

University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Bachelor's thesis

Automatic vehicle make and model recognition

Assignment

1. Get introduced with the issue of automatic vehicle make and model recognition and describe the current status of the solution.
2. Prepare a suitable image dataset for training and testing the classifier for the issue.
3. Based on a thorough analysis, choose the appropriate classification method, implement the method.
4. Find a suitable classifier topology and demonstrate this by experimenting on the dataset created.
5. Discuss the results and suggest additional extensions.

Declaration

I hereby declare that this bachelor's thesis is completely my own work and that I used only the cited sources.

Plzen, May 6, 2020

Radek Juppa

Abstrakt

Tato práce se věnuje problematice automatického rozpoznávání výrobce a modelů vozidel. Detailně popisuje postup výroby datasetu z veřejných zdrojů. Pro klasifikaci jsou použity konvoluční neuronové sítě. Práce srovnává výsledky experimentů prováděných na sítích ResNet-50 a VGG-16. Experimenty jsou implementovány v jazyce Python s využitím knihovny Keras/TF. Nejlepší dosažené výsledky jsou otestovány v reálné situaci. Závěrem jsou publikovány návrhy na vylepšení.

Abstract

This work describes the problems of vehicle make and model recognition. It presents in detail the creation of a suitable image dataset from public web resources. Extensive experiments are conducted to find suitable architecture of convolutional neural network. Great results were achieved by using CNN models based on ResNet-50 and VGG16. The work is implemented in Python using Keras/TF. In the conclusion, the actual proposal is evaluated and compared with real world scenarios. Further improvements are proposed.

Keywords: Vehicle Make and Model Recognition, VMRR, Convolution Neural Network, CNN, Keras, SVM, Augmentation

Acknowledgement

First of all, I would like to thank my research supervisors Mr Pavel Král and Mr Hruz Marek, without their assistance and guidance in every step throughout the process, this paper would have never been accomplished.

I wish to express my sincere thanks to MetaCentrum.cz for providing me with all the necessary facilities for the research.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

Contents

1	Introduction	1
2	Analysis	3
3	Theory	5
3.1	Linear SVM	5
3.2	Convolution	5
3.3	Neural Network	5
3.4	Convolution Neural Network	6
3.4.1	Convolution Layer	6
3.4.2	Pooling Layer	6
3.4.3	Fully Connected Layer	7
3.4.4	Dropout Layer	7
3.4.5	Batch normalization	7
3.4.6	Regularization	7
3.5	VGG-16	8
3.6	Resnet-50	9
4	Current State	10
4.1	CIFAR-10/100	10
4.2	ImageNet	11
4.3	CompCars	11
4.4	VMMRdb	11
4.5	Stanford Car Dataset	12
4.6	Summary	13
5	Building a Dataset	14
5.1	Web Scraping	14
5.2	Data Cleaning	15
5.3	Raw Data Statistics	16
5.4	Dataset for Car Make Recognition	17
5.4.1	Data Augmentation	18
5.5	Dataset for Car Model Recognition	21
5.6	Dataset for Year of Production Estimation	22

6 Experiments	23
6.0.1 The Best Topology for VGG16	24
6.0.2 The Best Topology for ResNet-50	26
6.1 Final Training	27
6.1.1 Car Make Recognition	27
6.1.2 ResNet-50 with the Final Dataset	28
6.1.3 VGG-16 with the Final Dataset	30
6.1.4 Car Model Recognition	32
6.1.5 Estimation of a Year of Production	33
6.2 Testing with Online Images	35
6.3 Testing with Real Photos	37
6.4 Analysis of Results	40
6.5 Conclusion	44
Appendices	45
A Content of DVD	46
B User Manual	47
B.1 Installation	47
B.2 Run Program	47

1 Introduction

Vehicle Make and Model Recognition (VMMR) is a part of image classification problem. The basic idea is to recognize make and model of a vehicle using suitable features extracted from the images of a vehicle. It is not a new area of interest. It come with the need of various access control systems for buildings, outdoor sites or traffic control systems.[1] The first systems provided just a basic detection of a vehicle or were able to identify whether a vehicle is a bus, truck or car [12]. Modern systems are able to identify make and model of vehicle with high accuracy. Many of VMMR systems are designed very specific, analyzing an frontal image of a vehicle from fixed point of view against road environment background. For example Petrović [13] states their system is capable of 93% accuracy tested on 1000 images containing 77 classes. Their algorithm locates a region of interest (ROI) for feature extraction. Obtained feature vector is classified using a nearest neighbour algorithm. VMMR is a recognition of very similar objects with relatively small differences. It makes it more challenging problem compared to other image classification problems. This type of problem is often referred to as fine-grained classification [8].

The recognition problem considered in this paper is more general. We recognize a car from any horizontal angle in any background. Our approach is based on deep convolutional neural networks. We distinguish 40 main manufacturers and their appropriate car models. In that way it is more challenging problem that can offer improvements in identification of vehicles. It can be used as an enhancement of License Plate Recognition (LPR) systems. LPR systems rely on license plates to identify a car. However, they fail in a situation where two license plates are illegally swapped. Here presented system provides additional information such as name of manufacturer of car, name of car model and estimated year of production even in a situation where no license plate is visible.

This work is structured as follows:

- **Analysis**

We describe what kind of problem we are facing to and propose solutions.

- **Theory** This section is a short recap of important terms and algorithms in machine learning field.

- **Current state**

A large dataset is required for training a CNN for an image classification task. Here we walk through the most interesting public datasets discussing their pros and cons.

- **Building a Dataset**

In this section we show how to compile own dataset from public web resources. We cover web scraping, data cleaning and balancing of dataset.

- **Experiments** We describe many experiments we conducted in order to find best topology of neural network. We cover two commonly used CNN: VGG16 and ResNet50. Both bring great results.

- **Results** And finally we present our results and discussing conditions. We also discuss testing in real world situation and proposing further improvements.

2 Analysis

To recognize a car make or a car model is a classification task. It is a supervised learning approach where labeled input data are provided. As there are many categories (car makes and car models) it implies multi-class classification. Our task is to find a mapping function from images of cars to discrete output categories - their appropriate car make. In machine learning, we have many algorithms suitable for classification. To choose right approach we need to consider what data we have. Approaches like decision tree or random forest are based on features or attributes. Our data are images. We can think about image as two dimensional vector of features. It implies a huge number of features. The most effective tools for image recognition is a deep neural network, especially convolutional neural network (CNN). They got the best results in international competitions, as stated bellow in Wikipedia article.

The resulting annual competition is now known as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The ILSVRC uses a "trimmed" list of only 1000 image categories or "classes", including 90 of the 120 dog breeds classified by the full ImageNet schema. The 2010s saw dramatic progress in image processing. Around 2011, a good ILSVRC classification error rate was 25%. In 2012, a deep convolutional neural net called AlexNet achieved 16%; in the next couple of years, error rates fell to a few percent. While the 2012 breakthrough "combined pieces that were all there before", the dramatic quantitative improvement marked the start of an industry-wide artificial intelligence boom. By 2015, researchers at Microsoft reported that their CNNs exceeded human ability at the narrow ILSVRC tasks. [19]

Generally we can say our output depends on our input. It means to get good results with CNN we need a great dataset. We will pay high attention to build our dataset.

A good source of images can be a local traffic control and surveillance systems. Unfortunately, we don't have access to such a system. So we can scrape images from the internet. We can use well structured websites like used car sellers. They usually expose a great number of cars to sell. Every car is accompanied with multiple images and its brief description. It is important to scrape more information about a car for creating labels. A disadvantage of web scraping is that we get data dirty. Some images show the interiors of cars. They are not suitable for our purpose. We need to pull them out. This process is called data cleaning. The final dataset should be clean distinct classes(car makes) of labeled images. When we got our dataset we are ready to start to train our neural net-

work. We decided to compare two popular CNN. The first is a simple neural network called VGG16 and the second one is a deep residual neural network called ResNet-50. We try to get the best results of both and compare them.

We conduct several experiments. The first is the recognition of car make. Then we pick a car make and we try to determine the car model and the last experiment is estimating a year of production. For every task, we need a separate dataset. We will need 3 separated datasets to conduct our experiments.

3 Theory

This section is a short recap of important terms and algorithms in machine learning field.

3.1 Linear SVM

Support Vector Machine - SVM [6] is a method of machine learning solving a binary classification problem. It searches an optimal division hyperplane in high dimensional feature space. Every image is projected as a set of features. In our case, the hyperplane divides

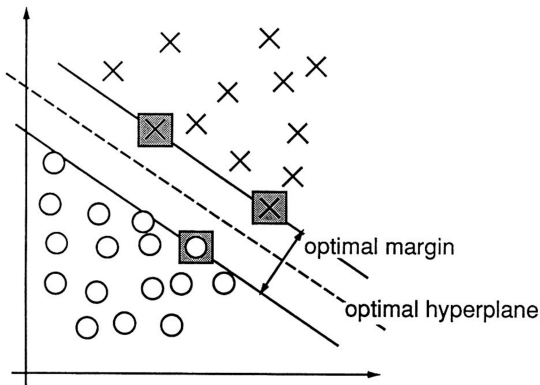


Figure 3.1: SVM [6]

all images into two distinct groups - interior or exterior images. Here is shown simplified version in 2D space in figure 3.1. SVM finds an optimal hyperplane. It means that the distance of the two nearest opposite samples is as big as possible. That's why it is sometimes called *Large Margin Classifier*. It is also why this method is suitable for our problem. Another good feature is that it works with a small number of samples.

3.2 Convolution

A convolution [16] is a mathematical operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other.

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\alpha)g(x - \alpha)d\alpha \tag{3.1}$$

3.3 Neural Network

A neural network (NN) [4] is a computational graph. It is a collection of connected nodes called neurons. Each neuron can receive an input signal. It combines the input

with its internal state using an activation function and produces output using an output function. It can pass output to other connected neurons. Neurons are organized in

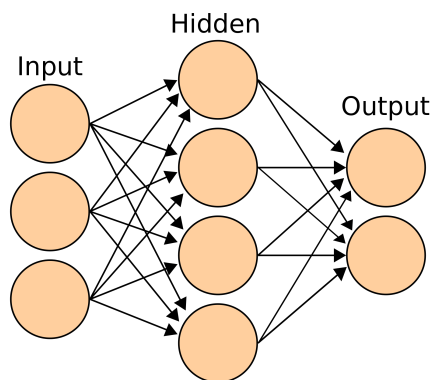


Figure 3.2: Neural network [5]

layers as we can see in Figure 3.2. For neural networks it is essential they are adaptive systems. Neurons are able to change their internal state (weights) depending on computational process. They use *backpropagation* algorithm [18]. It works with an *error function* that is computed on the output layer and propagated back to the previous layers to adjust weights of neurons to minimize the error in the next round. Modern neural networks are able to compute very complex hypothesis.

3.4 Convolution Neural Network

Convolutional neural networks (CNN) are a special kind of neural networks. They contain several convolutional layers. A convolutional layer is able to successfully capture the spatial dependencies in an image through the application of relevant filters. The filters are composed of weights. A convolution is calculated between the filters and the input data underneath. It works like a pattern matcher. In lower layers, general edges or patterns are matched. In deeper layers, more complex shapes are captured. That's why convolution takes place in image recognition. Convolutional layers create a map of features.

3.4.1 Convolution Layer

A convolution layer is one type of layers in CNN. It creates a map of features. In lower layers, general edges or patterns are matched. In deeper layers, more complex shapes are captured. They are often organized in blocks.

3.4.2 Pooling Layer

A pooling layer reduces dimension. It helps to reduce a noise. There are max pooling and average pooling depending on reduction.

3.4.3 Fully Connected Layer

A fully connected layer consists of neurons. Every neuron is connected with all neurons of the previous layer. A block of fully connected layers is called classifier and it is placed on the top of the network. Count of neurons in the last fully connected layer is equal to the count of classes for classification problems.

3.4.4 Dropout Layer

A dropout layer is a regularization method. It removes weights of random neurons in a layer. We lose some random pieces of information to improve independency of particular neurons that increases generalization.

3.4.5 Batch normalization

Batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. It speeds up learning.

3.4.6 Regularization

Regularization increases generalization. It means it can improve the generalization performance on new, unseen data. It prevents from overfitting of a model. It is based on penalization of complexity.

3.5 VGG-16

VGG-16(Visual Geometry Group) [15] is a convolutional neural network consisted of 12 convolutional layers and 4 fully connected layers. It uses a default 1000 classes softmax classifier for ImageNet dataset.

	Layer	Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Figure 3.3: VGG-16 Architecture

3.6 Resnet-50

Resnet-50 [10] is a residual convolutional neural network with a deep of 50 layers. It is deeper than VGG-16, but still having lower complexity. It won the 1st place on the ILSVRC 2015 classification task.

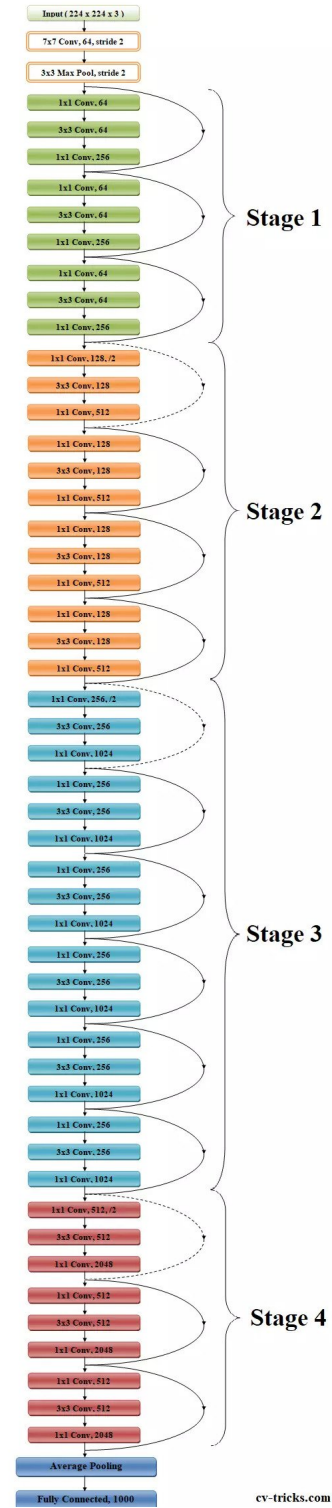


Figure 3.4: ResNet-50 Schema

[9]

4 Current State

Based on the previous analysis we choose CNN for recognition in this thesis. A large dataset is required to utilize CNN with supervised learning. There are many solutions related to this problem on the internet including interesting datasets. But they usually have some limits; e.g. the dataset is small or it is too general or cars are from a different continent etc. Let's go through some solutions to have a closer look.

4.1 CIFAR-10/100

Dataset CIFAR-10 [1] was created in 2010 and it includes 60.000 of small images 32x32 divided into 10 classes. Every class has 6000 samples. One class is an *Automobile*. It covers all type of vehicles. This dataset is too general for VMMR. There is another similar dataset CIFAR-100, which has 100 classes. Every class has 600 samples. But it has the same problem. It contains animals, plants, etc. This dataset is very general and it is not suitable for VMMR.

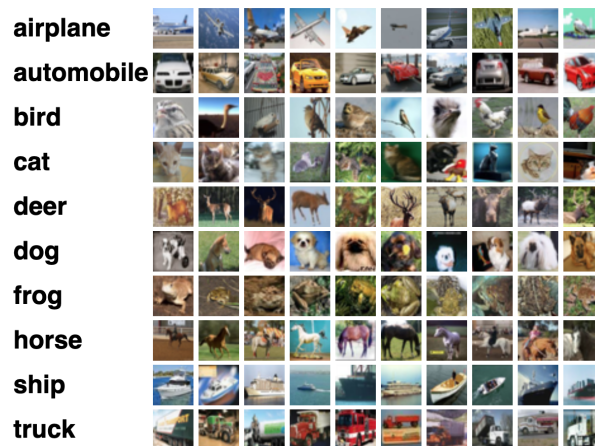


Figure 4.1: CIFAR-10

4.2 ImageNet

ImageNet [7] is a huge and very popular dataset of images categorized into 1000 classes. It was designed specially to support computer vision and it contains more than 14 million hand-annotated pictures. Although it was presented for the first time in 2009, it is still very popular and becomes a referral dataset for CNN. It contains category *vehicles*, where we can find cars, but they are not annotated with appropriate makes. This dataset is not suitable for VMMR task as it is too general.

4.3 CompCars

In 2015 Linjie Yang, Ping Luo, Chen Change Loy, Xiaoou Tang published work *A Large-Scale Car Dataset for Fine-Grained Categorization and Verification* [21]. Samples from this dataset come from two sources: from the web and from surveillance. There are more than 130 000 the entire images of cars and more than 25 000 images capturing the car parts. It covers 163 car makes with 1713 car models, mostly from North America. This dataset is a good choice for a VMMR task.

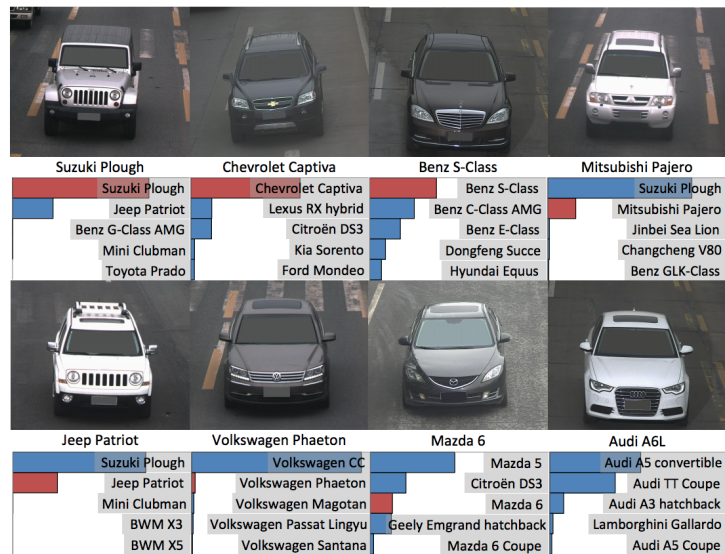


Figure 4.2: CompCar

4.4 VMMRdb

In 2017 F. Tafazzoli, K. Nishiyama and H. Frigui published own work *A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition* [17]. This paper is based on av-dataset created by web crawling from online resources like amazon.com and craigslist.com.

It contains more than 290,000 samples of cars divided into 9170 classes. Mostly cars from the USA. The dataset is well balanced and described. The author used a pre-train neural network ResNet50. VMMRdb is very good dataset for VMMR task.



Figure 5: Top-5 predicted classes of ResNet-50 for sample images from VMMRdb-3036 classified correctly by the model



Figure 4.3: VMMRdb

4.5 Stanford Car Dataset

Stanford car dataset [11] comes from 2013. It contains more than 16,000 of cars in 196 classes. It needs to mention this dataset was a part of *Large Scale Visual Recognition Challenge 2013 FGComp2013*. They reached 75% accuracy, later in 2018 reached 93.9%. Stanford Car dataset is relatively small but can be sufficient for a simple VMMR task.



Figure 4.4: Stanford Car Dataset

4.6 Summary

The biggest problem of above mentioned datasets is their geographical limit. Most of the car makes comes from the USA or Asia. Each continent has its own specific subgroup of car makes and car models. Since we want to test our results in real-life scenarios in central Europe, we definitely need to compile our own dataset.

5 Building a Dataset

The dataset is the most important part of our effort. Our results are directly depending on it. That's why we need to pay attention to build good dataset. We set several conditions for our dataset:

- dataset should cover most of all local car makes
- every car make should be represented approximately by the same number of samples
- every car make should be exposed from many angles (as long as we want to recognize a car make from the horizontal angle 360 degrees)
- generally more samples produce better results
- all samples must be labeled

We use the internet as our source of input data. The biggest suppliers of online car images are big used car dealers. AAAAuto.cz offers thousands of cars and every car has many high quality photographs with a neutral background on their website. There is enough information about every car like name of make, model and year of production. It makes it a good fit for our needs. There is one disadvantage, photographs contain also images of the interior of cars. Interiors are not good samples for our VMMR task as we want to recognize the shape of the car. We need to filter car interiors out.

5.1 Web Scraping

Web scraping is a technology helping download a large number of internet resources. In our case we downloaded images of cars. It is basically a program for downloading web resources, parsing its content, searching for links to related further resources and downloading them and the process is repeated. We used a Python library *Scrapy* [20] for downloading images of cars. It is important to mention that data is well structured on website AAAAuto.cz. This allows us download images by manufacturer and save it to folder with its name and name of its model and even year of production of the car. That means we scraped labeled data and stored them in hierarchical directory structure */make/model* which is very convenient for next processing.

5.2 Data Cleaning

As mention above there are images of car interiors in the dataset as we can see in Figure bellow 5.1. We need to separate these interior images from the rest of dataset. This is a binary classification task.

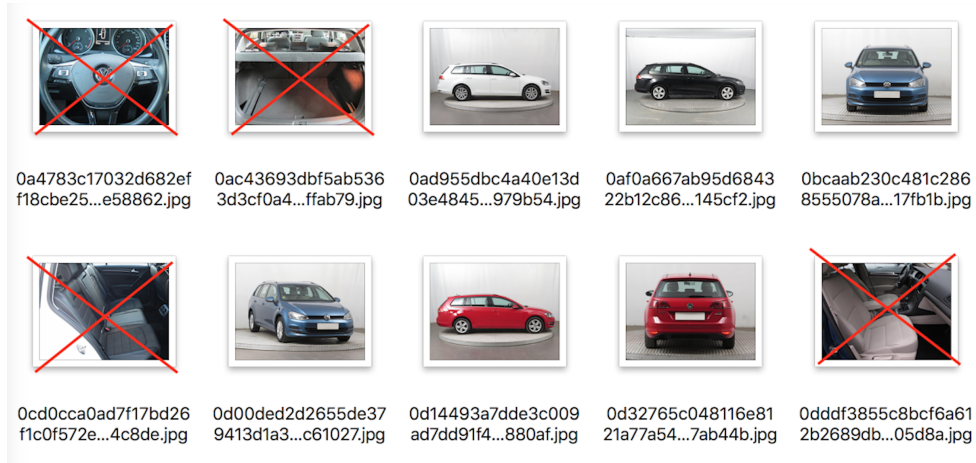


Figure 5.1: Raw Dataset polluted by car interiors

We used *Support Vector Machine (SVM)* 3.1 for making a decision whether a sample is an interior or not. We manually prepared data for two classes. We separated 1000 images with interior and 1000 images without interior (just a car). Then we used CNN ResNet-50 to process images to create a feature vector. We used output of the last convolutional layer as our feature vector as shown in Figure 5.2.

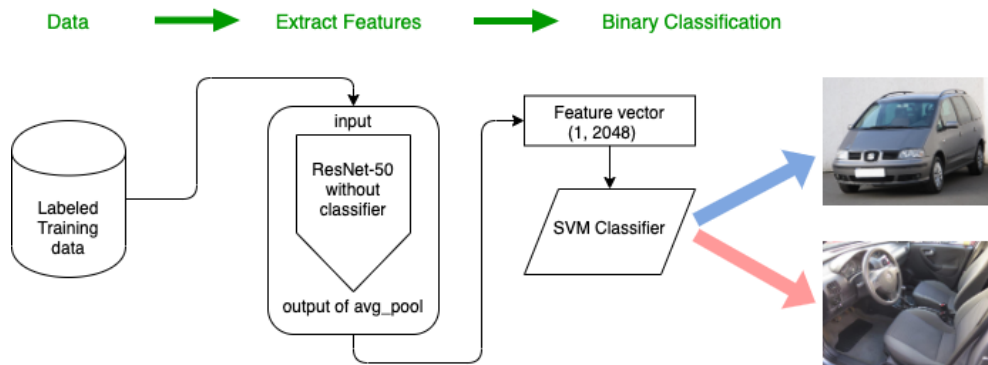


Figure 5.2: SVM Classifier

We trained *SVM* classifier with 800 of samples with high accuracy. Then we used this model to separate the rest of images. The result was perfect. All interiors were removed.

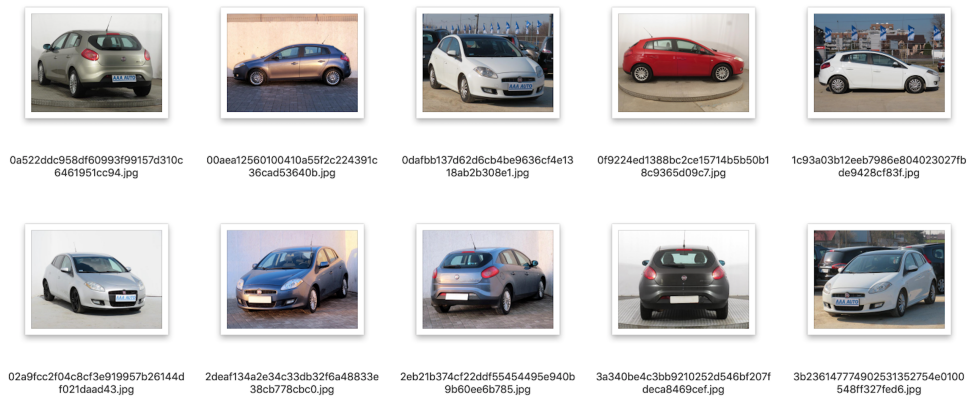


Figure 5.3: Cleaned Dataset

5.3 Raw Data Statistics

Here are some raw numbers describing our dataset:

- downloaded data from aaaauto.cz 7/2018 total 17GB
- downloaded data from aaaauto.cz 12/2018 total 18GB
- downloaded data from aaaauto.sk 3/2019 total 11GB
- downloaded data from aaaauto.pl 3/2019 total 5GB
- downloaded data from aaaauto.hu 3/2019 total 2GB
- downloaded data from aaaauto.cz 3/2019 total 7GB

After cleaning

- cleaned dataset total 18.5GB
- 58 car makes
- 331,000 labeled images
- size of images: 640x480 and 1024x768 pixels

The biggest problem of our dataset is its unbalanced distribution. As we can see in Figure 5.4 some car makes have tens of thousands of images, but there are few car makes with counts under 50 samples. We removed all classes smaller than 50 samples.

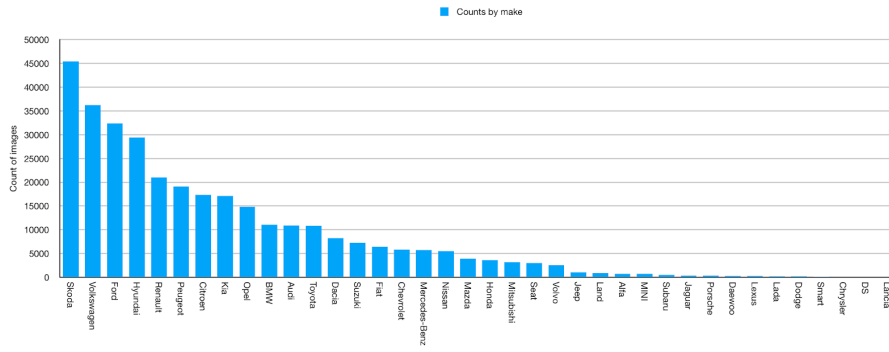


Figure 5.4: Distribution of samples in dataset

Here is a distribution of a year of car production:

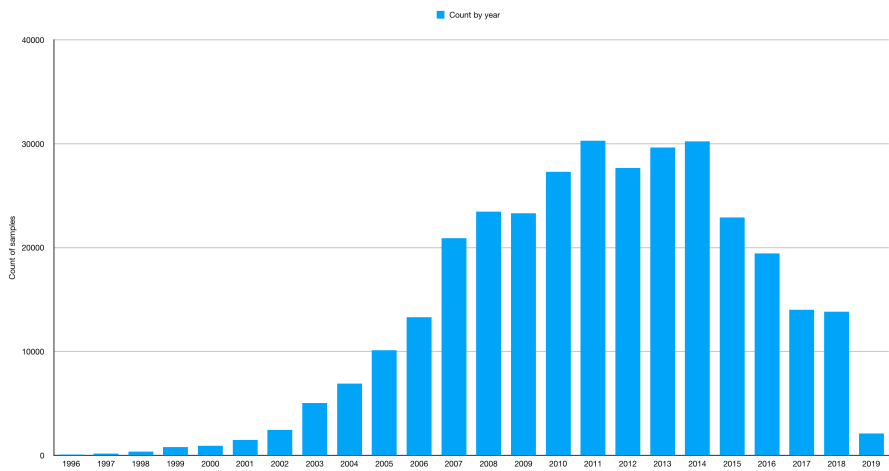


Figure 5.5: Distribution of Year of Production

There is another problem with particular car models. Every manufacturer produces several models. They are distinguished by their model name. A model name can have variations. For example cars *Volkswagen Golf* and *Volkswagen Golf Plus* and *Volkswagen Golf Sportvan* They are considered as a single model *Volkswagen Golf*. A model is determined by the first word after car make. All variations are merged.

5.4 Dataset for Car Make Recognition

In Figure 5.4 we can see that a cleaned dataset is very unbalanced. There are 17 classes with a total count under 50 samples. We skipped these "small" classes. We created a dataset with 41 bigger classes. The smallest class has 55 samples. The average count of samples in every class in the final dataset is designed to be 5000. There are 18 classes with a total count over 5000. The rest 23 classes have a total count of less than 5000 and

higher than 50. We used augmentation to increase number of samples of these classes up to 5000.

5.4.1 Data Augmentation

Data augmentation is a method for increasing the count of samples in classes with a small number of samples. The idea behind augmentation is to take an original picture and perform small adjustments to create a new image that is slightly different from the original but still has all features of the original class. These adjustments are i.e.: vertical or horizontal shifts, rotations, changing light or color, etc. There are two ways how to implement augmentation. The first is before training. It increases the count of samples in small classes. The second is during training. It just uses augmented images during training epochs. The second one is implemented in library *Keras* [3] in class *ImageDataGenerator*. Figure fig:augmentation shows examples of random shift and rotation.



Figure 5.6: An example of augmentation

We used both ways of augmentation, but we used different methods in every phase. In *Keras* we used geometric transformations like shifts, rotations and flips because these adjustments are suitable for all data. That means all data entering in our network are augmented data.

For "before training augmentation" we used library *OpenCV* [2] to adjust images by script. We implemented a few filters:

```
1 filters = ["lighter", "saturation", "blur", "invert", "contrast",  
2           "affinet", "grayscale", "hist", "foggy", "rainy", "drops"]
```

Here is an example of augmentation using *openCV* library. The first picture is an original.



Figure 5.7: Augmentations with OpenCV

The final dataset is called **UWB-VMMR-5000**. It contains 41 classes. The dataset is well balanced as shown in Figure 5.8. The average count of samples in every class is 5000.

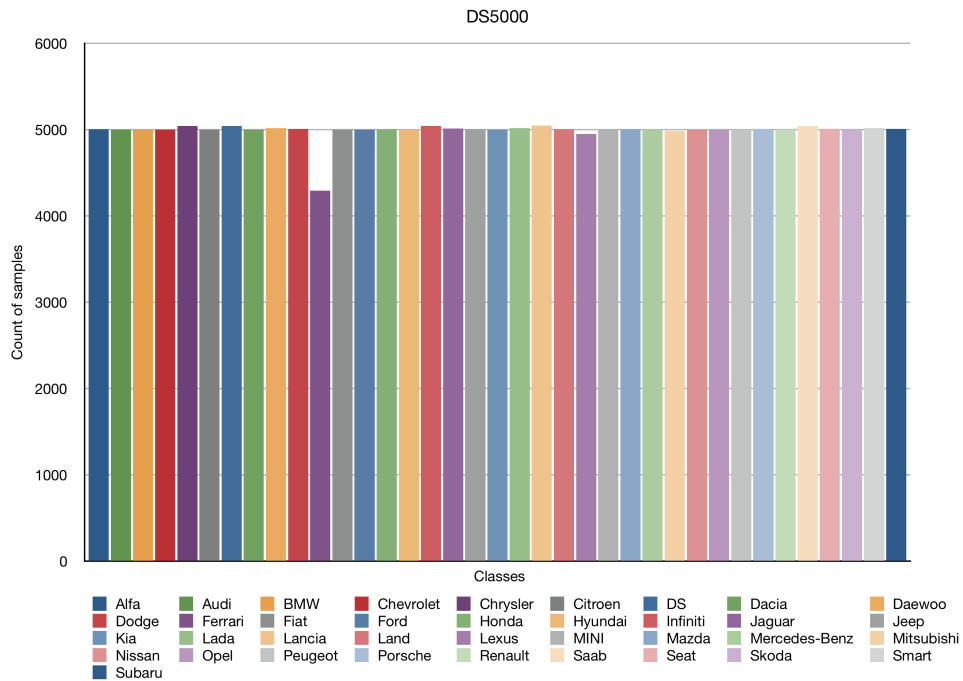


Figure 5.8: The final dataset UWB-VMMR-5000

5.5 Dataset for Car Model Recognition

For recognition of car models, we chose the manufacturer SKODA, because it is the class with the highest number of samples in our dataset. They have 10 models referenced in our dataset with high quantity. The models are: Citigo, Fabia, Favorit, Felicia, Karoq, Kodiaq, Octavia, Rapid, Roomster, Superb, Yeti. We created dataset **SKODA-5000** with 10 classes shown in Figure 5.9. Every class has an average 5000 samples. We used data augmentation the same way as in the previous dataset *UWB-VMMR-5000*.

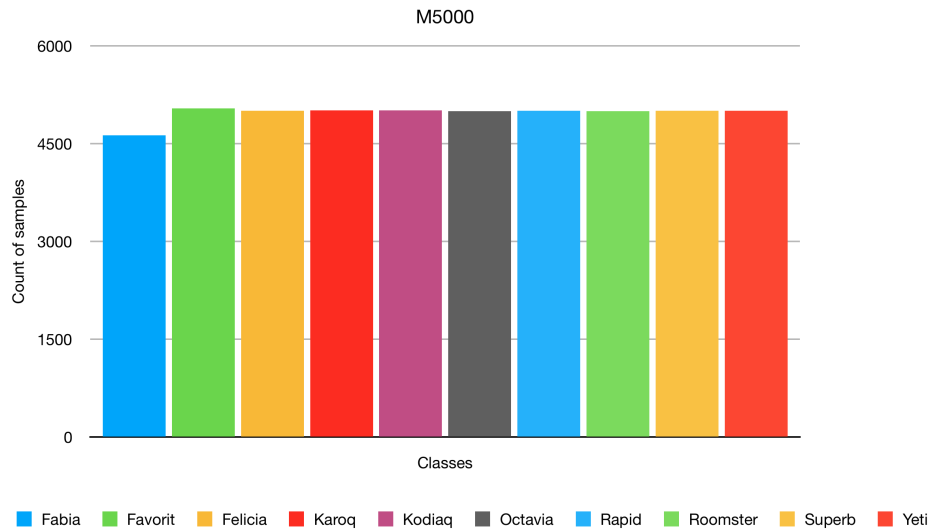


Figure 5.9: The final dataset SKODA-5000

5.6 Dataset for Year of Production Estimation

In our raw dataset, we have labeled data. Every sample is labeled with car make, car model and year of production. We chose manufacturer SKODA, model OCTAVIA as most frequented model and we built a new dataset called **OCTAVIA-2000** from it as shown in Figure 5.10. There are models since 1997 till 2018. These are 22 numeric values (years) for SKODA OCTAVIA. Because the values are numeric, it is more accurate to use regression than classification. Every value has an average of 2000 samples. We used the same data augmentation method as in the previous case to balance the dataset.



Figure 5.10: The final dataset OCTAVIA-2000

6 Experiments

Our goal is to find the NN architecture producing correct predictions. We can measure correctness of our trained model as a fraction of predictions our model got right divided by total predictions. We used standard accuracy. For this purpose we started with a minimal test dataset, named *Sample5*, because computing with full dataset is very time consuming task. First we find good performing architecture of CNN and than we use this architecture for computing model with full dataset. The minimal dataset has just 5 classes reflecting 5 car models: *Audi A3*, *Citroen C3*, *Ford Focus*, *Opel Astra*, *Toyota Raw*. Notice they are car models not car makes. Every class has 2000 samples. Data was split between training data and validation data with rate 80/20. We proceed several test with *VGG16* and *ResNet-50* upon dataset *Sample5* to find out how they behave depending on its parameters. Results of experiments are depicted in the table 6.1.

ID	CNN	ImageNet	Optimizer	Classifier	Val. Acc.
s50	VGG16	yes	SGD	5	20.3
s51	VGG16	no	SGD	4096 x 4096 x 5	98.7
s52	VGG16	yes	SGD	4096 x 4096 x 5	99.1
s52a	VGG16	yes	Adam	4096 x 4096 x 5	20.9
s52r	VGG16	yes	RMS	4096 x 4096 x 5	99.9
s53	VGG16	yes	SGD	4096 x 256 x 5	20.0
s54	VGG16	yes	SGD	4096 x 4096 x 256 x 5	99.7
s55	ResNet	yes	SGD	5	98.0
s56	ResNet	yes	SGD	4096 x 4096 x 5	92.7
s56n	ResNet	no	SGD	4096 x 4096 x 5	53.6
s56a	ResNet	yes	Adam	4096 x 4096 x 5	97.8
s56r	ResNet	yes	RMS	4096 x 4096 x 5	22.5
s57	ResNet	yes	SGD	4096 x 5	98.7
s58	ResNet	yes	SGD	4096 x 4096 x 256 x 5	97.9

ID - experiment ID

CNN - type of CNN

ImageNet - classifier pre-trained on ImageNet

Optimizer - method of optimization

Classifier - structure of classifier

Val.Acc. - validation accuracy

Table 6.1: Results of experiments

The parameters we tuned are:

- number of full-connected layers of classifier
- using a pre-trained weights of ImageNet or not
- changing optimizer (SGD, Adam, RMSprop)

The classifier consists of several full-connected layers and always end up with 5 neurons in this case. This is because we have just 5 classes reflecting 5 car models in our dataset.

We can see some configurations perform better than others. We can see it is better to use a pre-trained CNN. SGD optimizer performs well in both networks, furthermore RMS is even better with VGG in this dataset. It is better to use simple classifier for ResNet than VGG. All these results were used for final training later.

6.0.1 The Best Topology for VGG16

We conducted several experiments in order to tune parameters of the network as you can seen in table 6.1. VGG16 pre-trained on ImageNet with RMS optimizer and original classifier layers brought the outstanding results.

Test ID: s52r

```
1 model = VGG16(include_top=True, weights="imagenet")
2 model.layers.pop()
3 x = model.layers[-1].output # add last layer
4 x = Dense(len(classes), activation="softmax")(x)
5 finetuned_model = Model(model.input, x)
6 finetuned_model.summary()
7 opt = RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)
8 finetuned_model.compile(optimizer=opt,
9                          loss='categorical_crossentropy',
10                         metrics=['accuracy'])
```

We used the default setting of RMSprop optimizer with lr=0.001. It reached 99.9% validation accuracy! This is the best result at all.

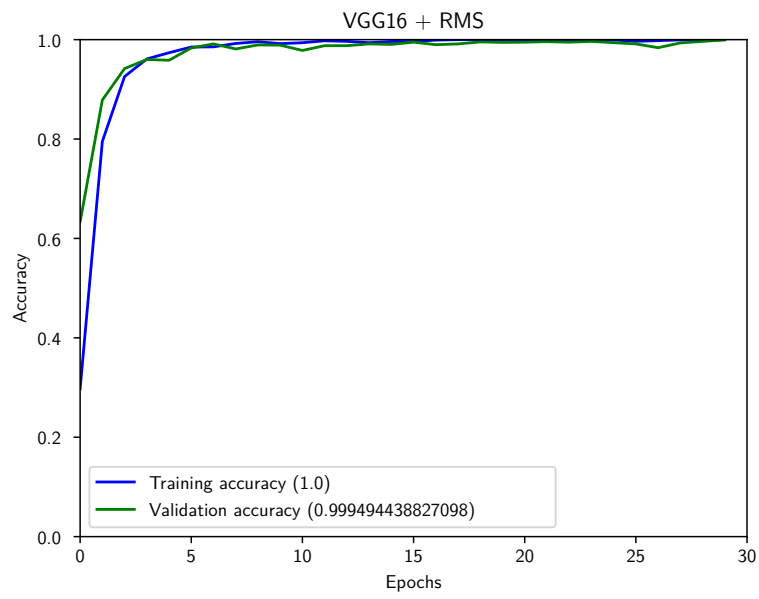


Figure 6.1: Accuracy

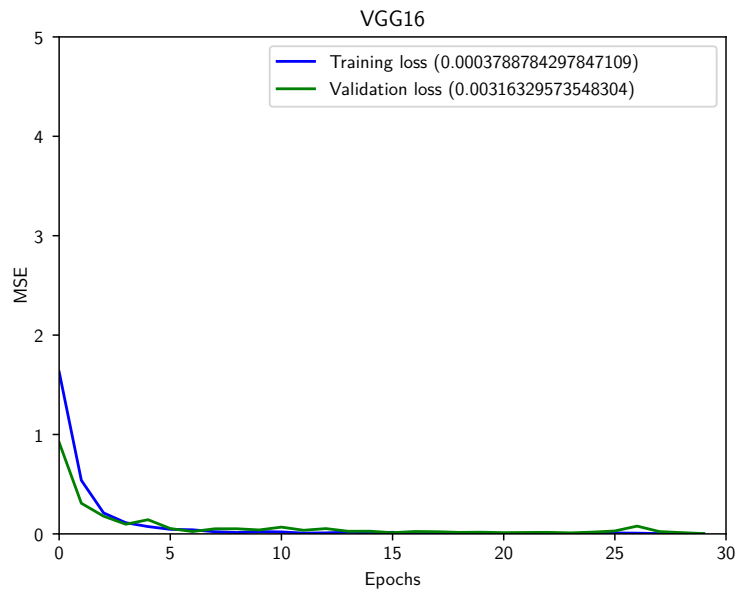


Figure 6.2: Loss

6.0.2 The Best Topology for ResNet-50

Same approach was applied with ResNet-50. Results are also described in table 6.1. ResNet-50 pre-trained on ImageNet with SGD optimizer and simplified classifier shows the best results in this case. We added 1 fully connected layer and the final 5-neurons layer on top of it.

Test ID: s57

```
1 model = ResNet50(include_top=True, weights="imagenet")
2 model.layers.pop()
3 x = model.layers[-1].output # add last layer
4 x = Dense(4096, activation="relu")(x)
5 x = Dense(len(classes), activation="softmax")(x)
6 finetuned_model = Model(model.input, x)
7 finetuned_model.summary()
8 opt = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
9 finetuned_model.compile(optimizer=opt, loss='categorical_crossentropy',
10                        metrics=['accuracy'])
```

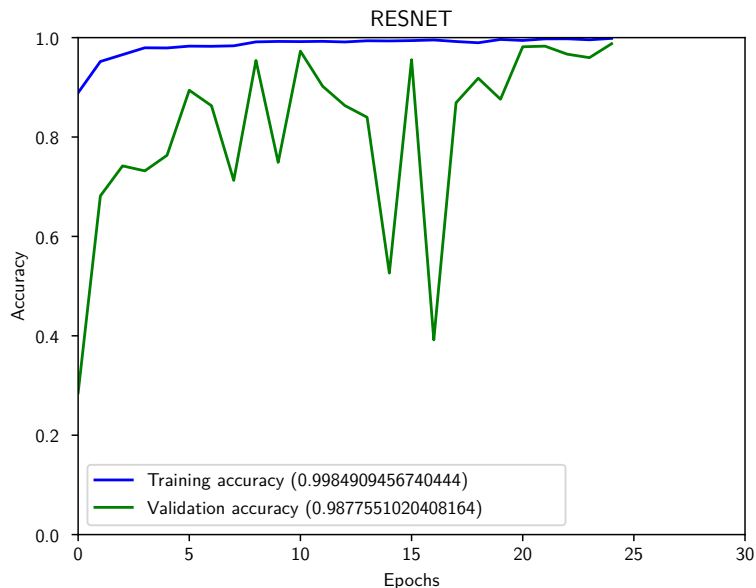


Figure 6.3: Accuracy

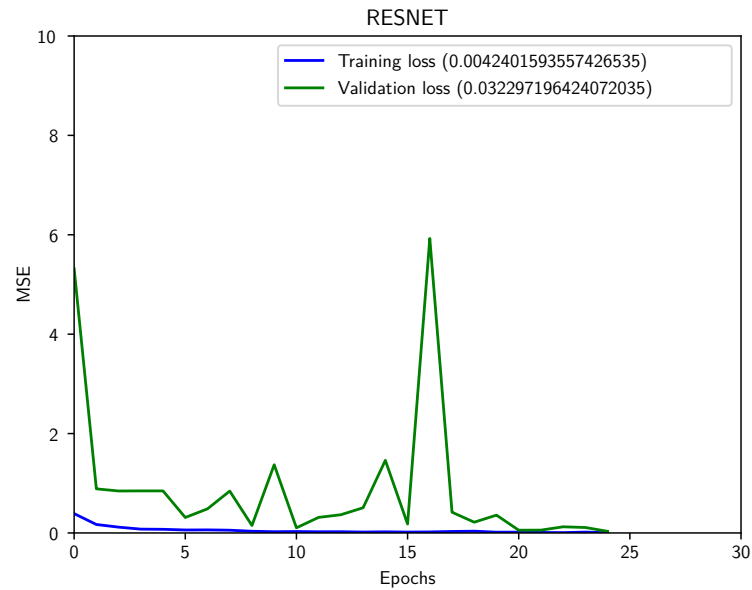


Figure 6.4: Loss

This is the best accuracy for ResNet-50. It reached 98.7% validation accuracy. But it is bumpy.

6.1 Final Training

Upon the previous experiments we chose the best training parameters and we will use them to train on whole dataset. In real world situation we need to do 3 steps of recognition:

1. a recognition of car make
2. a recognition of car model
3. an estimation of year of production

6.1.1 Car Make Recognition

We prepared own dataset *UWB-VMMR-5000* in Figure 5.8 for training CNN to recognized a car make. This dataset is well balanced and contains 41 popular car makes. As both networks *VGG-16* and *ResNet-50* performed very well during warm-up tests, we used both for final training.

6.1.2 ResNet-50 with the Final Dataset

```
1 model = ResNet50(include_top=True, weights="imagenet")
2 model.layers.pop()
3 x = model.layers[-1].output # add last layer
4 x = Dense(256, activation="relu",
5         kernel_regularizer=regularizers.l2(0.01))(x)
6 x = Dense(len(classes), activation="softmax")(x)
7 finetuned_model = Model(model.input, x)
8 finetuned_model.summary()
9 opt = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
10 finetuned_model.compile(optimizer=opt, loss='categorical_crossentropy',
11                        metrics=['accuracy'])
```

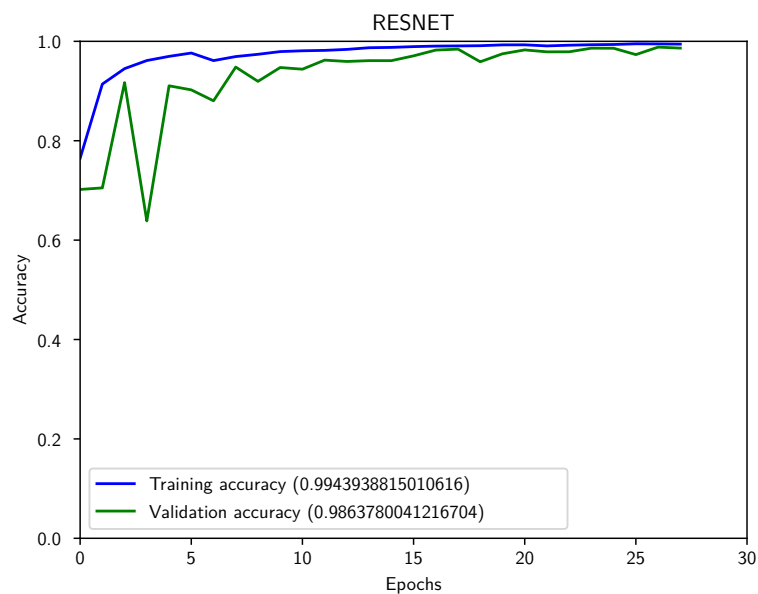


Figure 6.5: Accuracy

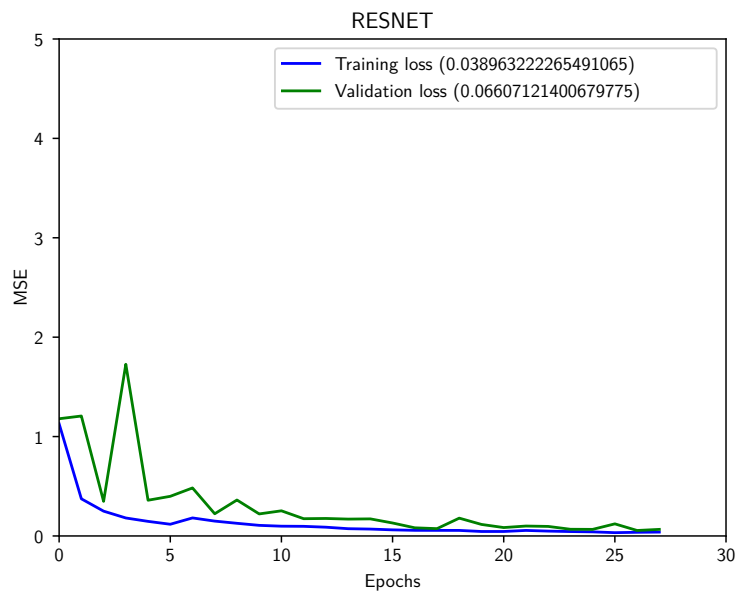


Figure 6.6: Loss

The validation accuracy on the final dataset reached 98.6%. We used *ResNet-50* with a simple classifier 256 x 41 and SGD optimizer.

6.1.3 VGG-16 with the Final Dataset

```
1 model = VGG16(include_top=True, weights="imagenet")
2 model.layers.pop()
3 x = model.layers[-1].output # add last layer
4 x = Dropout(0.1)(x)
5 x = Dense(256, activation="relu",
6         kernel_regularizer=regularizers.l2(0.01))(x)
7 x = Dense(len(classes), activation="softmax")(x)
8 finetuned_model = Model(model.input, x)
9 finetuned_model.summary()
10 opt = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
11 finetuned_model.compile(optimizer=opt, loss='categorical_crossentropy',
12                          metrics=['accuracy'])
```

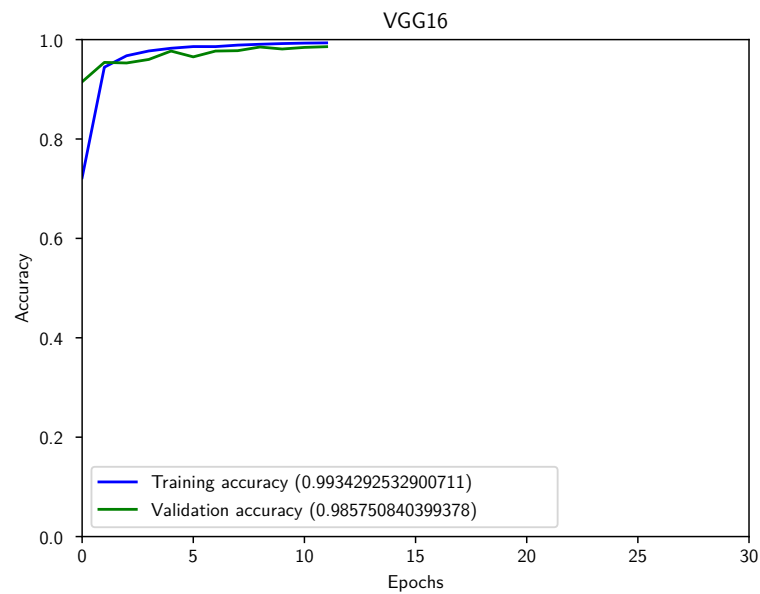


Figure 6.7: Accuracy

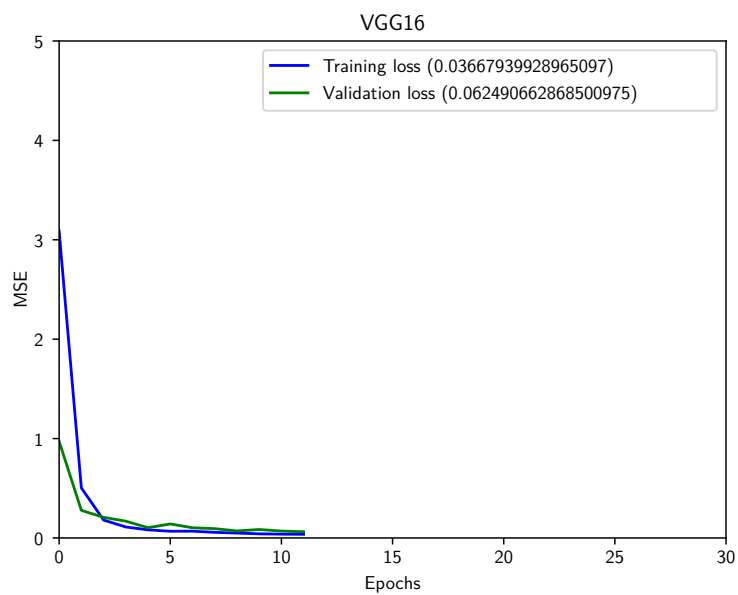


Figure 6.8: Loss

The validation accuracy on the final dataset reached 98.5% in the eleventh epoch. Then it started slowly decrease, so we stopped the learning process. We used *VGG-16* with extended classifier 4096 x 4096 x 256 x 41 and SGD optimizer.

6.1.4 Car Model Recognition

We prepared own dataset SKODA-5000 in Figure 5.9 for training CNN to recognized a car model. This dataset is well balanced and contains 10 popular car models of manufacturer SKODA. We used *ResNet-50* for final training with the same architecture as previous.

```
1 model = ResNet50(include_top=True, weights="imagenet")
2 model.layers.pop()
3 x = model.layers[-1].output # add last layer
4 x = Dense(256, activation="relu",
5         kernel_regularizer=regularizers.l2(0.01))(x)
6 x = Dense(len(classes), activation="softmax")(x)
7 finetuned_model = Model(model.input, x)
8 finetuned_model.summary()
9 opt = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
10 finetuned_model.compile(optimizer=opt, loss='categorical_crossentropy',
11                        metrics=['accuracy'])
```

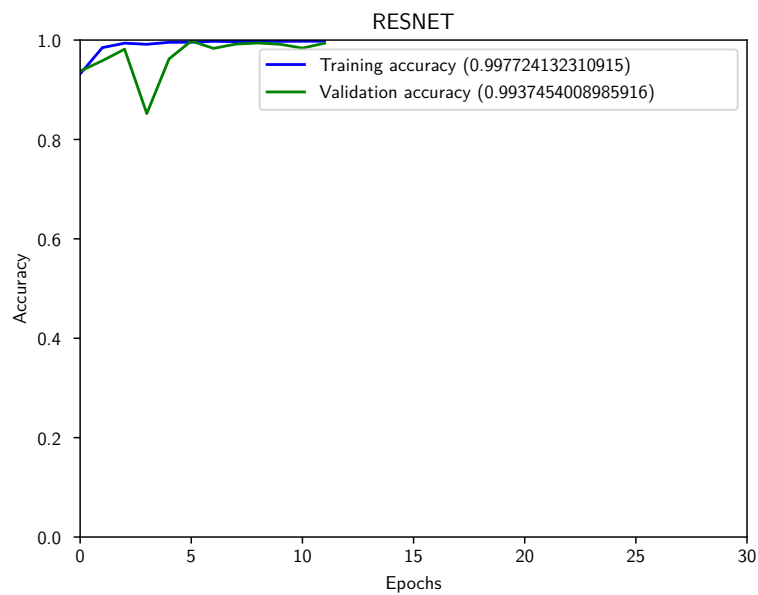


Figure 6.9: Accuracy

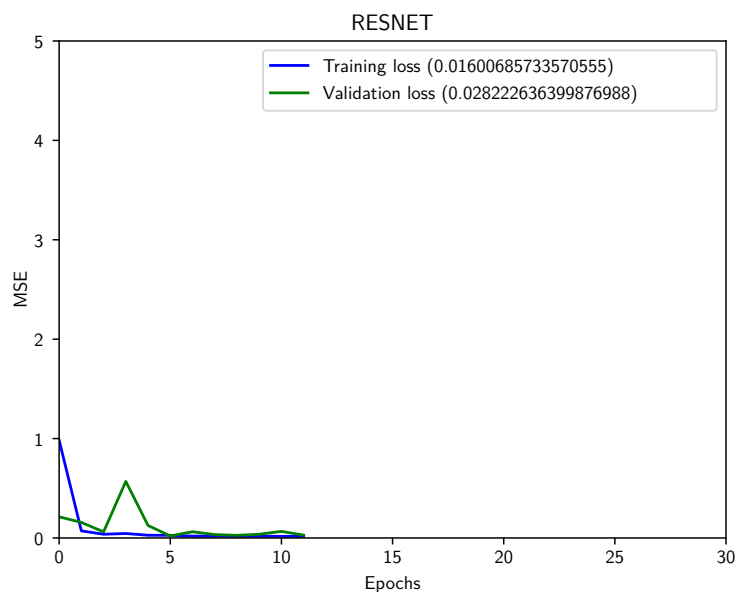


Figure 6.10: Loss

The validation accuracy on the final dataset reached 99.3%. We used *ResNet-50* with simple classifier 256 x 41 and SGD optimizer.

6.1.5 Estimation of a Year of Production

For estimation of a year of production of a car we used own dataset OCTAVIA-2000. There are 22 classes with a numeric value. We used regression for estimation of a year of production with just one neuron on the top. The loss function is mean square error.

```

1 model = ResNet50(include_top=True, weights="imagenet")
2 model.layers.pop()
3 x = model.layers[-1].output # add last layer
4 x = Dropout(0.1)(x)
5 x = Dense(256, activation="relu", kernel_regularizer=l2(0.01))(x)
6 x = Dense(1, activation="relu")(x)
7 finetuned_model = Model(model.input, x)
8 finetuned_model.summary()
9 opt = Adadelta(lr=0.001)
10 finetuned_model.compile(optimizer=opt, loss='mean_squared_error', metrics=['mse'])

```

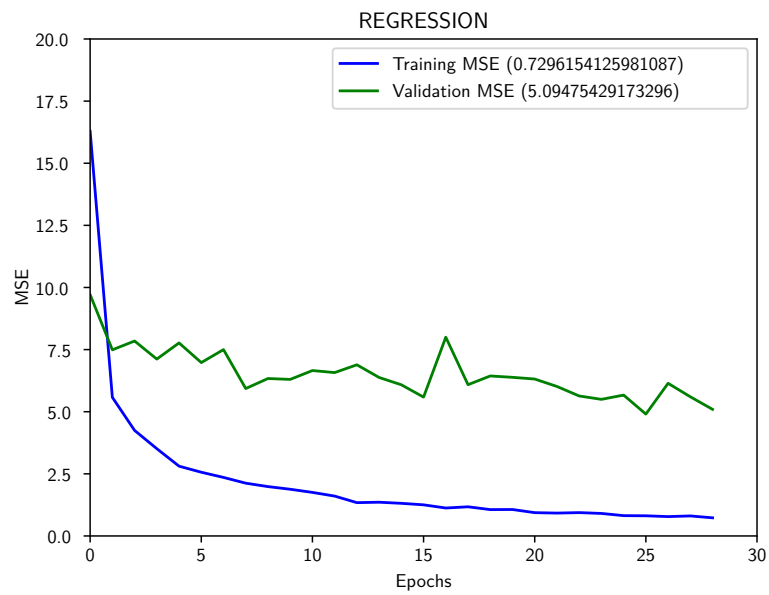



Figure 6.11: MSE

We can see that validation error is rather high, but we can explain it. Sometimes there is not any visual difference between the same car models with a different year of production. Car models can be just the same across a few years.

6.2 Testing with Online Images

We got amazing results during validation. Then we performed tests on photos from another website autoesa.cz. This is another used car dealer.

Input	Label	ResNet-50	VGG-16
	Subaru	Subaru (0.87)	Subaru (0.94)
	Skoda Citigo	Skoda (1.00) Citigo (0.97)	Skoda (1.00) Citigo (1.00)
	Skoda Fabia	Skoda (1.00) Fabia (1.00)	Skoda (1.00) Fabia (1.00)
	Skoda Fabia	Skoda (1.00) Fabia (1.00)	Skoda (1.00) Fabia (1.00)
	Skoda Fabia	Skoda (1.00) Fabia (1.00)	Skoda (1.00) Fabia (1.00)
	Skoda Felicia	Skoda (1.00) Felicia (0.99)	Skoda (0.99) Felicia (1.00)

Figure 6.12: Testing with web images

Input	Label	ResNet-50	VGG-16
	Ford	Ford (0.99)	Ford (1.00)
	Skoda Octavia 2012	Skoda (1.00) Octavia (1.00) 2011.8)	Skoda (1.00) Octavia (1.00)
	Skoda Octavia 2013	Skoda (1.00) Octavia (1.00) 2007.6)	Skoda (1.00) Octavia (1.00)
	Opel Astra	Opel (1.00)	Opel (1.00)
	Skoda Roomster	Skoda (1.00) Roomster (1.00)	Skoda (1.00) Roomster (1.00)
	Skoda Superb	Skoda (1.00) Superb (1.00)	Skoda (1.00) Superb (1.00)
	Toyota Yaris	Toyota (1.00)	Toyota (1.00)
	Volkswagen Passat	Volkswagen (1.00)	Volkswagen (1.00)
	Skoda Yeti	Skoda (1.00) Yeti (1.00)	Skoda (1.00) Yeti (1.00)

Figure 6.13: Testing with web images

Results of testing with web images are amazing. We tested all car makes, when it was SKODA, we tested a car model and when it was OCTAVIA, we tested its year of production. Almost all estimates were correct. Our CNN had not seen these pictures before as they come from a different website. But the composition of photographs is very similar to our training dataset: same point of view, very clean neutral background etc.

6.3 Testing with Real Photos

We got great results during validation and testing with online images. We wanted to verify these results in a real life scenario. Here are some photos from the streets of Pilsen taken by a mobile phone camera.



Input	Label	ResNet-50	VGG-16
	Dacia	Dacia (1.00)	Dacia (1.00)
	Ford	Ford (1.00)	Ford (1.00)
	Ford	Mercedes(0.20)	Mercedes(0.61)
	Ford	Ford (0.62)	Ford (0.99)
	Ford Ka	Peugeot (0.39)	Skoda (0.24) Superb (0.29)

Figure 6.14: Testing with street images

Input	Label	ResNet-50	VGG-16
	Nissan	Nissan (0.89)	Dacia (0.78)
	Opel	Peugeot (0.26)	Skoda (0.63) Fabia (0.58)
	Renault	Renault (0.99)	Renault (1.00)
	Skoda Fabia	Skoda (0.86) Superb (0.59)	Skoda (0.98) Octavia (0.40) 2015.1
	Skoda Fabia	Skoda (0.63) Fabia (1.00)	Skoda (0.99) Fabia (1.00)
	Skoda Fabia	Skoda (1.00) Fabia (0.97)	Skoda (1.00) Fabia (0.95)
	Skoda Fabia	Lada (0.23)	Volkswagen (0.32)
	Skoda Felicia	Skoda (0.64) Felicia (0.97)	Skoda (0.45) Felicia (0.96)
	Skoda Octavia	Skoda (0.80) Fabia (0.80)	Skoda (0.35) Fabia (0.82)

Figure 6.15: Testing with street images




Input	Label	ResNet-50	VGG-16
	Skoda Octavia	Volvo (0.50)	Skoda (0.84) Octavia (0.97) 2004.8
	Subaru Forester	Mitsubishi (0.43)	Opel (0.25)
	Ford	Honda (0.87)	Ford (0.35)

Figure 6.16: Testing with street images

The results of testing with real street photos are far from perfection. Accuracy of VGG16 is 0.64 and accuracy of Resnet-50 is 0.58. We can say VGG-16 performed slightly better than ResNet-50, but that is not the point. There are some serious issues when CNN was not able to recognize a make of car even from the front of the car. It means that CNN is not able to recognize a logo as a feature of a particular class.

6.4 Analysis of Results

Testing of street images was really interesting. Why some images were not recognized? To understand what's happened we used visualisation library *keras-vis*. It uses *Gradient-weighted Class Activation Mapping (Grad-CAM)* [14] and it can visualize regions of input that were 'important' for the prediction.

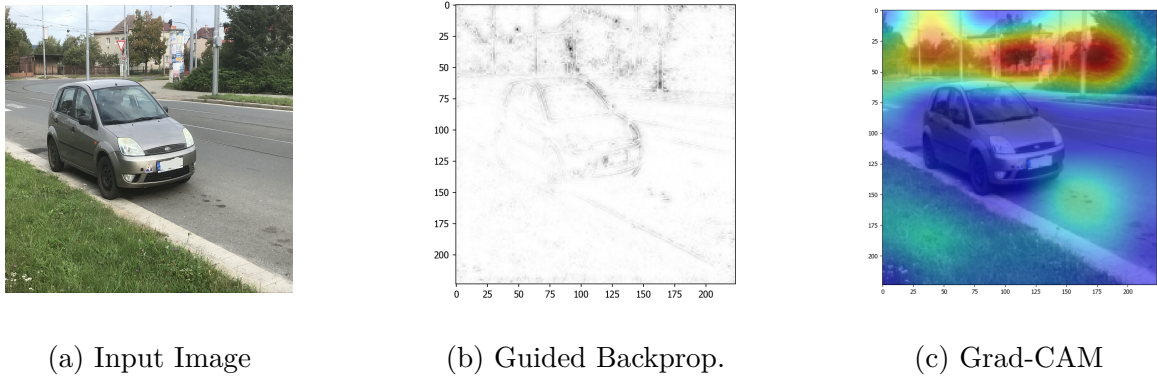
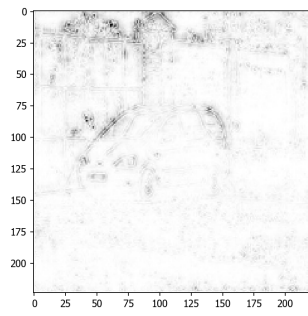


Figure 6.17: Failed estimation Skoda(0.24) Superb(0.29)

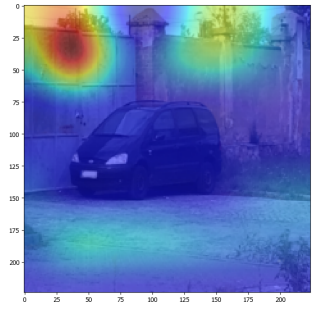
We can see *Ford Ka* in Figure 6.17. It was not recognized by CNN. In Figure 6.17 (b) we can see what regions of the input image was considered as 'important' by the network. We can see the most of activation came from the background above the car. In Figure 6.17 (c) we can see color visualization of 'attention' of the network. The network didn't pay attention to a shape of the car, but it was catching a noise from the background. That's why the network failed to recognized the car. Here are another two examples where the network failed.



(a) Input Image



(b) Guided Backprop.

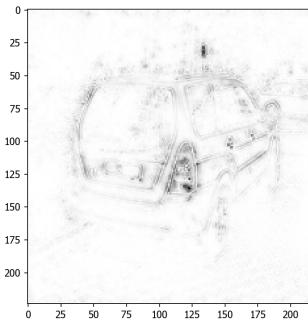


(c) Grad-CAM

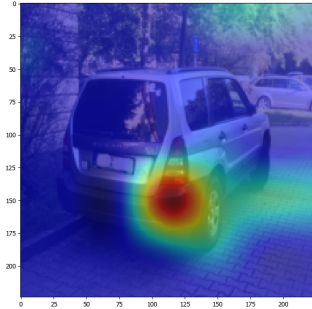
Figure 6.18: Failed estimation Mercedes (0.61)



(a) Input Image



(b) Guided Backprop.



(c) Grad-CAM

Figure 6.19: Failed estimation Opel (0.25)

Now let's have a look to successful recognition. Here are two examples where the network succeeded. We can compare them with negative ones.

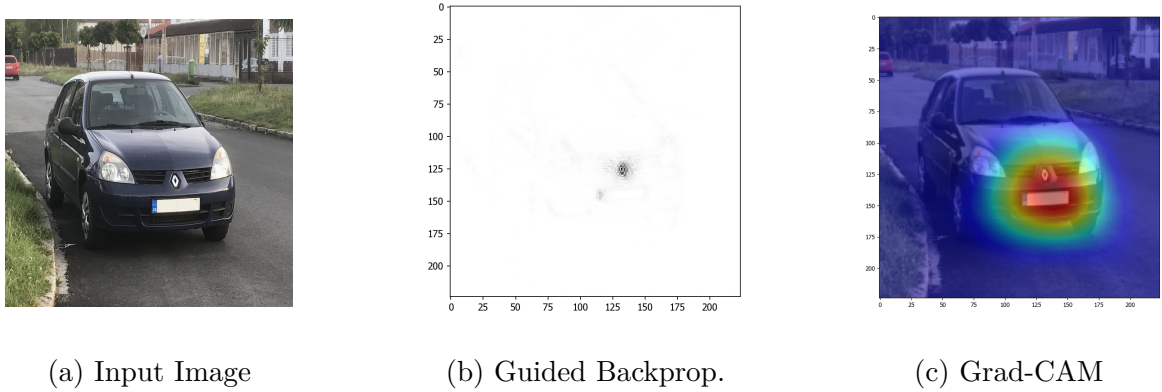


Figure 6.20: Correct estimation Renault (1.00)

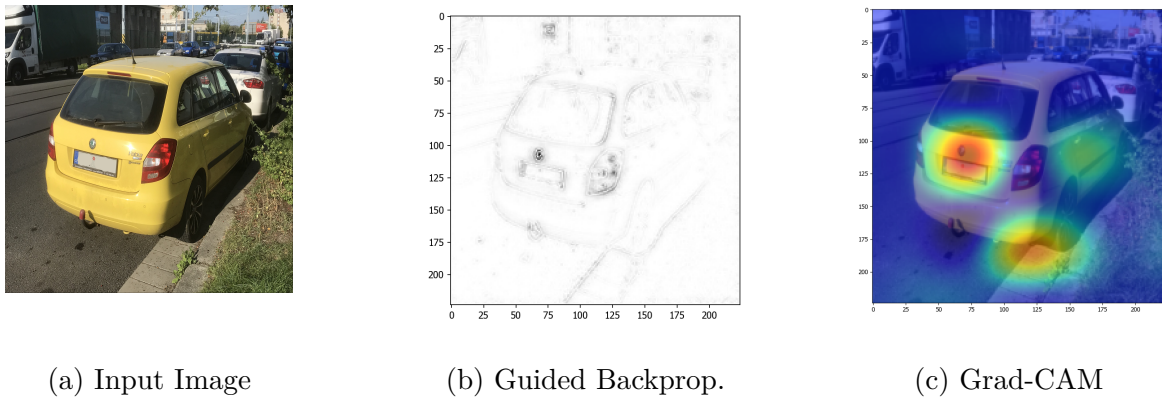


Figure 6.21: Correct estimation Skoda(0.99), Fabia(1.00)

We can see clearly that the network did not recognize a logo on the front of a car on negative tests, on the other hand when the network did recognize a logo of the manufacturer on positive tests then it determined its class confidently. The logo is an essential feature of a car make class. We used standard input size of image. It is (244 x 244) pixels. The area with a logo is very small. It is too small that the network doesn't recognize it in some cases.

Let's have a look at a visualization of an activation function at the last fully connected layer for several classes. We can see something like a logo on visualizations below 6.22.

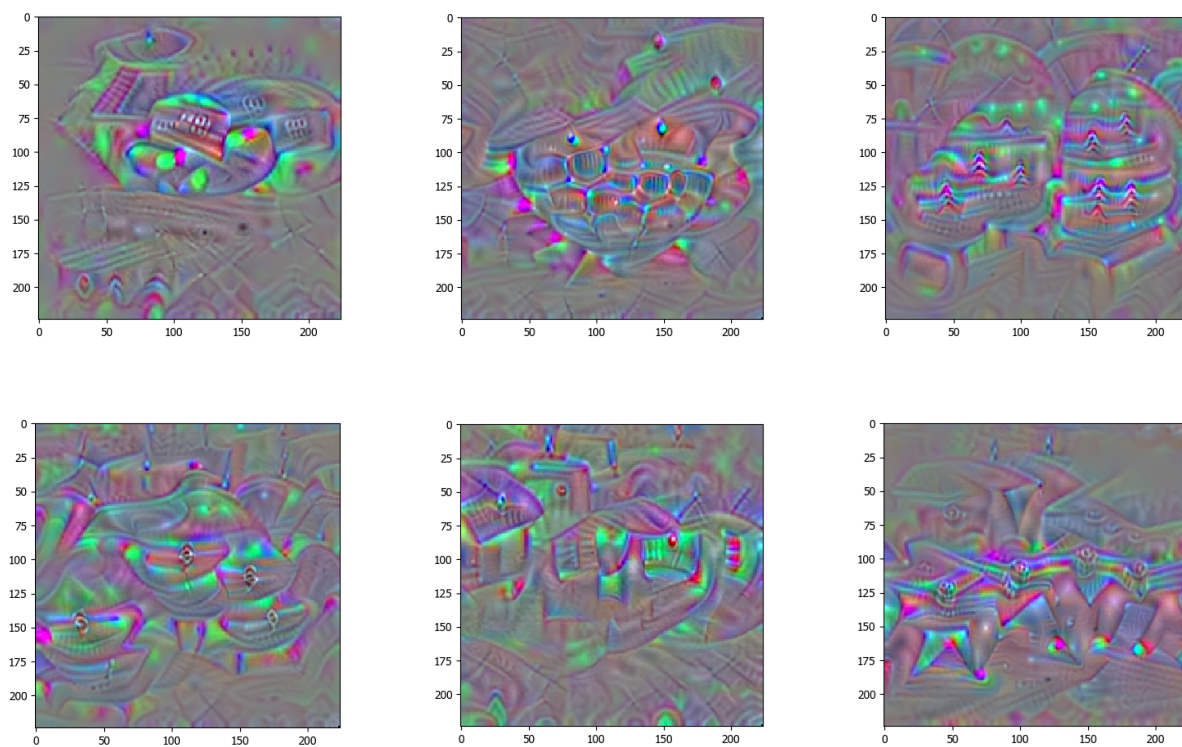


Figure 6.22: Audi, BMW, Citroen, Opel, Skoda, Toyota
(from top left to right)

6.5 Conclusion

We compiled own dataset for VMMR based on online public resources. We set up two common CNN architectures VGG-16 and ResNet-50 using Python and *Keras/TF* [3] backend. We conducted experiments to find out what topology of CNN performs best. We realized both networks work great and we can not say one is better over another - we confirmed that more important is a dataset. We trained CNN models for a car make, a car model and a year of production on our dataset. Despite we reached great results during validation (more than 98% accuracy), results of real life scenario tests performed not so well. We realized that our CNN models are sensitive to image background and sometimes they do not recognize a logo of the manufacturer. We also verified that if the network is able to recognize a logo, then it determines a car make confidently. That confirms hypothesis that a logo is the most important visual feature common for every car make. The reason why a logo is not always recognized is the small size of input image.

Upon this research we suggest the following improvements:

1. **Improve dataset**

Existing dataset *UWB-VMMR-5000* can be extended by images with various backgrounds. These can be personal photos of car from internet such as photos of cars from a street or individual selling ads. Adding these kind of images with different backgrounds can make final model more robust against the noise from the background in real life scenario.

2. **Enlarge input image size**

As it was mention before the size of input images is fairly small and the area of manufacturer logo is too small to catch all details of it. But the logo is a dominant feature of every car make class. That is way increasing of input size can improve performance of our model significantly. Of course it increases computational complexity of final model too. We need to find a compromise between computational complexity and size of input image.

3. **Optionally use a front car mask for input**

We can build a pipeline for the recognition process consisting of several steps. In the first step we detect car lights. Than we crop region of interest (ROI) with lights and process only this area. It contains most of car make features including an enlarged logo, lights and shape of frontal/rear mask. Disadvantage of this method is that if there are no lights in the input image, we can not use it. This approach is more advanced.

Appendices

A Content of DVD

There are saved all artefacts created during experiments.

Folders:

- **datasets** - contains collections of images
- **experiments** - various python scripts for training
- **models** - saved final models in .h5 format
- **results** - result files with progress of accuracy
- **scripts** - utility python scripts
- **tasks** - bash and python scripts used in computing grid environment
- **doc** - pdf version of this document
- **predict** - source of the final VMMR classifier

B User Manual

B.1 Installation

- Insert DVD disk into PC
- Copy folder **/predict/** to your hard drive
- CD to the destination folder **predict/**
- install python3 (*apt install python3-pip*)
- install dependencies (*pip3 install -r requirements.txt*)

Installation is complete.

B.2 Run Program

```
$ python3 predict_car.py
```

The program will start classification of images from the folder *predict/street/*. Results are displayed on screen in format: name of image, name of class(car make) and appropriate accuracy.

Bibliography

- [1] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *CIFAR Dataset*. URL: <https://www.cs.toronto.edu/~kriz/cifar.html>.
- [2] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).
- [3] François Chollet et al. *Keras*. <https://keras.io>. 2015.
- [4] Wikimedia Commons. *File:Artificial neural network.svg* — *Wikimedia Commons, the free media repository*. [Online; accessed 1-October-2019]. 2018. URL: https://commons.wikimedia.org/w/index.php?title=File:Artificial_neural_network.svg&oldid=279227349.
- [5] Wikimedia Commons. *File:Artificial neural network.svg* — *Wikimedia Commons, the free media repository*. [Online; accessed 3-May-2020]. 2020. URL: https://commons.wikimedia.org/w/index.php?title=File:Artificial_neural_network.svg&oldid=392271288.
- [6] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: <https://doi.org/10.1007/BF00994018>.
- [7] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.
- [8] Abhimanyu Dubey et al. “Maximum-Entropy Fine-Grained Classification”. In: *CoRR* abs/1809.05934 (2018). arXiv: 1809.05934. URL: <http://arxiv.org/abs/1809.05934>.
- [9] Utkarsh Gupta. *Detailed Guide to Understand and Implement ResNets*. <https://cv-tricks.com/keras/understand-implement-resnets/>. 2019.
- [10] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: <http://arxiv.org/abs/1512.03385>.
- [11] Jonathan Krause et al. “3D Object Representations for Fine-Grained Categorization”. In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.
- [12] Michael G. Madden and Daniel T. Munroe. “Multi-Class and Single-Class Classification Approaches to Vehicle Model Recognition from Images”. In: 2005.
- [13] V Petrovi and T Cootes. “Analysis of Features for Rigid Structure Vehicle Type Recognition”. In: 2 (Jan. 2004). DOI: 10.5244/C.18.61.

- [14] Ramprasaath R. Selvaraju et al. “Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization”. In: *CoRR* abs/1610.02391 (2016). arXiv: 1610.02391. URL: <http://arxiv.org/abs/1610.02391>.
- [15] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014).
- [16] Steven W. Smith. *The Scientist and Engineer’s Guide to Digital Signal Processing*. San Diego, CA, USA: California Technical Publishing, 1997. ISBN: 0-9660176-3-3.
- [17] Faezeh Tafazzoli, Hichem Frigui, and Keishin Nishiyama. “A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition”. In: July 2017, pp. 874–881. DOI: 10.1109/CVPRW.2017.121.
- [18] Wikipedia contributors. *Backpropagation* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 3-May-2020]. 2020. URL: <https://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=954469400>.
- [19] Wikipedia contributors. *ImageNet* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 5-April-2020]. 2020. URL: <https://en.wikipedia.org/w/index.php?title=ImageNet&oldid=946020209>.
- [20] Wikipedia contributors. *Scrapy* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 24-October-2019]. 2019. URL: <https://en.wikipedia.org/w/index.php?title=Scrapy&oldid=922577632>.
- [21] Linjie Yang et al. “A Large-Scale Car Dataset for Fine-Grained Categorization and Verification”. In: *CoRR* abs/1506.08959 (2015). arXiv: 1506.08959. URL: <http://arxiv.org/abs/1506.08959>.