BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

VEHICLE MAKE AND MODEL RECOGNITION
ROZPOZNÁNÍ VÝROBCE A MODELU VOZIDEL

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **ADAM GREGOR**
AUTOR PRÁCE

SUPERVISOR **ROMAN JURÁNEK,**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

146307

Institut:	Department of Computer Graphics and Multimedia (UPGM)
Student:	**Gregor Adam, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Computer Vision
Title:	**Vehicle Make and Model Recognition**
Category:	Computer vision
Academic year:	2022/23

Assignment:

1. Study the topic of vehicle classification in traffic systems.
2. Study methods for image classification with neural networks including the concepts of embedding, attention and transformers.
3. Compile a large scale dataset of images available on the web covering as wide range of vehicle makes and models as possible.
4. Implement methods for vehicle make and model recognition
5. Make experiments with the dataset and other available datasets. Evaluate the properties of the various methods.
6. Discuss possible applications of your methods and data in real systems with respect to accuracy, speed and hardware requirements.

Literature:

1. A. Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2021.
2. Ge, Zheng, et al. "Yolox: Exceeding yolo series in 2021." *arXiv preprint arXiv:2107.08430* (2021).

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Juránek Roman, Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	17.5.2023
Approval date:	6.4.2023

Abstract

In the practical part of the diploma thesis, the task of identifying the manufacturer and model of a vehicle (VMMR) was implemented. In the first part, a dataset of vehicles was compiled for machine learning purposes that consists of images from the Internet. This resulted in over 6 million images of cars, buses, motorbikes and trucks usable for the VMMR task. Next, as part of the experiments, a standard classification was used on a part of the dataset, when the encoder is followed by a classification layer implemented using a neural network. Also an approach with a supervised contrastive learning method, clustering embeddings from encoder for easier classification, was used. Since the first mentioned approach returned more accurate results, it was used in the further experiments. There, a larger portion of images from our dataset was used for training a classifier for the VMMR task. Other classifiers were trained on the Stanford Cars and Comprehensive cars datasets. Lastly, when comparing the functionality of the classifiers on different datasets we have found that the classifier trained on our dataset performed the best.

Abstrakt

V praktické části diplomové práce byla realizována úloha ozpoznání výrobce a modelu vozidla (VMMR). V první části byla pro účely strojového učení sestavena datová sada vozidel sestávající se z obrázků z Internetu. Takto bylo získáno přes 6 milionů obrázků aut, autobusů, motorek a dodávek, použitelných pro úlohu VMMR. Dále byla v rámci experimentů na část datové sady použita standardní klasifikace, kdy na enkodér navazuje klasifikační vrstva realizovaná použitím neuronové sítě, a přístup, kdy za pomocí metody supervised contrastive learning byly embeddingy z enkodérů shlukovány za účelem snazší klasifikace. Jelikož první uvedený přístup vracel přesnější výsledky, byl použit v dalších experimentech. V nich se použilo větší množství obrázků z naší datové sady k natrénování klasifikátoru pro VMMR. Další klasifikátory byly natrénovány na datových sadách Stanford Cars a Comprehensive cars. Posléze bylo při porovnávání funkčnosti klasifikátorů na různých datových sadách shledáno, že klasifikátor trénovaný na naší datové sadě si vedl nejlépe.

Keywords

vehicle dataset, VMMR, neural networks, computer vision

Klíčová slova

dataset vozidel, VMMR, neuronové sítě, počítačové vidění

Reference

GREGOR, Adam. *Vehicle Make and Model Recognition.* Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Roman Juránek,

Vehicle Make and Model Recognition

Declaration

I declare that I developed this master's thesis independently under the supervision of Roman Juránek I have listed all literary sources, publications and other sources from which I drew.

. . . .
Adam Gregor
May 15, 2023

Acknowledgements

I would like to thank my supervisor Roman Juránek for his guidance and all the support he had provided me with. I would also like to thank my proof readers Dan, Zdeněk, David and Lucie for doing this. And of course to my family for everything.

Computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Contents

List of Figures

Chapter 1

Introduction

Vehicle classification (VC) is a scientific discipline focused on the transformation of input vehicle information, into a discrete set of classes to which the vehicle belongs. The approaches and hardware used in the VC differ based on the task. These may be various, from detection of the vehicle's presence through classification of whether a vehicle is overweight to an estimation of its orientation or manufacturer recognition. The most commonly used hardware is a camera as it is capable of covering most of the desirable information. As well as many other fields of computer science, vehicle classification underwent a huge progression over the last decades. The ongoing increase of hardware's computational power together with the rapid progression in the field of machine learning (especially in deep-learning) allows for the creation of models which would not be possible to create in the past. Due to this, many VC applications are used in the modern world - automatic toll gates, parking space management, traffic security cameras or self-driving cars.

The main focus of this thesis is to perform a vehicle manufacturer and model recognition (VMMR) task. This is, unlike a simple detection of a vehicle, a fine-grained classification problem requiring a thorough knowledge of the vehicle's attributes. To succeed, it is necessary to get acquainted with different approaches of VC and select the correct approach to vehicle observation. It was also needed to collect a new dataset of vehicles as the existing ones are not comprehensive enough. We have hence collected the largest dataset of vehicles we are aware of. To train a model on such a dataset, we have experimented with classification approaches. Based on our findings, we have created a model outperforming models trained on other datasets.

In Chapter 2, VC methodologies are overviewed to find the best approach to the VMMR task. Chapter 3 focuses on vehicle datasets. The topic discussed in Chapter 4 is convolutional neural networks. In Chapter 5 a new vehicle dataset is created. In Chapter 6 experiments are performed on a subset of the dataset to find the best approach to the VMMR task. Finally, in chapter 7 a model is trained on a big portion of data from our dataset and compared with models trained on different datasets.

Chapter 2

Vehicle Classification Systems

In the modern world, vehicle classification is used in many places. It is an underlying approach for applications such as traffic flow monitoring, surveillance, parking systems or autonomous cars systems. Because of the great variety of applications, the VC tasks differ in their required outputs as well as the way of obtaining them. For example, while a parking system works only with information about a vehicle's presence in a given parking spot, an autonomous co-pilot works with information on the vehicle's speed, trajectory, position, etc.

In this chapter, we will focus on different methodologies used to capture vehicle attributes for the task of VC. We take a look at their principles, sensors they use and list of VC tasks for which they may be used. At the end of the chapter, approaches are compared against each other and the most suitable one is chosen for our task of VMMR.

2.1 Vehicle Classification Approaches

The methods of VC approaches can be classified by types of sensors used to obtain the input information. Based on the required input information and application type, corresponding sensor (or a set of sensors) is used, as each sensor provides different attributes which can be measured using it. Sensors also differ in the quality of measurement, setup and price. Based on used sensors, the methods can be divided as displayed in Fig. 2.1.

In Fig. 2.1, methods are divided into three main groups: non-intrusive, intrusive and off-roadway based on the installation process of their sensor:

- Non-intrusive methods use sensors positioned near the place of interest and may sometimes be used for several lanes (as with cameras). These sensors are generally easier to install and maintain and are unaffected by the quality of the road.
- Intrusive sensors are installed in the road of interest. This may be done by placing sensors into holes in the road, into tunnels underneath it or by placing them in a form of an anchor. This makes the installation and maintenance process more difficult.
- The last group of sensors are off-roadway sensors. While intrusive and non-intrusive sensors have local nature (they can be used only for the one place they were installed on), these sensors are mobile, which is making them global nature methods. Off-roadway methods usually make use of satellites, for example in a form of aerial view images or GPS module data.

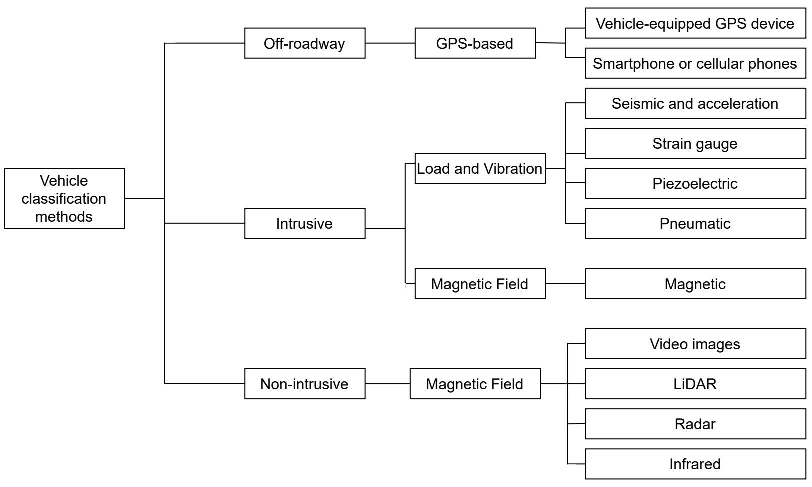


Figure 2.1: Categorization of vehicle detection methods. Image based on Shokravi et al. [30].

In the next sections, we take a closer look at some classes of methods based on their used sensors. The group of GPS-based methods is omitted, as their off-roadway nature is rendering them inappropriate for our task of vehicle and model classification for vehicles.

2.2 Image-Based Methods

This class of VC stands on processing a single or a sequence of images taken from the place of interest. Sensors capturing the images may be surveillance video cameras, omnidirectional cameras, aerial cameras, Closed-Circuit Television (CCTV) or normal cameras.

According to a review of available literature on VC made by Shorkavi et al. [30] and Sundaravalli et al. [28], most VC research focuses on image-based methods. This may be given by the increasing computational power of computers together with the amount of information that can be received from a single image or a sequence of images. From one image the information may be on the number of vehicles in the observed place, the vehicle's direction or **the type and model** of the observed vehicle. From several images, even the vehicle's speed can be calculated.

This wide spectrum of received information makes this class of methods suitable for many applications. Images taken for different applications may look like in Fig. 2.2. Note that while the sensor is the same, i.e. camera of some sort, the views of vehicles are different. The position between the vehicle and the camera is a crucial design part of applications using images and has to be set the way the application needs it to be. For example, in Fig. 2.2 a) we can get more information on the vehicle such as its direction, registration number etc. at the cost of having less area covered by the camera, making the produced images suitable for highway surveillance, for example. In Fig. 2.2 b) the situation is different as less information can be retrieved on individual vehicles. This decrease in information is however substituted by an increase in scope. Such image can be, for example, used by a parking management system as the information on the vehicle's presence can still be obtained.

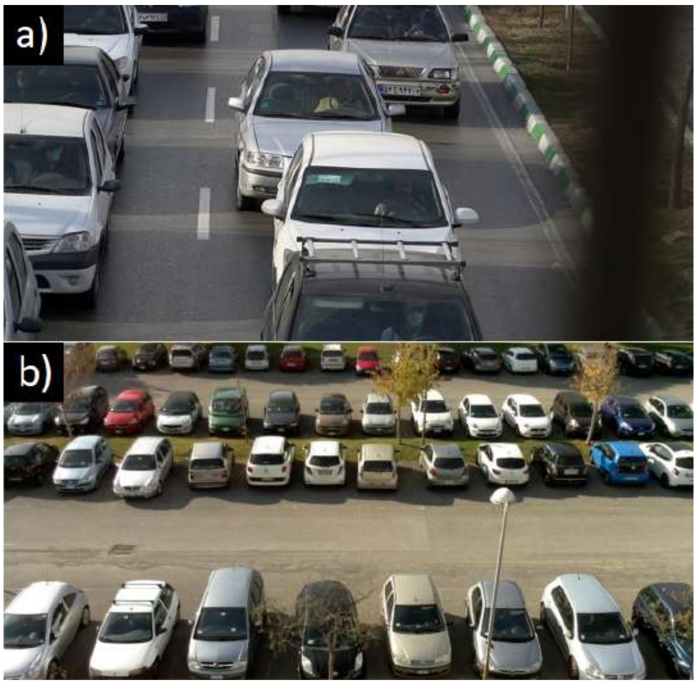


Figure 2.2: A comparison of different images used in image-based methods. Image a) taken from Maryam [25] b) taken from Amato et al. [1].

Image-based methods generally compose of pre-processing, feature extraction and selection and classification:

1. The purpose of pre-processing step is to enhance the quality of the input image to improve the accuracy of the classification. Some of the most commonly used techniques are:
    - Image segmentation – is used to locate the objects of interest in the image on which the classification is performed. The output masks of the object can also be used for background subtraction. An example of image semantic segmentation can be seen in Fig. 2.3.
    - Shadow removal – The object of interest may be detected together with its shadow as seen in Fig. 2.4. To acquire sufficient results from the classifier, it is necessary to separate them.
    - Occlusion handling – This pre-processing step manages to keep track of the object of interest even when it is in an occluded position.

2. In the feature extraction step, suitable features are extracted from the image to be further used in the detector. SIFT, ORB and SURF algorithms are often used to acquire feature vectors. Another used source of features is texture. Its feature extraction methods may be of first or second order, depending on how much information from the texture they utilize. Methods of first order do not use any structural information from the texture and an example of such method output is a histogram of the texture. Methods of second order utilize structural information and an example of such features may be angular and radial features of the texture's power spectrum. Another feature extraction approach is using neural networks as discussed in chapter 4.

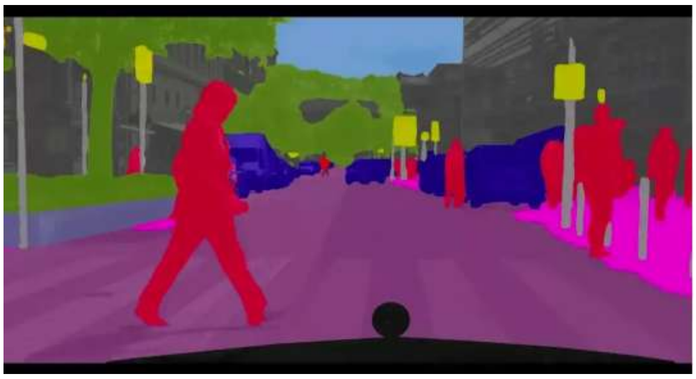


Figure 2.3: An example of image segmentation. Every class of object (e.g. person, road, and traffic sign) is coloured by its assigned colour. Image taken from Kanan [14].

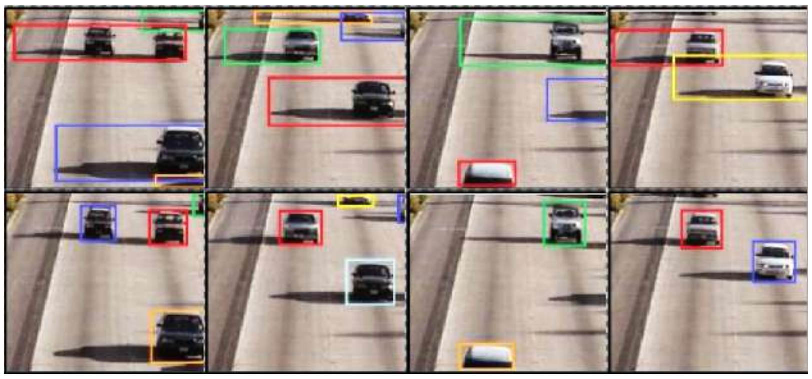


Figure 2.4: On the upper row, vehicles were incorrectly detected with their shadows, on the bottom row shadow removal was applied which corrected the detection. Image taken from Taha et al. [34].

3. The classification is the final step of the VC using image-based methods. According to Shorkavi et al. [30], the most commonly used algorithms are neural networks (NN) and support vector machines (SVN) for classification and recognition tasks and Gaussian mixture models (GMM) for image segmentation. Other algorithms mentioned are forest tree, nearest neighbour, decision tree learning, genetic fuzzy classifier, principal component classifier and Bayesian networks.

In Sundaravalli et al. [28], many of the compared VC methods were image-based, however, none of the classifications were used for the VMMR task. The finest classification granularity the algorithms provided was a classification of vehicles into groups based on their size. Other solutions that focus on VMMR task, such as Carnet[1] or Platerecognizer[2], are commercial. This means that **no reference solution is available**.

2.3 Other Vision-Based Methods

Other sensors that may be used for VC that are vision-based are radar, LiDAR and infrared cameras.

[1] https://carnet.ai
[2] https://platerecognizer.com

Radars are built on transmission of radio waves in predetermined directions. These radio waves are then reflected, scattered and absorbed by objects on contact. The reflected part of the transmitted signal is then received again by the radar, strengthened and may be further analysed. Besides using radar only for general detection of vehicles, the Doppler Effect may be utilised to detect the vehicle's speed, acceleration and direction. The Doppler Effect describes a change in a signal's frequency and wavelength based on the relative speed between the transmitter and receiver. Radars are insensitive to bad weather and offer a non-intrusive form of measurement.

LiDAR (Light Detection and Ranging), similarly to radar, emits a signal in predetermined directions which is then partially reflected, received and analysed. Instead of radar's radio waves, LiDAR emits laser rays. The distance of points that reflected the emitted laser ray is calculated using the time between laser ray emission and its retrieval as depicted in Eq. (2.1):

$$\Delta d = \frac{c \cdot t}{2} \tag{2.1}$$

where Δd is the distance between LiDAR and the observed point, c is the speed of light and t is the time between signal emission and retrieval. Retrieved points from measurement form a cloud point, that can be further modified, for example, to interpolate missing points or to eliminate outliners. LiDARs possess the same VC possibilities as radars, with increased vulnerability to bad weather conditions, as snowflakes or raindrops interfere with the emitted laser ray. LiDAR systems are generally less expensive than radar systems and offer easier application than radar systems.

The last vision-based methodology utilizes infrared light using infrared cameras. Infrared light possesses a higher wavelength than visible light (760 nm) and lower than microwave light (1 mm). The infrared images received by infrared cameras, sometimes called thermal, display the amount of energy produced by objects in them. This technology is expensive, sensitive to environmental conditions and provides a low quality of acquired images. It is most often used for night vision and battlefield vehicle classification.

2.4 Magnetic Sensor-Based Methods

Magnetic sensors detect distortion of the Earth's magnetic field when a vehicle is passing over it. The setup of magnetic sensors most commonly used for the task of VC is as part of single and dual-loop detectors.

These methodologies use one or two consecutive inductive loops intrusively placed in the road. When a vehicle rides over those loops a ferromagnetic effect is used for vehicle detection. As it is explained in Klein et al. [17]: “The ferromagnetic effect produced by the iron mass of the engine, transmission, or differential does not create a presence or passage indication by the controller. When the heavy ferrous engine enters the inductive loop's detection area, it increases the inductance of the wire loop. This effect occurs because the insertion of any iron core into the field of any inductor reduces the reluctance (i.e., a term that corresponds to the resistance of a magnetic circuit) of the flux path and, therefore, increases the net inductance. However, the peripheral metal of the vehicle has an opposite effect on the inductance due to eddy currents that are produced. The decrease in inductance from the eddy currents more than offsets the increase from the ferrous mass of the engine, and the net effect is an overall reduction in the inductance of the wire loop.“

The advantage of using a double loop detector instead of a single loop detector is, that the speed of the vehicle can be calculated with a higher precision, as it can be calculated using times of vehicle detection.

Magnetic sensors-based methods can be used to detect a vehicle's presence, speed, acceleration and direction.

2.5 Load and Vibration Sensors-Based Methods

The last group of methodologies that can be used for VC is one using sensors utilizing the movement and the mass of the vehicle.

A pneumatic tube is a sensor placed onto the inspected road and attached to it by some fixture (e.g. nails). When a vehicle rides over the pneumatic tube, a rubber pipe is quickly interrupted, producing a burst of air pressure. This burst is detected by the controller located by the side of the road and handled appropriately. Pneumatic tubes can cover several lanes of the road. A single pneumatic tube can be used to count vehicles and measure their weight, based on the strength of air pressure. When several tubes are used, speed and orientation can be measured as well.

Piezoelectric sensors generate a voltage on deformation. They are placed under the observed road and when a vehicle goes on the road above the sensor, its pressure on the road is recorded by a piezoelectric sensor underneath and an electric signal is sent to a control box near to it. Piezoelectric sensors are sensitive to temperature variations. They may be used for vehicle counting and further extended to be used for measuring of vehicle's speed, and orientation when placed in two subsequent places.

A strain gauge is a passive electronic sensor used for measuring the strain of the attached surface using the dependency between conductance and the conductor's geometry. An image of a strain gauge is depicted in Fig. 2.5. The strain gauge sensor may be used for the VC task by embedding it into the observed pavement and detecting the strain response caused by vehicles. This kind of sensor may be even used for a low-grain type of vehicle classification (motorcycle, bus, car etc.) as different vehicles cause different strain responses on the road. Strain gauge sensors may be used for the same measurements as piezoelectric sensors.

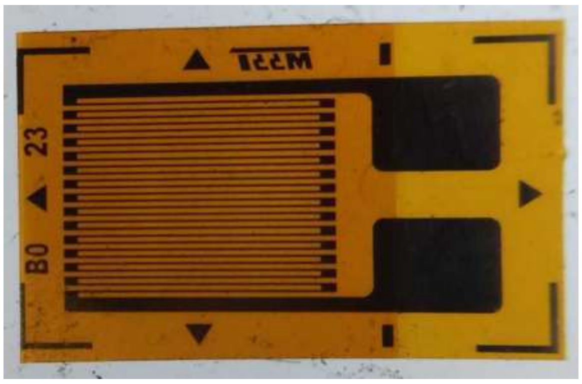


Figure 2.5: Strain gauge. Relative positions of conductive paths change the conductance of the sensor and hence allow for a strain response of attached surface detection. The image is taken from Wikipedia[3].

[3] https://en.wikipedia.org/wiki/Strain_gauge

The last types of sensors addressed in this chapter are seismic sensors. These sensors detect and measure the motion of the observed road when it is shaken by a passing vehicle. Similarly to gauge sensors, different types of vehicles have a different seismic footprints and hence, seismic sensors could be used for vehicle type classification. Their calibration is, however, problematic and so seismic sensors are most commonly used only for vehicle counting.

2.6 Summary

In this chapter, a wide range of approaches to VC was covered. In the first part of this chapter, we have listed those approaches into groups based on the technology they are based on and these groups were further merged according to the intrusion of the road, they are installed on.

The first groups of focus were the ones based on visual contact with the observed vehicle. Those approaches were based on cameras, radars, LiDARs and ultra-red sensors. There, we have discovered, that while the amount of information acquirable in poor conditions (i.e. snow, rain) goes as in Fig. 2.6 a), with a camera being able to capture the biggest amount of information, the price of the sensors is almost the opposite as depicted in Fig. 2.6 b) with the capable infra-red sensor being the most expensive.

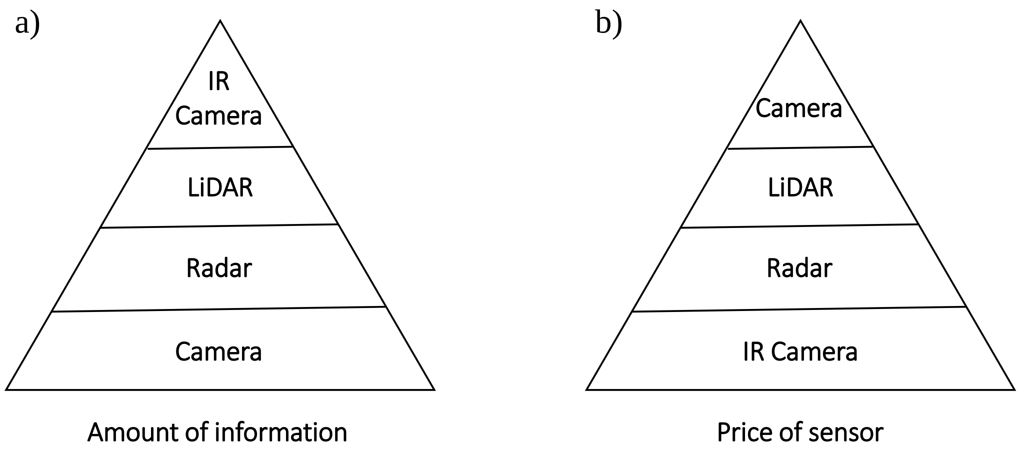


Figure 2.6: Comparison of vision-based sensors: a) the amount of information observable by the sensor in harsh environment b) sensor's price.

Then, we discussed the groups of sensors based on magnetism, load and vibration. All the possible uses of covered sensors are put in Table 2.1. This table is based on Shorkavi et al. [30] and together with information in Section 2.2 renders that for the VMMR task, it is necessary to use an image-based classification using NNs.

Method	**Cnt**	**Speed**	**Acc**	**Dir**	**Wght**	**Axle cfg**	**Type/Model**
Image-based	**yes**	**yes**	**yes**	**yes**	**no**	**no**	**yes**
Radar	yes	yes	yes	yes	no	no	no
LiDAR	yes	yes	yes	yes	no	no	no
Infrared	yes	yes	yes	yes	no	no	no
Inductive Loops	yes	no	yes	yes	no	no	no
Pneumatic	yes	yes	yes	yes	yes	yes	no
Piezoelectric	yes	yes	yes	yes	no	yes	no
Strain Gauge	yes	yes	yes	yes	no	yes	no
Seismic	yes	no	no	no	no	no	no

Table 2.1: Comparison of VC methodologies. *Cnt* stands for vehicle counting, *Speed* for speed measuring, *Acc* for acceleration, *Wght* for weigh measuring, *Dir* for direction recognition, *Axle cfg* for axle configuration and *Type/Model* for recognition of vehicle's type and model.

Chapter 3

Available Vehicle Datasets

As we have concluded in the last chapter, for our task of vehicle manufacturer and model classification, we are going to use NNs. To train a neural network, we need a dataset. In this chapter, we take a closer look at several different vehicle datasets. At the end of the chapter, a comparison of the datasets is performed and is decided whether any of them is usable for our task.

3.1 Stanford Cars Dataset

Stanford Cars dataset [19] is a collection of images and annotations of cars that was compiled by Stanford University in the year 2013. It contains 16 185 images of 196 classes of cars. Photos of the dataset capture the cars from various, mostly frontal, angles with the camera never being too high above the ground as in the case of surveillance cameras. Classes take the form of the car's manufacturer, its model name and the year of production, e.g. *Tesla, Model S, 2012*. Besides the classes, each dataset's entry possesses a bounding box denoting the car's precise position.

The dataset is split into two parts. The first, training part contains 8 144 images, while the second, the testing part, contains 8 041 images.

Stanford Cars dataset is widely used in computer vision research, particularly for tasks such as object recognition and classification. The dataset is public-available for non-commercial purposes. An example of the dataset's entries is displayed in Fig. 3.1.

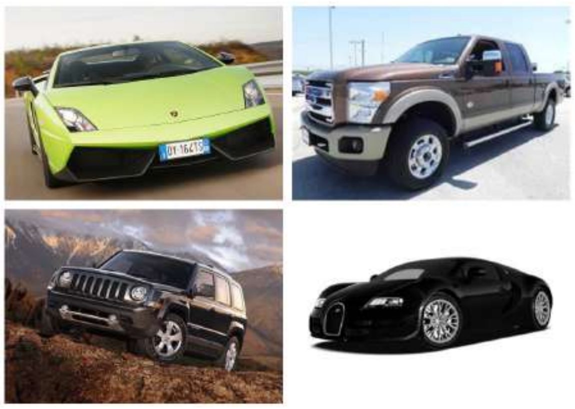


Figure 3.1: Four examples from the Stanford Cars dataset [19].

3.2 Comprehensive Cars

The next dataset is the Comprehensive Cars dataset [37], shortly called CompCars. It is a public-available dataset introduced in the year 2015.

The dataset is composed of two parts - a web-based part and a surveillance-based part. The web-based part contains images downloaded from the Internet (e.g. forums, public websites) and consists of 136 727 images covering 163 mostly Chinese car manufacturers and 1 716 car models from the years 2005 – 2015. Images of cars from this section are taken from various viewpoints. The surveillance-based part is made of 44 481 frontal images of cars obtained using surveillance cameras. This part of the dataset is annotated with bounding boxes of detected cars and their colour.

Entries of the dataset are annotated with the manufacturer's name, model, year of production and viewpoint of the image. Besides, each manufacturer-model-year triplet of the dataset (in the paper called just model), is labelled with five attributes – maximum speed, displacement, number of doors, number of seats and type of car (e.g. SUV or hatchback). These are present for various computer vision tasks and experiments on them are conducted in the paper. Lastly, for each of the previously mentioned triplets, 8 additional images of the car's details are provided. Four exterior parts (i.e. headlight, taillight, fog light and air intake) and four interior parts (i.e. console, steering wheel, dashboard and gear lever). An image of details is presented in Fig. 3.2. The dataset contains 27 618 images of details of cars. An example of the dataset's entries is displayed in Fig. 3.3.

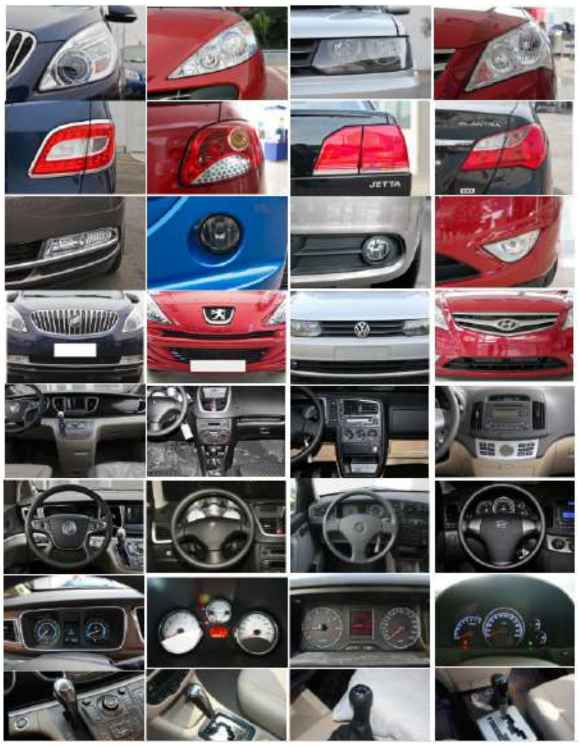


Figure 3.2: The CompCars dataset contains detail images for each model containded. From top to bottom - headlight, taillight, fog light and air intake, console, steering wheel, dashboard and gear lever.

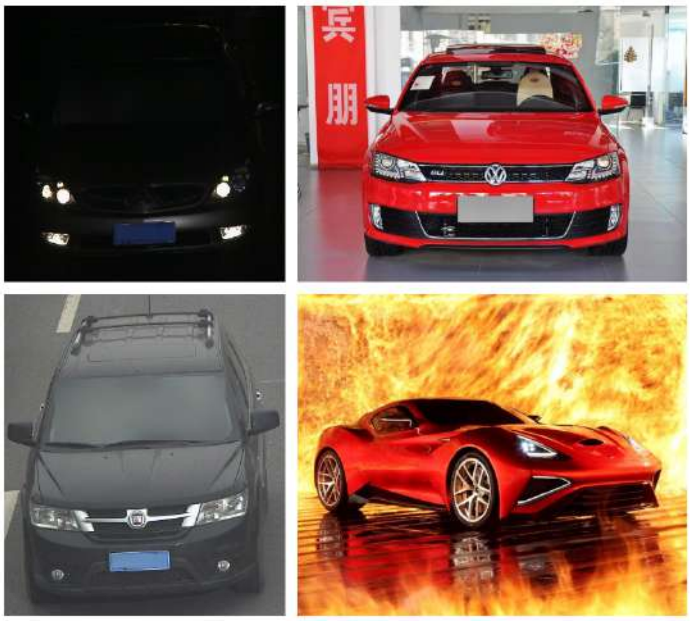


Figure 3.3: Four examples from the Comprehensive Cars dataset [37]. In the left colomn are surveillance-based entries, in the right on web-based ones.

3.3 VMMRdb

The VMMRdb car dataset [33] was presented in 2016 and is publicly available. It is assembled from 291 752 images labelled into 9 170 unique manufacturer-model-production year classes, mostly covering the years 1950 – 2016.

The dataset contains web-searched images and hence it covers a huge variety of acquisition devices, viewpoints of captured cars and light conditions. The main source of images were U.S. online shops and the annotation for each entry was generated automatically based on the title and description of each offer. An example of the dataset's entries is displayed in Fig. 3.4.

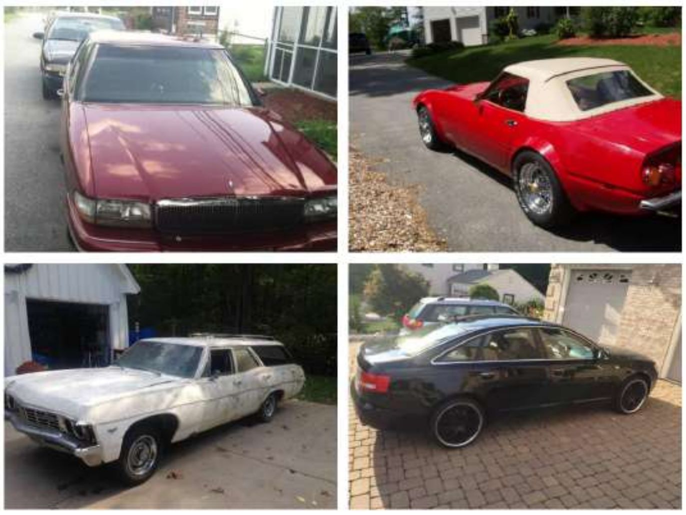


Figure 3.4: Four examples from the VMMRdb dataset [33].

3.4 Vehicle-1M Dataset

This dataset [11] was created in 2018 to provide a sufficient amount of data for the vehicle re-identification task experiments.

It was created from videos captured by surveillance cameras in different places and during different times of day. In the videos, frames containing vehicles were detected, annotated and used in the dataset. The dataset contains 936 051 images of 55 527 vehicles, captured from various angles.

Each image of the dataset is annotated with a vehicle ID label and the vehicle's manufacturer, its model name and the year of production. There are 400 models present in the dataset. Unlike the previous datasets, the Vehicle-1M dataset is not publicly available and for the acquisition, a form[1] must be filled and turned in.

3.5 BoxCars116K

The BoxCars116K [32] dataset from 2019 was introduced as a successor to the original BoxCars21k dataset [31]. It is a publicly available dataset consisting of vehicle images taken from 137 surveillance cameras from different places in Brno, Czechia, mounted near streets with a view of roads. As many cameras were used, it contains various view angles.

The dataset consists of two parts. The first part is the original BoxCars21k dataset improved by corrections of wrongly annotated entries. The second part contains new images taken from the videos obtained by the surveillance cameras on which vehicle detection was performed. Detected vehicles were then annotated by several annotators who assigned them manufacturer, model name, submodel and year of production (as in the original dataset). While the original BoxCars21k contains precisely three images per track, the new part contains an arbitrary number of images per track.

Both parts contain 27 496 vehicles located in 116 286 images. Totally, 45 different manufacturers were registered, forming 693 unique manufacturer-model-submodel-year quaternions. An example of the dataset's entries is displayed in Fig. 3.5.

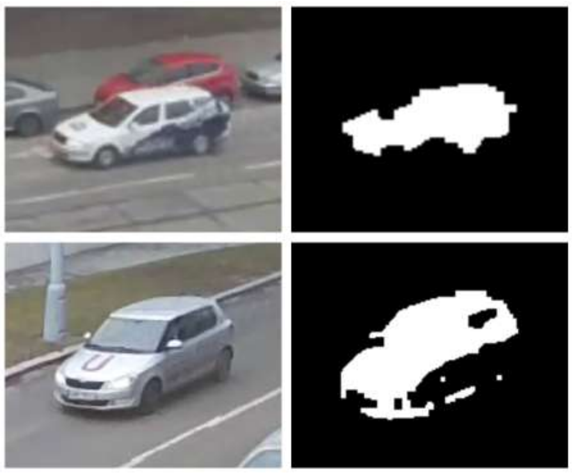


Figure 3.5: Four examples from the BoxCars116K dataset [32]. Every captured image (left column) has its mask of the vehicle (right column).

[1] http://www.nlpr.ia.ac.cn/iva/homepage/jqwang/assets/pdf/Vehicle-1MAGREEMENT.pdf

3.6 Summary

In this chapter, we have seen five vehicle datasets. Based on the information provided for each of them, we can observe that there are two general approaches to data retrieval. The first approach collects data directly from cameras, usually using surveillance cameras. This approach leads to smaller well-labelled datasets. The annotations tend to be human-made. The other approach utilizes the Internet as a source of data. These datasets are generally larger and contain wider variability of their entries. Their annotations are usually created automatically, which may result in incorrect annotations.

In Tables 3.1 and 3.2, several attributes of discussed datasets are shown. Also the dataset collected in chapter 5 is present (under Our Dataset name) to show its properties compared to the other datasets. Only unique manufacurer/model pairs are counted in Our Dataset's **# images** field. The attributes displayed are year of release, number of images in the dataset, number of classes, a country in which vehicles were observed, source of data (as discussed in the previous paragraph), variety of viewpoints from which images are taken and types of vehicles in the datasets (e.g. motorcycles or cars). In the table, insufficient parameters are underscored. It is evident that no dataset discussed in this chapter is sufficient for our task of VMMR on vehicles, including new ones, which would cover vehicles from all over the world. Hence, a new dataset, that would allow for such a task had to be created. It had to be created using the Internet available data as the sheer amount of vehicles would not be coverable by a manual acquisition.

Name	**Year**	**# images**	**# classes**	**Vehicle country**
Our Dataset	2022	6 124 103	9 064	Various
Stanford Cars	<u>2013</u>	<u>16 185</u>	<u>196</u>	<u>U.S.</u>
CompCars	<u>2015</u>	181 208	1 716+	<u>China</u>
VMMRdb	<u>2016</u>	291 752	9 170	<u>U.S.</u>
Vehicle-1M	2018	936 051	<u>400</u>	<u>China</u>
BoxCars116K	2019	116 286	<u>639</u>	<u>Czechia</u>

Table 3.1: Information on vehicle datasets. Underlined values denote insufficient properties. Insufficient properties are: too old datasets (5 years or older), too few images/classes and focus on vehicles from one country only

Name	**Data source**	**Viewpoints**	**Vehicles**
Our Dataset	Internet	Various	cars, buses, trucks, motorbikes
Stanford Cars	Unknown	Most is frontal	cars
CompCars	Internet and surveillance cam	Internet part unknown, surveillance part frontal	cars
VMMRdb	Internet	Various, online shops-like	cars
Vehicle-1M	Surveillance cams	Surveillance cams-like	Unknown
BoxCars116K	Surveillance cams	Surveillance cams-like	cars

Table 3.2: Information on vehicle datasets. Underlined values denote insufficient properties. Insufficient properties are: unknown source of images, invariant viewpoints and inclusion of only cars in the dataset and hence omitting other vehicles like motorcycles.

Chapter 4

Image Classification Using Neural Networks

To perform the VMMR task with satisfying results, we have to choose the correct methodology for NN's application. As Convolutional neural networks (CNNs) were already used for fine-grained classification in several vehicle-based tasks (Tafazzoli et al. [33], Yang et al. [37] and Sochor et al. [32]) , it is reasonable to think that their usage for our task can lead to good results. In this chapter, we discuss the principles of CNNs and concepts of embeddings, self-attention and transformers.

4.1 Convolutional Neural Networks

Convolutional neural networks, originally proposed and implemented by LeCun et al. [20], are a type of NNs containing convolutional layers. They are suitable for processing grid-like data of any dimensionality where related information pieces are located close to each other. Such data are mostly images in 2D, sound in 1D or medical volumetric data in 3D.

Convolutional layers are usually composed of 3×3, 5×5 or 7×7 learnable kernels performing convolution to extract features from the input data. The feature extraction using convolutional layers is translation invariant. After the feature extraction, the features form a feature map which is the layer's output. The kernels are applied to the input data in a sliding window fashion where the distance between appliances of kernels is called a stride. Convolution for 2D is defined in Eq. (4.1):

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n). \tag{4.1}$$

There, I is the input image, K is a convolutional kernel, i, j are coordinates of a pixel in I and m, n are sizes of a kernel. A graphical representation of 2D convolution is depicted in Fig. 4.1.

After convolutional layers, pooling layers often take place to merge several values from the feature map into one, reducing its size and hence the number of parameters needed for further layers. The most commonly used size of a convolutional filter is a 2×2, reducing the input data by 75%. There are two commonly used policies for merging. The first one is called max-pooling, outputting the highest number from the filter. The second one is called average pooling. This policy outputs the average of the number within the filter. Pooling layers are applied to the input data in a sliding window fashion. A graphical representation

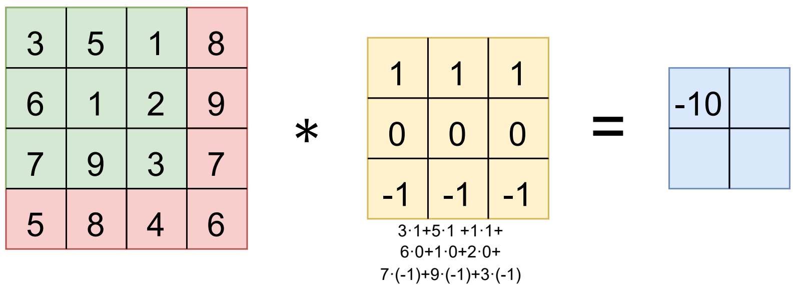


Figure 4.1: A graphical representation of 2D convolution using a 3 × 3 kernel.

of a pooling-layer is shown in Fig. 4.2. A general topology of CNNs for classification can

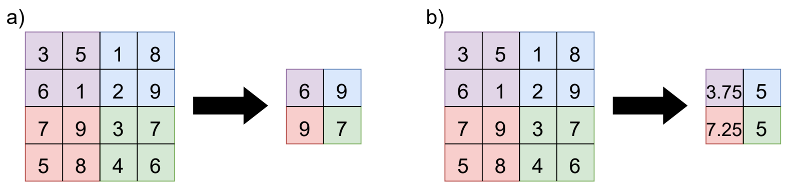


Figure 4.2: A graphical representation of pooling operation. In a) 2 × 2 max-pooling, b) 2 × 2 average pooling.

be divided into two parts. In the first part, convolutional layers are placed together with other layer types (e.g. pooling layers) to extract features from the input image. The second part contains fully-connected layers to use the previously extracted feature map as a feature vector and perform the classification based on it. An example of such network architecture is the ImageNet displayed in Fig. 4.3.

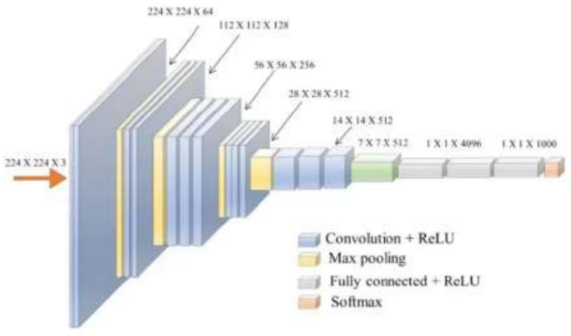


Figure 4.3: An ImageNet architecture of CNN. Convolutional layers are located in the first part while the second part consists of fully-connected layers. Image taken from Sharma [29].

4.2 Embeddings

As defined by Koershen [18], "An embedding is a mapping of a discrete - categorical - variable to a vector of continuous numbers". The mapping may be done in several ways.

One method of mapping the variable, not using NNs is called one-hot encoding. This approach encodes the categorical value into a vector $v \in B^N$, where $B = \{0, 1\}$ and N is the number of values, the variable may acquire. All values of the vector v are zero-valued, except one, which is set to one. The position of this set value is different for each value and hence every category is mapped into a unique vector.

This method has two major issues. The first issue is the length of the embedding. As stated above, the length of the embedding corresponds to the length of the list of possible values and for long lists (e.g., list of vehicle models, movies, songs etc.) the dimensionality of embeddings becomes unmanageable. The other issue is a lack of any information contained within the embedding, except an index of the element. Embeddings hence cannot be anyhow compared.

Both of these drawbacks are absent when using the second approach to embedding retrieval. This way, the embeddings are obtained as weights of a NN supervisory taught to process the categorical values (encoded e.g., using one-hot encoding) to retrieve results based on deeper knowledge of the input values (e.g., similarity or genre). This way, the NN is trained to 'understand' its inputs and to create embeddings that would contain semantic information. These embeddings are retrieved from the NN as weights of edges comming into the embedding layer. The process of embedding retrieval is depicted in Fig. 4.4.

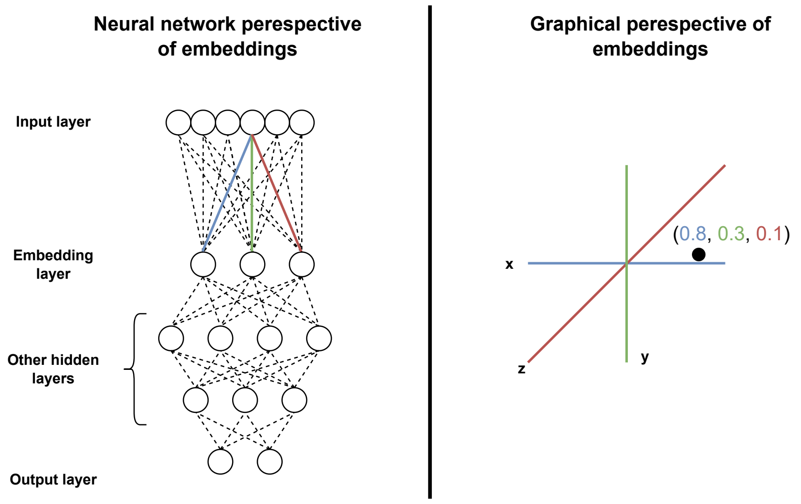

Neural network perespective of embeddings
Input layer
Embedding layer
Other hidden layers
Output layer
Graphical perespective of embeddings
(0.8, 0.3, 0.1)
x
y
z

Figure 4.4: From one-hot encoding, embeddings acquired from NN are acquired as weights of edges connecting embedding layer to a corresponding input layer element.

As the embeddings obtained from NN contain semantic information, this leads to several possible applications.

One such is a embedding vectors comparison. The comparison is based on a similarity coming from the task the NN was trained for. To measure the similarity, a cosine similarity or a dot product may be used.

Another application is their usage as an input into NNs. As they usually have lower dimensionality than the original encodings of the categorical values, they can be more easily processed by new NNs. Also, as they possess the semantics information, they are often suitable for a wider spectrum of tasks than the original encodings.

The last usage mentioned here is visualization. As similar values are close to each other, different groups of values and their relations to other groups can be observed, when the dimensionality of the embeddings is reduced into 2 or 3 dimensions. This can be achieved using a t-SNE algorithm. The t-SNE is a method for visualizing high-dimensional data. An example of such visualization of book genre is shown in Fig. 4.5.

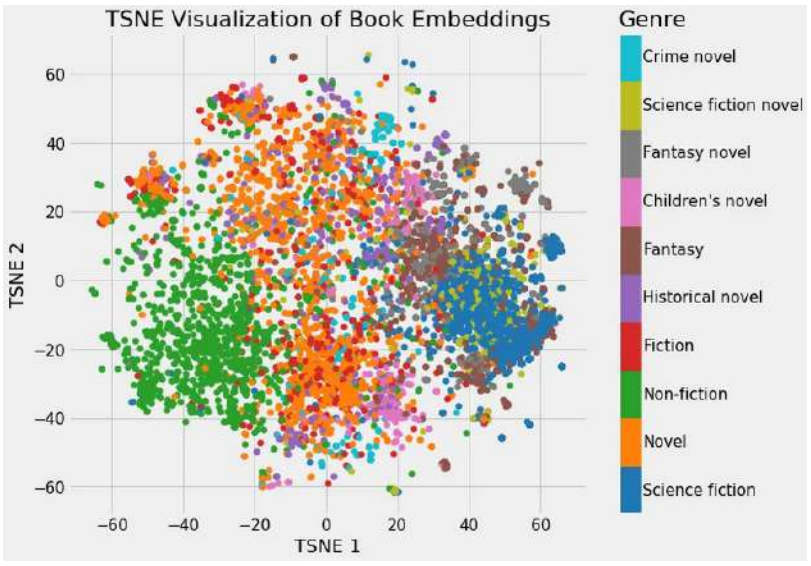


Figure 4.5: Usage of t-SNE algorithm on embeddings creates clusters of classes. Image taken from Koershen [18].

4.3 Attention

Attention, in the context of NNs, is a mechanism providing contextual information on processed input sequences, allowing the network to 'focus' on related pieces of data. This results in more effective usage of computational power.

Attention was originally as described by Bahdanau et al. [2], used in natural language processing to improve translations between languages as translators often lost context of the first part of the processed sentence when translating the later part of it. This was done by building a matrix of context between the source text and the output text (so-called cross-attention). When applied to computer vision, it is usually searched for a context within the sequences (feature maps/feature vectors) and hence the same sequence is used twice in the matrix (so called self-attention). In other words, the sequence is the input and the goal is to compute weights of context between every pair of its features. Using this, the original sequence may be updated to contain the context.

There are many ways to incorporate attention mechanism into a network. Three popular existing attention modules are Multi-Head Attention [35], Bottleneck Attention Module [26] and Convolutional Block Attention Module [36].

The multi-head attention, often called just self-attention layer, was designed as a part of a transformer network and is also used in some of its variations for image processing (e.g. Vision Transformers [9]). It is based on scaled dot-product attention, which calculates attention values for two sets of sequences by converting the input sequences into so-called $Value$, Key and $Query$ vectors. The principle of this metric will not be further discussed in this thesis as it exceeds its scope.

Using only one scaled dot-product block, it would only be possible for a feature (as a unit of input sequence), to pay attention to one other feature, which is not desirable.

Hence, a multi-headed attention block is used. It contains multiple scaled dot-product attention layers that work in parallel and do not share any weights. Their number is a hyperparameter of the model. The results of the parallel layers are concatenated and fed into a fully connected layer. Graphical visualization of the multi-head attention layer is shown in Fig. 4.6.

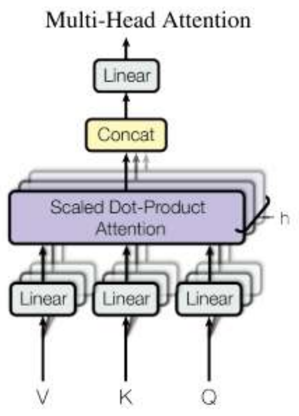


Figure 4.6: Illustration of a multi-head attention layer. Input vectors $Value$, Key, $Query$ calculated from the analyzed sequence. The layer is composed of several parallel scale dot-product attentions. Image taken from Vaswani et al. [35].

4.4 Transformers

Another state-of-the-art principle used in NNs are transformers models. The original transformer described in A. Vaswani et al. [35], is a network relying solely on attention, eliminating the usage of expensive convolutional or recurrent layers while maintaining the performance.

While the original transformer network was designed for natural language processing, several other variations of the transformer were created for the task of image processing. Examples of these variations are Vision Transformer [9], Image Trasnformer [27] or DETR [3].

Vision transformer (ViT) was proposed to follow the architecture of the original transformer as much as possible. A scheme of the networks is shown in Fig. 4.7.

The inputs are images $x \in \mathbb{R}^{H \times W \times C}$ that are split into square patches of size $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where H is height of the image, W is its width and C number of its channels. P is a length of patch and N is the total number of patches ($N = HW/P^2$).

Firstly, the patches are converted into D dimensional patch embeddings, by feeding them in flattened form into a trainable linear projector.

To the beginning of the sequence of embeddings, a learnable embedding is added. It is called $z_0^0 = x_{class}$, where upper index indicates position within sequence and the bottom one the number of transformer encoder layers it was processed by, 0 meaning none, L meaning every. The state of this embedding at the output of the transformer encoder (z_L^0) then serves as feature vector y which is used for classification by prediction head (Eq. 4.5).

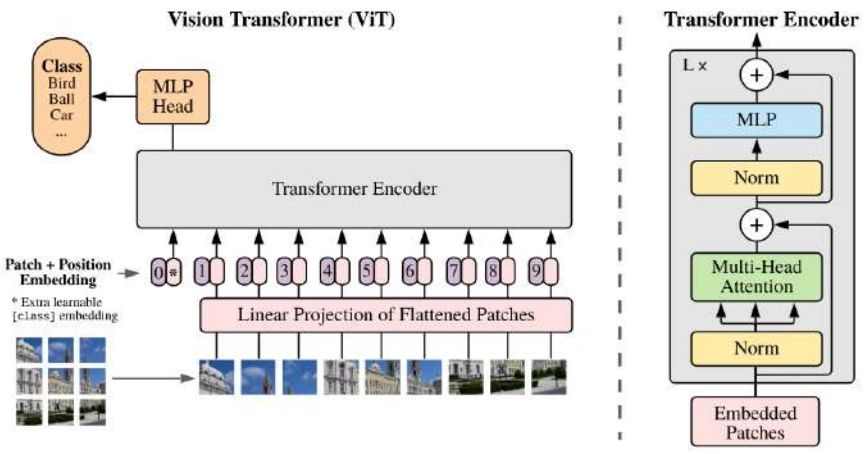


Figure 4.7: A scheme of ViT network. Image is parsed into patches and projected into patch embeddings. To them, x_{class} embending is concatenated and positioning embedding is added. These sequences are then processed by layers of Transformer Encoder. Image taken from Dosovitskiy et al. [9].

Before the initial sequence is fed into the transformer encoder, positional embeddings are added to the patch embeddings to keep information on locality. The resulting sequence, described in Eq. 4.2 is used as input of the encoder.

The transformer encoder consists of two alternating parts – multi-head attention (MHA, Eq. 4.3) and MLP blocks (MLP, Eq. 4.4). The encoder consists of L pairs of MHA and MLP block. Layernorm (LN) is performed before every block and residual connection after every block.

$$\mathbf{z}_0 = [x_{class}; x_p^1\mathbf{E}; x_p^2\mathbf{E}; \ldots; x_p^N\mathbf{E};]\mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \tag{4.2}$$

$$\mathbf{z}'_l = \mathrm{MHA}(\mathrm{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}, \qquad l = 1\ldots L \tag{4.3}$$

$$\mathbf{z}_l = \mathrm{MLP}(\mathrm{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l, \qquad l = 1\ldots L \tag{4.4}$$

$$y = \mathrm{LN}(\mathbf{z}_L^0) \tag{4.5}$$

As noted in Dosovitskiy et al. [9], when ViT is trained on mid-sized datasets (up to millions of entries), the results do not reach the performance of ResNet of comparable size. This happens due to the absence of convolutional layers, which offer translation invariant feature extraction and locality. These shortages are compensated when training on larger datasets (14M – 300M images). In these scenarios, ViTs are able to outperform image recognition benchmarks.

The ViT network is standardly trained on a large dataset and afterwards fine-tuned for smaller tasks. This is done by the replacement of the prediction head, consisting of MLP with one hidden block, with a new one, consisting of untrained fully-connected layer outputting K values, where K is the number of classes.

4.5 Summary

In this chapter, we have decided to use convolutional neural networks for our VMMR task. We have discussed their mechanism, what layers they use, and the standard topology of CNN. Next, we examined the attention mechanism in NNs and looked at the principle of multi-head attention. Finally, we have seen transformers, state-of-the-art networks relying only on attention mechanism and discussed how vision transformer translates the original model from natural language processing into computer vision.

Chapter 5

Dataset Acquisition

At the end of Chapter 3, we found that no available vehicle dataset is suitable for our VMMR task and therefore we will have to create one of our own. Besides, we have concluded that this dataset has to be based on data from the Internet, as we want to cover the widest range of vehicles possible.

For the purpose of the VMMR task, it is sufficient to contain only manufacturers, models and years of production of the vehicles in the dataset. However, as the dataset may be further used for different tasks, additional information should not be automatically omitted.

In this chapter, the methodology of this dataset creation is discussed.

5.1 The Dataset Acquisition Pipeline

When creating the new dataset, we followed pipeline as depicted in Fig. 5.1. The whole pipeline can be split into three phases:

1. The first phase (red) covers the creation of a list of vehicles which are to be contained in the dataset. As there was no list of vehicles, manufacturers and models to be covered in the beginning, one had to be made.

2. The second phase (yellow) focuses on obtaining of entries that are contained in the dataset. Using the lists from the previous phase, queries were created and passed to an internet search engine to obtain desired images of vehicles. The downloaded images were afterwards examined using NNs to ensure that a vehicle is present in them and to get its bounding boxes.

3. The third phase (green) then finishes the pipeline by splitting the dataset's entry list. This was done in order to know which entries are usable for training, validating and testing.

In the following sections, every step of the pipeline is explained in detail. The pipeline was implemented in Python 3.10. Additional modules and packages used were Python's **re** module for work with regular expressions, **request** for downloading HTML websites, **pandas** for local database operations, **Beautiful Soup** for processing downloaded HTML files, **duckduckgo_search** for interaction with the Duck Duck Go search engine and **imhdr** for image format inspection.

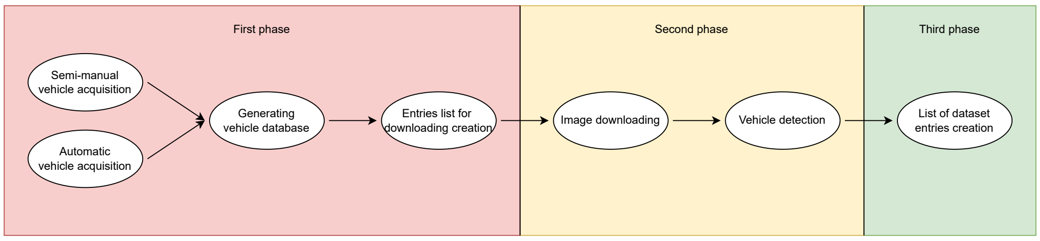

First phase
Second phase
Third phase
Semi-manual vehicle acquisition
Automatic vehicle acquisition
Generating vehicle database
Entries list for downloading creation
Image downloading
Vehicle detection
List of dataset entries creation

Figure 5.1: A scheme of dataset acquisition pipeline.

5.2 Semi-Manual Vehicle Acquisition

The first attempt to acquire a list of vehicles for our dataset was based on a semi-manual approach, which was built on the idea that it would be manually chosen from which webpage information should be extracted and how to obtain the information.

This approach used regular expressions to communicate which data to look for in the code of the downloaded webpages. As it only collected information on the vehicle's manufacturer, model and year it was produced in, it was possible to use named groups in the regular expression to assign the correct value to the corresponding attribute in the matched pattern. Such regular expression could look like:`<li>(?P<man>[\S]*) (?P<mod>.*)</li>`, where `man` and `mod` are named groups for manufacturer and model. If the information on the manufacturer or year of production was not available on the website, it was possible to set one statically for every vehicle found on it.

Once the analysis of the searched webpage was done, a proposal of entries found was available in a file generated by the script. There, it was possible to make modifications before the entries were permanently added to the database of all vehicles.

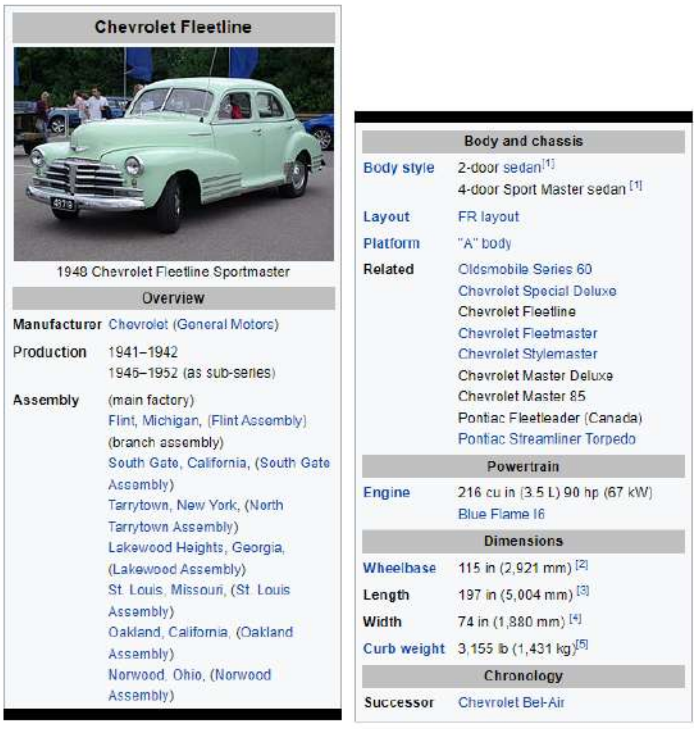
The problem with this approach became apparent when data were scrapped from Wikipedia - one webpage may contain several structures to store data (e.g. in the case of Audi[1] the webpage stores information on vehicles in 4 kinds of table and a list). This way, the correction of proposed results becomes unreasonably time demanding and building a huge dataset of vehicles would not be possible. Hence, in order to obtain a huge amount of data, a fully-automatic approach to data collection had to be applied, since omission of Wikipedia would lead to a reduction of obtained data.

5.3 Automatic Vehicle Acquisition

The other approach used automatic processing of web pages without any intervention from human to obtain the largest possible list of vehicles. For this purpose a high-level web scrapping was used where data were pulled out of HTML files automatically.

This solution is based on data retrieval purely from Wikipedia, specifically from tables appearing in wiki pages of models (see Fig. 5.2). In addition to the minimal viable information needed for the VMMR task, these tables also often offer other information, that could be used for different tasks. For this reason, not only manufacturer, model and years of production (if available) were saved for each entry. Entries also contain URLs of manufacturer and model and, mostly in case of cars, information on generations of the model and its body styles. Each table converts into one entry as discussed further in the chapter.

[1] https://en.wikipedia.org/wiki/List_of_Audi_vehicles



Chevrolet Fleetline

1948 Chevrolet Fleetline Sportmaster

Overview	
Manufacturer	Chevrolet (General Motors)
Production	1941–1942 1946–1952 (as sub-series)
Assembly	(main factory) Flint, Michigan, (Flint Assembly) (branch assembly) South Gate, California, (South Gate Assembly) Tarrytown, New York, (North Tarrytown Assembly) Lakewood Heights, Georgia, (Lakewood Assembly) St. Louis, Missouri, (St. Louis Assembly) Oakland, California, (Oakland Assembly) Norwood, Ohio, (Norwood Assembly)

Body and chassis	
Body style	2-door sedan[1] 4-door Sport Master sedan [1]
Layout	FR layout
Platform	"A" body
Related	Oldsmobile Series 60 Chevrolet Special Deluxe Chevrolet Fleetline Chevrolet Fleetmaster Chevrolet Stylemaster Chevrolet Master Deluxe Chevrolet Master 85 Pontiac Fleetleader (Canada) Pontiac Streamliner Torpedo
Powertrain	
Engine	216 cu in (3.5 L) 90 hp (67 kW) Blue Flame I6
Dimensions	
Wheelbase	115 in (2,921 mm) [2]
Length	197 in (5,004 mm) [3]
Width	74 in (1,880 mm) [4]
Curb weight	3,155 lb (1,431 kg)[5]
Chronology	
Successor	Chevrolet Bel-Air

Figure 5.2: An example of information table on vehicle model found on Wikipedia. Image was split in the place of black rectangle.

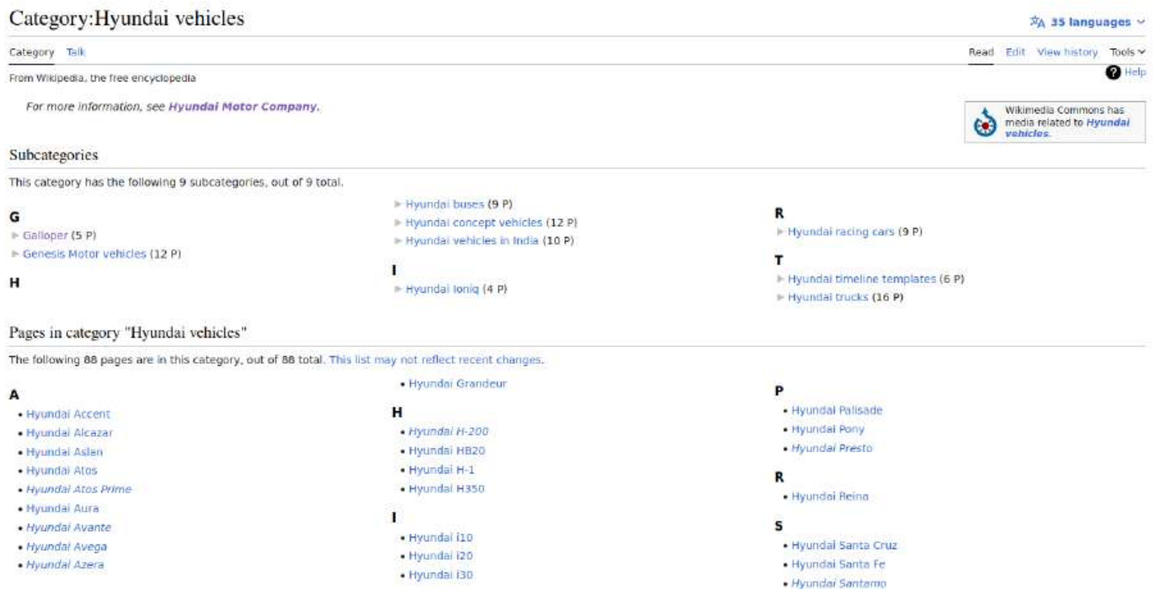
This focus on tables at wiki pages of models is possible due to the existence of pages aggregating wiki pages of manufacturers. These contain lists of wiki pages of models they manufactured. The manufacturer-aggregating pages are always focused on a certain type of vehicle. Eight vehicle types are covered this way – military vehicles, rolling stock, sailboats, aircraft, buses, motorcycles, trucks and cars. We used all of them for scrapping information on vehicles.

The scrapping consisted of 2 phases:

In the first phase, the goal was to obtain a list of manufacturers and URLs of their wiki pages. Upon manufacturer's name retrieval, it was passed to a function stripping it of supportive parts on names (e.g. changing name *Buses manufactured by Karosa* to *Karosa* or *Volvo vehicles* to *Volvo*). Also, as some of the aggregation-pages contained too many manufacturers to fit on a single page, they were spanned across several pages and were accessible by a `next page` button. The function collecting names of manufacturers had to be able to detect this situation and recursively scrape the other pages as well. At the end of this phase, lists of manufacturers and URLs of their wiki pages were available for all the vehicle types.

In the second phase, each manufacturer's wiki page was processed. There, it was observed whether references to subcategories are present (as in Fig. 5.3) as they possess other lists of vehicles. In case of their presence, the function processing wiki pages of manufacturers was recursively called on them. Then, each model wiki page available was visited and scraped.

On the wiki page of the model, it was tested whether it was already scrapped (e.g., the model was listed under several manufacturers, or several links led to the same page). This


Category:Hyundai vehicles

35 languages

Category Talk

Read Edit View history Tools

Help

From Wikipedia, the free encyclopedia

For more information, see ***Hyundai Motor Company.***

Wikimedia Commons has media related to ***Hyundai vehicles.***

Subcategories

This category has the following 9 subcategories, out of 9 total.

G
- Galloper (5 P)
- Genesis Motor vehicles (12 P)

H
- Hyundai buses (9 P)
- Hyundai concept vehicles (12 P)
- Hyundai vehicles in India (10 P)

I
- Hyundai Ioniq (4 P)

R
- Hyundai racing cars (9 P)

T
- Hyundai timeline templates (6 P)
- Hyundai trucks (16 P)

Pages in category "Hyundai vehicles"

The following 88 pages are in this category, out of 88 total. This list may not reflect recent changes.

A
- Hyundai Accent
- Hyundai Alcazar
- Hyundai Aslan
- Hyundai Atos
- *Hyundai Atos Prime*
- Hyundai Aura
- *Hyundai Avante*
- *Hyundai Avega*
- *Hyundai Azera*

- Hyundai Grandeur

H
- *Hyundai H-200*
- Hyundai HB20
- Hyundai H-1
- Hyundai H350

I
- Hyundai i10
- Hyundai i20
- Hyundai i30

P
- Hyundai Palisade
- Hyundai Pony
- *Hyundai Presto*

R
- Hyundai Reina

S
- Hyundai Santa Cruz
- Hyundai Santa Fe
- *Hyundai Santamo*

Figure 5.3: An example of a wiki page of a vehicle manufacturer. Beside a list of models manufactured it may also contain a list of subcategories leading to other lists of models.

had to be decided using MD5 coding of the page's content as Wikipedia did not change URL address when redirecting at the time of data scraping (see Volvo 200 series[2] and Volvo 240[3]) and the library used for HTML downloading was not able to detect such situations. If the page was visited for the first time, its MD5 code was inserted into a list of visited pages and it was searched for tables with information.

The processing of tables found began with their headers. It was used either as name of the model or as name of the model's generation, based on the order of the table within the wiki page as the first tables on the wiki pages contain information on models while the next ones on generations of the model (see Chevrolet Bel Air[4]). From the header's text, the name of the manufacturer was removed if present. If the header was used as a name of generation, its model name was inherited from the first table on the wiki page. The entry based on the first table of the page hence never had information on the generation as it covered the model. Next, the entry's other attributes were determined based on the table:

- Manufacturer – If it was not available in the table, the name of the manufacturer on which's wiki page the processed model was located was used.
- Years of production - The scrapping script had to handle various formats (e.g., 1988–89 or 2003 to present). If production years were not available, the field was left empty.
- Body types – For non-car vehicles, it was the type of vehicle. For cars, 41 body types were registered. If none of them was present, the field was left empty. If the car was manufactured in several body types, they were concatenated.

For every manufacturer from the aggregation pages, one CSV file was created and entries that were acquired from the manufacturer's list of models were saved into it. The

[2]https://en.wikipedia.org/wiki/Volvo_200_Series
[3]https://en.wikipedia.org/wiki/Volvo_240
[4]https://en.wikipedia.org/wiki/Chevrolet_Bel_Air

manufacturers files were placed into folders based on the vehicle type. While military vehicles, rolling stock, sailboats and aircraft have their individual folder, buses, motorcycles, trucks and cars share their folder as they are all road vehicles.

The scheme of entries within the CSV files is `manufacturer, model, generation, years of production, body types, wiki of model, wiki of manufacturer`. Entries example is

```
ford, fusion, , 2005-2020, sedan, wiki1, wiki2
ford, fusion, first generation, 2005-2012, , wiki1, wiki2
ford, fusion, second generation, 2012-2020, , wiki1, wiki2
ford, fusion hybrid, , 2008-2020, sedan, wiki1, wiki2
ford, fusion hybrid, first generation, 2008-2012, , wiki1, wiki2
ford, fusion hybrid, second generation, 2012-2020, , wiki1, wiki2
ford, fusion hybrid, fusion energi, 2013-2020, sedan, wiki1, wiki2
```

with `wiki1` being substitute for URLs of models pages (`/wiki/Ford_Fusion_(Americas)` and `/wiki/Ford_Fusion_Hybrid`) and `wiki2` being a substitute for URL of manufacturer (`/wiki/Category:Ford_Mondeo`).

The number of unique extracted entries is shown in Table 5.1. It is important to note that not all entries of aircraft are valid since there exist wiki pages of aircraft crashes that contain tables similar to the ones the scrapper script aimed for (see Tajik National Guard Mi-8 crash[5]). These were, however, also added into the list of models by the script.

Vehicle type	**# of manufacturer pages**	**# of entries**
Road vehicles	364	14 080
Aircraft	1 320	10 957
Military vehicles	10	90
Ships	95	4 337
Rolling Stock	20	1 023
Total	**1 809**	**30 489**

Table 5.1: Numbers of manufacturers visited per vehicle types and number of entries scraped.

Overall, the result of the automatic acquisition of vehicles list was a huge success, as not only a huge amount of road vehicles was covered, which would not be possible using the semi-automatic approach from the previous section, but also additional information was retrieved and other types of vehicles were covered.

5.4 Generating Vehicles Database

Once the information on vehicle data was extracted, it was scattered across several folders and hundreds of files. For better manipulation, it was found suitable to merge the collected data into just a few files with a format that would restrict the redundancy of information.

As the focus of the thesis is a VMMR task on road vehicles, we found it reasonable to split the data into files by type of vehicle – aircraft, military, ships, rail stock and road vehicles. YAML was chosen as a sufficient format as it does not repeat information and

[5]https://en.wikipedia.org/wiki/2010_Tajik_National_Guard_Mi-8_crash

remains human readable unlike many other formats (e.g., JSON). The scheme of YAML record for a single manufacturer was designed as depicted in Fig. 5.4. It can be seen that the whole file is made of a list of manufacturers. Those contain a list of countries, they manufacture in[6], their wiki page and a list of models they manufacture. A single model consists of a list of generations of the model ('None' is assigned to entries without any known generation). Finally, a generation consists of body types it was created in (e.g. sedan, pick-up, hatchback), its wiki page and a list of years in which it was produced.

```
- manufacturer_name:
  country:
  - country_name
  wiki: manufacturer_url
  models:
  - model_name:
    - generation_name:
      bodyClass: body_class
      wiki: model_wiki
      years:
      - from-to
```

Figure 5.4: A scheme of YAML entry of a manufacturer. Variables *manufacturer_name* stands for the name of manufacturer, *country_name* for the name of country it manufactures in, *manufacturer_url* for address of its wiki page, *model_name* for the name of model, *generation_name* for the name of the model's generation, *body_class* for the type of body it was manufactured in case of cars, for other vehicles this field contains a type of vehicle. The *model_wiki* stands for the address of the model's wiki page and *from-to* stands for years between which the generation of the model was made.

As names of manufacturers for the YAML files were taken from the CSV entries, where data from the information tables were contained, the list of manufacturers changed, compared to Table 5.1 as the information tables sometimes listed different manufacturers than the ones, which wiki page led to the model page. The amounts of manufacturers, models and generations in YAML files are in Table 5.2. To be counted, generation had to be different from 'None' (as discussed above). Between types of vehicles in the road vehicles group, the number of models is distributed as follows: cars – 7 000, motorcycles – 1 173, trucks – 505 and buses – 486.

Although this step was correct as it utilized information directly from pages of models, it also introduced a new problem: some of the different names of manufacturers address the same manufacturer (e.g. *ford motor company* and *ford*). This situation emerged since on Wikipedia, several models by the same manufacturer used different names.

This problem was addresed by manually creating a CSV table file, with scheme of `name, merge`. Here every manufacturer present in Road vehicles category would appear under its name present in the YAML in the `name` column and in the `merge` column, its merged name would appear. If merging occurred, this merged name would be in capital letters. If not,

[6]https://en.wikipedia.org/wiki/List_of_car_brands

Vehicle type	**# of manufacturers**	**# of models**	**# of generations**
Road vehicles	757	9 064	4 671
Aircraft	1 828	10 956	18
Military vehicles	19	92	0
Ships	95	4 337	2
Rolling Stock	47	955	85
Total	**2 746**	**25 404**	**4 776**

Table 5.2: Numbers of manufacturers, models and generations contained in YAML files per vehicle type.

this merged name would be the same as the original name. An example of entries in this file is:

```
alvis, alvis
a.l.f.a., ALFA ROMEO
alfa, ALFA ROMEO
alfa romeo, ALFA ROMEO
```

It can be seen that while the *alvis* manufactuter was not merged with any other manufacturer, the *a.l.f.a.*, *alfa* and *alfa romeo* were united under one manufacturer (as a.l.f.a. and alfa refer to the original name of the manufacturer).

Except for the historic names of the manufacturers, other most common reasons for merging were different names (e.g. *Volvo* and *Volvo cars*), manufacturer being a racing team under a different manufacturer (e.g. *M-Sport* is merged to *FORD*), and, mostly for Asian manufacturers, manufacturers being a marque of a bigger manufacturer (e.g. *Senova* is merged to *BAIC*).

This way, the original 757 road vehicles manufacturers were reduced to 399 entries. No changes were however applied to data themselves as these merges only appear in the CSV table from where they can be applied during experiments and can be changed if needed.

This part of the pipeline reduced the number of files to work with from 1 809 to 5 and changed the format of saved data from CSV to YAML to improve readability and lower the amount of redundancy.

5.5 Entries List for Downloading Creation

The last step before image acquisition was the creation of lists of models to be followed during the process of dataset downloading. These lists were obtained by reduction of information present in road vehicles YAML file and its serialization.

Only road vehicles file was processed because they are the main focus of this thesis. The serialization was done into files in CSV format. There were created 4 files, one for each type of vehicle in road vehicles – buses, cars, motorcycles and trucks. This was done in order to logically split the dataset and to be able to perform downloading in parallel on several machines.

Each entry of the CSV file consisted of name of the manufacturer, model name and year of production. There was an entry created for each year in years interval in the YAML file. Hence, a model produced between the years 1989 – 1998 would make for 10 entries. If no value was available for the model, *anyYear* value was set. If model consisted of several generations, the first was used as it usually described the whole model. The sizes of the lists are shown in Table 5.3.

Vehicle type	**# of entries**
Cars	53 845
Buses	5 225
Motorcycle	9 292
Trucks	6 830
Total	**75 192**

Table 5.3: Numbers of entries to be downloaded per type of road vehicles.

5.6 Image Downloading

After obtaining lists of vehicles to download, the next step was the automatic download of images which would make the backbone of our dataset.

Firstly, the means of image acquisition had to be chosen. A DuckDuckGo was found suitable as a search engine because it does not personalize the results of the search. Two approaches were tested for its utilization. The first approach used a search of images to retrieve their metadata based on a query. The metadata were then tested on their suitability (as discussed below) and the satisfactory images were downloaded serially. This approach will be addressed as `ddg_serial` from now on. In the second approach all images were downloaded in parallel and tested afterwards. This approach will be addressed as `ddg_parallel` from now on. Durations of these approaches for the acquisition of the first 100 images obtained from a query are displayed in Table 5.4. As it can be seen, the `ddg_parallel`, despite downloading images that are not used, outperforms the `ddg_serial` approach more than 4 times and hence was selected to be used.

	ddg_serial	**ddg_parallel**
T1	70.68s	14.35s
T2	69.22s	15.43s
T3	66.74s	17.08s
Average	**68.88s**	**15.62s**
Relative	**1**	**0.23**

Table 5.4: A comparison of two image retrieval approaches on acquisition of 100 images in three runs.

For each entry of the lists generated in the previous section a query for the search engine was generated as a concatenation of vehicle, manufacturer, model and year (e.g., *car chevrolet bel air 1963*). If an entry contained *anyYear* value of year, this part of the query was omitted. For every entry, up to 100 images were downloaded and processed.

The validation process consisted of a size check and a format check. The size check controlled whether the width and height of image are above 256 pixels. This was done to ensure that images will be usable for the VMMR task. A format check then subsequently inspected the input image to determine its format. Thirteeen formats[7] were recognized.

[7] https://docs.python.org/3/library/imghdr.html

This control was performed to discard images of unknown or corrupted formats and to unify the names of the formats so that the same formats are expressed uniformly (e.g., files of JPEG format may have extensions `.jpg`, `.jpeg`, `.JPEG` etc.). Insufficient images weren't used any further.

The images that passed the process were renamed to subsequent numbers, with each set of images downloaded from a single query starting from 0, and ending at $N-1$, where N is the number of images that passed the testing. The images were then moved into the hierarchical structure of the dataset in format: `vehicle/manufacturer/model/year` as displayed further, in Fig. 5.5. To prevent conflicts with the filesystem, slashes and ending dots were removed from names of manufacturers and models.

Besides images saving, for each image a file containing metadata was generated. These files holds the same name as the image and have a `.meta` extension and contain the following `key:value` information:

- `image_url` – a URL of the downloaded image
- `width` – width of the image in pixels
- `height` – height of the image in pixels
- `url` – a URL of web page containing the image
- `title` – title of the web page
- `query` – a query based on which the image was found
- `source` – a source from which the DuckDuckGo received results (usually Bing)
- `format` – the format of the image. In case of reference (see below), *ref* value is present
- `timestamp` – timestamp of the time of download
- `md5` – md5 code of the image

Lastly, as it was often searched for models in subsequent years, where no facelift was done, this led to repeated appearance of a single image in the dataset. To prevent the unnecessary usage of disk space, we have employed a reference mechanism. URL address of every downloaded image was kept and for every newly downloaded image, its URL was checked against the set of URLs in the dataset. If its URL was already present, instead of moving the image into the dataset, only a small file with `.ref` extension was generated containing the path within the dataset to the already-present image. This set of URLs was generated for each vehicle type individually, so the individual parts could be used independently as stand-alone datasets. The overall scheme of the dataset is depicted in Fig. 5.5. The numbers of acquired images are shown in the Table 5.5.

In this step we acquired a huge number of images from the Internet that would be the backbone of our dataset. These images then needed to be further processed based on their content.

5.7 Vehicle Detection

The previous step of the pipeline produced a dataset of vehicles. However, we did not acquire any information on which entries were usable, as some of the entries did not possess

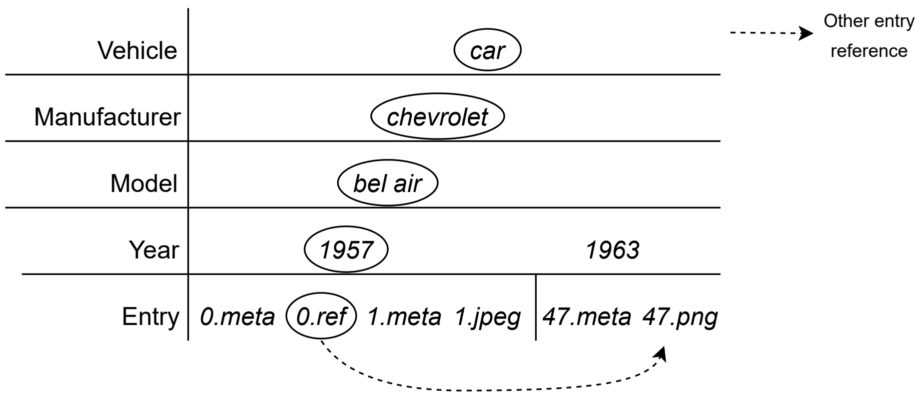


Figure 5.5: A hierarchical scheme of dataset structure. Vehicle, manufacturer, model, year and enty levels are present. One entry may refer to another. This follows both the logical structure of the dataset and the way the files are stored on disk as there are four layers of folders after which entries are stored.

Vehicle type	**# of images**	**# of references**	**Total**
Cars	3 046 120	1 616 230	**4 662 350**
Buses	175 189	259 449	**434 638**
Trucks	288 567	277 333	**565 900**
Motorcycles	530 413	308 427	**838 840**
Total	**4 040 289**	**2 461 439**	**6 501 728**

Table 5.5: Number of images downloaded for each type of road vehicle.

images of vehicles (e.g., contained only the motor of the vehicle) and even if they did, the information on their position was missing. To address this problem, we used a pre-trained neural network to locate vehicles in the image and extract their bounding boxes.

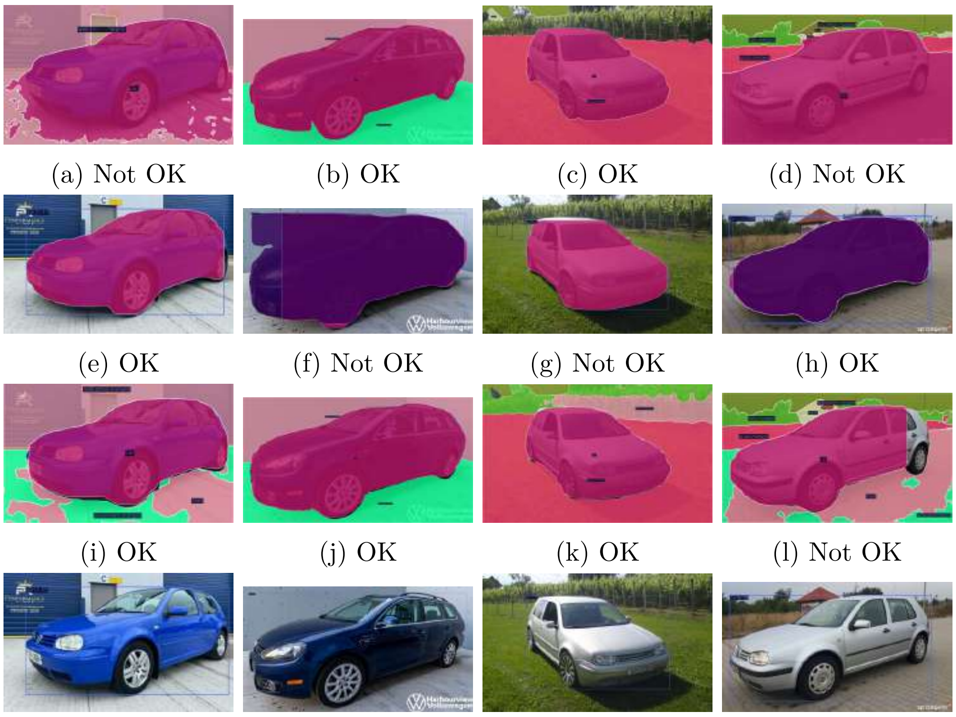
The first task was to choose a convenient pre-trained model providing suitable results. For this task, four models provided by `mmDetection`[8] were chosen, each trained on the COCO dataset [21]. Three of the networks performed instance segmentation: Maskformer [5], Mask_RCNN [12] and Panoptic_FPN [16]. The last performed only detection: TOOD [10]. Their empirical evaluation on a small data subset is presented in Table 5.6. Comparison of results from the networks is displayed in Fig. 5.6.

Network	**Note**
Maskformer	Tends to 'pour out' of the real boundaries of the vehicle
Mask_RCNN	Better than the Maskformer, still sometimes pours out
Panoptic_FPN	May perform cut-offs from vehicles
TOOD	Precise bounding boxes even in cases where previous networks failed

Table 5.6: Notes on different tested networks used for object detection.

The TOOD network provided the best results, hence it was used for further tests on images containing difficult light conditions, nonstandard colour palettes and peculiar vehi-

[8]https://mmdetection.readthedocs.io/en/latest/


(a) Not OK (b) OK (c) OK (d) Not OK

(e) OK (f) Not OK (g) Not OK (h) OK

(i) OK (j) OK (k) OK (l) Not OK

(m) OK (n) OK (o) OK (p) OK

Figure 5.6: A comparison of vehicle detecting NNs. Maskformer (first row) tends to 'spill' the detection outside the vehicle (a, d). Mask_RCNN (second row) a Panoptic_FPN (third row) sometetimes cut off parts of vehicles (f, g and l). TOOD (fourth row) performs good on all the figures the other networks failed on. Original images taken from the Internet.

cles. As the TOOD network was able to manage them (examples in Fig. 5.7), it was chosen to be the network used for detection of vehicles and their bounding boxes extraction.

The vehicle detection and bounding box extraction was performed on resized images so that no dimension was greater than 800 pixels. This was done to improve the performance of the network. Next, the inference was obtained from the network. All the detected objects within the image were saved. Then, a detection of the main vehicle (the object that would match the query provided to the search engine) was performed.

From the set of detected objects, every object with a low confidence was removed. This was done due to the observation that such objects were often badly detected or labeled. As a threshold, a confidence of 0.3 was used. Removed were also all objects too small to be used for the VMMR task. Objects with a width lower than 128px or a height lower than 90px were considered as such.

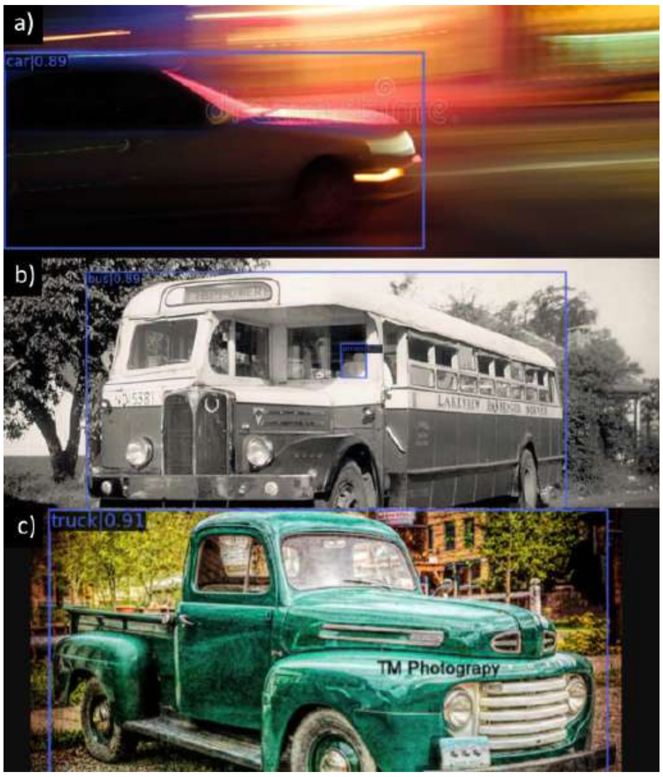


Figure 5.7: The TOOD model was able to perform well on a) blurred, night time images, b) atypical vehicles c) images with non-standard colour spectrum. Original images taken from the Internet.

Then, to handle cases, when several vehicles were found in the image, every remaining entry was evaluated based on its score which was calculated using its size, detection's confidence and a relevance of the entry's class. The score was calculated according to Eq. 5.1:

$$P = R \cdot W \cdot H \cdot C, \tag{5.1}$$

where W and H are dimensions of the evaluated bounding box and C is the confidence of the detection. The value R is calculated according to Eq. 5.2:

$$R = \begin{cases} 1, & if V_d = V_p \\ 0.75, & if V_d \in S_v \\ 0, & otherwise. \end{cases} \tag{5.2}$$

There, V_d is the class of the bounding box (from the COCO dataset), V_p is the class of vehicle being processed, and S_v a set of classes similar to the processed type of vehicle with which it may have been swapped. The S_v for motorcycles and buses is $S_v = \emptyset$, for cars $S_v = \{\text{truck}\}$ and for trucks $S_v = \{\text{car}, \text{bus}\}$. The values were set so that the detected vehicles are always preferred, but in case of their absence, similar vehicles may be also used. After each remaining entry was evaluated, the main vehicle was selected as:

- If no non-zero valued entry is present, the image has no main vehicle
- If one non-zero value is present, it is the main vehicle of the image

- If more than one non-zero value is present, the two vehicles with the highest values (P_1, P_2) and bounding boxes (B_1, B_2) are compared. If $P_1 > 1.2 \cdot P_2$, the highest score is significantly higher, and the vehicle with the highest score is the main vehicle. Also, when IoU[9]$(B_1, B_2) > 0.85$, they are the same vehicle and the vehicle with bounding box B_1 is the main vehicle
- Otherwise, the image has no main vehicle

This way, it was possible to use images even in cases when several vehicles were detected within.

Results of the detection were then saved into files of the same name as the input image with a `.bbox` extension. There, each detected object was saved in `COCO class, confidence, left_offset, top_offset, right_offset, bottom_offset` format. An example of an entry is:

```
motorcycle,0.73,79,398,767,812
person,0.86,747,407,945,720
person,0.84,328,236,622,701
person,0.76,100,331,318,530
motorcycle,0.63,721,549,1063,803
motorcycle,0.54,0,483,166,770
```

The offsets were upscaled based on the resizing ratio of the image, as the values from inference were produced for the downsampled version of the image. The first row of the file contains the main vehicle of the image. If no main vehicle for the image was found, the first row contains value 0. For our experiments, the main vehicles of images are used.

At this point of the pipeline, every image of the dataset had a file containing list of objects located within the image. It contained its classes, their confidence and their bounding boxes. The first row of this file contained the main vehicle of the image.

5.8 List of Dataset's Entries Creation

The last step of the dataset acquisition pipeline was the creation of entries lists and their division into training, validating and testing subsets. These are needed for the creation of models further in the thesis.

To acquire such subsets, a list of all usable entries for every type of vehicle was created first. There, every image containing a main vehicle was located, as well as every reference file referencing an image containing a main vehicle. These lists are composed of two CSV files *vehicle_available* and *vehicle_meta*, where *vehicle* is replaced with a specific vehicle type (e.g., *cars_available* and *cars_meta*), with scheme of the *vehicle_available* being in the order of appearance:

- `IDX` denotes the order of entry in the list
- `VEH, MAN, MOD, YR, ID` are parsed way to the entry. This information can be used as a ground truth for the VMMR task
- `IM_VEH, IM_MAN, IM_MOD, IM_YR, IM_ID` are parsed way to the referred entry. In case of images the values are the same as in the previous point

[9]Intersection over Union

- `IM_FORM` is the format of referred image. When joined with the previous bullet point using the '/' character, it provides path to the referred image
- `CLASS, CONV, BB_L, BB_T, BB_R, BB_B` are the same values as in first lines of `.bbxs` files
- `COMBINED` is join of `VEH, MAN, MOD, YR, ID` using the '/' character

The scheme of *vehicle_meta* is `IMG_URL, ORIG_W, ORIG_H, URL, TITLE, QUERY, SE, FORMAT, TIMESTAMP, MD5`, which copies the information in meta files discussed in Section 5.6. An entry from *vehicle_available* may be connected to its metadata using the `IDX` attribute. The whole list is split into two files as the *vehicle_meta* files take up a lot of space and there is no need to load them into memory for our task. The numbers of entries are displayed in Table 5.7.

Vehicle type	**# of usable entries**	**# of entries in total**
Cars	4 428 419	4 662 350
Buses	331 500	434 638
Trucks	543 835	565 900
Motorcycles	820 349	838 840
Total	**6 124 103**	**6 501 728**

Table 5.7: Numbers of usable entries in the dataset per vehicle type.

Finally, the lists were split into training, validating and testing sets. As a splitting ratio, we chose 0.7/0.15/0.15. Each vehicle type list was processed separately. This splitting was done in three steps:

- In the first step, the *vehicle_available* list was shuffled and split in the given ratio.
- As the list contained images and references, it may happen that a reference entry ended in a different subset than the image. This had to be dealt with, otherwise, the validation and testing of models could be performed on the same images they were trained on. In this step, reference entries with their referred images in different subsets are found and removed from subsets.
- Lastly, these reference entries are put into the same subsets as their referred images.

This way, the final subsets are not precisely in the given ratio. They are, however, sufficiently close as can be seen in Table 5.8.

Vehicle type	**Train size**	**Val size**	**Test size**	**Exact 70%**	**Exact 15%**
Cars	3 099 972	663 621	664 826	3 099 893	664 263
Buses	233 508	48 573	49 419	232 050	49 725
Trucks	382 015	80 578	81 242	380 684	81 575
Motorcycles	574 227	123 001	123 121	574 224	123 052
Total	**4 289 722**	**915 773**	**918 608**	**4 286 851**	**918 615**

Table 5.8: A sizes of individual subsets for each type of road vehicles.

5.9 Summary

In this chapter, we have explained the data acquisition pipeline, which was followed to obtain a new vehicle dataset. The dataset is based on information found on Wikipedia and contains public available images. For the information retrieval, two approaches were tested: semi-automatic and automatic. As the first one was not able to effectively collect data in the scope desired, the latter was used.

With over 4 milions of unique entries, it surpasses every dataset from Chapter 3 in size and variety of covered vehicles. However, evaluation of trained models needs to take place in order to confirm its usability and possible superiority to existing datasets.

Chapter 6

Algorithm Selection

Before experimenting on a large dataset, several experiments were performed on a small subset of the acquired data to find the best possible approach to the VMMR task. Two approaches to the classification were evaluated: a standard classification and an approach using embeddings. In this chapter, it is described how the subset was acquired and how the experiments were performed. Also, an evaluation of the experiments is presented.

6.1 Subset Acquisition

For the purpose of the approach-deciding experiments, it is important to obtain a small dataset (further referred to as **the subset**) that would be both compact, to allow for a fast evaluation of experiments, and robust to include a sufficient number of entries per each model and manufacturer present in it.

In the first step of the subset retrieval, we studied the whole dataset from the perspective of the reuse of images in different models/manufacturers classes. Fig. 6.1 and Table 6.1 show how many images were used under several models (e.g. the same image was used as Škoda Octavia and Škoda Fabia or Ford Mondeo) for each vehicle type. The values are written for images with ten or fewer uses, as listing all the values would be spatially demanding. Fig. 6.2 and Table 6.2 show the amounts of images used under different manufacturers (e.g. an image was used both as a model produced by Škoda and as a model produced by Ford). As can be seen, with the number of reuses between models/manufacturers, the amount of images decreases rapidly. For the subset creation, only images belonging to a single model/manufacturer were used to prevent using a single image in several classes.

From this section of the dataset, it was needed to select images from different manufacturers and models so that there was a lesser number of classes of manufacturers than the number of classes of models, while at the same time preventing from having too few classes of manufacturers. This was addressed by a search for models with at least 100 images. Such a model's manufacturer was then searched for other models with at least 100 images. If the manufacturer contained at least 3 such models, these models with 100 images were added into the subset. From one manufacturer, no more than 5 models were added to the subset. The 100 images were divided into training/validating/testing part in a ratio of 0.7/0.15/0.15.

This process was performed for every vehicle type and the results were merged. This way, a subset for experiments in this chapter was created. It contains 25 300 images of vehicles belonging to 249 classes of models and 47 classes of manufacturers. The number of

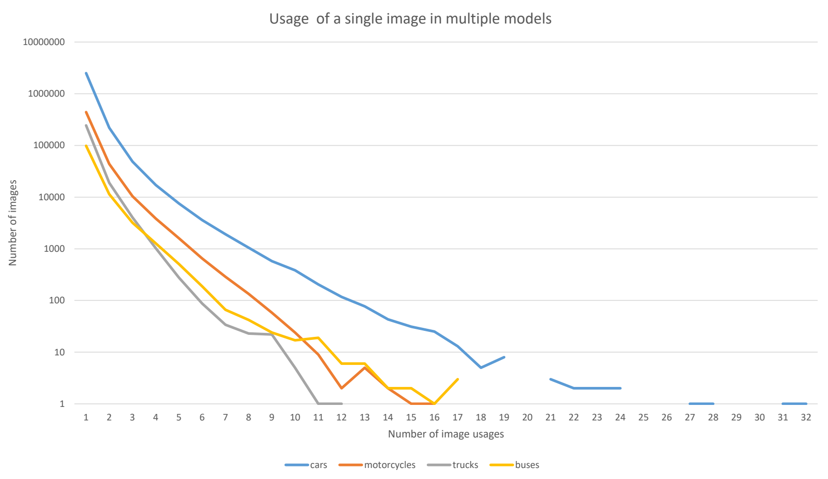


Figure 6.1: Graph depicting number of usages of images in different models.

Number of uses	**1**	**2**	**3**	**4**	**5**
# of car images	2 501 595	200 009	48 584	17 154	7 548
# of motorcycle images	441 519	43 912	10 382	3 841	1 603
# of trucks images	243 931	18 958	4 031	1 026	277
# of buses images	98 629	11 395	3 200	1273	507

Number of usages	**6**	**7**	**8**	**9**	**>=10**
# of car images	3 602	1 914	1052	578	922
# of motorcycle images	651	287	134	58	44
# of trucks images	87	34	23	22	7
# of buses images	188	66	42	24	56

Table 6.1: Numbers of images used in ten or fewer different models.

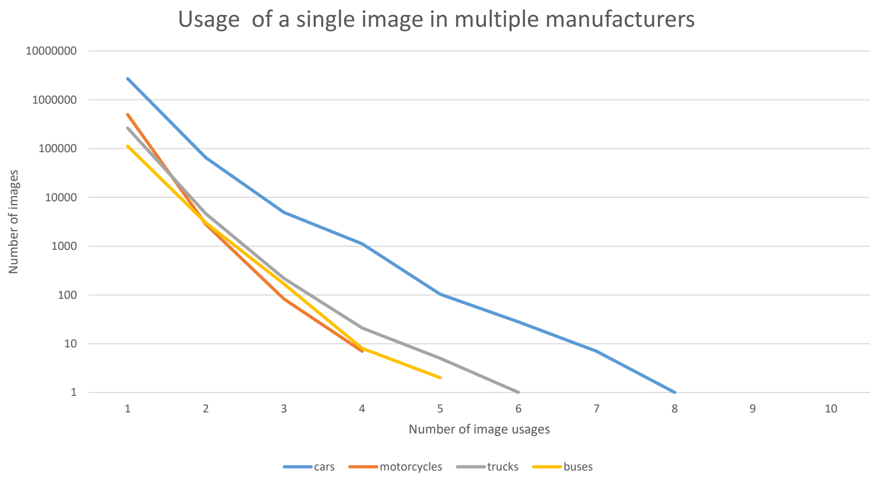


Figure 6.2: Graph depicting number of usages of images in different manufacturers.

Number of uses	**1**	**2**	**3**	**4**	**5**
# of car images	2 731 569	65 203	4 932	1 114	103
# of motorcycle images	499 544	2 796	83	7	0
# of trucks images	263 533	4 618	218	21	5
# of buses images	112 189	3 013	168	8	2

Number of usage	**6**	**7**	**8**
# of car images	28	7	1
# of motorcycle images	1	0	0
# of trucks images	1	0	0
# of buses images	0	0	0

Table 6.2: Numbers of images used different manufacturers.

manufacturers classes is stated after the merging of manufacturers as discussed in Section 5.4.

Lastly, all images in the subset were cropped according to their bounding box to speed up their loading time. This process, together with samples of images from the subset is present in Fig. 6.3.

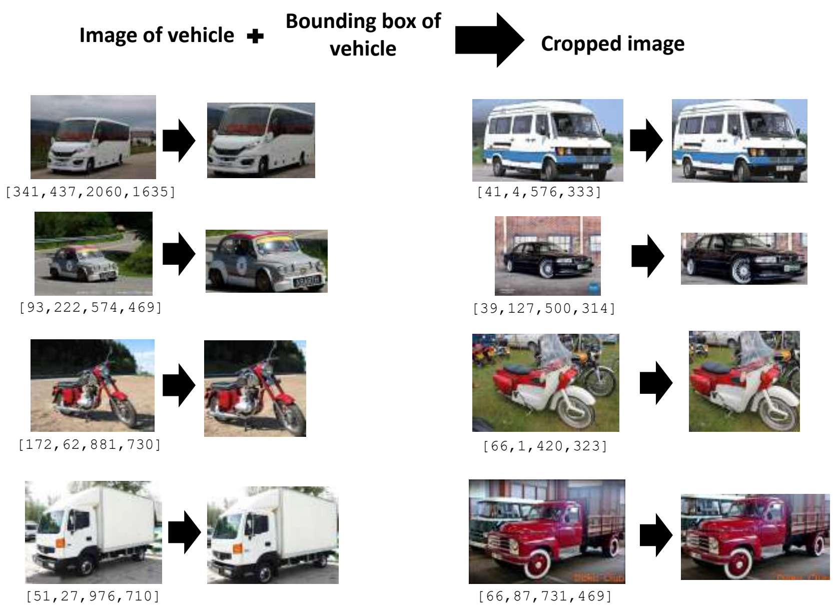


Figure 6.3: Before images are added to the subset, they are cropped based on the bounding box of their main vehicle.

6.2 Classifier Training Experiments

The first utilization of the subset acquired in the previous section was to train neural networks for the classification task. This was done using a mmClassification [6] toolkit on a computer equipped with Nvidia GeForce RTX 3060 with 12 GB VRAM.

The mmClassification is an open-source image classification toolbox based on PyTorch[1]. Together with training and testing of networks, it contains visualization options and detailed logging information. It provides a large amount of already implemented networks, for which it is possible to download weights and fine tune them or use them to acquire inference.

The models used for the experiments are listed in Table 6.3 together with the number of parameters they operate with and the size of their input. As can be seen, both convolutional (Resnet and ConvNext) and transformer based networks (VisionTransformer, Swin-Transformer, Swin-Transformer 2) were used. From now on, VisionTransformer will be abbreviated to ViT, Swin-Transformer to Swin and Swin-Transformer 2 to Swin2. The models were pretrained on the ImageNet 1K dataset and fine tuned. This was done to decrease the time needed for training and to increase the performance of transformer-based networks (according to Section 4.4). Except for the Swin2, we have used networks that take images of size 224×224 as input.[2]

Name	**# of parameters (M)**	**Input size**
Resnet 50 [13]	25.56	224×224
ConvNext [24] (base)	88.59	224×224
VisionTransformer [9] (base)	86.86	224×224
Swin-Transformer [23] (base, window size =7)	87.77	224×224
Swin-Transformer 2 [22] (base, window size = 16)	87.92	256×256

Table 6.3: List of networks used for experiments in this section. Together with their name, number of parameters and input size is shown.

Every experiment took 100 epochs. The batch size was 32. The optimizer used was Adam, with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $eps = 1 \times 10^{-8}$. Learning rate was modified during the training using a warmup and a learning rate decay mechanisms. A linear warmup was used for the first 20 epochs, starting with learning rate $lr_0 = 0.0$, going to $lr_{20} = 9 \times 10^{-4}$, where lr_x means learning rate at the x^{th} epoch. From the 20^{th} epoch, the learning rate was decreased using a cosine policy, ending at $lr_{100} = 1 \times 10^{-5}$. The whole course of the learning rate during an experiment is displayed in Fig. 6.4. During the experiment accuracy evaluation was performed after every epoch.

Before batches of data were put into the networks during the training, they were augmented to increase their variability. The augmentation steps were the following:

1. Images at the input were horizontally flipped with probability $p = 0.5$. This was possible as in the case of vehicles the horizontal flip (unlike a vertical one) does not change the nature of the image.

[1] https://pytorch.org/

[2] The reason the Swin2 network takes a larger input is the the different sizes of its window (8 or 16 instead of 7).

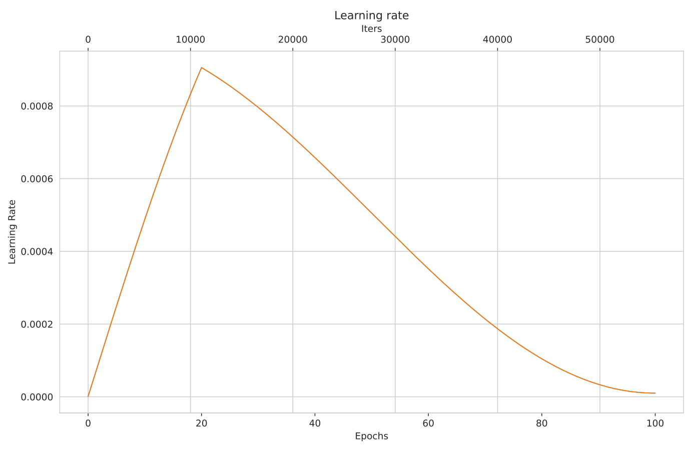


Figure 6.4: Learning rate during experiments in this section.

2. Images were reshaped into a square and resized according to the size expected by network. This means that images were reshaped into squares with an edge size of 224 or 256 pixels.

3. From the squared image, another squared image was randomly cut out and resized to the size of the original squared image. This cut-out square covered from 0.7 to 1.0 of the area of the original squared image. This was performed to slightly change the viewed part of the vehicle, while preventing from focusing on small parts of vehicles from which, the model would not be distinguishable (as with a handle or a side window).

4. The next step was adding a colour jitter to emulate different cameras, lenses and environmental conditions. Brightness and contrast factors of the jitter were taken from interval $\langle 0.6; 1.4 \rangle$. The saturation factor was taken from interval $\langle 0.9; 1.1 \rangle$. A factor of 1.0 means keeping the current state, while lower values means toning down and higher values toning up.

5. Afterwards, normalization was performed. The ImageNet dataset's $mean = (123.678, 116.280, 103.530)$ and $std = (58.395, 57.120, 57.375)$ values were used as the networks were pretrained on this dataset.

6. Lastly, transformation to grayscale was applied with probability $p = 0.2$ to emulate grayscale filters.

A visualization of this augmentation is depicted in Fig. 6.5. For batches used during validating and testing, only resize and normalization (i.e. steps 2 and 5) were applied.

Table 6.4, displays the results of the experiments. Besides model classification (listed under `mod` in the table), each network was also trained for a manufacturer (listed under `man` in the table) classification task. The values presented in the table were achieved on the testing part of the subset using a checkpoint of the network that achieved the highest Top1 accuracy during validation.

It can be seen that the best results were achieved using the ConvNext network, closely followed by Swin2 network. Please note that the models with the best results are the ones with the highest number of parameters. This may imply that for a complex task such as a

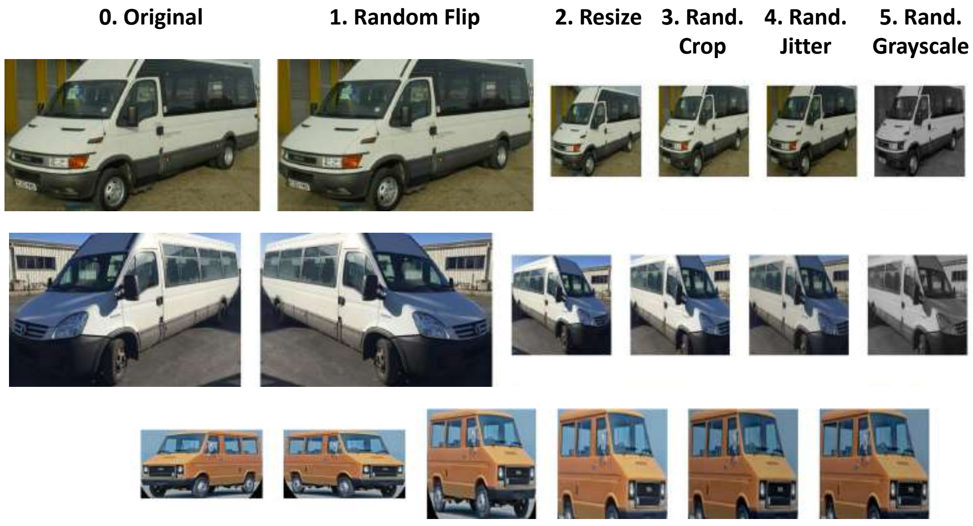


Figure 6.5: Visualization of subsequent steps of the data augmentation.

Name	**Top1 mod(%)**	**Top5 mod(%)**	**Top1 man(%)**	**Top5 man(%)**
Resnet 50	51.10	77.04	70.20	90.10
ConvNext	**65.19**	**86.78**	**81.30**	**95.01**
ViT	37.85	64.03	48.93	82.13
Swin	64.53	85.93	79.78	93.80
Swin2	65.10	86.43	81.23	94.82

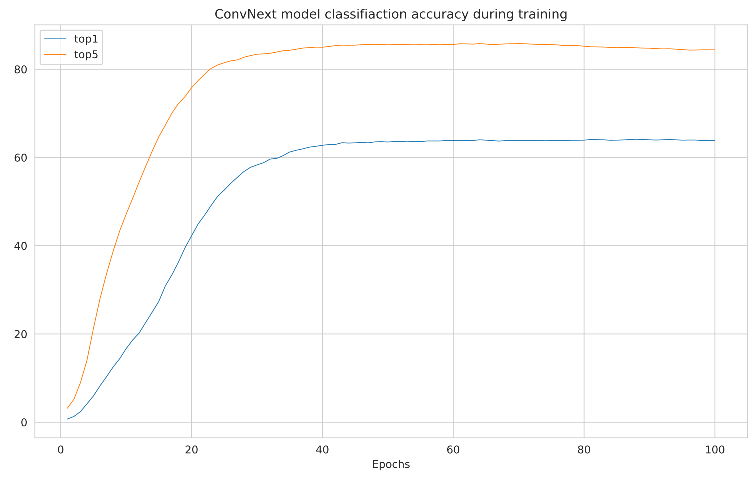
Table 6.4: Results of the classification experiments on the networks. For each type of network, its Top1 and Top5 performance are displayed for both model (mod) and manufacturer(man) classification tasks.

VMMR, large networks are needed as the result depends on many factors (e.g. the overall shape of the vehicle, its size, colour and shape of individual components that have to be understood and found in advance).

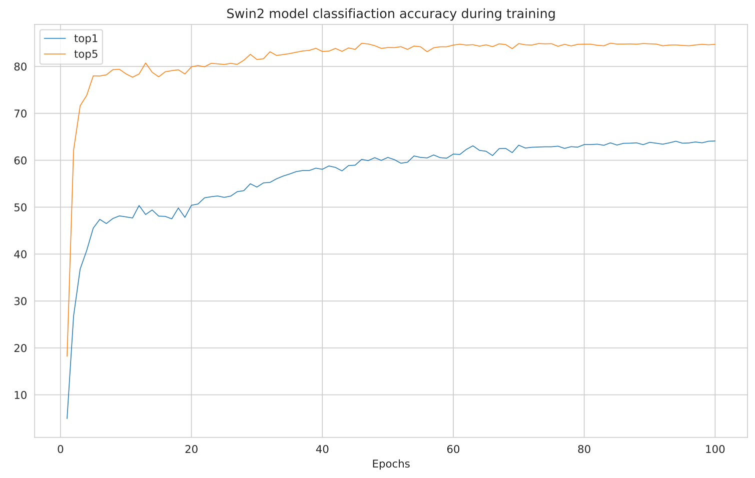
Despite none of the two best models performing above 70% on Top 1 accuracy on models, both of them achieved above 85% at Top 5 accuracy. When taking into account that the networks were trained on 17 710 images belonging to 249 classes, resulting in only 71 images per model on average, the experiments proved the capabilities of the networks used.

From the point of view of model and manufacturer classification, every network performed 10 – 15% better when the task was to classify a manufacturer instead of the model. These results prove that the experiments were executed well and the subset used was well assembled.

The accuracy of the ConvNext and Swin2 networks on model recognition during training is shown in Fig. 6.6.



(a) Accuracy of ConvNext on models during training.



(b) Accuracy of Swin2 on models during training.

Figure 6.6: Graphs of accuracy of ConvNext and Swin2 networks during training. While for the ConvNext grew the accuracy rather slowly, for the Swin2 it increased rapidly. The ConvNext results also do not contain fluctuations of the accuracy, unlike the Swin2.

6.3 Embedding Utilization Experiments

Another set of experiments utilizing the subset of the dataset explored a classification without the need of having a fully-connected classification head (i.e. realized using a one or several fully connected layers). This effort was carried out because with many classes, this fully-connected classification head takes a huge amount of resources. Instead, we have decided to utilize the embeddings vector taken from the encoder part of the network differently.

In Chen et al. [4], a self-supervised contrastive learning method is discussed, that uses a loss function which, in embedding vector space, pulls together members of the same classes and pulls away all other samples, effectively creating clusters of classes in the embedding space. This is done by duplication of processed batches, which are then independently augmented. The two batches are then merged and fed into the encoder network to retrieve their embeddings (as embeddings are the outputs of the encoder network). These embeddings are further passed into two subsequent fully connected layers with a ReLU nonlinearity in between them. The shapes of their input and outputs are $l_1 = (e_s, e_s)$ and $l_2 = (e_s, 128)$, where l_x denotes the layer so that l_1 precedes l_2, first of the tuple denotes the input size, the second its output and e_s the size of the embedding. Finally, the loss function then pulls together the pairs of embeddings originating from the same image, while pulling away every other embedding. A disadvantage of this self-supervised approach is that it repels every other embedding from the batch even in case of belonging to the same class. This issue is addressed by Khosla et al. [15] where a Supervised Contrastive learning method is introduced. This new method uses a new loss function which utilizes labels of the images to manage that all the embedding on the output of the encoder network belonging to the same class are pulled together. As the method uses labels, it becomes a supervised learning method instead of self-supervised. A schematic comparison of the two methods is depicted in Fig. 6.7.

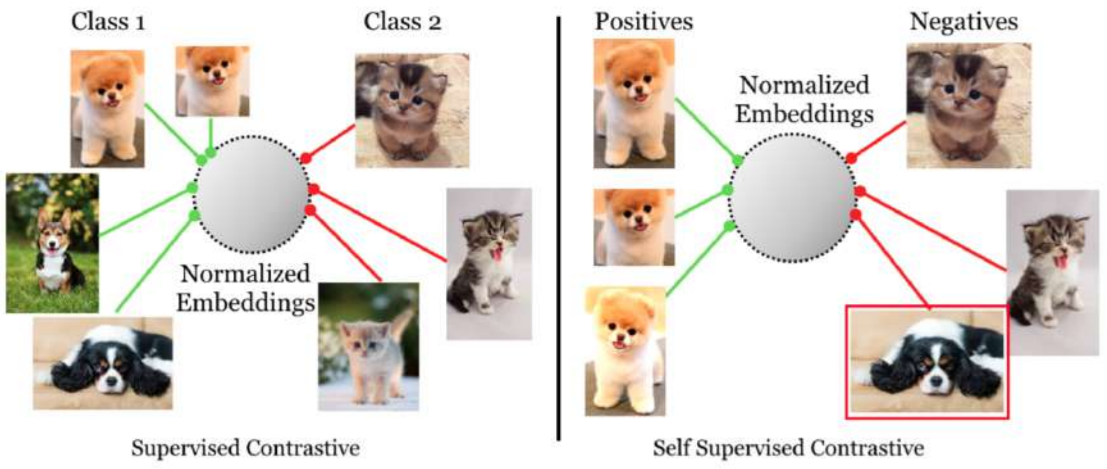


Figure 6.7: A comparison between Self-Supervised and Supervised contrastive learning. While in the Self-Supervised approach embedings of images from the same classes are repelled as no labels are known, in the supervised scenario only images from different classes are repelled. Image taken from Khosla et al. [15].

As we possess labels for the images, we have decided to use the supervised contrastive learning method to train an encoder, which would output class-clustered embeddings of input images. The classes are in our case the individual models of vehicles. On such embeddings, a non-neural network classifier (e.g. Gaussian mixture model or K-means clustering) would be trainable with lesser resources needed than using the fully-connected classification head.

All experiments described in this section were performed on metacetrum. It is a virtual organization providing computational and software resources freely for the academic community. There, GPU nVidia Tesla T4 with 16GB VRAM was used. It allows for running multiple experiments in parallel and provides GPUs with higher memory and hence increasing the possible batch size.

The PyTorch implementation of supervised contrastive learning was taken from the GitHub of Yonglong Tian[3] , one of the co-authors of the Supervised Contrastive learning paper.

During the experiments, we followed the paper as closely as possible. The networks were trained from scratch. Every experiment took 200 epochs, as suggested in the paper. For the batch size, we have experimentally found that 100 images is the highest size the GPU can handle (meaning 200 images goes to the network, as the batch is doubled). The batch size is important since in the loss function all elements on the output of the network are considered when calculating the loss value for each of them. For this reason, it is important to have a batch large enough. We found the batch size of 100 suitable, as in the paper a batch size of 256 was used for ImageNet with 1000 classes, while our subset only has 249 classes. Mixed precision training and gradient clipping were used to achieve this batch size. The optimizer used was SGD as in the paper. We used the initial learning rate $lr_0 = 0.05$ with cosine decay. The course of learning rate during the experiment is shown in Fig. 6.8. The temperature value used by the loss function was $\tau = 0.1$ as in the paper.

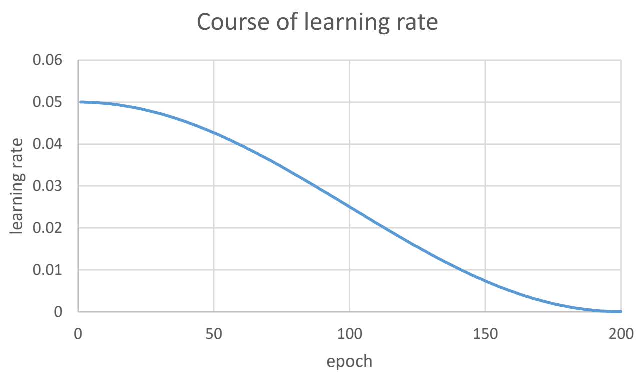


Figure 6.8: Course of learning rate during experiments in this section.

The biggest deviation from the paper was the size of the input images. While in the implementation the size of input images was 32×32, we did not find this suitable for our task of vehicle model recognition so we have increased the size to 224×224. We have also changed the implementation of ResNet networks provided in the implementation provided by Aladdin Persson[4] as the original version did omit the 3×3 max pooling layer at the beginning of the network resulting in higher memory requirements and limiting the maximal batch size at 8. If not specified otherwise, the data augmentation policy used was the same as in the previous section.

As the encoder network, ResNet 50 was used, producing embeddings of size 2048 if not stated otherwise. The ResNet models embeddings are linearized average pooled outputs of the last residual blocks.

In the first experiment, we tried to apply the method using the described setup. The course of the loss function is depicted in Fig. 6.9. While the loss on training data decreased continually, the validation loss started to stagnate around the value of 8. To confirm or reject the correctness of this behaviour, we applied a t-SNE algorithm to 2000 embeddings

[3]https://github.com/HobbitLong/SupContrast

[4]https://github.com/aladdinpersson/Machine-Learning-Collection/blob/master/ML/Pytorch/CNN_architectures/pytorch_resnet.py

from which we selected several classes to display. The result is displayed in Fig. 6.10. From the image is evident that no class-based clusters are created and hence the method does not work properly. From now on we considered any similar loss values in the validation phase as insufficient.

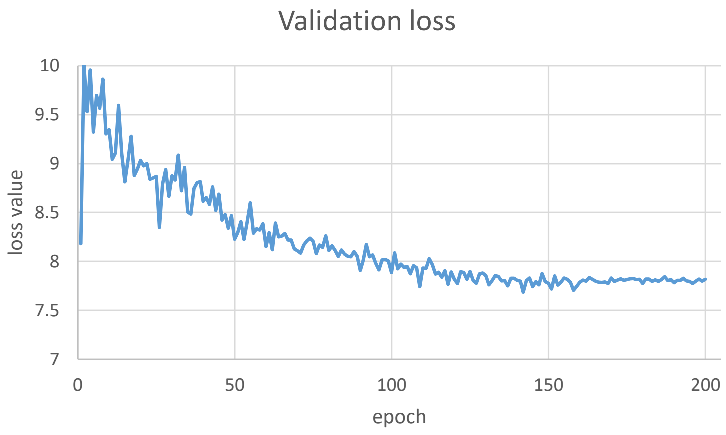


Figure 6.9: A validation loss of standard ResNet50 model.

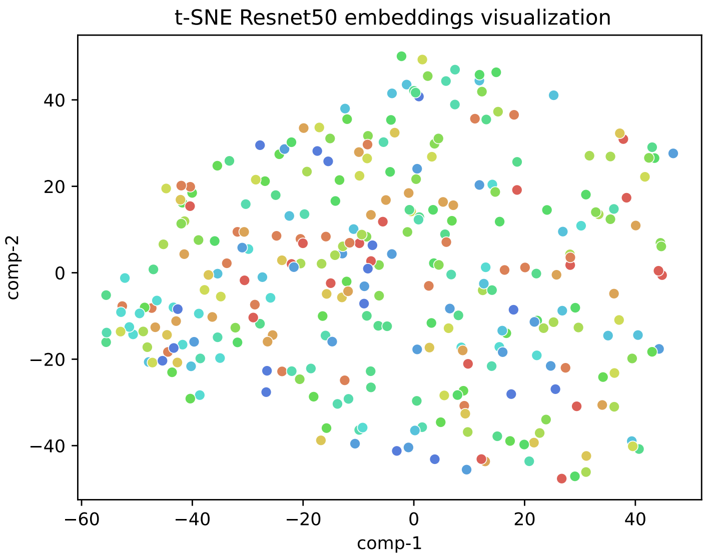


Figure 6.10: Visualization of embeddings retrieved after the first experiment. Different colour note different class. No signs of clustering are visible.

To fix this behaviour, we have reduced the size of the output embedding from 2048 to 256 as we have assumed a shorter vector may be easier to train as there are fewer parameters to fine tune while still providing a suitable number of dimensions for encoding 249 classes. The embedding length reduction was performed in the last residual block of the encoder, where the last 2D convolutional layer reduced the dimensionality to 256 instead of increasing it back to 2048 as was the input. As the dimensionality of the input and output of the residual block differed, an identity downsample had to be used to create the residual connection.

The result of the experiment with the embedding of size is shown in Fig. 6.11. As can be seen, the results on the validation data are still not suitable for further experiments.

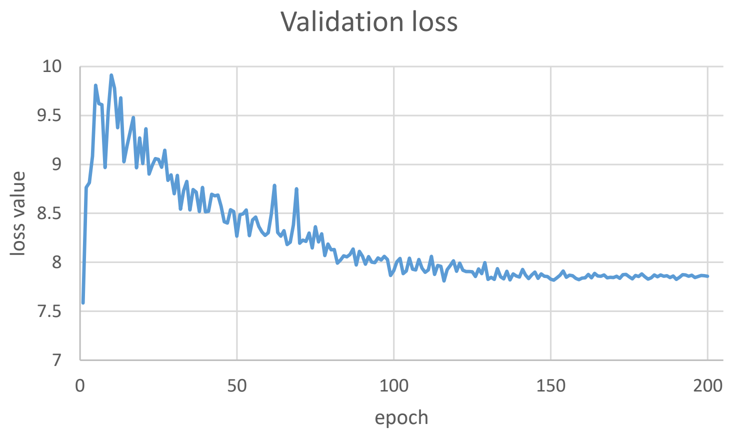


Figure 6.11: Validation loss of ResNet 50 model outputting embeddings of length 256.

In the next attempt to improve the results on validation data, we have tried to increase the variability of the dataset by using a different, more distortive transformation. We chose to use AutoAugment [7] augmentation with ImageNet policy. A comparison of images augmented by the original augmentation policy and images augmented using AutoAugment is displayed in Fig. 6.12. This step was chosen to eliminate overfitting that may occurred.

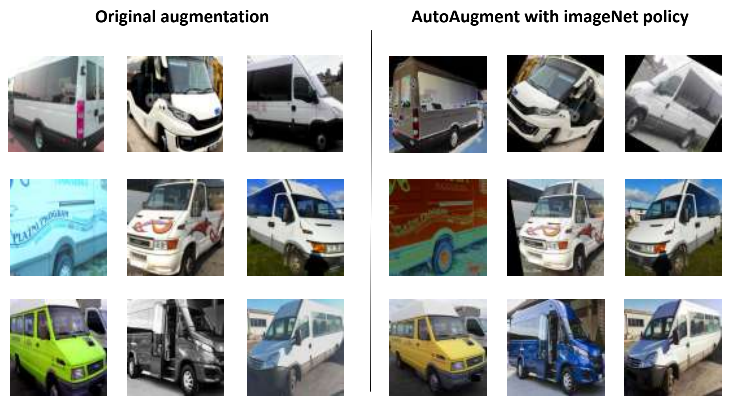


Figure 6.12: Comparison of the original augmentation policy (left) with images augmented using the ImageNet AutoAugment policy(right).

We ran the experiment with both sizes of the embedding vectors. The results are depicted in Fig. 6.13. It is evident that no change occurred and results are still unusable for further classification. Also the loss is less continuous when using this augmentation approach.

The last experiment we performed to improve the course of learning was the utilization of a larger neural network. This decision was made based on an assumption that a larger

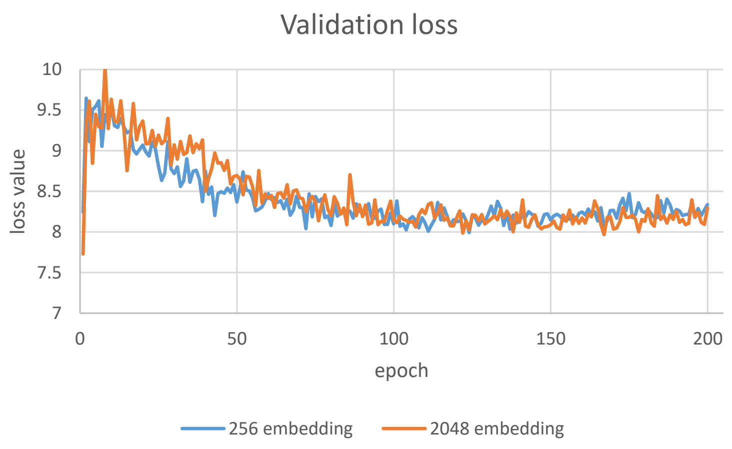


Figure 6.13: Validation loss of ResNet 50 model using the AutoAugment. Both sizes of vectors are used.

network would be able to generalize data easily which would lead to a further decrease of loss value on validation data during training. The chosen network was Resnet 101 with its 44.5M parameters. On this network, we have run experiments including both shorter and longer versions of embeddings and both the original data augmentation and AutoAugment. The results are displayed in Fig. 6.14. It is visible that during all the experiments the loss value of the validation stopped around the value of 8 again.

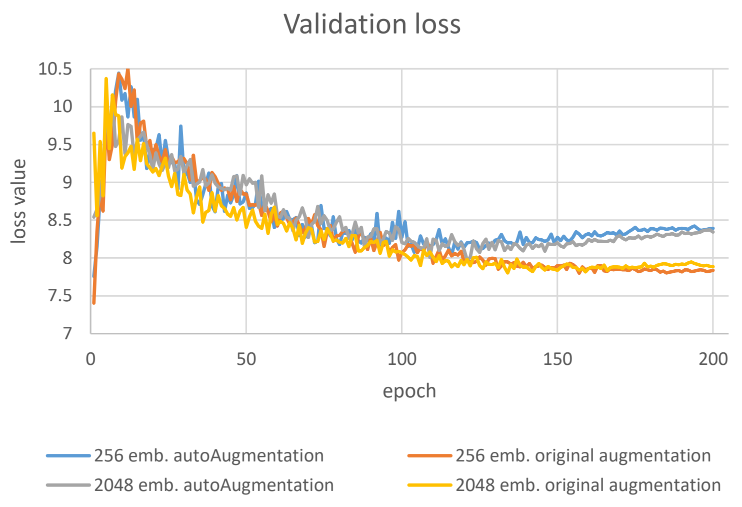


Figure 6.14: Validation loss of Resnet 101 network. All types of augmentation and lengths of vectors are used.

No other experiment aiming for the better embedding acquisition was run. As none of the experiments met the expectations, no further experiments with non-neural network classification were performed.

The reason for the failure of the experiments may be the nature of the task, where the classification of generally similar vehicle models proved to be a more difficult task than the classification of more distinguishable classes of the Cifar100 dataset and hence insufficient amount of training data. Another possible reason may be the huge increase in the size of images as they may need a different size of the embedding vectors. Yet another option is an unclear explanation in the paper/incorrect implementation (recall the missing pooling layer at the beginning of the ResNet networks). This could be resolved by running further experiments and obtaining more in-depth insight into the method and the implementation, however as this is not the primary task of the thesis, it is out of its scope.

6.4 Summary

In this chapter, we have discussed the creation of a subset of our vehicle dataset, which would be suitable for testing methodologies that could be used for the VMMR task.

The first usage of the subset was on standard classification approaches using neural networks. Several pretrained models were examined, both convolutional and transformer based. The best results were achieved using the ConvNext network, closely followed by the Swin2 network.

The next set of experiments tried to train an encoder network to output embeddings of input images which would form clusters based on a class of the input. Several experiments were conducted, however as the results were unsuitable for further processing, this idea was abandoned.

For the VMMR task on a larger dataset was hence chosen a standard neural network approach using large models.

Chapter 7

VMMR Experiments

While in previous chapters we have discussed the acquisition of a large dataset of vehicles and methodologies of utilizing it for the VMMR task, in this chapter we combine the two.

Firstly, we train a model on as much data as possible for the task of vehicle model recognition. Then, we use other publicly available datasets to train other models and lastly, we compare the accuracy of the models using images from all the datasets.

7.1 Large Scale Training

In the previous chapter, we discovered that for the task of vehicle model recognition, we will have to use the standard approach by using classification networks. This however implies usage of a huge classification head as the dataset we assembled is composed of several thousands of models.

To be able to train such models, we have decided to perform the training using the resources offered by metacentrum. There, we used nVidia A40 GPU for all the experiments discussed in this chapter. The use of the metacentrum however also brings a limitation – the `brno 2` node of the metacentrum, which hosts the A40 GPUs, despite offering 4.29TB of storage per user limits the maximum number of files to 2.5 million per user.

Due to this constraint, we were not able to use all the data we possess. We have decided to only use the car part of our dataset further reduced to images that occur only once in the dataset as the selection from Table 6.1 still exceeded the limitation on number of files. This led to a dataset containing 2 207 847 images split into train/val/test subsets of 1 557 973/325 750/324 124 images. The subsets were based on the distribution between subsets discussed in Section 5.8. This dataset contains 6 922 models from 318 manufacturers.

As the best results in the previous chapter were achieved using the ConvNext architecture, closely followed by the Swin2 architecture, we have decided to use both of them for the experiments. Both architectures were used in base and large versions. Specifications of the networks are in Table 7.1. We used models pretrained on ImageNet 21K [8]. The Swin2 models take inputs of size 192×192 as this was the only input size available for the pretrained models and further finetuning for a higher input would be time and technically demanding.

The augmentation of data and course of the experiments were mostly the same as in section 6.2. The differences were that the number of epochs decreased to 5 as the amount of data increased rapidly. Also, the batch size was set to 100. Lastly, the warmup of

Name	**# of parameters (M)**	**Input size**
ConvNext (base)	88.59	224×224
ConvNext (large)	197.77	224×224
Swin2 (base, window size = 12)	87.92	192×192
Swin2 (large, window size = 12)	196.74	192×192

Table 7.1: List of networks used for experiments in this section. Together with their name, number of parameters and input size is shown.

the learning rate to its maximal value took 10 000 batches. After that, the learning rate decreased with the cosine policy as before.

The results of models after the fifth epoch are displayed in Table 7.2. A graph of their validation accuracy is displayed in Fig. 7.1. It can be seen that the ConvNext architectures have beaten the Swin2 architectures in both versions. Furthermore, we can observe that the large models performed better than the base ones. The best results were hence obtained with the large ConvNext model and it will be used in further experiments. It is worth mentioning that the results obtained are superior to the results in Section 6.2 by almost 10% despite having roughly 36 times more classes. The author assumes that this behaviour confirms the validity of the dataset. For the best model, an experiment with recognition of only the manufacturer was conducted. As can be seen, the accuracy increased compared to the experiments on a smaller dataset.

Name	**Top1 mod(%)**	**Top5 mod(%)**	**Top1 man(%)**	**Top5 man(%)**
ConvNext B	73.62	90.6	/	/
ConvNext L	**74.17**	**91.13**	**88.9**	**96.42**
Swin2 B	68.75	88.36	/	/
Swin2 L	70.59	89.68	/	/

Table 7.2: L and B denote the version of the network (large/base). Results of the classification experiments on the networks after the fifth epoch of training.

7.2 Training on Different Datasets

To be able to compare the results of the model trained on our dataset with models trained on different datasets and to see the performance on images from outside of our dataset, we have downloaded additional car datasets and trained the large ConvNext models on them. To cover the widest possible range of manufacturers, we have decided to use the Stanford Cars dataset (196 car models, 16 185 images) to represent western manufacturers and the CompCars dataset (1 716 car models, 163 716 images) to represent eastern, mostly Chinese manufacturers. The datasets were split into training and testing parts in a 0.7/0.3 ratio per model.

All the images in the datasets were cropped according to the bounding boxes provided by them. This was done to set the same conditions for the classification.

The course of training the large ConvNext models on each of the datasets was mostly the same as in Section 6.2. The only difference was the number of epochs as to compensate

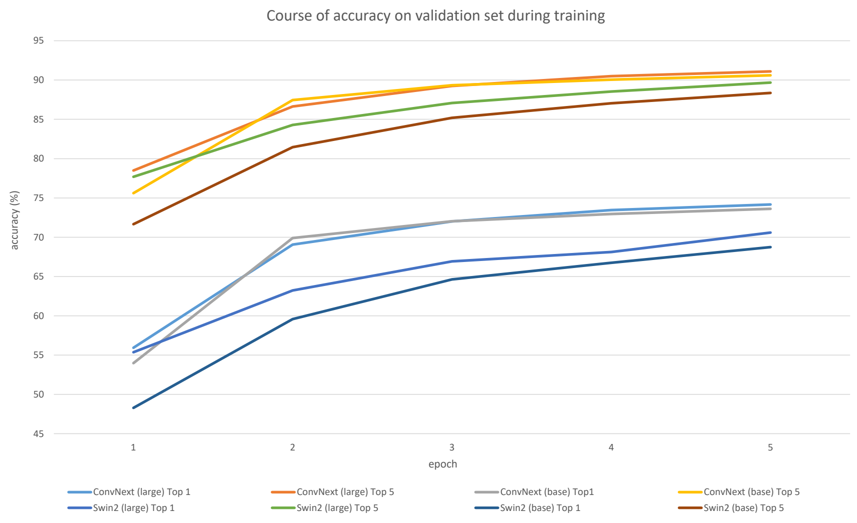


Figure 7.1: Validation loss of the networks discussed in this section after end of each training epoch.

for the smaller amount of images, the number of training epochs was increased from 5 to 50.

The highest achieved accuracy during the validation phase is shown in Table 7.3. As can be seen, the model trained on CompCars was able to reach high accuracy. This may be due to the sufficient amount of data needed for generalization and hand labelling every image within it (unlike for ours dataset), however, it may also be given by a low variety between individual images and the absence of challenging samples in the validation subset. The model trained on Stanford Cars dataset did not achieve such accuracy despite consisting of fewer classes of models. This is, presumably, due to the small size of the dataset that was not sufficient for training the large version of the ConvNext model.

Name	**Top1 mod(%)**	**Top5 mod(%)**
Stanford Cars model	40.76	74.75
CompCars model	93.52	98.52

Table 7.3: The best accuracy of models trained on different datasets.

7.3 Comparison of Accuracy on Different Datasets

The last experiment we conducted was focused on a comparison of the models from this chapter when used on images from outside of the datasets they were trained on. As the car models contained in the datasets used for training differ, it was not possible to make the comparison quantitatively. Instead, they have to be performed qualitatively when a human observer decides whether the classification was correct or not.

A set of images for the experiment was composed of samples from our dataset, CompCars and Standford Cars by selecting 10 models of cars from each. For each model selected, 2 images of it were randomly taken. This lead to a set composed of 3 datasets, containing

30 car models in 60 images. The images were taken from the test or validation parts of datasets. All the selected images together with their labels are shown in Appendix A.

For every classification, it was primarily examined, if the vehicle's model, according to the label value, appears in the Top 1 or Top 5 classes on the output of the model. If not, it was examined whether the manufacturer of the vehicle appeared in the results of the Top 1. It was also observed, if the model from the label appeared in the dataset the model was trained on. The table with results is in Appendix B. An example is shown in Fig. 7.2.

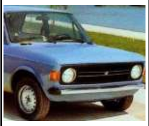
Image					
Label	Subaru Rex	VW Transporter	VW Transporter	Zastava Skala	Zastava Skala
Our DS	SUBARU Justy	VW Type 2	VW Transporter	ZASTAVA Skala	FIAT 128
Stanford	VW Golf *	VW Golf *	M-Benz Sprint *	Volvo C30*	Volvo C30 *
ConvCars	Jiangnan TT *	Jiabao V80 *	VW Defender *	FIAT 126P *	FIAT 126P *

Figure 7.2: An example of evaluation of images from different subsets on different classifier models. It consists of squared classified image, label and outputs from individual models coloured according to its correctness.

The table confirms findings from the previous sections that the classification of the images from the same dataset that was used for training results in labels appearing mostly in the Top 1 or Top 5 values. However, as the source of images shifts to different datasets, the results become less accurate. The main reason is the fact that the different datasets contain new models of vehicles, the classifiers were not trained to recognize. As can be seen, the highest number of entirely incorrectly (manufacturer was not in the Top 5 results) classified car models comes from not having the target model in the training dataset. In such cases the sheer number of models in the training datasets plays the key role – if we order the models by the number of classifications the target was not in the training dataset, we get the same sequence as if we ordered them by the number of classes in the training dataset (i.e. our model > CompCars model > Stanford model). For this reason, the model trained on our dataset outperforms the other models when used on images from other datasets. Interestingly, the manufacturers may be classified correctly when other models from them are present in the training dataset as in the case of Stanford model correctly recognizes LaFerrari being manufactured by Ferrari despite not having it in the training dataset.

We have observed that when using several images from the same class, the results of the models may occasionally vary. This may be caused by a different appearance of the vehicle (as in the case of the Chevrolet Fleetmaster samples) or by an absence of a significant identification mark (as in the case of the Audi S5). The occurrence of the phenomenon could be decreased by adding additional images to the training dataset.

When the second sample of the LaFerrari (displayed in Fig. 7.3) was classified using the model trained on our dataset, it was recognized as a model FXXK by Ferrari. After additional research, it was found to be the FXXK model, despite being incorrectly labelled as a LaFerrari. The same happened for the second insance of Chevrolet FleetMaster. These errors were caused by the nonsupervised nature of the creation of our dataset. This automatic self-correction creates room for experiments on implementation of a mechanism that would automatically correct wrongly labelled images. It is, however, outside of the scope of this thesis.

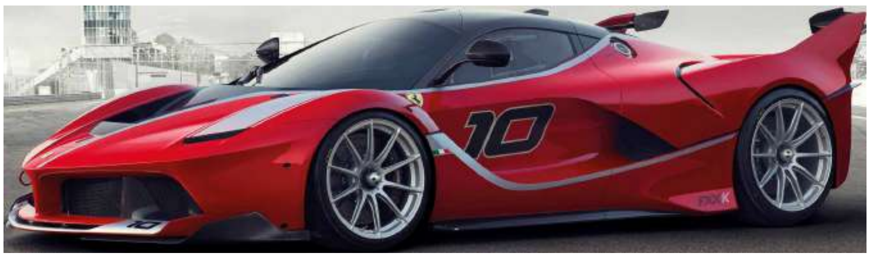


Figure 7.3: An image of Ferrari FXXK incorrectly labelled as Ferrari LaFerrari.

7.4 Summary

In this chapter, we have used unique images of cars from our dataset and trained a classifier on them. We have also trained classifiers on CompCars and Stanford Cars datasets to compare the performance of the models. When qualitatively compared, the classifier trained on our dataset outperforms the other models when used on data from different sources as covers the widest range of manufacturers and models.

Chapter 8

Conclusion

8.1 Summary

While many types of methodologies may be utilized to perform vehicle detection of some form (e.g., strain gauge for vehicle counting or LiDAR for detection of vehicle's direction), the only known approach used for the VMMR task is using image-processing and neural networks.

We have inspected many available vehicle datasets, but none was suitable for our task of a huge scale VMMR due to their outdated information, insufficient variability of models or focus on cars only. Hence, it was concluded that a new dataset had to be created.

This new dataset was acquired using automatic data scraping of information found on Wikipedia. The data were then processed and serialized into lists of vehicles which were downloaded. The downloading was done using the automatic orchestration of the DuckDuckGo search engine. More than 6.5M of images were downloaded and tested using a TOOD neural network for vehicle detection. More than 6.1M images were found usable and were put into the dataset, creating a dataset bigger than any of the inspected ones. Besides cars, the dataset also contains motorcycles, buses and trucks. The dataset was split into training, testing and validating subsets in order to be suitable for neural networks training.

Next, we experimented with a small subset of the dataset to find the most efficient way to use it for training a VMMR classifier. Two general approaches were tested. The first used a standard classification head at the end of an NN-based encoder. Further experiments discovered that the ConvNext network architecture performs the best in this case. The other general approach tried to structure embeddings of encoders using a supervised contrastive learning method to simplify the subsequent classification. Unfortunately, this approach did not perform well on our data and the standard approach to classification was used, despite being more resource-demanding.

Finally, a great part of our dataset (over 2M images) was used to train a classifier of car models that achieved accuracy Top 1: 74.17% and Top 5: 91.13%. Similarly, we used two other datasets (CompCars and Stanford Cars) to train other classifiers. The classifier trained on the CompCars outperformed the classifier trained on our dataset in terms of accuracy when experimenting on the datasets, they were trained on. However, when tested on images from different datasets, the model trained on our dataset outperformed the other models. This was due to the huge number of models present in the training dataset.

Overall, we have created a dataset which, despite being created mostly automatically and containing some incorrectly labelled images, is usable for training a VMMR classifier and that covers a huge variety of different models. As the accuracy of the classifiers trained on it in its current form does not go above 75% for the Top 1 values, it is not suitable for an industrial usage that would demand higher accuracy. As for the speed of the models the classifiers produce inference on average under 0.05s on the nVidia A40, Tesla T4 and RTX 3060 GPUs. However, it is important to keep in mind that the detection of vehicles has to precede the classification and hence the total time may increase up to 0.1s, leading to roughly 10 FPS when used in real-time applications. As for the hardware side of the application, since the trained model can be used for evaluation on the author's mid-tier RTX 3060 GPU, it can be said that running any form of application on a workstation would not be problematic. Using the model on a portable device (e.g. smartphone) would however require the computational part to be executed on a server-side, which could further decrease the system's speed.

8.2 Short-Term Perespective

A great portion of time spent on this thesis was spent on data manipulation. Now, the whole pipeline, beginning from the acquisition of existing vehicles and ending with the classification of vehicles in images, is complete, any further work should be less time demanding. This may in the short-term perspective include experiments on other approaches to classification, extending sources of information to cover a larger portion of eastern (mostly Chinese) manufacturers or merging models of manufacturers similarly to how we merged the manufacturers, as a single model may now be present under several names (e.g. Volkswagen's *Transporter 2* and *Type 2*). Another short-term task would be finding a way to overcome the limitations of the metacentrum and utilize more data during training, possibly across all the vehicle types.

8.3 Long-Term Perspective

From the long-term point of view, the ultimate goal of such datasets is to contain every possible model. This would require an automated process for scanning for newly manufactured models together with an expansion of present models to cover any new facelift. Another long-term goal may be a self-correction mechanism that would be able to detect wrongly-labelled images and correct them. Ultimately, a self-supervised dataset could automatically train new classifiers for every newly added model and offered it in the form of an application that would utilize it, together with vehicle detection.

Bibliography

[1] Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C. et al. Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*. 2017, vol. 72, p. 327–334. DOI: https://doi.org/10.1016/j.eswa.2016.10.055. ISSN 0957-4174. Available at: https://www.sciencedirect.com/science/article/pii/S095741741630598X.

[2] Bahdanau, D., Cho, K. H. and Bengio, Y. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. 2015, p. 1–15.

[3] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. et al. End-to-End Object Detection with Transformers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2020, 12346 LNCS, p. 213–229. ISSN 16113349.

[4] Chen, T., Kornblith, S., Norouzi, M. and Hinton, G. E. A Simple Framework for Contrastive Learning of Visual Representations. *CoRR*. 2020, abs/2002.05709. Available at: https://arxiv.org/abs/2002.05709.

[5] Cheng, B., Schwing, A. G. and Kirillov, A. Per-Pixel Classification is Not All You Need for Semantic Segmentation. *Advances in Neural Information Processing Systems*. 2021, vol. 22, NeurIPS, p. 17864–17875. ISSN 10495258.

[6] Contributors, M. *OpenMMLab's Classification Toolbox and Benchmark* [https://github.com/open-mmlab/mmclassification]. 2023.

[7] Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V. and Le, Q. V. AutoAugment: Learning Augmentation Policies from Data. *CoRR*. 2018, abs/1805.09501. Available at: http://arxiv.org/abs/1805.09501.

[8] Deng, J., Li, K., Do, M., Su, H. and Fei Fei, L. Construction and Analysis of a Large Scale Image Ontology. In: Vision Sciences Society. 2009.

[9] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020. Available at: http://arxiv.org/abs/2010.11929.

[10] Feng, C., Zhong, Y., Gao, Y., Scott, M. R. and Huang, W. TOOD: Task-aligned One-stage Object Detection. *Proceedings of the IEEE International Conference on Computer Vision*. 2021, p. 3490–3499. DOI: 10.1109/ICCV48922.2021.00349. ISSN 15505499.

[11] Guo, H., Zhao, C., Liu, Z., Wang, J. and Lu, H. Learning coarse-to-fine structured feature embedding for vehicle re-identification. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. 2018, vol. 776, p. 6853–6860. DOI: 10.1609/aaai.v32i1.12237. ISSN 2159-5399.

[12] He, K., Gkioxari, G., Dollár, P. and Girshick, R. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020, vol. 42, no. 2, p. 386–397. DOI: 10.1109/TPAMI.2018.2844175. ISSN 19393539.

[13] He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770–778.

[14] Kanan, V. *Object Segmentation* [online]. Vision Wizard, april 2020 [cit. 2023-1-15]. Available at: https://medium.com/visionwizard/object-segmentation-4fc67077a678.

[15] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y. et al. Supervised Contrastive Learning. *CoRR*. 2020, abs/2004.11362. Available at: https://arxiv.org/abs/2004.11362.

[16] Kirillov, A., Girshick, R., He, K. and Dollar, P. Panoptic feature pyramid networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2019, 2019-June, p. 6392–6401. DOI: 10.1109/CVPR.2019.00656. ISSN 10636919.

[17] Klein, L. A., Mills, M. K. and Gibson, D. R. P. *Traffic Detector Handbook: Third Edition - Volume I* [online]. Federal Highway Administration, may 2006 [cit. 2023-1-13]. Available at: https://www.fhwa.dot.gov/publications/research/operations/its/06108/02.cfm.

[18] Koehrsen, W. *Neural Network Embeddings Explained* [online]. Towards Data Sciencen, october 2018 [cit. 2023-1-13]. Available at: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526.

[19] Krause, J., Stark, M., Deng, J. and Fei Fei, L. 3D object representations for fine-grained categorization. *Proceedings of the IEEE International Conference on Computer Vision*. 2013, p. 554–561. DOI: 10.1109/ICCVW.2013.77.

[20] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E. et al. *Backpropagation applied to digit recognition*. 1989.

[21] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P. et al. Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014, 8693 LNCS, PART 5, p. 740–755. DOI: 10.1007/978-3-319-10602-1_48. ISSN 16113349.

[22] Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z. et al. Swin Transformer V2: Scaling Up Capacity and Resolution. arXiv. 2021. DOI: 10.48550/ARXIV.2111.09883. Available at: https://arxiv.org/abs/2111.09883.

[23] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y. et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *ArXiv preprint arXiv:2103.14030*. 2021.

[24] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T. et al. A ConvNet for the 2020s. *ArXiv preprint arXiv:2201.03545*. 2022.

[25] Maryam, B. *Vehicle Detection Using Deep Learning and YOLO Algorithm* [online]. github, september 2021 [cit. 2023-1-15]. Available at: https://github.com/MaryamBoneh/Vehicle-Detection.

[26] Park, J., Woo, S., Lee, J. Y. and Kweon, I. S. BAM: Bottleneck attention module. *British Machine Vision Conference 2018, BMVC 2018*. 2019.

[27] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N. et al. Image transformer. *35th International Conference on Machine Learning, ICML 2018*. 2018, vol. 9, p. 6453–6462.

[28] Sanghvi, K., Aralkar, A., Sanghvi, S. and Saha, I. A Survey on Image Classification Techniques. *SSRN Electronic Journal*. 2021, vol. 5, no. 3, p. 268–273. DOI: 10.2139/ssrn.3754116.

[29] Sharma, S. *How to train CNNs on ImageNet* [online]. Towards Data Science, may 2020 [cit. 2023-1-15]. Available at: https://towardsdatascience.com/how-to-train-cnns-on-imagenet-ab8dd48202a9.

[30] Shokravi, H., Shokravi, H., Bakhary, N., Heidarrezaei, M., Koloor, S. S. R. et al. A review on vehicle classification and potential use of smart vehicle-assisted techniques. *Sensors (Switzerland)*. 2020, vol. 20, no. 11, p. 1–29. DOI: 10.3390/s20113274. ISSN 14248220.

[31] Sochor, J., Herout, A. and Havel, J. BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016, 2016-Decem, p. 3006–3015. DOI: 10.1109/CVPR.2016.328. ISSN 10636919.

[32] Sochor, J., Spanhel, J. and Herout, A. BoxCars: Improving Fine-Grained Recognition of Vehicles Using 3-D Bounding Boxes in Traffic Surveillance. *IEEE Transactions on Intelligent Transportation Systems*. 2019, vol. 20, no. 1, p. 97–108. DOI: 10.1109/TITS.2018.2799228. ISSN 15249050.

[33] Tafazzoli, F. and Frigui, H. Vmmrdb.

[34] Taha, M., H. Zayed, H., E. Khalifa, M. and Nazmy, T. Moving Shadow Removal for Multi-Objects Tracking in Outdoor Environments. *International Journal of Computer Applications*. 2014, vol. 97, no. 10, p. 43–51. DOI: 10.5120/17047-7362.

[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is all you need. *Advances in Neural Information Processing Systems*. 2017, 2017-Decem, Nips, p. 5999–6009. ISSN 10495258.

[36] Woo, S., Park, J., Lee, J. Y. and Kweon, I. S. CBAM: Convolutional block attention module. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2018, 11211 LNCS, p. 3–19. ISSN 16113349.

[37] Yang, L., Luo, P., Loy, C. C. and Tang, X. A large-scale car dataset for fine-grained categorization and verification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 2015, 07-12-June, p. 3973–3981. DOI: 10.1109/CVPR.2015.7299023. ISSN 10636919.

Appendix A

Images Used for Comparison of Models Trained on Different Datasets

(a) Chevrolet Fleet-Master 1

(b) Chevrolet Fleet-Master 2

(c) Dodge Hornet 1

(d) Dodge Hornet 2

(e) Bentley Flying Spur 1

(f) Bentley Flying Spur 2

(g) Ferrari LaFerrari 1

(h) Ferrari LaFerrari 2

(i) Lada Vesta 1

(j) Lada Vesta 2

(k) Peugeot 5008 1

(l) Peugeot 5008 2

(m) Škoda Felicie 1

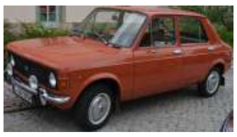
(n) Škoda Felicie 2

(o) Subaru Rex 1

(p) Subaru Rex 2

(q) Volkswagen Transporter 1

(r) Volkswagen Transporter 2


(s) Zastava Skala 1

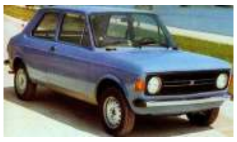
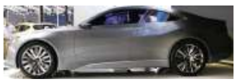

(t) Zastava Skala 2

Figure A.1: Cars used for comparison taken from our dataset.


(a) Shouwang Shouwang 1

(b) Shouwang Shouwang 2

(c) BYN Qin 1

(d) BYN Qin 2

(e) Chrysler Serbing 1

(f) Chrysler Serbing 2

(g) Futian Fengjing 1

(h) Futian Fengjing 2

(i) HongQi H7 1

(j) HongQi H7 2

(k) Lancia Stratos 1

(l) Lancia Stratos 2

(m) MG MG3 1

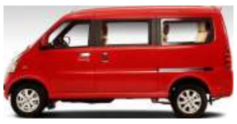
(n) MG MG3 2

(o) Mitsuoka Galue 1

(p) Mitsuoka Galue 2

(q) Quoros 3 1

(r) Quoros 3 2


(s) Shan Fujia 1

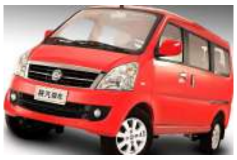

(t) Shan Fujia 2

Figure A.2: Cars used for comparison taken from our CompCars dataset.

(a) Audi S5 1

(b) Audi S5 2

(c) BMW M5 1

(d) BMW M5 2

(e) Buick Regal 1

(f) Buick Regal 2

(g) Cadillac Escalade 1

(h) Cadillac Escalade 2

(i) Dodge Challenger 1

(j) Dodge Challenger 2

(k) Fiat Abarth 500 1

(l) Fiat Abarth 500 2

(m) Ford Focus 1

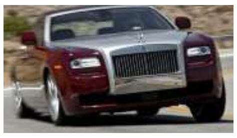
(n) Ford Focus 2

(o) Jaguar XK 1

(p) Jaguar XK 2

(q) MINI Cooper 1

(r) MINI Cooper 2


(s) Rolls Royce Ghost 1

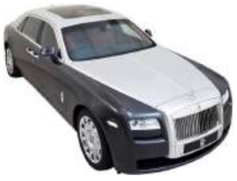

(t) Rolls Royce Ghost 2

Figure A.3: Cars used for comparison taken from Stanford Cars dataset.

Appendix B

Results of Model Comparison

Image					
Label	Bentley Spur	Bentley Spur	Dodge Hornet	Dodge Hornet	Ferrari LaFerrari
Our DS	Bentley Spur	Bentley Spur	DODGE Hornet	DODGE Hornet	Ferrari Laferrari
Stanford	Rolls Royce Ghost *	Bentley Continent *	Suzuki SX4 *	Dodge Journey *	Ferrari California *
ConvCars	Super Sport	Bentley Spur	Dodge Avenger *	Dodge Avenger *	Ferrari LaFerrari

Image					
Label	Ferrari LaFerrari	Chvrlt. FleetMaster	Chvrlt. FleetMaster	Lada Vesta	Lada Vesta
Our DS	Ferrari Fxxk **	CHVRLT Fleetmaster	CHVRLT Stylemstr **	AVTOVAZ Vesta	AVTOVAZ Vesta
Stanford	Ferrari California *	Volvo C30 *	Volvo C30 *	Volvo C30 *	Jeep Compass *
ConvCars	Ferrari LaFerrari	BMW Isetta *	Zhinuo 1E *	Nissan Navara *	King Kong CROSS *

Image					
Label	Peugeot 5008	Peugeot 5008	Škoda Felicia	Škoda Felicia	Subaru Rex
Our DS	Peugeot 5008	Peugeot 5008	SKODA Felicia	SKODA Felicia	SUBARU Rex
Stanford	Chrysler PT *	Jepp Compass *	Dodge CaravanVan*	Suzuku SX4 *	Suzuki SX4 *
ConvCars	Grand Vitara *	Grand Vitara *	Skoda Yeti *	Jiangnan TT *	Jiangnan TT *

Image					
Label	Subaru Rex	VW Transporter	VW Transporter	Zastava Skala	Zastava Skala
Our DS	SUBARU Justy	VW Type 2	VW Transporter	ZASTAVA Skala	FIAT 128
Stanford	VW Golf *	VW Golf *	M-Benz Sprint *	Volvo C30*	Volvo C30 *
ConvCars	Jiangnan TT *	Jiabao V80 *	VW Defender *	FIAT 126P *	FIAT 126P *

Image					
Label	BYD Quin	BYD Quin	Fujitan Fengjing	Fujitan Fengjing	HongQi H7
Our DS	Byd Qin	Byd Qin	TOYOTA Hiace *	TOYOTA Hiace *	FAW H7
Stanford	Acura TL Types-S *	BMW M3 *	M-Benz Sprinter *	GMC Savana Van *	BMW X6 *
ConvCars	BYD Quin	BYD Quin	Fujitan Fengjing	Fujitan Fengjing	HongQi H7

Image					
Label	HongQi H7	Chrysler Serbing	Chrysler Serbing	Lancia Stratos	Lancia Stratos
Our DS	FAW H7	CHRYSLER Serbing	CHRYSLER Serbing	PININFAR. Stratos	PININFAR. Stratos
Stanford	Acura RSX *	Audi RS 4	Chrysler Serbing	Ford GT Coupe *	Fisker Karma *
ConvCars	HongQi H7	Chrysler Serbing	Chrysler Serbing	Lancia Stratos	Lancia Stratos

Image					
Label	MG MG3	MG MG3	Mitsuoka Galue	Mitsuoka Galue	Quoros 3
Our DS	MG Mg3	SAIC Mg3	Mitsuoka Galue	Mitsuoka Galue	Quoros 3
Stanford	Suzuki SX4 *	VW Golf *	Ford Mustang *	Chrysler PT Cruiser *	Acura TL Type-S *
ConvCars	MG MG3	MG MG3	Mitsuoka Galue	Mitsuoka Galue	Quoros 3

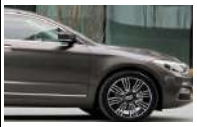
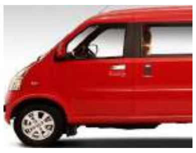
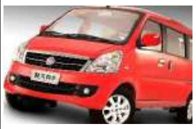
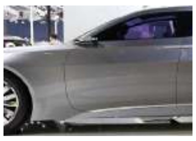
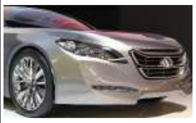
Image					
Label	Quoros 3	Shan Fija	Shan Fija	Shouwan Shouwan	Shouwan Shouwan
Our DS	Quoros 3	GAC Saboo *	DONGFENG Succe *	Joylong a-series *	Hyundai Genesis *
Stanford	BMW M5 *	M-Benz Sprinter *	Suzuki SX4 *	Audi TT *	Hyundai Genesis *
ConvCars	Quoros 3	Shan Fija	Shan Fija	Shouwan Shouwan	Shouwan Shouwan

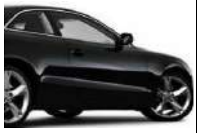
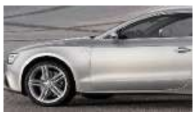
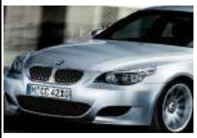
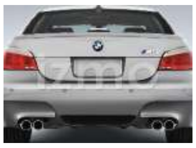
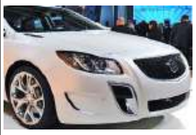
Image					
Label	Audi S5	Audi S5	BMW M5	BMW M5	Buick Regal
Our DS	Audi A5	Audi S5	BMW M5	BMW M5	BUICK Regal
Stanford	Audi S5	Bentley Continental	BMW M3	Mitsubishi Lancer	BMW M3
ConvCars	Audi A5	Audi S5	BMW M5	Lorinser E Class	BUICK Regal

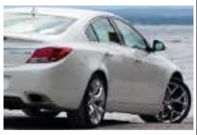
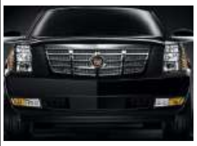
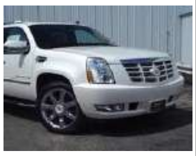
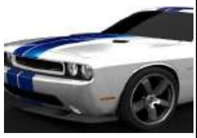
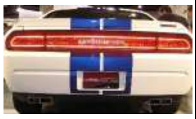
Image					
Label	Buick Regal	Cadillac Escalade	Cadillac Escalade	Dodge Challenger	Dodge Challenger
Our DS	BUICK regal	GM Cadillac Escalade	GM Cadillac Escalade	Dodge Atitude	Dodge Charger
Stanford	Mitsubishi Lancer	Dodge Journey	Cadillac Escalade	Dodge Challenger	Dodge Challenger
ConvCars	Vauxhall Corsa	Cadillac Escalade	Cadillac Escalade	Dodge Challenger	Dodge Challenger

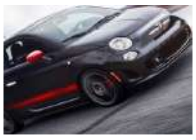
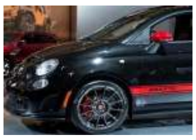
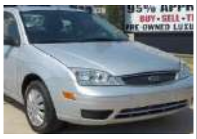
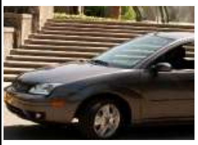
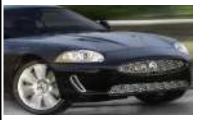
Image					
Label	Fiat Abarth 500	Fiat Abarth 500	Ford Focus	Ford Focus	Jaguar XK
Our DS	ABARTH 500	ABARTH 500	Ford Focus	Ford Focus	JAGUAR XK
Stanford	Fiat Abarth 500	Fiat Abarth 500	Ford Focus	Ford Focus	Bentley Continental
ConvCars	Fiat Abarth 500	Fiat Abarth 500	Ford New Focus	Chrey A5	Gaugar XK

Image					
Label	Jaguar XK	Mini Cooper	MINI Cooper	Rolls Royce Gost	Rolls Royce Ghost
Our DS	JAGUAR xk	Mini Coupé	Mini Coupé	ROLLS ROYCE Ghost	ROLLS ROYCE Ghost
Stanford	BMW M3	Mini Cooper	Mini Cooper	Rolls Royce Ghost	Rolls Royce Ghost
ConvCars	Gaugar XK	MINI ROADSTER	MINI ROADSTER	Rolls Royce Ghost	Rolls Royce Ghost

Table B.1: Results of comparisons of models trained on different datasets when clasifing images from various datasets. Legend: Green/Orange – The label model was in the 1/5 outputs of model with highest confidence. Purpe – The output of model with the highest confidence had the same manufacturer as the label model. Red – labeled model was not in any previous cathegory. * — the label model was not in the training dataset of the VMMR model. ** – the highest confidence value differed from the label model, however it was correct (error in label).