University Institute of Lisbon

Department of Information Science and Technology

Deep Learning for Large-Scale Fine-Grained Recognition of Cars

João Rodrigo Romão Marinho Pinto da Cruz

A Dissertation presented in partial fulfilment of the Requirements for the Degree of

Master in Computer Science

Supervisor

Prof. Dr. Luís Miguel Martins Nunes, Assistant professor

ISCTE-IUL

Co-Supervisor

Prof. Dr. Tomás Gomes da Silva Serpa Brandão, Assistant professor

ISCTE-IUL

2018

"It is not enough that you should understand about applied science in order that your work may increase man's blessings. Concern for the man himself and his fate must always form the chief interest of all technical endeavours."

Albert Einstein

Resumo

Deep Learning (DL) é um termo cada vez mais mencionado nos dias de hoje, com vastas aplicações em classificação de imagens e detecção de objectos. Por detrás de muitas destas aplicações está a utilização de *Convolutional Neural Networks* (CNN) cujo funcionamento é, para um dado *input* (imagem) e *output* (nome do objecto representado/classe), produzir representações que definem e permitem distinguir vários tipos de objectos.

As redes neuronais são computacionalmente exigentes e podem levar horas a ser treinadas. *Convolutional Neural Networks* são ainda mais exigentes visto o seu input ser, usualmente, imagens - um tipo de dados rico que contém muita informação. Com a rápida evolução do poder computacional aliada à evolução no campo de *Computer Vision* com recurso a CNNs é possível, somente nos últimos anos, treinar CNNs para classificação de imagens com alto nível de precisão.

Em sites de classificados de carros as imagens são um dos tipos de conteúdo mais importante. Todavia até aos dias de hoje, pouco conhecimento/metadados são gerados a partir das mesmas. O utilizador tem sempre que, para inserir um anúncio na plataforma, preencher um vasto número de campos, entre eles a categoria do veículo, a cor do carro e a respectiva marca, modelo e versão, e inserir uma imagem do carro para venda.

Nesta dissertação são utilizadas CNNs para o reconhecimento da marca, modelo e versão de carros em que se utiliza *transfer learning* e *fine-tuning* para transferir o conhecimento "aprendido" numa tarefa e adaptá-lo para outra. O trabalho é estendido de forma a demonstrar, também, a eficácia destas redes neuronais para as tarefas de reconhecimento da categoria do veículo e reconhecimento de cor de carros. Pretendemos validar como as CNNs se comportam nestes diferentes tipos de tarefas.

Abordagens como remoção do fundo da imagem e *data augmentation* são utilizadas para reduzir *overfitting.*

É obtido um dos maiores *datasets* para a tarefa de reconhecimento de marca, modelo e versão de carros, composto por 1,2 milhões de imagens pertencentes a 790 classes.

Os resultados apresentados são dos melhores para este tipo de tarefa (precisão de 92.7% com um *ensemble*) considerando tanto o número de classes a classificar como o número de imagens utilizadas.

Os resultados obtidos evidenciam a eficácia das arquitecturas de CNNs modernas para a classificação granular onde a variação intra-classe é reduzida e a variação da perspectiva é elevada, quando é utilizado um *dataset* de grandes dimensões.

Palavras-Chave: Reconhecimento de marca, modelo e versão, transfer learning, deep learning, convolutional neural networks, classificação de imagens, classificação granular

Abstract

Deep learning (DL) is widely used nowadays, with several applications in image classification and object detection. Among many of these applications is the use of Convolutional Neural Networks (CNNs) whose operation is: for a given input (image) and output (label/class), generate representations that define and allow to distinguish different kinds of objects.

Neural Networks are computationally demanding, taking hours to train. Convolutional Neural Networks are even more demanding since their input data are usually images – a rich data type that holds a lot of information. The fast evolution in Computer Vision, using deep learning techniques, and computing power recently allowed to train CNNs which can classify images with high precision.

In car classifieds websites images are one of the most important types of content. However, until today, little knowledge/metadata is produced from such images. In order to insert an advert in the platform, the user must upload an image of the car for sale and fill a certain number of fields, among them the vehicle category, the color of the car and its respective make, model and version.

In this dissertation, CNNs are used for the recognition of the make, model and version of cars where transfer learning and fine-tuning are two approaches used for transferring the knowledge learned in one task and adapting it to another. We extend the work to also validate the efficacy of these neural networks on the tasks of vehicle category and cars' color recognition. We pretend to validate how CNNs behave in these different tasks.

Approaches like background removal and data augmentation are explored for reducing overfitting.

We collected one of the largest datasets to date for the task of *make, model and version* recognition of cars, composed of 1.2 million images belonging to 790 labels.

The results obtained in the scope of this dissertation set a new state-of-the-art performance for this type of task (accuracy of 92.7% on an ensemble method) considering the number of classes to classify and the number of images used.

It is demonstrated the efficacy of the recent advances in CNN architectures in fine-grained classification where intra-class variation is small and viewpoint variation is high, when a large-scale dataset is used.

Keywords: Make, model and version recognition, transfer learning, deep learning, convolutional neural networks, image classification, fine-grained classification

Acknowledgments

First and foremost, my most important acknowledgment goes to my parents, António and Alcinda, and to my brothers, Nuno and Sofia, for their unconditional love and support. They were the ones who established the foundation that define who I am today.

Secondly, an acknowledgement to my supervisors, Prof. Dr. Luís Nunes and Prof. Dr. Tomás Brandão, for their mentorship and always constructive feedback.

I would like to thank my company for the opportunity to carry out this dissertation.

Finally, a word of appreciation to ISCTE, as institution of excellence, and to its academia.

Contents

List of Figures and Tables

Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DA	Data Augmentation
DL	Deep Learning
GPU	Graphics Processing Unit
HOG	Histogram of Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
RGB	Red Green Blue
SIFT	Scale-Invariant Feature Transform
SGD	Stochastic Gradient Descent
VMMR	Vehicle Make and Model Recognition

Chapter 1

Introduction

Computer vision has evolved greatly in recent years with performance improving year after year on image recognition tasks. The models trained with image data have gotten better as more robust and complex models are developed for image recognition tasks. With the use of deep learning and, more specifically, convolutional neural networks, results have improved drastically over the last years. However, computer vision has come a long way and as a research field it dates back to the early 60s where the goal was: how to analyse an input image in such a way that the computer can make 3D representations of it. The process was around pixel understanding by detecting edges, vertical lines and blobs. Until the early 2000s much of the work relied on computer representations of images and not much on image recognition and classification. Computation power and data quality and quantity were still scarce.

In the early 2000s, computer vision work started encompassing machine learning algorithms for image's feature detection and the first real-time face detection method appeared (Viola & Jones, 2001). Part of the problem has always been how to identify, efficiently and accurately, the set of features that correspond to a certain object, considering that images are such a rich and complex data type (for instance, an image with dimensions 800 x 600 has almost half a million pixel values).

Only in the late 2000s public datasets started proliferating. One of the most popular and largest one is ImageNet, which consists of 14 million images classified over 22 thousand categories (Deng et al., 2009). ImageNet's image labelling process was a manual one by leveraging

Amazon Mechanical Turk. There is a heavy manual task involved to individually label the images in most image datasets, which makes it hard to get a dataset with both high quality and quantity.

The ImageNet challenge (ILSVRC) is a competition with the goal to correctly classify 1000 categories from 1.4 million images (Russakovsky et al., 2015). There are other datasets that gained popularity as tools for benchmark. Examples of such datasets are MNIST (LeCun, Bottou, Bengio, & Haffner, 1998) for recognizing handwritten digits, CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) for recognizing objects from small images, COCO (T. Y. Lin et al., 2014) for object detection and Stanford Cars dataset (Krause, Stark, Deng, & Fei-Fei, 2013) for make, model and year recognition.

Convolutional neural networks were firstly used, with state-of-the-art results, in ILSVRC-2012, with a CNN called AlexNet (Krizhevsky, Sutskever, & Hinton, 2012). The top-5 error rate[1] in the ILSVRC was 25.8% in 2011 and AlexNet dropped this value to 16.4% the following year (2012). Deep learning (and CNN) models started appearing and the ILSVRC error rate started decreasing year by year with CNN architectures such as VGGnet (Simonyan & Zisserman, 2014) with a top-5 error rate of 8%, GoogleNet (Szegedy et al., 2015) with 6.7% and ResNet (He, Zhang, Ren, & Sun, 2016) with 3.57%.

Image classification is a task that can be applied to a plenitude of domains. The task of vehicle make and model recognition (VMMR), is one that falls into the fine-grained image classification category. It is easier to distinguish objects that are different, such as to distinguish a vehicle from a person, than it is to distinguish objects that share the same structural characteristics, such as breeds of dogs, species of birds and cars' makes and models. In the latter, the system must "learn" to differentiate between object classes from low level cues and parts.

Stanford Cars dataset (Krause et al., 2013) was one of the first datasets, with enough quality and quantity, that appeared for VMMR. It consists of 16 thousand images of over 196 categories of makes and models. It is used as a benchmark dataset for different approaches to image classification. The authors, by leveraging the work of Lazebnik, Schmid, & Ponce (2006) and Deng, Krause, & Fei-Fei (2013) to 3D representations of objects for fine-grained classification, set the accuracy baseline at 75.5% with an ensemble of both methods. They used a pre-deep learning approach. The 196 categories of the Stanford Cars dataset are fine-grained considering the images are all cars but the makes and models are visually different from each other.

[1] Top-5 error rate refers to the error rate of misclassifying a class considering the five predictions with highest probability

In classifieds websites, images are a mandatory and valuable component of a given advert. To insert an advert in such websites, the user must upload at least one image that represents the item being sold. Specifically, in car classifieds websites, the user has the laborious (but useful task of labelling) of having to upload several pictures and fill the corresponding fields that represent the attributes of the car including its make, model and version. No information exists about the images other than the information filled by the users. The images are stored and shown when the advert gets published. A lot of the information held by the image is not used for business value.

For an optimized and simplified user interaction, the process should be as follows: 1) the user wants to add a car for sale; 2) he uploads or takes an image of his car; 3) the category, color, make, model and version are automatically inferred from the uploaded car's image. The user could also get feedback about the quality of the image, where a "quality" image is one where the car is centred in the image with little to no background and with the bounding box enveloping the car as close as possible from the image resolutions.

In the present dissertation, we leverage deep learning methods for large-scale and fine-grained VMMR. We take advantage of having labelled data from one of Europe's biggest car classifieds website, Otomoto, to build and train one of the largest make, model and version dataset and achieve state-of-the-art results (top-1 accuracy[2] of 92.7%) considering the number of images (1.2 million) and classes (790) trained on. We further extend our work for the tasks of vehicle category and cars' color classification. We use transfer learning as the main methodology for fine-tuning CNNs to transfer the knowledge learnt in one task (a general one) and apply it to a new one. In simplistic terms, a convolutional neural network consists of millions of weights and hundreds of filters that can distinguish and classify a certain number of output classes. To apply transfer learning, we replace the fully connected layers that end the network (also called the "head" of the network). A fully connected layer is a classic neural network layer where all the neurons in one layer are connected to all the neurons in the next layer. This makes it possible to learn the non-linear combinations of the high-level features, learnt in the convolutional layers that correspond to the objective classes. After replacing the layers of the "head" of the network we continue to fine-tune the network so that the weights are updated. This leads to improved results (as the weights are already initialized) when comparing with training a network from scratch by randomly initializing a network's weights.

[2] Top-1 accuracy refers to the accuracy of correctly predicting the class considering only the prediction with highest probability

1.1 Objectives

Thousands of images are uploaded into car classifieds websites every day with no metadata being generated from these images. Images are simply stored and fetched when the advert gets published. Users of classifieds websites must fill several fields to post an advert. In the case of car classifieds this task is arduous given the number of fields to fill and the number of adverts to post. Some car dealers have hundreds of car adverts to post. Posting adverts manually is a time-consuming process that can be automated.

For a good user experience, a solution is to have the user fill as few fields as possible. Some steps are already being taken in more general purpose classifieds websites where a user can, for example, upload an image of a cell phone he has for sale and some characteristics of the phone are inferred from the image, for example, that it is a Samsung Galaxy from 2015.

Even though clear advances have taken place in image classification by successfully determining what is represented in an image from a set of 1000 classes (Russakovsky et al., 2015) only in the last years focus has been put into more fine-grained tasks where models need to be robust to low intra-class variation. An example of a fine-grained classification task is recognizing a car's make and model since cars often look very similar. Even humans face a hard time distinguishing between certain models of cars (for example different BMW and Mercedes-Benz models).

Some public datasets exist: Stanford Cars (Krause et al., 2013), CompCars (Luo, Loy, & Tang, 2015) and VMMRdb (Tafazzoli, Frigui, & Nishiyama, 2017). However, they lack quantity, quality and applicability either by having a low number of images or classes or an unequal distribution of images per class.

One of the first tackled objectives of this dissertation is to demonstrate what affects performance when using deep learning methods as a tool. This is done by performing multiple experiments on the Stanford Cars dataset. Even though it is trivial that larger datasets lead to improved results we demonstrate how much of an impact it has. With the intuition obtained by doing different experiments with the Stanford Cars dataset we then proceed with collecting and cleaning one of the largest cars' make, model and version dataset and achieve state-of-the-art results by training different CNN architectures. Existing public datasets for VMMR consist of a

couple thousands of images and often present unequal distribution per class. Aiming to demonstrate the real-world applicability of VMMR we build a dataset of 1.2 million images distributed over 790 classes. Furthermore, we want to demonstrate if CNNs are as useful in fine-grained classification as they are in more general classification tasks (such as ImageNet). This is done by looking at the test accuracy in the tasks of VMMR and vehicle category and color recognition task. Even though good results are obtained in smaller scale datasets, such as Stanford Cars in a fine-grained classification task, we want to demonstrate if the same state-of-the-art results are obtained when using a larger and finer grained dataset. We evidentiate how much of an impact dataset size has on performance. Model visualization techniques are used with the objective to understand in what parts of the image the network focuses on for the task of VMMR.

This dissertation has as the main objective: build a large-scale dataset, train CNN architectures and achieve a high test accuracy so that it can be applied to solve a real-world problem: the automation of content posting by having as the model's classes the make, model and versions of the different cars in the car classifieds website.

1.2 Dissertation Contribution

The current dissertation presents the following contributions:

- Analyze how CNNs performance is impacted by using different architectures, dataset quality and data augmentation techniques in fine-grained classification tasks. We benchmark Stanford Cars dataset, a cars' make, model and year dataset.
- Share insights in what kinds of representations CNNs are creating of input data with respect to VMMR, what parts of the car they favor for prediction and how they behave on very similar examples. This will be useful to visualize and check if the network is learning strong representations of output labels.
- Validate the efficacy and limitations of CNNs on complementary tasks: vehicle category and cars' color recognition. With respect to cars' color recognition, even though a CNN

uses the three color channels of images, we will visualize the results obtained and where the model is performing less optimally.

- Collect and clean one of the largest and most fine-grained datasets of make, model and versions, considering the number of images and classes being classified, and achieve state-of-the-art results by using the latest developments in deep learning and computational power. We also create subsets of this dataset to further validate and quantify the importance of dataset size in fine-grained classification.

1.3 Dissertation Structure

In Chapter 2, we review the state-of-the-art with respect to image classification by analysing the history of computer vision and how it evolved over the years until it reached the ubiquitous use of CNNs in recent years. We also provide a review of state-of-the-art research regarding VMMR. In Chapter 3, we perform a benchmark on the Stanford Cars dataset with recourse to different approaches and CNN architectures. After this benchmark is performed, in Chapter 4 we present Oto-790, a dataset consisting of 1.2 million images distributed over 790 classes, explain how it was obtained, cleaned and trained. Results obtained in Oto-790 are also presented in this chapter. In Chapter 5, we extend the work to cars' color and vehicle category recognition, which are two useful classification tasks for car classifieds websites. Finally, Chapter 6 draws the main conclusions for the work performed in the scope of this dissertation and presents topics for future research.

Chapter 2

State of the Art

In this chapter, we start by reviewing what problems early computer vision was trying to solve. We continue by reviewing the evolution in Deep Learning as a field, how CNNs came into play and what they consist of. We proceed by reviewing the most important advances in CNN architectures and how they performed in the ILSVRC. The chapter finalizes with a study on the publicly available car datasets and what results were obtained with them.

2.1 Early Computer Vision

The goal of computer vision is to have a computer do the same as the human's visual system by classifying, detecting objects and understanding the scene, which humans excel at. However, there is a semantic gap between what humans and computers "see". Images, in simplistic terms, are a grid of numbers – how can we create reasoning from pixel values? The same object can be positioned differently in different images and have multiple perspectives, but it is still the same object. However, if the same object appears in different positions of an image, its overall pixel representation is different. Example: an image with an object positioned in the top left corner of an image has a different pixel representation than if the same object was positioned

in the middle of the image. Occlusion and viewpoint are, also, obstacles in image classification since only a part or the same object in a different shape may be present in the image.

Furthermore, illumination conditions, background and intra-class variation play an important role in object recognition.

Work in computer vision started by trying to solve simple problems. One of the first works in computer vision, as a field, dates back to the early 60s where researchers were trying to create 3D computer representations of 2D images by detecting edges, lines and blobs (Roberts, 1965).

In the 70s, research revolved around how to deconstruct image objects into a collection of simpler shapes and understand the spatial relations between them (Fischler & Elschlager, 1973; Binford, 1971).

Continuing the same line of thought about creating simpler representations of an input image was the work of Lowe (1999) where different objects were outlined in images and then compared to understand what object it was.

In the late 90s, the Normalized Cuts method (Shi & Malik, 1997) was presented as one novel approach that tried to simplify the task of computer vision. Instead of trying to understand what is in the image, by recognizing low level features such as edges and lines, they performed image segmentation, by decomposing the image into meaningful blobs. Back then, computing power was still scarce, image quality was much lower than today's and datasets were almost non-existing.

It was only in the early 2000s that, with the use of machine learning algorithms, research works started appearing with real-world application. The work of Viola & Jones (2001) used machine learning algorithms to perform face detection. They used Haar-like features (Fig. 1) and AdaBoosting (Freund & Schapire, 1997) for effective and real-time face detection. Haar-features are rectangles of different areas that compute the sum of pixel values along the entire input image. A classifier is built on top of these features to recognize objects.

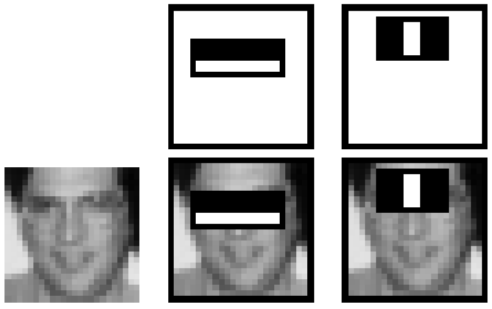


Figure 1 - Haar-like features for face detection (Viola & Jones, 2001)

SIFT features (Lowe, 2004) and HOG descriptors (Dalal & Triggs, 2005) (Fig.2) were also important for developing a way to encode images in such a way that they can then be used efficiently by a machine learning algorithm. The former determines scale, illumination, orientation and occlusion invariant features from an input image. For the latter, a histogram of gradients is determined for each image region. Gradient vectors are computed for each pixel and discretized into 20 degrees (histogram) buckets. The image is divided into blocks and the histogram of gradients of all blocks are concatenated into a single vector. These vectors are then passed into a machine learning classifier for object recognition. A gradient vector is the direction over the x and y-axis of the maximum increase in the loss function.

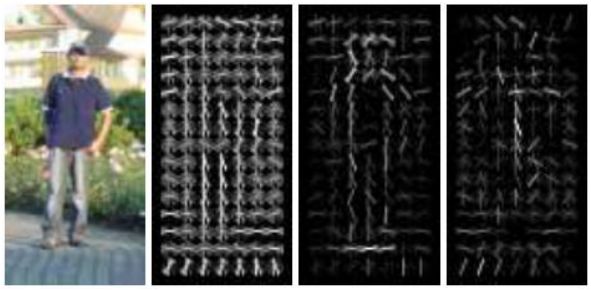


Figure 2 - HOG descriptors representation (Dalal & Triggs, 2005)

In 2009, HOG features are used in the work of Felzenszwalb, Girshick, Mcallester, & Ramanan (2009) which, instead of using them directly as input for classifiers, parts learnt from these HOG features are the input of the model.

Most of this work was the state-of-the-art in computer vision, before the introduction of convolutional neural networks. Also, part of this work is included in how convolutional neural networks operate, such as encoding images in other representations (collection of features) and working with the same image in different scales.

2.2 Deep Learning for Computer Vision

Deep learning is a subfield of machine learning that uses artificial neural networks (ANNs) to develop ever more meaningful abstractions of input data. The abstractions learnt in one layer of the network are passed onto the following layer, and so on. The deep in DL means that instead of using a shallow network (consisting of only a few layers), the networks used contain hundreds of layers so that more meaningful abstractions can be constructed from input data. Low-level abstractions are learnt in early layers of the network and higher-level in latter layers.

The main paradigm shift in deep learning, and machine learning in general, is to go from a hand-engineered and rule based system to one where the rules are learnt from the data (Fig. 3). In the case of deep learning, one big advantage, is that the models do the feature engineering on their own.

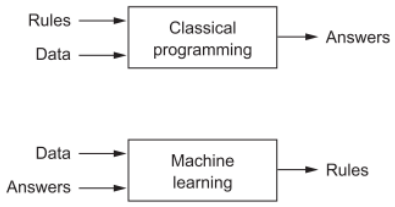


Figure 3 - Paradigm shift (Chollet, 2017)

Work in the deep learning field, even before the term was coined, dates back to the early 40s, with the creation of the first mathematical model of a neural network (McCulloch & Pitts, 1943), followed by the advent of the Perceptron in the late 50s (Rosenblatt, 1958). The backpropagation mechanism for Artificial Neural Networks (ANNs) is introduced as early as the 70s (Linnainmaa, 1976) but only described how it could be used in neural networks in the 80s (Rumelhart, Hinton, & Williams, 1986). Backpropagation is an integral part of ANNs and allows a neural network to adjust its weights in order to minimize its loss function.

In the late 90s LeCun et al. (1998) introduced a 7-layer convolutional neural network capable of recognizing hand written digits. One of its applications was to recognize hand-written checks in banks.

A research breakthrough came only in 2012 when convolutional neural networks, allied with the latest developments in Graphics Processing Units (GPUs), achieved human-level accuracy in image classification. They won the ILSVRC of that year, by beating the previous year winner, with a 41% lower top-5 error rate (that used SIFT features). The CNN architecture, called AlexNet (Fig. 4), achieved top-1 and top-5 test error rate of 36.7% and 15.3% respectively (Krizhevsky et al., 2012). Top-x error rate refers to the fraction of images the model predicts incorrectly considering the top-x most probable guesses.

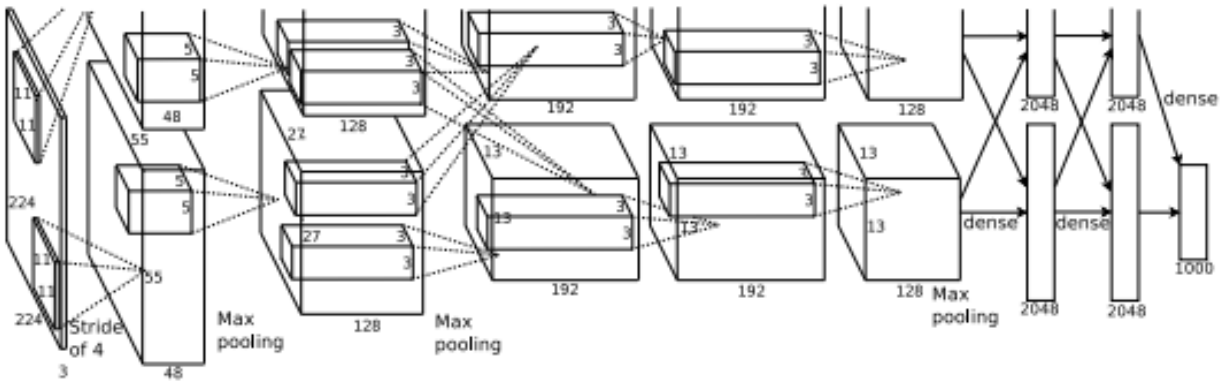


Figure 4 - AlexNet architecture (Krizhevsky et al., 2012)

Since then, the use of CNNs for the task of object classification, detection and segmentation boomed. Convolutional neural networks are the most widely used method in deep learning given their short need for feature engineering and data preparation.

Deep learning has been the stepping-stone for achieving near human-level accuracy in the areas of image classification, speech recognition, text-to-speech translation, autonomous driving and natural language processing (Chollet, 2017).

We will proceed by reviewing the main building blocks of CNNs.

2.2.1 CNN Considerations

Convolutional neural networks are similar to regular neural networks in the sense that they also contain neurons, layers, learnable weights and a loss function on the last layer of the network. However, CNNs include layers that try to mimic the visual cortex, which makes them especially useful for image classification tasks. Since their input, usually consists of images, little pre-processing is needed. Nevertheless, a good dataset quality is still required. A convolutional neural network will itself learn the kernels/filters (features) that distinguish objects from different classes. The main premise of CNNs is that little to no feature engineering is needed since the network will itself do the feature engineering by creating ever more meaningful abstract representations of input data.

Images are represented by their pixel values in the RGB channel with dimensions height × width × 3 or × 1 if we are considering images in the grayscale.

There are four types of layers to consider when building CNNs (Fei-Fei, L., Karpathy, A., 2016):

- Convolutional Layer – consists of sliding learnable filters through the input image dimensionality and compute dot products between the filter and the input image pixels. If we have an image of dimensions 32 × 32 × 3 and 40 filters of dimensions 3 × 3 × 3, with zero padding and stride of one we will have an output volume of dimensions 30 × 30 × 40. For W = 32 as the input volume size, F = 3 as the filter size, S = 1 as the stride and P = 0 as the padding, the output volume will have size equal to the following formula:

$$\frac{(W - F + 2P)}{S} + 1 = 30$$

  Therefore, the output volume will have dimensions: 30 × 30 × 40 where 40 is the number of filters applied. Stride refers to the jump, in pixels, vertical and diagonally, in each step of the pooling or convolution operation. The input volume depth always matches the filter depth and the number of filters corresponds to the output volume depth. These filters will then get "excited" or activated when they see particular edges or vertical lines (patterns) in an image in early layers of the network and some higher level representations like wheels or cats' ears on succeeding layers.

- Pooling Layer – is responsible for performing a downsizing on the volume to reduce the number of parameters to learn and compute, and to reduce overfitting. Considering a pooling layer with hyperparameters: spatial extent of 2 (filter size), stride of 2 and an input volume of dimensions 28 × 28 × 6, the output volume will be 14 × 14 × 6, so a non-overlapping operation occurs. In LeCun et al. (1998), average pooling is used because of its popularity in early days of CNNs but nowadays a pooling layer often performs a max pooling operation, which picks the maximum value in the volume region being considered (Fig. 5). So, whilst reducing the volume size we are also favoring high number values that perform better in the convolution operation and might be representative of some feature.

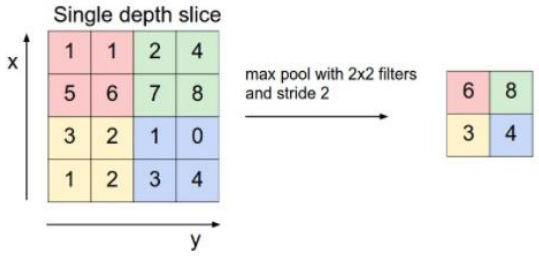


Figure 5 - Pooling Operation (Fei-Fei, L., Karpathy, A., 2016)

- Fully-Connected Layer – just like in regular neural network this layer is connected to all the values of the previous volume and is represented by a vector of size equal to the number of neurons.
- ReLu – responsible for choosing the max between zero and the input values, removing negative values. The size of the volume remains unchanged.

CNNs often use a combination of succeeding convolutional and pooling layers followed by some fully connected layers. In the work of Springenberg, Dosovitskiy, Brox, & Riedmiller (2014) the usability of pooling layers is questioned and replaced by convolutional layers with increased stride. Their architecture, that did not use pooling layers, obtained state-of-the-art results in image classification with a top-1 accuracy of 41,2% on the ILSVRC-2012. This is on par with Krizhevsky et al. (2012) 40,7% accuracy. It took only four days to train and has 6 times less parameters than AlexNet (Krizhevsky et al., 2012). In M. Lin, Chen, & Yan (2013) pooling layers are replaced by 1 × 1 convolutional layers as a means for dimensionality reduction, since convolutional layers are

applied throughout the volume depth allowing dimensionality reduction along that same depth according to the amount of filters used and their stride.

As in CNNs the input are images, they have the following two characteristics (Fei-Fei, L., Karpathy, A., 2016):

- Parameter sharing: a kernel/filter that is useful in detecting a feature in a section of an image can detect the same feature in another section of an image. Therefore, there is no need to relearn that for each of the volume pixels. With the use of parameter sharing we assume that a CNN will be translationally invariant, meaning if a face is recognized in the left side of an image it is supposed to also recognize when positioned in the middle or the right side of the image. This might not be the preferred behavior when we only want to detect objects if they are located in a certain part of an image.
- Local connectivity: each neuron of the output volume is only connected to a region in the input volume (same size as the kernel/filter). If we have as input a 32 × 32 image and we convolve it with six 5 × 5 filters, each neuron of the output volume is only influenced by 25 values of the input image.

These two characteristics allow CNNs to be computationally more efficient.

When applying CNNs to a domain problem such as cars recognition the strategy rarely is to train it from scratch by randomly setting the network's weights. What tends to be done, instead, is to first load the weights of a CNN trained on a general-purpose classification task (commonly ImageNet) and apply transfer learning to update the network weights to the desired/new task. The more distinct the new dataset (domain problem) is from the original one (ImageNet for example) the more layers must be fine-tuned. Fine-tuning refers to the process of updating the weights of a network.

We might not want to fine-tune early layers of a network since they are responsible for detecting low level features such as lines and edges that are applicable to different classification tasks. Fine-tuning these early layers may lead to overfitting. Overfitting occurs when a trained model performs well on seen data and bad on unseen data, which means it is generalizing poorly. This is often observed if a gap starts occurring between train and test accuracy while fine-tuning the network.

When fine-tuning CNNs, a practice used to reduce the risk of overfitting is to use data augmentation. Given that it is often hard to obtain a reliable and large enough labelled dataset, that represents most real-world use cases, data augmentation is often included in the pipeline of training a convolutional neural network to generate more data. This is achieved by applying transformations on dataset images.

Below are examples of techniques commonly used for data augmentation:

- Original

- Rotation – rotate an image by an amount of degrees. Useful for viewpoint variation.

- Horizontal and vertical shift – translates an image.

- Zoom – apply zoom on an image. Useful when the classes we are classifying are in different scales.

- Shear – apply a shear transformation on the image.

- Horizontal flip – if we are not considering the orientation of an object for classification (if it is facing left or right) horizontal flip can be used to generate a more robust dataset and for the model to create more robust representations of the data.

The most beneficial data augmentation techniques are horizontal flip, which allows to easily duplicate the size of a dataset and create robustness by considering objects in both orientations, and width and height shift, which translates part of the image.

The performance of a machine learning model can be evaluated using the following metrics:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Precision refers to the ratio between the number of correctly classified examples of a class from all that were classified in that class. Recall refers to the ratio of the number of correctly classified examples over all examples that belong to that class. F1-score is the balance between both precision and recall. F1-score is a good metric to compare models.

2.3 CNN Breakthroughs

Up until 2012 computational power was not sufficient for the expensive task of training a deep convolutional neural network. LeCun et al. (1998) work on digit recognition with CNNs was a breakthrough in accuracy and applicability since it solved a real-world problem: recognition of hand-written bank checks. But computational power did not make it possible to translate this work

to other, more complex, domain problems. Since the inception of ImageNet's dataset and challenge, the performance of CNNs started increasing, with the use of ever more elaborate CNN architectures. In this section, we will review CNN breakthroughs in recent years in the tasks of image classification and object detection and how they performed on ILSVRC. These CNN architectures are pervasive in different domains and have allowed to achieve near human level accuracy in the task of image classification and object detection.

2.3.1 Image Classification

2.3.1.1 AlexNet

The breakthrough in CNNs occurred in 2012 with AlexNet (Krizhevsky et al., 2012). Until then there were no datasets big enough that models trained on these datasets could detect thousands of different objects. Most of the tasks with good scores were simple ones such as digit recognition on the MNIST dataset. AlexNet was a breakthrough since it was a CNN that was trained on the ImageNet dataset, a dataset composed of 1.2 million images among 1000 categories/classes. AlexNet's architecture consists of five convolutional layers and three fully connected layers. The first three convolutional layers have alternating max pooling layers and two techniques are applied to avoid overfitting:

- Dropout (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) is applied in the fully connected layers. Dropout works by setting the weights of neurons to zero with a probability of 50. It performs weight resetting to prevent overfitting. With dropout, twice the amount of iterations is needed for weights to start converging.
- Data augmentation is the second technique applied. Image translations, on 224 × 224 patches of 256 × 256 input images with a stride of one, and horizontal reflections are applied to increase the dataset size by a factor of:

$$(256 - 224) * (256 - 224) * 2 = 2048$$

AlexNet achieved a top-1 test error rate of 36.7% and a top-5 test error rate of 15.3% ON ILSVRC-2012.

2.3.1.2 VGGnet

VGGnet (Simonyan & Zisserman, 2014) is built upon the work of AlexNet in the sense that dropout is also used in order to prevent overfitting. There are fundamental differences though. The network is much deeper (16 and 19 layers) and the filter sizes are the smallest for understanding left/right, up/down and centre, with dimensions 3 × 3. Also, multiple convolutional layers are used one after the other without max pooling layers in-between. A benchmark is made on ImageNet with network's depth ranging from 11 to 19 layers. It is noticeable that results improve as the network gets deeper, going from 29.6% top-1 and 10.4% top-5 error rate (11 layers) to 25.5% and 8% error rate (19 layers). The accuracy improvement from 16 to 19 layers is very small. An absolute percentage decrease of 1.7 points is noticeable when changing network architecture from 16 layers with 1 × 1 convolutional layers to 16 layers with 3 × 3 convolutional layers.

Whilst not the winner of ILSVRC in that year, VGGnet is a network's architecture widely used for fine-tuning, for its simplicity.

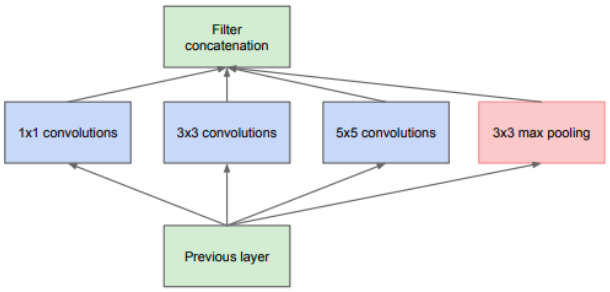
2.3.1.3 GoogleNet/Inception

GoogleNet (Szegedy, 2013) made more progress on how CNNs are engineered, in order to improve accuracy. It was relevant because it did not follow the typical convolution-pooling architecture. It introduced the concept of an inception module: a series of different steps that could be executed in parallel (1 × 1 conv, 1 × 1 conv followed by 3 × 3 conv, 1 × 1 conv followed by 5 × 5 conv and 3 × 3 max pooling followed by 1 × 1 conv. These would consist on an inception module with dimensionality reduction, since it uses 1 × 1 conv). It is also deeper than VGGnet, with 22 convolutional layers, while being computationally reasonable (only 5 million parameters, 12 times less than AlexNet). Two drawbacks of simply increasing the network size are outlined: performance reduction and the fact that bigger conv-nets lead to greater number of parameters to learn which in turn may lead to overfitting.

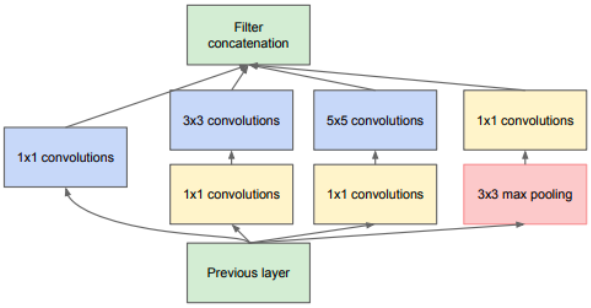
The inception module (Fig. 6) consists in the application of differently sized filters and then concatenating the result. The output volume has the depth of the sum of all the filters' convolution operation which can be executed in parallel. 1 × 1 conv operations are applied in advance for depth reduction while maintaining dimensions, for performance optimization.

The network starts with a regular conv-pool architecture and then multiple inception modules are applied. Fully connected layers are removed for parameter computation reduction.

GoogleNet (Inception v1) won ILSVRC-2014 with 6.7% top-5 error rate. Slightly better than VGGnet. In just two years' time the top-5 error rate decreased from 16.5% to 6.7%.



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

Figure 6 - Inception Module (Szegedy et al., 2015)

2.3.1.4 ResNet

With 152 layers ResNet (He et al., 2016) won the ILSVRC-2015 with 3.57% top-5 error rate (on an ensemble architecture, usually responsible for a 2% improvement). It proved the point that networks should be deeper but not plainly deeper. Plain convolutional networks with 56 layers show worse results than a 20-layer network. It is hypothesized that deeper networks are harder to optimize. As ResNet networks get deeper, on the other hand, results get better. It is hypothesized that deeper networks should be as good as shallower networks.

The concept of identity mapping or skip connection is introduced to pass through stacked layers (two weight layers in the image, commonly 3 × 3 conv in a ResNet) that do not have useful weights. Furthermore, in regular convolutional networks, there is the assumption that each layer only needs to know about what is learnt in the previous layer. With the identity mapping, information is preserved across multiple layers. By doing this, performance is not impacted by an ever-deeper network.

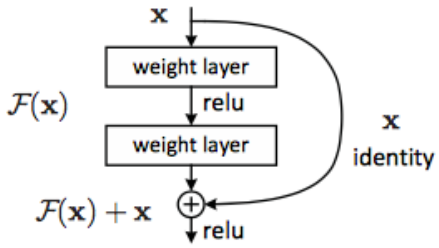


Figure 7 - Identity mapping (He et al., 2016)

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Figure 8 - ResNet accuracy comparison (He et al., 2016)

A plain conv-net (like VGGnet) shows worse results as it gets deeper (plain-34 in Fig.8). A ResNet on the other hand shows almost a 3% top-1 improvement when going from a ResNet-34 to a ResNet-152.

2.3.1.5 Inception-ResNet

In 2016 a new CNN architecture was presented (Szegedy, Ioffe, Vanhoucke, & Alemi, 2016) that joined the key concepts behind Inception and ResNet: the inception module and the skip connection. It outperforms both ResNet-152 and Inception on the ILSVRC with a top-1 and top-5 error of 19.9% and 4.9% respectively. The inception module of Inception allows to concatenate feature volumes of different kernel sizes (different sized focal area) and the skip connection of ResNet allows for the network to be deeper whilst improving precision.

2.3.2 Object Detection/Localization

As described in section 2.3.1 all the network's architectures mentioned and their scores refer to the object classification task. However, ILSVRC (Russakovsky et al., 2015) also has an object localization challenge where each training example has the corresponding bounding box of where the object is located in the image. Most of the architectures described in section 2.3.1 also compete for the object detection challenge. VGGnet won the localization challenge of that year (2015).

The output of an object localization architecture can be:

- Class specific if the output is the bounding box coordinates plus the class associated.
- Class agnostic if the output is only the bounding box.

AlexNet (Krizhevsky et al., 2012) won the ILSVRC-2012 classification and localization challenge even though their architecture for localization was not published by the authors.

Overfeat (Sermanet et al., 2013) won the ILSVRC-2013 localization challenge with its architecture being very similar to the one of (Krizhevsky et al., 2012). Localization is implemented through a method of sliding window. A CNN is applied to multiple crops of an image and the crop is labelled with an object class. The drawback is that passing multiple crops of an image to a CNN is computationally expensive since a forward pass must be done on the network for each of them. One solution is that of Selective Search (Uijlings, Sande, Gevers, & Smeulders, 2012) where a bottom-up approach is made to find regions of interest. Then regions are continuously merged until we are confident of the regions that remain. Finally bounding boxes' coordinates are extracted from the remaining regions.

Overfeat had the lowest top-5 error rate of 29.9% on the ILSVRC13 localization challenge and the highest mean average precision (mAP) with a value of 24.3%. mAP is a metric for evaluating how good the model is performing object detection. Average precision is the area of the precision/recall curve which translates to how much of a balance there is between precision and recall. The mean average precision is the mean, over all classes, of the average precision.

On 2014, R-CNN (Girshick, Donahue, Darrell, & Malik, 2014) appeared and beated Overfeat with a mAP of 31.4%, making use of Selective Search to generate 2000 region proposals at test-time that are then merged consecutively until a final region proposal is identified.

R-CNN proved to be slow because a forward pass must be done on the CNN for each region proposal. R-CNN was iterated to include the bottleneck steps into the CNN training (Girshick, 2015; Ren, He, Girshick, & Sun, 2017). Test-time speed went from 49 secs per image on R-CNN to 2.3 sec on Fast-RCNN (Girshick, 2015), to 0.2 sec on Faster-RCNN (Ren et al., 2017).

YOLO (Redmon, Divvala, Girshick, & Farhadi, 2015) and SSD (W. Liu et al., 2016) differ from previous work as all the steps in object detection are included in the CNN training step. YOLO only proposes 98 bounding boxes when comparing with the 2000 of Selective Search. With this they achieve 21 frames per second inference speed while maintaining a high mAP value. Their fast version can detect objects at 155 frames per second (Table 1). In the case of SSD their core relies on predicting box offsets by using small convolutional filters applied to feature maps. This allows to increase mAP significantly.

For *instance segmentation*, an extension is made to Faster-R-CNN to identify, pixel by pixel, the boundaries of an object in an image. A branch (fully convolutional network) is added to the architecture of Faster-R-CNN that runs in parallel with the bounding box and classification regression network giving as an output a matrix of 1s and 0s depending whether the pixel corresponds to the object or not.

Mask R-CNN (He, Gkioxari, Dollár, & Girshick, 2017) beated the winners of COCO 2015 and 2016 segmentation challenges.

Table 1 - Comparison between different object detection architectures (W. Liu et al., 2016)

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

2.4 Cars Fine-grained Classification

Fine-grained classifications tasks try to classify objects that are very similar visually, such as breeds of dogs, species of birds or models of cars.

With respect to public datasets of models of cars it is difficult to find and build one that is fine-grained and has enough quantity and quality as the dataset usually needs to be manually labelled.

We will review existing datasets of cars in the upcoming sections of this chapter.

2.4.1 CompCars (2015)

CompCars (Luo, Loy, & Tang, 2015) is composed of 431 different makes and models. 30955 images depict the entire car and 20349 images only represent parts of it. The dataset is fine-grained to the model level. The authors fine-tuned their dataset on three popular CNN architectures, achieving the following results:

Table 2 - CompCars results (Luo et al., 2015)

Model	AlexNet	Overfeat	GoogLeNet
Top-1	0.819	0.879	0.912
Top-5	0.940	0.969	0.981

Dehghan, Masood, Shu, & Ortiz (2017) first collected a 3 million image dataset of makes and models. They trained their dataset on a CNN whose architecture is not mentioned. The trained model can recognize 59 vehicles makes and 818 models. Their data collection strategy was:

- Collect 5 million images from various sources
- Semi-automated process to remove undesired images (which might have had impact on the training phase)
- A team of human annotators remove the remaining undesired images

There is no information about the accuracy of this first training phase.

Their data pre-processing step consisted of aligning the images to the centre and removing the background through bounding boxes. Background has shown to have great impact on image recognition tasks since it may have misleading information. Afterwards they fine-tune their model to CompCars. They achieve 95.88% top-1 and 99.53% top-5 accuracy. While the performance is state-of-the-art we cannot make further conclusions since no information is provided about the performance on the 3 million image dataset.

2.4.2 Stanford Cars (2013)

Stanford Cars dataset (Krause et al., 2013) is composed of 16185 images across 196 different car models. The authors leveraged the work of Lazebnik, Schmid, & Ponce (2006) and their previous work (Deng, Krause, & Fei-Fei, 2013) from 2D to 3D representations of objects for fine-grained classification, and set the accuracy baseline at 75.5% with an ensemble of both methods. This was the dataset's accuracy baseline with the use of a pre-deep learning approach.

In the work of Krause, Gebru, Deng, Li, & Li (2014) a mix of CNN and HOG features is applied to better distinguish cars' makes and models. Images are compared among similar ones by detecting the nearest neighbours in terms of HOG features and the background is removed to prevent noise. Parts are then selected by choosing the features with highest energy in terms of HOG. Results are 73.9% top-1 accuracy.

In the work of D. Liu & Wang (2016) VGGnet, GoogLeNet and CaffeNet (similar to AlexNet) are used as pre-trained networks and transfer learning is applied, on the Stanford Cars Dataset, which extrapolates the use of pre-trained networks employed in some classification task (for example ImageNet) and repurposes them for another task (for example make and model recognition). Bounding boxes, which are available in the dataset, are used for background removal. Both GoogLeNet and VGGnet are improvements over AlexNet therefore they obtain far better results.

GoogLeNet and VGGnet with around 80% top-1 accuracy and 95% top-5 accuracy. Transfer learning is considered necessary when dealing with a large number of classes to predict and a small dataset.

Krause, Jin, Yang, & Li (2015) implemented an unsupervised pipeline. Co-segmentation is used to identify the mask (pixel by pixel boundaries) of similar objects. A graph-like approach is made to group together objects, which belong to the same class, according to their pose. The cosine distance between the fourth-layer convolutional features is measured to identify objects in similar poses. Parts are generated without hand-made part annotations by setting up several points in the object mask. These points are then expanded so that they cover an area according to the object bounding box (Fig. 9). Their results are state-of-the-art with 92.8% accuracy.

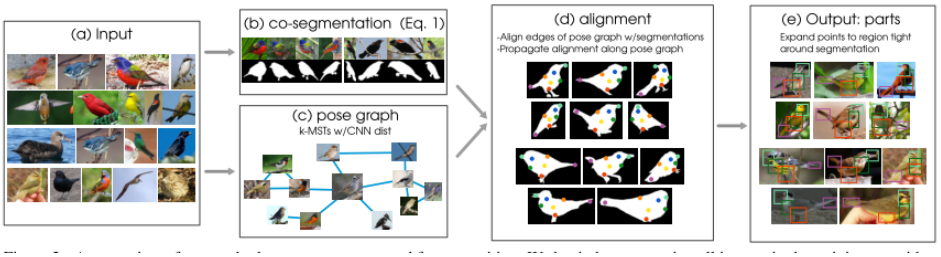


Figure 9 - Krause et al., 2015 Pipeline (Krause et al., 2015)

Dehghan et al. (2017), as mentioned in section 2.4.1, also fine-tuned their model to Stanford Cars and achieved 93.6% top-1 accuracy.

2.4.3 BoxCars (2017)

BoxCars (Sochor, Herout, & Havel, 2016) is a cars dataset of 148 fine-grained classes within 63750 images. The authors take advantage of the fact that the images are taken from road surveillance cameras to, through the viewpoint, determine their 3D bounding boxes and normalize/unpack the image into a plane. The plane image contains the cars' front, rear, side and roof. To exclude some classes with too few examples they build two datasets as follows:

Table 3 - BoxCars datasets (Sochor et al., 2016)

	# types	# training samples	# test samples
medium	77	40,152	19,590
hard	87	37,689	18,939

The medium dataset is composed of 77 classes of make, model and sub-models. In the hard dataset, the year is also considered. They fine-tuned a CNN with the following results, where *Unp* means the normalized/unpacked dataset:

Table 4 - BoxCars results (Sochor et al., 2016)

	baseline		Unp	
	med	hard	med	hard
Top-1	0.772	0.733	**0.832**	0.794
Top-5	0.924	0.903	**0.945**	0.926

2.4.4 VMMRdb (2017)

VMMRdb (Tafazzoli et al., 2017) is a fine-grained make, model, year dataset composed of 291 thousand images belonging to 9170 classes. The vehicles market release years range from 1950 to 2016. As there are few examples per class (mean class distribution of 32), two datasets are presented:

- VMMRdb-3036: where only classes with more than 20 images are selected. Totals 246.000 images.
- VMMRdb-51: with 51 classes where the year is ignored.

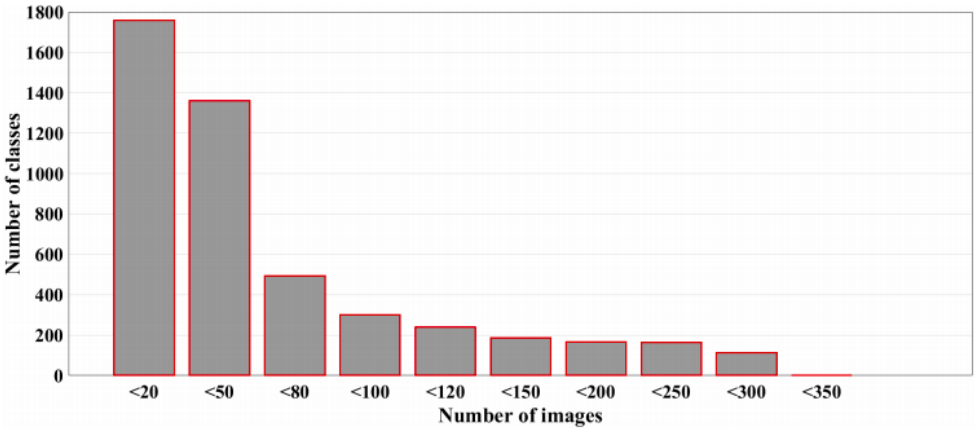


Figure 10 - VMMRdb-3036 class distribution (Tafazzoli et al., 2017)

The authors fine-tune a ResNet on both datasets. Results are 90.26% top-1 and 93.48% top-3 accuracy on VMMRdb-51. They fine-tune VMMRdb-3036 with a learning rate of 0.01 for 200 epochs on a GTX 1080 for 110 hours. They apply horizontal flip transformations. Results are 51.76% top-1 and 92.90% top-5 accuracy. Authors claim that the low accuracy is justified that with the inclusion of the year and by having many classes the level of difficulty for fine-grained classification increases. We partially agree but point out that almost half of the classes only have 20-50 examples of images (Fig. 10) which is remarkably low. Furthermore, only horizontal flip is applied as a data augmentation technique.

We conclude that the dataset, in general, lacks examples per class - partially because getting labelled data is a difficult task.

Chapter 3

Stanford Cars Benchmark

In this chapter, Stanford Cars dataset is used to perform different experiments with different approaches. We chose Stanford Cars dataset as it is the public dataset with highest quality, in terms of the distribution of examples per class, and the fact that it has car images in different viewpoints. Furthermore, it is not a medium-to-large scale dataset which is useful for carrying multiple experiments in shorter periods of time. The objective of the experiments carried throughout this chapter is to understand how different data augmentation techniques impact performance when using convolutional neural networks. The results obtained will reinforce the need for datasets of higher quality and quantity for the task of fine-grained classification. While we have the intuition that applying data augmentation leads to guaranteed benefits we want to understand how much they impact, something we did not find in the research reviewed.

A thorough analysis of how CNNs perform in fine-grained classification is done. Furthermore, we achieve state-of-the-art results when making use of the latest developments in CNN architectures. We extend the work of D. Liu & Wang (2016) and achieve a top-1 accuracy of 88.3% with the use of data augmentation when comparing with the authors 80%. Furthermore, we increase the accuracy to 91% with an ensemble of three models.

We found the dataset sensitive to certain hyperparameters. Hyperparameters are configurable values that are set before the training process begins. In the context of transfer learning these can be the optimizer algorithm such as Stochastic Gradient Descent (SGD), the learning rate and the batch size. Moreover, when carrying experiments on a 50-50 split of the Stanford Cars dataset, the models were very sensitive to certain learning rate values. Picking a too high learning rate value

caused the network to overfit too fast and picking a too low learning rate value caused the network to not "learn" at all.

In this chapter and on the following we use Keras (initially released in 2015). Keras is a deep learning library built as a wrapper over Tensorflow. We use it for its well-defined and flexible API, even though other deep learning libraries exist such as Caffe (2013), TensorFlow (2015), PyTorch (2016) and MxNet (2017).

In terms of hardware/platform for training and inference we use a Nvidia GTX 1060 and Google Colab for parallelizing certain experiments. For comparison, considering the same batch size, an epoch on a GTX 1060 is 33% faster at training than Google Colab which uses a shared Tesla K80. Batch size refers to the number of samples that will be propagated through the network in each step. The bigger the batch size the more GPU RAM is necessary.

3.1 Stanford Cars Fine-tuning

Stanford Cars dataset is composed of 196 classes of make, model and year. The dataset comes in a 50-50 split of 8,144 training images and 8,041 testing images. Each image in the dataset has the corresponding label and bounding box. Fig. 11 shows the number of classes per make.

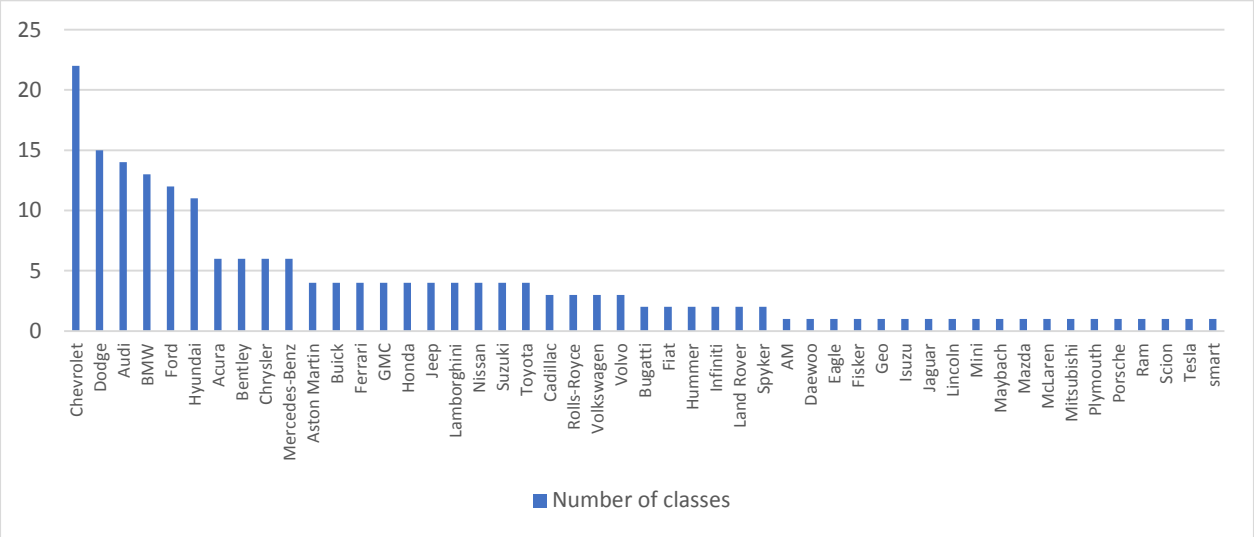


Figure 11 - Number of classes per make

In most experiments described throughout this chapter we utilize VGGnet (16 layers version) as the chosen CNN architecture. The network is pre-initialized with ImageNet weights and transfer learning is applied to repurpose it for the make, model and year classification task. For all experiments the hyperparameters used are the following:

- Learning rate: 0.005
- Batch size: 16 (number of images passed on the network in each iteration)
- Stochastic Gradient Descent optimizer

In the case of fine-tuning VGGnet, two thirds of the network layers are kept frozen which means that their weights will not be updated. More layers should be unfrozen if the model is not overfitting to training data or performance is not improving. With enough data, it is possible to fine-tune the entire network or even train a network from scratch, but, as experiments will show, even though Stanford Cars dataset is a good benchmark dataset, it does not have enough data – nevertheless we achieve good results.

For the first experiment, we start by making no modifications on the dataset: the 50-50 train-test split is maintained, and no resizing is applied (Fig. 12). In the second experiment, we removed most of the background and resized the images by using the provided bounding boxes (Fig. 13). In the third experiment, besides removing the background, we applied horizontal flip as a data augmentation technique (Fig. 14). The network is fine-tuned for 15 epochs for each experiment. An epoch is a pass of the entire training data over the network until the test accuracy is calculated. Considering we only have 16 thousand images to train on, each epoch takes approximately four minutes.

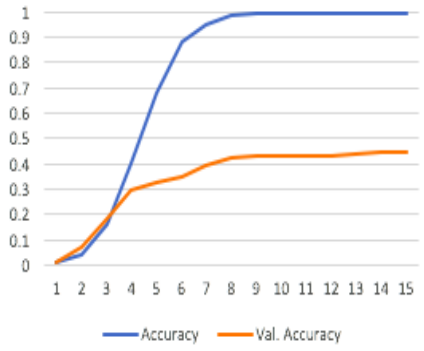


Figure 12 - Without resizing

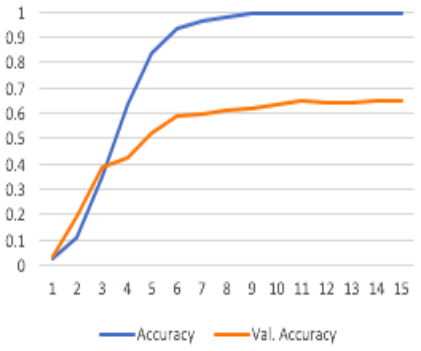


Figure 13 - With resizing

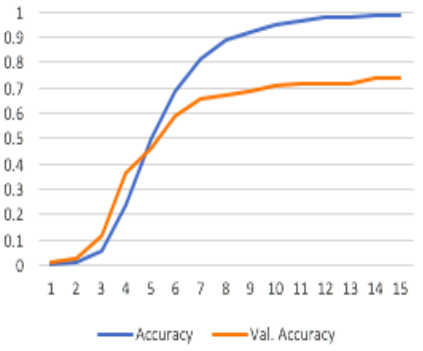


Figure 14 - Resizing and horizontal flip

In Figs. 12 and 13 we can see that the model overfits rapidly to training data. We can see that until the 3rd-4th epoch the model is performing well but afterwards test accuracy starts plateauing whilst training accuracy continues to increase until it reaches the value of 1. Even though both experiments show overfitting to training data, the test accuracy proves to be 0.2 higher when applying bounding boxes (Fig. 13). In the case of the third experiment, by applying both horizontal flip as a data augmentation technique and background removal, the test accuracy proves to be 0.3 higher (Fig. 14).

While all the three experiments overfit to the training data, observed by the gap between train and test accuracy, the model takes longer to overfit when bounding boxes and horizontal flip is applied.

Figs. 15 and 16 show the train and test accuracy evolution by using the following data augmentation (DA) techniques:

- Shear range: 0.2
- Zoom range: 0.2
- Horizontal flip
- Horizontal shift range: 0.2
- Vertical shift range: 0.2

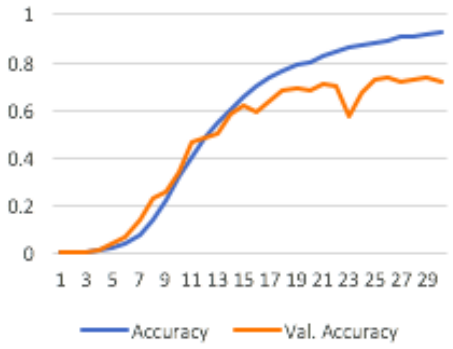


Figure 15 - VGGnet with DA

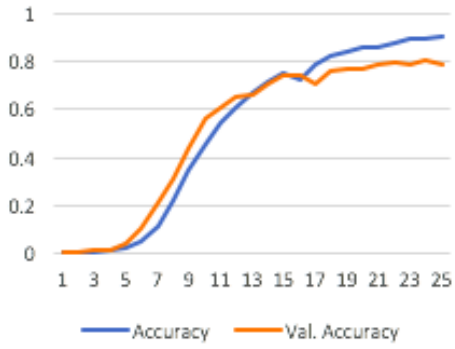
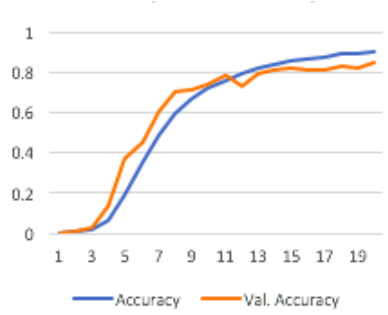


Figure 16 - VGGnet with DA and resizing

For Fig. 16, images are resized by applying the provided bounding boxes. Besides trying to understand the impact of applying multiple data augmentation techniques we also want to see the difference when the background is also removed.

We can observe that, by applying multiple DA techniques, the model takes longer to overfit and the test accuracy increases drastically, reaching 71%. By applying bounding boxes in conjunction with the same DA techniques, the test accuracy reaches 80%, whilst taking fewer epochs.

Given the improvements obtained by using data augmentation and background removal we proceed with a performance comparison of three different CNN architectures (VGGnet, ResNet50 and InceptionResNetV2), this time on a 70-30 split, with DA and background removal (Figs. 17-19) on the Stanford Cars dataset.



. Figure 17 - VGGnet

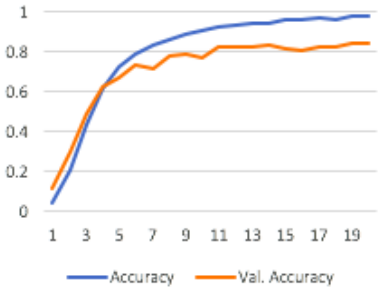


Figure 18 - InceptionResNetV2

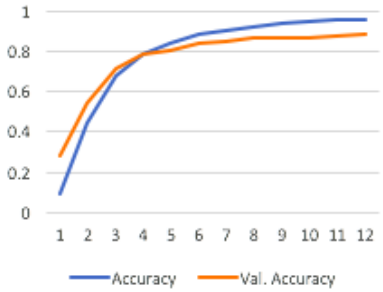


Figure 19 - ResNet50

We observe that, with the 70-30 split, the model is overfitting less and the test accuracy is:

- 85% for VGGnet
- 84% for InceptionResNetV2
- 88.3% for ResNet50

The better results that were achieved, when compared with the 50-50 split, may be due to the lack of representative training data on a 50-50 split. Data augmentation proves to be useful, in increasing performance, in scenarios where getting more data is not an option (although more raw data is preferred).

The results presented are better than those described by D. Liu & Wang (2016). A justification for these superior results could be that the authors do not refer the use of data augmentation which is used in the above described experiments and effectively improves performance.

3.2 Model Dissimilarity

In this section, we want to understand how the VGGnet and InceptionResNetV2, trained in the previous section, differ from each other.

In Tables 5-8 we compare the top-20 classes of the lowest and highest performing classes with respect to their f1-score on the VGGnet and InceptionResNetV2. 12 out of 20 classes from the top-20 lowest classes match between the two models whereas 5 out of 20 classes from the top-20 highest classes match.

Table 5 - VGGnet worst performing classes by F1-score

Class	f1_score	precision	recall	support
Chevrolet Express Cargo Van 2007	0.18	0.18	0.18	11.0
Chevrolet Express Van 2007	0.41	0.41	0.41	17.0
Acura RL Sedan 2012	0.5	0.83	0.36	14.0
Audi TTS Coupe 2012	0.5	0.75	0.38	24.0
Audi S5 Coupe 2012	0.54	0.5	0.58	24.0
Hyundai Accent Sedan 2012	0.55	0.38	1.0	6.0
Audi 100 Sedan 1994	0.58	0.57	0.59	22.0
Audi S4 Sedan 2012	0.58	0.9	0.43	21.0
Audi TT Hatchback 2011	0.58	0.48	0.73	22.0
Dodge Caliber Wagon 2012	0.59	0.63	0.55	22.0
Chevrolet Tahoe Hybrid SUV 2012	0.61	0.71	0.53	19.0
BMW M6 Convertible 2010	0.62	0.55	0.7	23.0
Ferrari 458 Italia Coupe 2012	0.62	0.8	0.5	24.0
Audi TT RS Coupe 2012	0.63	0.71	0.57	21.0
BMW 6 Series Convertible 2007	0.63	0.56	0.73	26.0
Chevrolet Silverado 1500 Regular Cab 2012	0.63	0.58	0.69	26.0
Chevrolet Silverado 2500HD Regular Cab 2012	0.63	0.67	0.6	20.0
Audi 100 Wagon 1994	0.64	0.54	0.79	24.0
Audi A5 Coupe 2012	0.64	0.59	0.7	23.0
Bugatti Veyron 16.4 Convertible 2009	0.64	0.73	0.57	14.0

Table 6- InceptionResNetV2 worst performing classes by F1-score

Class	f1_score	precision	recall	support
Chevrolet Silverado 1500 Regular Cab 2012	0.07	1.0	0.04	26.0
Chevrolet Express Van 2007	0.31	0.44	0.24	17.0
Chevrolet Express Cargo Van 2007	0.37	0.6	0.27	11.0
Mercedes-Benz Sprinter Van 2012	0.41	1.0	0.26	23.0
Aston Martin Virage Convertible 2012	0.42	1.0	0.27	15.0
Audi S4 Sedan 2012	0.45	0.7	0.33	21.0
Acura RL Sedan 2012	0.48	0.36	0.71	14.0
Chevrolet Silverado 1500 Hybrid Crew Cab 2012	0.52	0.5	0.55	22.0
Audi S5 Coupe 2012	0.54	0.54	0.54	24.0
Audi 100 Wagon 1994	0.57	0.52	0.62	24.0
Bentley Continental GT Coupe 2007	0.59	0.92	0.43	28.0
Audi TT Hatchback 2011	0.6	0.67	0.55	22.0
Chevrolet Silverado 2500HD Regular Cab 2012	0.6	0.44	0.95	20.0
Audi 100 Sedan 1994	0.63	0.53	0.77	22.0
Dodge Caliber Wagon 2012	0.65	0.59	0.73	22.0
Chevrolet Malibu Sedan 2007	0.65	0.51	0.88	26.0
Audi A5 Coupe 2012	0.66	0.54	0.83	23.0
BMW M6 Convertible 2010	0.67	0.81	0.57	23.0
Honda Accord Sedan 2012	0.67	0.51	0.95	20.0
Chevrolet Avalanche Crew Cab 2012	0.68	0.88	0.56	27.0

Table 7 - VGGnet best performing classes by F1-score

Class	f1_score	precision	recall	support
Chevrolet HHR SS 2010	1.0	1.0	1.0	18.0
Chrysler PT Cruiser Convertible 2008	1.0	1.0	1.0	27.0
Infiniti QX56 SUV 2011	1.0	1.0	1.0	14.0
BMW X3 SUV 2012	0.98	0.95	1.0	20.0
Buick Enclave SUV 2012	0.98	1.0	0.96	24.0
Hyundai Tucson SUV 2012	0.98	0.96	1.0	25.0
Jeep Wrangler SUV 2012	0.98	1.0	0.96	25.0
Lincoln Town Car Sedan 2011	0.98	1.0	0.95	21.0
Buick Verano Sedan 2012	0.97	0.95	1.0	19.0
Fiat 500 Convertible 2012	0.97	1.0	0.93	15.0
Ford E-Series Wagon Van 2012	0.97	0.95	1.0	19.0
Acura Integra Type R 2001	0.96	0.93	1.0	26.0
Cadillac SRX SUV 2012	0.96	0.96	0.96	23.0
GMC Acadia SUV 2012	0.96	0.96	0.96	26.0
Hyundai Genesis Sedan 2012	0.96	1.0	0.92	25.0
Volkswagen Beetle Hatchback 2012	0.96	0.92	1.0	24.0
Volvo XC90 SUV 2007	0.96	0.93	1.0	25.0
Fiat 500 Abarth 2012	0.95	0.9	1.0	9.0
Isuzu Ascender SUV 2008	0.95	1.0	0.91	22.0
Nissan 240SX Coupe 1998	0.95	0.9	1.0	28.0

Table 8 - InceptionResNetV2 best performing classes by F1-score

Class	f1_score	precision	recall	support
Chevrolet HHR SS 2010	1.0	1.0	1.0	18.0
Chrysler PT Cruiser Convertible 2008	1.0	1.0	1.0	27.0
Volkswagen Beetle Hatchback 2012	1.0	1.0	1.0	24.0
GMC Terrain SUV 2012	0.98	1.0	0.96	23.0
Land Rover Range Rover SUV 2012	0.98	1.0	0.96	24.0
Nissan Leaf Hatchback 2012	0.98	1.0	0.96	24.0
Ford F-150 Regular Cab 2012	0.98	0.96	1.0	24.0
Jeep Wrangler SUV 2012	0.98	0.96	1.0	25.0
Plymouth Neon Coupe 1999	0.98	0.96	1.0	26.0
Buick Verano Sedan 2012	0.97	1.0	0.95	19.0
Ford E-Series Wagon Van 2012	0.97	1.0	0.95	19.0
Mercedes-Benz SL-Class Coupe 2009	0.97	1.0	0.94	18.0
Nissan NV Passenger Van 2012	0.97	1.0	0.95	20.0
Mazda Tribute SUV 2011	0.97	0.95	1.0	18.0
Buick Enclave SUV 2012	0.96	1.0	0.92	24.0
Ford Fiesta Sedan 2012	0.96	1.0	0.92	24.0
Volvo XC90 SUV 2007	0.96	1.0	0.92	25.0
Geo Metro Convertible 1993	0.96	0.96	0.96	26.0
Volvo 240 Sedan 1993	0.96	0.96	0.96	27.0
Ford F-450 Super Duty Crew Cab 2012	0.96	0.92	1.0	23.0

We further analyze the difference between the two models by plotting the f1-score absolute difference by the number of classes between VGGnet and InceptionResNetV2 (Fig. 20). This shows how different the models are at inference for each class. Fig. 20 shows that there is one class that has a f1-score difference of 0.56 and 62 classes that have a f1-score difference higher than 0.10, outlining the different performance between the two models.

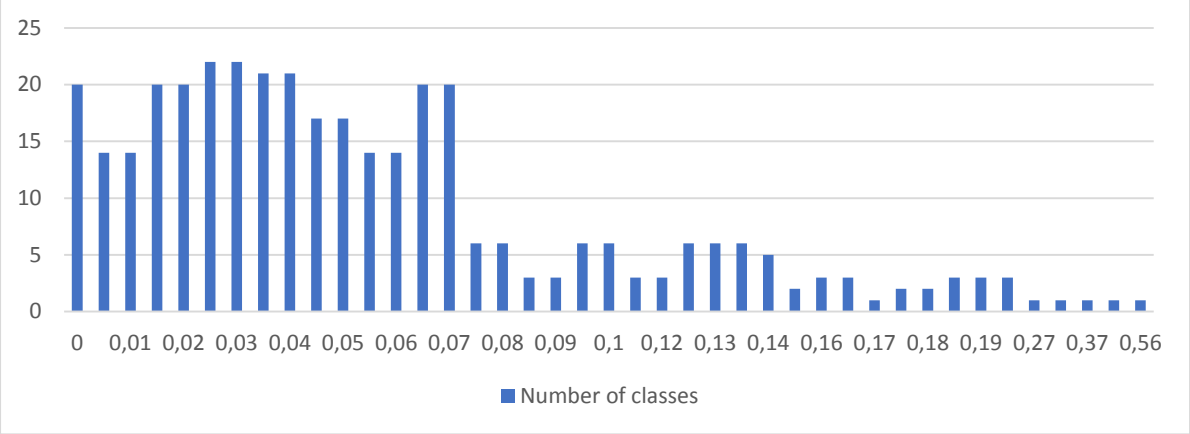


Figure 20 - F1-score absolute difference between VGGnet and InceptionResNetV2

Given the difference between the two models, in the next section we create a model ensemble that aggregates all models and averages their predictions.

3.3 Stanford Cars 3-Model Ensemble

A model ensemble is a common technique that combines predictions of different models in order to improve accuracy. This is key when different network architectures are used, since they show distinctive results with respect to what they are predicting accurately or not.

Two examples of ensemble methods are Bagging (Breiman, 1996) that trains models of one type, and Boosting (Schapire, 2001) that trains models that iteratively take into consideration previous misclassified training points, are two examples of ensemble methods.

Model ensembles are a common technique used in the ImageNet challenge and Kaggle (“Kaggle”, 2010) competitions. Ensembles are usually done on three or more models. Model

ensembles are popular but not so widespread in deep learning because of the time it takes to train each model. It is also not so feasible in real-world applications because a forward-pass (inference) would need to be performed in all the models instead of just one. One possible, but not explored technique is the one referred by Huang et al. (2017) where only one model is trained but SGD is influenced so that it reaches multiple local minima. Snapshots are taken in each local minima and an ensemble is created with all snapshots.

In the current dissertation, we make a model ensemble of the three fine-tuned models of section 3.1, VGGnet, Resnet50 and InceptionResNetV2. This is done by, at inference time, doing a forward pass of the test data in each fine-tuned model and averaging the predictions of the three models. Considering that the model outputs a probability score for each label (in the case of Stanford Cars dataset this represents a vector of 196 values for each image prediction), for creating a model ensemble we do a vote averaging by summing the vectors of the predictions and dividing by the number of models. The final vector represents the final scores.

As observed in section 3.2 only 25% of the top-20 classes with respect to their f1-score match in the two models. Figs. 21-23, below, show the respective confusion matrixes of the three models. We observe that certain data points do not match between the models. Even though the accuracy is similar, the confusion matrices are distinct in certain data points.

VGGnet had an accuracy of 85.2% (763 incorrect out of 4513 test data images), ResNet50 of 88.3% (527/4513) and InceptionResnetV2 of 84% (717/4513). By combining the two models with the lowest accuracy the accuracy increases substantially to 89% (486/4513). By adding ResNet50 the accuracy further increases to 91% (405/4513).

Fig. 24 shows the 3-model ensemble’s confusion matrix.

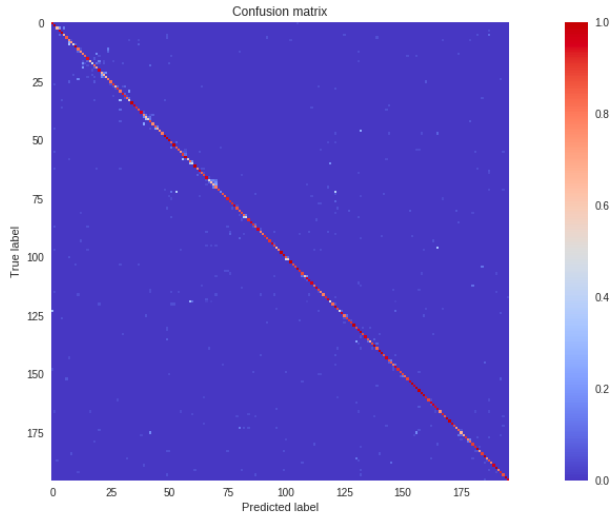


Figure 21 - VGGnet Confusion Matrix

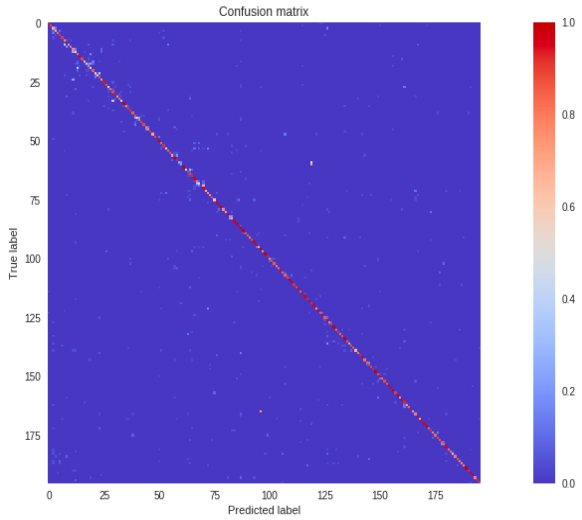


Figure 22 - InceptionResNetV2 Confusion Matrix

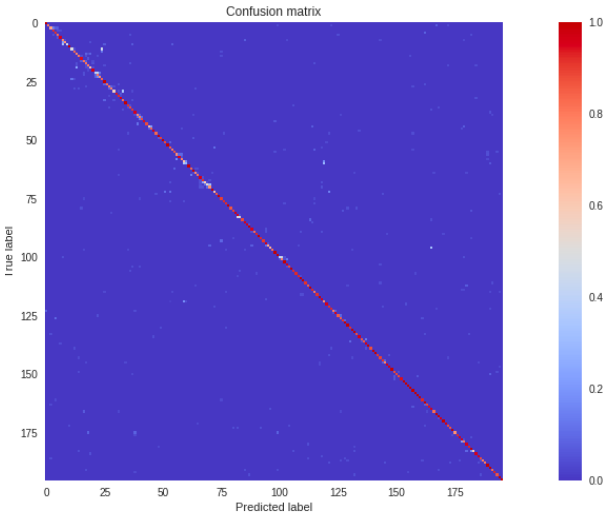


Figure 23 - ResNet50 Confusion Matrix

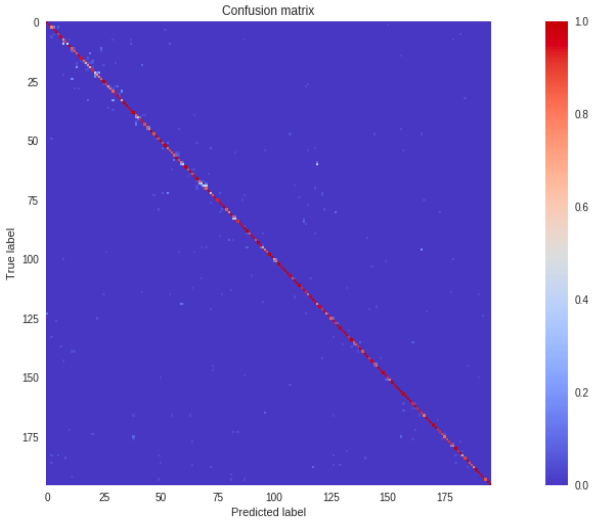


Figure 24 - Ensemble Confusion Matrix

The confusion matrices clearly show that the model is accurate in most data points. However, there are two cases where the model is not so accurate: similar looking cars belonging to the same make and similar looking cars belonging to different makes. Indices 11-24 in the confusion matrices show different Acura models. Indices 65-70 in Fig. 21 also show low class accuracy. These indices represent four distinct Chevrolet Silverado 1500 models and a Chevrolet Silverado 2500. The classes are ordered alphabetically so we can see strong data points around the line of x = y.

We are interested in looking at similar cars belonging to different makes where the model is not so certain about the correct class. For all true labels, we determine the top-5 predictions, look at high probability of wrong predictions and outline some examples:

Figure 25 - Left column - Chevrolet Express Van 2007 and GMC Savana Van 2012.
Middle column - Chevrolet Tahoe Hybrid SUV 2012 and GMC Yukon Hybrid SUV 2012.
Right column - Dodge Sprinter Cargo Van 2009 and Mercedes-Benz Sprinter Van 2012

There are more misclassifications for the cases where the correct class belongs to the same make but different model, than for the cases where the correct class belongs to a different make.

Even though the global model accuracy is quite high it is hypothesized that cars belonging to the same make but different models are being affected by the fact that there are no sufficient examples to distinguish between them on the training dataset. Of course, some cars are almost impossible to distinguish given their visual similarity, even for the human being.

It is interesting to note how the model only gets confused by very similar cars which means it is creating the right representations to distinguish among all the classes.

3.4 CNN Model Visualization

When a model is trained on a dataset it is useful to look at the representations the model is creating that allows it to distinguish among different classes and classify them correctly. It is also important to look at what parts of an input image the model is focusing on to output a class and if it is considering the background or not.

It will also be interesting to observe that the parts of the image the model is using for classifying makes and models differ from what one supposes they would be. A deep neural network learns from simple to complex filters as the network goes deeper, and it maps combinations of these filters to a set of classes. We should not anthropomorphize them (Chollet, 2017). In this section, we will consider four different practices of model visualization: activation maximization of convolutional filters, activation maximization of dense layers of specific output labels, saliency and attention (heat) maps of class activation; and apply them to the Stanford Cars dataset. We use keras-vis (Kotikalapudi & Raghavendra, 2017) as an open-source tool for model visualization.

3.4.1 Activation Maximization

In this section, we use the VGGnet model fine-tuned on the Stanford Cars dataset, described in section 3.1, for filter visualization. It obtained 85% accuracy and is chosen for its architecture simplicity when comparing with InceptionResNetV2 and ResNet50.

The network architecture is the following, depicted in Table 9:

Table 9 - VGGnet Network architecture (Generated from Keras' network summary method)

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 1024)	25691136
dense_2 (Dense)	(None, 1024)	1049600
dense_3 (Dense)	(None, 196)	200900

In Table 9, displayed above, we can see the names of the convolutional layers for which we will apply activation maximization. We can also see the output shape for each layer, such as the downsizing that is applied when passing through a MaxPooling2D layer, and the number of weights that can be learnt in each layer (Param # column). Also, depicted in Table 9, is the Flatten layer, which transforms a volume into a vector and the Dense layer which is a fully connected layer.

Figs. 26 and 27 show the input image that most activate each of the first 16 filters of 5 convolutional layers, by applying activation maximization on the fine-tuned VGGnet. Activation maximization works by starting with an image with random pixel values. This image is forward-passed on the network to compute the activation at a certain filter. Then backpropagation is applied to compute its gradient. Finally, a fraction of the gradient is added to the input image to adjust its color (Yosinski, Clune, Nguyen, Fuchs, & Lipson, 2015) (Chollet, 2017). This process is repeated until there is a high activation value for the altered input image. This process described by Erhan, Bengio, Courville, & Vincent (2009) and Zeiler & Fergus (2014) was applied on the MNIST and ImageNet datasets respectively.

As Fig. 26 shows CNNs "learn" simple looking patterns in earlier convolutional layers of the network and more complex representations in later convolutional layers (Fig. 27).

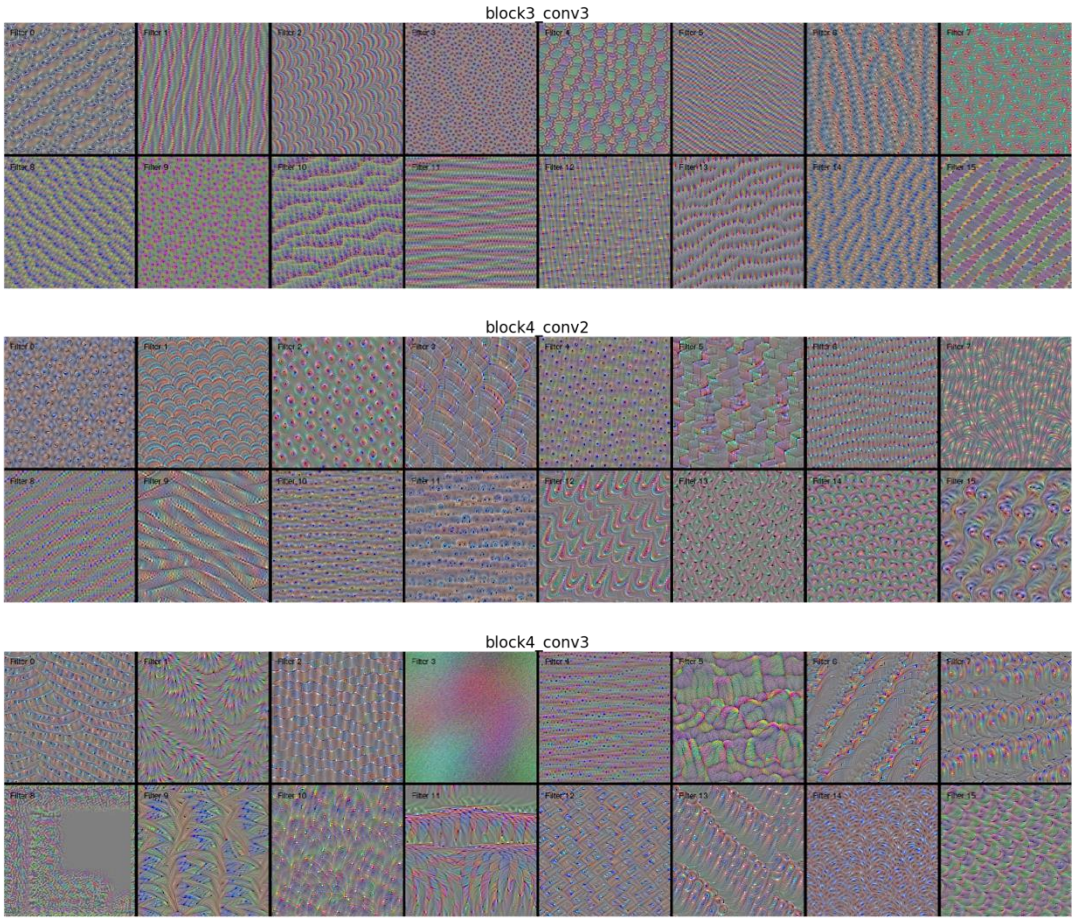


Figure 26 - VGGnet earlier layers filter visualization

This process is called convolutional filter visualization dismissing the output labels since we are not optimizing for any class but for what patterns the filter will be activated by. By using combinations of these filters, convolutional neural networks are able to classify with high precision a large number of different classes.

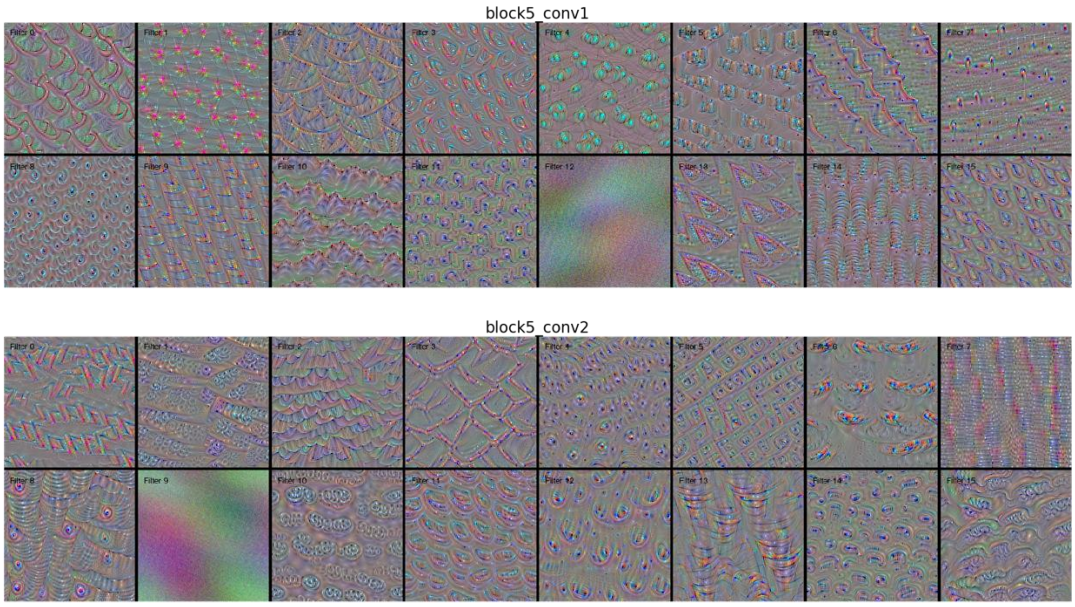


Figure 27 - VGGnet later layers filters representation

We can further visualize the input images that most activate a certain output label (Simonyan, Vedaldi, & Zisserman, 2013). This process is called activation maximization where we specify what class we are maximizing for. We will maximize an input image for a given output label and for the last layer of the CNN, the dense layer. Fig. 28, below, shows 12 classes and the input image that maximizes each of the 12 output labels. The images show mostly representations of the car's front light and grille. These are the parts of the car that most activate for each class label and what the network is mostly using for classification.

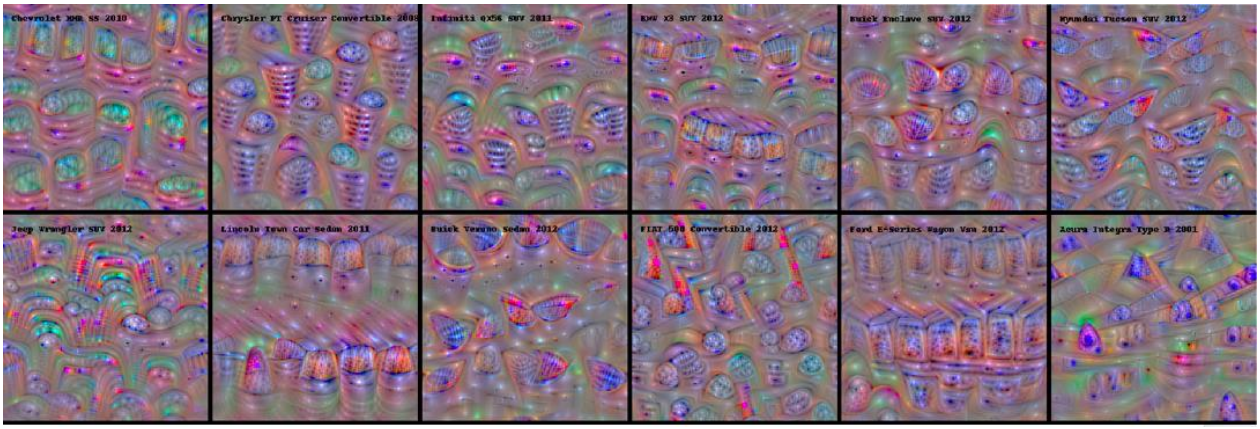


Figure 28 – Stanford Cars dataset sample classes activation maximization

3.4.2 Saliency and Attention Maps

Saliency and attention maps show the pixels that most contribute to the output label. Saliency maps were first introduced by Simonyan et al. (2013) and it works by computing the gradient for an output class for a given input image. We use keras-vis (Kotikalapudi & Raghavendra, 2017) for visualization of attention and saliency maps:

- with the guided-backpropagation method (Springenberg et al., 2014) that sets to zero negative gradients, leading to crispier results for saliency maps.
- with the grad-cam method (Selvaraju et al., 2017) for attention (heat) map visualization. Class activation heatmaps show which parts of the image most contribute for an output label and their importance (Chollet, 2017).

Both methods show, given an image, the pixel-by-pixel importance for an output label. In other domain problems, this is useful if multiple objects are represented in an image. Given an image, where objects of different classes are present, we might want to see if the network is "looking" at the right object when classifying it. In the context of fine-grained classification of cars, of classified websites, this is not the case since most images only have one car present. However, what we do want to see is if the background is being ignored and what parts of the car the network is using for prediction. This will greatly differ from what we supposed it would be. As we will see the network "learns" effectively the most distinguish part of the car that allows to distinguish and classify each of the 196 different classes of the Stanford Cars dataset.

In this section, we will observe what parts of the car the network is using for prediction. We will do this by looking at the attention and saliency maps of the same car and the output label, in different viewpoints. We will then consider different cars belonging to different make, model and years and look at their attention and saliency maps. This will allow us to validate that the network focuses on similar parts of the car for different classes but with different heatmaps.

As mentioned above we start by verifying the attention and saliency maps of a Smart Fortwo convertible (selected from the dataset in different viewpoints) to understand which parts of the image are selected for make, model and version recognition in different viewpoints (Fig. 29, below). It is interesting to observe that the network focuses on different parts of the image in different viewpoints but mostly near the front and back light. For the car portrayed in Fig. 29 the network is also focusing in the upper part of the car near the side window.

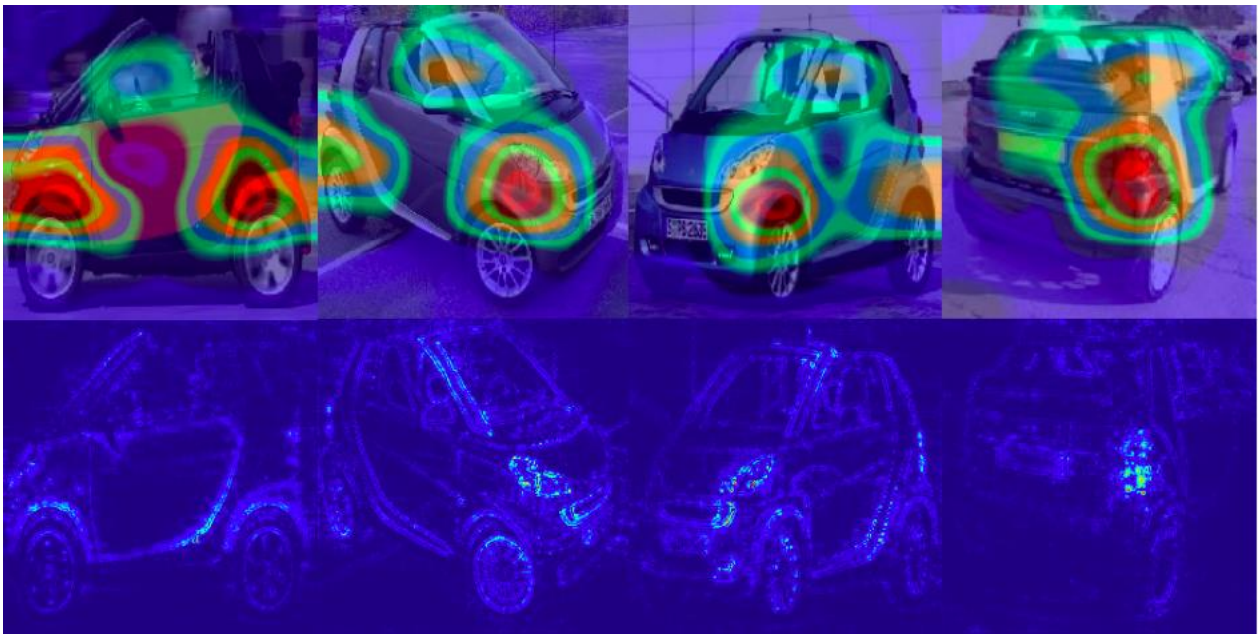


Figure 29 - Smart Fortwo Convertible 2012 attention and saliency maps

We then compare the activation maximization, attention and saliency maps for 5 different make, model and year cars, displayed in Fig. 30, below. We can observe the similarity between the input image that most contributes to an output label (activation maximization) and the attention map of each of the input images. The activation maximization representations portrayed in the upper row of Fig. 30 match the attention maps (middle row) for all the cars to a certain extent. In the top row, which shows the activation maximization, the front grid of the Hyundai, the front light and window of the Chevrolet HHR and the front lights in the case of the Fiat 500 are very visible. In the case of the heat maps (third row) we see that the background is ignored. For all the five examples portrayed in Fig. 30 we see that the part of the car most used for make, model and version recognition is the front light and grille.

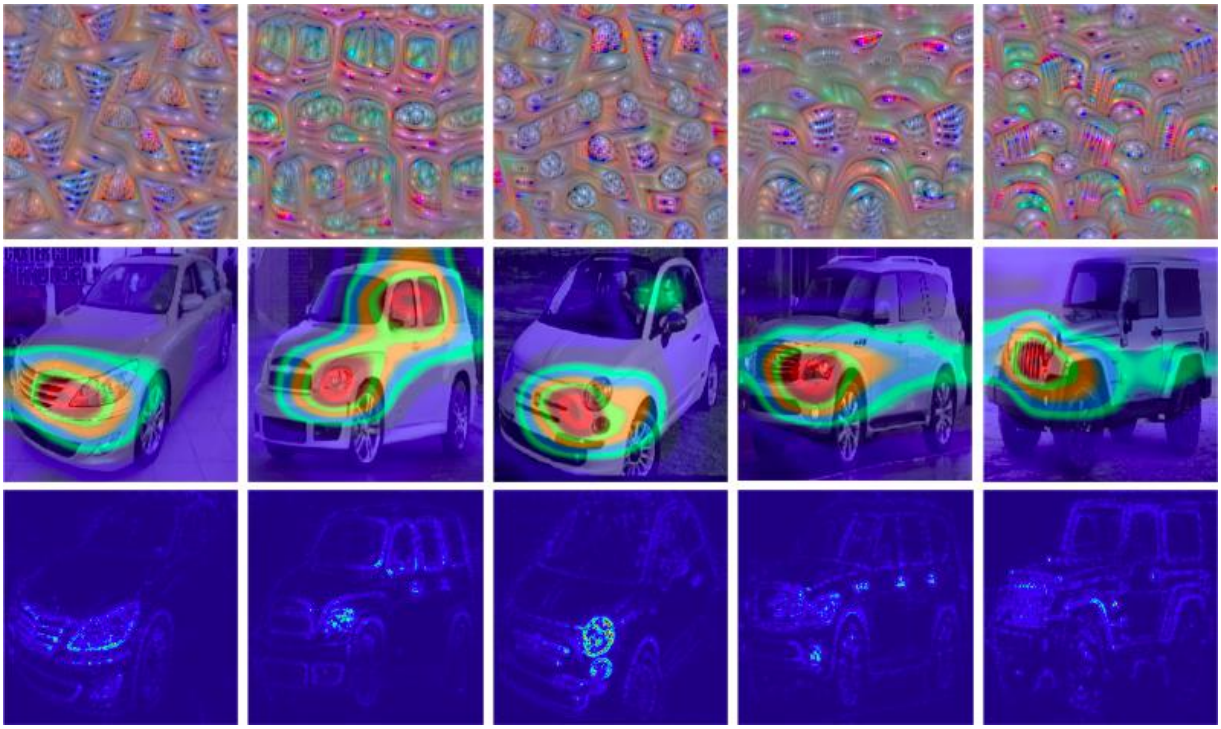


Figure 30 - Hyundai Genesis Sedan 2012, Chevrolet HHR SS 2010, Fiat 500 Convertible 2012, Infiniti QX56 SUV 2011, Jeep Wrangler SUV 2012 class filters, activation maximization and saliency maps

We proceed by looking at two classes that are very similar but present a low accuracy score. Fig. 31 shows the input images (first column) that most activate two fine-grained labels: the Audi TT Hatchback and the Audi TT RS. It also shows their similarity. Given two images from the dataset, one from each of the classes, Fig. 31 also shows the similarity in the attention and saliency maps.

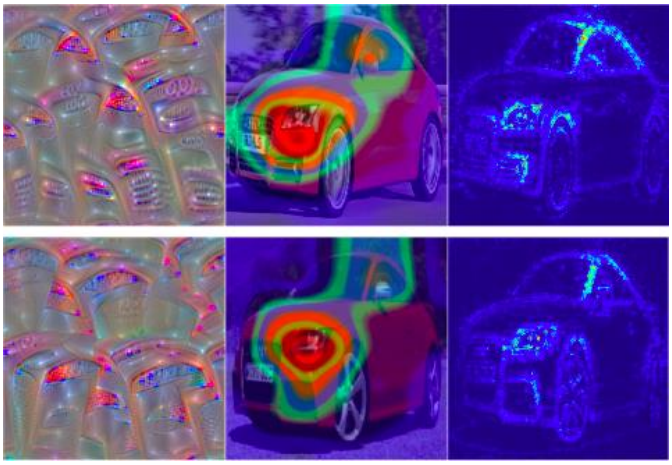


Figure 31 - On the top Audi TT Hatchback 2011 class activation maximization, heat and saliency map. On the bottom, Audi TT RS Coupe 2012 class activation maximization, heat and saliency map

3.5 Remarks

In this chapter, a benchmark of the Stanford Cars dataset was carried by applying transfer learning on different state-of-the-art CNN architectures namely: VGGnet, ResNet50 and InceptionResNetV2. We outperformed D. Liu & Wang (2016) 80% accuracy by achieving a top-1 accuracy of 88.3% with the use of several data augmentation techniques. With an ensemble of three models we increase this accuracy to 91%. The data augmentation techniques applied substantially increase the model's accuracy on the test dataset and make it more robust throughout the training process. Furthermore, the removal of the background, by applying the provided bounding boxes on each image also lead to improved results.

The subtle differences in the confusion matrixes in the different models was the intuition for performing a model ensemble, which lead to an increase of 2.7% in the test accuracy.

Activation maximization and attention and saliency maps were applied in order to understand which parts of the image are most relevant for the task of make, model and year classification and how they change in different viewpoints. The networks are mostly focusing on the front lights and the grille for distinguishing between the 196 different class possibilities.

Chapter 4

Oto-790

In this chapter, we present a new dataset, called Oto-790, collected from one of the highest traffic car classifieds website, Otomoto.

Existing public datasets for fine-grained classification of cars range from small to medium scale. Car images often need to be manually labelled, which leads to an unbalanced and small-scale dataset. We take advantage of the fact that data, in classified websites, is already labelled, to collect one of the largest datasets to date for the task of fine-grained classification of cars. It consists of 1.47 million images belonging to 790 different makes, models and versions. It surpasses all previously reviewed car datasets in terms of the number of images and classes. It also surpasses them in terms of quality, both in image quality and class distribution.

Based on the datasets reviewed in section 2.4 we felt there was a need for a large-scale dataset for two reasons:

- To demonstrate the effectiveness of CNN architectures when a larger and finer-grained dataset is used.
- To develop a model that has real-world applicability for automated content posting and moderation by having as classes the make, model and versions combinations that exist in a car classifieds website.

4.1 Oto-790 dataset

Having benchmarked Stanford Cars dataset with different architectures and approaches we proceeded to obtain a dataset.

We collected 1.48 million images belonging to a car classifieds website, Otomoto. The website has 1.7 million adverts belonging to 2147 different make, model and version. We made the decision, for time and dataset balance constrains, of only considering the most popular 797 make, model and versions which represent 96% of all adverts. Seven of these 797 classes are aggregators of unidentified model and versions (such as Other Audis) so after removing these we are left with 790 classes.

To understand the fine-grained aspect of the dataset, Fig. 32 shows the distribution of classes per make. They are distributed over 44 makes. This number is lower than the one of Stanford Cars dataset, but we have four times more classes, elucidating the fine-grained aspect of the dataset. Mercedes Benz is the make with most models and versions: 57.

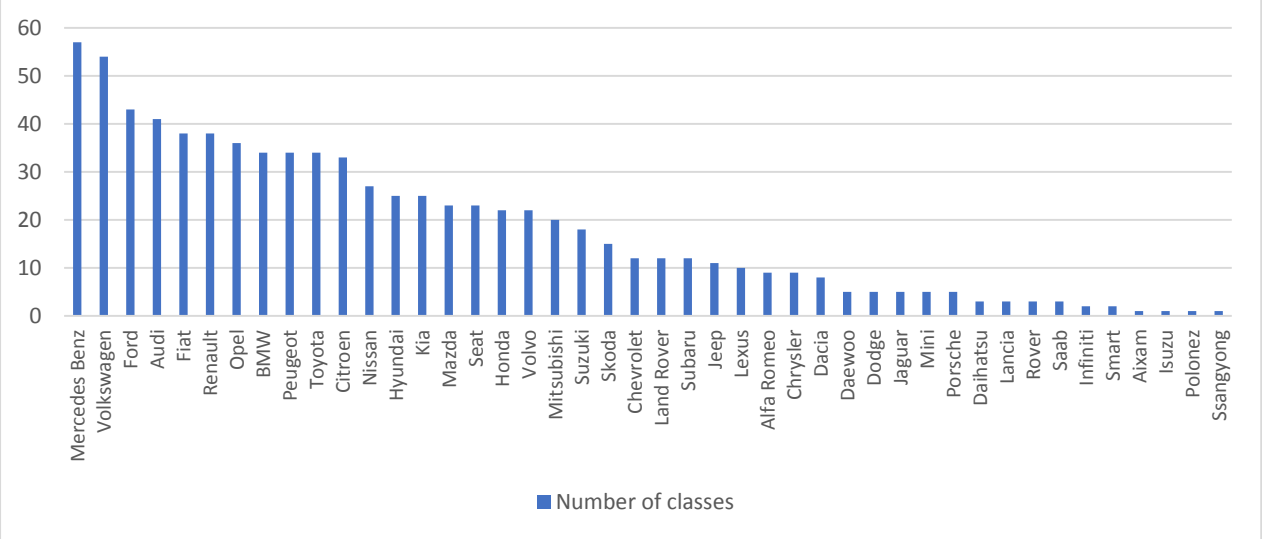


Figure 32 - Number of classes per make

A pipeline is developed to automatically obtain and clean the dataset (Fig. 33). We start by automatically downloading the first three images of each advert until a maximum of 2000 are obtained for each make, model and version. This guarantees that we obtain images in different

viewpoints. This process is done by requesting the image warehouse with each of the image identifiers. Images are then stored in a folder corresponding to each make, model and version. Images are downloaded with a 732 × 438 resolution. This first dataset consists of 1.48 million images.

This initial dataset contains images of a car's interior, wheel, luggage and seats, which are not useful images for the task of make, model and version classification. These are considered invalid images. We proceeded with labelling 1000 images of this dataset as valid and 1000 as invalid. We consider a valid image one that captures the entire car's exterior: front, side and back facing. We trained a CNN on a 70-30 split of this 2000 images' dataset to predict an image as valid or invalid. It does so with 96% test accuracy. We run this valid-invalid car classifier on the entire dataset to remove unwanted images. We are left with 1.2 million images belonging to 790 different make, model and version with a mean class distribution of 1518 images.

Given the importance the background has on accuracy and the fact that it does not contain useful information, we use Tensorflow object detection API and SSD (W. Liu et al., 2016) with MobileNet (Howard et al., 2017) trained on COCO dataset (T. Y. Lin et al., 2014) for object detection of cars. If it detects a car with enough confidence in an image (preferably on the centre), the background is removed and the aspect ratio is changed, as close as possible to a 1:1 aspect ratio, and it is considered a valid image for VMMR.

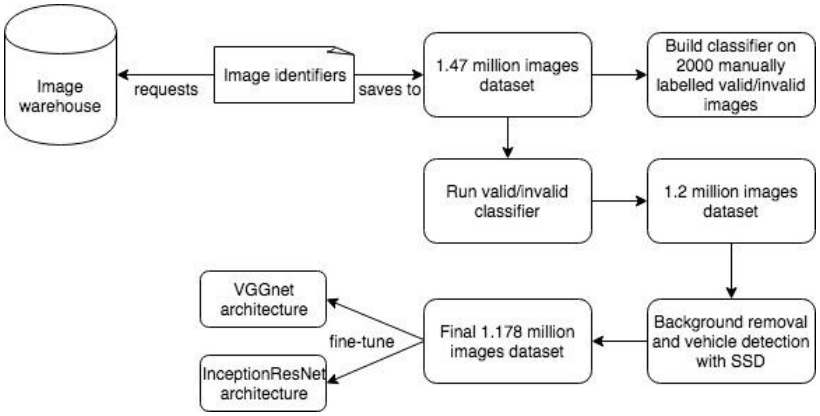


Figure 33 - Oto-790 pipeline

Since some cars are not detected in the image by SSD the final dataset consists of 1.178 million images. The dataset is split into a 70-30 train-test ratio (840 thousand training images and 360 thousand test images).

We apply transfer learning by fine-tuning an InceptionResNetV2 model, pre-initialized with ImageNet weights, on this dataset. The learning rate is set to $5e^{-4}$. Fig. 34 shows the train and test accuracy over epochs.

Until epoch 6 only the head of the network is being fine-tuned. We made this decision because we did not know, beforehand, how fine-tuning would perform at this scale. Sometimes just fine-tuning the top layers of the network is enough if the similarity between the pre-trained and the new dataset is high. It is also considered a good strategy to iteratively allow more layers to be fine-tuned after every x epochs. Given that the test accuracy was not improving considerably, we unfroze the top one third layers of the network at epoch 6. At epoch 14 another third of the network's layers is unfrozen. Each epoch takes 16 hours to run on a GTX 1060. The entire training/fine-tuning process takes two weeks of continuous computing.

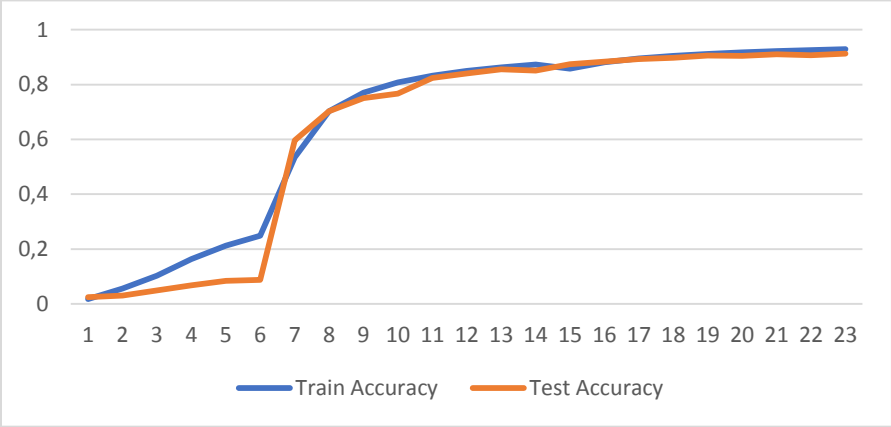


Figure 34 - InceptionResNetV2 Accuracy/Epochs (make, model and version recognition)

Data augmentation is used as in the Stanford Cars dataset benchmark to make the network not see the same data after each epoch. Given the robustness of the dataset and the low learning rate, the training process could have started with only a third of the layers frozen (to avoid the irregular increase at epoch 6).

We only fine-tune 2/3s of the network's layers for three reasons:

- GPU memory limits. Considering a GTX 1060 has 6GB of ram it is not possible, with the same batch size, to train all the layers of the network.
- Early layers of a network only activate for low-level cues and parts that are general to different domains.

- A high test-accuracy is achieved, regardless of not training the entire network.

InceptionResNetV2 achieves a 91.3% top-1 test accuracy (Fig. 34, displayed above) which means the model predicts the correct class 91.3% of the times for all the 360 thousand images of the test set.

Additionally, the top-3 accuracy is 97.75% and top-5 accuracy is 98.35% (Fig. 35). It is important to look at the top-x accuracy when dealing with a dataset that was not manually labelled/validated for two reasons: to understand how much the model is mislabelling at the fine-grained model/version level; to understand how many mislabelled images are there in the dataset. To understand how many images are mislabelled/invalid we look at the top-30 accuracy: 99.15%. This means that, for all the test examples, the model does not predict correctly, out of the top-30 predicted classes, 0.85% of the times. This means that around three thousand images of the original dataset are invalid. In an ideal scenario, it is best to have a validation dataset that is manually validated but considering the number of classes and viewpoint variations that would need to be validated we chose to rely on the test set accuracy (which is still a reliable one for image data).

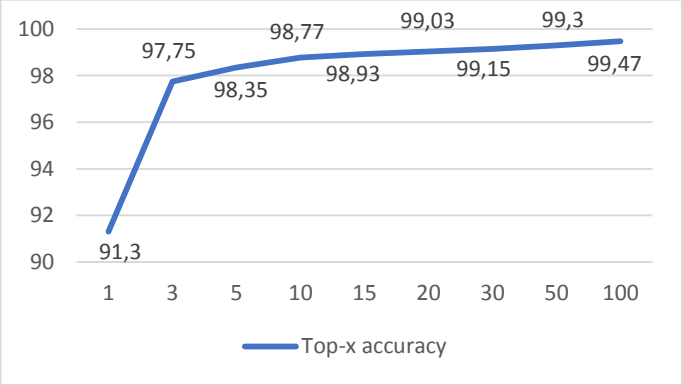


Figure 35 – Top-x accuracy of InceptionResNetV2 on Oto-790

Furthermore, by asking the question: how many incorrect predictions are there in the test dataset where the model is at least 99% sure it is a certain output label when it is not, the answer is 2805.

We outline some examples:

Figure 36 – Incorrect guesses with high certainty

As we can see, a minor portion of the dataset is mislabeled. Other examples exist where the model is 99% certain that it belongs to a certain make, model and version but it belongs to the same make and model but to a different version. The percentage difference between top-1 accuracy and top-3 and top-5 accuracy is where the model can be improved. For 6-7% of the test set the model is misclassifying mostly at the model/version level.

Fig. 37 shows 12 random examples of wrong predictions of top-5 accuracy. We see misclassifications at the model/version level and at the make level, given the similarity between certain models of different makes.

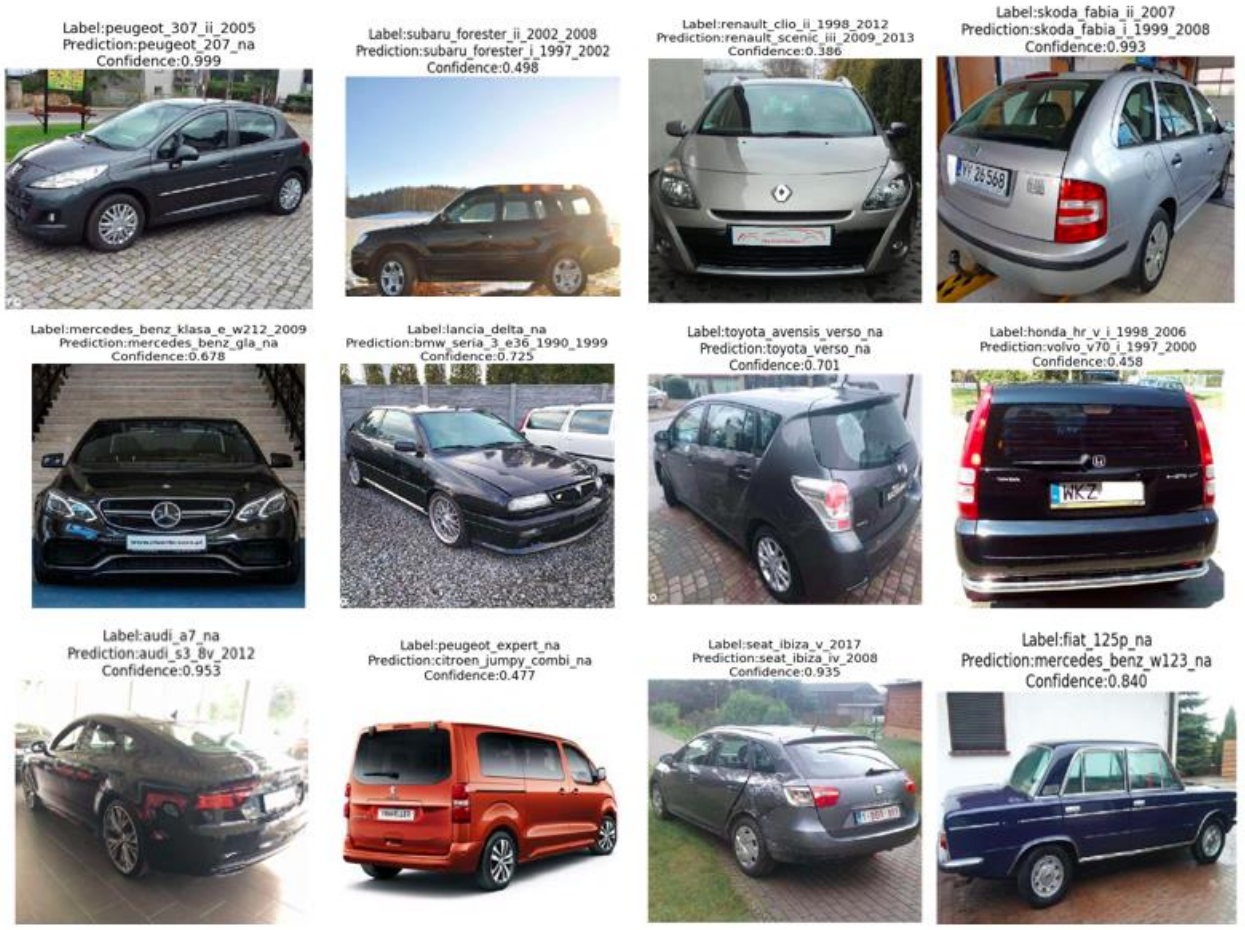


Figure 37 - Examples of wrong predictions

Besides the mislabelled examples of Fig. 36, presented earlier, in Fig. 37 we can see other two use cases of the model's wrong predictions:

- The model is very certain about wrong guesses, such as misclassifying very similar looking cars such as the Peugeot 307 for the 207 with 99.9% certainty and the Skoda Fabia II for the Skoda Fabia I with 99.3% certainty.
- The model is not so certain about wrong guesses such as the Peugeot Expert for the Citroen Jumpy Combi with 47.7% certainty and Honda HR V with the Volvo V70 with 45.8%.

The same learning rate and data augmentation techniques, used for fine-tuning InceptionResNet on the Oto-790, are used for fine-tuning VGGnet. Fig. 38, below, shows the accuracy evolution over epochs. Each epoch takes 14 hours and the whole training process takes 6.5 days. 2/3 of the network is kept frozen from the start of the transfer learning process.

Even though only one third of the network is being fine-tuned a high test accuracy is obtained. VGGnet fine-tuning achieves 89% top-1, 96.8% top-3 and 97.7% top-5 accuracy.

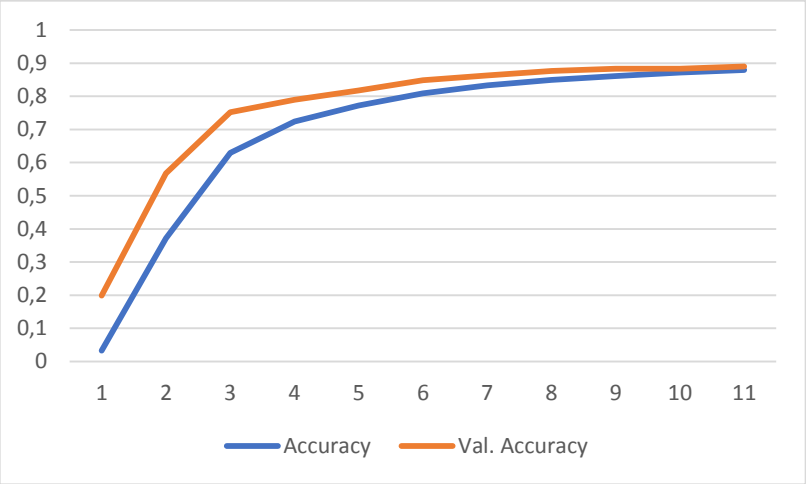


Figure 38 - VGGnet Accuracy/Epochs (make, model and version recognition)

Fig. 38 displays the robustness of the training process carried on the Oto-790 with the use of state-of-the-art CNN architectures such as VGGnet. We can see that the network is not overfitting and it is continuously getting better after each epoch. This differs from what we observed on the Stanford Cars dataset where the gap between train and test accuracy was more evident.

The fact that validation accuracy is higher than training accuracy is because we are applying data augmentation on training data but not on validation data. This will make the model more robust. It also does not make sense to apply data augmentation on validation data since that is the ground truth and not augmented images of this set. However, applying data augmentation on training data makes it harder for the model to output a higher training accuracy and that is why we see a higher validation accuracy than a training accuracy. Even though the data augmentation techniques are the same as those applied in the Stanford Cars dataset eventually in future work we will need to experiment different values for the data augmentation techniques applied. The hypothesis is that, maybe, we are being too aggressive with the values of the data augmentation techniques applied on an already robust dataset.

Following the same analysis performed with the Stanford Cars dataset we proceed by visualizing how the network is performing in specific examples. Fig. 39 shows two very fine-grained examples of classes with high accuracy and f1-score, the Mercedes Benz Class A W168 and W169 and the Peugeot 206 and 206 CC. We can see that the network is focusing on the part

that most distinguishes the two classes, the area around the front light in the case of the Mercedes and the entire side of the car in the case of the Peugeot. To take into consideration that in the case of the Peugeot part of the most distinguishing factor is also the window and the back of the car, which the attention map, in fact, shows.

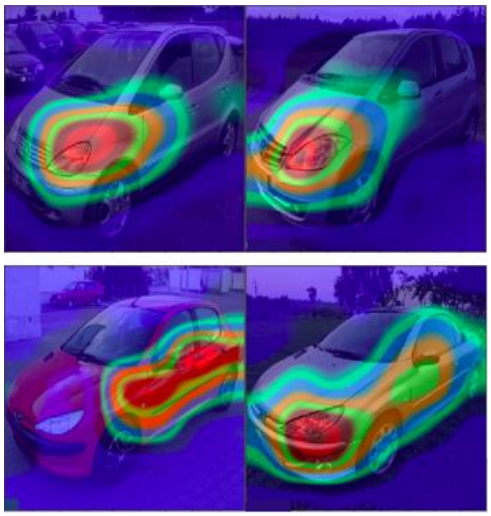


Figure 39 - Fine-grained classes saliency map comparison – Mercedes Benz Class A and Peugeot 206

4.2 Oto-790 2-Model Ensemble

Optimally model ensembles are performed on 3 or more models. In this section, following the same strategy applied on the Stanford Cars dataset, a model ensemble is performed on the two models trained on the Oto-790 dataset, InceptionResNetV2 and VGGnet.

InceptionResNetV2 reached an accuracy of 91.3% while VGGnet of 89%. By averaging the predictions of both models the test accuracy reaches a value of 92.7%: a 1.4 points increase.

Appendix A shows the model ensemble's accuracy for all the 790 classes.

Table 10 shows the classes that are most affected by fine-grained models and where the second guess has the highest probability.

Table 10 - Oto-790 model ensemble most uncertain classes

First guess	%	Second guess	%
Chrysler Grand Voyager	0.507	**Chrysler Voyager Iii**	0.43
Mercedes Benz W124	0.525	**Mercedes Benz Klasa E W124**	0.407
Chevrolet Kalos	0.576	**Daewoo Kalos**	0.371
Seat Altea	0.603	**Seat Toledo Iii**	0.371
Fiat Brava	0.574	**Fiat Bravo I**	0.366
Ford C Max I	0.604	**Ford Grand C**	0.362
Renault Scenic Ii	0.639	**Renault Grand Scenic**	0.36
Ford Galaxy Mk2	0.589	**Ford Galaxy Mk1**	0.344
Toyota Prius Iii	0.624	**Toyota Prius Iv**	0.327
Renault Trafic Ii	0.596	**Nissan Primastar**	0.323
Seat Cordoba I	0.551	**Seat Ibiza Ii**	0.317
Seat Toledo Ii	0.655	**Seat Leon I**	0.31
Audi S8 D4	0.654	**Audi A8 D4**	0.305
Volkswagen Multivan	0.279	**Volkswagen Transporter T6**	0.271
Volkswagen New Beetle	0.716	**Volkswagen Beetle**	0.269
Volkswagen Beetle	0.707	**Volkswagen New Beetle**	0.268
Volkswagen Polo V	0.71	**Volkswagen Polo Vi**	0.251
Mini Cooper	0.599	**Mini One**	0.249
Ford Tourneo Custom	0.757	**Ford Transit Custom**	0.232
Fiat Scudo	0.569	**Peugeot Expert**	0.226
Fiat Grande Punto	0.457	**Fiat Punto Ii**	0.222
Volkswagen Transporter T6	0.643	**Volkswagen Transporter T5**	0.222
Renault Espace Iv	0.773	**Renault Grand Espace**	0.219
Fiat Punto 2012	0.737	**Fiat Punto Evo**	0.205
Volkswagen Passat Cc	0.768	**Volkswagen Cc**	0.2

4.3 How much data is enough data?

Following the experiments carried out throughout the dissertation, first with a medium-scale dataset (Stanford Cars) and later with a large-scale dataset (Oto-790) we want to evaluate the impact of dataset size on performance. Such experiment is partially done in (Tafazzoli et al., 2017) but dataset unbalance leads to inconclusive results.

In this section, we consider Oto-790 dataset to carry two tests: evaluate train-test accuracy on 10% and 25% of the entire dataset, maintaining the same hyperparameters and 70-30 split.

We use InceptionResNetV2 since it is the one that had the highest test accuracy on the Oto-790 dataset. Figs. 40 and 41 show the evolution of train and test accuracy for the dataset consisting of 10% and 25% of the initial Oto-790 dataset, respectively.

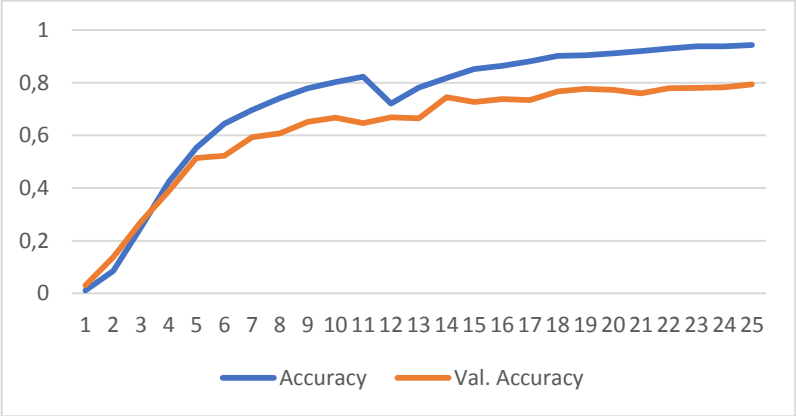


Figure 40 - InceptionResNetV2 fine-tuned on 10% of Oto-790

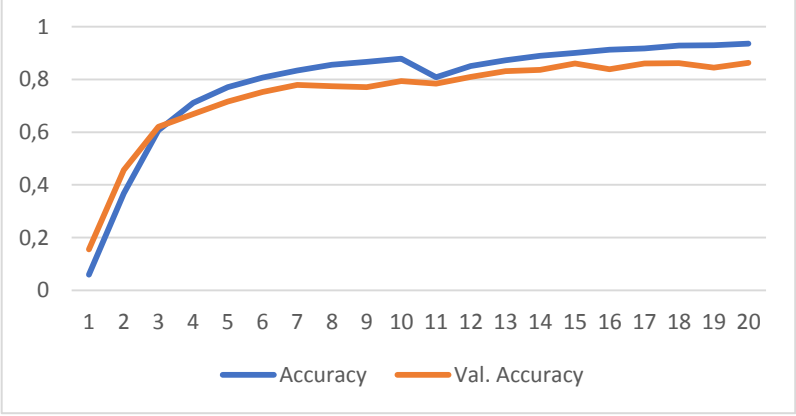


Figure 41 - InceptionResNetV2 fine-tuned on 25% of Oto-790

For a dataset consisting of 120 thousand images (10% of Oto-790) each epoch takes 1h20m and a top-1 test accuracy of almost 80% is achieved. For a dataset consisting of 300 thousand images (25% of Oto-790) each epoch takes 3 hours and the accuracy increases to 86%.

This validates the hypothesis that to get state-of-the-art results in cars' fine-grained classification a large-scale dataset is necessary (80% to 91.3% accuracy when going from 120 thousand to 1.2 million images).

Chapter 5

Car Color and Vehicle Category Recognition

We further test the feasibility of using CNNs in two fine-grained and distinct tasks: car color and vehicle category recognition. Besides identifying the make, model and version it is also useful to classify the color of a car and the category of a vehicle considering that, in classifieds websites, the color and category are also mandatory fields that could be filled automatically. In the case of color classification of cars, we want to access how CNNs work at identifying color patterns, considering that CNNs make representations for each of the RGB channels.

5.1 Car Color Recognition

For the color classification task, we collect a maximum of three thousand examples for each of the 13 existing colors in the car classifieds website: black, blue, brown, brown-beige, dark-red, green, grey, red, silver, violet, white, yellow and yellow-gold. These will be the classes the model will try to predict. Some of the colors are similar such as dark-red and violet, brown and brown-beige and grey and white.

Figure 42 - Black, Blue, Brown, Brown-Beige, Dark-Red, Green, Grey, Red, Violet, Silver, White, Yellow-Gold, Yellow dataset examples

Fine-tuning a VGGnet architecture in this task allows for a 72% test accuracy. We consider this a good value considering the fine-grained aspect of some colors and the variability in illumination conditions and positioning that effectively affect performance in color recognition.

In the confusion matrix portrayed in Fig. 43 we can see some similar looking classes: yellow-gold being predicted as brown-beige; brown as brown-beige; violet as black, blue, brown and grey; red as dark-red, and yellow as yellow-gold. To observe that it misclassifies more often red as dark-red than dark-red as red.

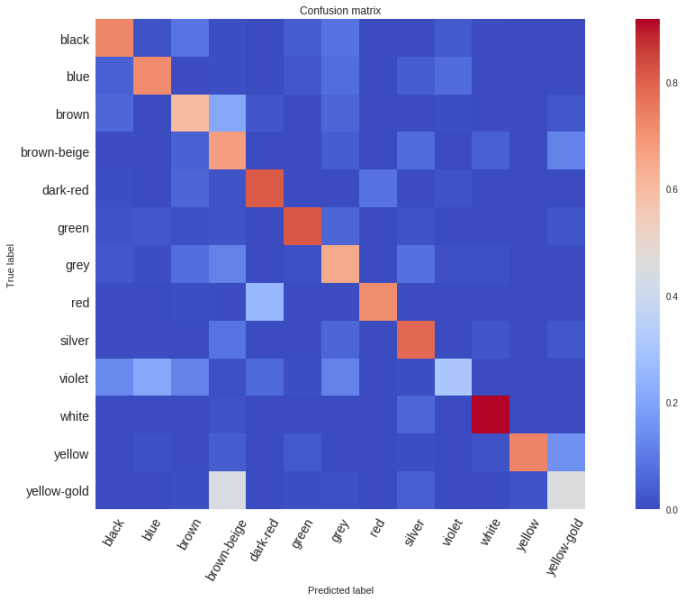


Figure 43 - 13-color Confusion Matrix

To understand how strong are the activations for each output class we look at the activation maximization for each one (Fig. 44). We can see that the network is activating strongly for some of the classes such as blue, green, red, white and yellow. These are also the ones that show the best precision (Table 11).

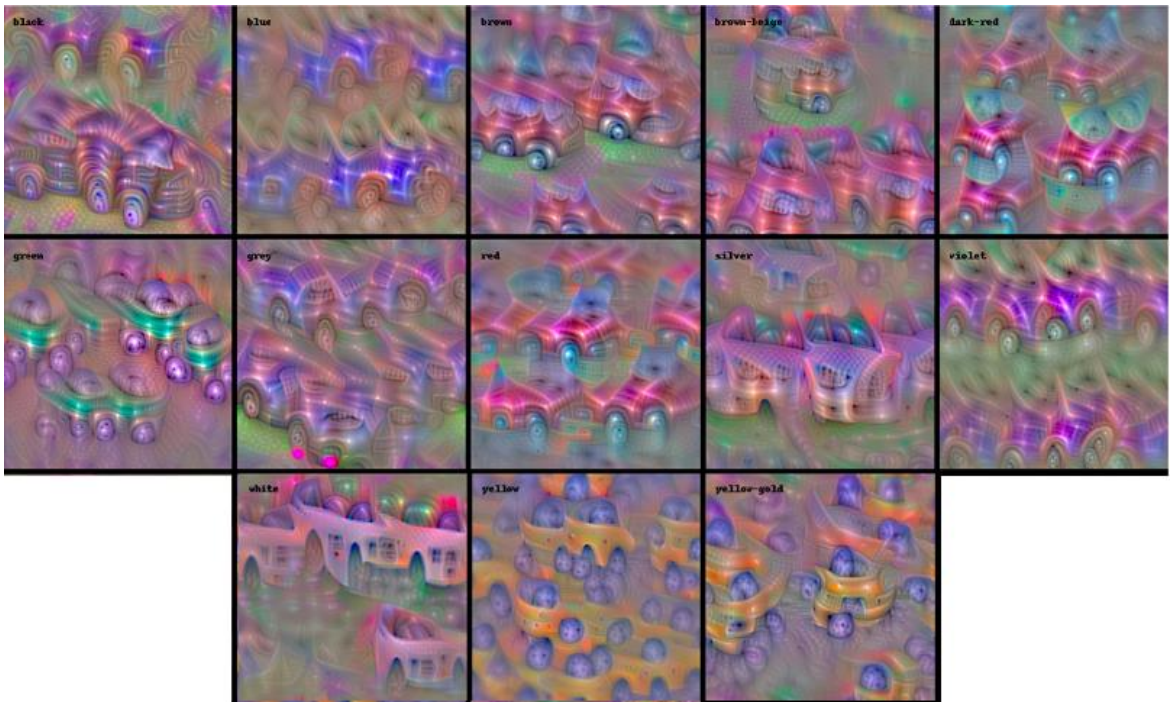


Figure 44 - Black, Blue, Brown, Brown-beige, Dark-red, Green, Grey, Red, Silver, Violet, White, Yellow, Yellow-gold

Table 11 - 13-color Classes Classification Report

	Precision	**Recall**	**F1-score**
Black	0.76	0.73	0.74
Blue	0.83	0.72	0.77
Brown	0.64	0.60	0.62
Brown-beige	0.41	0.67	0.51
Dark-red	0.72	0.81	0.77
Green	0.88	0.81	0.84
Grey	0.60	0.65	0.62
Red	0.87	0.71	0.79
Silver	0.71	0.79	0.75
Violet	0.43	0.31	0.36
White	0.89	0.92	0.90
Yellow	0.92	0.73	0.81
Yellow-gold	0.64	0.46	0.53
Avg/Total	**0.72**	**0.70**	**0.70**

5.1.1 8-Color Classes

For validating our results with the work of Zhang, Li, Zhuo, Zhang, & Li (2017) we remove 5 classes: brown-beige, dark-red, silver, violet and yellow-gold. The only class that differs from the ones of Zhang et al. (2017) is brown which in their work is cyan.

The strategy is the same: fine-tune a VGGnet model for the same number of epochs: 5. A considerate increase in accuracy is achieved. Test accuracy reaches the value of 85% after 5 epochs.

Table 12 shows the classification report. The worst performing classes are black, brown and grey. These classes share some similarity and are easily misclassified in different illumination conditions.

Table 12 - 8-color Classification Report

	Precision	**Recall**	**F1-score**
Black	0.79	0.82	0.80
Blue	0.86	0.82	0.84
Brown	0.74	0.80	0.77
Green	0.91	0.82	0.86
Grey	0.73	0.73	0.73
Red	0.96	0.97	0.97
White	0.96	0.97	0.96
Yellow	0.92	0.93	0.92
Avg/Total	**0.85**	**0.85**	**0.85**

By looking at the confusion matrix displayed below we see that there is considerate confusion between black, blue, brown, green and grey colored cars. On the other hand, the network shows high precision with red, white and yellow colored cars (Fig. 45).

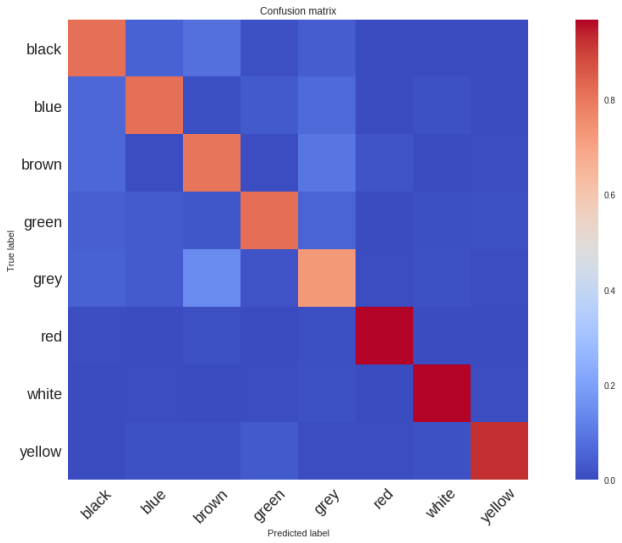


Figure 45 - 8-color Confusion Matrix

The results obtained in the 8-color model are better than the ones obtained in the 13-color one since we are considering fewer classes (85% accuracy with 8 color classes and 72% with 13). With fewer classes there will also be fewer similar looking cars.

Our work shows worse results than Zhang et al. (2017) 94.27% accuracy but we make no assumptions in vehicle viewpoint. In the work of Zhang et al. (2017) images consist of only the car's front view taken in roadways.

Some examples of wrong predictions are outlined below:

Figure 46 - Examples of wrong color prediction

We can see that illumination is the most important factor in color recognition as even the human eye faces a hard time recognizing what is the color portrayed in some of the examples.

5.2 Vehicle Category Classification

One simpler but useful task is vehicle category classification. We consider the 8 categories of the classifieds website: trucks, cars, commercials, parts, agriculture, trailers, construction and motorbikes. For each category, we collect 2.3 thousand examples and perform a 75-25 split, totalling 18733 training examples and 6245 test examples.

Figure 47 - Example of each category

We fine-tune a VGGnet architecture for the task of category classification. We fine-tune the network for five epochs considering this is a less fine-grained task. A test accuracy of 94% is achieved.

Table 13 shows the classification report. Agriculture is the lowest performing class followed by construction. Some examples categorized as agriculture are erroneously classified as construction and vice-versa (Fig. 48). These two classes include a wide range of heavy machinery vehicles in different shapes and sizes.

Table 13 - Category Classification Report

	Precision	**Recall**	**F1-score**
Agriculture	0.86	0.91	0.89
Cars	0.96	0.97	0.96
Commercial	0.90	0.94	0.92
Construction	0.90	0.87	0.89
Motorbikes	0.99	0.98	0.99
Parts	1.00	1.00	1.00
Trailers	0.92	0.93	0.93
Trucks	0.96	0.88	0.91
Avg/Total	0.94	0.94	0.94

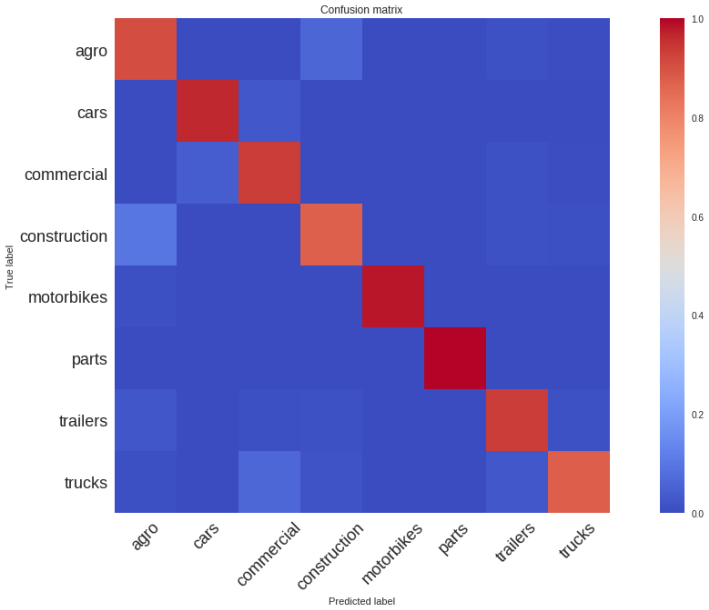


Figure 48 - Category Confusion Matrix

Figs. 49 and 50, below, show some examples of high probability of wrong guesses. Some of the examples are hard to predict, such as the wheel of an agriculture vehicle, which is predicted as a car (1st image). Two examples of cars are predicted as commercial (3rd and 4th image). This is the class that the model misclassifies a car for (confusion matrix above). The 5th image of a quadbike is predicted as an agriculture machine. Given the automated process of data collection we cannot guarantee that there are enough examples of quad-bikes in the dataset. The last image shows an abstract looking vehicle that is a trailer but is classified as an agriculture machine.

Figure 49 - Left - Label: Agro, Predicted: Cars; Middle – Label: Construction, Predicted: Agro; Right – Label: Commercial, Predicted: Cars

Figure 50 - Left - Label: Cars, Predicted: Commercial; Center – Label: Motorbikes, Predicted: Agro; Right – Label: Trailers, Predicted: Agro

Fig. 51 shows the input images that maximize each label. It is interesting to see how each label can be recognized by looking at what the network considers the optimal input. Agriculture and Construction labels share some similarity and a crane-like structure is visible. The front of the Truck is the most distinguishing and determining factor (bottom right image). Cars (2nd) and Motorbikes (6th) show that the network is focusing on the top structure, wheels and front of the car and the wheels and chassis of the motorbike.

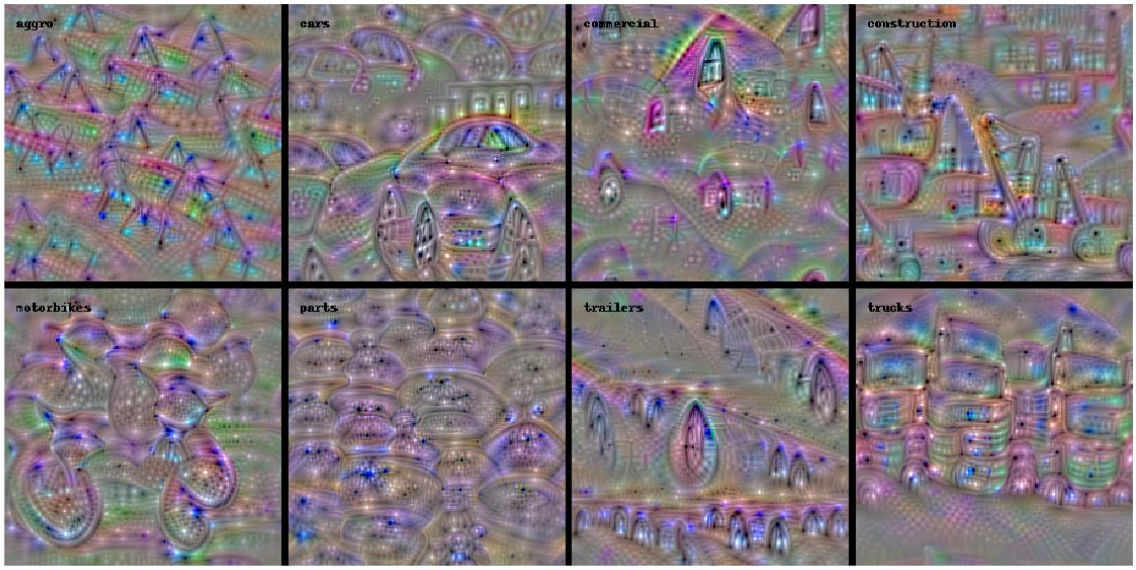


Figure 51 - Agriculture, Cars, Commercial, Construction, Motorbikes, Parts, Trailers, Trucks Activation Maximization Inputs

5.3 Remarks

CNNs show their efficacy on a wide range of tasks. For the case where the objective is to predict a vehicle's category the network achieves high precision, with a test accuracy of 94% without dataset cleaning/validation. The network only shows more difficulty when distinguishing commercials from cars and agriculture from construction machines. We believe the accuracy could be higher with a dataset of increase size and dataset cleaning and validation by removing false positives. For each class the network is being activated by robust patterns that closely represent the class label.

For the task of car's color recognition CNNs show difficulties when dealing with very similar colors such as black, grey and brown that are affected by illumination conditions. Even for the human eye some examples are not clear of what is the color of the car.

Chapter 6

Conclusions and Future Steps

CNNs are a breakthrough in computer vision. Their applicability to different domains make them pervasive in today's world. The biggest advantage of CNNs when compared with previous approaches to computer vision is that the CNNs learn, themselves, the features that allow them to distinguish and classify different objects. Furthermore, little to no knowledge is needed about the domain problem to be solved. Even though the average human performs badly at recognizing models of cars, we are able to build state-of-the-art models that do so with high precision, with the use of a large-scale dataset.

In the current dissertation, we began by using Stanford Cars dataset as a benchmark dataset for experimenting different CNN architectures and approaches. We achieved a 91% top-1 accuracy when comparing with 74% of Krause et al. (2014), 80% of D. Liu & Wang (2016) and 92.8% of Krause et al. (2015). No heavy hand-engineered process is carried. Applying multiple data augmentation techniques and removing most of the background in the images allows the network to overfit less to the training data, achieve better performance (8.3% points higher when comparing with the work of D. Liu & Wang (2016)) and generalize better to unseen data.

Afterwards, we collected a dataset composed of 1.47 million car images. We first trained a model to predict an image as valid or invalid for the final task of VMMR. This model was applied on the entire 1.47 million images to remove a large number of invalid images. Most of the background of valid images was removed as it does not contain useful information and leads to improved results. We then fine-tuned state-of-the-art CNN architectures. Transfer learning shows

itself as the *de facto* approach for training such CNNs and achieve state-of-the-art results. ImageNet also shows itself as the *de facto* dataset for pre-training such networks since general patterns are learnt (from ImageNet's 1000, very distinct classes). Pre-initializing a network's weights with ImageNet ones is a common approach in deep learning for image classification. It is then followed by a fine-tuning process.

With the Oto-790 dataset state-of-the-art results are achieved, with a top-1 accuracy of 92.7% on an ensemble method. This accuracy is the baseline but we believe it could be even higher with additional ground truth validation. However, given the number of images being considered (1.47 million) this process is not feasible. Still, the results are state-of-the-art for a large-scale dataset, considering the number of images and the number of classes to predict. It is important to note that, apart from the information provided by customers and the manual labelling of a small number of images (2000) to find invalid images, the dataset was not subject to any additional labelling processes. A more exhaustive labelling process could lead to a further improvement on the results. Furthermore, dataset size can grow substantially if no restrictions are made in the number of images to download and if we include car images from other classified websites.

Model visualization has shown to be useful in order to get insights on how the model is behaving even at the very fine-grained level. We observed that the most important parts of the car used for VMMR, given an image of a car's front view, is the front light and the grille.

We extended the work to validate the efficacy of CNNs in the tasks of vehicle category classification and car's color recognition. In the former we get a test accuracy of 94% and in the latter a test accuracy of 85% (for the 8-color's classification task).

Given the democratization of computing power, the models presented in this dissertation can be used in a centralized way by "living" in a web server responding to API requests or, in a decentralized way, distributed with the mobile application for on-device inference. Techniques such as quantization could be used to substantially reduce the model size.

A valid use case for these models is automation of content moderation and validation. Content moderation is still a manual work in car classifieds websites where the support team is validating new adverts being posted, if the images are appropriate, match the description of the car and if the car is in the right category (whether it is a car or a commercial one). By having this process automated with the make, model and version inference, the invalid images could be flagged automatically. As we saw in earlier sections of the dissertation the Oto-790 dataset contained a

minor portion of mislabelled images. With the VMMR models built in the scope of this dissertation, a semi-supervised process could be made in order to remove these mislabelled images from the dataset.

The CNNs trained on the Oto-790 dataset have enormous value on their own, given they perform better than most humans in the task of make, model and version recognition. Furthermore, they have real-world applicability by having as classes the make, model and versions of a classifieds website. These models can be applied to automatically predict a car make, model and version from an image and partially automate the process of inserting an advert in these platforms.

Given that each site has its own set of make, model and versions, in future work the aim is to extend the work carried and train a model per site, expand to more classes and use more images. This process is simplified given the automated process for collecting and cleaning the dataset. The dataset built had 1.2 million images over 790 classes but can be scaled to be composed of more images and classes.

An important suggestion for future work is to explore hyperparameter optimization to automatically find the best hyperparameters for a given CNN architecture. A significant portion of the work relied on experimenting different hyperparameters, some of the times with no success.

The classification models developed in the scope of this dissertation were applied to recognize the vehicle's category and the car's color, make, model and version. However, they could be extended to other use cases such as determining the car's orientation and the body type. A model that predicts the car's orientation can then be used to validate that a dataset has, for each class, examples in different viewpoints. It can also be used to determine in which car's orientation the model is performing best/worst.

Each advert of a car classified website has images of different parts of the car (front, side and back facing, interior, wheels). A classifier can also be built to classify what is represented in the image not at the make, model and version level, but at a more general level such as classifying an image as a wheel, car interior, etc. This could then be analysed to see if people that are looking for cars are more likely to contact a seller if the advert has images of certain parts of a car.

Appendices

Appendix A

790 classes model ensemble accuracy by class

bmw seria 1 f20	1.0	honda hr v i	0.99	renault megane iv	0.99
chevrolet orlando	1.0	honda hr v ii	0.99	renault talisman	0.99
chrysler pt cruiser	1.0	hyundai atos	0.99	renault vel satis	0.99
citroen ds4	1.0	hyundai getz	0.99	rover 75	0.99
fiat multipla	1.0	hyundai i10 ii	0.99	seat ateca	0.99
ford tourneo connect ii	1.0	hyundai ix20	0.99	skoda karoq	0.99
honda accord viii	1.0	hyundai ix35	0.99	skoda yeti	0.99
hyundai terracan	1.0	hyundai santa fe i	0.99	ssangyong rexton	0.99
infiniti fx ii	1.0	hyundai santa fe ii	0.99	subaru forester iv	0.99
jeep renegade	1.0	hyundai tucson ii	0.99	suzuki alto	0.99
land rover discovery	1.0	jeep cherokee iii	0.99	suzuki ignis	0.99
lexus gs iv	1.0	jeep cherokee v	0.99	suzuki jimny	0.99
mazda cx 7	1.0	jeep patriot	0.99	suzuki vitara ii	0.99
mercedes benz vaneo	1.0	kia carens ii	0.99	toyota auris ii	0.99
peugeot 1007	1.0	kia cerato	0.99	toyota aygo i	0.99
peugeot 301	1.0	kia rio ii	0.99	toyota aygo ii	0.99
renault Kadjar	1.0	kia soul i	0.99	toyota corolla seria e15	0.99
smart forfour i	1.0	kia sportage ii	0.99	toyota land cruiser iv	0.99
subaru forester ii	1.0	kia sportage iv	0.99	toyota yaris verso	0.99
suzuki celerio	1.0	kia venga	0.99	volkswagen amarok	0.99
toyota avensis verso	1.0	land rover range rover evoque	0.99	volkswagen fox	0.99
toyota rav4 iv	1.0	lexus gs iii	0.99	volkswagen phaeton	0.99
volvo c30	1.0	lexus ls iv	0.99	volkswagen scirocco	0.99
volvo xc 70	1.0	lexus nx	0.99	volkswagen touran iii	0.99
alfa romeo 156	0.99	lexus rx ii	0.99	volvo v70 iii	0.99
alfa romeo stelvio	0.99	mazda 2 i	0.99	volvo xc 90 ii	0.99
audi a2	0.99	mazda premacy	0.99	alfa romeo 147	0.98
audi a4 allroad	0.99	mazda rx 8	0.99	alfa romeo giulia	0.98
audi q3	0.99	mazda tribute	0.99	alfa romeo gt	0.98
audi tt 8j	0.99	mercedes benz klasa b w246	0.99	alfa romeo mito	0.98
bmw seria 2	0.99	mercedes benz klasa s w220	0.99	audi a8 d2	0.98
bmw seria 6 e63 e64	0.99	mitsubishi outlander iii	0.99	audi q5 fy	0.98
bmw seria 7 f01	0.99	nissan almera n15	0.99	audi q7 i	0.98
bmw x1 e84	0.99	nissan murano z51	0.99	bmw seria 1 e87	0.98
bmw x3 f25	0.99	nissan note i	0.99	bmw seria 7 e65	0.98
bmw x5 e70	0.99	nissan note ii	0.99	bmw x3 e83	0.98
chevrolet captiva	0.99	nissan pulsar	0.99	bmw x5 e53	0.98
chevrolet cruze	0.99	nissan x trail ii	0.99	bmw x6 e71	0.98
chevrolet epica	0.99	opel adam	0.99	bmw z4	0.98
citroen c6	0.99	opel agila a	0.99	chrysler 300c	0.98
citroen c elysee	0.99	opel combo c	0.99	citroen c2	0.98
dacia duster	0.99	opel crossland x	0.99	citroen c3 iii	0.98
dodge caliber	0.99	opel meriva ii	0.99	citroen c3 picasso	0.98
fiat 500l	0.99	opel mokka	0.99	citroen c3 pluriel	0.98
fiat idea	0.99	opel tigra i	0.99	citroen c4 cactus	0.98
fiat linea	0.99	peugeot 2008	0.99	citroen c4 ii	0.98
fiat panda iii	0.99	peugeot 306	0.99	citroen ds5	0.98
ford fusion	0.99	peugeot 607	0.99	fiat 500x	0.98
ford ka i	0.99	peugeot 807	0.99	fiat cinquecento	0.98
ford puma	0.99	renault fluence	0.99	fiat croma ii	0.98
honda fr v	0.99	renault laguna iii	0.99	fiat palio	0.98

fiat punto i	0.98	opel antara	0.98	ford mondeo mk3	0.97
fiat sedici	0.98	opel corsa b	0.98	ford mondeo mk4	0.97
fiat stilo	0.98	opel corsa e	0.98	ford s max i	0.97
fiat tipo ii	0.98	opel frontera b	0.98	honda accord vii	0.97
fiat ulysse	0.98	opel grandland x	0.98	honda cr v iii	0.97
fiat uno ii	0.98	opel insignia a	0.98	honda cr v iv	0.97
ford b max	0.98	opel signum	0.98	hyundai accent	0.97
ford ecosport	0.98	opel tigra ii 2004 2009	0.98	hyundai coupe	0.97
ford ka ii	0.98	peugeot 107	0.98	hyundai i30 ii	0.97
ford kuga i	0.98	peugeot 208	0.98	hyundai sonata	0.97
ford kuga ii	0.98	peugeot 308 t8	0.98	hyundai tucson i	0.97
ford mondeo mk2	0.98	porsche panamera	0.98	jaguar x type	0.97
ford tourneo connect I	0.98	renault captur	0.98	jeep grand cherokee iii	0.97
honda cr v ii	0.98	renault koleos	0.98	jeep grand cherokee iv	0.97
honda jazz ii	0.98	renault twingo ii	0.98	kia carens iii	0.97
hyundai i10 i	0.98	saab 9 3 i	0.98	kia optima	0.97
hyundai i40	0.98	skoda kodiaq	0.98	kia sorento i	0.97
hyundai matrix	0.98	suzuki swift v	0.98	kia sportage i	0.97
jaguar s type	0.98	toyota avensis i	0.98	lancia lybra	0.97
jaguar xe	0.98	toyota c hr	0.98	land rover discovery ii	0.97
jaguar xf x250	0.98	toyota rav4 i	0.98	land rover freelander i	0.97
jeep compass	0.98	toyota sienna	0.98	lexus rx iii	0.97
kia carens iv	0.98	toyota yaris i	0.98	mazda 5 ii	0.97
kia carnival	0.98	volkswagen arteon	0.98	mazda 626 v	0.97
kia picanto ii	0.98	volkswagen eos	0.98	mazda 6 ii	0.97
kia rio iii	0.98	volkswagen golf sportsvan	0.98	mazda cx 3	0.97
kia sportage iii	0.98	volkswagen golf vii	0.98	mazda cx 5	0.97
lancia delta	0.98	volkswagen passat b7	0.98	mazda mpv	0.97
lancia ypsilon	0.98	volkswagen t roc	0.98	mercedes benz cla	0.97
land rover discovery iii	0.98	volkswagen tiguan i	0.98	mercedes benz clk w208	0.97
lexus is i	0.98	volkswagen touareg ii	0.98	mercedes benz klasa a w169	0.97
lexus is ii	0.98	volvo s80 ii	0.98	mercedes benz klasa c w204	0.97
mazda 2 ii	0.98	volvo v40 ii	0.98	mercedes benz sl r230	0.97
mazda 3 ii	0.98	volvo xc 90 i	0.98	mercedes benz slk r170	0.97
mazda 6 iii	0.98	alfa romeo 159	0.97	mitsubishi colt cj0	0.97
mazda mx 5	0.98	alfa romeo 166	0.97	mitsubishi galant viii	0.97
mercedes benz glk glk 220	0.98	audi a6 allroad c6	0.97	mitsubishi lancer vii	0.97
mercedes benz klasa a w176	0.98	audi q5 8r	0.97	mitsubishi pajero iv	0.97
mercedes benz klasa	0.98	audi tt 8n	0.97	mitsubishi pajero pinin	0.97
mercedes benz sl r129	0.98	bmw seria 3 e90	0.97	mitsubishi space star i	0.97
mercedes benz sprinter iii	0.98	bmw seria 6 f12 13	0.97	nissan juke	0.97
mitsubishi grandis	0.98	chevrolet spark	0.97	nissan primera p12	0.97
mitsubishi outlander ii	0.98	citroen c1 ii	0.97	nissan x trail i	0.97
mitsubishi space star ii	0.98	citroen xsara i	0.97	opel corsa c	0.97
nissan almera tino	0.98	dacia lodgy	0.97	peugeot 106 ii	0.97
nissan micra k11	0.98	daewoo tico	0.97	peugeot 206 plus	0.97
nissan micra k13	0.98	daihatsu terios	0.97	peugeot 3008 i	0.97
nissan micra k14	0.98	fiat 500	0.97	peugeot 4007	0.97
nissan murano z50	0.98	fiat panda ii	0.97	peugeot 508	0.97
nissan primera p11	0.98	ford escort mk7	0.97	peugeot rcz	0.97
opel agila b	0.98	ford fiesta mk8	0.97	polonez caro	0.97

porsche 911 991	0.97	honda cr v i	0.96	toyota avensis iii	0.96
porsche cayenne i	0.97	honda jazz iii	0.96	toyota celica	0.96
porsche cayenne ii	0.97	honda prelude	0.96	toyota corolla seria e11	0.96
renault clio iv	0.97	hyundai elantra iv	0.96	toyota hilux	0.96
renault espace v	0.97	hyundai i20 i	0.96	toyota previa	0.96
renault modus	0.97	hyundai i30 i	0.96	toyota prius ii	0.96
renault twingo i	0.97	hyundai kona	0.96	toyota rav4 iii	0.96
renault twingo iii	0.97	kia ceed ii	0.96	toyota yaris ii	0.96
saab 9 5	0.97	kia rio i	0.96	volkswagen lupo	0.96
seat arona	0.97	kia sorento ii	0.96	volkswagen passat b3	0.96
seat exeo	0.97	kia sorento iii	0.96	volvo s60 i	0.96
skoda fabia i	0.97	land rover range rover sport	0.96	volvo s80 i	0.96
skoda superb i	0.97	lexus ct	0.96	volvo v50	0.96
suzuki grand vitara i	0.97	mazda 3 i	0.96	volvo xc 60 ii	0.96
suzuki grand vitara ii	0.97	mazda cx 9	0.96	audi s8 d4	0.95
suzuki liana	0.97	mercedes benz cl c216	0.96	bmw seria 4	0.95
suzuki splash	0.97	mercedes benz cls c219	0.96	bmw seria 7 e38	0.95
suzuki sx4 i	0.97	mercedes benz gl x164	0.96	bmw x1 f48	0.95
suzuki sx4 s	0.97	mercedes benz gla	0.96	bmw x6 f16	0.95
toyota auris i	0.97	mercedes benz glc	0.96	chevrolet camaro	0.95
toyota corolla seria e16	0.97	mercedes benz klasa b w245	0.96	chrysler pacifica	0.95
toyota yaris iii	0.97	mercedes benz klasa e w211	0.96	chrysler sebring	0.95
volkswagen golf plus	0.97	mercedes benz klasa e w212	0.96	citroen c5 i	0.95
volkswagen jetta a6	0.97	mercedes benz klasa g w463	0.96	citroen c5 iii	0.95
volkswagen passat b4	0.97	mercedes benz ml w163	0.96	dacia dokker	0.95
volkswagen passat b6	0.97	mini countryman	0.96	daewoo lanos	0.95
volkswagen touareg i	0.97	mitsubishi asx	0.96	fiat 125p	0.95
volkswagen touran ii	0.97	mitsubishi l200	0.96	fiat albea	0.95
volvo c70	0.97	nissan micra k12	0.96	fiat freemont	0.95
volvo v90	0.97	nissan qashqai ii	0.96	ford fiesta mk6	0.95
alfa romeo giulietta	0.96	opel astra f	0.96	ford s max ii	0.95
audi a1	0.96	opel astra j	0.96	honda accord vi	0.95
audi a4 b8	0.96	opel corsa d	0.96	honda civic x	0.95
audi a6 allroad c5	0.96	opel vectra b	0.96	honda odyssey	0.95
audi a6 c4	0.96	opel zafira c	0.96	hyundai i30 iii	0.95
audi a8 d3	0.96	peugeot 207 cc	0.96	hyundai santa fe iii	0.95
audi q7 ii	0.96	peugeot 308 cc	0.96	infiniti q50	0.95
audi s3 8l	0.96	peugeot 406	0.96	jeep wrangler iii	0.95
bmw x3 g01	0.96	peugeot 5008 i	0.96	kia pro ceed	0.95
bmw x5 f15	0.96	renault laguna i	0.96	land rover range rover iv	0.95
citroen saxo	0.96	renault laguna ii	0.96	land rover range rover sport	0.95
dacia logan i	0.96	rover 25	0.96	mazda 323f ii	0.95
daewoo matiz	0.96	seat alhambra ii	0.96	mazda 3 iii	0.95
daihatsu cuore	0.96	seat arosa	0.96	mercedes benz citan	0.95
daihatsu sirion	0.96	seat leon iii	0.96	mercedes benz gl x166	0.95
fiat doblo i	0.96	skoda superb ii	0.96	mercedes benz klasa a w168	0.95
ford cougar	0.96	smart fortwo i	0.96	mercedes benz klasa c w202	0.95
ford focus mk3	0.96	subaru forester iii	0.96	mercedes benz klasa s w140	0.95
ford galaxy mk2	0.96	subaru impreza gh	0.96	mercedes benz slk r171	0.95
honda city	0.96	suzuki wagon r	0.96	mercedes benz vito w638	0.95
honda civic viii	0.96	toyota avensis ii	0.96	mitsubishi carisma ii	0.95

mitsubishi lancer viii	0.95	mazda 323f iii	0.94	skoda rapid	0.93
mitsubishi outlander i	0.95	mazda 6 i	0.94	subaru impreza gc	0.93
nissan x trail iii	0.95	mercedes benz cl c215	0.94	subaru outback iv	0.93
opel zafira a	0.95	mercedes benz gle	0.94	subaru xv	0.93
opel zafira b	0.95	mercedes benz klasa e w210	0.94	suzuki swift iv	0.93
peugeot 308 t7	0.95	mercedes benz ml w166	0.94	toyota camry	0.93
porsche macan	0.95	mercedes benz vito w639	0.94	toyota corolla verso	0.93
renault megane iii	0.95	mercedes benz w201	0.94	volkswagen passat b5	0.93
rover 45	0.95	mitsubishi colt z30	0.94	volkswagen polo iv	0.93
seat ibiza ii fl	0.95	opel frontera a	0.94	volkswagen tiguan ii	0.93
seat leon ii	0.95	opel omega b	0.94	volvo v70 i	0.93
seat mii	0.95	opel vivaro ii	0.94	aixam city	0.92
skoda fabia ii	0.95	peugeot 307 cc	0.94	audi 80 b4	0.92
skoda octavia ii	0.95	peugeot 307 ii	0.94	audi a6 c7	0.92
skoda octavia iii	0.95	saab 9 3 ii	0.94	audi q2	0.92
skoda superb iii	0.95	seat toledo iv	0.94	bmw seria 3 e30	0.92
subaru forester i	0.95	skoda citigo	0.94	citroen c3 i	0.92
subaru impreza gd	0.95	skoda felicia	0.94	citroen c crosser	0.92
subaru legacy iv	0.95	skoda roomster	0.94	fiat 126	0.92
subaru legacy v	0.95	suzuki baleno	0.94	ford focus mk1	0.92
suzuki samurai	0.95	volkswagen caddy ii	0.94	ford mondeo mk5	0.92
suzuki swift iii	0.95	volkswagen passat b8	0.94	ford transit v	0.92
suzuki vitara i	0.95	volkswagen polo iii	0.94	ford transit vi	0.92
toyota corolla seria e12	0.95	audi 80 b3	0.93	honda accord v	0.92
toyota land cruiser iii	0.95	audi a6 c6	0.93	honda civic ix	0.92
toyota land cruiser vi	0.95	audi a7	0.93	honda civic v	0.92
toyota prius iv	0.95	bmw x4	0.93	hyundai galloper	0.92
toyota rav4 ii	0.95	citroen c4 i	0.93	jeep grand cherokee ii	0.92
toyota verso	0.95	citroen c5 ii	0.93	land rover defender	0.92
volkswagen golf ii	0.95	fiat bravo i	0.93	land rover discovery iv	0.92
volkswagen golf vi	0.95	fiat bravo ii	0.93	land rover freelander ii	0.92
volkswagen up	0.95	fiat seicento	0.93	land rover range rover iii	0.92
volvo v60	0.95	ford fiesta mk7	0.93	mazda 5 i	0.92
volvo v70 ii	0.95	ford maverick	0.93	mercedes benz gls	0.92
audi a3 8l	0.94	kia ceed i	0.93	mercedes benz w123	0.92
audi a6 allroad c7	0.94	mazda 2 iii	0.93	mitsubishi pajero ii	0.92
bmw seria 5 e34	0.94	mercedes benz clk w209	0.93	nissan almera n16	0.92
bmw seria 7 g11 12	0.94	mercedes benz sprinter ii	0.93	opel combo d	0.92
chevrolet aveo	0.94	mitsubishi lancer evolution	0.93	opel insignia b	0.92
chevrolet matiz	0.94	mitsubishi pajero iii	0.93	volkswagen touran i	0.92
citroen c4 grand picasso ii	0.94	nissan patrol gr ii y61	0.93	volvo s40 ii	0.92
citroen ds3	0.94	opel meriva i	0.93	audi a4 b5	0.91
dacia sandero i	0.94	opel omega b fl	0.93	audi a4 b7	0.91
dodge durango	0.94	peugeot 206 cc	0.93	audi a4 b9	0.91
ford edge	0.94	peugeot 206	0.93	bmw seria 5 g30	0.91
ford explorer	0.94	peugeot 407	0.93	chrysler voyager ii	0.91
ford focus mk2	0.94	peugeot 5008 ii	0.93	citroen c3 ii	0.91
ford mustang	0.94	renault kangoo i	0.93	citroen c4 grand picasso i	0.91
honda civic vii	0.94	renault megane i	0.93	dodge journey	0.91
jaguar xj x351	0.94	renault megane ii	0.93	fiat punto ii fl	0.91
kia picanto i	0.94	renault trafic iii	0.93	fiat qubo	0.91

ford fiesta mk5	0.91	audi a3 8v	0.88	peugeot expert	0.82
ford ranger	0.91	audi a4 b6	0.88	renault grand scenic iii	0.82
jeep grand cherokee i	0.91	audi rs6	0.88	toyota prius iii	0.82
kia magentis	0.91	chevrolet lacetti	0.88	volkswagen cc	0.82
lexus is iii	0.91	citroen c1 i	0.88	volkswagen transporter t4	0.82
mercedes benz klasa c w203	0.91	citroen c4 picasso i	0.88	fiat punto evo	0.81
mercedes benz klasa c w205	0.91	dodge grand caravan	0.88	mercedes benz viano	0.81
mercedes benz klasa s w126	0.91	fiat ducato iii	0.88	nissan patrol gr i y60	0.81
mercedes benz klasa v ii	0.91	fiat siena	0.88	opel astra g	0.81
opel astra k	0.91	kia rio iv	0.88	renault scenic iii	0.81
peugeot 207	0.91	mercedes benz cls c218	0.88	seat ibiza iii	0.81
peugeot partner i	0.91	opel astra h	0.88	audi a5 f5	0.8
peugeot partner ii	0.91	seat alhambra i	0.88	bmw m5	0.8
renault espace iii	0.91	volkswagen passat b5 fl	0.88	nissan qashqai i	0.8
volkswagen golf iv	0.91	volkswagen sharan i	0.88	opel vectra c	0.8
volkswagen golf v	0.91	bmw m3	0.87	seat toledo iii	0.8
volvo xc 60 i	0.91	bmw seria 3 e36	0.87	chevrolet corvette	0.79
audi a3 8p	0.9	citroen xsara picasso	0.87	citroen berlingo ii	0.79
citroen c8	0.9	ford f150	0.87	peugeot 307 i	0.79
dacia logan ii	0.9	mini cooper s	0.87	volkswagen passat cc	0.79
fiat punto ii	0.9	nissan primastar	0.87	volkswagen tiguan allspace	0.79
ford fiesta mk4	0.9	renault thalia	0.87	chevrolet nubira	0.78
mercedes benz klasa e w213	0.9	volvo s90	0.87	seat cordoba i	0.78
mitsubishi carisma i	0.9	bmw seria 3 e46	0.86	audi 100 c4	0.77
nissan pathfinder	0.9	bmw seria 5 e39	0.86	ford galaxy mk1	0.77
nissan qashqai 2	0.9	dacia sandero ii	0.86	chevrolet kalos	0.76
peugeot 3008 ii	0.9	ford grand c	0.86	citroen jumpy combi	0.76
renault clio iii	0.9	mercedes benz sl r107	0.86	fiat brava	0.76
seat ibiza v	0.9	mini clubman	0.86	renault espace iv	0.76
subaru outback iii	0.9	renault kangoo iv	0.86	volkswagen polo v	0.76
volkswagen golf iii	0.9	audi a6 c5	0.85	daewoo kalos	0.75
volkswagen jetta a5	0.9	daewoo nubira	0.85	fiat ducato ii	0.75
volkswagen sharan ii	0.9	jeep cherokee ii	0.85	volkswagen transporter t5	0.75
audi a5 8t	0.89	nissan navara	0.85	renault grand scenic ii	0.74
bmw seria 3 f30	0.89	opel vivaro i	0.85	audi s5	0.73
bmw seria 5 e60	0.89	renault clio i	0.85	renault scenic iv	0.73
dacia sandero stepway	0.89	volkswagen caddy iv	0.85	volkswagen new beetle	0.73
dodge ram	0.89	volvo s40 i	0.85	audi s3 8v	0.72
fiat doblo ii	0.89	citroen berlingo i	0.84	audi s6 c5	0.72
honda civic vi	0.89	isuzu d max	0.84	fiat grande punto	0.72
hyundai i20 ii	0.89	renault clio ii	0.84	mercedes benz klasa e w124	0.72
nissan terrano ii	0.89	seat leon i	0.84	seat ibiza ii	0.72
renault kangoo ii	0.89	bmw seria 5 f10	0.83	seat toledo ii	0.72
renault scenic i	0.89	fiat fiorino	0.83	audi a8 d4	0.71
seat altea xl	0.89	seat cordoba ii	0.83	ford transit custom	0.71
seat ibiza iv	0.89	volkswagen polo vi	0.83	ford c max ii	0.7
skoda octavia i	0.89	volvo s60 ii	0.83	ford galaxy mk2	0.7
volkswagen bora	0.89	citroen c4 picasso ii	0.82	mercedes benz ml w164	0.7
volkswagen caddy iii	0.89	citroen xsara ii	0.82	volkswagen beetle	0.7
volkswagen crafter	0.89	mercedes benz klasa s w222	0.82	fiat scudo	0.69
volvo v40 i	0.89	mercedes benz vito w447	0.82	ford tourneo custom	0.69

peugeot boxer	0.69	chrysler town country ii	0.64	mini one	0.55
renault grand scenic iv	0.69	mercedes benz w124	0.63	volkswagen caravelle	0.55
chrysler voyager iii	0.68	ford c max i	0.61	chrysler grand voyager iii	0.52
peugeot bipper	0.68	citroen nemo	0.6	mini cooper	0.52
renault scenic ii	0.68	volkswagen multivan	0.6	volkswagen transporter t6	0.51
renault trafic ii	0.68	fiat punto	0.59	renault grand espace	0.49
chrysler grand voyager iv	0.66	seat altea	0.59	mercedes benz ml w164	0.43
seat cordoba i fl	0.66	ford focus c	0.56	mercedes benz klasa s w221	0.39
mercedes benz klasa s w221	0.65				

Bibliography

Binford, T. O. (1971). Visual Perception by Computer. Proceedings of the IEEE Conference on Systems and Control (Miami, FL)

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/BF00058655

Chollet, F. (2017). *Deep Learning with Python*. *KI - Künstliche Intelligenz*. https://doi.org/citeulike-article-id:10054678

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, *I*, 886–893. https://doi.org/10.1109/CVPR.2005.177

Dehghan, A., Masood, S. Z., Shu, G., & Ortiz, E. G. (2017). View Independent Vehicle Make, Model and Color Recognition Using Convolutional Neural Network, 1–7. Retrieved from http://arxiv.org/abs/1702.01721

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Li, F.-F. (2009). ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPRW.2009.5206848

Deng, J., Krause, J., & Fei-Fei, L. (2013). Fine-Grained Crowdsourcing for Fine-Grained Recognition.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. *Bernoulli*, (1341), 1–13. Retrieved from http://igva2012.wikispaces.asu.edu/file/view/Erhan+2009+Visualizing+higher+layer+features+of+a+deep+network.pdf

Felzenszwalb, P. F., Girshick, R. B., Mcallester, D., & Ramanan, D. (2009). Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1–20. https://doi.org/10.1109/TPAMI.2009.167

Fischler, M. A., & Elschlager, R. A. (1973). The Representation and Matching of Pictorial Structures Representation. *IEEE Transactions on Computers*, *C-22*(1), 67–92. https://doi.org/10.1109/T-C.1973.223602

Freund, Y., & Schapire, R. (1997). A decision theoretic generalisation of online learning. *Computer and System Sciences*, *55*(1), 119–139.

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, *2015 Inter*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. https://doi.org/10.1109/ICCV.2017.322

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (n.d.). Improving neural networks by preventing co-adaptation of feature detectors arXiv : 1207 . 0580v1 [cs . NE] 3 Jul 2012, 1–18.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. https://doi.org/arXiv:1704.04861

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., & Weinberger, K. Q. (2017). Snapshot Ensembles: Train 1, get M for free, 1–14. Retrieved from http://arxiv.org/abs/1704.00109

Kaggle (2010). Retrieved from https://www.kaggle.com

Kotikalapudi, Raghavendra (2017). Keras-vis. Retrieved from https://github.com/raghakot/keras-vis

Keypoints, S., & Lowe, D. G. (2004). Distinctive Image Features from. *International Journal of Computer Vision*, *60*(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Krause, J., Gebru, T., Deng, J., Li, L. J., & Li, F. F. (2014). Learning features and parts for fine-grained recognition. *Proceedings - International Conference on Pattern Recognition*, 26–33. https://doi.org/10.1109/ICPR.2014.15

Krause, J., Jin, H., Yang, J., & Li, F. F. (2015). Fine-grained recognition without part annotations. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07–12–June*, 5546–5555. https://doi.org/10.1109/CVPR.2015.7299194

Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3D object representations for fine-grained categorization. *Proceedings of the IEEE International Conference on Computer Vision*, 554–561. https://doi.org/10.1109/ICCVW.2013.77

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. *… Science Department, University of Toronto, Tech. …*, 1–60. https://doi.org/10.1.1.222.9220

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features : spatial pyramid matching for recognizing natural scene categories To cite this version : Beyond Bags of Features : Spatial Pyramid Matching for Recognizing Natural Scene Categories. https://doi.org/10.1109/CVPR.2006.68

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Lin, M., Chen, Q., & Yan, S. (2013). Network In Network, 1–10. https://doi.org/10.1109/ASRU.2015.7404828

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8693 LNCS*(PART 5), 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *Bit*, *16*(2), 146–160. https://doi.org/10.1007/BF01931367

Liu, D., & Wang, Y. (2016). Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning. Retrieved from http://cs231n.stanford.edu/reports/2015/pdfs/lediurfinal.pdf

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*(December), 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1150–1157 vol.2. https://doi.org/10.1109/ICCV.1999.790410

Luo, P., Loy, C. C., & Tang, X. (n.d.). A Large-Scale Car Dataset for Fine-Grained Categorization and Verification.

McCulloch, W. S., & Pitts, W. (1943). A Logical Calculus of the Idea Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133. https://doi.org/10.1007/BF02478259

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Roberts, L. Gi. (1965). Machine perception of three-dimensional solids. *PhD Thesis*, (November), 159–197. https://doi.org/http://hdl.handle.net/1721.1/11589

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in …. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Schapire, R. E. (2003). The boosting approach to machine learning: an overview. *Nonlinear Estimation and Classification*, *171*, 149–171. https://doi.org/10.1.1.24.5565

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *Proceedings of the IEEE International Conference on Computer Vision*, *2017–Octob*, 618–626. https://doi.org/10.1109/ICCV.2017.74

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. https://doi.org/10.1109/CVPR.2015.7299176

Shi, J., & Malik, J. (2005). Normalized Cuts and Image Segmentation Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *22*(March), 888–905. https://doi.org/10.1109/CVPR.1997.609407

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 1–8. https://doi.org/10.1007/s10909-016-1606-9

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, 1–14. https://doi.org/10.1016/j.infsof.2008.09.005

Sochor, J., Herout, A., & Havel, J. (2016). BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3006–3015. https://doi.org/10.1109/CVPR.2016.328

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for Simplicity: The All Convolutional Net, 1–14. https://doi.org/10.1163/_q3_SIM_00374

Szegedy, C. (2013). Deep Neural Networks for Object Detection. *Nips 2013*, 1–9. https://doi.org/10.1109/CVPR.2014.276

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. https://doi.org/10.1016/j.patrec.2014.01.008

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07–12–June*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Tafazzoli, F., Frigui, H., & Nishiyama, K. (2017). A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2017–July*, 874–881. https://doi.org/10.1109/CVPRW.2017.121

Uijlings, J. R. R., Sande, K. E. A. Van De, Gevers, T., & Smeulders, A. W. M. (2012). Selective Search for Object Recognition.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I-511-I-518. https://doi.org/10.1109/CVPR.2001.990517

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding Neural Networks Through Deep Visualization. Retrieved from http://arxiv.org/abs/1506.06579

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

Zhang, Q., Li, J., Zhuo, L., Zhang, H., & Li, X. (2017). Vehicle Color Recognition with Vehicle-Color Saliency Detection and Dual-Orientational Dimensionality Reduction of CNN Deep Features. *Sensing and Imaging*, *18*(1), 1–15. https://doi.org/10.1007/s11220-017-0173-8