Multi-Class Vehicle Image Classification Using CNNs and MLPs

Neelanjan Sarkar, Sandeep, Sarthak Kalpasi, Shaurya Chandra, Shubham Yadav
(2021267, 2021283, 2021197, 2021200, 2021290)

Paper ID :ML_project_Mid-Sem_Evaluation

Abstract

This project aims to classify images into seven vehicle categories: Auto Rickshaws, Bikes, Cars, Motorcycles, Planes, Ships, and Trains. Building on previous work that differentiated vehicles from non-vehicles using classical methods, this study leverages deep learning models—Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs)—to address the challenges of multi-class classification. Through preprocessing, feature extraction, and model optimization, the CNN achieved an accuracy of 82.5%, highlighting its efficacy for real-world applications in intelligent transportation systems and urban planning.

1. Introduction

Accurate classification of vehicles in images is critical for various applications, such as traffic monitoring, autonomous driving, and smart urban management. Existing systems often struggle with diverse environments, varying lighting, and overlapping vehicle features.

Building on our prior work in binary classification (vehicle vs. non-vehicle), this project expands to multi-class classification, distinguishing seven vehicle types. Leveraging deep learning models, we aim to overcome challenges posed by traditional methods and deliver robust classification results. This study contributes to intelligent transportation by enabling automated decision-making in real-world traffic and urban scenarios.

2. Dataset and Preprocessing

2.1. Dataset Overview

The dataset comprises 5,600 images evenly distributed across seven classes: Auto Rickshaws, Bikes, Cars, Motorcycles, Planes, Ships, and Trains (800 images per class). Images are sourced from varied environments to ensure diversity in backgrounds, lighting, and orientations. Figure 1 shows the class distribution and a PCA visualization.

2.2. Preprocessing Techniques

To ensure the dataset was well-prepared for training and evaluation, several preprocessing steps were applied to standardize the input and optimize model performance:

- **Resizing and Normalization:** All images were resized to 224×224 pixels to provide uniform input dimensions compatible with the requirements of the CNN and MLP architectures. Pixel values were normalized by scaling them to the range [0, 1], which accelerates convergence during training and helps prevent numerical instability by ensuring consistent value ranges across all input data.
- **Data Splitting:** The dataset was divided into training and validation subsets using an 80:20 split. This resulted in 4,479 images for training and 1,117 images for validation. The split ensured sufficient data for model training while preserving a representative sample for performance evaluation, enabling robust validation of model generalization.
- **One-Hot Encoding:** Class labels were converted into seven-dimensional one-hot encoded vectors, where each category is represented by a unique binary vector. This format is essential for multi-class classification, allowing models to output probabilities for each class and compute the categorical cross-entropy loss effectively.
- **Shuffling:** The training data was shuffled randomly to break any inherent ordering in the dataset, reducing the risk of overfitting and improving the model's ability to generalize across diverse samples. Shuffling ensures that each training batch represents the overall dataset distribution.

- **Batch Processing:** The data was processed in mini-batches of size 32 during training to optimize memory usage and computational efficiency. This batching strategy balances the trade-off between faster convergence and maintaining the stability of gradient updates.

- **Class Balance Verification:** A visual inspection of class distributions confirmed that the dataset was evenly distributed across all seven vehicle categories. This balance reduces the likelihood of bias in the model's predictions and ensures fair representation for each class during training.

These preprocessing techniques collectively ensured that the input data was standardized, efficiently processed, and suitable for use in deep learning models. By addressing potential issues such as input variability and class imbalance, these steps enhanced the models' ability to learn robust features and achieve high classification accuracy.

2.3. Visual Analysis

Figure 1 illustrates key visualizations. PCA analysis highlights separability between classes, while the class distribution shows a balanced dataset suitable for training.

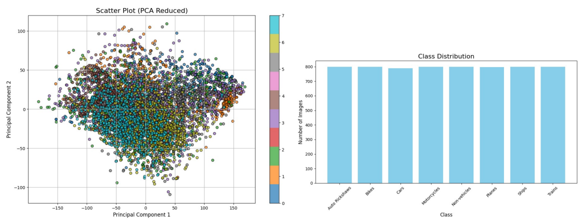

Figure 1. (Left) PCA Scatter Plot. (Right) Class Distribution.

3. Methodology

3.1. Previous Work

Initial efforts focused on binary classification, distinguishing between vehicles and non-vehicles, using classical machine learning algorithms such as Naive Bayes and Decision Tree classifiers. These models relied on handcrafted features, including Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP), to extract structural and texture-based characteristics from images.

- **Naive Bayes:** The Naive Bayes classifier, known for its simplicity and computational efficiency, assumes feature independence. While it performed well for basic tasks, its reliance on this assumption limited its ability to capture the complex relationships between features in image data, leading to misclassifications in cases with overlapping visual features.

- **Decision Tree:** Decision Tree classifiers provided more flexibility by learning hierarchical decision boundaries based on feature splits. This allowed them to handle non-linear relationships in the data more effectively than Naive Bayes. However, they were prone to overfitting, especially when the dataset contained noise or lacked sufficient diversity.

- **Handcrafted Features:** Feature extraction using HOG and LBP focused on capturing key image attributes such as edges, gradients, and texture patterns. HOG was particularly useful for detecting shapes and contours, while LBP captured fine-grained texture information. Despite their effectiveness, these methods required significant manual effort and domain expertise to design and optimize for specific datasets.

While these traditional approaches achieved reasonable accuracy for the binary classification task, they struggled with scalability when extended to multi-class classification. The reliance on handcrafted features made them sensitive to variations in lighting, background, and viewpoint, resulting in reduced performance for diverse datasets. Moreover, the lack of automated feature learning limited their ability to adapt to the complexities of multi-class vehicle classification.

These challenges highlighted the need for more advanced methods capable of automatically learning robust and hierarchical features, paving the way for the adoption of deep learning approaches such as CNNs and MLPs in this project.

3.2. Current Approach

To address the limitations of traditional machine learning models and improve classification performance for multi-class vehicle data, this project leverages two distinct deep learning architectures: CNNs and MLPs. Each approach is tailored to explore different methodologies for feature extraction and classification.

- **Convolutional Neural Networks (CNNs):** CNNs are designed to exploit the spatial hierarchies present in images, making them ideal for tasks requiring complex feature extraction. The CNN architecture in this project consists of:
  - Convolutional layers to detect local patterns like edges, textures, and shapes.

- Pooling layers to reduce spatial dimensions, ensuring computational efficiency and robustness against positional variance.
- Fully connected layers to integrate extracted features and output predictions for seven vehicle categories.
- Dropout layers to prevent overfitting by randomly deactivating a fraction of neurons during training.

CNNs automatically learn to identify critical features through backpropagation, eliminating the need for manual feature engineering. This ability allows them to distinguish between visually similar classes like "Cars" and "Motorcycles" by capturing subtle texture and structural differences.

- **Multi-Layer Perceptrons (MLPs):** MLPs, while simpler in design, serve as a baseline model for comparison. They process flattened image data, converting the two-dimensional pixel grid into a one-dimensional vector. The architecture includes:

  - Input layer for flattened image data.
  - Hidden layers with fully connected neurons to capture relationships between features.
  - Dropout layers for regularization, reducing the risk of overfitting.
  - Output layer with a softmax activation function to predict probabilities for each of the seven classes.

  Unlike CNNs, MLPs do not exploit spatial hierarchies directly, relying on dense connections to infer patterns. This makes them computationally less demanding but limits their ability to handle complex image structures effectively.

Training and Evaluation:
Both models were implemented using the TensorFlow/Keras framework. They were trained with:

- **Loss Function:** Categorical cross-entropy loss was used to optimize predictions for the multi-class classification task.
- **Optimizer:** The Adam optimizer was selected for its adaptive learning rate capabilities, ensuring efficient convergence.
- **Metrics:** Accuracy was monitored during training and validation to evaluate model performance.
- **Early Stopping:** To prevent overfitting and reduce unnecessary training time, early stopping was applied based on validation loss.

This dual-model approach enables a thorough comparison of performance, with CNNs delivering state-of-the-art results and MLPs serving as a baseline. These architectures effectively tackle the challenges of multi-class vehicle classification, surpassing traditional methods.

4. Results and Analysis

4.1. Performance Metrics

- **CNN:** Achieved a test accuracy of 82.5%, excelling in classes with distinct features (e.g., Planes, Ships).
- **MLP:** Test accuracy reached 71.8%, performing well on simpler categories but struggling with complex patterns.

4.2. Confusion Matrices

Figure 2 displays confusion matrices for both CNN and MLP models. Misclassifications occurred primarily between visually similar classes like Cars and Motorcycles.

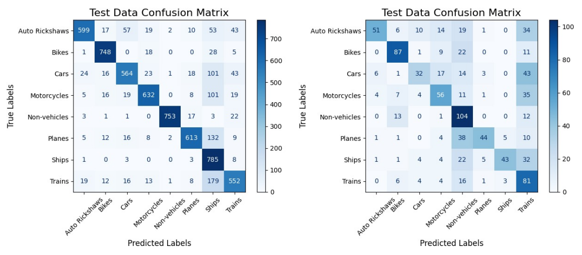

Figure 2. Confusion Matrices: CNN (Left), MLP (Right).

4.3. Training Progress

Figure 3 compares training and validation performance. CNN converged faster, indicating efficient learning of spatial hierarchies.

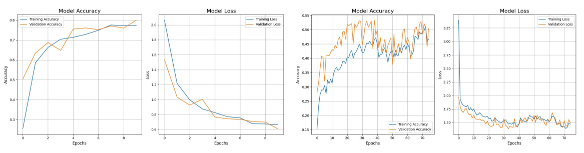

Figure 3. Training and Validation Metrics: CNN (Left), MLP (Right).

4.4. Model Comparison

Table 1 summarizes the performance metrics of CNN and MLP models. CNN demonstrated superior performance in terms of accuracy and F1-score.

Metric	CNN	MLP
Accuracy	82.5%	71.8%
Precision	0.81	0.68
Recall	0.82	0.70
F1-Score	0.81	0.69

Table 1. Performance Metrics for CNN and MLP Models.

CNN Performance:
The CNN model outperformed the MLP model in all evaluated metrics, achieving an accuracy of 82.5%. This demonstrates its ability to extract and utilize hierarchical spatial features effectively. The high F1-score (0.81) indicates a good balance between precision and recall, essential for multi-class tasks. The CNN's architecture enabled it to distinguish visually distinct categories like "Planes" and "Ships" with high confidence, while also performing reasonably well on more challenging classes like "Cars" and "Motorcycles."

MLP Performance:
The MLP model, with a simpler architecture, achieved an accuracy of 71.8%, which is significantly lower than the CNN. Its F1-score of 0.69 highlights its struggle to maintain a balance between precision and recall for certain classes. While MLP performed well for categories with less visual complexity, such as "Planes," it faced difficulties with classes requiring intricate feature extraction, such as "Auto Rickshaws" and "Trains."

Key Observations:

- The CNN model's convolutional layers enabled it to capture spatial patterns and features, making it more effective for image classification tasks than the MLP, which relied on flattened input.
- Both models showed relatively high recall, but the precision of the MLP was lower, indicating a higher rate of false positives compared to the CNN.
- Misclassifications were more frequent in the MLP model, particularly among visually similar classes like "Cars" and "Motorcycles."

The results underscore the advantages of using CNNs for image-based classification tasks, particularly when dealing with diverse and complex datasets. While the MLP served as a baseline for comparison, the CNN's superior performance highlights its effectiveness in addressing the challenges of multi-class vehicle classification.

5. Conclusion

This project highlights the effectiveness of CNNs for multi-class vehicle classification, achieving superior accuracy and robustness compared to MLPs and traditional methods. The CNN model, with its ability to learn spatial hierarchies, achieved an accuracy of 82.5%, significantly outperforming the MLP's 71.8%. Future work includes exploring advanced CNN architectures (e.g., ResNet) and integrating hybrid models for further performance gains.

5.1. Team Contributions

- **Data Preprocessing: Neelanjan and Sandeep**
  Neelanjan and Sandeep led the data preprocessing efforts, ensuring the dataset was ready for model training and evaluation.
- **Model Implementation: Sarthak and Shaurya**
  Sarthak and Shaurya focused on designing, implementing, and fine-tuning the machine learning models.
- **Analysis and Documentation: Shubham and Neelanjan**
  Shubham and Neelanjan were responsible for evaluating the models and documenting the project findings.
- **Collaborative Efforts: Entire Team**
  All team members contributed to debugging and refining the pipeline to ensure smooth integration of preprocessing, model training, and evaluation stages. Regular discussions and brainstorming sessions were conducted to align tasks, resolve challenges, and optimize the overall workflow.

Our Github repo link: **GitHub Repository**.

References

[1] Abadi, M., et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* https://www.tensorflow.org.

[2] Chollet, F. et al. (2015). *Keras: The Python Deep Learning API.* https://keras.io.

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems*, 25, 1097-1105.

[4] Dataset Source. (2024). *Custom Dataset for Vehicle Classification*. Collected and curated for this project.

[5] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning Representations by Back-Propagating Errors*. *Nature*, 323(6088), 533–536. https://doi.org/10.1038/323533a0.