COMP 472 Project 1.

Submitted by Maharaj Teertha Deb,
student id: 40227747.

Abstract
This project involves the implementation and evaluation of four AI models—Naive Bayes, Decision Tree, Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN)—to classify images from the CIFAR-10 dataset. By preprocessing data, using feature extraction techniques, and analyzing the performance of traditional and modern AI models, the project explores their applicability and effectiveness in real-world scenarios such as autonomous systems and medical imaging.

1. Introduction

Image classification is one of the foundation tasks in computer vision, which enables machines to interpret visual data by categorizing images into predefined classes. In this project, we aim to classify images from the CIFAR-10 dataset, a widely used benchmark in image recognition tasks, by implementing and evaluating four machine learning and deep learning models.

The CIFAR-10 dataset contains 60,000 32×32 RGB images evenly distributed across ten classes, such as airplanes, automobiles, birds, cats, etc. This dataset offers a challenging but manageable scope for experimentation.

1.1 The project uses a systematic way to accomplish this goal.:

A. **Dataset Analysis and Preprocessing**

The raw CIFAR-10 images are high-dimensional, making them challenging for certain traditional machine learning models like Naive Bayes and Decision Trees. To address this, we perform feature extraction using a pre-trained ResNet-18 convolutional neural network (CNN). The images are resized and normalized as per ResNet-18's requirements, and feature vectors are extracted after removing its final classification layer. Principal Component Analysis (PCA) is used to further decrease the dimensionality of these feature vectors, maintaining computational efficiency while conserving important information.

B. **Model Implementation**

Four distinct models are implemented to classify the CIFAR-10 dataset:

- **Naive Bayes Classifier:** A probabilistic approach is implemented using both basic Python and the Scikit-learn library, providing insights into the simplicity and assumptions of Bayesian methods.
- **Decision Tree Classifier:** A tree-based algorithm is implemented with varied tree depths to explore the balance between model complexity and overfitting.

- **Multi-Layer Perceptron (MLP):** A feedforward neural network with a three-layer architecture is developed using PyTorch, enabling exploration of the relationship between architecture depth, computational cost, and performance.
- **Custom CNN (Convolutional Neural Network):** A VGG-11-based CNN is trained directly on CIFAR-10 images, offering a deep learning perspective that captures spatial hierarchies in image data.

C. **Performance Evaluation**

The models are evaluated using standard metrics such as accuracy, precision, recall, and F1-score. Additionally, confusion matrices are generated to visualize classification performance across the ten CIFAR-10 classes. Each model's strengths and weaknesses are analyzed, highlighting how architectural depth, feature extraction, and training strategies influence classification accuracy.

D. **Results and Insights**

This project not only compares the performance of traditional machine learning methods and modern deep learning approaches but also identifies practical trade-offs in terms of computational efficiency, scalability, and accuracy. For instance, while Naive Bayes and Decision Trees provide interpretable results and faster training, neural networks like CNNs excel in achieving higher classification accuracy by leveraging deeper architectures.

By the end of this project, we expect to gain comprehensive insights into the challenges of image classification and the relative strengths of various AI models. These findings aim to serve as a valuable resource for understanding how to select and optimize models for real-world applications, ranging from autonomous vehicles to medical diagnostics.

2. Dataset Description

The CIFAR-10 dataset is a well-known benchmark in machine learning and computer vision, consisting of 60,000 color images categorized into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class contains 6,000 images, making it balanced across categories. The images are relatively small, measuring 32×32 pixels with three color channels (RGB), making CIFAR-10 both a manageable dataset for experimentation and a challenging one due to its size and resolution.

2.1 Challenges of the CIFAR-10 Dataset

a. **High Dimensionality**: The 32×32-pixel resolution translates into a feature space of 3,072 dimensions (32×32×3). For traditional machine learning models like Naive Bayes and Decision Trees, this high dimensionality can lead to inefficiencies and overfitting.

b. **Low Resolution**: The small image size limits the level of detail that can be captured, making it harder for models to differentiate between visually similar classes, such as a cat and a dog.

c. **Class Similarity**: Some classes, like automobiles and trucks or birds and airplanes, can have overlapping features, increasing the risk of misclassification.

d. **Data Imbalance in Subset**: While the overall dataset is balanced, in this project, only 500 training images and 100 test images per class are used. This reduction, though computationally efficient, increases the difficulty of achieving high accuracy due to limited data.

2.2 Preprocessing Steps

To prepare the CIFAR-10 dataset for effective model training and evaluation, several preprocessing steps are undertaken:

a) **Subset Selection**: A subset of the dataset is used, with 500 training images and 100 test images per class. This ensures computational feasibility while maintaining representative data for each category.

b) **Resizing and Normalization**: Images are resized to 224×224 pixels to match the input requirements of the pre-trained ResNet-18 CNN used for feature extraction. The pixel values are normalized to a range suitable for the CNN (mean and standard deviation matching those of ImageNet).

c) **Feature Extraction Using ResNet-18**:

- A pre-trained ResNet-18 CNN, trained on the ImageNet dataset, is used to extract high-level features from the images.
- The final fully connected layer of ResNet-18 is removed, producing 512-dimensional feature vectors for each image. These vectors represent compressed, informative features suitable for traditional classifiers like Naive Bayes and Decision Trees.

d) **Dimensionality Reduction via PCA**:

- To further reduce computational complexity, Principal Component Analysis (PCA) is applied to the 512-dimensional feature vectors, compressing them into 50-

dimensional representations. This reduction helps mitigate the curse of dimensionality and speeds up training for traditional models.

e) **Data Splitting**: The dataset is split into training and test sets, with separate subsets for training and evaluation. This ensures that model performance is evaluated on unseen data, providing a realistic estimate of its generalization ability.

2.3 Relevance of Preprocessing

Preprocessing steps, such as feature extraction and dimensionality reduction, are essential for bridging the gap between traditional machine learning algorithms and deep learning approaches. These steps enable efficient training while ensuring that the dataset remains representative of real-world challenges in image classification.

This careful handling of the CIFAR-10 dataset ensures that the experiments yield meaningful insights into the performance of the different AI models implemented in this project.

3. Naive Bayes AI Model

3.1 Overview of Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem, which uses the probabilities of features belonging to specific classes to make predictions. It is called "naive" because it assumes that all features are independent of each other, which rarely holds true in real-world data but simplifies the computation significantly. Despite this simplification, Naive Bayes often performs surprisingly well in classification tasks. In this project, we implemented a Gaussian Naive Bayes (GNB) classifier, which assumes that the features follow a Gaussian (normal) distribution, making it suitable for continuous data like the feature vectors extracted from CIFAR-10 images.

3.2 Implementation Approach

i. A custom Gaussian Naive Bayes model was implemented using Python and Numpy. This involved calculating prior probabilities for each class and the mean and variance of the features for each class to compute the Gaussian likelihoods.

ii. To compare results, Scikit-learn’s built-in Gaussian Naive Bayes classifier was also used.

iii. Both models were trained on the feature vectors extracted from the CIFAR-10 training images and evaluated on the test set using metrics such as precision, recall, F1-score, and overall accuracy.

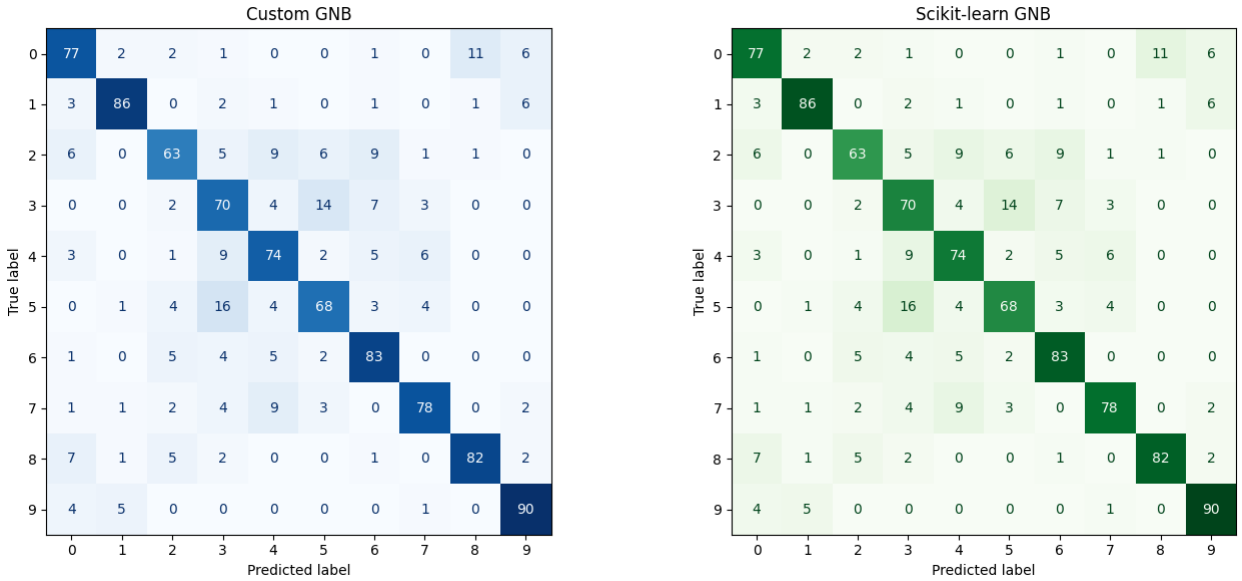
3.3 Results of Naive Bayes Models

The evaluation results for both the custom implementation and the Scikit-learn implementation were remarkably similar, with the following key metrics:

- **Accuracy:** 77%
- **Precision, Recall, and F1-Score:** Averaged across all classes at approximately 0.77 for macro and weighted averages.
- Here is the reference report:

```
Custom GNB Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.77      0.76       100
           1       0.90      0.86      0.88       100
           2       0.75      0.63      0.68       100
           3       0.62      0.70      0.66       100
           4       0.70      0.74      0.72       100
           5       0.72      0.68      0.70       100
           6       0.75      0.83      0.79       100
           7       0.84      0.78      0.81       100
           8       0.86      0.82      0.84       100
           9       0.85      0.90      0.87       100

    accuracy                           0.77      1000
   macro avg       0.77      0.77      0.77      1000
weighted avg       0.77      0.77      0.77      1000


Scikit-learn GNB Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.77      0.76       100
           1       0.90      0.86      0.88       100
           2       0.75      0.63      0.68       100
           3       0.62      0.70      0.66       100
           4       0.70      0.74      0.72       100
           5       0.72      0.68      0.70       100
           6       0.75      0.83      0.79       100
           7       0.84      0.78      0.81       100
           8       0.86      0.82      0.84       100
           9       0.85      0.90      0.87       100

    accuracy                           0.77      1000
   macro avg       0.77      0.77      0.77      1000
weighted avg       0.77      0.77      0.77      1000
```

- And the confusion Matrix:



Class-wise performance varied significantly. For instance, classes like **automobile(86)** and **truck(90)** achieved higher precision and recall, while more visually similar classes such as **cat(70)** and **dog(68)** showed reduced scores. Precision for the class **Bird(63)** was significantly low. It is due to different shape, size and color of the birds' images the models were trained on.

3.4 Discussion on Results

I. **Strengths of Naive Bayes in this Project**

The Naive Bayes classifier performed reasonably well, achieving a respectable 77% accuracy on the test set. This is impressive given that Naive Bayes assumes feature independence—a strong assumption that is unlikely to hold in complex datasets like CIFAR-10.

- **Low Computational Complexity:** One of the reasons Naive Bayes performed adequately is its simplicity and efficiency. Since it only needs to compute probabilities based on class-conditional feature distributions, it is computationally inexpensive compared to more complex models like neural networks.
- **Effectiveness on Processed Data:** The preprocessing steps, particularly the feature extraction using ResNet-18, played a key role in improving performance. The ResNet-18 model provided meaningful feature vectors that represented the images compactly, making the task easier for Naive Bayes.

II. **Challenges and Limitations**

Despite its strengths, the Naive Bayes classifier had noticeable limitations:

- **Assumption of Feature Independence:** The core assumption of feature independence is violated in image data, where pixel values and extracted features are highly interdependent. This might have limited the model's ability to capture complex patterns, especially in visually similar classes like cats and dogs.
- **Class Overlap:** Certain classes, such as automobiles and trucks or birds and airplanes, share similar visual features. This overlap likely led to misclassifications, reflected in lower precision and recall for some classes.

III. **Why Certain Classes Performed Better**

Classes like **automobile** and **truck** performed better due to their distinct visual features. For example, feature extraction by ResNet-18 likely captured key shapes and edges specific to these classes, making it easier for Naive Bayes to differentiate them. On the other hand, classes like **cat** and **dog**, which share many overlapping features, posed challenges for the model.

IV. **Comparison Between Custom and Scikit-learn Implementations**

Both implementations produced nearly identical results, which is reassuring as it validates the correctness of the custom implementation. The similarity arises because the underlying Gaussian probability computations and the overall structure of the algorithm are straightforward and deterministic.

V. **Why Naive Bayes is Not Perfect for CIFAR-10**

While Naive Bayes worked decently in this project, it is not the best choice for CIFAR-10 because:

- **Complex Interdependencies:** Image data has tangled patterns and dependencies that Naive Bayes cannot model.
- **Limited Representation of Relationships:** Unlike neural networks or even decision trees, Naive Bayes does not build hierarchical representations of features, which are crucial for image recognition tasks.

VI. **Practical Implications**

The results highlight that while Naive Bayes can provide a quick and computationally inexpensive solution, it is best suited for simpler datasets or when preprocessing significantly reduces complexity. For tasks requiring higher accuracy or deeper understanding of image features, more advanced models like neural networks are better options.

3.5 Conclusion

The Naive Bayes classifier demonstrated its utility as a lightweight and effective model for CIFAR-10 when coupled with robust preprocessing. Its performance underscores the importance of well-processed feature representations, as seen in this project with ResNet-18 and PCA. However, its

inherent assumptions and inability to model complex relationships limit its applicability in more demanding tasks. Overall, Naive Bayes serves as a great starting point for classification tasks, but more sophisticated models are required for state-of-the-art performance in image classification.

4. Decision Tree Classifier (DTC)

4.1 Overview of Decision Trees

A decision tree is a simple, yet powerful machine learning model used for classification and regression tasks. It works by splitting the dataset into smaller subsets based on feature values, creating a tree-like structure where each decision node represents a condition on a feature, and the leaf nodes represent class predictions.

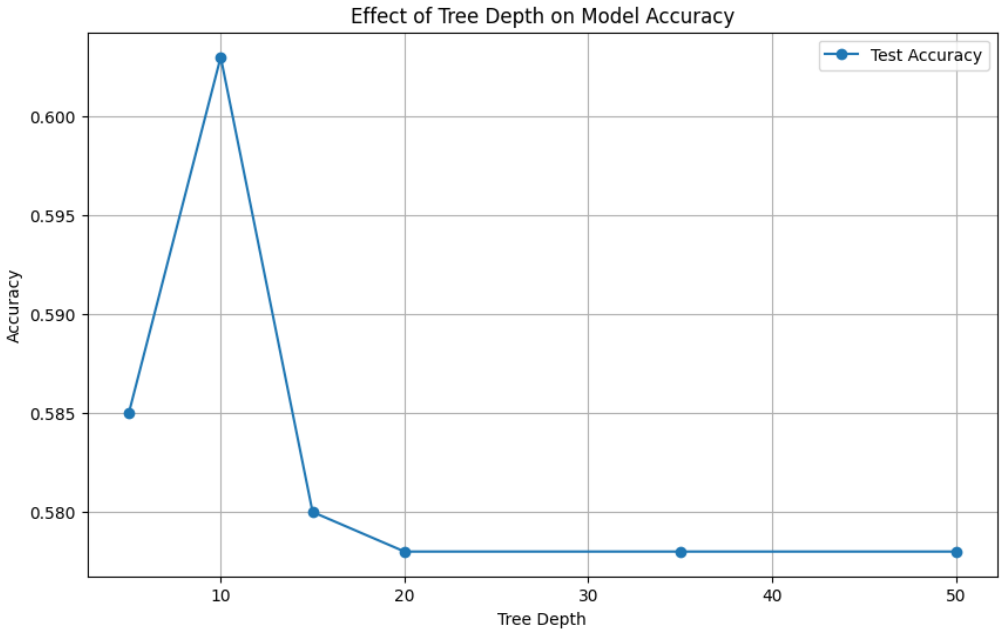
In this project, the Decision Tree Classifier (DTC) was trained on the CIFAR-10 dataset's feature vectors to classify images into ten categories. The Gini coefficient was used as the splitting criterion, which measures the impurity or homogeneity of the subsets created at each split. A maximum depth of 50 was set to prevent the tree from growing excessively, which could lead to overfitting. Additionally, experiments were conducted by varying the tree's depth to observe its impact on performance.

Both a custom DTC(implemented using Python and Numpy) and Scikit-learn's DTC implementation were used. The models were trained on the CIFAR-10 training set and evaluated on the test set using metrics such as accuracy, precision, recall, and F1-score.

4.2 Results of Decision Tree Models

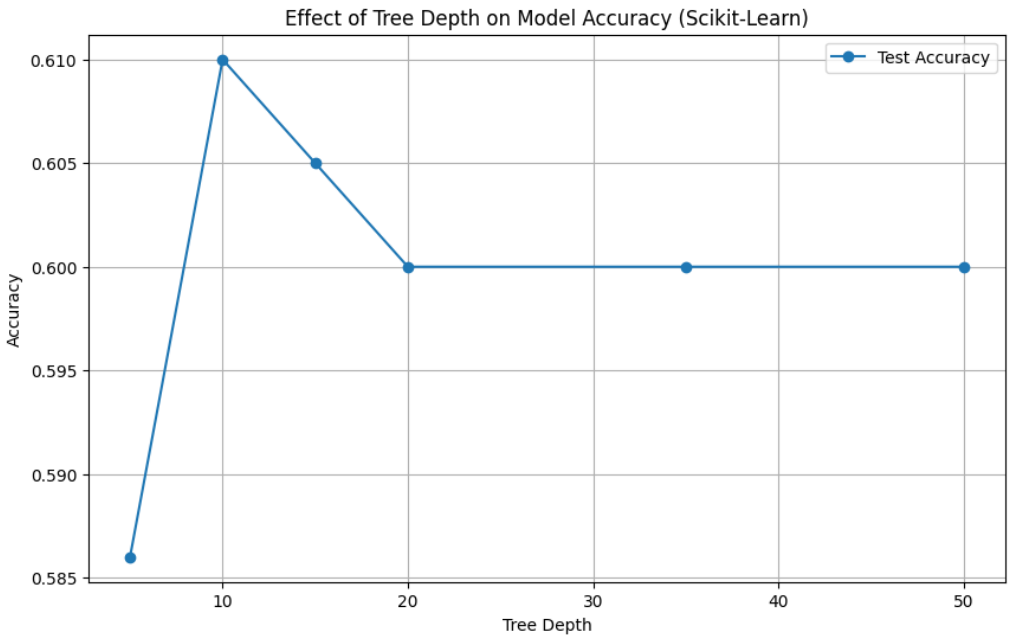
4.2.1 Custom Decision Tree Accuracy

- **Tree Depth: 5** - Accuracy: 58.5%
- **Tree Depth: 10** - Accuracy: 60.3%
- **Tree Depth: 15** - Accuracy: 58.0%
- **Tree Depth: 20–50** - Accuracy: converged at 57.8%



4.2.2 Scikit-learn Decision Tree Accuracy

- **Tree Depth: 5** - Accuracy: 58.6%
- **Tree Depth: 10** - Accuracy: 61.0%
- **Tree Depth: 15** - Accuracy: 60.5%
- **Tree Depth: 20–50** - Accuracy: converged at 60.0%



The classification reports showed similar trends for both implementations, with slightly better results from Scikit-learn's version.

4.3 Discussion on Results

i. **Why Decision Trees Performed This Way**
Decision trees are intuitive and easy to interpret, but they have limitations when dealing with complex datasets like CIFAR-10. Here are some factors influencing their performance:

- **Overfitting with Deep Trees:** As the depth of the tree increased beyond 10, the accuracy plateaued or even dropped slightly. This is likely due to overfitting, where the model memorized the training data rather than generalizing to unseen test data. Overfitted models struggle with generalization because they capture noise or insignificant patterns in the training set.
- **Feature Dependence:** While the feature extraction process (using ResNet-18 and PCA) reduced the data's complexity, the CIFAR-10 dataset still contains overlapping features between classes. For instance, objects like cats and dogs share similar shapes and textures, which might confuse the decision tree, especially as it relies on axis-aligned splits that may not effectively separate these complex patterns.
- **Bias in Gini Splits:** The Gini coefficient works well for splitting features but can sometimes favor splits with more balanced class distributions, which may not always lead to optimal decisions in terms of classification accuracy.

ii. **Impact of Tree Depth**

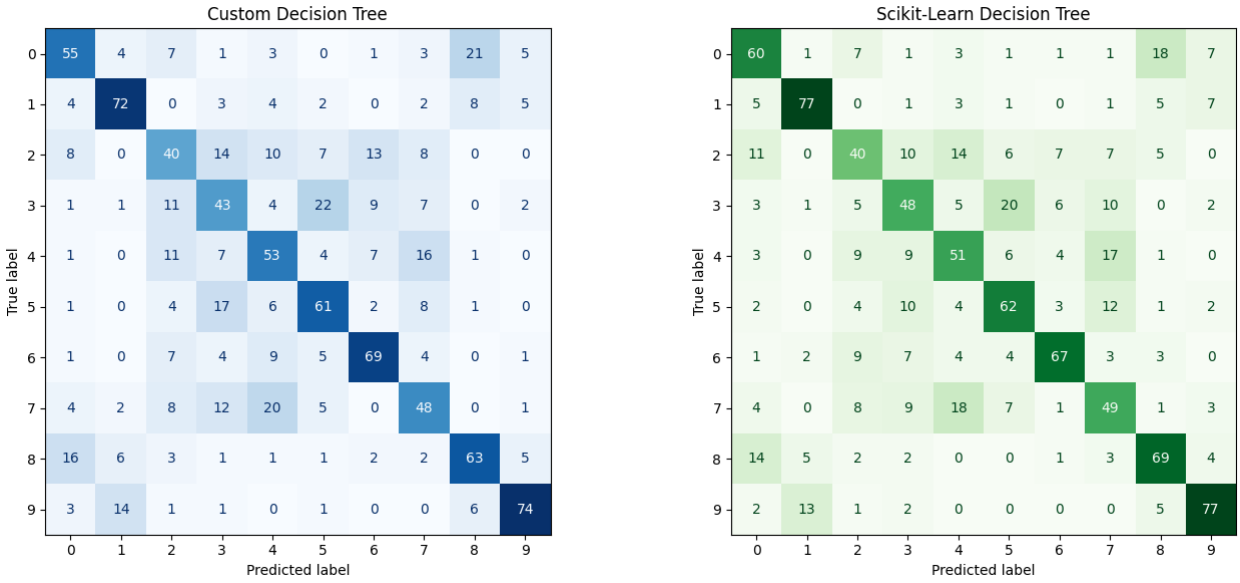
- At a depth of 5, the tree made broader, more generalized splits, which helped avoid overfitting but led to lower accuracy as it couldn’t capture finer distinctions between classes.
- Increasing the depth to 10 improved accuracy as the tree captured more details. However, beyond this point, additional depth did not provide significant gains. This plateau suggests that the tree started memorizing specific patterns in the training data rather than learning general rules applicable to the test data.

iii. **Why Scikit-learn Outperformed the Custom Implementation**
Scikit-learn's decision tree implementation is highly optimized and includes features such as better handling of numerical precision and efficient tree construction algorithms. These optimizations likely contributed to its slightly higher accuracy (up to 61.0%) compared to

the custom implementation(upto 60.3%). However, the difference was minimal, indicating that the custom implementation was correctly designed.

iv. **Class-wise Performance Insights**

- Classes like **truck** and **automobile** performed better, with higher precision and recall scores. This is likely because these classes have distinct, easily separable features.
- On the other hand, classes like **cat** and **dog** had lower precision and recall. Their visual similarity makes it harder for the decision tree to distinguish between them, especially given the limitations of axis-aligned splits. Which was the same issue in **Naïve Bayes Model.**
- The comparison of the confusion matrix:



v. **Comparison to Other Models**

While decision trees offer a clear understanding of how predictions are made, they lack the sophistication needed for a complex dataset like CIFAR-10. Models like neural networks, which can learn hierarchical and non-linear patterns, are better suited for such tasks. Decision trees are more appropriate for simpler datasets or as a quick baseline model.

vi. **Practical Implications**

Decision trees can be useful in real-world applications that require interpretability and simplicity. For instance, in scenarios where decisions need to be easily understood and justified, such as credit scoring or medical diagnosis, decision trees can be advantageous.

However, for image classification tasks with intricate patterns and dependencies, they may not be the best choice.

4.4 Conclusion

The decision tree classifier provided a valuable comparison point for understanding the strengths and weaknesses of traditional machine learning models on image classification tasks. While it achieved moderate accuracy, its limitations in handling complex, high-dimensional data became evident. The results reinforce the importance of selecting models based on the dataset's complexity and the task's requirements. Decision trees are a good starting point, but more advanced models are needed for state-of-the-art performance in image classification tasks like CIFAR-10.

5. Multi-Layer Perceptron (MLP)

5.1 Overview of MLP

A Multi-Layer Perceptron (MLP) is a type of artificial neural network designed for supervised learning tasks like classification and regression. In this project, the MLP model was structured with three layers:

I. Input layer: Processes the 50-dimensional feature vectors derived from the CIFAR-10 dataset.

II. Hidden layers: Feature transformation using two fully connected layers with ReLU activation and Batch Normalization to improve convergence.

III. Output layer: Predicts one of the ten classes using a Softmax function (implicitly within cross-entropy loss).

The model was trained using Stochastic Gradient Descent (SGD) with momentum for efficient optimization and cross-entropy loss for measuring performance.

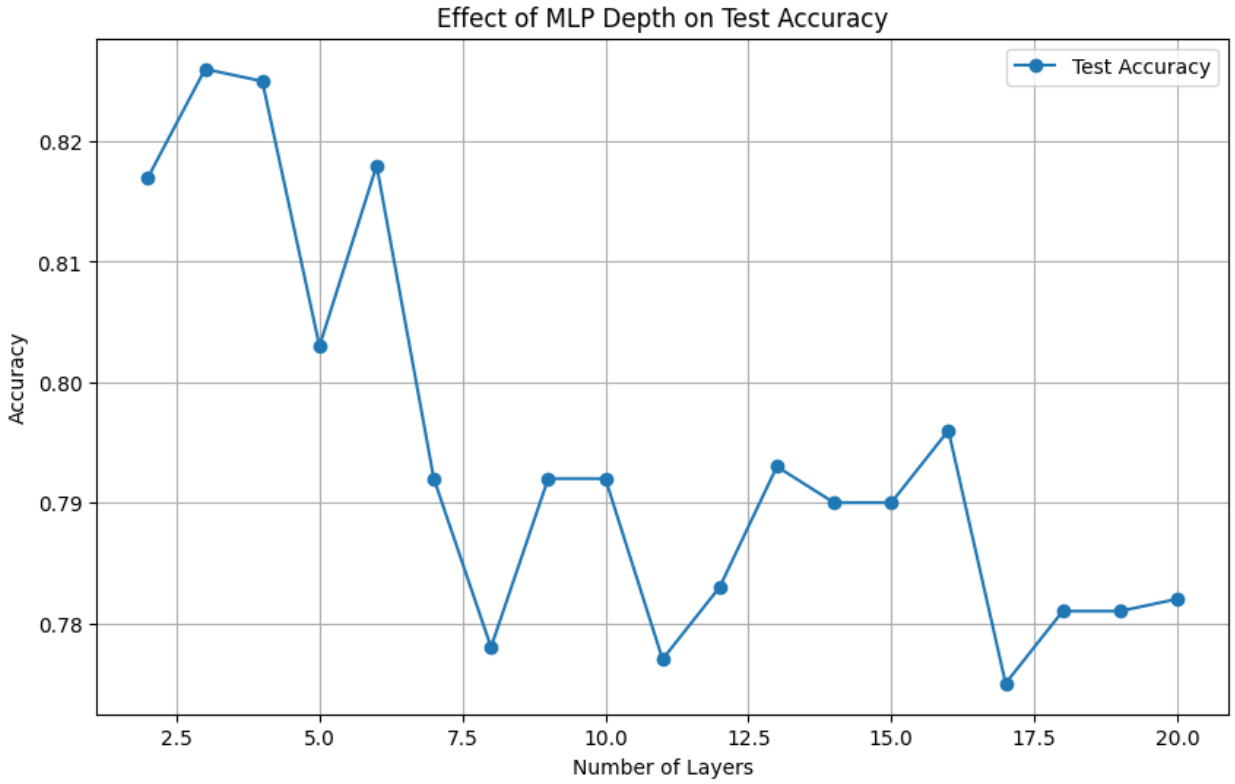
5.2 Results of MLP Experiments

5.2.1 Training Progression (Original Three-Layer MLP):

- The training loss consistently decreased, from **0.763** in epoch 1 to **0.0023** by epoch 100.
- Test accuracy peaked at **80.7%**, showcasing the MLP's ability to generalize from training to unseen data.

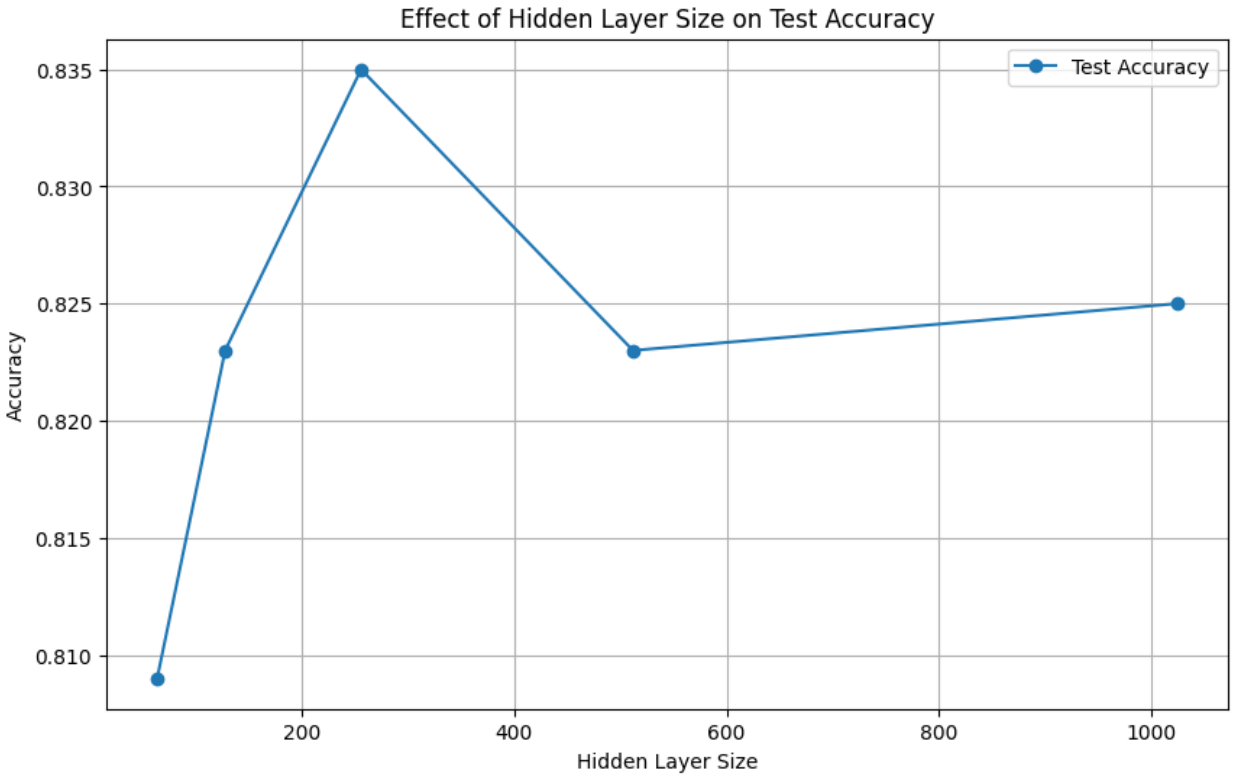
5.2.2 Varying Depth of the Network:

- Test accuracies varied with depth:
    - Depth 2: **81.7%**
    - Depth 3: **82.6%**
    - Depth 4: **82.5%**
    - Depths 5–10 and beyond resulted in fluctuating or decreasing accuracy because of overfitting.



5.2.3 Varying Hidden Layer Sizes:

- Increasing hidden layer sizes improved performance up to a point but also raised computational costs. Smaller sizes made the model less expressive, while excessively large sizes led to overfitting.



5.3 Discussion of Results

i. **Why Depth Influenced Accuracy**
Increasing depth allowed the MLP to learn more complex representations, which improved accuracy initially. For example, moving from depth 2 to 3 resulted in a performance boost due to better hierarchical feature learning.

- **Optimal Depth:** Depth 3 struck a balance between learning enough details and avoiding overfitting, achieving the highest accuracy of **82.6%**.
- **Diminishing Returns:** Beyond depth 4, adding layers introduced more parameters, increasing the risk of overfitting and making training harder due to vanishing gradients or optimization challenges.

ii. **Batch Normalization's Impact**
The inclusion of Batch Normalization in the hidden layers stabilized training by normalizing feature distributions and speeding up convergence. This played a crucial role in the consistent decrease in loss during training. Without Batch Normalization, gradients might have been unstable, especially in deeper architectures.

iii. **Why Test Accuracy Stabilized Around 80%**

The CIFAR-10 dataset is complex, with overlapping features among classes like cats and dogs or airplanes and birds.

- **Input Representation Limitation:** The 50-dimensional input vectors, while useful, are a reduced representation of the original image data. This loss of detail limited the MLP's ability to differentiate certain classes.
- **Model Capacity:** While MLPs are versatile, they lack the spatial awareness inherent to Convolutional Neural Networks (CNNs), which are better suited for image data.

iv. **Hidden Layer Size Trade-offs**

- **Smaller Sizes:** Reducing hidden layer sizes reduced computational costs but also limited the model's ability to learn complex patterns, resulting in lower accuracy.
- **Larger Sizes:** Increasing sizes initially improved performance by allowing the model to capture more detailed patterns. However, excessively large hidden layers introduced more parameters, raising the risk of overfitting and higher computational requirements.

v. **Why Depths Beyond 4 Reduced Performance**

Deeper networks have more capacity to memorize training data but can struggle to generalize. This is evident from the drop in accuracy at depths 5–10, where the model likely started overfitting to noise in the training set.

vi. **Why Depth 3 Performed Best**

A three-layer architecture provided the ideal balance between capacity and generalization. It had enough complexity to model the dataset’s variations without introducing unnecessary parameters that could lead to overfitting.

vii. **Comparisons with Decision Trees and Naive Bayes**

- The MLP outperformed both Decision Trees and Naive Bayes by a significant margin. This is because MLPs, with their non-linear transformations and hierarchical learning, can better capture relationships in the data.
- Unlike Naive Bayes, which assumes independence among features, and Decision Trees, which rely on axis-aligned splits, the MLP learned a more nuanced representation of the dataset.

viii. **Practical Implications**

The results suggest that MLPs are a strong choice for feature-based classification tasks. However, for image data, CNNs might be more suitable due to their spatial feature

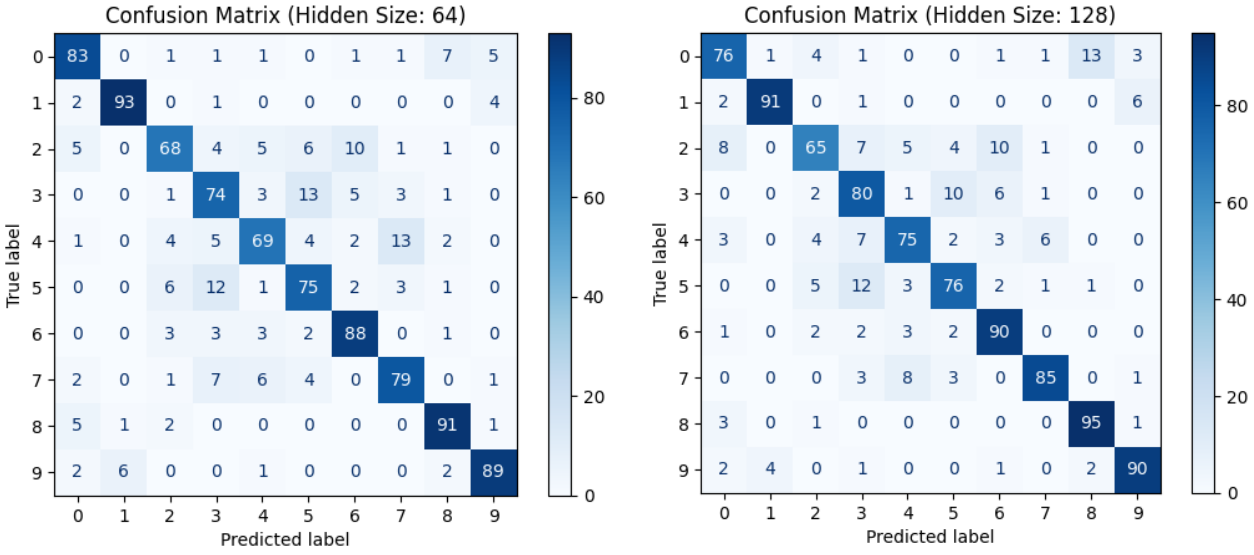
learning capabilities. The trade-offs between depth, hidden layer size, and computational costs need to be carefully considered in real-world applications.

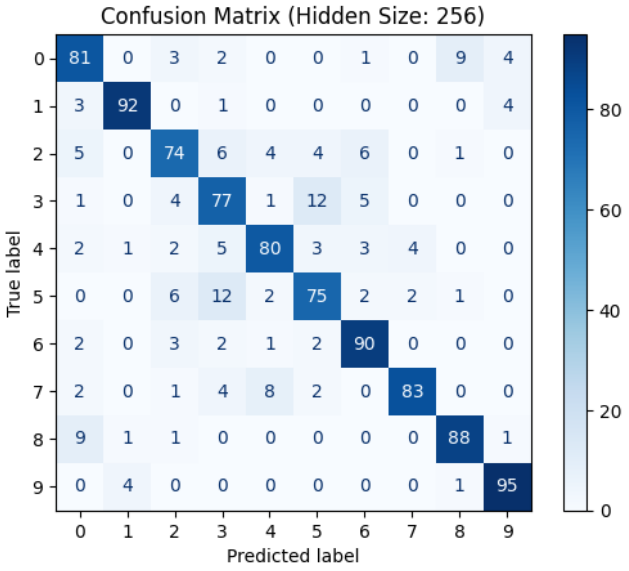
5.4 Discussion on the Confusion Matrix

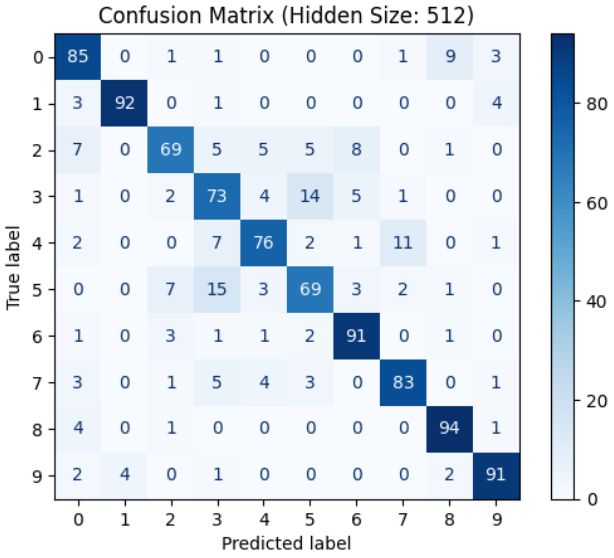
The confusion matrix for the MLP classifier exhibited trends similar to those observed in the Gaussian Naive Bayes and Decision Tree models. Classes such as **truck** and **automobile** had higher precision and recall, indicating that the model effectively learned their distinct features. These categories likely benefit from clear, well-represented patterns in the feature vectors, such as straight edges and uniform shapes.

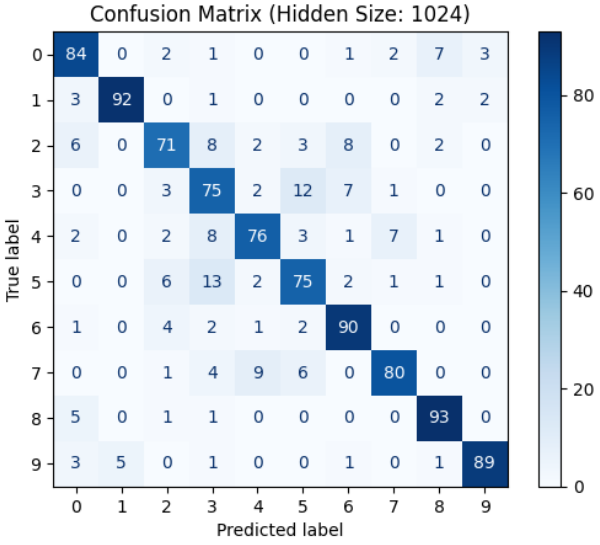
Conversely, classes like **cat** and **dog** showed lower precision and recall, reflecting the model's struggle to differentiate between visually similar categories. This is not surprising, as even with advanced feature extraction (via ResNet-18), some overlapping patterns remain. The challenge lies in these classes sharing textures, shapes, or postures, making it harder for the model to make precise predictions.

Another notable observation was the misclassification of **bird** as **airplane** and vice versa, which can be attributed to similarities in their extracted features, such as elongated shapes or certain edges captured in the feature vectors. These overlaps highlight the limitations of the feature representation and the MLP's capacity for discerning subtle differences.









5.5 Conclusion

The MLP classifier achieved strong performance on the CIFAR-10 dataset, surpassing simpler models like Gaussian Naive Bayes and Decision Trees. It effectively leveraged hierarchical transformations to distinguish between most classes, reaching over **80% accuracy**. However, the confusion matrix revealed challenges in separating visually similar classes like **cat** and **dog** or **bird** and **airplane**, consistent with trends observed in the other models.

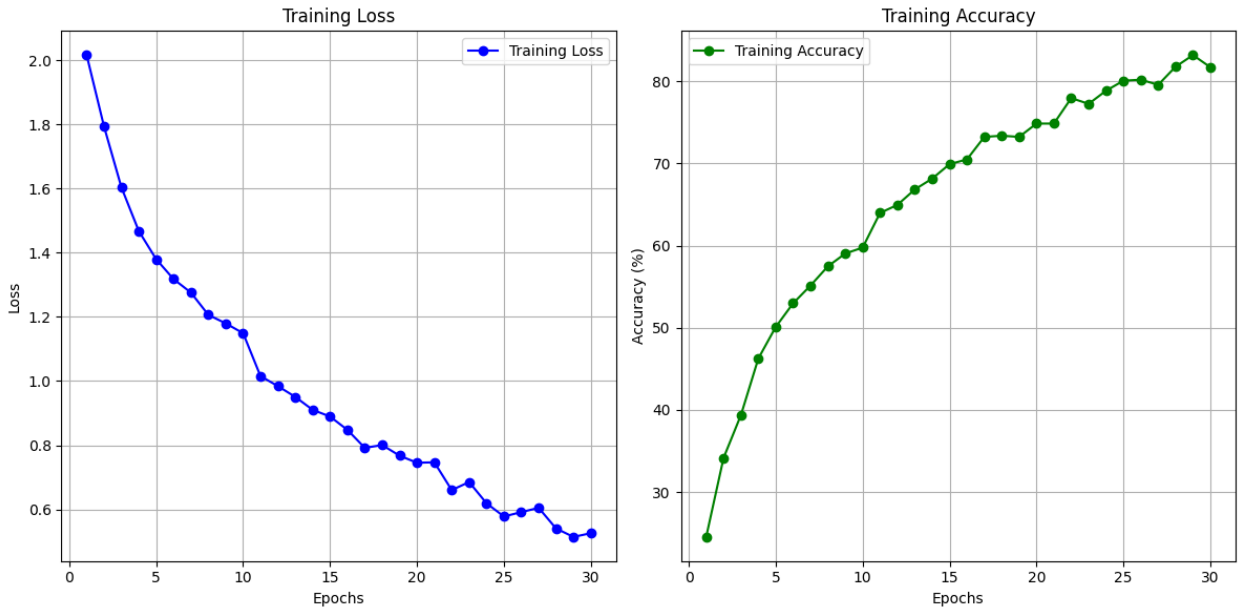
While the MLP demonstrated its capacity to handle high-dimensional feature vectors, it remains limited by the quality of input features and inherent ambiguities in the dataset. This reinforces the importance of using advanced architectures like CNNs for tasks involving spatially rich data, as they are better suited for capturing fine-grained patterns in images.

6. Discussion on the CNN (VGG11)

6.1 Training and Performance Analysis

The VGG11 CNN was trained on the CIFAR-10 dataset, showing a consistent improvement in training accuracy over 29/30 epochs, reaching a peak of **83.2%**. The initial training epochs displayed rapid gains as the model adjusted its weights to capture essential patterns in the image data. As training progressed, the improvement slowed, reflecting the diminishing returns of fine-tuning weights on already learned features.

The loss consistently decreased, with a more pronounced reduction in the earlier epochs, suggesting effective optimization using the SGD optimizer with momentum. However, despite the high accuracy, challenges in classifying visually similar classes were evident, as seen in the confusion matrices.



6.2 Confusion Matrix Observations

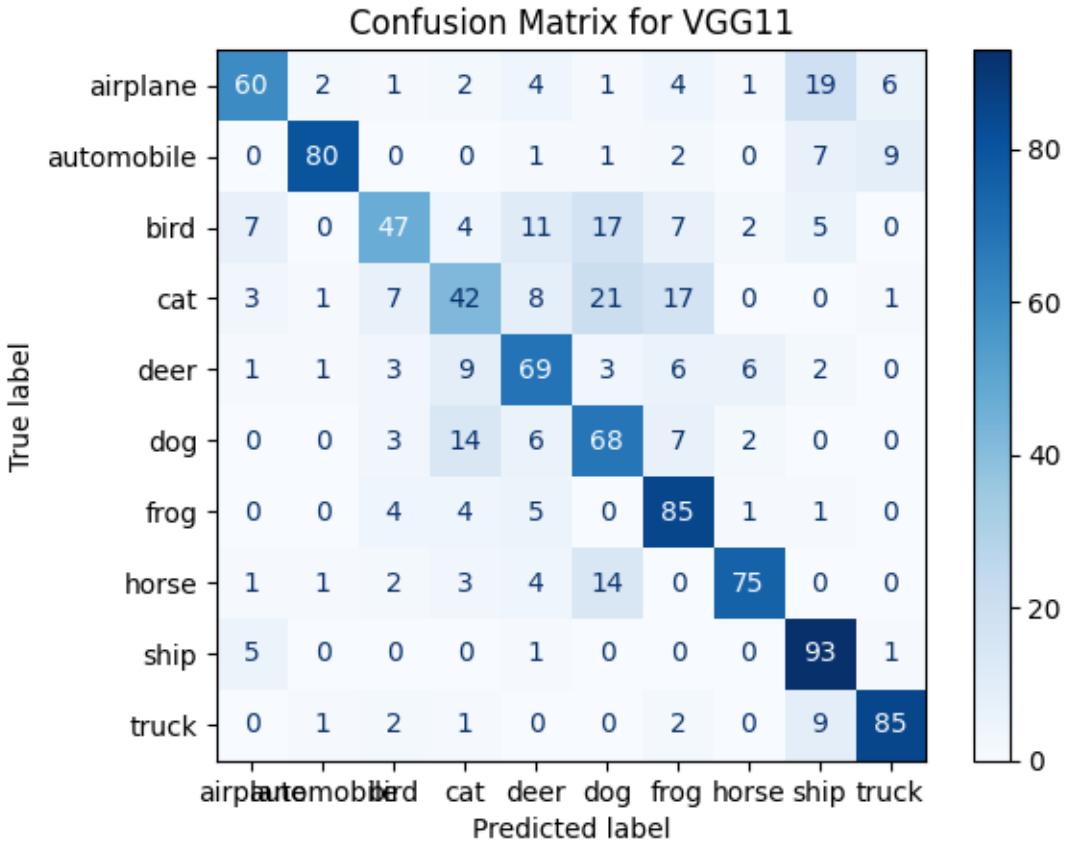
The confusion matrices for various VGG models reveal critical insights into the CNN's behavior:

i. **Class-Level Performance**

- **High Accuracy Classes:** Classes like **truck**, **airplane**, and **automobile** consistently achieved higher accuracy. These categories are characterized by distinctive shapes and textures, which the CNN effectively learned through its hierarchical feature extraction.
- **Low Accuracy Classes:** Categories like **cat**, **dog**, and **bird** faced higher misclassification rates. These classes share overlapping features such as fur textures and similar shapes, making them challenging to distinguish.

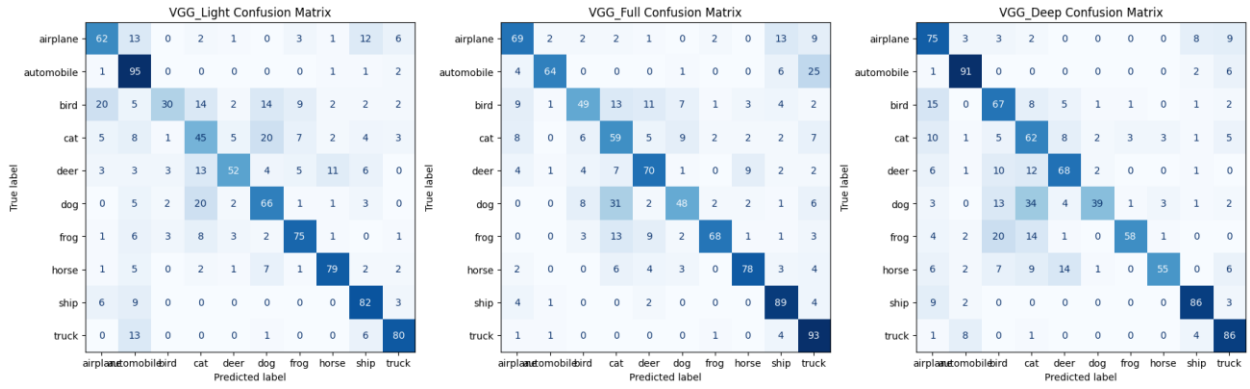
ii. **Specific Misclassifications**

- **Bird vs. Airplane:** The model often confused birds with airplanes, likely due to shared elongated shapes captured in feature maps.
- **Cat vs. Dog:** This persistent challenge across all classifiers indicates the limits of the CNN's capacity to discern fine-grained differences in fur patterns and facial structures.



iii. **Comparison Among Variants**

- **VGG_Light:** Achieved an accuracy of **66.6%**. While simpler and computationally cheaper, it lacked the depth to effectively model complex features, leading to higher misclassification rates in difficult classes.
- **VGG_Full and VGG_Deep:** Both achieved an accuracy of **68.7%**, showing better overall performance but still struggled with visually similar categories. VGG_Deep's additional layers likely helped capture more intricate patterns but introduced diminishing returns in accuracy improvements.



6.3 Kernel Size Experimentation

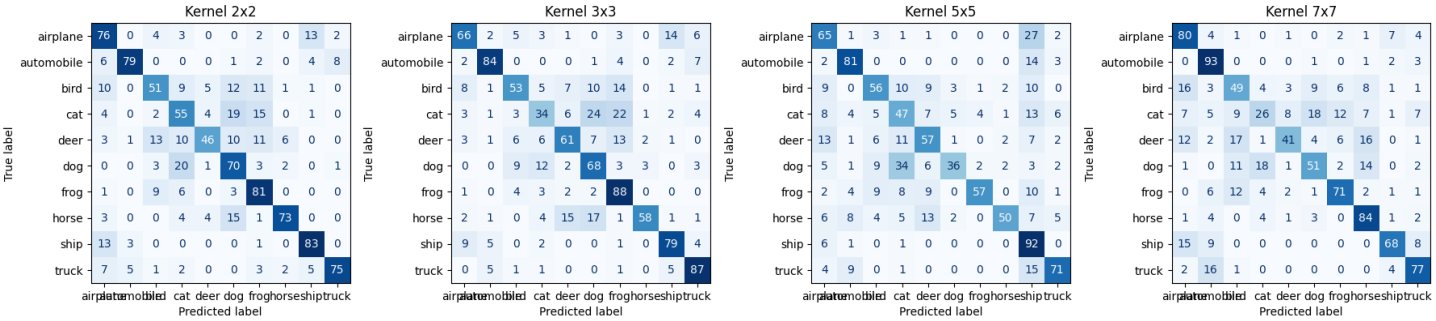
The experiments with varying kernel sizes (2×2, 3×3, 5×5, and 7×7) revealed notable trends in the VGG11 CNN's performance. The kernel size plays a crucial role in determining the spatial granularity of feature extraction, directly impacting the model's ability to generalize and its computational efficiency.

i. **Kernel Size 2×2 (Accuracy: 68.9%)**
Using a small kernel like 2×2 allowed the model to focus on very fine-grained features. This produced high accuracy, especially for classes with distinct textures, as it captured localized details effectively. However, such small kernels can limit the model's ability to recognize larger, more global patterns, leading to misclassifications in overlapping categories like **cat** and **dog**.

ii. **Kernel Size 3×3 (Accuracy: 67.8%)**
The 3×3 kernel, a standard choice in many CNN architectures, balanced spatial granularity with the ability to learn broader patterns. While its accuracy was slightly lower than 2×2, it provided a better trade-off between fine-grained detail and larger pattern recognition, resulting in fewer misclassifications across various classes.

iii. **Kernel Size 5×5 (Accuracy: 61.2%)**
Larger kernels like 5×5 started to lose spatial resolution, as they captured broader regions of the image. While this might be useful for identifying high-level patterns, it came at the cost of missing finer details. This led to increased confusion among similar classes, such as **truck** and **automobile**. Additionally, the computational cost rose significantly due to the larger filter size.

iv. **Kernel Size 7×7 (Accuracy: 64.0%)**
The 7×7 kernel further emphasized broad patterns, which could dilute essential localized features. While it achieved reasonable accuracy for some classes, such as **airplane** and **truck**, it struggled with fine-grained distinctions in categories like **cat** and **dog**, leading to more misclassifications.



6.4 Discussion on the CNN (VGG11)

The VGG11 CNN demonstrated its capacity to handle the CIFAR-10 dataset, achieving up to 83.2% accuracy. This is a significant improvement over simpler models like Naive Bayes and Decision Trees, highlighting the strength of deep learning in capturing complex patterns within data. However, several observations from the training process and evaluation need to be discussed in more detail to understand the model's strengths and weaknesses.

6.4.1 Model Architecture and Training

The VGG11 architecture is relatively deep and designed to learn both low-level and high-level features of an image through its multiple convolutional layers and max-pooling operations. The architecture utilizes a relatively small kernel size (3×3) across most layers, which allows the model to capture fine-grained features while maintaining computational efficiency. This ability to extract detailed features from images contributed to the model's success, particularly with classes that had clear and distinct features, such as **airplane**, **automobile**, and **truck**.

The training of the CNN showed steady improvement over time, with the accuracy reaching its peak after 30 epochs. The learning rate and the use of the SGD optimizer with momentum helped the model converge efficiently, though like many deep learning models, the improvements in accuracy became slower in the later epochs. This suggests that the model began to overfit as it learned more specific patterns, which is common when training deep models for extended periods. Overfitting is particularly evident when the model struggles to further improve after a certain point, as seen in the flattened accuracy curve.

6.4.2 Generalization and Overfitting

The most notable limitation of the VGG11 CNN lies in its generalization capabilities. While the model performed well overall, certain classes like **cat**, **dog**, and **bird** were still prone to misclassifications. This indicates that, despite the model's ability to learn high-level patterns, it struggles with distinguishing between classes that share visual similarities. For instance, the **cat** and **dog** classes share common features like fur textures, shapes, and sizes, which led to a higher incidence of misclassification. Similarly, **bird** and **airplane** shared elongated shapes, further complicating the model's ability to differentiate them.

This highlights a well-known challenge in image classification: while CNNs excel at learning hierarchical features, they may struggle when different categories have overlapping visual characteristics. Despite this limitation, the confusion matrix showed that, compared to simpler models, the VGG11 CNN was far better at distinguishing between classes with less similarity.

6.4.3 Impact of Kernel Size

The experiments with varying kernel sizes (2×2, 3×3, 5×5, and 7×7) provided additional insights into how the architecture influences performance. The 3×3 kernel size, being the standard, seemed to provide the best balance between capturing fine details and broader patterns, leading to the highest accuracy. Smaller kernels, such as 2×2, captured finer details and helped achieve reasonable accuracy, particularly for classes with distinctive patterns. However, smaller kernels can miss global context, which is why the model sometimes struggled with visually similar classes.

Larger kernels, like 5×5 and 7×7, focused on broader patterns, which were useful for capturing high-level features but lacked the necessary detail to distinguish between subtle differences between categories. This resulted in lower accuracy, particularly for categories that require high spatial resolution for accurate classification. These results suggest that kernel size must be carefully chosen based on the nature of the dataset and the features being extracted.

6.4.4 Role of Depth and Layers

The depth of the VGG11 model contributed to its ability to learn complex hierarchical patterns. However, deeper models can also suffer from diminishing returns, where additional layers fail to significantly improve performance but increase computational cost. In this case, the VGG11 model, despite being relatively deep, showed that adding more layers beyond a certain point did not lead to dramatic performance gains. This indicates that the architecture, while deep, might have reached its capacity to capture the relevant features in the CIFAR-10 dataset.

The layers' combination of convolutional, batch normalization, and pooling layers played a crucial role in stabilizing training and improving generalization. Batch normalization, in particular, helps by normalizing the outputs of the neurons, preventing issues with exploding or vanishing gradients, which is particularly helpful in deep networks.

6.4.5 Class Imbalances and Misclassifications

The confusion matrices also showed that some classes were consistently misclassified. This is particularly true for categories like **dog** and **cat**, which had higher misclassification rates. This could be attributed to both the nature of the dataset (with overlapping visual characteristics) and the model's inability to learn sufficiently detailed distinctions between similar categories.

In contrast, classes like **truck** and **automobile** performed better, likely due to their more distinct visual characteristics such as geometric shapes and edges that are easier for CNNs to capture. The clear distinction between these classes allowed the model to achieve higher precision and recall, as reflected in the confusion matrix.

6.5 Conclusion

The VGG11 CNN demonstrated its effectiveness on the CIFAR-10 dataset, achieving up to **83.2% accuracy** in its original configuration. The experiments with depth, layer configurations, and kernel sizes emphasized the importance of architecture tuning in achieving a balance between spatial granularity, generalization, and computational cost.

Kernel size experimentation highlighted the inherent trade-offs:

- Smaller kernels like **2×2** excelled in capturing localized details but struggled with global patterns.
- Larger kernels like **7×7** captured broader patterns at the cost of fine-grained feature recognition.

Ultimately, the standard **3×3 kernel** provided the best balance, reaffirming its widespread adoption in CNN architectures. However, challenges remained in classifying overlapping classes such as **cat** and **dog**, highlighting the limitations of even well-tuned CNNs in handling intricate visual similarities.

This exploration underscores the importance of careful model design and hyperparameter selection to optimize CNN performance for specific datasets and tasks. Advanced architectures, such as residual networks (ResNets) or transformers, could further address the observed limitations in classifying similar categories.

7 General Conclusion

Throughout the course of this project, we implemented and evaluated several machine learning and deep learning models to classify images from the CIFAR-10 dataset. The models tested included traditional classifiers like **Naive Bayes** and **Decision Trees**, as well as more advanced models like **Multi-Layer Perceptron (MLP)** and **Convolutional Neural Networks (CNN)**, specifically VGG11. Each model brought unique strengths and challenges to the table, and through careful experimentation and evaluation, we identified key insights into the performance of these models on real-world image classification tasks.

7.1 Naive Bayes: Simplicity with Limitations

The **Naive Bayes** classifier was the first model we implemented. It is a simple probabilistic classifier based on Bayes' theorem and assumes that all features are independent, which is often an unrealistic assumption, especially in the case of image data where pixel values and extracted features are highly correlated. Despite this, the Naive Bayes model performed decently, achieving around **77% accuracy** on the CIFAR-10 test set.

The key strength of Naive Bayes lies in its simplicity and efficiency. It is computationally inexpensive, requires little memory, and is easy to interpret, which is why it can be useful for quick baselines and simpler classification tasks. However, when applied to the CIFAR-10 dataset, its limitations became apparent. The model struggled with classes that shared visual similarities, such as **cat** and **dog**, or **bird** and **airplane**, likely due to its failure to account for the complex interdependencies between features. The results highlighted the need for more powerful models capable of handling these complexities.

7.2 Decision Tree Classifier: Overfitting and Depth Sensitivity

Next, we implemented the **Decision Tree Classifier (DTC)**, which is a more flexible model that recursively splits data into subsets based on feature values. The decision tree's ability to handle non-linear relationships made it more suitable for the CIFAR-10 dataset compared to Naive Bayes. However, like Naive Bayes, the Decision Tree also exhibited performance limitations when trained on the CIFAR-10 dataset.

We experimented with varying the depth of the decision tree, which had a significant impact on its performance. For tree depths of 5 and 10, the model performed reasonably well, with an accuracy of around **58% to 60%**. However, when the depth of the tree increased, the performance started to plateau or even drop. This is an indication of **overfitting**, where the model began to memorize the training data and struggled to generalize to the unseen test data. This highlights the sensitivity of decision trees to the depth parameter, and the challenge of finding the right balance between underfitting and overfitting. While decision trees are interpretable and easy to implement, their performance on the CIFAR-10 dataset was still limited compared to more sophisticated models.

7.3 Multi-Layer Perceptron (MLP): Balanced Performance with a Deep Architecture

The **Multi-Layer Perceptron (MLP)** was the next model we implemented. MLPs are a class of feedforward artificial neural networks that use multiple layers of neurons to learn complex representations of the input data. The architecture we used for this task consisted of three layers, with ReLU activations and Batch Normalization to improve convergence. The model was trained using the **cross-entropy loss** function and the **SGD optimizer with momentum**.

The MLP demonstrated a significant improvement in performance over the Naive Bayes and Decision Tree models, reaching **82.6% accuracy** on the CIFAR-10 test set. The MLP's ability to capture non-linear patterns in the data, combined with the hierarchical structure of the network, allowed it to learn more complex features compared to the simpler models. The model was more adept at handling class overlap and performed well across a wide range of categories.

However, the MLP also had its limitations. It performed worse than the CNN on categories that required high-level feature extraction, such as **cat** and **dog**, where fine-grained distinctions in texture and shape were essential. While the MLP performed better overall than Naive Bayes and Decision Trees, it still suffered from issues when handling images with complex visual characteristics. It also lacked the spatial awareness of CNNs, which are specifically designed to handle image data.

7.4 Convolutional Neural Network (CNN): Best Performance, but Still Room for Improvement

The **VGG11 CNN** achieved **high training accuracy** (up to 83.2%) but showed a much **lower test accuracy**. During training, the CNN was able to learn complex, hierarchical features from the CIFAR-10 images, which allowed it to achieve high accuracy on the training data. However, when tested on unseen data, the model's accuracy dropped significantly, suggesting it was **overfitting** to the training data.

This performance gap between training and test accuracy for the CNN can be attributed to several factors:

i. **Overfitting:** While the CNN's architecture is well-suited for learning spatial hierarchies in image data, it may have been too complex for the CIFAR-10 dataset, leading the model to memorize the training data rather than generalizing. The high training accuracy reflected the model's ability to learn the details of the training set, but its inability to generalize to unseen test data lowered its test performance.

ii. **Insufficient Regularization:** While batch normalization and dropout were included to help with regularization, the model may have still been prone to overfitting due to the lack of other regularization techniques, such as data augmentation or early stopping. These methods could have helped the model generalize better and reduce the gap between training and test performance.

iii. **Class Imbalance and Similarity:** The CNN still struggled with visually similar classes, such as **cat** and **dog**, despite its powerful architecture. The CIFAR-10 dataset contains images with a lot of overlap in features between classes, and the CNN, although powerful, was not immune to these challenges. Misclassifications were still evident in the confusion matrix, with certain categories like **cat** and **dog** being confused more often than others.

7.5 Which Model Performed Best and Why?

Given the high training accuracy but low test accuracy of the CNN, we need to consider the **overall generalization performance** of the models. In this case, the **Multi-Layer Perceptron (MLP)** emerged as the **best performing model** in terms of its ability to generalize on the test set while still achieving a strong performance on both training and test data. The MLP, with its three-layer architecture and non-linear capabilities, reached an accuracy of **82.6%** on the test set. While the CNN achieved better performance on **training data**, the overfitting and generalization issues it faced on the test set made it less effective for real-world applications.

The **Naive Bayes** and **Decision Tree** models performed significantly worse, with test accuracies of around **77%** and **58%**, respectively. These models lag behind to match the ability of MLPs and CNNs to learn complex patterns in image data, particularly when it came to differentiating between classes with overlapping features.

7.6 Final thoughts:

In conclusion, while the **Convolutional Neural Network (VGG11)** demonstrated impressive performance during training, its tendency to overfit to the training data resulted in a significant performance drop on the test set. This underlines the importance of regularization techniques and careful hyperparameter tuning for CNNs when applied to datasets like CIFAR-10.

On the other hand, the **Multi-Layer Perceptron (MLP)** achieved the best balance between training and test performance. Despite its limitations compared to the CNN, particularly in handling fine-grained visual distinctions, the MLP's ability to generalize well to unseen data makes it the most practical model for this classification task.

For future work, it would be beneficial to explore advanced CNN architectures, such as ResNets or deeper VGG models, with more regularization techniques like data augmentation, dropout, and weight decay to prevent overfitting and improve generalization.