hidden layers of 256 perceptrons each,
  - Two larger hidden layers of 1024 perceptrons each.

  The models were trained for 20 epochs using a batch size of 32. We employed CrossEntropyLoss function and SGD (stochastic gradient descent) with a learning rate of 0.01 and a momentum of 0.9.

- **Convolutional Neural Network (CNN):**

  The CNN model followed the VGG11 architecture, it consists of multiple convolutional layers followed by fully connected layers and dropout for regularization. We used a default kernel size of 3, then experimented by adding an extra convolutional layer, removing the last convolutional layer, but also modifying the kernel size to 5 to compare the accuracy. The modifications helped us see how well CNN performs under different circumstances. Training was conducted for 50 epochs, with a batch size of 32. CrossEntropyLoss was used as our loss function, and the Optimizer was SGD with a learning rate of 0.001 and a momentum of 0.9. In order to enhance the training, data augmentation was also applied.

2.3. Training Methodology:

- **Epochs:** 20-50 epochs for MLP and CNN based on performance convergence.
- **Learning Rate:** 0.01 for MLP and 0.001 for CNN.
- **Optimizer:** SGD with momentum of 0.9.

3. Results:

3.1. Evaluation Metrics:

For each model and its variants, the following evaluation metrics were computed: - **Accuracy** - **Precision** - **Recall** - **F1-measure**

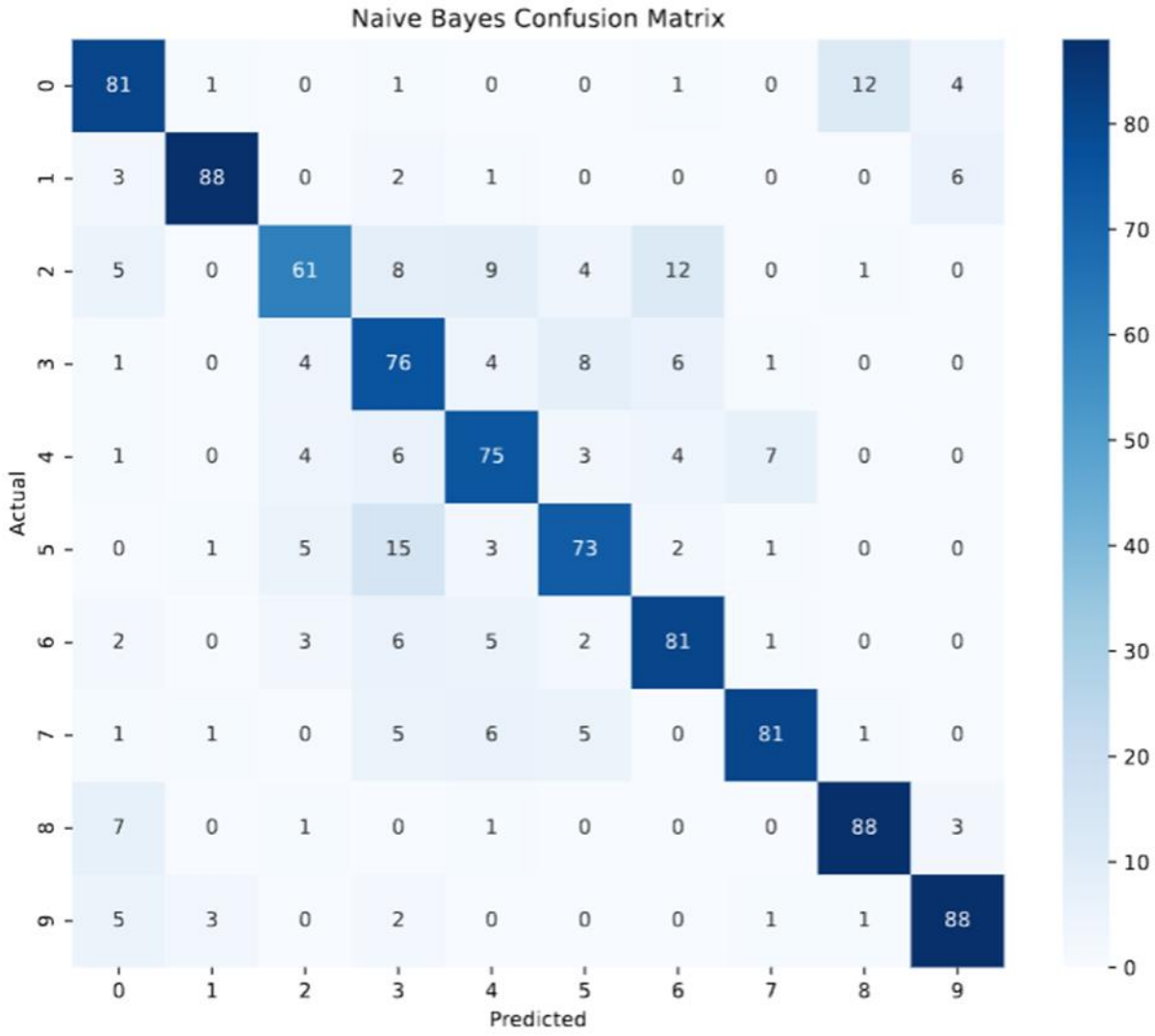
Model	Accuracy	Precision	Recall	F1 Score
Naive Bayes	0.792	0.797	0.792	0.792
Naive Bayes Scikit	0.792	0.797	0.792	0.792
Decision Tree - Max Depth: 10	0.599	0.604	0.599	0.6
Decision Tree - Max Depth: 20	0.578	0.581	0.578	0.578
Decision Tree - Max Depth: 50	0.585	0.589	0.585	0.586
Decision Tree Scikit - Max Depth: 10	0.601	0.614	0.601	0.605
Decision Tree Scikit - Max Depth: 20	0.586	0.589	0.586	0.586
Decision Tree Scikit - Max Depth: 50	0.589	0.589	0.589	0.588
MLP - Network Depth: 2-layer	0.773	0.772	0.773	0.771
MLP - Network Depth: 1 hidden layer	0.751	0.754	0.751	0.75
MLP - Network Depth: 3-layer MLP	0.781	0.785	0.781	0.78
MLP - Network Depth: Smaller hidden layers	0.724	0.73	0.724	0.72
MLP - Network Depth: Larger hidden layers	0.794	0.798	0.794	0.794
cnn_kernel_size_3_add_extra_layer.pth	0.719	0.725	0.719	0.719
cnn_kernel_size_3_remove_last_layer.pth	0.699	0.737	0.699	0.686
cnn_kernel_size_3_default.pth	0.731	0.73	0.731	0.727
cnn_kernel_size_5_default.pth	0.676	0.686	0.676	0.675

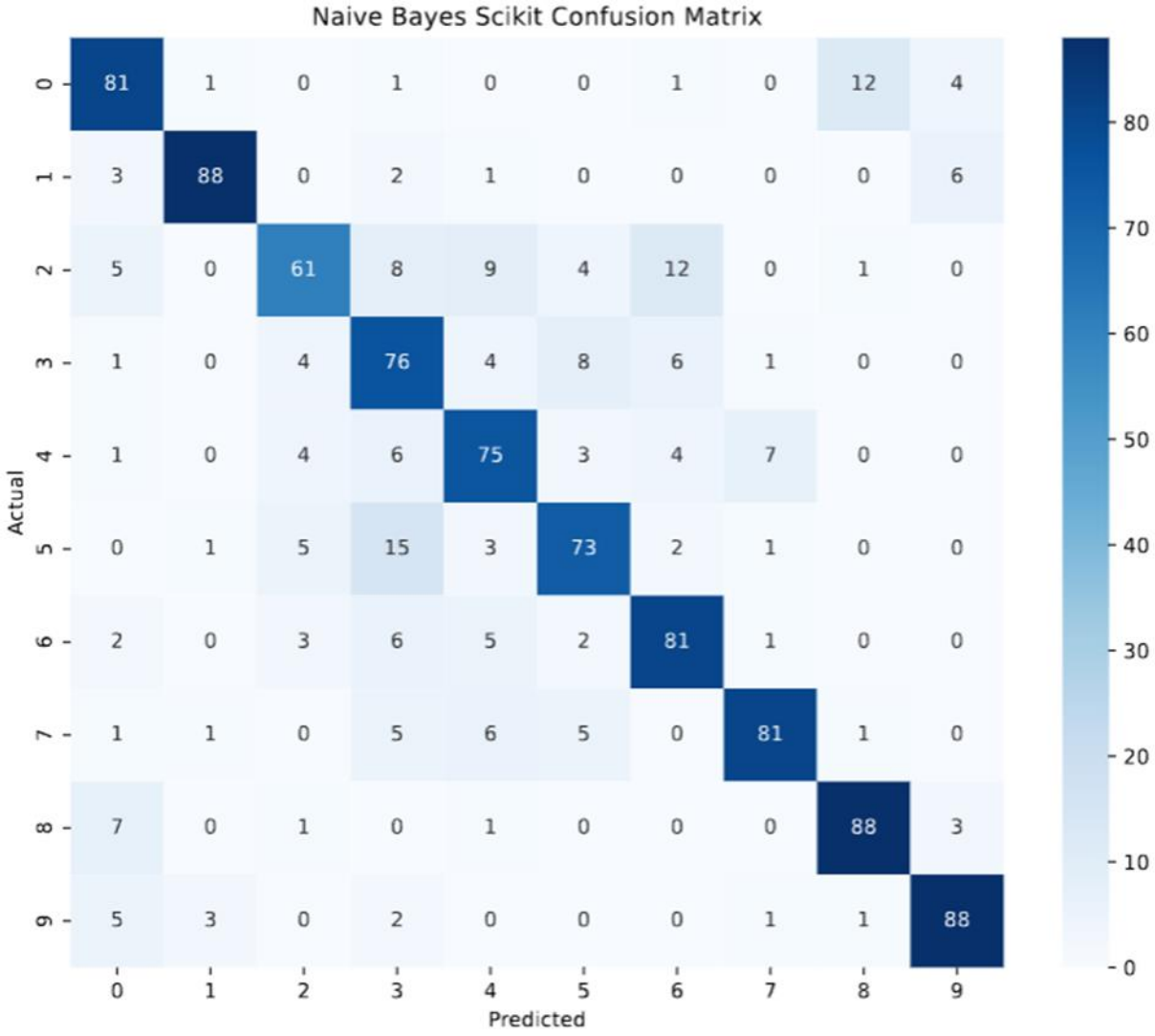
3.2. Confusion Matrices:

Confusion matrices were generated to evaluate the performance of the models, highlighting misclassifications.

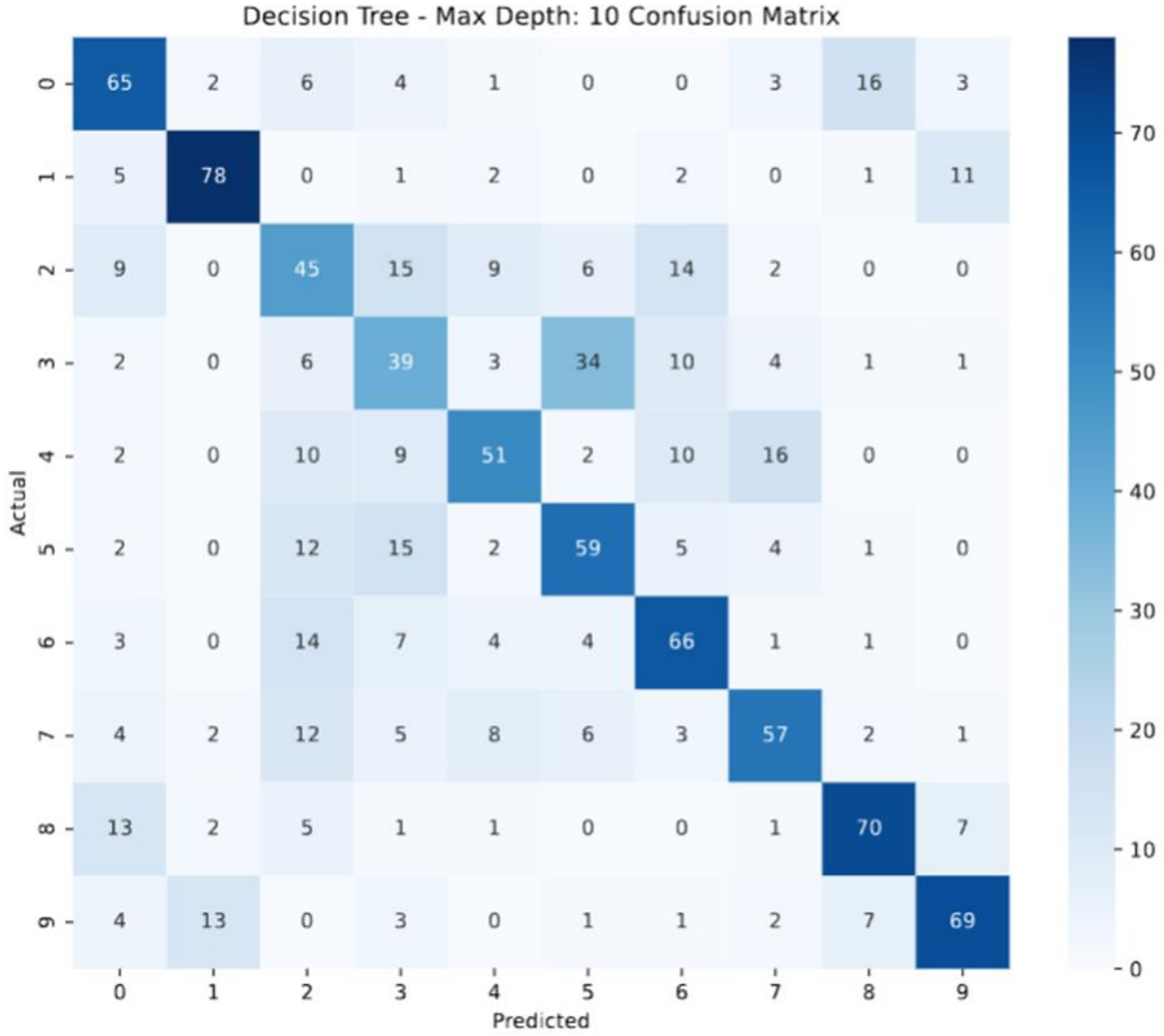
- **Naive Bayes:**

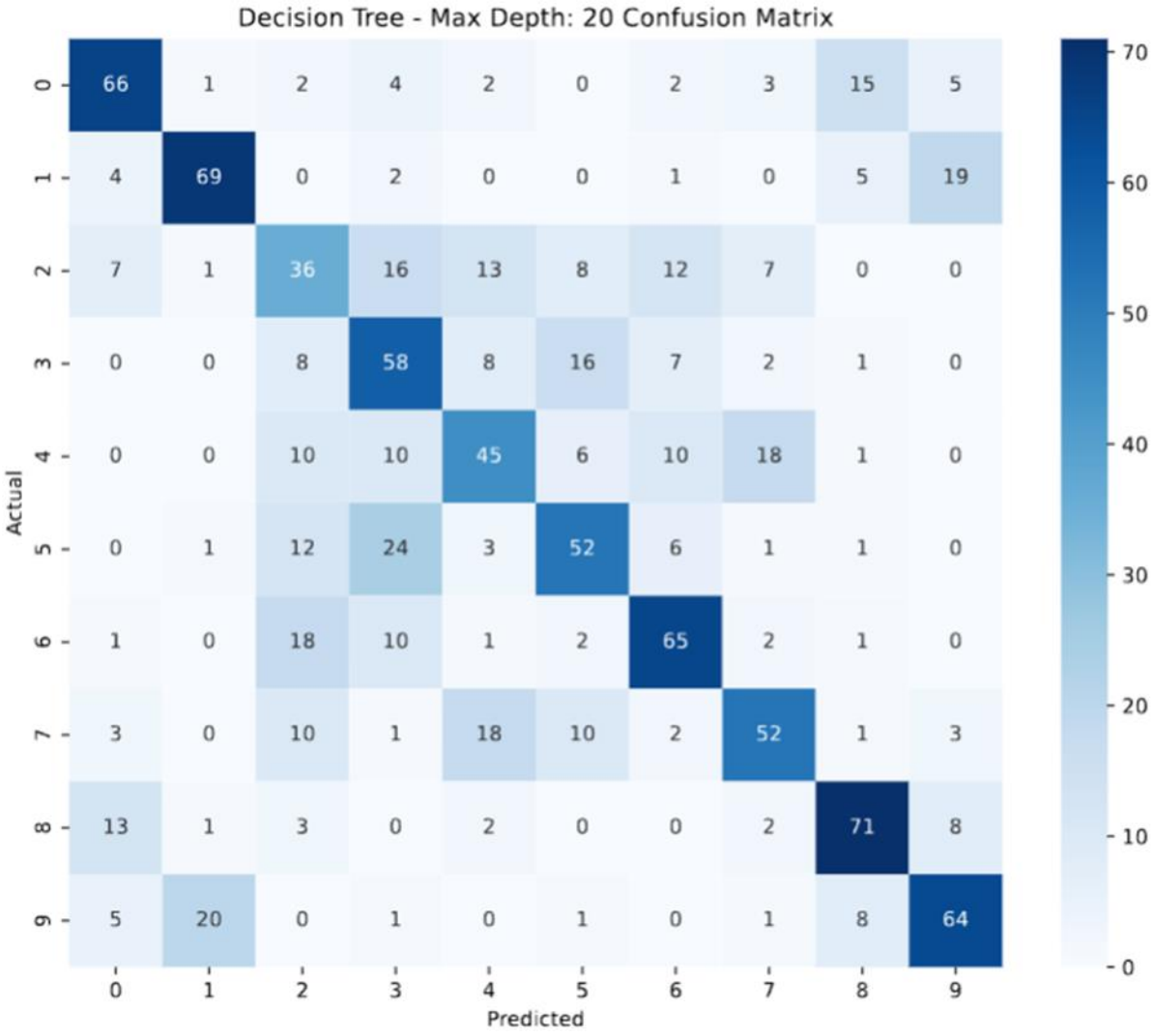
Both the custom and the Scikit-Learn implementations of the Naive Bayes Model performed identically, achieving the accuracy, recall, and f1 score of 79.2%, and the precision of 79.7%. The consistency shows the robustness of the probabilistic approach. This would likely be due to the assumption that the features are independent

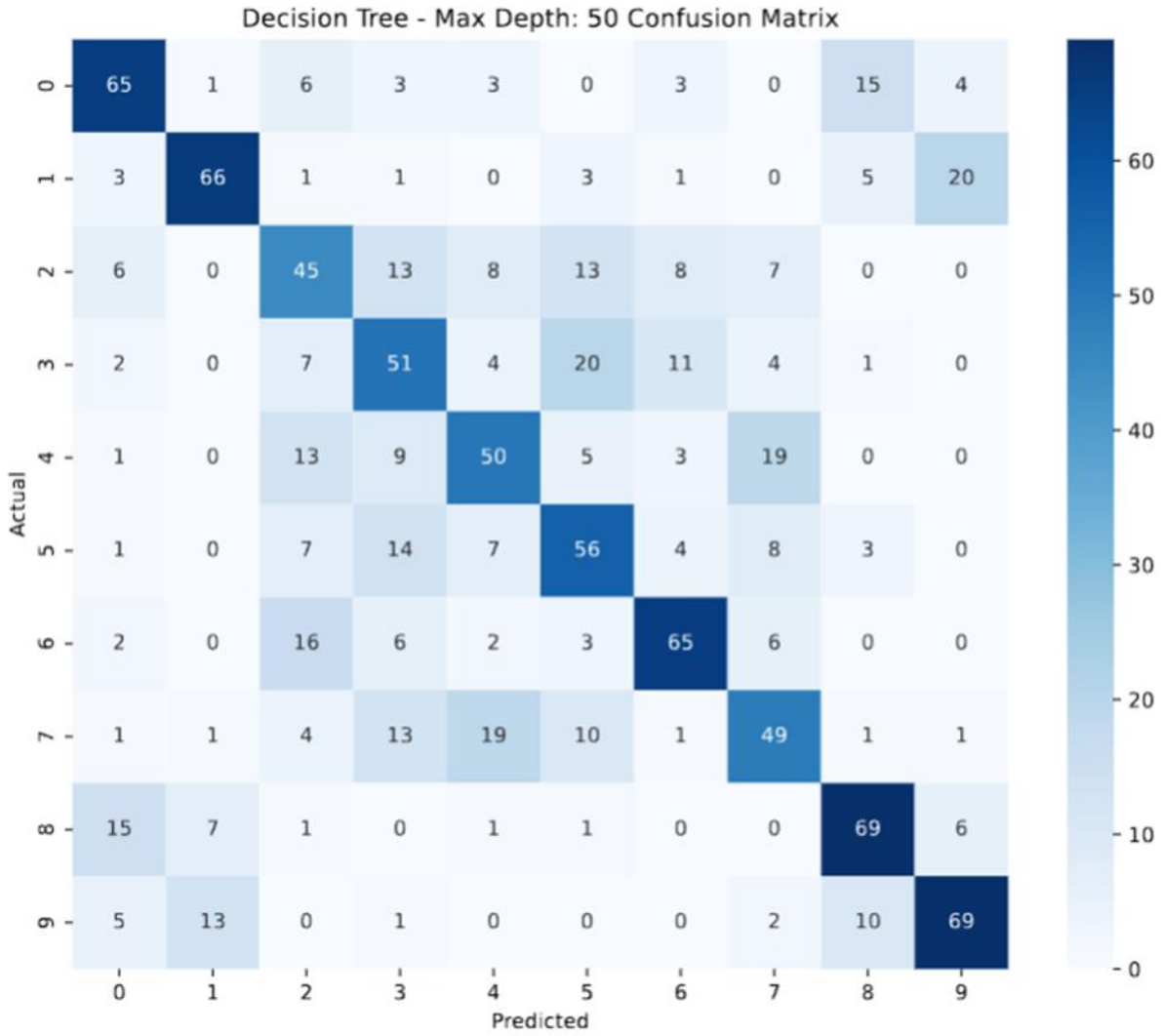


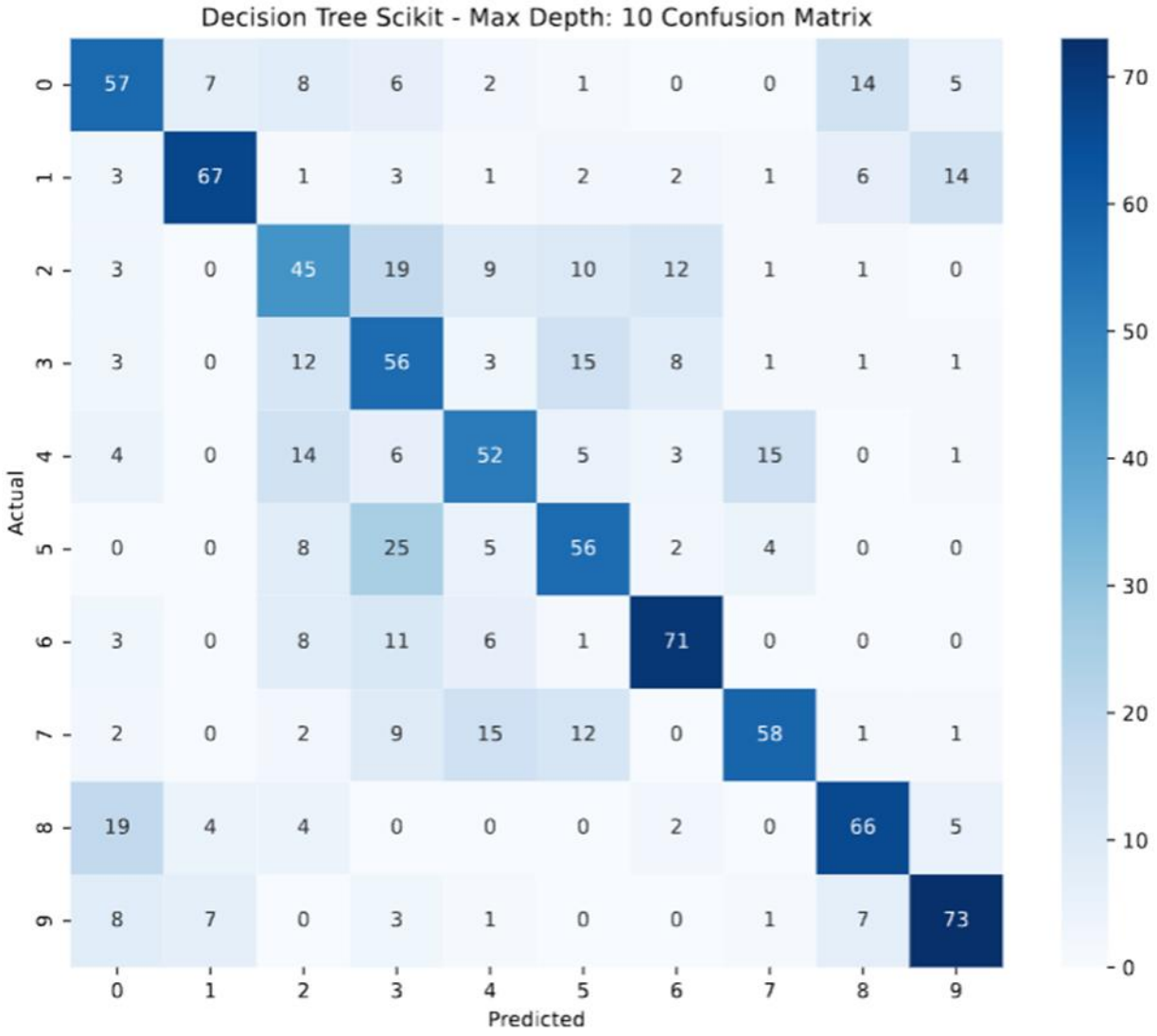


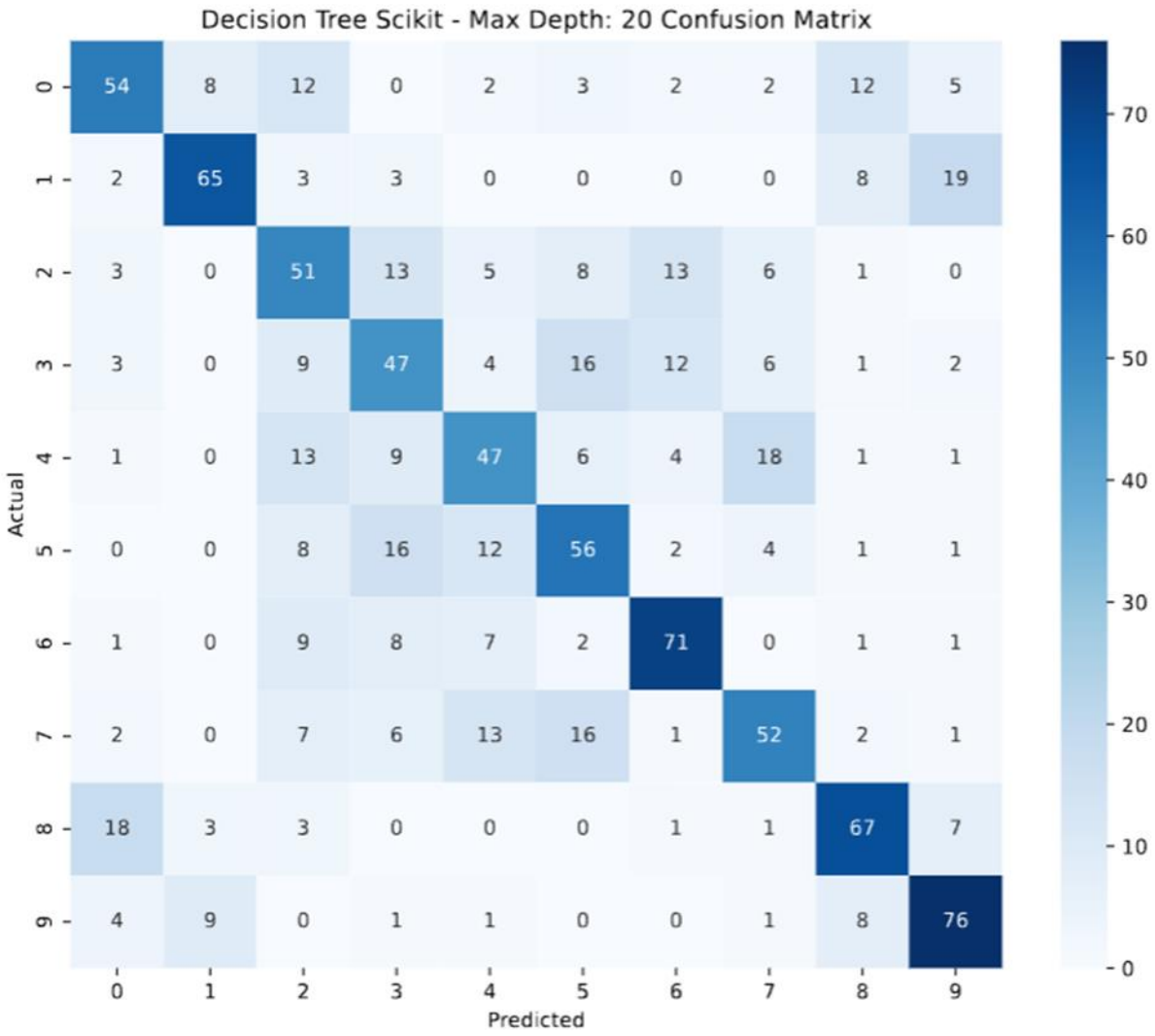
- **Decision Tree:**
  Our custom implementation with varying depths: 10, 20, 50 demonstrated declining performance, as depth increased, which suggests that overfitting would cause the precision to go down, especially when it becomes too complex. The Sci-kit learn implementation consistently outperformed our models, likely due to the optimized splitting and pruning strategies they implemented.

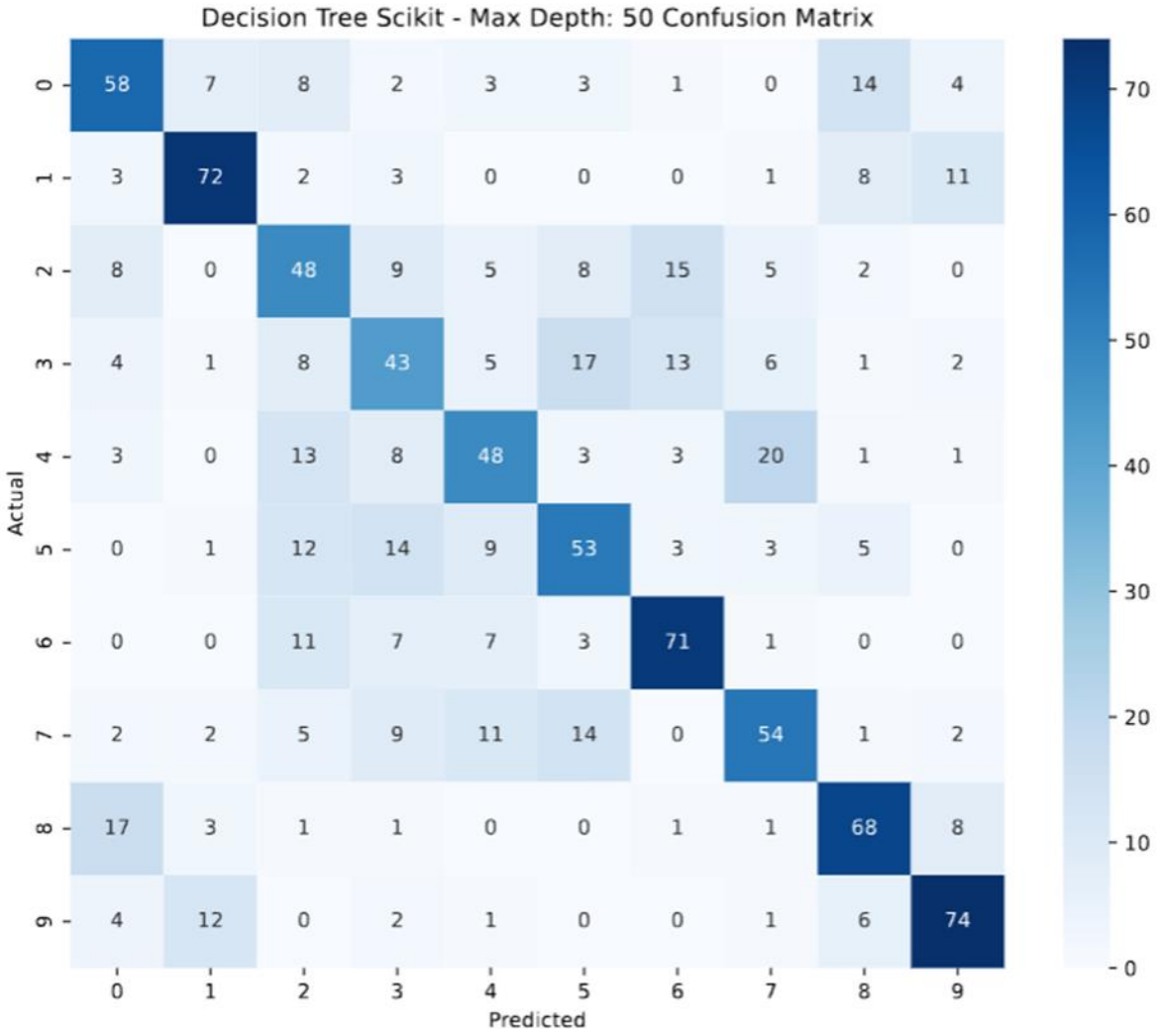

Decision Tree - Max Depth: 10 Confusion Matrix
Actual
Predicted
0 65 2 6 4 1 0 0 3 16 3
1 5 78 0 1 2 0 2 0 1 11
2 9 0 45 15 9 6 14 2 0 0
3 2 0 6 39 3 34 10 4 1 1
4 2 0 10 9 51 2 10 16 0 0
5 2 0 12 15 2 59 5 4 1 0
6 3 0 14 7 4 4 66 1 1 0
7 4 2 12 5 8 6 3 57 2 1
8 13 2 5 1 1 0 0 1 70 7
9 4 13 0 3 0 1 1 2 7 69
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0


Decision Tree - Max Depth: 20 Confusion Matrix
Actual
Predicted
0 1 2 3 4 5 6 7 8 9
66 1 2 4 2 0 2 3 15 5
4 69 0 2 0 0 1 0 5 19
7 1 36 16 13 8 12 7 0 0
0 0 8 58 8 16 7 2 1 0
0 0 10 10 45 6 10 18 1 0
0 1 12 24 3 52 6 1 1 0
1 0 18 10 1 2 65 2 1 0
3 0 10 1 18 10 2 52 1 3
13 1 3 0 2 0 0 2 71 8
5 20 0 1 0 1 0 1 8 64
70 60 50 40 30 20 10 0

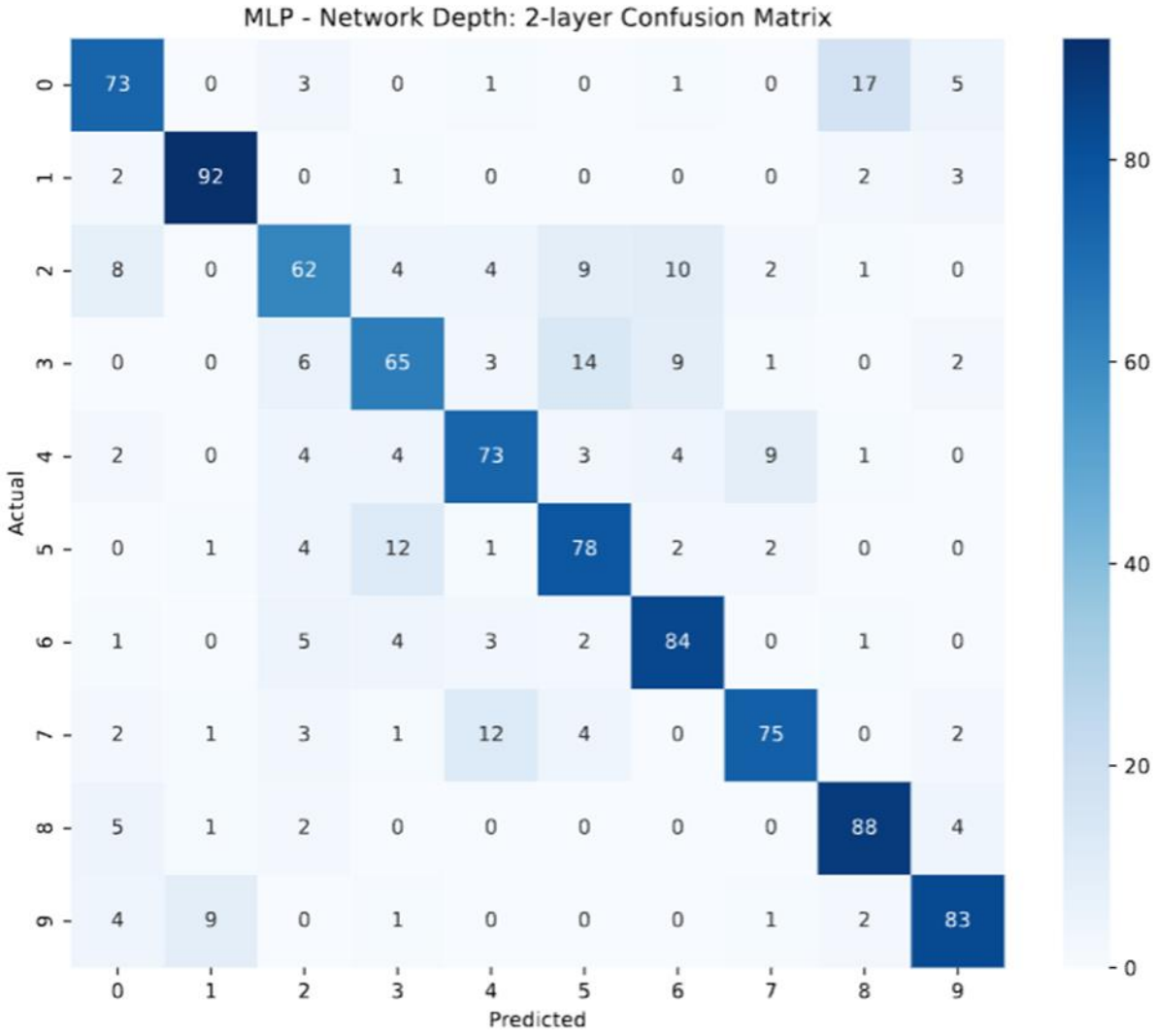

Decision Tree - Max Depth: 50 Confusion Matrix
Actual
Predicted
0 65 1 6 3 3 0 3 0 15 4
1 3 66 1 1 0 3 1 0 5 20
2 6 0 45 13 8 13 8 7 0 0
3 2 0 7 51 4 20 11 4 1 0
4 1 0 13 9 50 5 3 19 0 0
5 1 0 7 14 7 56 4 8 3 0
6 2 0 16 6 2 3 65 6 0 0
7 1 1 4 13 19 10 1 49 1 1
8 15 7 1 0 1 1 0 0 69 6
9 5 13 0 1 0 0 0 2 10 69
0 1 2 3 4 5 6 7 8 9
60 50 40 30 20 10 0

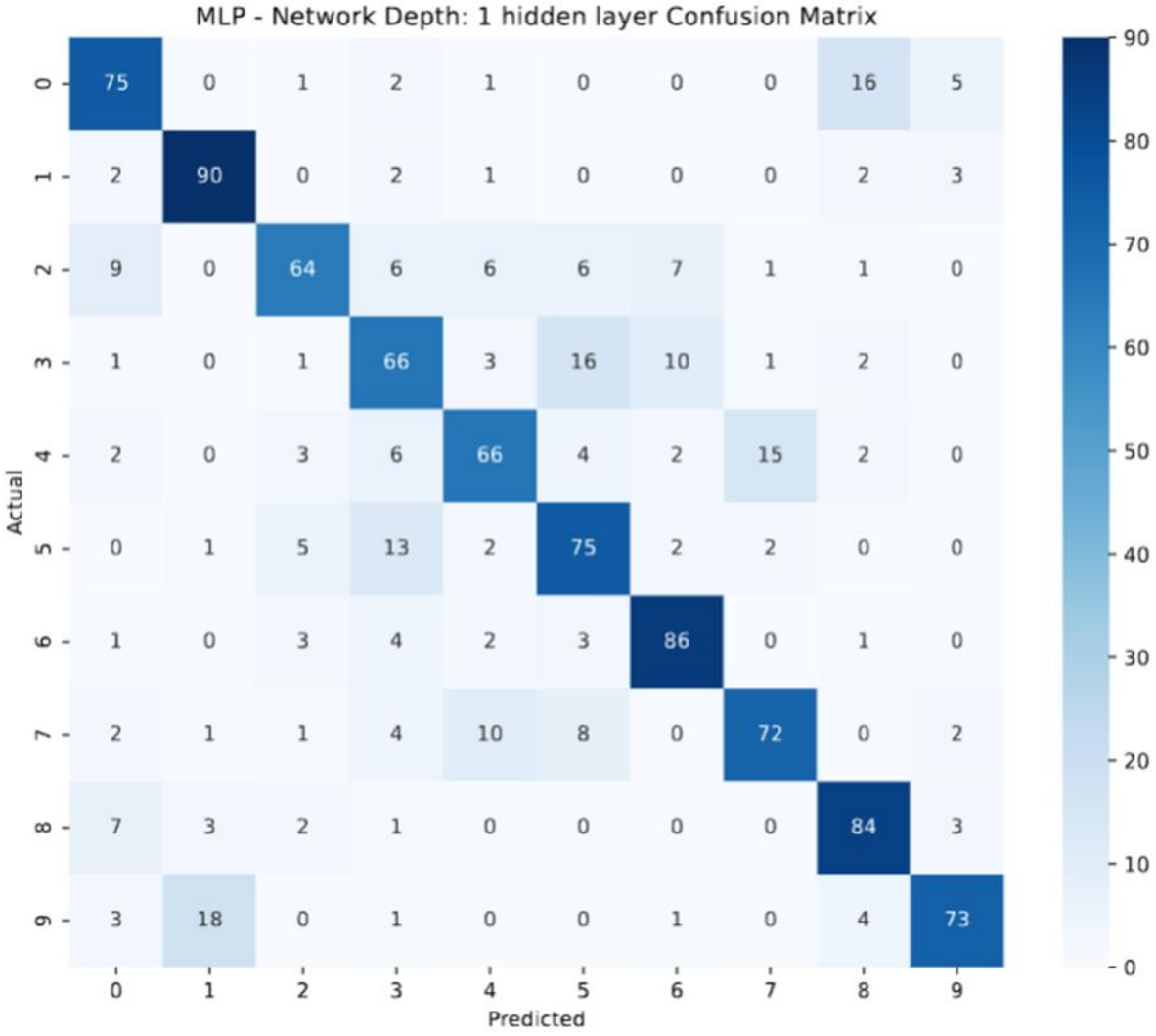

Decision Tree Scikit - Max Depth: 10 Confusion Matrix
Actual
Predicted
0 57 7 8 6 2 1 0 0 14 5
1 3 67 1 3 1 2 2 1 6 14
2 3 0 45 19 9 10 12 1 1 0
3 3 0 12 56 3 15 8 1 1 1
4 4 0 14 6 52 5 3 15 0 1
5 0 0 8 25 5 56 2 4 0 0
6 3 0 8 11 6 1 71 0 0 0
7 2 0 2 9 15 12 0 58 1 1
8 19 4 4 0 0 0 2 0 66 5
9 8 7 0 3 1 0 0 1 7 73
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0

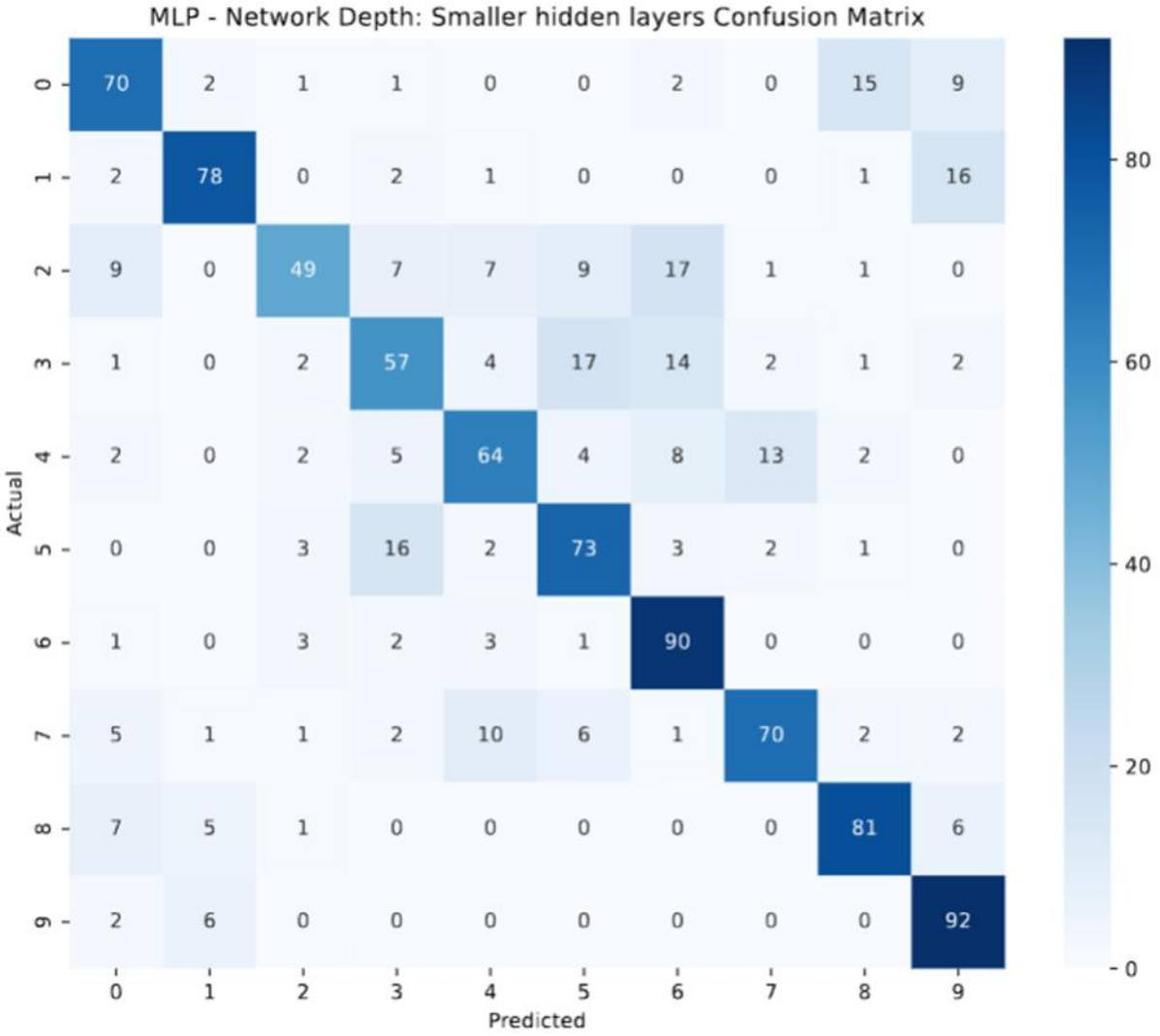

Decision Tree Scikit - Max Depth: 20 Confusion Matrix
Actual
Predicted
0 54 8 12 0 2 3 2 2 12 5
1 2 65 3 3 0 0 0 0 8 19
2 3 0 51 13 5 8 13 6 1 0
3 3 0 9 47 4 16 12 6 1 2
4 1 0 13 9 47 6 4 18 1 1
5 0 0 8 16 12 56 2 4 1 1
6 1 0 9 8 7 2 71 0 1 1
7 2 0 7 6 13 16 1 52 2 1
8 18 3 3 0 0 0 1 1 67 7
9 4 9 0 1 1 0 0 1 8 76
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0

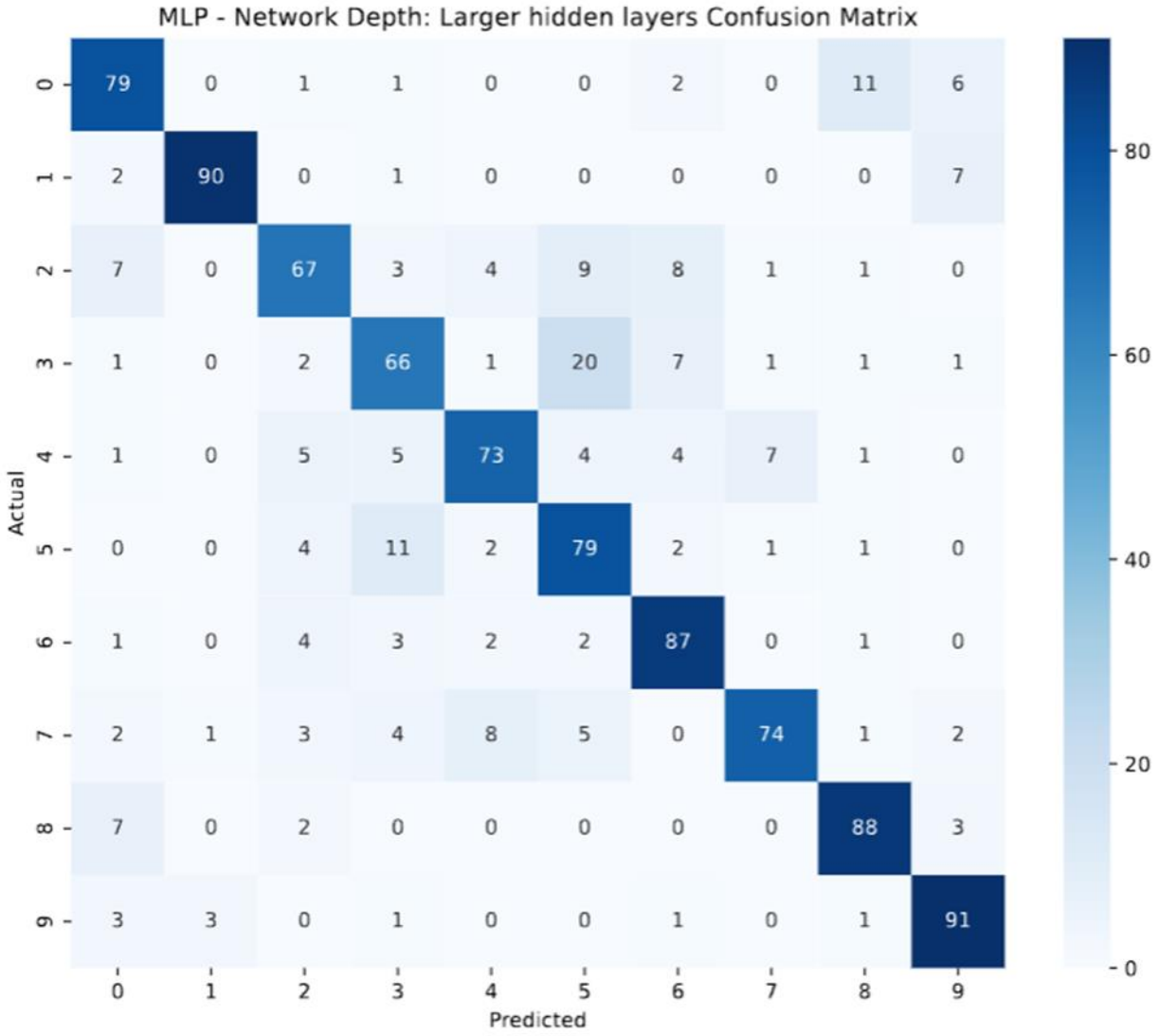


- **Multi-Layer Perceptron (MLP):**
  A larger architecture, meaning larger hidden layers achieved the highest accuracy of 79.4%, leveraging its capacity to capture complex patterns.
  The smaller network, meaning smaller hidden layers underperformed, with an accuracy of 72.4%.
  So we can see that in MLP, having more hidden layers would be more beneficial since complex patterns could be captured better compared to less hidden layers.


MLP - Network Depth: 2-layer Confusion Matrix
Actual
Predicted
0 1 2 3 4 5 6 7 8 9
73 0 3 0 1 0 1 0 17 5
2 92 0 1 0 0 0 0 2 3
8 0 62 4 4 9 10 2 1 0
0 0 6 65 3 14 9 1 0 2
2 0 4 4 73 3 4 9 1 0
0 1 4 12 1 78 2 2 0 0
1 0 5 4 3 2 84 0 1 0
2 1 3 1 12 4 0 75 0 2
5 1 2 0 0 0 0 0 88 4
4 9 0 1 0 0 0 1 2 83
0
20
40
60
80

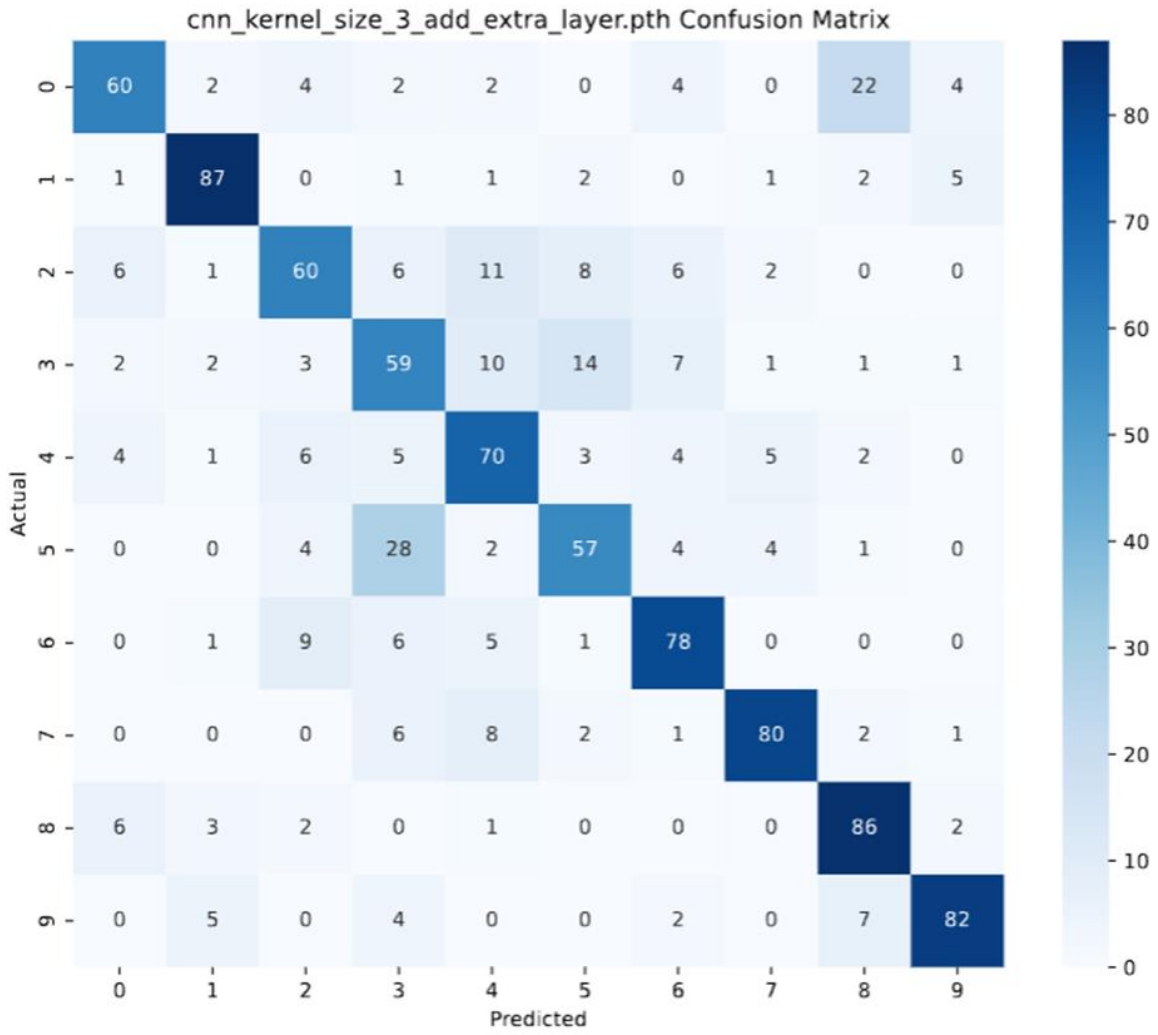

MLP - Network Depth: 1 hidden layer Confusion Matrix
Actual
Predicted
0 1 2 3 4 5 6 7 8 9
75 0 1 2 1 0 0 0 16 5
2 90 0 2 1 0 0 0 2 3
9 0 64 6 6 6 7 1 1 0
1 0 1 66 3 16 10 1 2 0
2 0 3 6 66 4 2 15 2 0
0 1 5 13 2 75 2 2 0 0
1 0 3 4 2 3 86 0 1 0
2 1 1 4 10 8 0 72 0 2
7 3 2 1 0 0 0 0 84 3
3 18 0 1 0 0 1 0 4 73
90 80 70 60 50 40 30 20 10 0

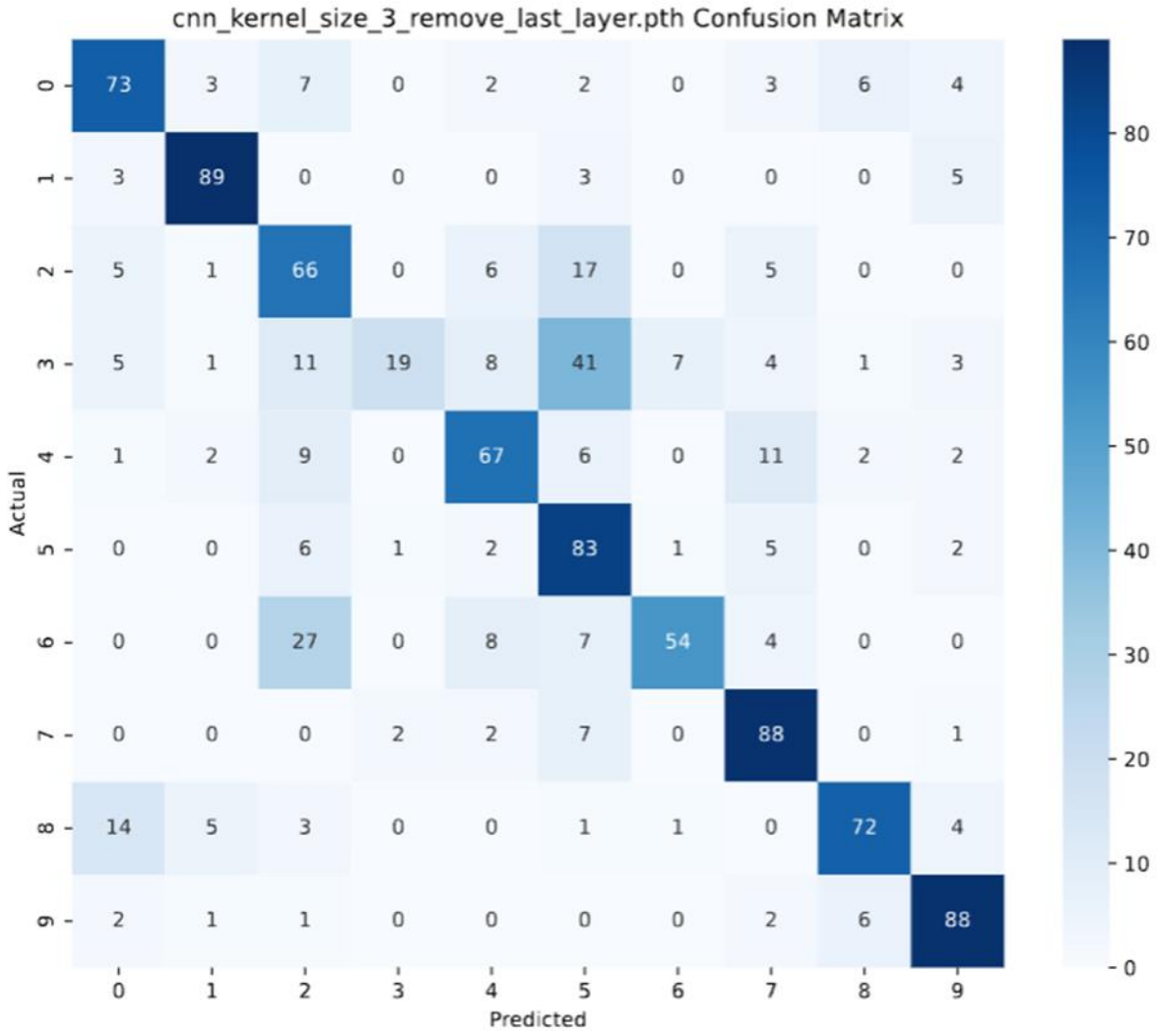

MLP - Network Depth: Smaller hidden layers Confusion Matrix
Actual
Predicted
0 1 2 3 4 5 6 7 8 9
70 2 1 1 0 0 2 0 15 9
2 78 0 2 1 0 0 0 1 16
9 0 49 7 7 9 17 1 1 0
1 0 2 57 4 17 14 2 1 2
2 0 2 5 64 4 8 13 2 0
0 0 3 16 2 73 3 2 1 0
1 0 3 2 3 1 90 0 0 0
5 1 1 2 10 6 1 70 2 2
7 5 1 0 0 0 0 0 81 6
2 6 0 0 0 0 0 0 0 92
0 20 40 60 80

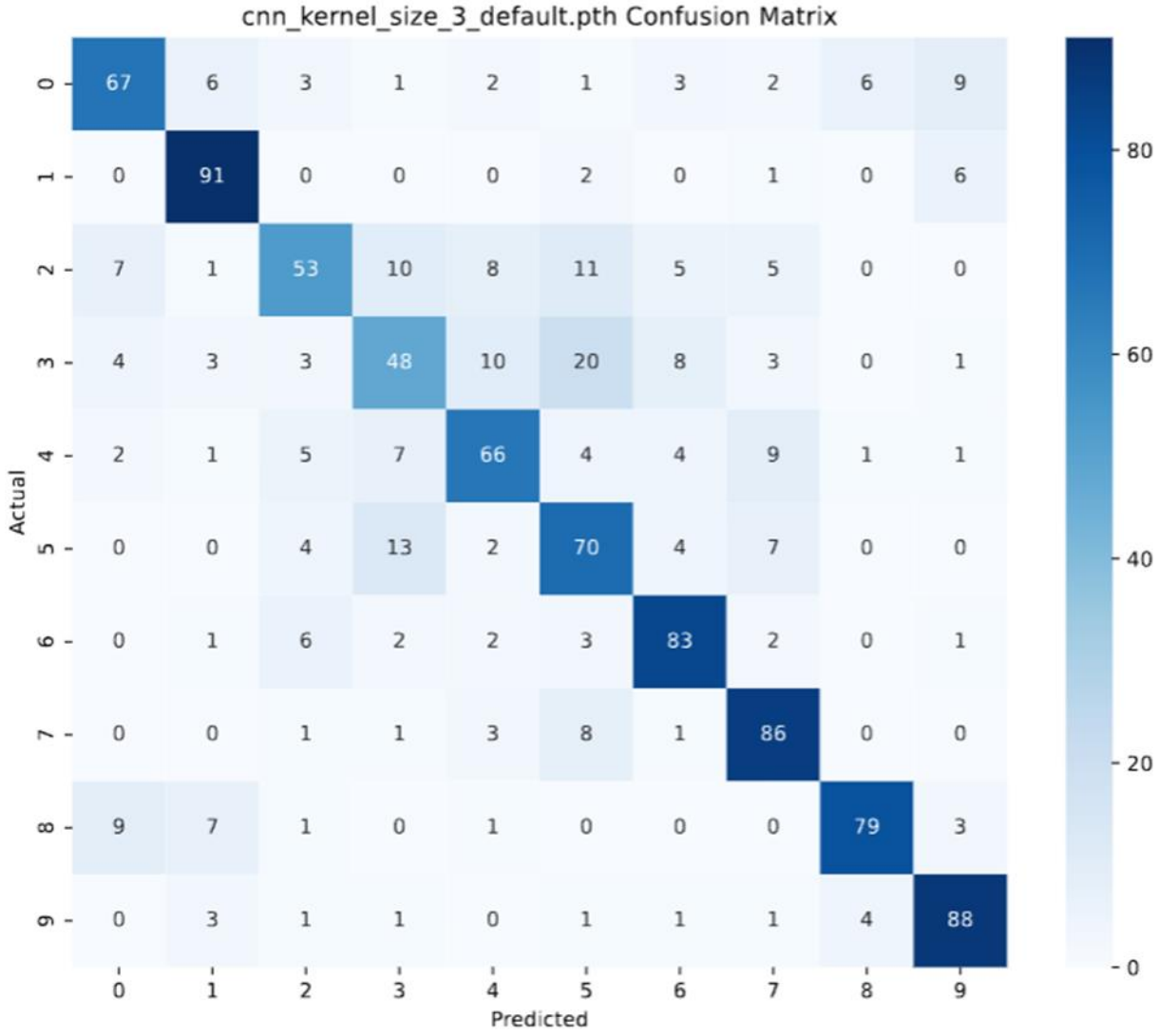


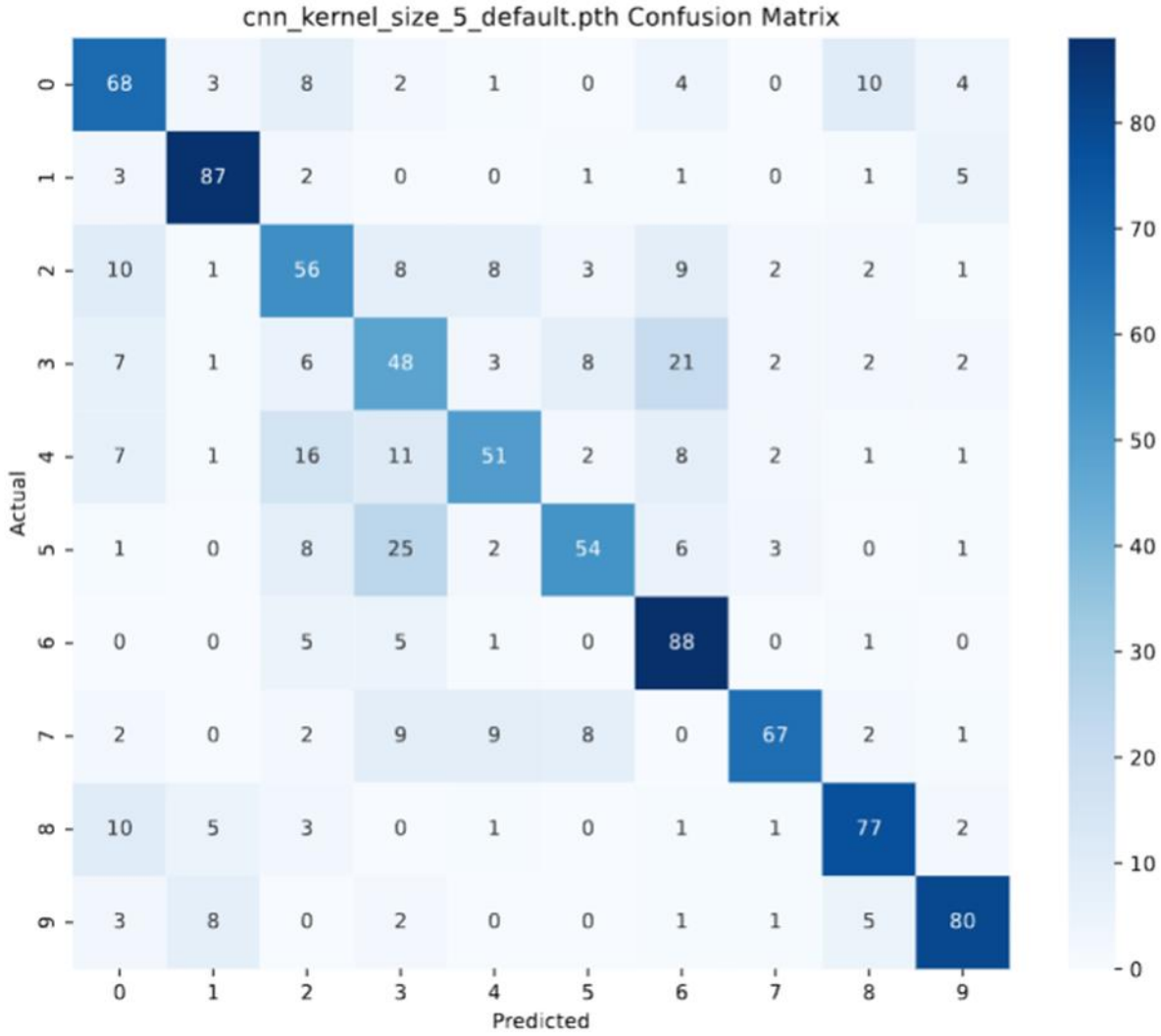
- **Convolutional Neural Network (CNN):**

  Using the default kernel size of 3, we noticed that it outperformed better overall, achieving 73.1% accuracy.
  When we added an extra convolutional layer, it caused the performance to drop slightly, while when we removed the last layer, the accuracy reduced notably to 69.9%.


cnn_kernel_size_3_add_extra_layer.pth Confusion Matrix
Actual
Predicted
0 60 2 4 2 2 0 4 0 22 4
1 1 87 0 1 1 2 0 1 2 5
2 6 1 60 6 11 8 6 2 0 0
3 2 2 3 59 10 14 7 1 1 1
4 4 1 6 5 70 3 4 5 2 0
5 0 0 4 28 2 57 4 4 1 0
6 0 1 9 6 5 1 78 0 0 0
7 0 0 0 6 8 2 1 80 2 1
8 6 3 2 0 1 0 0 0 86 2
9 0 5 0 4 0 0 2 0 7 82
0 1 2 3 4 5 6 7 8 9
80 70 60 50 40 30 20 10 0


cnn_kernel_size_3_remove_last_layer.pth Confusion Matrix
Actual
Predicted
0 1 2 3 4 5 6 7 8 9
73 3 7 0 2 2 0 3 6 4
3 89 0 0 0 3 0 0 0 5
5 1 66 0 6 17 0 5 0 0
5 1 11 19 8 41 7 4 1 3
1 2 9 0 67 6 0 11 2 2
0 0 6 1 2 83 1 5 0 2
0 0 27 0 8 7 54 4 0 0
0 0 0 2 2 7 0 88 0 1
14 5 3 0 0 1 1 0 72 4
2 1 1 0 0 0 0 2 6 88
0 10 20 30 40 50 60 70 80


cnn_kernel_size_3_default.pth Confusion Matrix
Actual
Predicted
0 67 6 3 1 2 1 3 2 6 9
1 0 91 0 0 0 2 0 1 0 6
2 7 1 53 10 8 11 5 5 0 0
3 4 3 3 48 10 20 8 3 0 1
4 2 1 5 7 66 4 4 9 1 1
5 0 0 4 13 2 70 4 7 0 0
6 0 1 6 2 2 3 83 2 0 1
7 0 0 1 1 3 8 1 86 0 0
8 9 7 1 0 1 0 0 0 79 3
9 0 3 1 1 0 1 1 1 4 88
0 1 2 3 4 5 6 7 8 9
80
60
40
20
0



4. Confusion Matrix Insight:

The confusion matrices for all models revealed common misclassifications:

- Naive Bayes struggles with classes that have overlapping feature distributions, such as classes 3(cats), and 5(dogs).
- Decision trees showed an improved performance compared to naive bayes at certain classes when we had a lower depth, but when we were overfitting by having larger depths, the precision was reduced for some classes.
- MLP consistently performed better for classes that had distinct feature patterns, for example 0(airplane) and 1(automobile), due to its ability to recognize complex patterns [2].

- CNN showed its ability to capture spatial relationships, performing well on classes like 0(airplanes) and 9(truck). Unfortunately, it was struggling when it came to classes that also had similar texture such as 3(cats) and 5(dogs)[1].

4.1 Frequently confused classes

- Models like Naive Bayes and Decision trees misclassified animals, such as cats, dogs, and horses, and that is due to overlapping features in those images.

- CNN often misclassified classes where objects shared the same textures.

4.2 Well Recognized classes

Airplanes (0), automobiles (1), and trucks (9) were consistently recognized in all models, their distinct features, as well as not having overlapping features might be the reason that they had a high success rate.

4.3 implications in the context of facial image analysis

The CNN's ability to capture spatial hierarchies and patterns makes it well suited for facial image recognition [3]. The MLP model is usually not well suited for spatial hierarchies, so it would not be very good. Decision Trees have a low precision, and recall, so it might not be ideal for facial recognition, since it tends to overfit. Naive bayes assumes feature independance, and that would limit its effectiveness in capturing the interdependencies of the facial features

4.4 Influence of parameters:

- Depth: In decision trees, increasing the depth of the tree resulted in overfitting since we had more complex feature splits. MLPs were benefiting from the increased depth, which let it perform better. On the other hand, CNNs improved slightly, but ended up overfitting.

- Layer Size: Having larger layers helped the MLP model to perform better overall. Smaller hidden layers were not able to detect complex patterns, and that lowered the accuracy of the model.

- Kernel Size: The default kernel size given to us was 3, and it provided good results overall, larger kernel size was able to capture broader features, but it diminished the performance.

4.5 Summary of Findings:

Based on the results we have so far, the MLP with larger hidden layers performed the best. The Naive Bayes model was limited by the fact that we assumed feature independence, and the Decision Tree had somewhat not very good results when the depth was increased, due to overfitting. The CNN did not perform as good as the naive baye or MLP, the reason could be related to hyperparameters, or the kernel size. Extensive testing and parameter tuning for the CNN would be needed to optimize the model.

References:

[1] T. Kalra, “Why Traditional CNNs May Fail for Texture-Based Classification,” *Medium*, May 1, 2020. [Online]. Available: https://medium.com/@trapti.kalra/why-traditional-cnns-may-fail-for-texture-based-classification-3b49d6b94b6f. [Accessed: Nov. 25, 2024].

[2] “MLP vs. CNN: A Simple Comparison,” *Peculiar Coding Endeavours*, Nov. 2019. [Online]. Available: https://www.peculiar-coding-endeavours.com/2019/mlp_vs_cnn/?utm_source=chatgpt.com. [Accessed: 25-Nov-2024].

[3] Deepgram. “Convolutional Neural Networks (CNNs).” *AI Glossary*. Available: https://deepgram.com/ai-glossary/convolutional-neural-networks. [Accessed: Nov. 25, 2024].