# Assignment 2: CNNs for Image Classification

Manusinh Thakor
The University of Adelaide
manusinh.thakor@student.adelaide.edu.au

## 1. Introduction

Convolutional Neural Networks (CNNs) have revolutionized the field of image classification, enabling machines to interpret and categorize visual data with unprecedented accuracy. Inspired by the human visual system, CNNs consist of multiple layers that automatically learn hierarchical feature representations from raw pixel data, effectively capturing complex patterns and structures within images. This deep learning approach has significantly advanced computer vision applications, including object detection, facial recognition, and medical image analysis[2].

The architecture of CNNs typically includes convolutional layers that apply filters to extract features, pooling layers that reduce dimensionality, and fully connected layers that perform classification. The depth and complexity of these networks allow them to model intricate relationships in data, leading to superior performance compared to traditional machine learning methods. Notably, the development of CNNs has been facilitated by the availability of large labeled datasets and advancements in computational resources, particularly Graphics Processing Units (GPUs)[5].

A seminal moment in the evolution of CNNs was the introduction of AlexNet by Krizhevsky et al. in 2012, which achieved remarkable success in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This achievement demonstrated the potential of deep learning models in handling large-scale image classification tasks. Subsequent architectures, such as VGGNet, GoogLeNet, and ResNet, have further pushed the boundaries of performance and efficiency in image classification[6].

In this assignment, we explore the implementation of CNNs for image classification by developing a custom CNN model from scratch and employing pre-trained architectures like ResNet-18 and MobileNetV2. The custom model allows for a foundational understanding of CNN components and their interactions, while the pre-trained models leverage transfer learning to achieve high accuracy with reduced training time and computational resources. This approach provides a comprehensive perspective on the design and application of CNNs in image classification tasks.

## 2. Related Work

Convolutional Neural Networks (CNNs) have become foundational in image classification, driving advancements in computer vision by learning hierarchical patterns in visual data. The introduction of AlexNet by Krizhevsky et al. in 2012 marked a milestone, achieving remarkable accuracy on the ImageNet dataset and showcasing CNNs' potential in large-scale image classification tasks. AlexNet employed deep convolutional layers and ReLU activations to overcome vanishing gradient issues, setting the stage for further innovations [2].

Following AlexNet, Simonyan and Zisserman introduced VGGNet, a model characterized by its depth and the use of small convolutional filters. VGGNet demonstrated that deeper networks could improve classification performance, albeit with increased computational demands . To address computational efficiency, GoogLeNet, developed by Szegedy et al., introduced the Inception module, which processes multi-scale features in parallel, effectively balancing performance and efficiency [6]. ResNet, proposed by He et al., revolutionized CNN design by introducing residual learning with skip connections, which mitigated the degradation problem encountered in very deep networks. This innovation enabled ResNet to reach unprecedented depths, making it highly effective in complex image classification tasks [1].

More recently, Sandler et al. developed MobileNetV2, a model optimized for mobile and embedded devices. By using depthwise separable convolutions and inverted residuals, MobileNetV2 achieves high accuracy with a reduced computational footprint, making it suitable for resource-constrained environments. This model represents the latest in CNN architecture, focusing on efficiency without sacrificing performance [4].

Collectively, these advancements have established a spectrum of CNN architectures, from deep and computationally intensive models like ResNet to lightweight architectures like MobileNetV2. These models have broadened CNN applications across fields, from autonomous driving to medical image analysis. Including architecture diagrams for AlexNet, VGGNet, ResNet, and MobileNetV2 can pro-

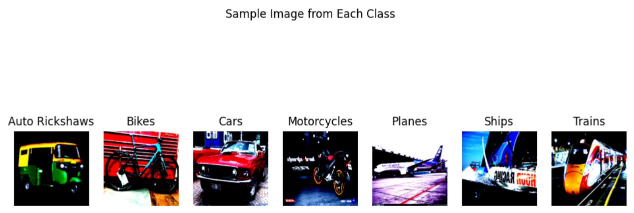


Figure 1. Classes

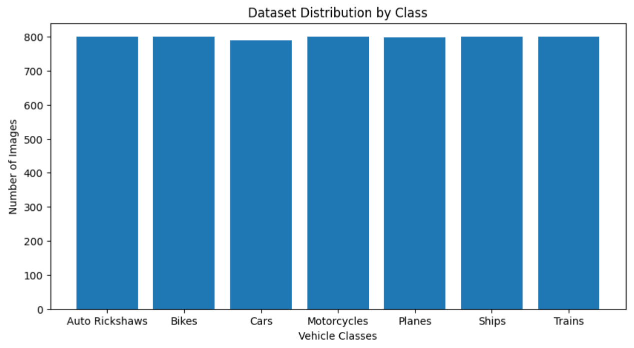


Figure 2. Class Dataset Distribution

vide visual clarity on how each innovation contributes to performance, depth, or efficiency.

## 3. Methodology

### 3.1. Dataset

In this project, I aimed to classify vehicle images into multiple categories using convolutional neural networks (CNNs). The dataset used was the Vehicle Classification Dataset from Kaggle [3] , which consists of various classes representing different vehicle types. The dataset was preprocessed, resized, and normalized to make it suitable for model training.

I used a dataset structured in folders, with each folder representing a different vehicle class, to facilitate multi-class classification. To ensure consistency across model inputs, I applied preprocessing transformations using PyTorch's `transforms` module. Each image was resized to 256x256 pixels and normalized with mean values of `[0.485, 0.456, 0.406]` and standard deviations of `[0.229, 0.224, 0.225]`, which are typical for models trained on natural images.

### 3.2. Model Implementation

I trained two types of models: a custom CNN designed from scratch, and two pre-trained architectures, ResNet-18 and MobileNet V2, without using pre-trained weights to observe their performance on this specific dataset. For the CNN model built from scratch, the architecture included basic convolutional and pooling layers, followed by fully connected layers for classification. ResNet-18 and MobileNet V2 were chosen for their robustness in image classification tasks and efficient parameter utilization.

### 3.3. Training Strategy

I trained three models for vehicle classification: a custom CNN, ResNet-18 with hyperparameter tuning, and MobileNetV2. The custom CNN was trained on the training and testing datasets, serving as a baseline model.

For ResNet-18, I performed hyperparameter tuning on both the training and validation datasets, testing various optimizer and learning rate combinations. The hyperparameter search included different configurations of Adam and SGD optimizers with varying learning rates, momentums, and weight decay values. After selecting the best-performing parameters, I retrained ResNet-18 on the training and testing datasets to evaluate its final performance.

Lastly, I trained MobileNetV2 on the training and testing datasets, providing a comparison for the custom CNN and ResNet-18 models in terms of accuracy and computational efficiency. This multi-model training approach allowed for a comprehensive comparison of model performance across different architectures and training strategies.

### 3.4. Evaluation

Each model's performance was evaluated using accuracy metrics and loss curves. I plotted the loss and accuracy curves for both training and testing sets, which allowed me to monitor the training process closely. The best model was selected based on its testing accuracy and overall stability, considering factors such as overfitting signs and generalization capabilities.

## 4. Experimental Analysis

### 4.1. Results of Custom CNN

The training and testing loss curves figure 3 show a clear convergence trend. The training loss consistently decreases over the epochs, indicating effective learning. The testing loss initially follows a similar trend but shows slight fluctuations in the latter epochs, suggesting mild overfitting. This behavior implies that while the model generalizes reasonably well, further regularization or fine-tuning might improve its stability on unseen data

The custom CNN model achieved strong performance metrics on the vehicle classification task, with an overall testing accuracy of **82%**. Additionally, the model recorded a weighted precision, recall, and F1-score of **0.82** each, highlighting its balanced performance across classes.

### 4.2. Results of ResNet-18

For the ResNet-18 model, I tested multiple hyperparameter combinations **??**, including various optimizers (Adam and SGD), learning rates, momentums, and weight decay

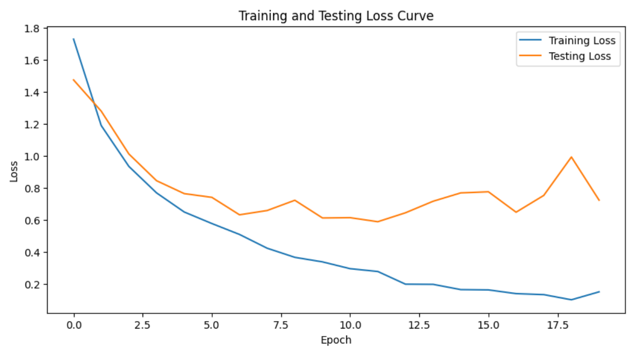


Figure 3. Loss Curve of CNN model from Scratch

| Metric | Value |
|---|---|
| Testing Accuracy | 82 % |
| Weighted Precision | 0.82 |
| Weighted Recall | 0.82 |
| Weighted F1-Score | 0.82 |

Table 1. CNN model's performance across key metrics

values. After evaluating each configuration, I selected the best-performing model and trained it for 120 epochs to assess its performance on the vehicle classification task.

| Optimizer | Learning Rate | Momentum | Weight Decay |
|---|---|---|---|
| Adam | 0.001 | - | 0.0001 |
| Adam | 0.0001 | - | 0.001 |
| SGD | 0.01 | 0.9 | 0.0001 |
| SGD | 0.005 | 0.9 | 0.0001 |
| Adam | 0.01 | - | 0.0001 |
| SGD | 0.01 | 0.9 | 0.0001 |

Table 2. Hyperparameters

The training log for the selected ResNet-18 model shows a gradual improvement in both training and testing loss over the epochs. Initially, both losses decrease steadily, with the model showing consistent gains in accuracy.

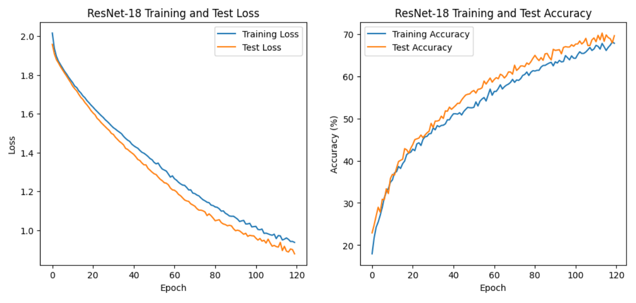


Figure 4. Resnet-18 Final Model Loss Curve

The training accuracy increased from **17.94%** in the first epoch to **67.79%** in the final epoch, while the testing accuracy improved from **22.90%** to **69.59%**. This progression indicates that the model is learning effectively and would likely benefit from additional training iterations to reach maximum accuracy.

| Metric | Value |
|---|---|
| Testing Accuracy | 70% |
| Weighted Precision | 0.70 |
| Weighted Recall | 0.70 |
| Weighted F1-Score | 0.70 |

Table 3. ResNet-18 final model's performance across key metrics

The ResNet-18 model demonstrates good convergence, with loss and accuracy steadily improving throughout training. Although the testing accuracy of 70% indicates room for further improvement, this model could likely achieve even higher performance with additional epochs. The relatively balanced precision, recall, and F1-score across classes suggest that the model generalizes reasonably well across different vehicle types.

This analysis concludes that ResNet-18 is a technically sound model with consistent convergence and balanced performance. Additional training iterations could further enhance its accuracy and overall performance, making it a reliable choice for this multi-class vehicle classification task.

### 4.3. Comparison of ResNet-18 and MobileNetV2 Results

MobileNetV2 performed consistently across most classes, with notably high precision and recall for **Bikes** (precision 0.85, recall 0.93) and **Motorcycles** (precision 0.70, recall 0.88). This model achieved a higher testing accuracy of 74% in just 30 epochs, demonstrating a more efficient training process compared to ResNet-18. This could be attributed to the higher learning rate (0.001), which helped MobileNetV2 learn faster, though it showed slight fluctuations towards the end, suggesting potential overfitting with fewer epochs.

MobileNetV2 demonstrated faster convergence, reaching 74% accuracy in just 30 epochs, likely due to its lightweight architecture and higher learning rate. In contrast, ResNet-18 required more epochs to reach a similar

| Metric | ResNet-18 | MobileNetV2 |
|---|---|---|
| Learning Rate | 0.0001 | 0.001 |
| Testing Accuracy | 0.70 | 0.74 |
| Weighted Precision | 0.70 | 0.74 |
| Weighted Recall | 0.70 | 0.74 |
| Weighted F1-Score | 0.70 | 0.74 |
| Convergence Speed | Slow | Fast |
| Potential Overfitting | Minimal | Slight |

Table 4. Comparison of ResNet-18 and MobileNetV2 Performance on Vehicle Classification

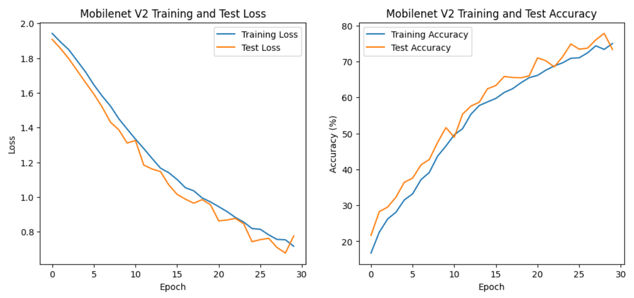
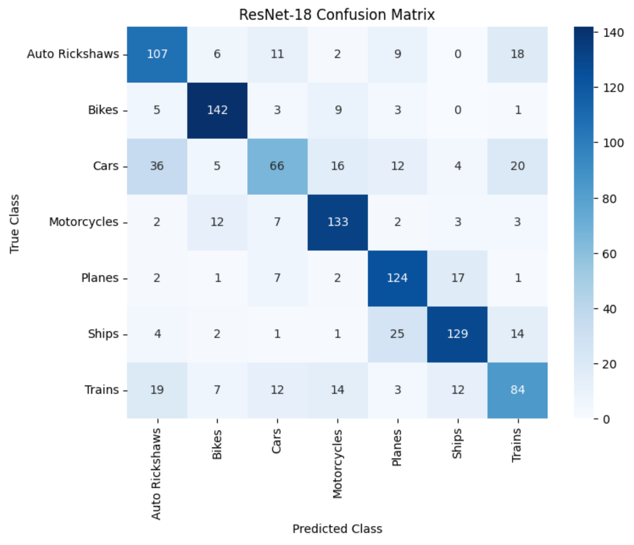


Figure 5. Mobilenet V2 Loss Curve

level of accuracy but showed consistent improvement, suggesting it could further benefit from extended training. MobileNetV2 generally achieved better recall and precision for most classes, indicating superior generalization across vehicle types, especially excelling with classes like Bikes and Motorcycles. ResNet-18, however, showed variability in performance with certain classes, such as Cars and Trains. Overall, MobileNetV2 is well-suited for applications where training time and computational efficiency are essential, while ResNet-18 offers a strong alternative for scenarios where extended training is feasible and further improvements are desired.

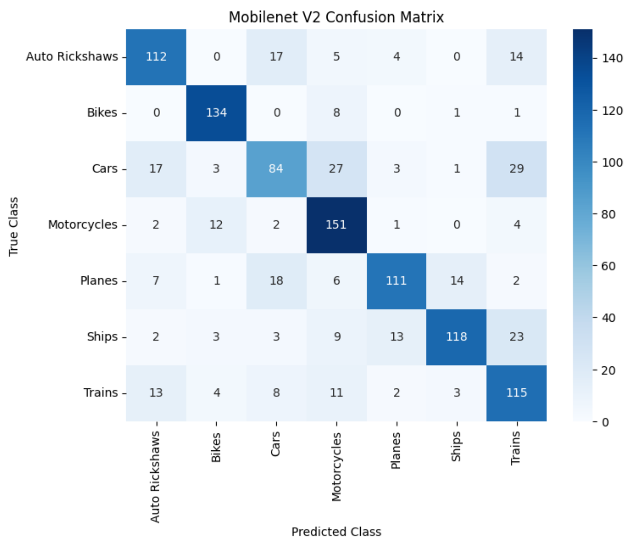


| True \ Predicted | Auto Rickshaws | Bikes | Cars | Motorcycles | Planes | Ships | Trains |
|---|---|---|---|---|---|---|---|
| Auto Rickshaws | 107 | 6 | 11 | 2 | 9 | 0 | 18 |
| Bikes | 5 | 142 | 3 | 9 | 3 | 0 | 1 |
| Cars | 36 | 5 | 66 | 16 | 12 | 4 | 20 |
| Motorcycles | 2 | 12 | 7 | 133 | 2 | 3 | 3 |
| Planes | 2 | 1 | 7 | 2 | 124 | 17 | 1 |
| Ships | 4 | 2 | 1 | 1 | 25 | 129 | 14 |
| Trains | 19 | 7 | 12 | 14 | 3 | 12 | 84 |

Figure 6. Resnet-18 Confusion Matrix

## 5. Code

You can find the code of the project here `https://github.com/Manusinh-Thakor/Assignment_2-Deep-Learning`



| True \ Predicted | Auto Rickshaws | Bikes | Cars | Motorcycles | Planes | Ships | Trains |
|---|---|---|---|---|---|---|---|
| Auto Rickshaws | 112 | 0 | 17 | 5 | 4 | 0 | 14 |
| Bikes | 0 | 134 | 0 | 8 | 0 | 1 | 1 |
| Cars | 17 | 3 | 84 | 27 | 3 | 1 | 29 |
| Motorcycles | 2 | 12 | 2 | 151 | 1 | 0 | 4 |
| Planes | 7 | 1 | 18 | 6 | 111 | 14 | 2 |
| Ships | 2 | 3 | 3 | 9 | 13 | 118 | 23 |
| Trains | 13 | 4 | 8 | 11 | 2 | 3 | 115 |

Figure 7. Mobilenet V2 Confusion Matrix

## 6. Conclusion

In conclusion, this study explored the performance of three CNN models for vehicle classification: a custom CNN model from scratch, ResNet-18, and MobileNetV2. The custom CNN model provided a solid baseline with moderate accuracy, proving useful for initial analysis. MobileNetV2, with its lightweight architecture and higher learning rate, achieved faster convergence, reaching a commendable accuracy within just 30 epochs, demonstrating its efficiency and suitability for applications where training time and computational resources are limited. However, ResNet-18 emerged as the most promising model overall. Despite requiring more epochs to reach optimal performance, ResNet-18 showed steady improvements throughout training and achieved robust accuracy and balanced class performance. Given additional epochs, ResNet-18 has the potential to achieve even higher accuracy, making it the preferred model for this task when computational resources and training time allow. This progression from a simple CNN to MobileNetV2 and finally to ResNet-18 highlights the trade-offs between model complexity, training time, and accuracy in deep learning for image classification.

This table highlights that the **Custom CNN** achieved the highest accuracy and balanced metrics, making it a solid baseline. **MobileNetV2** provided rapid convergence and efficiency, suitable for faster training cycles with moderate accuracy. **ResNet-18**, while slower to converge, displayed continuous improvement and holds potential for even higher accuracy with further training, making it the preferred choice when accuracy and class performance are prioritized over training time.

| Metric | Custom CNN | MobileNetV2 | ResNet-18 |
|---|---|---|---|
| Testing Accuracy | 0.82 | 0.74 | 0.70 |
| Weighted Precision | 0.82 | 0.74 | 0.70 |
| Weighted Recall | 0.82 | 0.74 | 0.70 |
| Weighted F1-Score | 0.82 | 0.74 | 0.70 |
| Convergence Speed | Moderate | Fast | Slow |
| Computational Demand | Low | Moderate | High |
| Suitability for Training Efficiency | Moderate | High | Moderate |
| Potential for Extended Accuracy | Limited | Moderate | High |

Table 5. Final Comparison of Custom CNN, MobileNetV2, and ResNet-18 Performance

## References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[3] Mohamed Maher. Vehicle classification dataset. `https://www.kaggle.com/api/v1/datasets/download/mohamedmaher5/vehicle-classification`, 2021. Accessed: YYYY-MM-DD.

[4] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.