# Vehicle Detection and Classification by Using Fast R-CNN. Technical Report.

ibrahim ahmethan

August 2023

## 1 Introduction

The rapid urbanization and growth of vehicular traffic require efficient systems for vehicle detection and classification. Such systems hold exceptional importance in a variety of applications, ranging from urban traffic management to advanced driver assistance systems (ADAS) [1]. Traditional methods for vehicle detection involved hand-crafted features combined with classifiers like Support Vector Machines (SVM) [2]. However, with the advent of deep learning, Convolutional Neural Networks (CNNs) have become the standard approach for tasks involving image and video analysis due to their capacity to learn hierarchical features directly from data [3].

One of the significant advancements in object detection using deep learning has been the introduction of region-based Convolutional Neural Networks (R-CNNs) [4]. R-CNNs, and their subsequent iterations like Fast R-CNN, have demonstrated a substantial improvement in detection performance by integrating region proposal mechanisms directly within the deep learning framework, reducing the need for separate hand-crafted region-proposing algorithms [5]. Fast R-CNN, in particular, enhances the detection speed by sharing the computation of convolutional layers across multiple proposals, making it suitable for real-time applications like vehicle detection in traffic video feeds [5]. In this project, fast R-CNN is adapted to train the model for the purpose of object detection.

## 2 Network Architecture

The Faster R-CNN with ResNet-50 and FPN is the chosen architecture for vehicle detection. It integrates region proposals directly, utilizing the ResNet-50 backbone for deep feature extraction. The Feature Pyramid Network (FPN) addition allows for efficient multi-scale object detection, making this model both rapid and precise for real-time tasks[5].

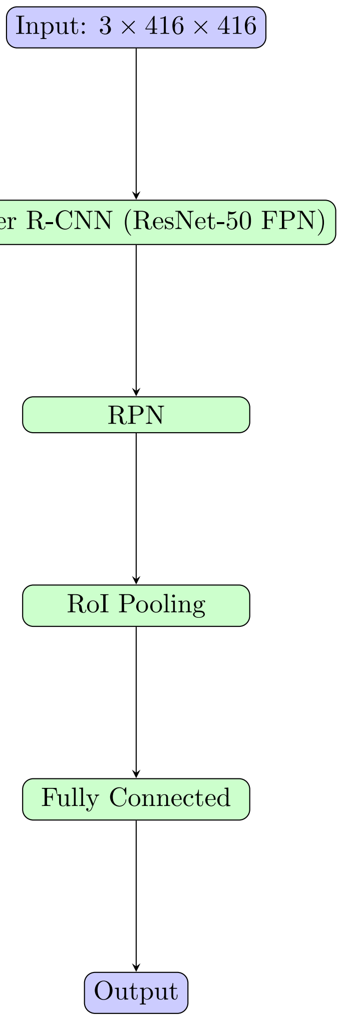

Input: $3 \times 416 \times 416$
Faster R-CNN (ResNet-50 FPN)
RPN
RoI Pooling
Fully Connected
Output

Figure 1: Architecture for vehicle detection using Faster R-CNN with input size $3 \times 416 \times 416$.

## 2.1 Model Choice: Faster R-CNN with ResNet-50 FPN

The selected model for the task of vehicle detection and classification is the Faster R-CNN architecture combined with the ResNet-50 backbone with Feature Pyramid Network (FPN) for feature extraction. This model strikes a balance between accuracy and speed, making it suitable for real-time detection tasks.

## 2.2 Feature Extraction: ResNet-50 with FPN

The foundation of the Faster R-CNN model used in this project is based on the ResNet-50 architecture. ResNet, which stands for Residual Network, is characterized by its “skip connections” or “shortcuts”, which helps in avoiding the vanishing gradient problem during deep network training. The depth of 50 layers ensures a robust feature extraction capability, suitable for a diverse set of objects in images.

Further, the addition of the FPN helps the model in detecting objects at different scales more efficiently. An FPN augments the standard convolutional network with a top-down pathway and lateral connections, assisting in producing feature pyramids with high-level semantic features.

## 2.3 Detection Mechanism: Region Proposal Network (RPN)

One of the notable advancements in the Faster R-CNN over its previous models, the Fast R-CNN, is the introduction of the Region Proposal Network (RPN). The RPN automatically suggests potential bounding boxes in the image where an object might exist. This eliminates the dependency on external methods for region proposals, making the process faster and more integrated.

## 2.4 ROI Pooling

Once the regions are proposed by the RPN, RoI (Region of Interest) pooling is employed to convert these proposed regions into a fixed size so that they can be fed into a fully connected network. It’s a way to transform the varying sizes of the proposals to a consistent size suitable for classification.

## 2.5 Classification and Bounding Box Regression

Once the regions are proposed by the RPN, RoI (Region of Interest) pooling is employed to convert these proposed regions into a fixed size so that they can be fed into a fully connected network. It’s a way to transform the varying sizes of the proposals to a consistent size suitable for classification.

## 2.6 Classification and Bounding Box Regression

After ROI pooling, the model has a series of fully connected layers. These layers help in two main tasks:

1. Classifying the proposed region into one of the object classes (in this case, vehicle types such as ’auto’, ’bus’, ’car’, and so on).

2. Refining the bounding box coordinates to better fit the object.

### 2.7 Hyperparameters and Training Configuration

**Optimizer**: Adam optimizer with a learning rate of $\alpha = 0.001$. The Adam optimizer combines the properties of Adagrad and RMSprop algorithms, making it efficient for deep learning models.

**Loss Function**: Multi-task loss, which is a combination of classification loss (cross-entropy) and bounding box regression loss (smooth L1 loss).

**Batch Size**: Typically set to 32 or 64 for faster convergence.

**Epochs**: A predefined number of iterations through the dataset. Common choices might range from 50 to 200, depending on the dataset size and convergence rate.

**Weight Decay**: A regularization term (if used) to prevent overfitting, typically set around 0.0005.

**Momentum**: Commonly used with SGD optimizer, typically set to 0.9 to stabilize the training direction.

## 3 Dataset

The dataset offers a comprehensive insight into the distribution of different vehicle classes typically encountered in urban and suburban traffic scenarios. Comprising a total of 24,098 vehicle instances, the dataset is diverse and reflective of real-world vehicular density and variety. It encompasses eight major vehicle classes: auto, bus, car, lcv (light commercial vehicle), motorcycle, multiaxle vehicles, tractor, and truck. The predominance of cars and motorcycles, with 11,425 and 7,285 instances respectively, underscores the common vehicular makeup of most modern cities. On the other hand, the presence of specific categories like multiaxle vehicles and tractors provides a detailed understanding of the less common but significant vehicle types on the road. Designed meticulously, this dataset serves as a robust foundation for developing and testing vehicle detection and classification systems.

## 3.1 Dataset Classes Distribution

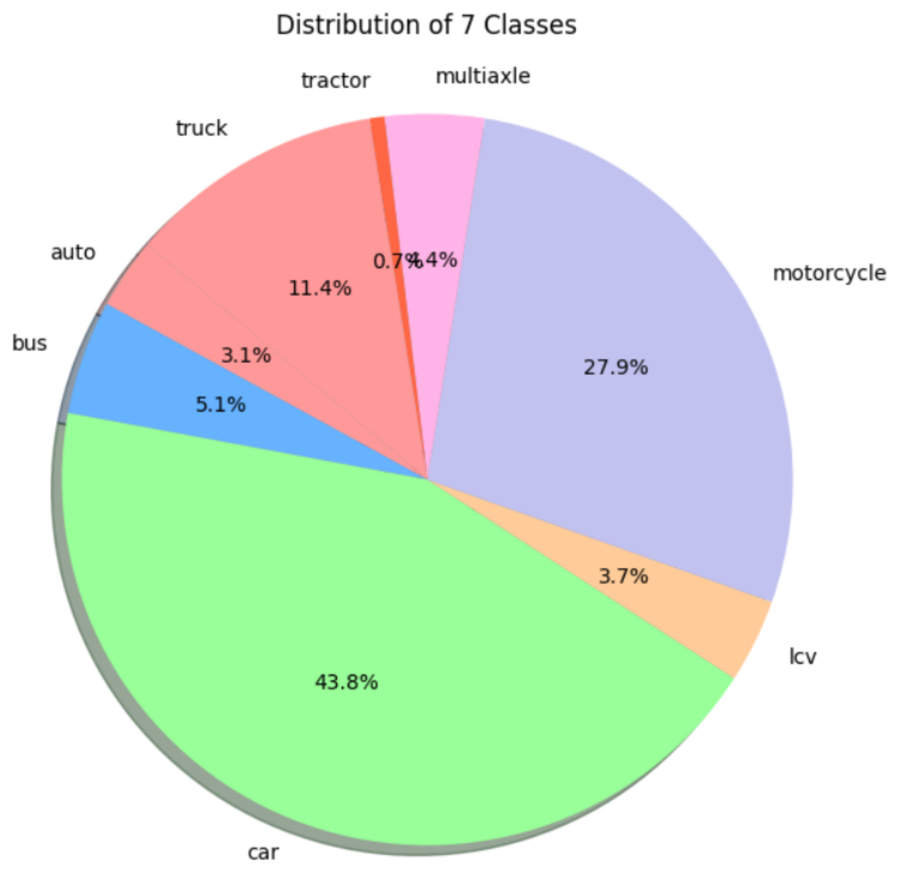


Figure 2: Classes distribution

## 3.2 Dataset Augmentation with Albumentations

Image augmentation is an effective strategy to enhance the diversity of the training set without the need for new data. By applying diverse transformations to our images, we ensure that our model is robust and can generalize better to unseen data. The specific augmentations applied are[8][12]:

1. **Horizontal Flip**: This transformation randomly flips the images on the horizontal axis. It ensures that the model remains orientation-agnostic and doesn't overfit to a particular direction.

2. **ShiftScaleRotate**: A mixture of three separate transformations:

    - **Shift**: Translates the image in both the X and Y axes.

- **Scale**: Resizes the image.
- **Rotate**: Rotates the image, with a specified limit (in our case, up to 15 degrees). The resultant borders from this rotation are managed using a constant value, ensuring image dimensions remain consistent.

3. **CLAHE (Contrast Limited Adaptive Histogram Equalization)**: Enhances the image contrast by redistributing its lightness values. It's a beneficial transformation for images that suffer from shadows or varied lighting conditions.
4. **GaussNoise**: Infuses the image with Gaussian noise. This ensures the model does not memorize specific pixel values, making it robust against naturally occurring noise in real-world images.
5. **LongestMaxSize**: Ensures the image is resized such that its longest dimension matches the specified size (416 pixels in our configuration), maintaining the original aspect ratio.
6. **PadIfNeeded**: If the image's height or width doesn't meet the required dimensions (both set to 416 pixels), this transformation pads the image using a constant value.

It's important to mention that these transformations also ensure the bounding box annotations, using the 'pascal_voc' format. This ensures that as the images undergo transformations, their corresponding bounding boxes are adjusted appropriately. Furthermore, The augmentation process is done during the training process(on the fly) or in other words, online augmentation.

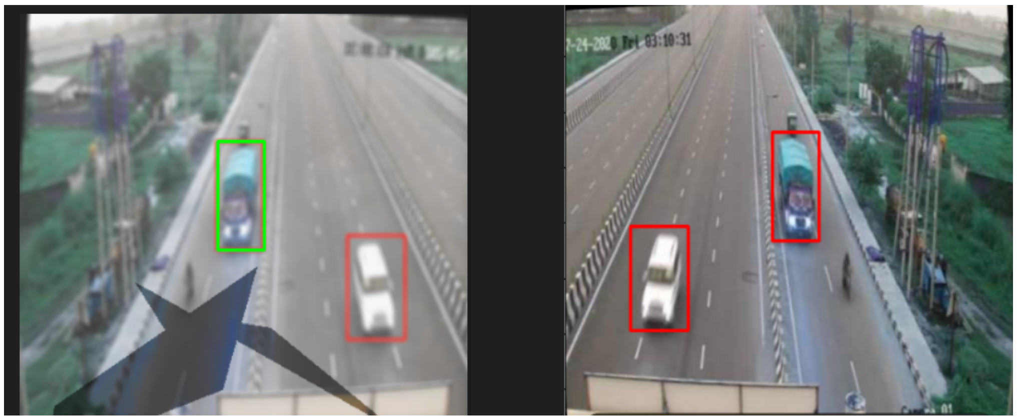


Figure 3: Left Image: Augmented Image. Right Image: Original Image

## 4 Model Evaluation

### 4.1 Training and Testing

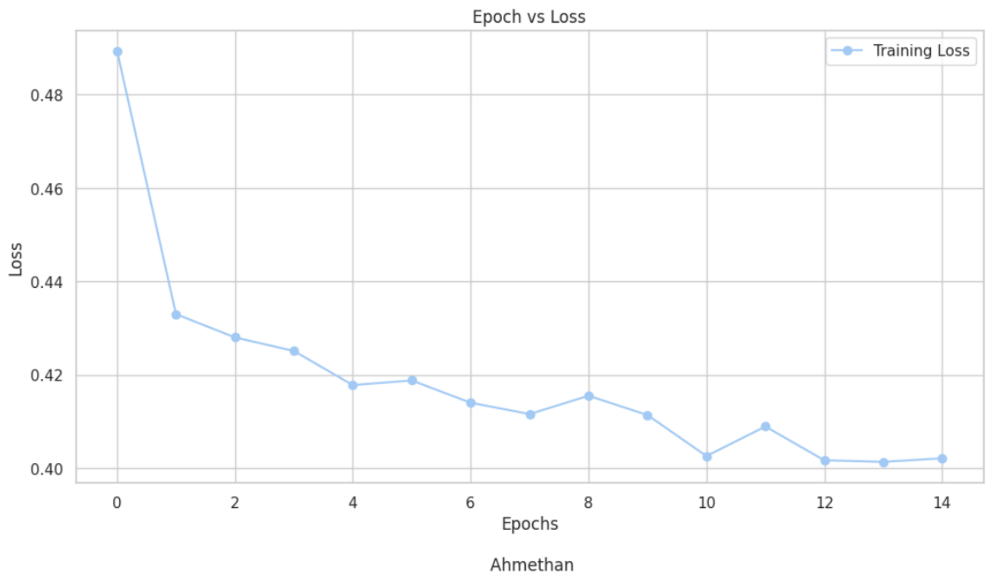


Figure 4: Loss Graph of training

1. **Initial Loss**: Started high at 0.489.
2. **Convergence**: Loss generally decreased over time, indicating learning.
3. **Fluctuations**: Minor ups and downs hint at data noise or model tuning needs.
4. **Potential Overfitting**: Fluctuations in later epochs could suggest overfitting. Comparing with training loss can confirm.
5. **Stability**: Loss stabilized around 0.40-0.41 in later epochs.
6. **Optimal Epoch**: Lowest loss at epoch 13 (0.4017).

In our recent assessment of the vehicle detection and classification model, we discerned challenges in its ability to consistently and accurately detect and classify specific vehicle types. A few potential reasons for these challenges include:

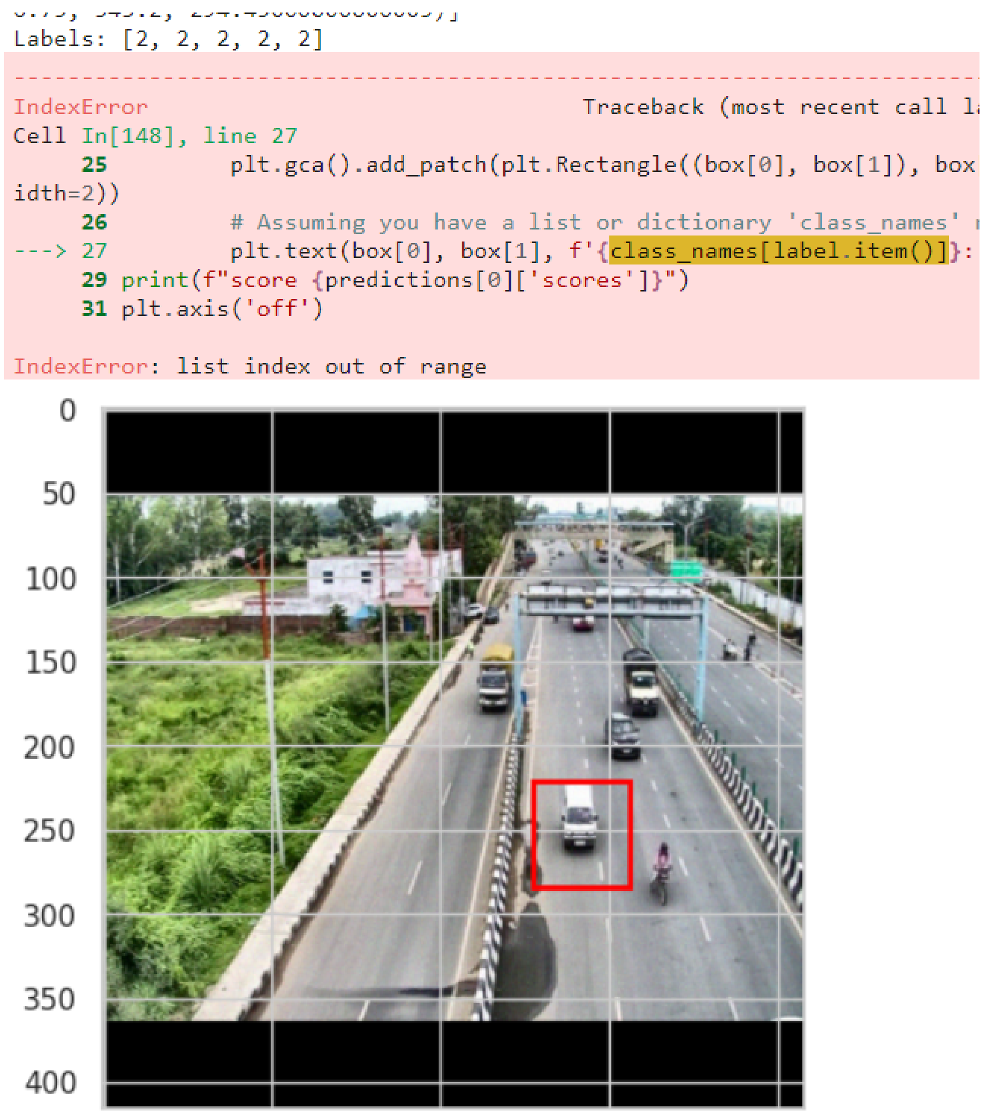
1. **Data Distribution:** The training dataset may not adequately represent the real-world diversity of vehicles, potentially leading to biases in detection and classification.
2. **Model Complexity:** Our current architecture might not have sufficient complexity to capture intricate patterns across all vehicle types.
3. **Feature Extraction:** The model might be prioritizing less relevant features for classification, overshadowing critical discriminative features.
4. **Hyperparameter Tuning:** Suboptimal hyperparameters can limit the model's learning capability and generalization.

Given the aforementioned potential pitfalls, we recommend:

1. **Dataset Enrichment:** Procure or generate more diverse examples of vehicles, with an emphasis on underrepresented or misclassified categories. An external data source or crowdsourcing could provide added richness.
2. **Model Evaluation:** Consider employing advanced architectures or ensemble methods that combine the strengths of multiple models to improve overall performance.
3. **Feature Engineering:** Investigate the primary features the model currently leverages for decision-making and explore ways to refine or introduce new, more predictive features.
4. **Hyperparameter Optimization:** Utilize methods like grid search or Bayesian optimization to find an optimal set of hyperparameters.
5. **Regularization Techniques:** Incorporate methods like dropout or early stopping to reduce potential overfitting and enhance generalization.
6. **Feedback Loop:** Establish a system where misclassifications can be flagged, analyzed, and fed back into the training process for continuous model improvement.

**Conclusion**

By addressing these potential issues systematically, we aspire to enhance our model's accuracy and reliability in vehicle detection and classification.



```
[illegible]
Labels: [2, 2, 2, 2, 2]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call l
Cell In[148], line 27
     25         plt.gca().add_patch(plt.Rectangle((box[0], box[1]), box
idth=2))
     26         # Assuming you have a list or dictionary 'class_names' 
---> 27         plt.text(box[0], box[1], f'{class_names[label.item()]}:
     29 print(f"score {predictions[0]['scores']}")
     31 plt.axis('off')

IndexError: list index out of range
```

Figure 5: Correctly Detected Vehicle

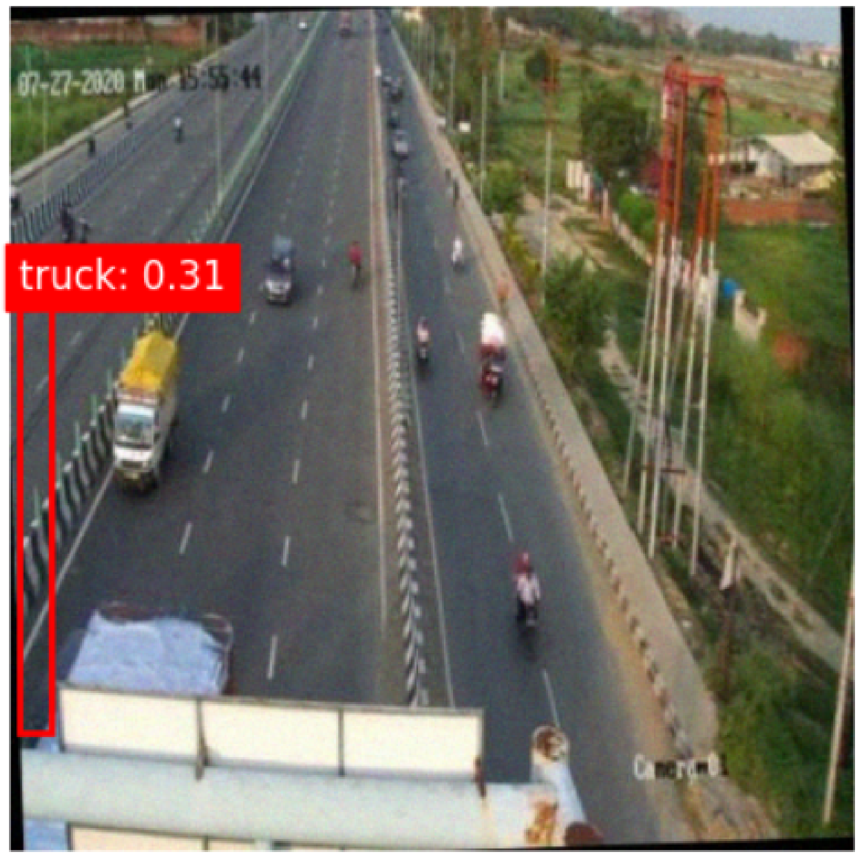


Figure 6: Falsely Detected Vehicle

## References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[2] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings*

*of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[5] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[6] `https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html`

[7] `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`

[8] `https://towardsdatascience.com/complete-guide-to-data-augmentation-for-computer-vision-`

[9] `https://pytorch.org/vision/stable/transforms.html`

[10] `https://pytorch.org/vision/stable/generated/torchvision.transforms.Compose.html#torchvision.transforms.Compose`

[11] `https://pytorch.org/vision/stable/auto_examples/plot_transforms_v2_e2e.html#sphx-glr-auto-examples-plot-transforms-v2-e2e-py`

[12] `https://albumentations.ai/`

[13] `https://pytorch.org/vision/stable/auto_examples/plot_transforms_v2.html`

[14] `https://github.com/cleanlab/cleanlab/blob/master/cleanlab/object_detection/filter.py`

[15] `https://github.com/pytorch/vision/tree/main`