COMP 472
Artificial Intelligence

Final Project

Presented by :

Louisa-Lina Meziane
(40133119)

Presented to :

Dr. Kefaya Qaddoum

Fall 2024
November 25th, 2024

Table of Content

Introduction

This project involves the design, training and evaluation of four AI models: Gaussian Naive Bayes, Decision Tree, Multi-layer Perceptron and Convolutional Neural Network. In the context of this project, it was requested to classify the images using the CIFAR-10 dataset: a benchmark compromising low-resolution images across ten categories: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.

The primary aim is to investigate the strengths and limitations of these models. To do so it was requested to evaluate their performance through metrics such as accuracy, precision, recall, and F1 score, and examine the effects of variations in architecture and parameters. The project integrates advanced practices, including feature extraction with a pre-trained ResNet-18 model, while exploring different model variants namely: Gaussian Naive Bayes, Decision Tree, Multi-layer Perceptron and convolutional Neural Network. This approach not only has a goal to improve classification itself but also mirrors the practical challenges encountered in real-world AI development.

A GitHub repository is used to further showcase the work here is the link Github link to access it: https://github.com/louisa-lina/AIModelsClassification

Model Architectures and Training

- Highlight the changes made for variants of each model, specifically noting how each deviates from the main model.

1. Naive Bayes

- **Main Model:** Custom implementation
  - Manually computes class priors, means, and variances.
  - Logarithmic computation of prior probabilities and Gaussian likelihood for predictions.

- **Variant:** Scikit-learn's GaussianNB
  - Uses Scikit-learn’s built-in implementation for Gaussian Naive Bayes.
  - Optimized for computational efficiency and numerical stability.

2. Decision Tree

- **Main Model:** Custom implementation
  - Implements Gini coefficient-based splitting.
  - Recursively builds the tree up to a maximum depth (default 50).
  - Custom logic for determining the best split and prediction.

- **Variant:** Scikit-learn's DecisionTreeClassifier.
  - Scikit-learn provides optimized and prebuilt tree-splitting and evaluation methods.
  - Allows exploration of hyperparameters (e.g., depth) without implementing custom logic.

3. Multi-Layer Perceptron (MLP)

- **Base Model:** 3-layer architecture (Linear → ReLU → Linear → BatchNorm → ReLU → Linear).

- **Variants:** Adjustments in-depth and hidden layer size to create:
    - **Deeper MLP:** Adds more hidden layers, increasing the model's capacity.
    - **Shallower MLP:** Reduces the number of hidden layers, simplifying the model.
    - **Larger Hidden MLP:** Increases the size of the hidden layers for higher capacity at the cost of computational complexity.
    - **Smaller Hidden MLP:** Decreases the size of the hidden layers to reduce model complexity and training time.

4. Convolutional Neural Network (CNN) - VGG11

- **Base Model:** Standard VGG11 architecture
    - Predefined convolutional layers with 3x3 kernels and batch normalization.
    - Fully connected layers for classification with dropout for regularization.

- **Variants:**
    - **Extra Layers (+2 and +4):** Additional convolutional layers appended to the base architecture, increasing depth.
    - **Custom Kernel Sizes (3x3, 5x5):** Adjusts kernel sizes in convolutional layers, affecting spatial granularity and computational cost.

- Detail the training methodology: number of epochs, learning rate, loss function used, and other relevant training hyperparameters

1. **Naive Bayes**
    - **Custom:** No training-specific hyperparameters (direct computation of class priors, means, and variances).

- **Scikit-learn:** Built-in hyperparameters managed internally.

2. **Decision Tree**
    - **Custom:**
        - **Max Depth:** Varied (10, 20, 30, 40, 50).
    - **Scikit-learn:**
        - **Max Depth:** Same as the custom model for comparison.

3. **Multi-Layer Perceptron (MLP)**
    - **Epochs:** 20.
    - **Learning Rate:** 0.01.
    - **Optimizer:** SGD with momentum = 0.9.
    - **Loss Function:** CrossEntropyLoss.
    - **Batch Size:** Entire dataset converted to tensors for direct computation.

4. **Convolutional Neural Network (CNN) - VGG11**
    - **Epochs:** 20.
    - **Learning Rate:** 0.01.
    - **Optimizer:** SGD with momentum = 0.9.
    - **Loss Function:** CrossEntropyLoss.
    - **Batch Size:** 64 (using DataLoader for batches).
    - **Additional Settings for Variants:**
        - **Extra Layers:** Increases depth by appending additional convolutional layers.
        - **Kernel Sizes:** Adjusted to 3x3 or 5x5 for evaluating spatial granularity.

- Mention any optimization algorithms or techniques used, like mini-batch gradient descent, Adam optimizer, etc.

Optimization Algorithms and Techniques Used

1. **Naive Bayes**
    - No optimization algorithms were used (direct computation of priors, means, and variances).

2. **Decision Tree**
    - No optimization algorithms were used (recursive tree-building with Gini coefficient).

3. **Multi-Layer Perceptron (MLP)**
    - **Optimizer:** SGD with momentum = 0.9.
    - **Technique:** Batch gradient descent (entire dataset as tensors).

4. **Convolutional Neural Network (CNN) - VGG11**
    - **Optimizer:** SGD with momentum = 0.9.
    - **Technique:** Mini-batch gradient descent (batch size = 64).

Evaluation

Training

Model	Accuracy	Precision	Recall	F1-Measure
Naive Bayes Custom	0.8162	0.8192	0.8162	0.8165
Naive Bayes Scikit-learn	0.8162	0.8192	0.8162	0.8165
Decision Tree Custom (10)	0.8822	0.8883	0.8822	0.8838
Decision Tree Custom (20)	1	1	1	1
Decision Tree Custom (30)	1	1	1	1
Decision Tree Custom (40)	1	1	1	1
Decision Tree Custom (50)	1	1	1	1
Decision Tree Scikit-learn (10)	0.8822	0.8886	0.8822	0.8838
Decision Tree Scikit-learn (20)	1	1	1	1
Decision Tree Scikit-learn (30)	1	1	1	1
Decision Tree Scikit-learn (40)	1	1	1	1
Decision Tree Scikit-learn (50)	1	1	1	1
MLP Base Model	0.8004	0.8004	0.8004	0.7986
MLP Deeper	0.8180	0.8183	0.8180	0.8170
MLP Shallower	0.7882	0.7902	0.7882	0.7879
MLP Larger Hidden	0.8256	0.8254	0.8256	0.8247
MLP Smaller Hidden	0.7736	0.7739	0.7736	0.7712
CNN Base Model	0.8046	0.8292	0.8046	0.8034
CNN Extra Layers (+2)	0.9314	0.9353	0.9314	0.9310
CNN Extra Layers (+4)	0.9270	0.9317	0.9270	0.9256
CNN Custom Kernel Sizes (3x3)	0.7574	0.7981	0.7574	0.7502
CNN Custom Kernel Sizes (5x5)	0.7716	0.8031	0.7716	0.7684

Testing

Model	Accuracy	Precision	Recall	F1-Measure
Naive Bayes Custom	0.7920	0.7974	0.7920	0.7925
Naive Bayes Scikit-learn	0.7920	0.7974	0.7920	0.7925
Decision Tree Custom (10)	0.5990	0.6099	0.5990	0.6023
Decision Tree Custom (20)	0.5820	0.5831	0.5820	0.5815
Decision Tree Custom (30)	0.5820	0.5831	0.5820	0.5815
Decision Tree Custom (40)	0.5820	0.5831	0.5820	0.5815
Decision Tree Custom (50)	0.5820	0.5831	0.5820	0.5815
Decision Tree Scikit-learn (10)	0.5980	0.6084	0.59802	0.6012
Decision Tree Scikit-learn (20)	0.5690	0.5716	0.5690	0.5690
Decision Tree Scikit-learn (30)	0.5880	0.5926	0.5880	0.5896
Decision Tree Scikit-learn (40)	0.5810	0.5834	0.5810	0.5810
Decision Tree Scikit-learn (50)	0.5760	0.5794	0.5760	0.5766
MLP Base Model	0.7770	0.7827	0.7770	0.7756
MLP Deeper	0.7790	0.7832	0.7790	0.7782
MLP Shallower	0.7680	0.7731	0.7680	0.7674
MLP Larger Hidden	0.7990	0.8047	0.7990	0.7986
MLP Smaller Hidden	0.7430	0.7434	0.7430	0.7403
CNN Base Model	0.6120	0.6587	0.6120	0.6140
CNN Extra Layers (+2)	0.6560	0.6576	0.6560	0.6449
CNN Extra Layers (+4)	0.6280	0.6432	0.6280	0.6190
CNN Custom Kernel Sizes (3x3)	0.6030	0.6395	0.6030	0.5936
CNN Custom Kernel Sizes (5x5)	0.6070	0.6274	0.6070	0.6010

- Offer insights into each model's performance relative to the others. For instance, if one model has a higher recall but lower precision, discuss its implications in the context of facial image analysis.

1. **Naive Bayes**
    - **Training vs. Testing:** Comparable metrics in both phases (~0.79 accuracy in testing).
    - **Performance:** Balanced precision and recall indicate stable generalization.
    - **Implication:** Performs well for datasets with distinct class features but struggles with overlapping classes.

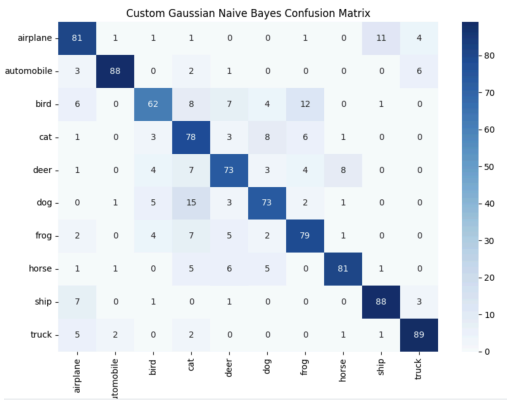
2. **Decision Tree**
    - **Custom vs. Scikit-learn:** Similar performance with both implementations.
    - **Overfitting:** Near-perfect training metrics at depth ≥20 but low testing accuracy (~0.58), indicating overfitting.
    - **Implication:** Poor generalization for complex, high-dimensional data like images.

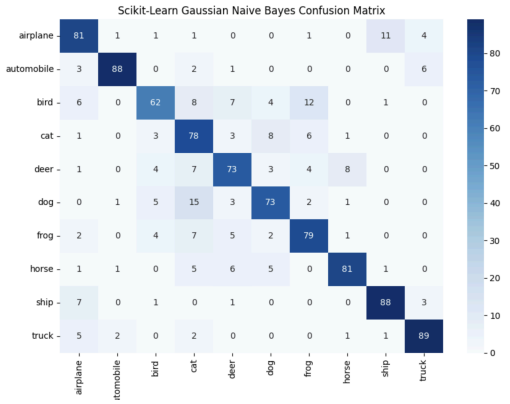
3. **MLP**
    - **Base Model:** Moderate testing accuracy (~0.77) with balanced precision and recall.
    - **Variants:**
        - **Deeper MLP:** Slight improvement in accuracy (~0.78) and recall, indicating better feature learning.
        - **Shallower MLP:** Lower accuracy (~0.76), reflecting insufficient capacity.
        - **Larger Hidden:** Best testing accuracy (~0.80), highlighting the benefits of higher capacity.
        - **Smaller Hidden:** Worst testing performance (~0.74), demonstrating limited capacity.
    - **Implication:** Deeper architectures with larger hidden layers generalize better for image data.

4. **CNN (VGG11)**
    - **Base Model:** Testing accuracy (~0.61) is relatively low, but precision > recall indicates a focus on fewer, more confident predictions.
    - **Variants:**
        - **Extra Layers:** Increased depth (e.g., +2 layers) improves testing accuracy (~0.66) but not recall, favouring confident predictions.
        - **Kernel Sizes:** 3x3 and 5x5 kernels perform similarly (~0.60 testing accuracy), with smaller kernels showing slightly higher recall.
    - **Implication:** Additional layers improve feature extraction, while kernel size adjustments have minimal impact on high-dimensional data.

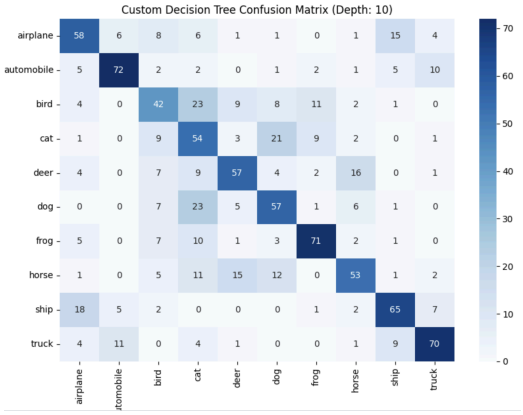
- Display the confusion matrices for each model.

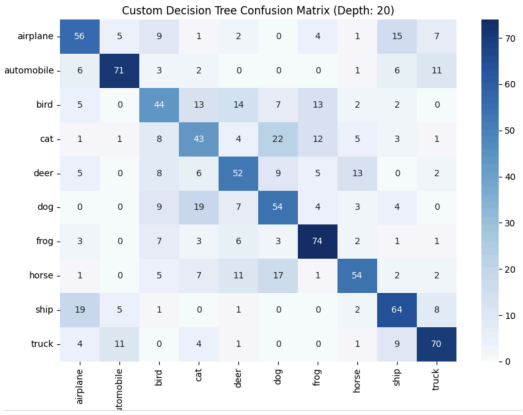
Naive Bayes:

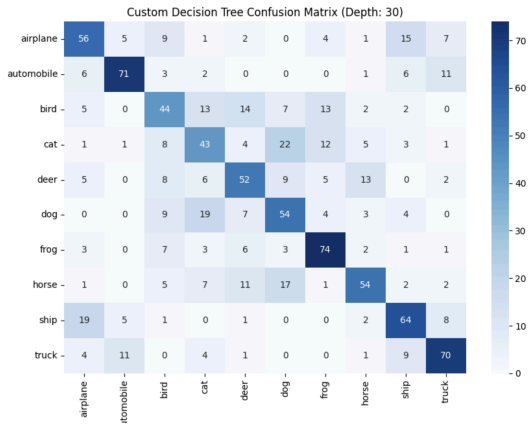


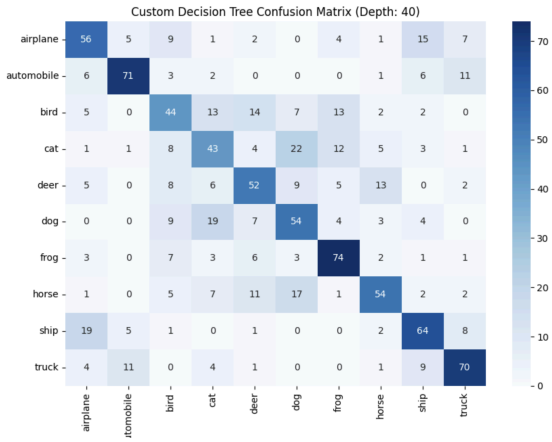


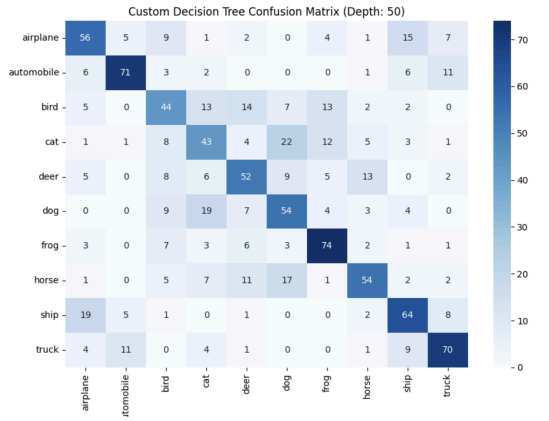
Decision Tree:

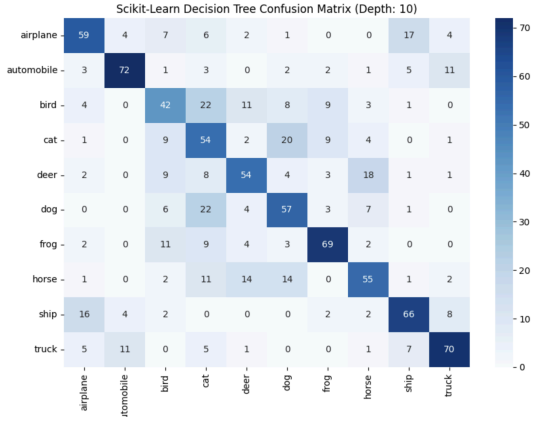


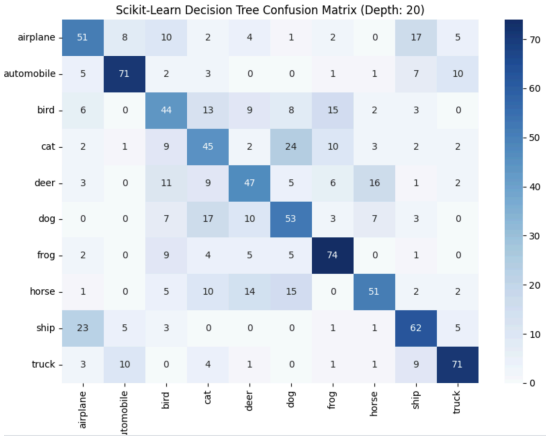


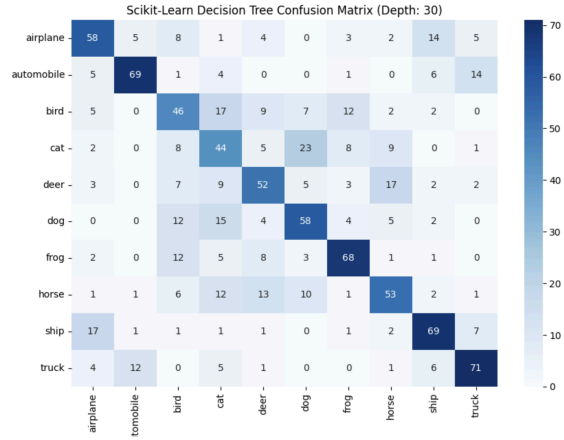


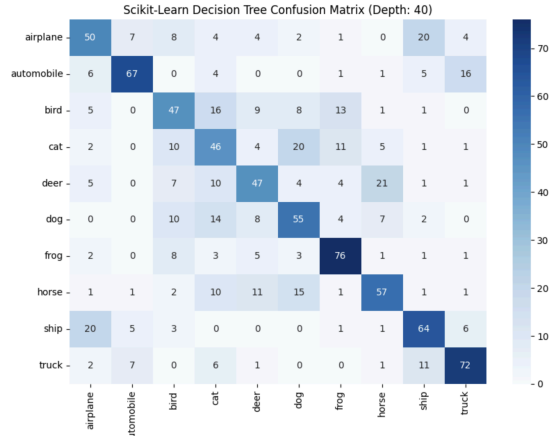


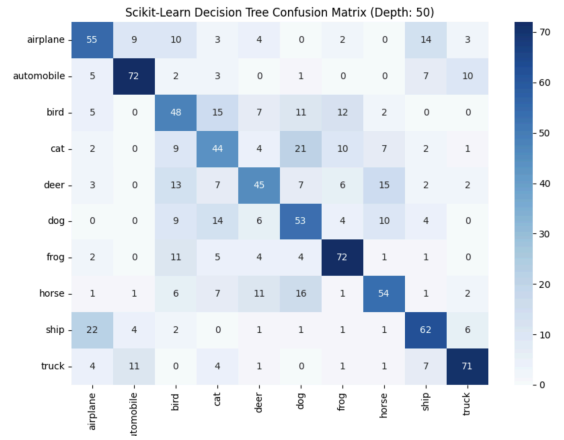

Custom Decision Tree Confusion Matrix (Depth: 50)
airplane 56 5 9 1 2 0 4 1 15 7
automobile 6 71 3 2 0 0 0 1 6 11
bird 5 0 44 13 14 7 13 2 2 0
cat 1 1 8 43 4 22 12 5 3 1
deer 5 0 8 6 52 9 5 13 0 2
dog 0 0 9 19 7 54 4 3 4 0
frog 3 0 7 3 6 3 74 2 1 1
horse 1 0 5 7 11 17 1 54 2 2
ship 19 5 1 0 1 0 0 2 64 8
truck 4 11 0 4 1 0 0 1 9 70
airplane automobile bird cat deer dog frog horse ship truck
70 60 50 40 30 20 10 0


Scikit-Learn Decision Tree Confusion Matrix (Depth: 10)
airplane 59 4 7 6 2 1 0 0 17 4
automobile 3 72 1 3 0 2 2 1 5 11
bird 4 0 42 22 11 8 9 3 1 0
cat 1 0 9 54 2 20 9 4 0 1
deer 2 0 9 8 54 4 3 18 1 1
dog 0 0 6 22 4 57 3 7 1 0
frog 2 0 11 9 4 3 69 2 0 0
horse 1 0 2 11 14 14 0 55 1 2
ship 16 4 2 0 0 0 2 2 66 8
truck 5 11 0 5 1 0 0 1 7 70
airplane automobile bird cat deer dog frog horse ship truck
70 60 50 40 30 20 10 0

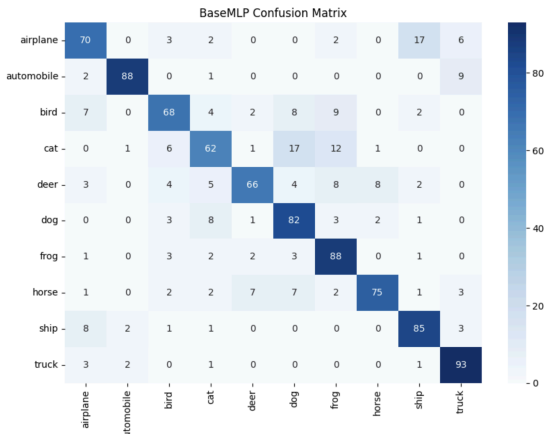

Scikit-Learn Decision Tree Confusion Matrix (Depth: 20)
airplane 51 8 10 2 4 1 2 0 17 5
automobile 5 71 2 3 0 0 1 1 7 10
bird 6 0 44 13 9 8 15 2 3 0
cat 2 1 9 45 2 24 10 3 2 2
deer 3 0 11 9 47 5 6 16 1 2
dog 0 0 7 17 10 53 3 7 3 0
frog 2 0 9 4 5 5 74 0 1 0
horse 1 0 5 10 14 15 0 51 2 2
ship 23 5 3 0 0 0 1 1 62 5
truck 3 10 0 4 1 0 1 1 9 71
airplane automobile bird cat deer dog frog horse ship truck
70 60 50 40 30 20 10 0

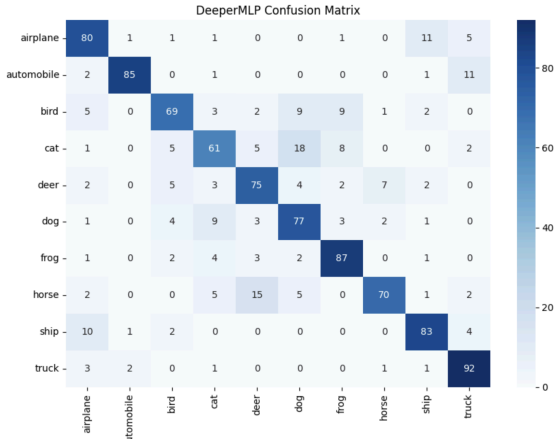

Scikit-Learn Decision Tree Confusion Matrix (Depth: 30)
airplane 58 5 8 1 4 0 3 2 14 5
automobile 5 69 1 4 0 0 1 0 6 14
bird 5 0 46 17 9 7 12 2 2 0
cat 2 0 8 44 5 23 8 9 0 1
deer 3 0 7 9 52 5 3 17 2 2
dog 0 0 12 15 4 58 4 5 2 0
frog 2 0 12 5 8 3 68 1 1 0
horse 1 1 6 12 13 10 1 53 2 1
ship 17 1 1 1 1 0 1 2 69 7
truck 4 12 0 5 1 0 0 1 6 71
airplane automobile bird cat deer dog frog horse ship truck
70 60 50 40 30 20 10 0

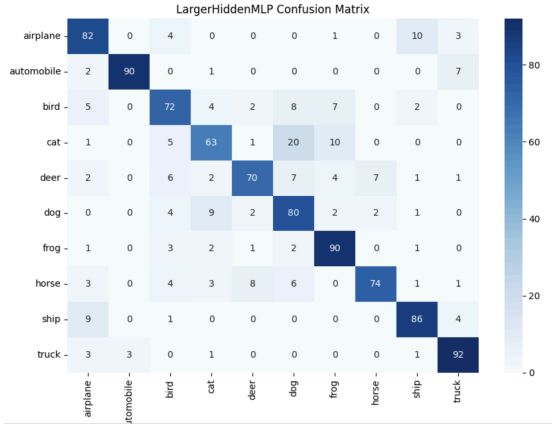

Scikit-Learn Decision Tree Confusion Matrix (Depth: 40)
airplane 50 7 8 4 4 2 1 0 20 4
automobile 6 67 0 4 0 0 1 1 5 16
bird 5 0 47 16 9 8 13 1 1 0
cat 2 0 10 46 4 20 11 5 1 1
deer 5 0 7 10 47 4 4 21 1 1
dog 0 0 10 14 8 55 4 7 2 0
frog 2 0 8 3 5 3 76 1 1 1
horse 1 1 2 10 11 15 1 57 1 1
ship 20 5 3 0 0 0 1 1 64 6
truck 2 7 0 6 1 0 0 1 11 72
airplane automobile bird cat deer dog frog horse ship truck
70 60 50 40 30 20 10 0

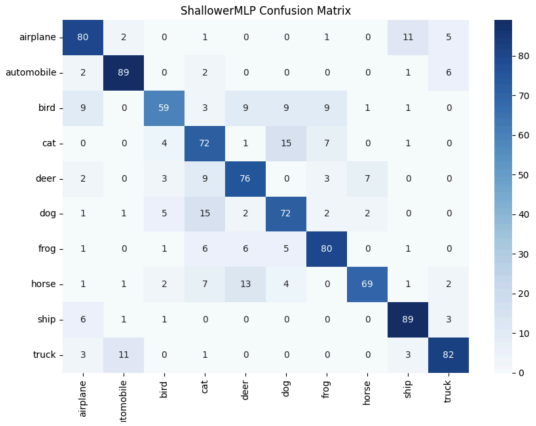

Scikit-Learn Decision Tree Confusion Matrix (Depth: 50)
airplane 55 9 10 3 4 0 2 0 14 3
automobile 5 72 2 3 0 1 0 0 7 10
bird 5 0 48 15 7 11 12 2 0 0
cat 2 0 9 44 4 21 10 7 2 1
deer 3 0 13 7 45 7 6 15 2 2
dog 0 0 9 14 6 53 4 10 4 0
frog 2 0 11 5 4 4 72 1 1 0
horse 1 1 6 7 11 16 1 54 1 2
ship 22 4 2 0 1 1 1 1 62 6
truck 4 11 0 4 1 0 1 1 7 71
airplane automobile bird cat deer dog frog horse ship truck
70 60 50 40 30 20 10 0

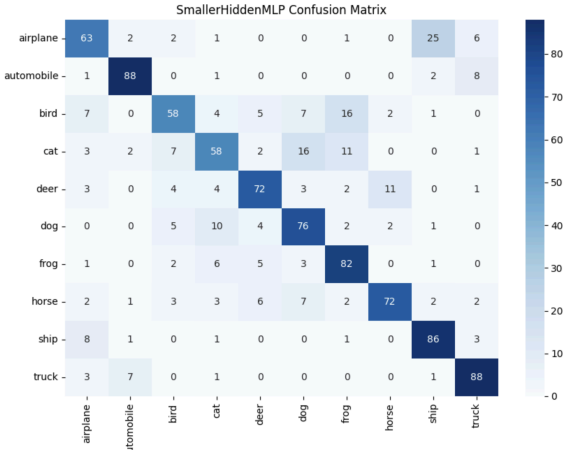
Multi-Layer Perceptron:



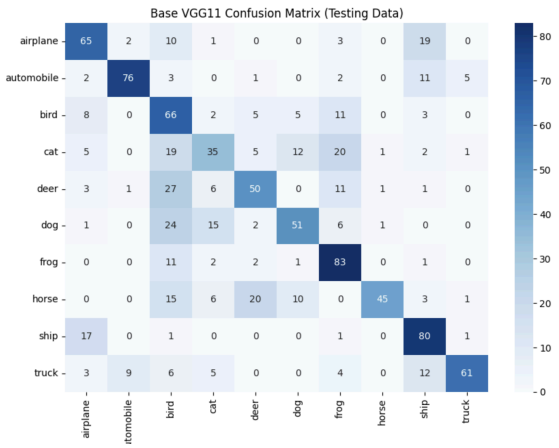


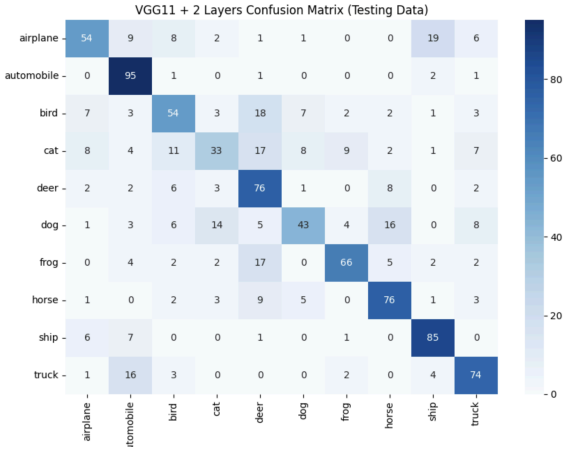


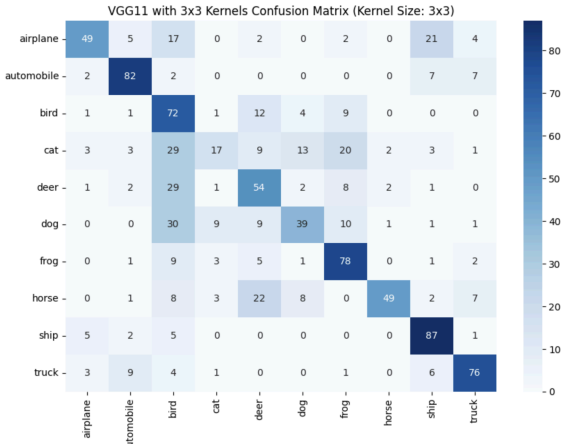


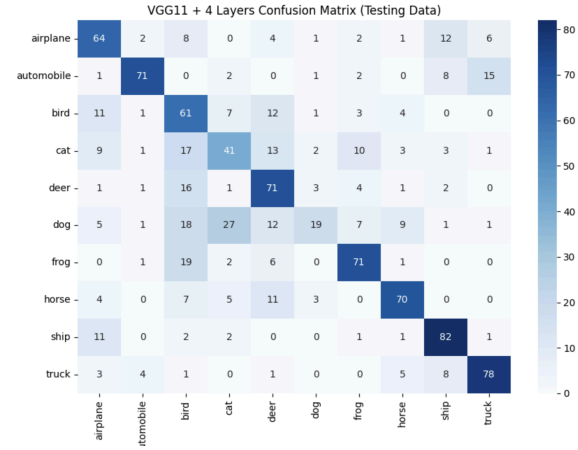


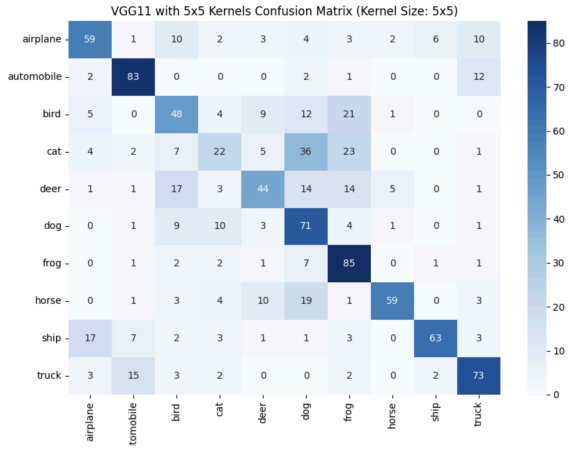
Convolutional Neural Network:











- Identify which classes were most frequently confused and discuss any model-specific reasons that might be behind this misclassification.

1. **Naive Bayes (Custom & Scikit-learn):**

   Both Naive Bayes implementations classified 62 images correctly, but they frequently misclassified Frog and Deer (4 times each), Dog (5 times), and Cat (3 times). The model's assumption of feature independence likely caused these misclassifications, as it struggles to account for the interdependence of features in animal classes, which often share overlapping characteristics like fur texture or similar body shapes.

2. **Decision Tree (Custom & Scikit-learn):**

   For a depth of 10, the Custom Decision Tree correctly identified Bird only 42 times, frequently misclassifying it as Cat, Deer, Dog, Frog, Horse, and occasionally Ship. Similarly, Horse was correctly classified 53 times, with misclassifications spread across Deer, Dog, Frog, Cat, and Bird. At greater depths (20, 30, 40, and 50), Cat and Bird were the most misclassified, with Cat correctly identified 43 times and Bird 44 times, often confused with Deer, Dog, Frog, and Horse. The Scikit-learn Decision Tree followed a similar trend, with Bird being the lowest correctly identified class at 42 for depth 10 and 44 for greater depths. The frequent misclassification derives from the tree's reliance on simple decision boundaries, which are insufficient for distinguishing visually similar animal classes

3. **MLP Variants:**

   In the Base MLP, Cat was correctly classified 62 times but misclassified as Dog (8 times), Deer (5 times), Bird (4 times), and others to a lesser extent. The Deeper MLP slightly reduced misclassification, with Cat correctly identified 61 times, but still frequently confused with Dog (9 times) and Deer (3 times). The Larger Hidden MLP performed best, correctly classifying Cat 63 times, while still confusing it with Dog (9

times) and Bird (4 times). In contrast, the Smaller Hidden MLP showed reduced performance, correctly identifying Cat 58 times and frequently misclassifying it as Dog (10 times), Frog (6 times), and Bird (4 times). The frequent confusion among animal classes suggests that smaller models lack the capacity to effectively learn detailed features, while larger and deeper models mitigate but do not entirely eliminate these challenges.

4. **CNN Variants (VGG11 and Modifications):**

The Base VGG11 model had Cat as the lowest-performing class, with only 35 correct classifications and frequent misclassifications such as Deer, Dog, Frog, and Horse. Horse was the second lowest, correctly identified 45 times but confused with Bird, Cat, Deer, and Dog. With +2 Layers, Cat was identified correctly 33 times, misclassified with Bird, Deer, Dog, and Frog, while Dog was the second lowest with 43 correct identifications and frequent misclassifications as Bird, Cat, and Deer. With +4 Layers, the performance further degraded for Dog, which was correctly identified only 19 times, and Cat correctly identified 41 times but was often confused with Bird, Deer, and Frog. For models with 3x3 and 5x5 kernels, Cat consistently remained the lowest-performing class, correctly identified 17 and 22 times, respectively. Despite the improvements in feature extraction with additional layers and kernel size adjustments, the CNNs struggled to distinguish visually similar animal classes due to shared textures and features.

- Highlight well-recognized classes and speculate on the reasons behind their success.

Across all models, certain classes such as Automobile, Ship, Truck, and Frog consistently exhibited higher recognition accuracy, though specific performance varied between models.

For Gaussian Naive Bayes:

Both the Custom and Scikit-learn implementations recognized Automobile (88), Ship (88), and Truck (89) correctly. These classes are likely well-recognized due to their distinct global features, such as consistent shapes and textures, which align well with Naive Bayes' assumptions of feature independence.

In Decision Tree models:

Automobile, Frog, and Truck were the most accurately identified classes, with accuracy ranging between 72-74 for the Custom implementation across various depths and 69-79 for the Scikit-learn version. These classes likely have distinguishable decision boundaries due to their distinct features, such as the smooth outlines of automobiles and trucks. For the frog, it is the only animal that distinguishes itself by its amphibian properties: it does not have fur or feathers which clearly distinguishes itself from the other animal classes.

MLP models:

Also showed strong recognition for Automobile, Frog, and Truck, with the Base MLP achieving high accuracy: Automobile (88), Frog (88), and Truck (93). Variants like Larger Hidden MLP improved on this, with Automobile (90), Frog (90), and Truck (92). The MLP's ability to capture mid-level patterns, such as outlines and edges, explains its effectiveness in recognizing these structured objects.

For CNNs:

The recognition accuracy varied by architecture, but specific classes like Automobiles, Ship, and Frog were consistently well-identified. The Base VGG11 identified Frog (83) and Ship (80) accurately, while VGG11 + 2 layers significantly improved Automobile (95) and Ship (85) recognition. Kernel variations also performed well, with Ship (87) and Automobile (83) in the 3x3 kernel variant and Frog (85) and Automobile (83) in the 5x5 kernel variant. CNNs excel at learning spatial hierarchies, making them particularly effective for classes with distinct local patterns.

- Reflect on how depth (for the decision tree, MLP, and CNN) influenced performance. Did it seem to make the model capture more detailed features or overfit?

Increasing depth influenced performance differently across models. For Decision Trees, greater depth (from 10 to 50) consistently led to overfitting, as shown by perfect training accuracy (1.0 across all depths) but significantly lower testing accuracy (~0.58 for custom and ~0.57-0.59 for Scikit-learn variants). This indicates that while deeper trees memorized training data effectively, they failed to generalize, highlighting their inability to capture detailed features in complex datasets like CIFAR-10.

In contrast, MLP models showed modest improvements with depth. The Base MLP achieved a testing accuracy of 0.77, while the Deeper MLP slightly improved to 0.779. The Larger Hidden MLP performed best with an accuracy of 0.799, indicating that increasing hidden layer size was more impactful than depth for feature learning. Shallower and smaller hidden MLPs showed reduced performance, highlighting their limited capacity to capture detailed features. Depth in MLPs improved performance without significant overfitting, suggesting that additional layers enabled better feature representation.

For CNNs, moderate depth enhancements improved performance. The Base VGG11 had a testing accuracy of 0.612, which increased to 0.656 with +2 layers but slightly declined to 0.628 with +4 layers, indicating diminishing returns. Kernel size variations (3x3 vs. 5x5) showed minor differences, with the 5x5 kernel achieving slightly better accuracy (0.607 vs. 0.603). This suggests that while CNN depth helps in capturing more detailed spatial features, excessive depth or larger kernels may lead to computational overhead without significant performance gains. Overall, depth had the most balanced effect in CNNs, improving feature extraction without overfitting.

- Discuss how layer size (for MLP) and kernel size (for CNN) variations affected the model's recognition abilities.

  MLP Layer Size:

  Larger Hidden Layers: Improved recognition abilities, with the LargerHiddenMLP achieving the highest testing accuracy (0.799) and better class recognition, such as Automobile (90) and Truck (92). Larger layers enhanced feature learning by providing more capacity to model complex relationships.

  Smaller Hidden Layers: Reduced recognition abilities, with SmallerHiddenMLP showing lower accuracy (0.743) and weaker class recognition (Automobile: 88, Truck: 88). Smaller layers lacked the capacity for detailed feature extraction.

  CNN Kernel Size:

  3x3 Kernels: Testing accuracy was 0.603, with classes like Ship (87) well-recognized. Smaller kernels captured finer spatial details but lacked sufficient coverage for broader patterns.

  5x5 Kernels: Slightly better accuracy (0.607) and recognition, such as Frog (85) and Automobile (83). Larger kernels captured broader features but added computational complexity as it took more time to compute.

- Summarize the primary findings: which model performed best and why?

  The MLP with Larger Hidden Layers appeared as the best-performing model, achieving the highest testing accuracy of 0.799 and strong recognition for classes like Automobile (90) and Truck (92). Its success is attributed to the increased capacity of larger hidden layers, enabling more effective feature extraction and better generalization

compared to smaller or shallower architectures. Among CNNs, the VGG11 + 2 layers was the best variant, with a testing accuracy of 0.656, as its moderate depth enhanced hierarchical feature learning without overfitting, excelling in recognizing structured classes like Ship (85) and Automobile (95). Overall, MLPs excelled due to their ability to capture complex features, while CNNs showed promise for spatially distinct patterns. In contrast, Decision Trees and Naive Bayes struggled with the high-dimensional complexity of the dataset, limiting their performance.