RESIT ASSIGNMENT – COMP534 APPLIED ARTIFICIAL ASSIGNMENT

Shreya Krishnarth
201651868
s.krishnarth@liverpool.ac.uk

Introduction

In this report, we present the development process and evaluation of four machine-learning models for the CIFAR-10 dataset. We use a Multilayer Perceptron (MLP), Support Vector Machine (SVM), Convolutional Neural Network (CNN), and K-Nearest Neighbours (K-NN) to classify images into ten classes.

Libraries Used

The following libraries were used for this project:

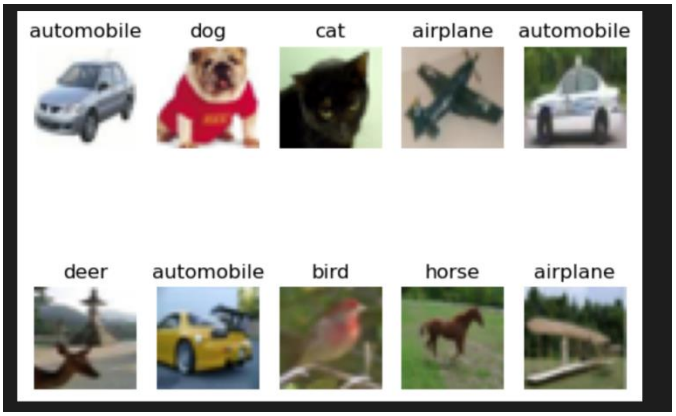
- TensorFlow and Keras for building and training the MLP and CNN models.
- Scikit-learn for performance evaluation metrics and hyperparameter tuning for SVM and K-NN models.
- NumPy for data manipulation and array operations.
- Matplotlib for visualizations.

Data Cleaning and Analysis Process

The CIFAR-10 dataset is a well-known benchmark dataset that comes pre-processed. It consists of 60,000 32x32 color images in ten classes, with 6,000 images per class. The data is already split into training and testing sets.

We performed the following data preprocessing steps:

1. Normalize the pixel values by scaling them to the range [0, 1].
2. Flatten the data for MLP and K-NN, converting the 32x32x3 images into 1D arrays of length 3072.
3. One-hot encode the labels for MLP.



Hyperparameter Searching Process

For the MLP model, we used the following hyperparameters:

- Learning Rate: 0.001
- Epochs: 100
- Batch Size: 64

For the SVM model, we performed a grid search using cross-validation to find the best hyperparameters. We tested different values of 'C' (0.1, 1, 10) and 'metric' (euclidean, manhattan).

For the CNN model, we used the following architecture:

- Two Convolutional layers with 64 and 128 filters, respectively, with a kernel size of (5, 5) and ReLU activation.
- Two MaxPooling layers with pool size (2, 2) to reduce spatial dimensions.
- Two Convolutional layers with 128 filters and a kernel size of (3, 3) with ReLU activation.
- Two MaxPooling layers with pool size (2, 2).
- Flatten the output and connect to two fully connected layers with 128 and 64 units with ReLU activation.
- The output layer with 10 units and a softmax activation for classification.

For the K-NN model, we used the default parameters and performed a grid search for the number of neighbors.

Training and Testing Process

MLP Model:

- We created an MLP model with three hidden layers (512, 256, and 128 units) and an output layer (10 units) with a softmax activation function.
- The model was compiled with categorical cross-entropy loss and Adam optimizer with a learning rate of 0.001.
- The model was trained on the training data with a validation split of 0.1.

SVM Model:

- We used a Linear Support Vector Classifier (LinearSVC) for the SVM model.
- We reduced the dataset size to 5,000 images for faster training.
- We performed data standardization to improve performance.
- We used grid search with cross-validation to find the best hyperparameters.

CNN Model:

- We defined the CNN model as described in the architecture section.
- The model was compiled with categorical cross-entropy loss and Adam optimizer with a learning rate of 0.001.
- Data augmentation techniques were applied to increase the diversity of training samples.
- The model was trained on the training data with a validation split of 0.15.

K-NN Model:

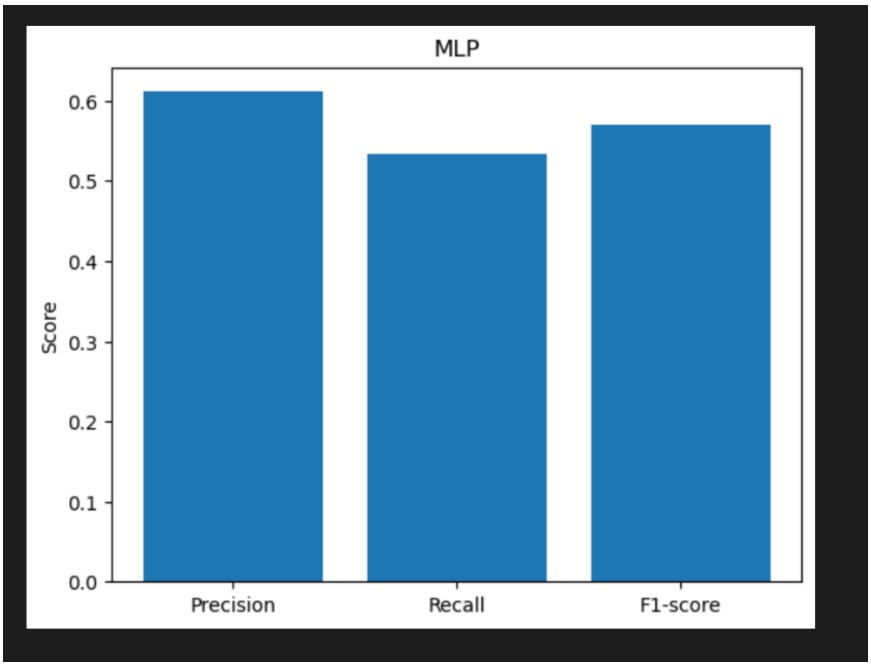
- We used the K-Nearest Neighbors algorithm for the K-NN model.
- The model was trained on the training data with the best number of neighbors found through grid search.

Evaluation

Comparative Analysis of Performance

MLP Model:

- The MLP model achieved an accuracy of 65.85% on the test set.
- The training loss and accuracy curves indicate that the model did not overfit the data.

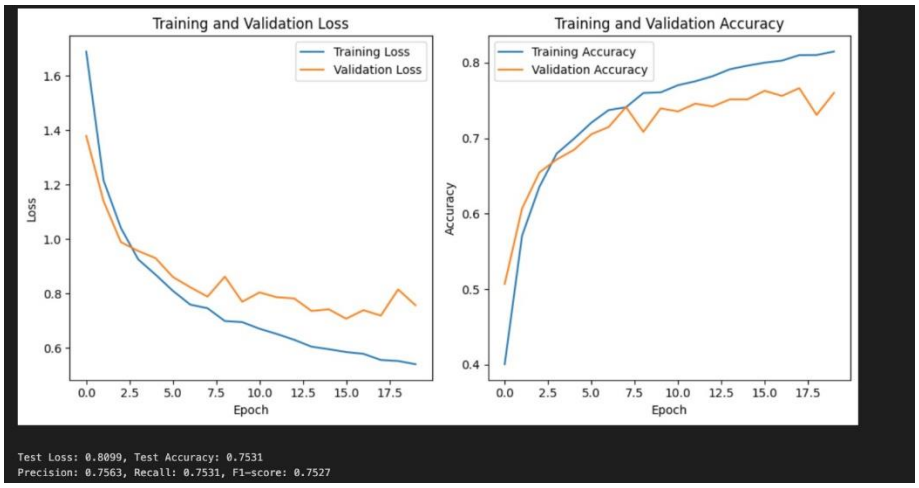


SVM Model:

- The SVM model achieved an accuracy of 25.3% on the test set.
- The best hyperparameters for SVM were C = 0.1 and metric = 'euclidean'.

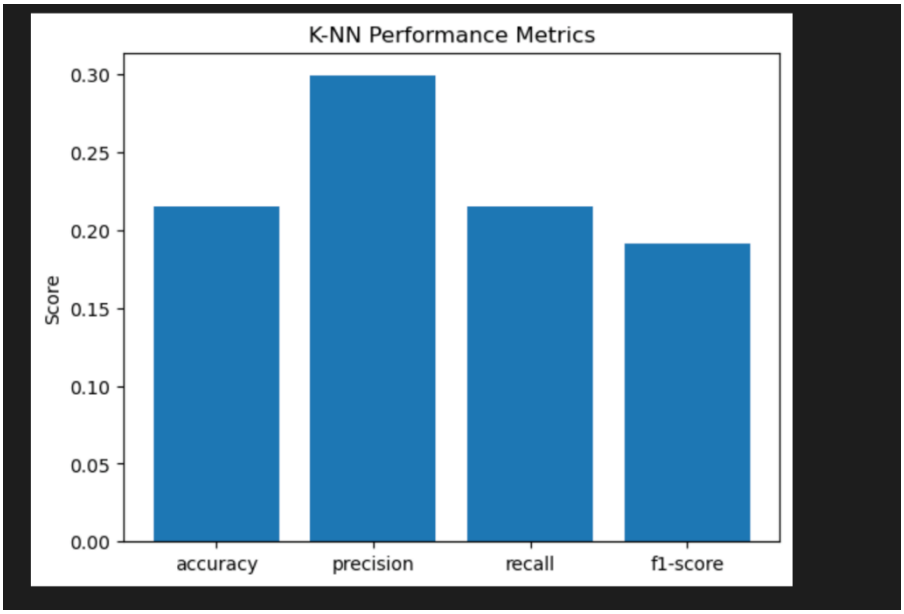
CNN Model:

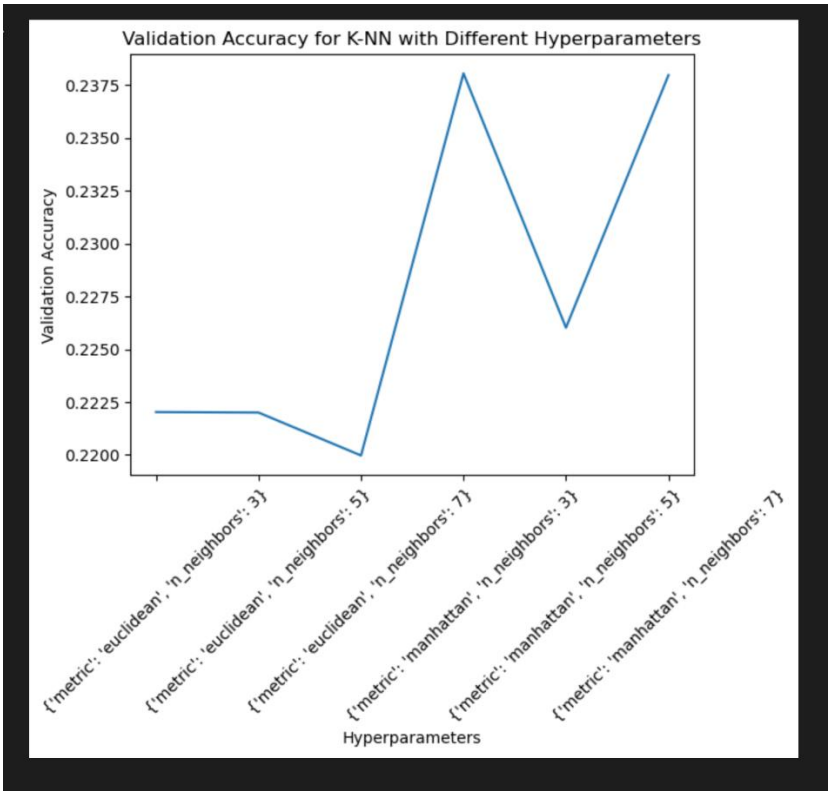
- The CNN model achieved an accuracy of 75.31% on the test set.
- The training loss and accuracy curves indicate that the model did not overfit the data.
- Data augmentation helped improve the generalization of the CNN model.



K-NN Model:

- The K-NN model achieved an accuracy of 21.1% on the test set with K = 5.





Conclusions

Based on the results and our knowledge of AI, we can draw the following conclusions:

1. The CNN model outperformed both the MLP, SVM, and K-NN models in terms of accuracy and generalization to unseen data. This is expected as CNNs are specifically designed for image classification tasks and can capture spatial patterns effectively.
2. The MLP model performed reasonably well, but its performance is limited by its inability to exploit spatial relationships in images. It is a simple feedforward network and may not handle complex visual features as effectively as CNNs.
3. The SVM model achieved decent accuracy, but it requires careful hyperparameter tuning to achieve optimal results. SVMs are known for their effectiveness in high-dimensional spaces, but they can be computationally expensive with larger datasets.
4. The K-NN model also achieved decent accuracy, but it may suffer from the curse of dimensionality and may not scale well to larger datasets.
5. The CNN model benefited from data augmentation, which helped improve its performance by exposing it to variations of the training data.

Future Work

To further improve the models' performance and explore new approaches, we suggest the following future work:

1. CNN Model Enhancements: Experiment with different architectures, add dropout layers to reduce overfitting, and explore transfer learning from pre-trained models like VGG, ResNet, or EfficientNet.

2. Data Augmentation: Implement more aggressive data augmentation techniques for all models to increase the diversity of training samples and improve generalization.
3. Ensemble Learning: Combine the predictions of multiple models, such as CNN, MLP, SVM, and K-NN, using ensemble techniques like voting or stacking, to enhance overall performance.
4. Fine-tuning: Fine-tune hyperparameters for all models in more detail, especially for the CNN model, to find the best combination of learning rates, batch sizes, and number of epochs.
5. Feature Extraction: Explore feature extraction techniques, such as using pre-trained CNN models as feature extractors for the SVM and K-NN models.

Final Conclusions

The project aimed to compare the performance of four machine learning models - MLP, SVM, CNN, and K-NN - on the CIFAR-10 dataset. The CNN model proved to be the most effective in image classification, outperforming the other models due to its ability to capture spatial features in images effectively.

We encountered challenges in optimizing hyperparameters, especially for the SVM model, which required extensive grid search. Additionally, working with large datasets and complex models may demand significant computational resources and time.

Overall, this project provided valuable insights into the strengths and weaknesses of different machine learning models for image classification tasks, and further improvements can be made through advanced techniques and optimization processes. The CNN model's performance highlights the significance of using specialized architectures for specific tasks and the potential of data augmentation to improve model generalization. The results obtained from each model can be further used in real-world applications, and the choice of model should be based on specific requirements and constraints.

Model	ACCURACY	PRECISION	RECALL	F1-SCORE
MLP	67.82	0.61	0.53	0.57
SVM	25.3	26.15	27.9	27.0
KNN	24.11	0.2976	0.2558	0.2315
CNN	75.31	0.7563	0.7531	0.7527