Vehicle Re-identification Using a Multi-View Vehicle Dataset

by

Anubhab Guha

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2022 by the
Graduate Supervisory Committee:

Yezhou Yang, Chair
Duo Lu
Ayan Banerjee

ARIZONA STATE UNIVERSITY

May 2022

ABSTRACT

There has been an explosion in the amount of data on the internet because of modern technology – especially image data – as a consequence of an exponential growth in the number of cameras existing in the world right now; from more extensive surveillance camera systems to billions of people walking around with smartphones in their pockets that come with built-in cameras. With this sudden increase in the accessibility of cameras, most of the data that is getting captured through these devices is ending up on the internet. Researchers soon took leverage of this data by creating large-scale datasets. However, generating a dataset – let alone a large-scale one – requires a lot of man-hours. This work presents an algorithm that makes use of optical flow and feature matching, along with utilizing localization outputs from a Mask R-CNN, to generate large-scale vehicle datasets without much human supervision. Additionally, this work proposes a novel multi-view vehicle dataset (MVVdb) of 500 vehicles which is also generated using the aforementioned algorithm.

There are various research problems in computer vision that can leverage a multi-view dataset, e.g., 3D pose estimation, and 3D object detection. On the other hand, a multi-view vehicle dataset can be used for a 2D image to 3D shape prediction, generation of 3D vehicle models, and even a more robust vehicle make and model recognition. In this work, a ResNet is trained on the multi-view vehicle dataset to perform vehicle re-identification, which is fundamentally similar to a vehicle make and recognition problem – also showcasing the usability of the MVVdb dataset.

DEDICATION

Mom and dad, this for you. Thanks for trusting me and being patient with me.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor, Dr. Yezhou Yang, for his continuous support, guidance, and patience over the years through the ups and downs of my Master's. His steady encouragement has let me purse my interest in research during the course of my Master's. Because of his class in Spring '20, CSE 598: Perception in Robotics, I was introduced to Computer Vision, and his immense knowledge and experience encouraged me to take up this task.

Next, I would like to thank Dr. Duo Lu for his steady encouragement and providing a moral support throughout the course of my research. When I started working on this project, Duo was a PhD student and just another member at our research lab. Getting to see him become an assistant professor and a member of my thesis committee feels like a happy ending to this journey. I would also like to express my gratitude to him for his immense contribution towards my thesis. This would not have been possible without him.

I would like to thank Dr. Ayan Banerjee as well for being a member in my thesis committee. His class CSE 535: Mobile Computing really helped me strengthen my knowledge in Computer Vision.

Finally, I would like to take this opportunity to thank my mom, dad, and my sister for everything they have done for me. I would also like to thank my lab mate Prabal Bijoy Dutta for his continuous support in my research. Additionally, I would like to thank my friends Aranyak Maity, Kevin Kim, and many others for their help and moral support.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION AND MOTIVATION

One summer in 1956, on the campus of Dartmouth College, an interdisciplinary seminar was being held where scientists and researchers put their heads together to conceptualize theories about intelligent machines with an ability to think. That was the birthplace of Artificial Intelligence (AI). Artificial Intelligence started gaining momentum. Researchers from universities around the world were optimistic about this new emerging field and started advocating AI as a technology that could change the world. Researchers started getting a lot of funding for AI research as the field started flourishing. Research labs for AI started popping up all around the world. One of them was MIT AI Lab co-founded by Marvin Minsky in 1966. Minsky received a research grant to hire an undergraduate student, Gerald Sussman, to connect a camera to a computer and to make the computer interpret what it saw through the camera. Hence, Computer Vision essentially originated as a summer project.

Researchers had high expectations and were overtly optimistic about AI. In 1970, Minsky told Life Magazine, "In from three to eight years we will have a machine with the general intelligence of an average human being" (Minsky, 1970, Life Magazine). He was convinced that within a generation, the general problem of Artificial Intelligence will be solved and that there will be almost no differences between the intellect of an intelligent machine and that of a human being (Minsky, 1967). But none of those dreams came to fruition. By the end of 1970, governments and corporations who were granting funds for AI research started losing faith in AI themselves. The results achieved by the

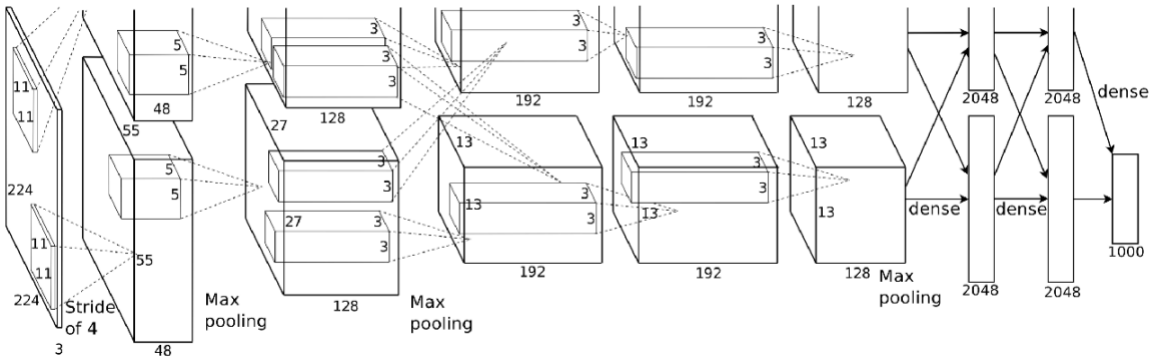
researchers fell short of their lofty ambitions, and soon the funding dried up. What followed next was known as the “AI Winter”. This era of AI frenzy failed to reach its immense and also unreasonable potential.

However, during the same time, two neuroscientists conducted some experiments on visual cortical neurons and published a paper (Hubel & Wiesel, 1959) which later turned out to be one of the most influential papers in Computer Vision. They ran an experiment where they inserted electrodes into the visual cortex area of a cat’s brain and tried to study the activity of the neurons as they showed the cat shapes and objects through a projector. Accidentally, in the middle of a session of their experiment, as they were changing the slide of the projector, they noticed unusual neuron activity. The duo examined it and realized that the neuron fired when the cat noticed the line formed by the sharp edge of the glass slide move. The researchers inferred that visual processing starts with processing simpler structures like the outline of an object. That is the core principle behind Computer Vision, but that was not realized until 2012 when AlexNet (Krizhevsky et al., 2012) won the ImageNet (Deng et al., 2009) competition, ILSVRC. AlexNet was one of the most influential papers since the advent of convolutional neural networks (ConvNets) (LeCun et al., 1998) to have leveraged the potential of ConvNets as shown in figure 1.1. Since then, the error rates in image classification in the ImageNet challenge fell down to a few percent and all the winners since then have been convolutional neural networks. AlexNet opened the door to solve image classification problems that were not possible earlier.

With evolving convolutional neural networks, came the necessity of vast amount of image data. Thanks to Moore’s Law, algorithms that were theories just a few years ago

Figure 1.1

Architecture of AlexNet with 5 Convolutional Layers (Krizhevsky et al., 2012)

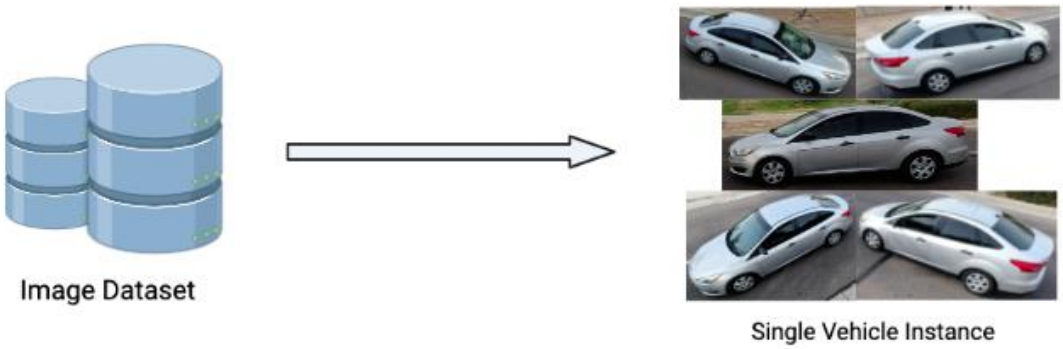


now have practical applications because of faster computers. But, to realize the full potential of these models or algorithms, we need to train these models on the vast amount of data. Owing to the exponential growth in the number of images on the internet, researchers soon leveraged that by building large-scale datasets. In 2010 ImageNet (Deng et al., 2009) was released, which was the first of its kind large-scale image dataset. In the subsequent years, other researchers followed suit by generating other large-scale image datasets.

To date, there has been no open-source, large-scale, multi-view vehicle datasets. Hence, in this thesis, we present a novel dataset of roadside images of vehicles captured from 5 different viewpoints (figure 1.2) as one of the main contributions of this work. Although this data is not large-scale because of the lack of resources, this work will also present an ingenious approach to autonomously curate such a dataset with a similar camera system. Building a dataset – let alone a large-scale one – requires a significant amount of man hours. We collected approximately 5 hours of videos from 5 different

Figure 1.2

Dataset of Roadside Images of Vehicles from 5 Different Viewpoints



viewpoints, from which an image dataset of vehicles across the 5 different viewpoints was curated using an automated algorithm that incorporated an ingenious but simple method of calculating optical flow vectors, tracking bounding boxes, and feature matching in a single viewpoint and multiple viewpoints across the frames. To generate bounding boxes and segmentation masks, a Mask R-CNN (He et al., 2017) was used. Finally, to put this dataset to work, we attempted to solve the vehicle tracking problem by building an image classifier to perform vehicle re-identification.

In this work, we are solving the problem of re-identifying objects, vehicles specifically, across multiple viewpoints. With the recent inflation in cameras connected to computers or pocket computers i.e., smartphones, a new problem has emerged, which is re-identifying an object across multiple cameras or devices. Uses of cameras in security systems have been prevalent for a long time, and more recently are paired up with intelligent and sometimes autonomous systems; may it be for localizing an object in a scene or detecting the motion of an object. There has always been a need for tagging

objects across multiple viewpoints in security systems; for tracking vehicles across multiple cameras in a traffic monitoring system, or even humans in a smaller environment. Thus, we attempted to solve the problem of tracking vehicles across multiple viewpoints by approaching the problem through vehicle re-identification. Our plan was to collect a dataset of vehicles across multiple viewpoints and then to train a deep neural network to re-identify a vehicle. We trained a ResNet (He et al., 2016) to re-identify a vehicle from the dataset it was trained on. For the train-test split, we randomly selected four viewpoints to train the model on and then the other viewpoint was used for evaluation or re-identification.

Generating a Dataset

Vision data is one of the most widely used forms of data. Vision data is deep-rooted in various industries; ranging from fashion to medical, e-commerce, and social media to name a few. Through the boom of social media, data in the form of images have exploded on the internet. With the immense access to image data throughout the internet, researchers and corporations quickly realized that they were sitting on a goldmine of data and how valuable this data is. Thus, with the growth of image data on the internet, started the rise of large-scale image datasets.

Although there are now multiple large-scale image datasets – some of the datasets as large as containing over 9 million images, such as ImageNet – there has been a lack of image datasets of vehicles captured from multiple viewpoints. That is one such research problem that this work strives to solve. This work is introducing a novel multi-view vehicle image dataset, called MVVdb. However, since this proposed dataset is not a

large-scale dataset because of a lack of resources, the subsequent chapters also outline an ingenious method of generating and curating such a dataset if used a similar camera system to the one used for this proposed dataset.

Image Classification

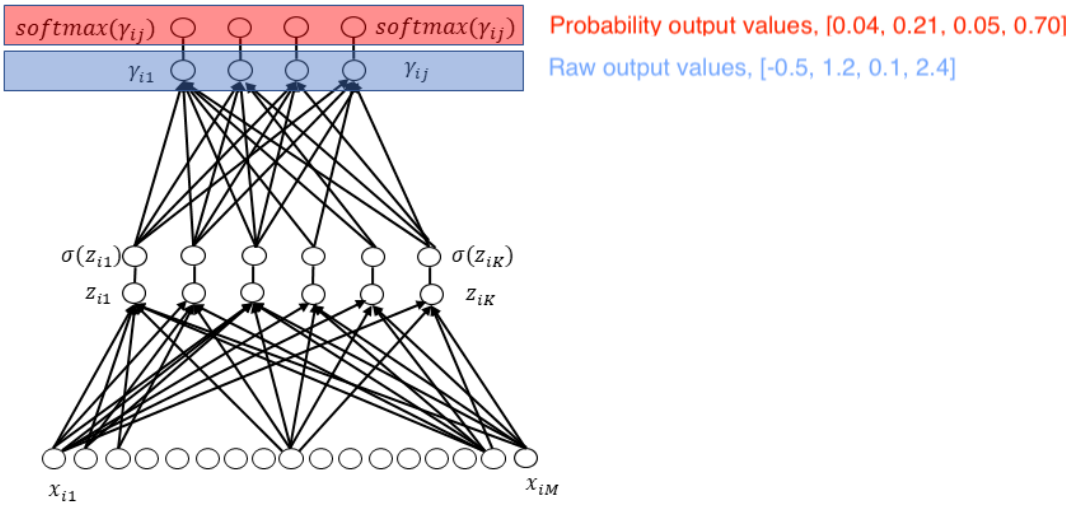
Image Classification, sometimes referred to as Image Recognition, is the process of analyzing the features of a raster image and then associating that image with a class or label. In general, there are two different types of classifications, namely (1) single-label classifications and (2) multi-label classifications.

Single-label classifications are the most common type of classifications where an instance is associated with a single label. The purpose of a single-label classifier is to predict a single label or class for an instance. A single-label classification problem can be further categorized into (1) binary classification, where a classifier only classifies an instance between two classes, and (2) multi-class classification, where there are multiple classes in the dataset. Although multi-class classification and multi-label classification sound similar, however, they are not and are inherently different. Multi-class classification is, as stated earlier, a type of single-label classification where there are multiple classes but only one possible answer. The classes in a multi-class classification problem are mutually exclusive and hence, it uses a Softmax activation function (figure 1.3). On the other hand, multi-label classification problems are non-mutually exclusive and use a Sigmoid activation function.

The output of a single-label classifier is a vector of the same length as the number of labels of the dataset, each value of the vector denoting a raw output score regarding if

Figure 1.3

Visual Representation of a Softmax Activation Function



Note. A Softmax activation is only used as the final layer after the last fully connected layer to transform the similarity scores to probability scores.

the instance belongs to that class. A single-label classifier utilizes a Softmax activation function which transforms the output values of the last fully connected layer to probabilities as shown in the formula in figure 1.4. Our work on vehicle re-identification is also a multi-class problem, each vehicle being a separate class.

Research Value

There is a significant lack of vehicle image dataset from multi-view cameras. Lots of existing Computer Vision problems will heavily benefit from such a multi-view vehicle dataset e.g., image to 3D model generation, or 2D image to 3D shape prediction. The multi-view aspect of the dataset can also be disregarded and can be used for vehicle

Figure 1.4

General Formula of a Softmax Function

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j} = 1,\ldots,\text{K}$$

Note. z stands for the output while k is the number of classes.

make and model recognition – if properly labelled. Creating a large-scale dataset requires significant manual input and a serious amount of man hours. This work proposes the utilization of simple techniques like optical overflow, feature tracking, and vehicle localization using bounding boxes to autonomously generate a large-scale dataset consisting of vehicle images captured from surveillance cameras.

For years camera systems have been used for the purpose of security. The motor vehicle department has been using cameras for decades to keep the roads safe. Most city intersections across the US and random speed traps on the state or interstate highways use cameras to monitor them. They surveil vehicles on the road so that no infringement or violation of the law goes unpunished. They monitor actions like jumping red lights, overspeeding, and even hit-and-runs. Initially, all these systems were operated manually. However, as computers started becoming intelligent, the desire to automate tedious manual processes took over and a lot of these processes were automated. Nowadays, the traffic lights are automated, and the motor vehicle department has even adopted to use autonomous systems to recognize license plates of vehicles using cameras and generate

tickets if any law is broken, like overspeeding, jumping red lights, or not stopping at the stop sign. Artificial Intelligence or Computer Vision has been so successful, that there is now a massive network of cameras sprawling across every metropolis. With such a massive network of camera systems, there is a new problem to track these vehicles across multiple devices or viewpoints.

Contributions

This work presents a series of significant contributions in the field of Computer Vision:

Multi-view Dataset

One of the main contributions of this work is a novel dataset consisting of roadside images of vehicles captured from multiple viewpoints -- 5 different viewpoints to be exact -- from all around a vehicle, so that every bit of the body surface of a vehicle can be visualized. This is especially important if one wants to use the dataset to generate 3D models of a vehicle along with reconstructing its texture. Other than that, it is also a robust dataset for research along the lines of 2D image to 3D shape prediction or even fine-grained vehicle make and model recognition.

Guide to Curate a Structured Dataset from Similar Videos

Although a dataset is part of the contributions made by this work, we could not build a large-scale dataset because of the lack of resources. So, this thesis also outlines a guide on how to autonomously generate – even a large-scale dataset if needed – from recorded roadside videos with a similar camera system to the one used in this work.

Vehicle Re-identification

To display the functionality of our new novel dataset, we put the dataset to use to tackle the problem of Vehicle Re-identification through Image Retrieval. We showcase how this dataset stacks up when using an image classifier like ResNet to train itself and then perform image retrieval. This work essentially also addresses the fine-grained make and model recognition problem as vehicle re-identification is fundamentally fine-grained make and model recognition if the dataset is labelled.

Structure of Thesis

This work will first perform a background study and review some of the existing large-scale vehicle datasets. Next, we will review some of the past work on image retrieval systems or image re-identification systems. We will also do a related work study on make and model classification of vehicles because it is nothing but an image retrieval problem with the vehicle's make and model labels on the dataset. Next, we will go over the notable state-of-the-art models in image classification as that is the route we chose to tackle the image re-identification problem. Furthermore, this thesis will cover the intuitive approach taken to curate a structured dataset from unedited roadside videos of vehicles. Following, there will be a detailed account of the classification task to achieve re-identification of the vehicles of the given dataset. Lastly, we will go over the results of our research and compare it with other comparable works. Finally, some concluding remarks and areas of future work will be detailed.

CHAPTER 2

BACKGROUND AND RELATED WORK

Having briefly covered the foundation for generating a dataset and image classification, in this section we will go through a background study of the past and the existing state-of-the-art large-scale image datasets and deep neural networks for image classification. Additionally, we will look at large-scale image datasets and image classifiers from the context of vehicle re-identification –i.e., vehicle or car datasets designed specifically for vehicle re-identification or fine-grained make and model classification, and neural network models catered towards the same respectively.

Large-Scale Image Datasets

With an exponential growth in camera systems or devices with cameras in recent times, there has been an explosion in the number of images on the internet. The development of social media has further amplified that because of the number of images being uploaded to social media on a daily basis. On average, more than 300 million images are uploaded daily to the internet. Soon researchers and corporations recognized the significance of all this data and realized the gold mine of data they were sitting on.

In 2010, ImageNet (Deng et al., 2009), the first large-scale image dataset for object detection and recognition was released. The dataset was a one of its kind and the first of its kind as it contained more than 9 million annotated images with even hand-annotated bounding boxes for some of the instances. Since then, the dataset has been

updated in 2014 and now comprises of more than 14 million images across over 20,000 categories or classes.

Along with ImageNet came CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009) which are nothing but labelled subsets of the 80 million tiny images dataset (Torralba et al., 2008). Both CIFAR-10 and CIFAR-100 datasets contain 60,000 labelled images, the only difference being is that CIFAR-10 consists of 10 classes with 6000 images per class and the CIFAR-100 dataset consists of 100 classes – only 600 images per class. Since the instances per class for the CIFAR-10 dataset is lesser than that of ImageNet, models trained on the CIFAR-10 dataset perform worse than that same model trained on the ImageNet dataset with a higher top-5 percentage error. Furthermore, the CIFAR-100 dataset performs even worse than the CIFAR-10 dataset as it has even lesser images per class. However, the classes for both CIFAR-10 and CIFAR-100 are mutually exclusive while ImageNet is not. CIFAR-10 and CIFAR-100 are also low-resolution, 32x32, so allows a researcher to quickly train and test their models on this dataset.

Next came PASCAL VOC (Everingham et al., 2010) which provided an image dataset for object classification. The dataset is annotated with around 500,000 images across 20 object classes. Through KITTI Vision Benchmark Dataset (Geiger et al., 2013), we got a first of its kind large-scale video dataset captured from the point of view of a vehicle driving around a mid-size city. The dataset includes data from two high-resolution color and grayscale video cameras. For accurate ground truth, the dataset also includes data from a laser scanner and a GPS localization system.

Microsoft Common Objects in Context (COCO) (Lin et al., 2014) is a large-scale dataset for object detection, segmentation, and even captioning. The dataset consists of

80 object categories and 91 stuff categories. In this dataset, stuff categories are defined as amorphous backgrounds e.g., grass, sky. The dataset includes over 1.5 million instances with superpixel segmentation, object or stuff labels, and captioning of the image with context. Since then, there have been plenty of large-scale datasets that have come out but for more targeted problems. In this work, the main subject of our problem statement are vehicles, so in the next section, we are going to take a look at car datasets.

Car Datasets in Computer Vision

There was not any popular public large-scale solely dedicated for vehicles before the Car-197 (Krause et al., 2013) dataset came out. It consists of 197 classes with a total of 16,185 images. The classes are in the fine-grained level –i.e., make, model, and year. The images are collected from the internet using a web crawler, and then data deduplication methods were implemented to acquire a subset of distinct images. Finally, they used Amazon Mechanical Turk (AMT) to determine if the images belong to their respective classes. One disadvantage of such a dataset is that since all the images are obtained from the internet, there is a possibility that a model trained on it can be biased and may not work in a practical scenario. Practically, cameras are installed at high angles to monitor vehicles, an angle from where images are not usually found on the internet – most images on the internet are from the eye level.

Since computer vision tasks regarding cars or vehicles were still highly neglected by the computer vision research community, the Comprehensive Cars (CompCars) dataset was released by the authors (Yang et al., 2015). The dataset includes both images from the internet and that of a surveillance nature. The internet images dataset consists of

163 car makes and 1,716 car models and the surveillance images dataset consists of 50,000 images from the front view. All the images in the dataset do not present a full-body picture, some of the images are just of car parts. However, if a full car is visible in an image, that image is annotated with a bounding box and orientation. The CompCars dataset is structured into a tree structure where the hierarchy consists of three layers –i.e., car make, car model, and year of manufacture. Apart from the images, each car model is also annotated with five attributes –i.e., maximum speed, displacement, number of doors, number of seats, and car type (twelve different kinds of car types are defined e.g., sedan, hatchback). Although the dataset including data for orientation of the vehicles was a novelty, however, the quantity distribution is not balanced across the viewpoints and the different car models.

Another large-scale vehicle make and model dataset that we should talk about is the VMMRdb (Tafazzoli et al., 2017) dataset. The VMMRdb dataset is significantly larger than other car datasets. The dataset includes 9,170 classes consisting of 291,752 images. The vehicles are annotated with their make and model along with their production year. Just like the Car-197 dataset, the authors used a web crawler to gather images and relevant information –i.e., the title and description to automatically annotate images -- from Craigslist and Amazon.

Due to the lack of high-quality and large-scale vehicle re-identification datasets, the VehicleID dataset was released by the authors (Liu et al., 2016). Till now we saw datasets that served more towards the recognition of car attributes (e.g., make, model, and color). This paper presents a dataset that caters more toward vehicle re-identification. The authors used a surveillance camera system to collect the dataset. The data is collected

through a network of multiple real-world surveillance cameras during daytime amassing over 200,000 images of over 26,000 vehicles. Additionally, the authors manually annotated 90,196 images across 10,319 vehicles with their ground truth values i.e., their make and their model.

There has been another large-scale dataset for vehicle re-identification that strives to push towards the research of challenging problems of vehicle re-identification. The dataset VERI-Wild (Lou et al., 2019) was released to address the problem of not having enough real-world scenarios in vehicle re-identification research. The dataset wanted to include high viewpoint variations like how surveillance cameras are installed in the real world, varying lighting conditions, and complex and changing backgrounds. To achieve their goal, the authors captured their data from a large surveillance camera network consisting of 174 cameras across a large urban district. The camera recorded data for 24 hours a day -- to account for the diverse illuminating conditions – and did that for a month. That resulted in a vehicle image dataset consisting of over 400,000 images and over 40,000 vehicle IDs. The authors used a YOLO-V2 (Redmon et al., 2017) to detect and draw bounding boxes around a vehicle. The VERI-Wild dataset is the first of its kind dataset where the data is collected from unconstrained conditions –i.e., the wild, hence, the name of the dataset.

Deep Neural Image Classifiers

Before Convolutional Neural Networks (CNN), researchers used to connect each pixel of an image to a simple raw multi-layer perceptron. This method was highly inefficient as pixels are spatially correlated. Then CNNs were introduced when LeNet

(LeCun et al., 1998) came out. The idea of the paper was to architect an algorithm that can classify high-dimensional patterns – primarily for OCR and character recognition in documents -- and to do that we need to first extract relevant low-dimensional features. Hence, convolutional neural networks were born. The authors of LeNet feed an image to a network and perform convolutions given a specific filter. The output is then passed through an activation function, following which the output repeats the same process and goes through a sequence of convolution, activation, and pooling. At the end, we are only left with high-level relevant features which is then flattened and fed to two repeating fully connected layers. Thus, "the first CNN" was built.

Although CNN was first designed in 1998 (LeCun et al., 1998), it was not fully utilized until AlexNet (Krizhevsky et al., 2012). AlexNet applied the same concept of LeNet of using successive convolutions to capture more and more subtle features. AlexNet adapted to use a ReLU activation function for better gradient propagation, whereas the original LeNet used a TanH activation function. The AlexNet paper also introduced data augmentation –i.e., randomly augmenting an input image like rotating, translating, or cropping – to build a more robust image classifier model. AlexNet was the state-of-the-art model when it came out, in 2012, with a 15.4% top-5 error on the ImageNet (Deng et al., 2009) competition.

The next breakthrough in image classification came when the authors of VGGNet (Simonyan et al., 2014) came out pushing the depth of a network to 16-19 layers. During its time of inception, 16 or 19 layers were considered very deep. However, through VGGNet the authors inferred that increasing the depth of a network with smaller convolution filters performs significantly better and achieves a better hierarchical

representation of visual data. The network keeps it simple by using exclusively only 3x3 convolution filters stacked on top on each other when needed (as shown in figure 2.1) which (1) promotes more non-linearity as three 3x3 convolution layer is better than a single 7x7 layer, (2) decreases the number of parameters, which is a good thing given the increase in the depth of the layers. However, the authors found it difficult to converge a very deep network and hence, VGGNet is painfully slow to train.

Till now all the models that we saw, they all followed the same trend –i.e., to go deeper. But at some point, researchers had to realize than just going deeper is not the answer. Going deeper and deeper may not increase performance after a certain point and may start doing actually the opposite because of the vanishing gradient problem. If there are too many layers using a certain activation function, the gradient diminishes very fast, and the model stops learning effectively as shown in figure 2.2. To counter this problem researchers came out with the ResNet (He et al., 2016) where the authors introduced residual learning. Although ResNet is much deeper than VGGNet, ResNets are much easier to train. To this day, it is one of the best performing models on the ImageNet with a 3.57% error.

Vehicle Re-identification or Fine-grained Image Classification

The authors who came out with the CompCars (Yang et al., 2015) dataset perform both fine-grained image classification and vehicle re-identification in their work to exhibit the functionality of their dataset. The authors train a CNN model to compare the classification performance in specific viewpoints and all viewpoints. Experiments showed that the CNN model that learned from all the viewpoints performed better than

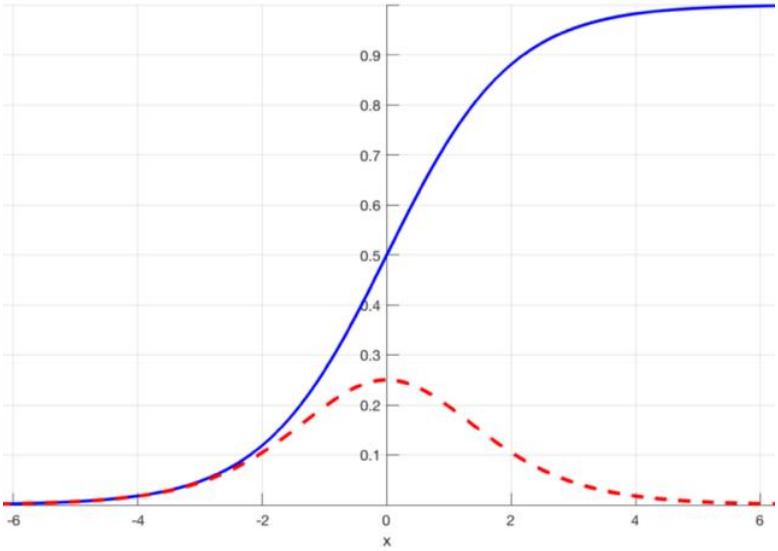
Figure 2.1

Architecture of VGG as Shown in Very Deep Convolutional Networks for Large-Scale Image Recognition (Simonyan et al., 2014)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.2

Graph Showing the Vanishing Gradient Problem



Note. The blue curve represents the Sigmoid function while the red dashed curve represents the derivative of the Sigmoid function.

the ones trained on specific viewpoints. This exhibits that the CNN model was able to learn from the different viewpoints. For the vehicle re-identification task, the authors tweaked the classifier used for the fine-grained classification to behave like a feature extractor and then applied a Joint Bayesian (Chen et al., 2012) to train a model. The authors also adopted the CNN feature extractor, coupled it with a SVM, and then did a comparative study.

The authors of the VMMRdb (Tafazzoli et al., 2017) dataset trained a CNN model to perform fine-grained make and model recognition and compared it with the same model trained on CompCars. The authors wanted to test the models with testing images

front challenging sources to simulate real-world behavior. Although CompCars performed significantly better on its own testing dataset, however when tested on a surveillance image from the VMMRdb dataset, the performance fell drastically with an accuracy of 50.05%. On the contrary, the CNN trained on the VMMRdb performed slightly better when tested on a non-VMMRdb image with an accuracy of 52.85%.

VehicleID (Liu et al., 2016) dataset was utilized to perform vehicle re-identification by the authors and they trained a GoogleNet (Szegedy et al., 2015) on the Vehicle ID dataset along with the CompCars dataset to perform a comparative study. The GoogleNet trained on the VehicleID dataset performed around 3% better than the one performed on the CompCars dataset. On the other hand, the authors of VERI-Wild (Lou et al., 2019) dataset proposes a Feature Distance Adversarial Network (FDA-Net) to perform vehicle re-identification and performs better than GoogleNet trained on the same dataset with an accuracy of 70.84%.

CHAPTER 3

MULTI-VIEW VEHICLE DATASET

Having covered the contributions of previous vehicle datasets and the thought process and motivation behind their existence, we will discuss the Multi-View Vehicle Dataset (MVVdb) that we propose in this work. In our review of previous literature, we have seen that there are generally three types of vehicle datasets, (1) datasets where the vehicle images are collected from the internet, (2) datasets where vehicle images are taken from surveillance cameras or camera systems simulating a surveillance network, (3) a combination of both images from the internet and surveillance images. In this section, we will describe our system of generating a dataset from videos collected from multiple viewpoints.

The main motivating factor behind the development of this system are:

- to design a multi-view dataset for 3D Computer Vision problems
- to design a dataset with surveillance type images of vehicles
- to create a blueprint of a method to generate a large-scale dataset without significant manual input

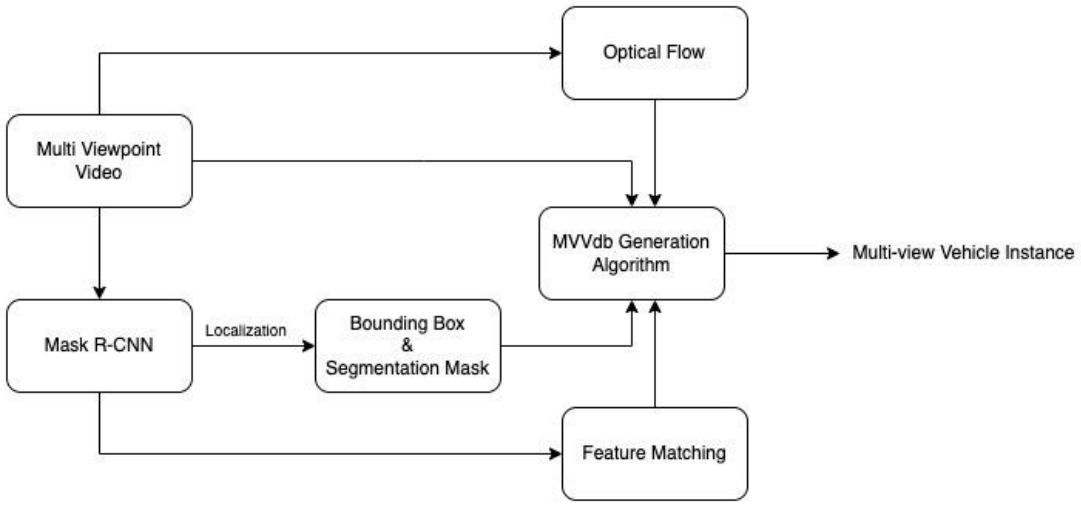
This chapter will detail the development and the path taken to design this system and the dataset. We will go through the core components of this system and explain the reason and motivation behind the decisions taken. Evaluation of this dataset and vehicle re-identification performed on this dataset will be left for the subsequent chapters.

General System Architecture

One of the main contributions of this work is creating a novel dataset of roadside images of vehicles from multiple cameras, simulating a surveillance camera system by attaching the cameras on top of tall camera tripods. To capture a single instance of a vehicle from multiple angles, the cameras need to be synced together. Only two ways to sync video streams from multiple cameras are (1) to start the recording on all cameras together, and (2) somehow sync the video streams post-production using an audio or visual cue. Since we did not have the resources to use cameras that can be synced during the recording of the data, we went for the latter option to sync the video streams in post-production. Next, we run a region-based convolutional neural network – we opt for Mask R-CNN (He et al., 2017) because of the need to generate segmentation masks -- on our dataset of videos to perform object localization and detection. Using the bounding boxes outputted by the Mask R-CNN, we track and validate if an instance of a vehicle is actually the same vehicle or not in two consecutive frames. Additionally, we track the motion of a vehicle across frames using optical flow vectors to validate the deduction made from the bounding boxes. Since our dataset is a multi-view dataset, we must not only track and validate that an instance of a vehicle is the same vehicle across multiple frames, but we have to also authenticate that an instance of a vehicle is seen from two different viewpoints are the same vehicle. To do that, we use both feature point matching and bounding box localization. If everything is validated, we save the images from multiple viewpoints to that particular instance in the dataset. Figure 3.1 provides a flowchart and a high-level overview of this whole process.

Figure 3.1

Flowchart Generalizing the Dataset Generation Method



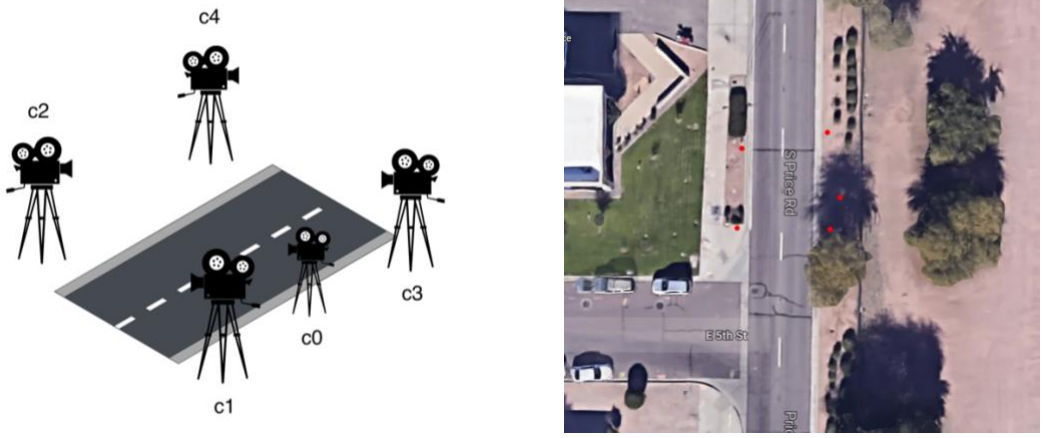
Characteristics of the Videos

The dataset proposed by us in this work being a multi-view dataset, we record the videos for the dataset generation from 5 different cameras – 4 cameras on tall tripods at four corners and a single camera on a shorter tripod at the side to get a side profile of the vehicles – as shown in figure 3.2 (a). The cameras are used to record only one particular section of the road as shown in figure 3.2 (b). All cameras used to record the roadside videos are GoPros, all recording at 4k resolution. The four cameras in the corners are attached to tall tripods to replicate a surveillance camera network. The fifth camera on the side is placed on a shorter tripod at eye level because its purpose is to capture the side profile and register as many distinct features on the vehicle as possible.

The road we choose to record is a two-lane, one-direction road. The videos are all recorded in Arizona which is sunny and bright most of the year. If there is glare or

Figure 3.2

Visual Representation of the Location of the 5 Cameras



(a) (b)

Note. In this figure, (a) shows a graphical representation of the position of the cameras with respect to the road, and (b) shows the exact location of the cameras on the world map calculated using Perspective-n-Point.

reflection from the vehicles because of the sunshine, the cameras will miss out on important features that can be used by a model to distinguish between two car models. To prevent such a circumstance, we made the decision to record the videos early morning, when it is not too bright outside. We also chose a section of the road which has significant shade from the sun because of trees. However, choosing such a portion of the road resulted in a merging side road being in the direct field of view of our cameras as shown in figure 3.3. This is especially problematic, because if a vehicle merges from the side road to the main road, then it means that the vehicle is only visible from two of our cameras, and hence, we do not want that vehicle in our dataset. We take care of such a

Figure 3.3

Viewpoints c0 and c3 Showing a Merging Side Road

(a)

(b)

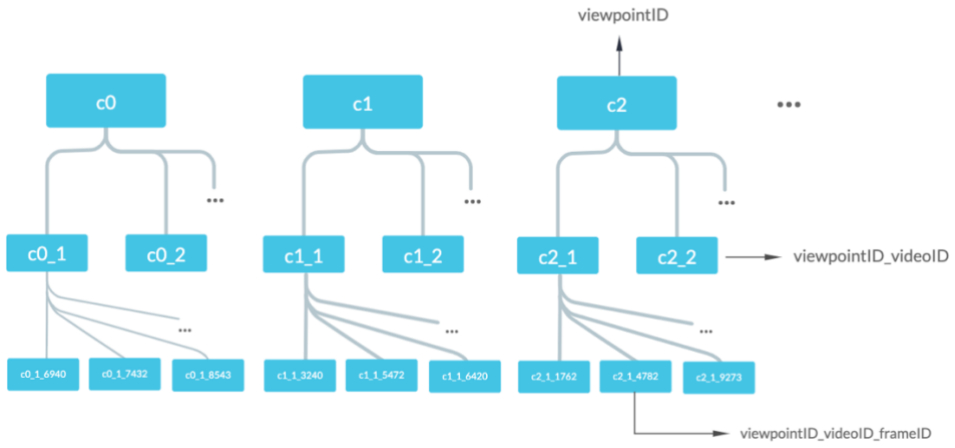
Note. Figure (a) showing the side road from the viewpoint c0, (b) showing the side road from c3.

scenario using an ingenious method and we will be discussing it in the following sections.

After the videos are recorded, we take the data from each camera and put them in separate folders dedicated for separate viewpoints. We name the folders after the viewpoints, viz., *c0*, *c1*, *c2*, *c3*, *c4*, and *c5*. We will call this the viewpoint ID. We recorded around 6 hours of video from each camera and the videos are broken up into 63 different videos of approximately 6 mins each by the GoPro. The videos are indexed – starting from 0 – and named with the viewpoint it was captured from, followed by an underscore and the index (e.g., a video indexed 6th from the viewpoint c1 is named *c1_6*). We will call this the video ID. An image taken from a video will be named after the viewpoint ID, video ID, and the frame number (frame ID) – all separate by underscores. That will be called the vehicle's vehicle ID. The naming structure is shown in figure 3.4.

Figure 3.4

Showing the Naming Hierarchy of the Files



Since we could not use cameras which were self-synced, we use a visual cue to sync the video streams in post-production. We use a torch and flash its light so that it is visible from all cameras simultaneously. Then we write a program to run the separate videos frame by frame using OpenCV and note down the frame numbers on which the flashlight was turned on in each viewpoint. We consider the viewpoint *c0* to be the reference viewpoint because if a vehicle is visible from viewpoint *c0,* then it is visible from all the other viewpoints. Using c0 as the reference, we write a program to calculate what the video ID and the frame ID is – of frame 0 of a given video in viewpoint ID *c0* – in the other viewpoints. For example, frame ID 0 of video ID *c0_5* is frame ID 7054 of video ID *c1_1* in viewpoint *c1,* frame ID 4910 of video ID *c2_0* in viewpoint *c2,* frame ID 6895 of video ID *c3_3* in viewpoint *c3,* and frame ID 8760 of video ID *c4_2* in viewpoint *c4* (as shown in figure 3.5).

Mask R-CNN and its Results

To perform object localization of the vehicles in the dataset of videos, we run a Mask R-CNN to predict the location of a detected object and segmentation results through bounding boxes and segmentation masks. The model also outputs class labels – based on the dataset the pre-trained Mask R-CNN was trained on – along with its confidence scores. We load the weight of a pre-trained Mask R-CNN trained on the MS-COCO (Lin et al., 2014) dataset. MS-COCO has 81 classes where the class for cars is 3 and the class for trucks is 8. Hence, if Mask R-CNN detects a car or a truck, it outputs 3 or 8 respectively, its confidence score of that object belonging to that class, bounding box

Figure 3.5

Figure Showing the Frame Correspondence List

Ref	c1	c2	c3	c4
('c0_4', 6984)	('c1_1', 4240)	('c2_0', 2096)	('c3_3', 4081)	('c4_2', 5946)
('c0_5', 0)	('c1_1', 7054)	('c2_0', 4910)	('c3_3', 6895)	('c4_2', 8760)
('c0_6', 0)	('c1_2', 7053)	('c2_1', 4916)	('c3_4', 6895)	('c4_3', 8760)
('c0_7', 0)	('c1_3', 7053)	('c2_2', 4914)	('c3_5', 6895)	('c4_4', 8760)
('c0_8', 0)	('c1_4', 7055)	('c2_3', 4915)	('c3_6', 6896)	('c4_5', 8761)
('c0_9', 0)	('c1_5', 7054)	('c2_4', 4917)	('c3_7', 6896)	('c4_6', 8760)
('c0_10', 0)	('c1_6', 7054)	('c2_5', 4914)	('c3_8', 6896)	('c4_7', 8762)
('c0_11', 0)	('c1_7', 7054)	('c2_6', 4913)	('c3_9', 6896)	('c4_8', 8761)
('c0_12', 0)	('c1_8', 7054)	('c2_7', 4916)	('c3_10', 6895)	('c4_9', 8761)
('c0_13', 0)	('c1_9', 7054)	('c2_8', 4916)	('c3_11', 6895)	('c4_10', 8761)
('c0_14', 0)	('c1_10', 7054)	('c2_9', 4915)	('c3_12', 6895)	('c4_11', 8759)
('c0_15', 0)	('c1_11', 7055)	('c2_10', 4915)	('c3_13', 6896)	('c4_12', 8761)
('c0_16', 0)	('c1_12', 7055)	('c2_11', 4914)	('c3_14', 6896)	('c4_13', 8761)
('c0_17', 0)	('c1_13', 7055)	('c2_12', 4916)	('c3_15', 6896)	('c4_14', 8761)
('c0_18', 0)	('c1_14', 7057)	('c2_13', 4916)	('c3_16', 6897)	('c4_15', 8762)
('c0_19', 0)	('c1_15', 7056)	('c2_14', 4916)	('c3_17', 6897)	('c4_16', 8762)
('c0_20', 0)	('c1_16', 7055)	('c2_15', 4915)	('c3_18', 6896)	('c4_17', 8762)
('c0_21', 0)	('c1_17', 7054)	('c2_16', 4917)	('c3_19', 6896)	('c4_18', 8762)
('c0_22', 0)	('c1_18', 7054)	('c2_17', 4915)	('c3_20', 6895)	('c4_19', 8760)
('c0_23', 0)	('c1_19', 7055)	('c2_18', 4918)	('c3_21', 6896)	('c4_20', 8762)
('c0_24', 0)	('c1_20', 7055)	('c2_19', 4917)	('c3_22', 6896)	('c4_21', 8762)
('c0_25', 0)	('c1_21', 7054)	('c2_20', 4917)	('c3_23', 6896)	('c4_22', 8762)
('c0_26', 0)	('c1_22', 7055)	('c2_21', 4918)	('c3_24', 6897)	('c4_23', 8763)
('c0_27', 0)	('c1_23', 7054)	('c2_22', 4917)	('c3_25', 6897)	('c4_24', 8762)
('c0_28', 0)	('c1_24', 7054)	('c2_23', 4916)	('c3_26', 6897)	('c4_25', 8762)

for localizing the object, and segmentation mask. Before running Mask R-CNN, the resolution of the videos is brought down from 4k resolution to 1920x1080. This is because running Mask R-CNN on data with a 4k resolution will be too computationally expensive. All the outputs are saved in separate ".npy" files at the frame level, e.g., for frame 4562 of video 6 of viewpoint c0, there are separate ".npy" files are bounding boxes (*c0_6_boxes_4562.npy*), segmentation masks (*c0_6_masks_4562.npy*), class labels (*c0_6_classes_4562.npy*), and confidence scores (*c0_6_scores_4562.npy*). Saving the outputs in separate files was one of the mistakes that we did. Each video is around 6 mins long, has approximately 10,000 frames. If there are four outputs in separate files for each frame then that is 10,000 times 4, 40,000 files for one video. And we have 63 videos – that is a lot of files. The whole dataset of Mask R-CNN outputs amounted to around 500 gigabytes. We used a 5 TB Western Digital Elements external hard disk to save all the data. With a read speed of only 100 MB/s, it created a bottleneck while accessing the data for generating the multi-view dataset. Along with that, the fragmentation of data caused because of over 1 million files made the data access painfully slow.

Vehicle Tracking Across Multiple Frames

To generate the Multi-View Vehicle dataset, we need to group all the images of one instance of a vehicle and put them under a single record or folder. For that to be possible, we need to track the vehicle across frames and multiple viewpoints. In this section, we will only discuss our method to track a vehicle across frames from a single viewpoint. In the next section, we will discuss how to do the same across multiple viewpoints.

We implement an ingenious technique to validate if two detected vehicles in two consecutive frames belong to the same instance or not. We apply optical flow to track the motion of the vehicle and check bounding box overlapping to infer if vehicles across frames are the same vehicle or not.

Optical Flow

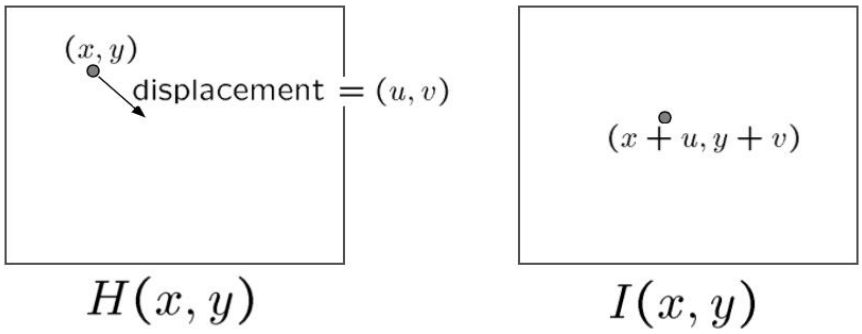
If there are two images of an object, optical flow is the apparent motion of the object between the two images. Optical flow is represented using a 2D vector field where each vector is a displacement vector depicting the motion of a pixel or a feature between two images. As illustrated in figure 3.6, if there is a pixel in image H at (x, y) and the same pixel in image I is at (x+u, y+v), then the optical flow is (u, v).

For many years researchers have been trying to solve the equation for u and v and the Lucas-Kanade method is the most popular one. In this work we are using the Lucas-Kanade method of optical flow. The Lucas-Kanade method assumes that the neighboring pixels around a feature point will have the same motion. If there is a feature point on an image, the Lucas-Kanade method takes a 3x3 grid around the feature point and assumes that the 9 points will have the same motion.

We use OpenCV's calcOpticalFlowPyrLK() to calculate the optical flow of a couple of feature points of a vehicle. We use OpenCV's goodFeaturesToTrack() – which finds the n strongest feature points in an image using the Shi-Tomasi corner point detector – to detect good features for tracking. Since most of the area of frame is

Figure 3.6

Optical Flow of a Pixel Between Two Images



mostly a constant background, we do not want to extract features in these areas to track for optical flow – we are only interested to track feature points on a vehicle. Hence, we us the bounding boxes around the vehicles predicted by the Mask R-CNN and use those bounding boxes as region of interest. For a given frame, we get the bounding box of that frame, add a padding around the box, and then use that area as our region of interest to extract feature points using the Shi-Tomasi corner point detector. We need to add padding around a bounding box because not all the predicted bounding boxes perfectly encapsulate the vehicles, some bounding boxes cut through the front part or the rear part of the vehicles. We use the region of interest to crop out the vehicle from the image, extract feature points using the Shi-Tomasi corner point detector and add the offset to the feature point coordinates to translate those points to the whole image (as shown in figure 3.7).

We get the first frame, detect feature points using the Shi-Tomasi corner detector, run a loop to get consecutive frames, detect their feature points, and calculate the optical

Figure 3.7

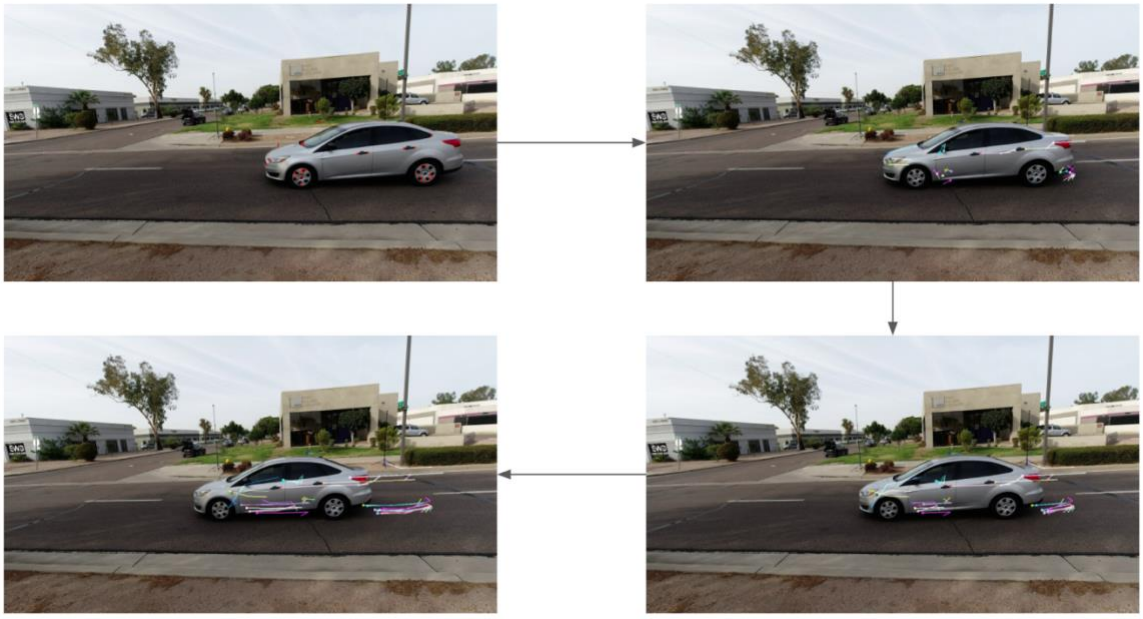
Flowchart of Shi-Tomasi Corner Detector in ROI

flow of the feature points between the two frames. We first attempt to calculate the optical flow by extracting features from the vehicle image, but the algorithm has a hard time to track the feature across frames (as shown in figure 3.8). Alternatively, we performed optical flow on the segmentation masks which were much easier to track across frames as shown in figure 3.9. Although the Shi-Tomasi corner detector only detects feature points from the edges of the segmentation masks, we do not need the feature points to be more extensive than that cause we are only using the features and the optical flow to validate that two vehicles on two consecutive frames are indeed the same vehicle.

We generate the segmentation mask binary images by first making the image a single channel image and assign the bit value 0 to black out the whole image. Then we

Figure 3.8

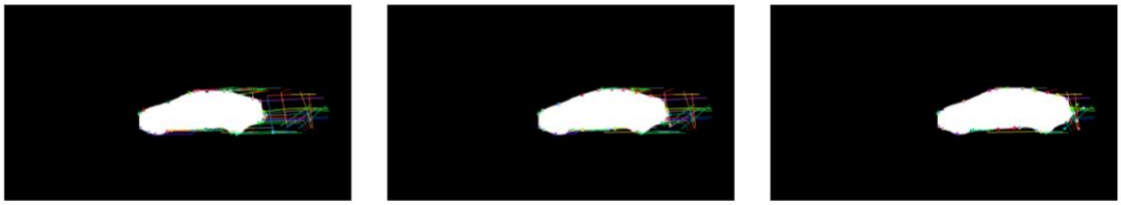
Calculating Optical Flow using Feature Points from Vehicle Images



access the npy file for the segmentation mask of that given frame and then traverse it to access the index of the pixel where to assign the bit value 1 to make it white. After traversing the whole npy while we are left with a binary image where the background is black and the vehicle segmentation is white. We use this binary image to detect the edges of the segmentation mask and track it across frames using Lucas-Kanade optical flow. If two consecutive images have notable optical flow and preserves significant features across two frames, then it is the same vehicle. Although we can validate it is the same vehicle, we cannot localize the vehicle using the feature points in a single frame. It is necessary to localize to make sure we do not include the vehicles from the side roads in our dataset. For that we will be using the bounding box.

Figure 3.9

Calculating Optical Flow using Feature Points from Segmentation Masks



Bounding Box Localization & Overlap for Frames

In this subsection we will first discuss how we are using bounding boxes to validate if a vehicle in two consecutive frames is the same vehicle or not. Later on, we will discuss how we validate across multiple viewpoints.

We use the bounding boxes generated by Mask R-CNN to check for bounding box overlap to infer if vehicles in consecutive frames are the same vehicle or not. As illustrated in figure 3.10, if a vehicle moves across multiple consecutive frames, then there will be significant bounding box overlap. We calculate the intersection over union (IoU) of two bounding boxes from two consecutive frames, and if the IoU is over 0.45, we consider it to be a significant overlap and classify them together. Intersection over union is calculated by first calculating the area of intersection of two boxes and then dividing the intersection with the calculated union of the boxes (as shown in figure 3.11).

There are other strategies to further enforce our dataset and make it robust. As you can see in figure 3.12, there are scenarios where the vehicle is entering or exiting the frame and partially out of frame, but the Mask R-CNN sometimes still localizes the

Figure 3.10

Progression of a Vehicle and its Bounding Box Across Three Consecutive Frames

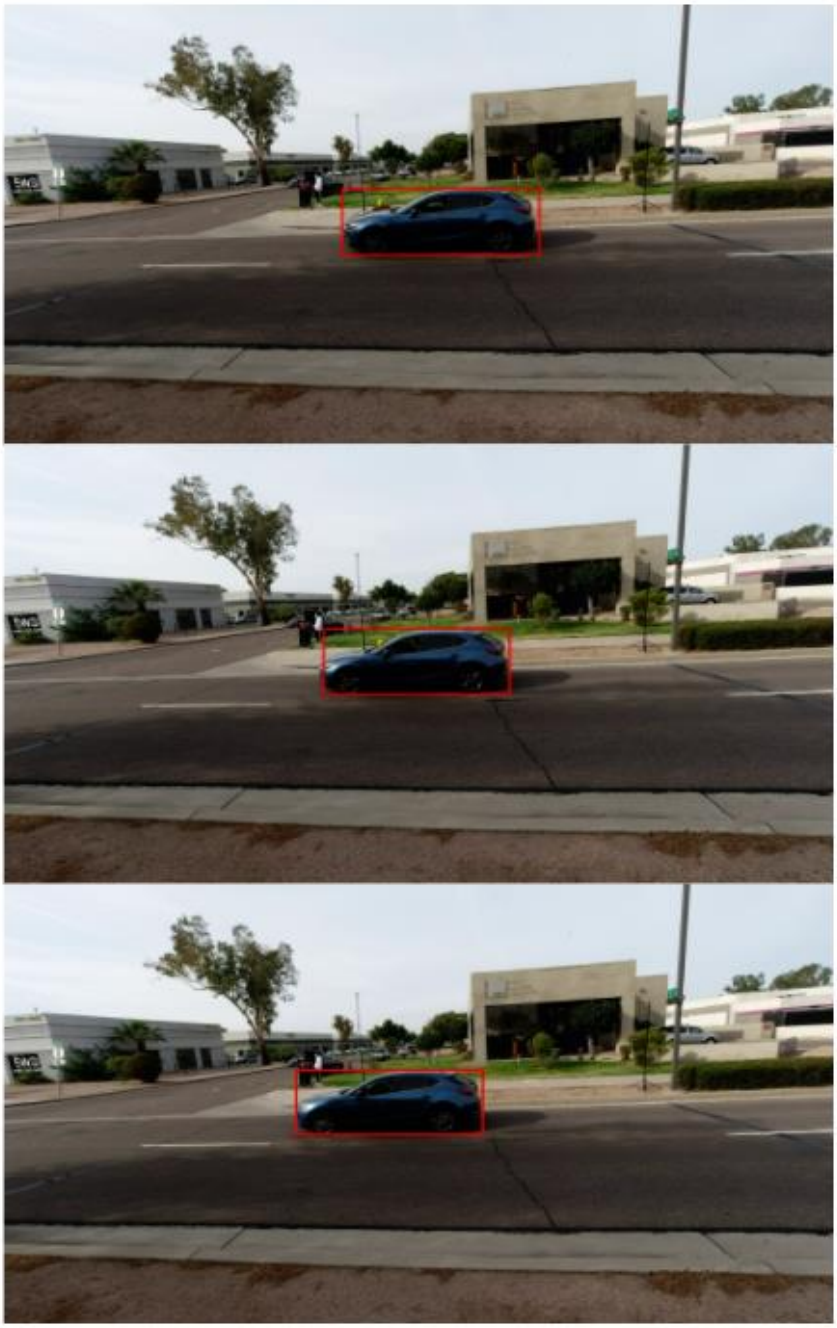
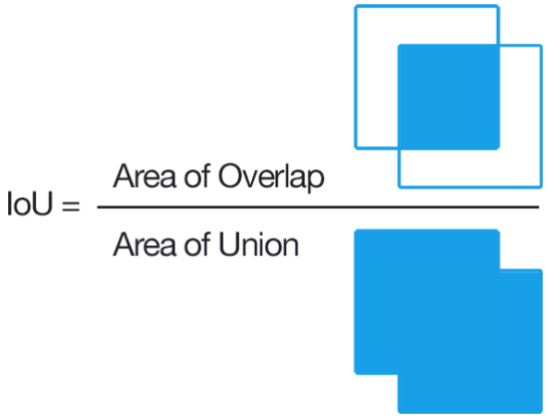


Figure 3.11

Calculating Intersection Over Union



vehicle and predicts a bounding box. We do not want these vehicle images in our MVVdb dataset because if someone intends to use our dataset for 3D object reconstruction from 2D the frame. We use the pixel coordinates of the bounding boxes to enforce that we do not consider the images where the vehicle is too close to the edges, e.g., for viewpoint *c0*, if the corners of the bounding boxes are not at least 30 pixels away from the edge of the frame, then we do not consider that vehicle image in our dataset.

Additionally, we also use bounding box pixel coordinates to eliminate vehicle instances we do not want in our MVVdb dataset. As mentioned earlier, there is a side road in the field of view of some of our cameras. We do not want the vehicles parked on that road or driving down that road (figure 3.13 (a)) to be included in our dataset because then it will not be a multi-view dataset – since the vehicle will not be visible from all the cameras. We contemplated to programmatically blackout some pixel regions (as shown in figure 3.13 (b)), but then there is a chance that it will also blackout some of the bigger

Figure 3.12

Mask R-CNN Localizing Vehicle Even When Vehicle is Partially Out of Frame

(a) (b)

Note. Figure (a) shows a Mask R-CNN localization for a vehicle partially out of frame when entering a frame, (b) same for a vehicle exiting a frame.

vehicles travelling on our main road. Taking the viewpoint c0 as example, to tackle this problem we measured the x coordinate of the region where the side road is merging on to the main road and put a condition that if the lower right bounding box corner of any vehicle is above that x coordinate, we will not consider that vehicle in our dataset.

The final challenge is to tackle the problem of occlusion which is an easy fix. For a single frame, if there are two bounding boxes that overlap, then we ignore those two vehicles. Since the bounding boxes do not perfectly encapsulate a vehicle but extends out in most cases, we do not classify it to be occlusion for any IoU over 0.0, but only if it is over a very small value of 0.05.

Figure 3.13

Frames From Viewpoint c0 Showing the Side Road

(a)

(b)

Note. In the above images, (a) shows a car driving down the side road, and (b) shows the side road blacked out.

Vehicle Tracking Across Multiple Viewpoints

In the previous section we discussed how to track a vehicle across multiple consecutive frames to validate if that vehicle is indeed one single instance or not. Similarly, in this section we will discuss our method of validating the same but across multiple viewpoints. Validating such is important because, as shown in figure 3.14, if there are two vehicles in a single frame visible from multiple viewpoints, then our automated dataset generation system needs to differentiate between the two in order to build a multi-view dataset. In this section we will discuss how we achieve that.

Feature Matching

Feature matching is defined as extracting interesting features and descriptors from an image and establishing preliminary matches with features extracted from another or

Figure 3.14

Two Vehicles in a Single Frame From Viewpoint c2

multiple other images. Feature matching is used in many Computer Vision problems e.g., stereo calibration, object tracking. In this work, we will be using feature matching to extract relevant features of a vehicle from two different viewpoints and then try to match those features to validate if they are actually the same vehicle or not.

To extract the feature points, we use the ORB (Oriented FAST and Rotated BRIEF) (Rublee et al., 2011). The ORB detector also generates binary string descriptors of these extracted features that can be used to differentiate one feature from another. ORB generates local descriptors which only gives a concise representation of the feature point's local neighborhood. We use the ORB detector instead of SIFT and SURF because the latter two algorithms are patented. Also, ORB works as good as SIFT and better than SURF while being substantially faster than both of them.

We detect the feature points and compute the descriptors using ORB. Our next step is to match the local descriptors of two images to see which feature points exists in both the images. We use the Brute-Force matcher to match the local descriptors which, as the name suggests, is a brute force method of matching descriptors. Hamming distance is used to match the descriptors as ORB generates binary string descriptors. We find the best matches and then sort the matches in an ascending order of distance. Finally, we check if the top k matches are below a certain threshold of distance; if they are, then are classified as the same vehicle.

However, using feature matching does not always give us favorable results. If there are two vehicles in a frame of similar shape and color, Brute-Force matcher struggled to match the ORB feature points. To use as a support when ORB feature

matching is not producing favorable results, we use the position of the bounding box in a frame to determine if a vehicle in two separate viewpoints is the same vehicle or not.

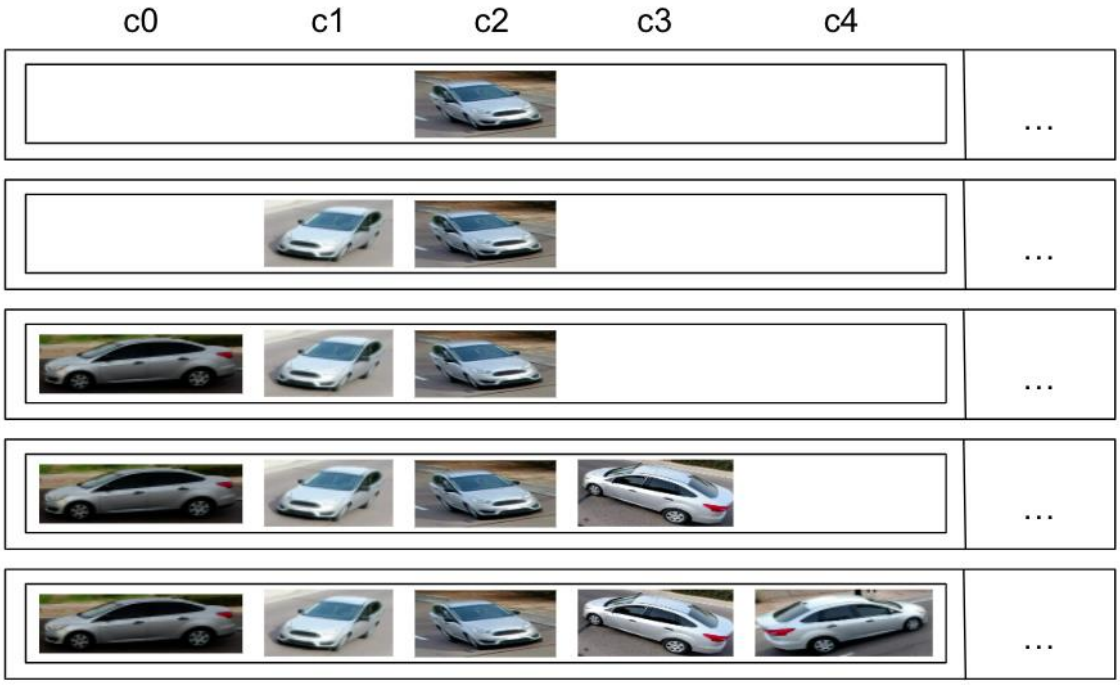
Bounding Box Localization for Multiple Viewpoints

In this subsection, we will discuss how we are using bounding boxes to validate if a vehicle seen from two different viewpoints is the same vehicle or not. To perform the validation, we have a simple solution. From viewpoint *c2*, we first get to see the vehicle before it is visible from any other viewpoint. Once it is visible from *c2*, it is inserted into a dynamic 2D data structure as shown in figure 3.15 – where a dynamic array contains another array (representing all the bounding boxes of a vehicle) of length 5, each index representing each viewpoint, where all the values are initialized with null. If there is only one vehicle visible from *c2*, when the same vehicle starts to get into the field of view of other cameras, then we simply add the bounding box of the vehicle to the already existing array representing the vehicle. But, if two cars are driving really close to each other, then there may be a possible scenario that two vehicles are visible from the viewpoint *c2* before they are visible from the other viewpoints. If that is the scenario, then when the first vehicle enters the field of view of another camera for the first time, there's a dilemma of which array to add the bounding box to. To resolve this dilemma, we simply check which bounding box that is visible from *c2* is ahead by comparing the coordinates of the bounding boxes. Whichever vehicle is ahead, we add the first instance seen from the other viewpoints to that array. For example, let us say we have two vehicles *v1* and *v2* visible from viewpoint *c2*, and no vehicles are visible from any of the other viewpoints. As we progress in time, a vehicle will begin to appear in another viewpoint, let us say

that viewpoint is *c1*. So, the question arises if the vehicle visible from *c1* is *v1* or *v2*. We know that whichever vehicle is ahead in the corresponding frame in *c2* will be the first vehicle that will be visible from another viewpoint, *c1* in this example. Therefore, we compare the position of the bounding boxes in *c2* to check which vehicle is ahead, and then we assign the vehicle visible from *c1* to that instance.

Figure 3.15

Dynamic 2D Data Structure Representing Vehicle Instances



CHAPTER 4

VEHICLE RE-IDENTIFICATION

The main contribution of our work is the proposal of a novel multi-view vehicle dataset (MVVdb). In this chapter, to help showcase how our MVVdb dataset stacks up against real-world challenges, we attempt to solve the vision problem of vehicle re-identification. In the following sections, we will first discuss what a re-identification problem is and its nuances. Followed by that, we will talk about our approach to tackling the problem by training a deep neural network. We will go through the details of the neural network and lastly, talk about our training strategy.

Problem Summary

There has been an explosion in the amount of data on the internet because of modern technology – especially image data. There has been an exponential growth in the number of cameras out there in the world, from more widespread surveillance cameras to webcams on all our laptops – all of us even walk around with cameras in our pockets that come with our mobile phone. With this sudden increase in the accessibility of cameras, most of the data that is getting captured through these devices is ending up on the internet for a variety of reasons. One of the biggest reasons being social media. People nowadays are able to capture every moment of their lives through a camera and share it will millions of people owing to social media. Other than that, online shopping has emerged in modern times and is preferred more over brick-and-mortar stores. With the millions of items currently getting sold online, comes millions of more images getting uploaded to

the internet. As a researcher, the best part being all this data is publicly available and can be accessed by anyone. Additionally, the use of image data is also widespread in industries like banking and surveillance, as already mentioned. With this increasing amount of image data comes the problem of image search. For example, if someone sees a dress that they like and wants to buy the same or similar dress online, they will want to perform a search on the database of an online e-commerce website. Let us take another example – a bank teller receives a fraudulent check and wants to confirm if they have received any similar fraudulent checks in the past by performing an image retrieval of all the similar checks. So, this is what an image retrieval problem is.

Image retrieval, in its most barebones form, is the problem of finding similar images from a database of images if given a query image. The most traditional forms of image retrieval include finding similarity in metadata such as captioning or keywords and calculating a similarity measure between such. However, in that case it still fundamentally remains a Natural Language Processing (NLP) problem rather than a vision problem. If image retrieval really needs to be practical, then it needs to be solved as a vision problem because majority of the image data that exists today are not annotated. That is what we aim to solve in this work. Vehicle re-identification is essentially an image retrieval problem. It is nothing but searching for a specific vehicle from a database of vehicle images when given a query vehicle image, but this time by computing similarities between inherent visual cues instead of finding similarities in annotated metadata.

Similarly, vehicle re-identification can also be described in terms of a vehicle make and model recognition problem. In vehicle make and model recognition, we train a

model on a labelled (make and model) dataset. After training, given an image, the model should be able to classify the image with its correct make and model. Our vehicle re-identification problem is essentially the same problem but without a dataset that has been labelled with the vehicle's make and model.

Approach towards Vehicle Re-Identification

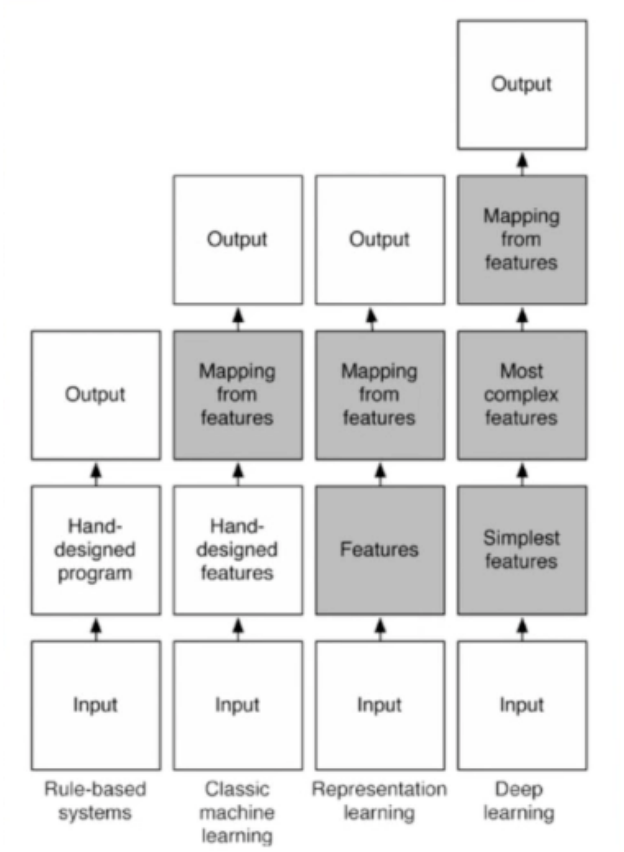
Our approach towards solving the vehicle re-identification problem is by training a deep neural network. For this task, we chose to use a ResNet-50 especially because of its performance on the ImageNet dataset. We believe that the pre-trained weights of the ResNet trained using the ImageNet should correlate well with our multi-view vehicle dataset. Our vehicle re-identification is simulating the problem scenario in surveillance system to identify a vehicle given the vehicle has been seen before. Since it is re-identification, in a practical setting, we should be re-identifying a vehicle seen through a camera that has been seen before through other cameras. So, for the training set and validation set split, we randomly select four viewpoints out of the five in our dataset for every individual vehicle instance and use it as our training set. Evidently, the last fifth viewpoint the model is not trained on is used as our validation set. We choose to fine-tune the pre-trained model by unfreezing all the layers and retrain the whole model again, end to end, with a very low learning rate.

ResNet Architecture

Since AlexNet (Krizhevsky et al., 2012), there has been a trend of going deeper with the layers. The more layers an input goes through, a model can extract more complex features and enrich the level of the features (shown in figure 4.1). Previous ImageNet models that came before Resnet were between 16 to 30 layers which made them capable in extracting high level features. However, with a deeper network comes the problem of vanishing gradient. To update the weights of a model, we need to use

Figure 4.1

Illustration Showing How a Deep Network Extracts Features

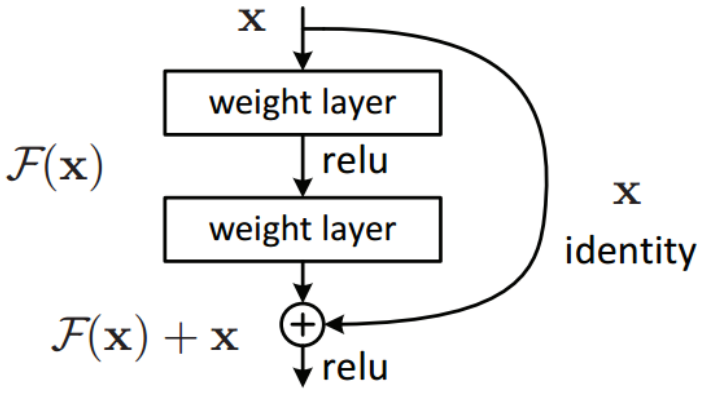


backpropagation while using the chain rule of derivative calculus. The repeated multiplications because of the chain rule throughout the many layers of a deep neural network may minimize the gradient drastically.

ResNet brought forward a solution to the vanishing gradient problem by introducing Residual Blocks which uses skip connections (shown in figure 4.2). A residual block takes the input of a convolution block and bypasses it or skips the layers of the convolution block to the output of the block. Then it is known as a residual block. This is done because we do not want the gradient to minimize exponentially and potentially causing the problem of a vanishing gradient. To do such, we want to add we add the input *x* to the output of the convolutional block *f(x)*, which is then we feed to the next layer as an input *y*.

Figure 4.2

Residual Block Architecture (He et al., 2016)



$$y = f(x) + x$$

Now, our objective is to skip past the input and feed it as an input to our next convolution block. Hence, to do that we want the value of the loss function f(x) to be 0.

$$f(x) = 0$$

After all the convolutions, the output of the last layer is passed through a global average pooling layer which then feeds the output to a dense fully connected network.

In this work, we are using a ResNet-50 which is 50 layers deep. Although it is three times deeper than a VGG16 (Simonyan et al., 2014), ResNet still has lesser parameters, which means it is faster. The architecture of ResNet is shown in figure 4.3.

Training Approaches

As mentioned previously, the ResNet model is initialized with pre-trained weight from its training on the ImageNet dataset. Instead of keeping the previous convolutional layers frozen, we fine-tune the model by unfreezing all the layers and training on our MVVdb dataset. Although we assume the ImageNet weights will transfer nicely to our dataset, we still opt to unfreeze the layers for fine-tuning to make the model adaptable to the high-level features of our dataset. Also, the surveillance type camera angle of our dataset is novel to the ImageNet as most vehicle images there are captured from eye level. We could not train the model on the whole MVVdb dataset generated due to hardware limitations. The dataset was stored on an external hard drive with a read speed much slower than what needed. Additionally, the million plus files containing the output of the Mask R-CNN caused data fragmentation and worsened the read speed drastically,

Figure 4.3

All ResNet Architectures along with ResNet-50 (He et al., 2016)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

in the process increasing computing complexity. So, we trained our model on a subset of our dataset, comprising of 114 instances or classes.

Since vehicle images in our dataset are cropped depending on the bounding box, the input images to the model are variable-sized input. To tackle that problem, we thought about using a Spatial Pooling Layer (SPP) which suggests of using multiple pooling layers with different scales between the last convolution layer and the fully connected layer instead of just one traditional pooling layer. However, if we implemented that, we would not have been able to stack images in batches for batch training and utilize the GPU performance. So, instead we opted to pad the images and then to scale down the resolution to 224x224.

We trained over 100 epochs with 30 as batch size. We opted to use Stochastic Gradient Descent (SGD) as our optimizer. We set the learning rate low at 1e-5 with a

momentum of 0.9 (table 4.1). In the next chapter we will discuss the results that we achieved.

Table 4.1

Hyperparameters for ResNet-50

Epoch	Batch Size	Optimizer	Learning Rate	Momentum
100	30	SGD	1e-5	0.9

CHAPTER 5

RESULTS

In this chapter, we will explore the results we achieved when generating the MVVdb dataset and when performing vehicle re-identification. In the first section we will discuss how our system or algorithm for generating a multi-view dataset works and if we are getting favorable results or not. In the following section, we will go over the results of our vehicle re-identification and how well a ResNet-50 model performed in that task.

MVVdb Results

We were able to generate approximately 500 instances for our MVVdb dataset from the recorded videos, but only till the 14th video in context of the reference viewpoint (*c0*). We could have generated plenty more instances from our multi-view roadside videos, but we could not due to a minor technical error. When the Mask R-CNN was run on our videos, for a vehicle within the frame of *c0_19*, the model predicted a bounding box but could not predict a segmentation mask for a specific frame. That resulted in an error in our dataset generation algorithm, as our algorithm searches for both the bounding box and the segmentation mask. We assume if the dataset is generated for all 62 videos, we will have a multi-view dataset of approximately 3000 vehicles.

Our algorithm for automatically generating a dataset performed really well most of the time. The purpose of this algorithm was to generate a multi-view dataset given a multi-view set of videos without any manual input. Although we did not achieve that

hundred percent of the time, it still performed as well as expected. However, all the hiccups our model faced was not because of the algorithm itself, but because of the bounding box localization output produced by the Mask R-CNN.

For instances of sedans and hatchbacks, Mask R-CNN performed consistently and the best. As a result, the instance images of sedans and hatchbacks are also the best among the others (as shown in figure 5.1 and figure 5.2 respectively). One of the limitations for our MVVdb dataset are pickup trucks, vans, and trucks. The reason is mostly because of poor detection and localization by Mask R-CNN. Pickup trucks and trucks have unconventional vehicle shapes which are troublesome for the Mask R-CNN. As you can see in figure 5.3 that Mask R-CNN struggled to detect this pickup truck properly and localize it with its bounding box. However, we can see that Mask R-CNN performed well for the instance of a pickup truck in figure 5.4 because of its boxier shape due to the metal bars. For the case of trucks, they are too big and take up lot of area in the frame for the Mask R-CNN to properly localize it (shown in figure 5.5). However, if a truck is far away from the camera and only occupies a smaller area in a frame, Mask R-CNN performs much better in localizing and predicting a bounding box (figure 5.6). Mask R-CNN also failed to properly detect and localize any other big vehicles (e.g., SUVs) especially if it is driving down the lane closer to the camera (figure 5.7).

Poor detection and localization results by Mask R-CNN also effects in the fragmentation of the MVVdb dataset. Because of poor detection, when a pickup truck or a truck is driving down the section of the road there are a lot of frames in between where the Mask R-CNN fails to detect and localize the vehicle. For example, if a vehicle enters the frame in frame 0 and is detected by the Mask R-CNN but then fails to detect and

localize it again till it reaches frame 4, then our algorithm thinks that the vehicle already left the scene, and hence, creates a new instance with only image of the vehicle detected at frame 0 and then starts to create a new instance once it detects it again at frame 4. To fix this issue, we eliminated all instances which did not have at least 4 images from all the viewpoints.

Vehicle Re-identification Results

In this work, we attempt to solve the problem of vehicle re-identification using a deep neural network with the help of our own created multi-view vehicle dataset, MVVdb. We fine-tune a pre-trained ResNet-50 model on our MVVdb dataset over 100 epochs and a batch size of 30. We also train a ResNet-50 model from scratch on the MVVdb dataset and compare the results.

The pre-trained ResNet converges well before 100 epochs. The pre-trained ResNet reaches an accuracy of 98.1% while the ResNet trained from scratch reaches 78.7%. The pre-trained model achieves a mAP score of 0.942 while the model trained from scratch achieves a mAP score of 0.587. The comparative study between the two models is showed in figure 5.8. It also shows that the pre-trained ResNet-50 converges well before 100 epochs. So, we perform another comparative study, but this time over 30 epochs, and throw a pre-trained VGG16 model into the mix. The comparative study between a pre-trained ResNet-50, a ResNet-50 trained from scratch, and a pre-trained VGG16 is shown in figure 5.9. The pre-trained ResNet achieves an accuracy of 95.9% with a mAP score of 0.882, while the VGG16 only reaches an accuracy of 29.8% with a mAP score of 0.256. On the defense of VGG16, the model has more weights to train than

the ResNet-50, so it takes longer to converge. But the results show that the ResNet-50 was successful in re-identifying a vehicle when seeing from a different viewpoint. All comparative results of the accuracy and the mAP score between all the models are listed in the table 5.1.

Table 5.1

Comparative Study of the Accuracy and mAP Scores of the Models

Epochs	Metrics	Pre-trained RestNet-50	Scratch ResNet-50	Pre-trained VGG16
100 Epochs	Accuracy	0.981	0.787	-
	mAp	0.943	0.587	-
30 Epochs	Accuracy	0.959	0.610	0.298
	mAp	0.882	0.309	0.256

CHAPTER 6

CONCLUSION AND FUTURE DIRECTIONS

In this chapter we will discuss the conclusion of our research and the future directions that can be explored. In the first section we will firstly discuss about our primary contribution in this work, the MVVdb dataset and our method of generating it independently without human intervention. Additionally, we will examine the performance of our vehicle re-identification performed with our MVVdb dataset. In the section after that, we will discuss the different avenues of research that we or other researchers can embark upon taking advantage of our dataset. We will discuss various ways we can improve our dataset and make it more robust. Finally, we will carefully look at the possible future use cases of our multi-view dataset other than vehicle re-identification and how it may help to solve further research problems.

Conclusions

In this work, we proposed a novel vehicle dataset with images captured from multiple viewpoints. Along with that, we also promised an autonomous dataset generation algorithm which, if given videos recorded from a camera system similar to the one showcased in our work, has the potential to automatically generate a large-scale multi-view dataset with minor tweaks to the algorithm as needed. Although we were successful in creating and delivering our dataset, we could not generate a dataset as big as we expected due to an unprecedented issue. Nonetheless, our work still showed promise and demonstrated potential in scaling to generate a large-scale dataset. Our algorithm is

reliant on the output and performance of Mask R-CNN or any other localization and segmentation model to a great extent. If we can get our hands on a more robust localization algorithm, the autonomous dataset generation algorithm will also improve consequently. We will discuss one such idea in the next section.

We also contributed, in this work, by attempting to solve the vehicle re-identification problem with the help of a deep neural network. It also acted as a showcase of one of the different use cases of our MVVdb dataset. We selected to use a ResNet-50 as our neural network to tackle this task because of its extraordinary performance on the ImageNet dataset. We fine-tuned the ResNet model pre-trained on the ImageNet dataset on our multi-view vehicle dataset. We trained the ResNet on only 114 instances or classes due to computing complexity. The ResNet performed really well at re-identifying vehicles with a top-1 validation accuracy of 98.1% and mean average precision of 0.943.

Future Directions

In this section we will discuss different measures that we can take to further improve our work and future research directions that can be explored. As discussed in the previous section, our autonomous dataset generation algorithm can be vastly improved with the help of a better localization and segmentation algorithm. We propose to re-train the Mask R-CNN model for future development of this work. For the Mask R-CNN to perform better on the MVVdb dataset, it should be re-trained on the same. We can manually annotate a subset of our dataset by hand drawing bounding boxes along with segmentation masks, and then feed it to a Mask R-CNN to re-train it. The subset should be biased towards the kind of vehicles Mask R-CNN performs the worst on (e.g., pickup

trucks, vans, trucks). The neural network already performs significantly well on sedans and hatchbacks, so annotating and re-training on those will not improve the model by much. We should also focus on big vehicles or SUVs or images where the vehicle is considerably close to the camera filling up most of the frame. If trained on a subset of 400 – 500 annotated images, we believe Mask R-CNN will produce substantially better output.

Other than vehicle re-identification, the MVVdb dataset can also be used for fine-grained vehicle make and model recognition. Vehicle re-identification and vehicle make and model recognition are practically the same problem – the only difference being the vehicle re-identification does not need a labelled dataset, while fine-grained vehicle make and model does. One can also take our MVVdb dataset and label the instances for fine-grained vehicle make and model recognition. In our dataset, from viewpoint c2 and c4, we can read the license plates of the vehicles. By going through the license plates and looking up the make and model of the vehicle registered to that license plate online, we can label our dataset. There is also future scope of research in implementing an Automated License Plate Reader (ALPR) system – like Ap et al., 2020 – on the MVVdb dataset. Other than that, our dataset is ideal for solving 3D vision problems. Our dataset can be utilized to generate 3D voxel models from multiple viewpoint images by carving a block of voxels. Then by doing a pixel to voxel mapping, we can essentially create a 3D model which can be viewed from novel viewpoints. With the help of this textured 3D reconstructed voxel models, we can practically synthesize a large-scale dataset from a small dataset, essentially solving the problem to data collection.

All in all, there are multiple scopes of improving the dataset proposed. There are also various research opportunities that can be explored with the help of our dataset.

REFERENCES

Ap, N. P., Vigneshwaran, T., Arappradhan, M. S., & Madhanraj, R. (2020) "Automatic Number Plate Detection in Vehicles using Faster R-CNN," *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pp. 1-6, doi: 10.1109/ICSCAN49426.2020.9262400.

Chen, D., Cao, X., Wang, L., Wen, F., & Sun, J. (2012). Bayesian Face Revisited: A Joint Formulation. *ECCV*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 *IEEE conference on computer vision and pattern recognition* (pp. 248–255).

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303-338, 2010

Geiger, A., Lenz, P., Stiller, C., Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR),* Geiger2013IJRR.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

He, K., Zhang, X., Ren, S., Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 37. 10.1109/TPAMI.2015.2389824.

He, K., Gkioxari, G., Dollár, P., & Girshick, R.B. (2017). Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980-2988.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3), 574–591. https://doi.org/10.1113/jphysiol.1959.sp006308

Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3D Object Representations for Fine-Grained Categorization. *2013 IEEE International Conference on Computer Vision Workshops*, 554-561.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. , , 32--33.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou & K. Q.

Weinberger (ed.), *Advances in Neural Information Processing Systems 25* (pp. 1097--1105). Curran Associates, Inc..

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* (p./pp. 2278--2324),.

Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C.L. (2014). Microsoft COCO: Common Objects in Context. *ECCV*.

Liu, H., Tian, Y., Wang, Y., Pang, L., & Huang, T. (2016). Deep Relative Distance Learning: Tell the Difference between Similar Vehicles. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2167-2175.

Lou, Y., Bai, Y., Liu, J., Wang, S., & Duan, L. (2019). "VERI-Wild: A Large Dataset and a New Method for Vehicle Re-Identification in the Wild," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 3230-3238, doi: 10.1109/CVPR.2019.00335.

Minsky, M. (1967). *Computation: finite and infinite machines*. Englewood Cliffs, N.J: Prentice-Hall.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 6517-6525.

Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision* (p./pp. 2564-2571), November, .

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556,* .

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition* (p./pp. 1--9), .

Tafazzoli, F., Frigui, H., & Nishiyama, K. (2017). A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* 874-881.

Torralba, A. & Fergus, R. & Freeman, W. (2008). 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE transactions on pattern analysis and machine intelligence.* 30. 1958-70. 10.1109/TPAMI.2008.128.

Yang, L., Luo, P., Loy, C. C. & Tang, X. (2015). A large-scale car dataset for fine-grained categorization and verification.. *CVPR* (p./pp. 3973-3981), : IEEE Computer Society. ISBN: 978-1-4673-6964-0

APPENDIX A

REFERENCED FIGURES

Figure 5.1

Results for Sedans

Figure 5.2

Results for Hatchbacks

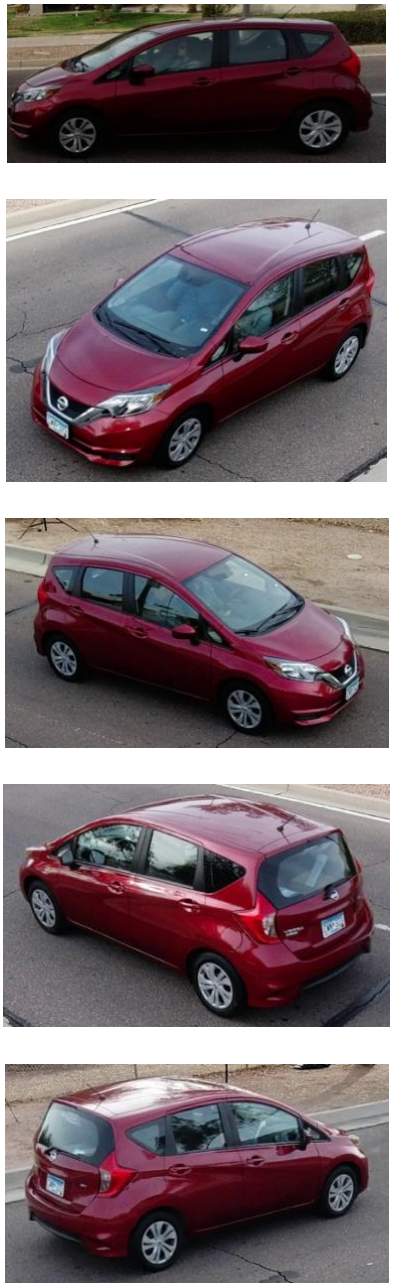


Figure 5.3

Poor Detection Result by Mask R-CNN

Figure 5.4

Proper Detection by Mask R-CNN due to Boxy Silhouette

Figure 5.5

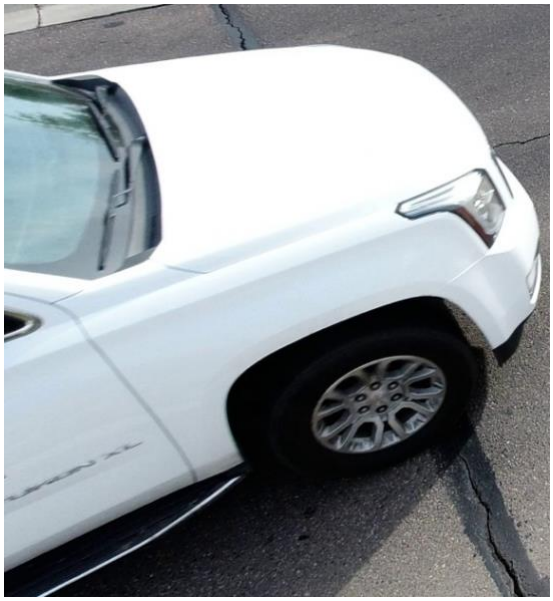
Showing Poor Localization Results by Mask R-CNN for Trucks

Figure 5.6

Localization Results by Mask R-CNN when Bigger Vehicles are Far Away

Figure 5.7

Localization Results by Mask R-CNN for a Big SUV



(a)

(b)

Note. In the above image (a) showing poor localization result by Mask R-CNN for a big SUV when closer to camera, and (b) showing better localization result when farther away from camera.

Figure 5.8

Comparative Study Between a Pre-trained vs Scratch ResNet-50 Trained Over 100 Epochs

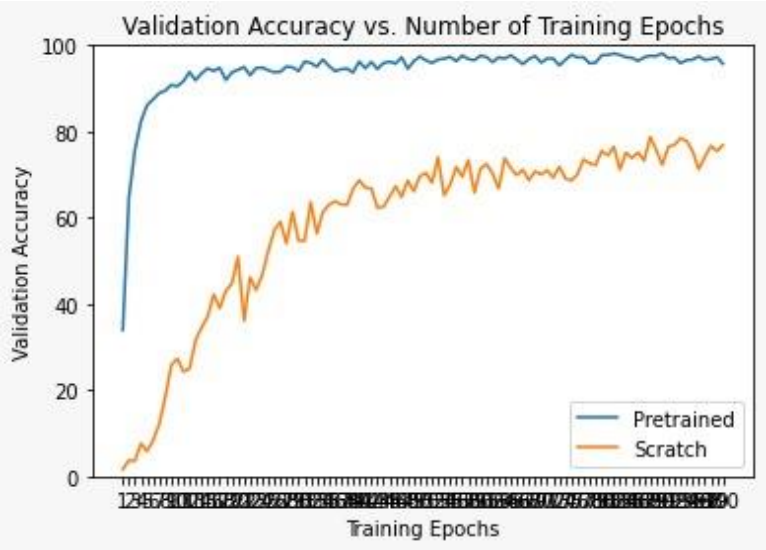


Figure 5.9

Comparative Study Between a Pre-trained vs Scratch ResNet-50 vs VGG16 Trained Over 30 Epochs

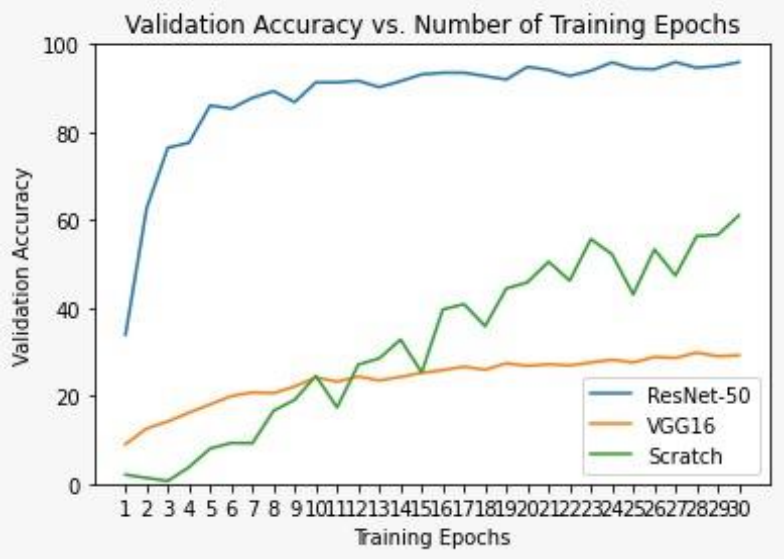


Table 2.1

Quantity Distribution of CompCars Dataset

Viewpoint	Number in total	Number per model
Front (F)	18431	10.9
Rear (R)	13513	8.0
Side (S)	23551	14.0
Front-side (FS)	49301	29.2
Rear-side (RS)	31150	18.5

Note. Numbers of viewpoint images are not balanced among different car models because the images of some less popular car models are difficult to collect.

APPENDIX B

VEHICLE RE-IDENTIFICATION RESULTS

The following are the results the various models achieved when performing vehicle re-identification.

Figure A.1

Pre-trained ResNet-50 Error Over 100 Epochs

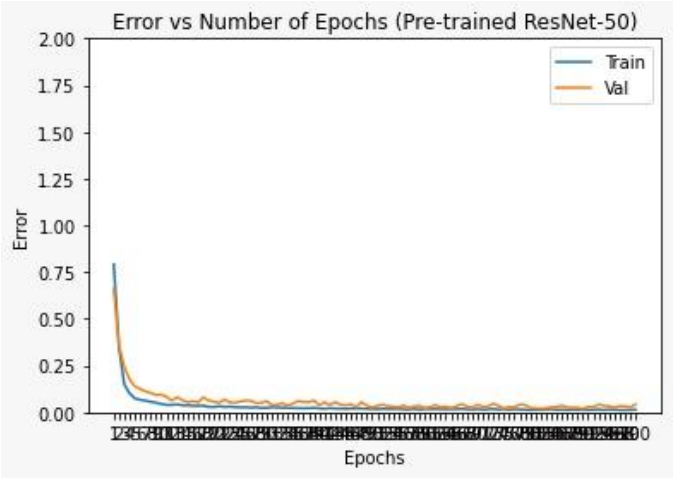


Figure A.2

Pre-trained ResNet-50 Loss Over 100 Epochs

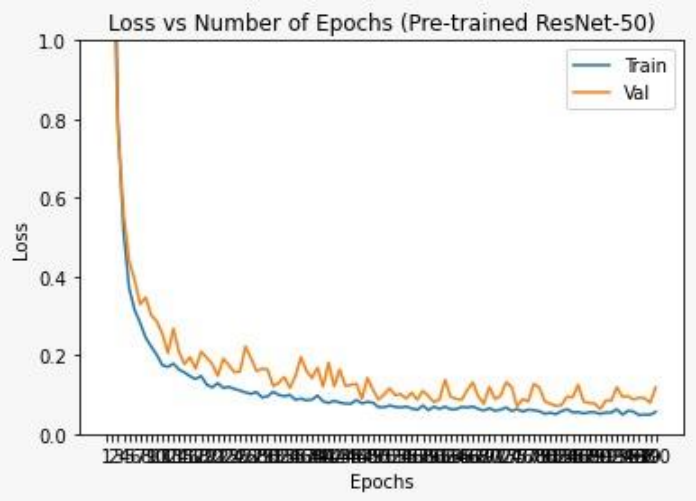


Figure A.3

Scratch ResNet-50 Error Over 100 Epochs

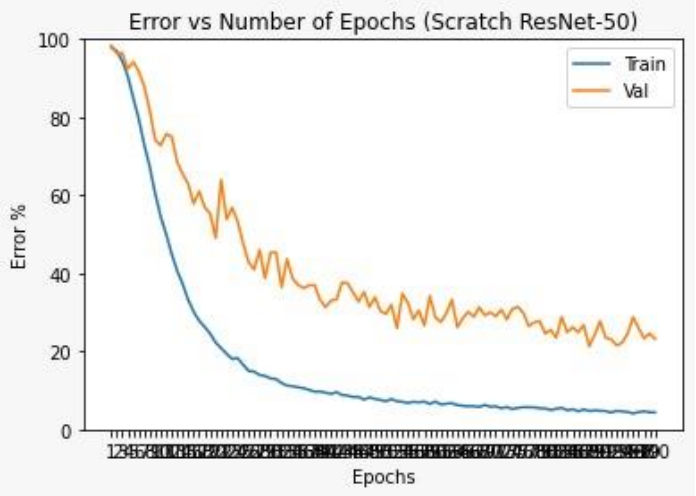


Figure A.4

Scratch ResNet-50 Loss Over 100 Epochs

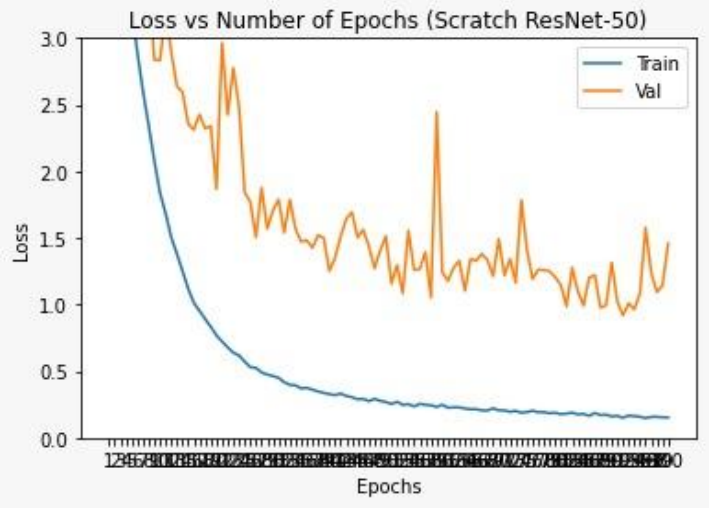


Figure A.5

Pre-trained ResNet-50 Error Over 30 Epochs

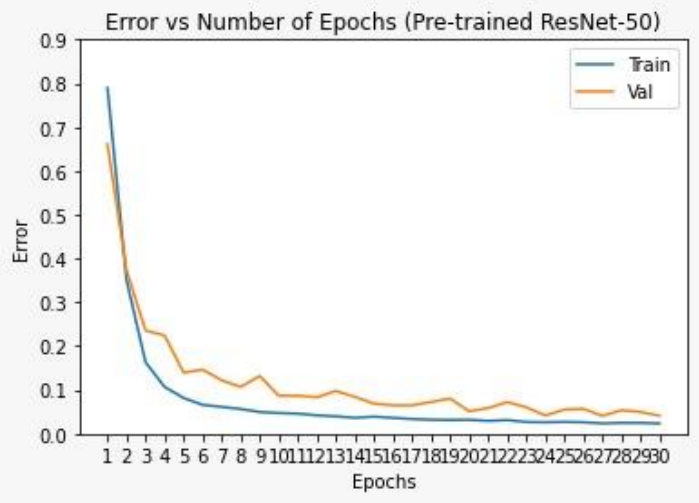


Figure A.6

Pre-trained ResNet-50 Loss Over 30 Epochs

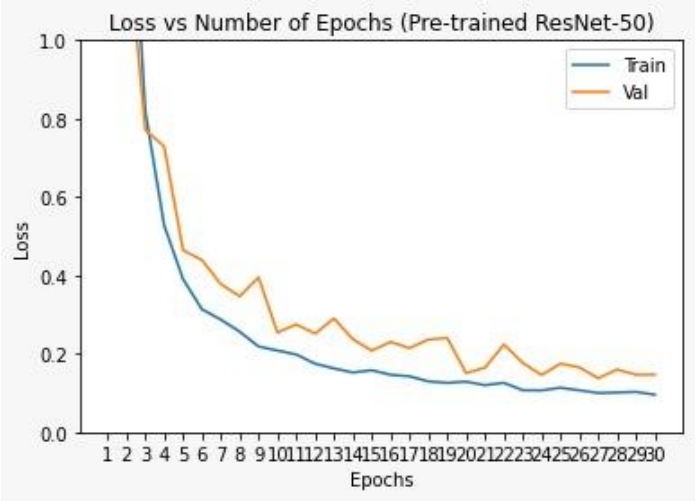


Figure A.7

Pre-trained VGG16 Error Over 30 Epochs

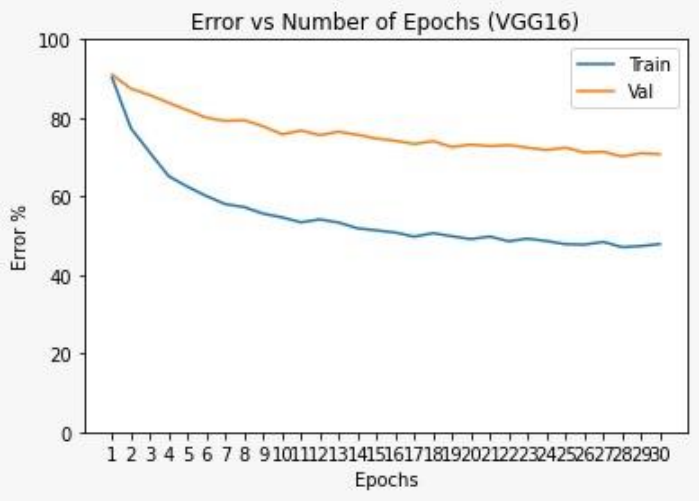


Figure A.8

Pre-trained VGG16 Loss Over 30 Epochs

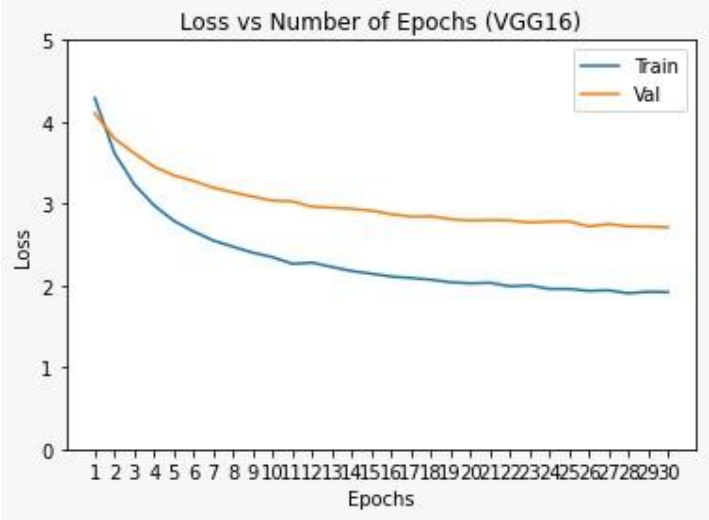


Figure A.9

Scratch ResNet-50 Error Over 30 Epochs

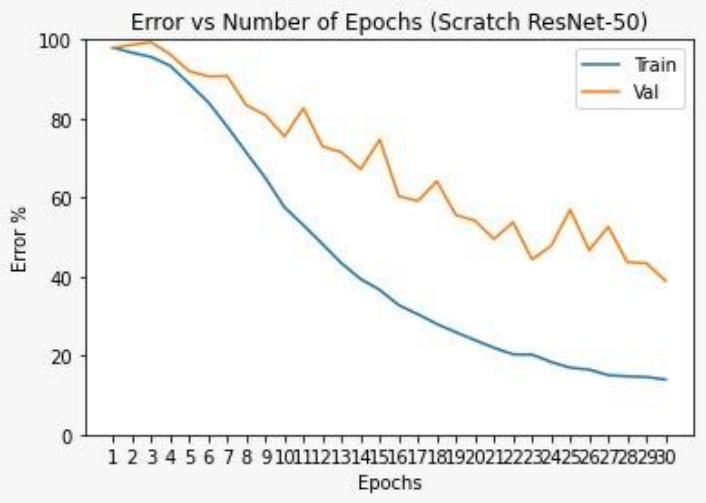
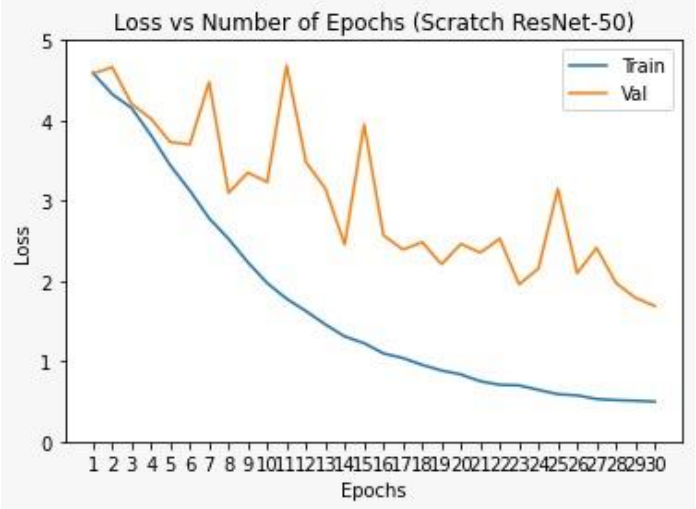


Figure A.10

Scratch ResNet-50 Loss Over 30 Epochs



APPENDIX C

LINK TO MVVDB DATASET

The dataset for this work is available at

https://www.dropbox.com/s/9hnq95y1db6kvyg/MVVdb.zip?dl=0

APPENDIX D

LINK TO CODE REPOSITORY

The code for this work is available at https://github.com/JonGuha/vehicle-re-identification