

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# **Automatic vehicle damage detection with images**

**José Pedro Lobo Marinho Trocado Moreira**

WORKING VERSION



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Luís Filipe Pinto de Almeida Teixeira

May 2, 2017



# **Automatic vehicle damage detection with images**

**José Pedro Lobo Marinho Trocado Moreira**

Mestrado Integrado em Engenharia Informática e Computação

May 2, 2017



# Abstract

Visual image classification is a research area that involves both computer vision and machine learning. The task of visually classifying an object consists in assigning an object to a category, or set of categories the object belongs to.

Traditionally, visual classification tasks are performed using a two layered system, made up of a first layer featuring an out-of-the-shelf feature extractor and detector, and a second classifier layer. In most recent years, convolutional neural networks have been shown to outperform such previously used systems.

Cars have a paramount role in today's world, and being able to automatically classify damages in cars is of great interest specially to the car insurance industry. Car insurance companies deal with car inspections on a daily basis. Such inspections are a manual, lengthy and sometimes faulty processes. Processes that bring costs and inconveniences to costumers and insurance companies alike. Even though the total replacement of such manual inspection processes might still be far away, developing systems to aid, accelerate or enhance the process might be possible with today's technology.

There isn't, to my knowledge, much work developed in automatic visual car damage classification, and none of it employs these recent performance improvements in image classification made possible through the use of CNNs. This happens in spite of some recent research pointing at the fact that modern CNN technology does in fact, outperform traditional methods in non damage, car related image classification tasks.

I hope to successfully apply state-of-the-art Convolutional Neural Network technology to solve the problem of automatically identifying, distinguishing and locating damages in car images. I intend to develop a working prototype of a system that will be able to tell if a given photograph exhibits a car with damages or not, and possibly identifying, to some extent, the damaged areas within the car. The most promising CNN architectures will be used, taking in account both its classification accuracy as well as training and classification times.

In order to be able to develop such system, a suitable dataset was gathered. The dataset is very unbalanced in terms of the represented classes. Such imbalances have important effects that were corrected with suitable techniques to prevent a significant performance degradation. The dataset is used to both train and measure the performance of the system. Since no car damage datasets are freely available, the used dataset is composed of images gathered using search engines and car crash agencies galleries.



# Resumo

A classificação automática de imagens é uma área científica que se encontra na fronteira entre a Visão por Computador e *Machine Learning*. A classificação de imagens é um processo que em suma, consiste na atribuição de imagens a uma ou mais categorias.

Até muito recentemente, os sistemas utilizados para fazer a classificação automática de imagens, eram constituídos por duas camadas distintas. A primeira camada é tipicamente constituída por um *Feature Extractor* e *Feature Detector*, sendo que estes subsistemas são constituídos por complexos algoritmos já existentes. A segunda camada é usualmente constituída por um classificador. Nos últimos anos, as *convolutional neural networks* (CNNs) têm-se demonstrado capazes de obter melhores resultados face a estes sistemas mais tradicionais.

Os carros têm um papel muito importante no mundo contemporâneo e a classificação automática de danos é, também por isso, bastante relevante para a indústria seguradora, entre outras. As seguradoras automóveis têm que lidar frequentemente com inspeções a veículos danificados. Este processo é normalmente moroso e ineficiente, com inconvenientes e custos quer para a empresa, quer para os segurados. Embora a automatização total destes processos possa ainda não ser possível, a utilização de sistemas de classificação automática de danos pode acelerar e melhorar estes processos manuais com a tecnologia de que dispomos hoje.

Não existe, do meu conhecimento, nenhuma investigação feita, que aplique os avanços mais recentes na área das CNNs à deteção automática de danos em veículos. Pese embora exista já alguma investigação que indica uma boa performance deste tipo de sistemas em tarefas de classificação relativas a imagens de veículos.

A presente dissertação ilustra a aplicação dos melhores e mais recentes avanços na área de classificação de imagem, nomeadamente os últimos e mais inovadores avanços em CNNs, à classificação de imagens de veículos danificados. Para isso foi desenvolvido um protótipo de um sistema capaz de identificar e classificar o grau dos danos evidenciados em imagens, bem como localizar, com algum grau de precisão, a área do veículo que se encontra danificada. Serão usadas as arquiteturas de CNNs mais promissoras, baseando o critério de escolha das arquiteturas em aspetos relacionados com a percentagem de erro de classificação, bem como na rapidez com que a classificação é feita.

Para o treino e teste do protótipo, foi criado um conjunto de dados que permite treinar e avaliar a performance do sistema, uma vez que tal conjunto de dados não se encontra disponível. Para tal, foram recolhidas imagens de pesquisas efetuadas em motores de busca, bem como recolhidas de *websites* de agências de segurança automóvel.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	2
1.2	Motivation and Objectives . . . . .	2
1.3	Dissertation Structure . . . . .	3
<b>2</b>	<b>Related Work</b>	<b>5</b>
2.1	Introduction . . . . .	5
2.2	Related Work . . . . .	5
2.2.1	Image based car damage detection . . . . .	5
2.2.2	Image Classification . . . . .	6
2.2.3	Available Datasets . . . . .	10
2.3	Existing Technologies . . . . .	11
2.4	Conclusions . . . . .	12
<b>3</b>	<b>Automatic detection of damages in cars</b>	<b>13</b>
3.1	Problem Definition . . . . .	14
3.1.1	Damage Severity Detection . . . . .	15
3.1.2	Damage Location Estimation . . . . .	15
3.2	Proposed Solution . . . . .	16
3.2.1	Distinguishing different types of damages . . . . .	16
3.2.2	Locating the damages . . . . .	16
3.3	Conclusions . . . . .	17
<b>4</b>	<b>Implementation</b>	<b>19</b>
4.1	Architectures . . . . .	19
4.1.1	Google LeNet . . . . .	19
4.1.2	Residual Network . . . . .	20
4.2	Changes in Topology . . . . .	21
4.3	Training Parameters . . . . .	22
4.4	Data Augmentation . . . . .	22
4.4.1	Mirroring . . . . .	22
4.4.2	Random Cropping . . . . .	23
4.5	Ensemble Methods . . . . .	23
4.5.1	Boosting . . . . .	23
4.5.2	Bagging . . . . .	24
4.6	Dataset Imbalances . . . . .	24
4.6.1	Random Oversampling . . . . .	24
4.6.2	Random Under-sampling . . . . .	25

## CONTENTS

4.6.3	Data Augmentation . . . . .	25
4.6.4	Impact on performance . . . . .	25
4.7	Implementation Details Summary . . . . .	26
<b>5</b>	<b>Results and Outputs</b>	<b>27</b>
5.1	Dataset . . . . .	27
5.2	Damage Severity Detection . . . . .	28
5.2.1	Google LeNet . . . . .	28
5.2.2	Resnet . . . . .	31
5.2.3	Human Performance . . . . .	31
5.3	Damage Location Estimation . . . . .	33
5.3.1	Google LeNet . . . . .	33
5.3.2	Resnet . . . . .	33
5.4	Chapter Conclusions . . . . .	36
<b>6</b>	<b>Conclusions and Future Work</b>	<b>39</b>
6.1	Conclusions . . . . .	39
6.2	Future Work . . . . .	40
	<b>References</b>	<b>43</b>

# List of Figures

2.1	Evolution of error of the top performing system on two ILSVRC tasks [ <a href="#">RDS<sup>+</sup>15</a> ]	8
2.2	Example of a section of a CNN learning a residual function $F(x)$ instead of the function $O(x)$ . . . . .	9
3.1	Image depicting a situation where the reflection makes the dent visible . . . . .	13
3.2	Another image depicting a situation where the reflection makes the dent visible .	14
3.3	System for distinguishing different kinds of damages . . . . .	17
3.4	System for distinguishing different kinds of damages . . . . .	17
4.1	Inception block basic components [ <a href="#">LCC<sup>+</sup>14</a> ] . . . . .	20
4.2	Google LeNet [ <a href="#">LCC<sup>+</sup>14</a> ] basic topology . . . . .	21
4.3	Residual Network 152 layer [ <a href="#">HZRS15a</a> ] basic topology . . . . .	22
5.1	Some examples from the dataset. The first, second, third and fourth columns feature examples of undamaged cars, cars with scratches, cars with mild damages and cars with severe damages, respectively . . . . .	29
5.2	Confusion matrix for Google LeNet architecture . . . . .	30
5.3	Confusion matrix for Google LeNet architecture ensemble of 3 base models . . .	32
5.4	Confusion matrix for Resnet architecture . . . . .	32
5.5	Confusion matrix for Google LeNet architecture . . . . .	34
5.6	Confusion matrix for Google LeNet ensemble . . . . .	35
5.7	Confusion matrix for Google LeNet architecture . . . . .	37

## LIST OF FIGURES

# List of Tables

2.1	Main characteristic of some frameworks with support for CNN . . . . .	12
3.1	Damage severity scale . . . . .	15
3.2	Damage location categories to be used . . . . .	16
5.1	Dataset distribution across damage severity scale . . . . .	29
5.2	Dataset distribution across damage location scale . . . . .	29
5.3	Performance metrics . . . . .	30
5.4	Performance metrics for the ensemble of 3 Google LeNet base models . . . . .	32
5.5	Performance metrics for the Resnet architecture . . . . .	34
5.6	Performance metrics for a single Google LeNet model . . . . .	34
5.7	Performance metrics for an ensemble of three Google LeNet base models . . . . .	35
5.8	Performance metrics for a single Resnet50 model . . . . .	37

## LIST OF TABLES

# Abbreviations

NN	Neural Network
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
SVM	Support Vector Machine
CAD	Computer Aided Design
ILSVRC	Internet Large Scale Visual Recognition Contest
ReLU	Rectified Linear Unit
MCC	Matthews correlation coefficient
ANSI	American National Standards Institute
NSC	National Safety Council
SGD	Stochastic Gradient Descent



# Chapter 1

## Introduction

Automatically detecting damages in images containing cars is a special case of an image classification task because, at its most basic level, detecting damages in images consists of assigning an image to a particular category or set of categories.

There is a lot of research done in image classification but, there aren't many works, that I am aware of, in car visual damage detection. Nevertheless, being able to automatically detect damages in cars is a research topic that has many possible real-world applications. Car insurance companies and car rentals have to deal with car damages on a daily basis. It often happens that cars have to be inspected for damages, often in circumstances that are inconvenient for customers, and costly to the companies themselves. It is therefore important to be able to automate car damage detection, making it both more convenient and cheaper.

This document starts by giving a broad overview of the current existing solutions for the problem of automatically identifying car damages in images. Solutions and strategies to solve problems that are only similar to the ones addressed here, will often be mentioned due to the lack of more specific research on the field of automated car damage detection. Some of these solutions, mainly the more promising ones, are later applied to the problem of automatically identifying damages in car images. A comparison of results with the ones obtained by previous works on the same problem will be given whenever possible.

The methodology used throughout this project consists of adapting solutions that are known to work in a variety of image classification problems, to the particular problem of visually identifying damages in cars and then evaluating their performance relatively to other existing solutions. Several classification systems will be tested, favouring the ones that can be used in practice. The dataset used to validate and evaluate performance of the developed solutions will be composed of images previously gathered from search engines, labelled for training and testing purposes.

### 1.1 Context

This thesis was proposed and is being developed in collaboration with Deloitte Consultores S.A., a consulting firm operating in Portugal. Deloitte Consultores S.A. works for many insurance companies in Portugal and abroad. Some of these companies are car insurance companies that deal with car damages on a daily basis, and to whom car damage related problems and processes are important.

### 1.2 Motivation and Objectives

Cars have a central role in today's world. Many people use cars every single day. There are businesses that depend on cars, and some of them base their activity in dealing with car damages and accidents. Non automated visual inspection of cars is a common task in some businesses. This is mainly the case of insurance companies. Cars are inspected both when a new coverage is bought and when an issue is reported to the insurance company. In both situations, inspections are only done by sampling. This happens because the cost of having an expert drive down to an often damaged vehicle, in order to inspect it is too high. It also causes delays for customer and company alike. This is especially important in cases where the insurance is not bought in one of the insurance companies dealerships, e.g. car insurance bought through the Internet. If the car for which the insurance was bought had to be inspected by an insurance company, the whole purpose of selling insurance through the Internet would be defeated; The customer would no longer have the convenience of quickly buying an insurance, since he would have to wait for the inspection to be performed before the effectiveness of the insurance was granted. On the other hand, the insurance company would no longer have the benefit of reduced costs in selling its products through these channels, because an expert would have to drive down to the insured car location, possibly a remote location, in order to perform the inspection. Both these inconveniences would be mitigated if the client uploaded some photos of the car to be automatically inspected by a system that could assert if the car has damages or not. Insurance companies often mitigate the problem by reducing the number of inspections they make. This is achieved either by not making them at all, or by making them by sampling, in an attempt to reduce fraud while not spending too much money inspecting it. Although an automated inspection might not be as effective as an inspection performed by an expert, it makes it possible to inspect all the vehicles, instead of just a few. On the other end, performing such inspections alongside with expert inspections might contribute to detect systematic biases, making it easier to discover potential fraudulent inspection processes or mistakes.

The possibility of classifying and distinguishing between different kinds of damages opens up the door for other more complex car damage related problems to be solved in an automated way. Other regression and classification problems that might include repair cost estimation or damaged parts estimation, problems of the uttermost importance for car insurance and car repair companies. These processes are also susceptible of fraud, and are also not automated nowadays.

## Introduction

Even though cars are very important to many people, they have been somehow neglected by the computer vision research community [YLCLT15].

Cars are very challenging to work with in computer vision. They present great variation in shape and form by slight variations of viewpoint [YLCLT15]. This, combined with the fact that cars usually have highly reflective metal bodies that make them very susceptible to inter object reflection, makes the problem of automatically detecting damages in car images a very challenging one. This is especially the case if the damages are relatively small, when they are easily mistaken by reflections.

There is some research done in the field of automatic damage detection in vehicles [J<sup>+</sup>13]. The approach taken often involves the usage of CAD models to estimate image perspective and assert differences between what's in the image and what the image is expected to have [J<sup>+</sup>13]. On the other hand, there is the research done in the broader field of image classification. This research often takes a different strategy in solving image classification problems. Image classification technology has endured tremendous changes in the last decade. We now have image classification systems that can allegedly outperform human beings [HZRS15b] in some tasks, a feat that could not be achieved by decades long research with traditional approaches. While the non traditional approach followed [J<sup>+</sup>13] made perfect sense when the state of the art image classification systems yielded results that were not good enough for this tasks, it now makes complete sense to apply the more performant and up to date state of the art technology to tackle the same problems, hopefully with better results.

Automated car damage detection using images is therefore an interesting research topic both due to the lack of work done in the field and due to the hope that new state of the art methods might bring serious performance improvements. Also, due to the importance cars have in many places, and the fact that cars present interesting unsolved challenges that state of the art image classification technologies might now be able to overcome, make this an even more interesting research topic.

In this document, I'll document the development of a prototype of a system, capable of identifying, and possibly locating damages in car images and evaluate it's performance comparing it to existing systems.

A dataset will be gathered for the purpose of training the system and evaluating its performance. This dataset will be built by gathering images from search engines and other publicly available datasets.

### 1.3 Dissertation Structure

In Chapter 2, the state of the art will be presented. Relevant research done, mainly in image classification, will be discussed and detailed here. Some existing technology often used in image classification is also discussed here. In Chapter 3, the context and problem this project tries to solve are presented in detail and the solution, based on the research detailed on the previous chapter, is discussed and described. In Chapter 4, implementation specific details are presented. Some

## Introduction

possible implementation related difficulties and constraints are also detailed here. In Chapter 5, the results and outputs obtained will be discussed and detailed. In Chapter 6, general conclusions are drawn and some hypothesis for possible future work are presented.

## Chapter 2

# Related Work

### 2.1 Introduction

Detecting damages in images of cars is a very specific research topic and therefore, there isn't much work done in this very specific research area. If we look at the more general field of image classification, the body of work is quite big. The more relevant work done in the last few years will be discussed here. Special attention will be given to research that involves image classification of cars or more generally, image classification of complex and noisy datasets. In computer vision, some of the most advanced research in image classification technology is usually benchmarked in challenges involving large image datasets such as Imagenet [DDS<sup>+</sup>09] or Pascal [EVGW<sup>+</sup>10] and therefore, performance on those datasets will be used as benchmark for discussed methods when no further information is available.

### 2.2 Related Work

#### 2.2.1 Image based car damage detection

The research done in automatic damage detection in car images [J<sup>+</sup>13] doesn't follow the standard image classification approach, that consists in using a classifier on top of hand-crafted features. The research found that addresses automatic damage detection in cars uses 3D CAD models of undamaged vehicles to obtain ground truth information. This is done in order to infer what the vehicle with damage would have looked like, had it not been damaged. This is achieved using 3D pose estimation algorithms [J<sup>+</sup>13]. Segmentation algorithms are then used to identify different areas of the car [J<sup>+</sup>13]. Edges are then extracted and the ones that are not predicted from the ground truth information are further processed [J<sup>+</sup>13]. In order to assert if these edges really should be considered damages, some heuristics are used. This is done in order to distinguish between edges that come from inter object reflections and those coming from real damages. This task alone of reflection detection, yields an MCC of no more than 0.34 [J<sup>+</sup>13]. The MCC is a

correlation coefficient between the observed and predicted binary classifications. These heuristics depend on multi-view geometry and therefore require two or more photographs of the same part of the vehicle [J<sup>+</sup>13].

This approach presents some major limitations that should not go unnoticed. Firstly, the work is only aimed towards automatic detection of mild damages in the form of scratches or peeled off paint [J<sup>+</sup>13]. Although those are damages that occur more frequently, other kinds of damages such as small dents, also occur very often, and are of great importance to real world applications. Also, the fact that the system needs a 3D CAD model of the damaged car is very limiting because it requires the user of the system to have a different CAD model for every car model whose damages are to be detected. Furthermore, it requires the system to know the specific model of the damaged car beforehand. Taking into account that there are thousands of different car models this might be drawback of using such a method. It should also be noted that the 3D CAD models with the required precision for the algorithm to work cost around USD 5000 each [J<sup>+</sup>13]. Lastly, the fact that some parts of the algorithm require the input to the system to be composed of more than one view of the damaged area is a major drawback as it might impede the detection in most real world situations, requiring a specialised image gathering procedure.

### 2.2.2 Image Classification

Image classification is a research area where important progresses have been made in the last few years. Figure 2.1 shows the reported error of the winning teams on two major tasks on ILSVRC [RDS<sup>+</sup>15], an important computer vision competition. The contest is composed of several tasks, two of which are an image classification task and a location task. These tasks, performed on an extremely noisy dataset [DDS<sup>+</sup>09], require the contending systems to automatically classify objects in one of 200 classes [RDS<sup>+</sup>15]. The progress made in the last years, of which the steady decrease in the error value is evidence, makes it clear that this is a research area where very important progresses were made.

Competitions like this one are often used as benchmarks for state of the art computer vision systems. Usually, these competitions make use of massive collections of images like ImageNet [DDS<sup>+</sup>09] or PASCAL [EVGW<sup>+</sup>10]. ILSVRC [RDS<sup>+</sup>15] results will be used in this subsection as evidence to show the progress done in recent research in image classification technology and it's applicability to solving demanding unconstrained problems like detecting damages in images of cars.

#### 2.2.2.1 Traditional Approach

The standard approach, that had state of the art results for many years in image classification problems, involves the usage of a two layered system. The two layers typically involved in these systems are a first layer composed of a feature extractor and a feature detector, and a second layer made of a classifier.

## Related Work

Feature extractors and detectors are algorithms, of which SIFT [Low99] is an example, that are carefully designed to extract features that are invariant to scaling, translation and rotations, from images or video frames. These feature extractors take images as input and output a vector of features. These algorithms are widely used in computer vision applications, and not only image classification system, because they allow practitioners to conveniently extract characteristics from images in a reliable and generic way. The extracted features, allow practitioners to identify similar objects, or instances of the same object, in different images. This property is very important for image classification systems that rely on these features to be able to classify the images. Feature extractors and detectors are not machine learning algorithms, they are hard coded procedures that are designed to extract features the same way in every image, hoping that these features are general enough to be used in a wide variety of situations and applications.

In image classification systems, the extracted features are handed to a classifier that learns how to classify images based on the extracted features. Classifiers rely on the capacity the extracted features have of making it possible to distinguish different classes of images. The classifier cannot correctly classify images if the provided features are unsuitable for the task.

Systems such as these were used for many years in real-world applications and important image classification contests such as Imagenet [DDS<sup>+</sup>09] whose top performing systems' results can be seen on Figure 2.1. The teams whose results are shown in Figure 2.1 for the years of 2010 and 2011 used systems that had this structure. The year of 2012 was a turning point in ILSVRC, the top performing systems started using NNs, instead of the more traditional systems.

### 2.2.2.2 Convolutional Neural Networks

The change in paradigm that the usage of NNs encompasses is a very important one. Prior to the usage of NNs in image classification, the practitioner had to use explicitly coded algorithms for detecting features. This was a job that often involved a lot of work. Machine Learning was not used throughout the whole system, only the last layer of the system, the classifier, had a capacity to learn and adapt itself to the specific problem. NNs, and especially CNNs, dramatically change that. CNNs extract features from images and learn how to do it. The practitioner doesn't need to craft complicated hard coded algorithms to extract those features.

Since the introduction of the first CNNs in the ILSVRC [RDS<sup>+</sup>15] in 2012, all the winning teams for the classification task have used different types of CNNs [KSH12, LCC<sup>+</sup>14, HZRS15a]. CNNs systems now clearly outperform other image classification systems such as the ones described in Section 2.2.2.1. State of the art CNNs have even crossed the boundary of what is considered to be the error rate for human beings in image classification tasks [HZRS15b]. Ever since its introduction in 2012, CNNs have been making steady progress in decreasing the error rate in both tasks. This is due to the introduction of new strategies that further improve the convergence and training speed of CNNs.

CNNs are a particular kind of NNs that give their name to the fact that they make extensive use of Convolutional Layers.

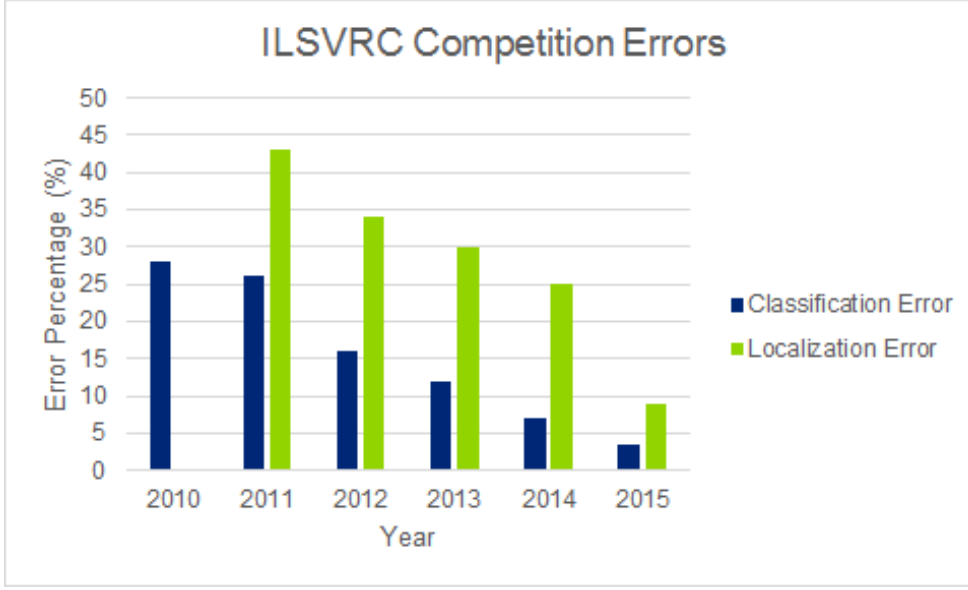


Figure 2.1: Evolution of error of the top performing system on two ILSVRC tasks [RDS<sup>+</sup>15]

Convolutional Layers consist of a learnable filter of fixed size (kernel) to be applied to images. The name comes from the fact that at each forward pass, the learned filter is convolved with the image resulting on a 2-dimensional map.

Another type of layer widely used in CNNs are the pooling layers. These layers perform a type of down-sampling on the input signal. There are various types of Pooling layers, max-pooling being the most common.

ReLU or Rectified Linear Units, are also widely used in CNNs. ReLUs are units that apply a function similar to Equation 2.1. There are often alternatives for these types of units, Equation 2.2 and 2.3 show two of them.

$$\text{ReLU} : f(x) = \max(0, x) \quad (2.1)$$

$$\text{Tanh} : f(x) = \tanh(x) \quad (2.2)$$

$$\text{Sigm} : f(x) = (1 + e^{-x})^{-1} \quad (2.3)$$

Fully Connected Layers are used too, as in more standard NNs. These layers consist of an array of neurons, each of which is connected to every single output of the previous layer.

Loss Layers are typically used at the end of the CNN to figure out the penalty for a given output and to provide feedback used by the CNN to learn.

In addition to these more standard set of layers, modern CNNs make use of a very extensive set of techniques and strategies that greatly enhance their performance. Dropout is one of such techniques [KSH12]. It consists on temporarily deactivating some neurons within the net, preventing it from over-fitting. This temporary deactivation of neurons is typically applied only to

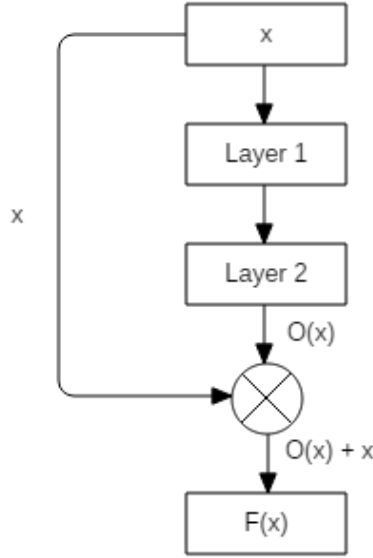


Figure 2.2: Example of a section of a CNN learning a residual function  $F(x)$  instead of the function  $O(x)$

Fully-connected layers. CNNs nowadays also make extensive use of GPU processing power that allows for a faster training, an idea pioneered in ILSVRC in 2012 [KSH12] that is now a standard.

Evidence shows that one way of improving a CNN performance is by stacking more layers and making the network effectively deeper [SLJ<sup>+</sup>15, SZ14]. The problem that often arises is that as more layers are stacked, the performance saturates and eventually starts to decrease at some point [SGS15, HS15]. To address this issue, the idea of deep residual networks has been proposed [HZRS15a]. Residual Networks try, with minor changes to the architecture, and with virtually no impact on performance, to approximate residual functions instead of standard functions. If we have a CNN with input  $x$  and we want the output to approximate a function  $O(x)$ , we can instead, approximate  $F(x) = O(x) - x$  and then compute  $O(x)$  from  $F(x)$  since  $O(x) = F(x) + x$  as shown in Figure 2.2. Evidence shows that CNNs train better and converge faster when the approximated function is in fact  $F(x)$ , a residual function [HZRS15a]. This approach opens door for usage of even deeper and more powerful models.

Some pre-processing techniques are also used to improve the data fed to the CNN. These techniques include normalization and zero-centring of the data. These techniques are mainly used in order to overcome the vanishing and exploding gradient problems [HBFS01], that often affects NN systems' training. Both problems arise when models are trained using gradient-based methods. Gradients are often computed using the chain rule. In case of near zero gradients, this lead to several very small numbers being multiplied together in order to calculate the changes in the last layers. This causes the last layers to suffer virtually no change at all in its outputs as the training progresses. The opposite happens when activation functions have high derivative values, leading

to the uncontrolled increase of the gradient of the last layers.

As NN systems tend to require a lot of examples, better results are obtained when data augmentation techniques are used. CNN frameworks often incorporate mechanisms to augment data. Cropping [KSH12] is a very simple technique where several crops of a single image are fed to the NN, augmenting the dataset by a factor equal to the number of crops. Rotations [KSH12] of the image are also used, augmenting the dataset by a factor equal to the number of rotations done.

The techniques summarised here, and others, together with a very complete set of open-source tools, foster the usage of CNNs, allowing for the fast creation of systems that perform very well in noisy datasets, clearly outperforming older image classification systems.

### 2.2.2.3 Image Classification on Cars

The most recent body of work in image classification aimed towards cars is also small [YLCLT15, ZTHW12]. Cars are objects that have very particular characteristics. Slight changes in view-point result in big visual differences and car models are also quite different from one another [YLCLT15], presenting interesting challenges to modern computer vision algorithms. The work I'm aware of, includes car verification, fine-grained model classification and attribute prediction tasks, done using CNN based systems [YLCLT15] and pose estimation [ZTHW12] using a 3D model based approach. In the case of fine-grained model classification, where the car model is supposed to be figured-out from a single car image, top-1 accuracy, depending on the orientation of the photographed vehicle, ranges from 0.43 to 0.76 on 431 possible classes [YLCLT15]. These results, although apparently not very good, are in fact very promising taking into account that some car models are very similar to one another, and the mistakes made by the system are often sensible ones, where the system classifies the image as belonging to a model that resembles the correct one. On the attribute prediction tasks, that included maximum speed estimation, door number, seat number or car type estimation among other tasks, the performance was also very promising with accuracies ranging from 0.54 to 0.62 for car type estimation, and 0.67 to 0.83 for door number estimation [YLCLT15]. Again, it should be noted that it is effectively very difficult to estimate number of doors a car has from some viewpoints, or to assert the car type from certain perspectives. In the case of car verification, where the task consisted of identifying if two images featured the same car model, results using CNNs were no lower than 0.76, a remarkable accuracy taking into account that the two pictures to be tested might have been taken in very different poses.

These results, clearly show that it is possible for CNNs to accurately perform difficult computer vision tasks on car images, making sensible predictions even under difficult conditions.

### 2.2.3 Available Datasets

There are some publicly available datasets featuring cars [YLCLT15, OLP09, GGA<sup>+</sup>12]. All the datasets I'm aware of, contain images featuring only undamaged cars. Even though these datasets are useful for training and testing models to perform tasks such as car recognition and pose estimation, that might be part of the prototype to be developed during this thesis, they are not

enough to train and test models for damage classification. To train models for such tasks, images will have to be gathered from search engines and car safety agencies databases.

### 2.3 Existing Technologies

During the last few years, a lot of frameworks for fast development and prototyping of NNs have emerged. Most of the available software is open source. These frameworks often differ from one another in their interfaces and support for pre-trained models.

Caffe [\[caf\]](#) is a framework written in C++ and Python that has Python, command line, Matlab and C++ interfaces, making it a very attractive candidate for fast prototyping of CNNs. It supports CNNs and has support for CUDA and cuDNN technology. Caffe is very popular among researchers and has a big community.

Another important framework is Theano [\[The\]](#). It is written in Python and has Python interface. It supports CNNs and makes use of CUDA technology. It lacks a command line interface to train or fine-tune models. It also has support for pretrained models.

Torch [\[tor\]](#) is another option. It supports CNNs and CUDA. It has its own scripting interface, a C/C++ interface and a command line interface too. Uses LuaJIT as the preferred scripting language. This last feature might be a drawback since I'm not familiar with Lua.

Deeplearning4j [\[dee\]](#) is a framework for NNs implemented in Java that has interfaces for various languages that run on the JVM such as Java, Clojure and Scala. It has no command line interface. It has CUDA support and has pre-trained models. The lack of command line interface and the fact that it isn't a mature framework are drawbacks.

Tensorflow [\[tf\]](#) is another major open source framework for NNs. It has a Python interface but lacks a CLI interface. It has some support for pre-trained models, but the support for these models is not as good as Caffe's.

Keras [\[ker\]](#) is a framework that relies on a Theano or Tensorflow backend. It exposes a slightly friendlier interface API than Theano or Tensorflow, and also offers a slightly better support for pretrained models. It lacks a CLI.

CNTK [\[cnt\]](#) is another framework for working with CNN's. It has a Python and C++ interface and also a CLI. It has great support for Windows OS but a slightly worse support for Linux based systems, which is a drawback.

Deep CNNs often require a lot of examples for training. This constraint is especially important in cases where labelled data is not very abundant as is the case of the problem addressed here. One way of mitigating this problem is by using pre-trained nets, that require less data for training, although their performance might be slightly worse [\[VGBP15\]](#). Since this is a very important issue to be taken into account during this work, frameworks not having pre-trained models were omitted from this discussion.

Some of this data discussed here is summarised in Table [2.1](#)

Table 2.1: Main characteristic of some frameworks with support for CNN

Framework	Language	Command Line Interface	CUDA Support
Caffe	C++ and Python	Yes	Yes
Theano	Python	No	Yes
Torch	C/C++ and Lua	Yes	Yes
Deeplearning4j	Java, Scala and Clojure	No	Yes
Tensorflow	C++ and Python	No	Yes
Keras	Python	No	Yes
CNTK	C++, Python and BrainScript	Yes	Yes

Taking into account the amount of evidence gathered and discussed here, I believe that Caffe is the best option to be used for the system to be developed. Caffe's major benefits include the fact that it has great support for pre-trained CNN models, as well as a big community.

## 2.4 Conclusions

It is clear now, that CNNs outperform more traditional systems in some computer vision tasks such as image classification. CNN technology has recently crossed the boundary of human performance in object recognition [HZRS15b] and recent developments allow it to take more advantage of deeper and more powerful architectures [HZRS15a]. The performances exhibited by modern CNNs [RDS<sup>+</sup>15] make it possible to apply them to problems in computer vision that were previously solved using other approaches. Visually identifying damages in images of cars may very well be one of these areas where the important advancements in CNN technology might prove beneficial. Therefore, it makes perfect sense to apply these new CNN architectures and techniques to solving the problem of visually identifying damages in images of cars.

There is a wide range of tools for CNN system development. One of the more complete and widely used tools is Caffe. It allows for fast prototyping of NN systems by using pre-trained models and by allowing an interaction with the framework using a command line interface, without the need of writing code to perform simple tasks. It will be the framework used for developing the prototype of a system to visually identify and localise damages in images of cars.

## Chapter 3

# Automatic detection of damages in cars

The problem of identifying damages in images of cars is far from trivial. Cars are complex objects with important intra-class differences but also subtle ones. A van and a city car are not very alike, even though both are cars. On the other hand, two car models from the same maker often are very much alike, while they are in fact different models and exhibit subtle but important differences. To make things worse, a photo taken from a car's side view is very different from a photo taken from a front view perspective, making cars a very difficult class of objects to work with in computer vision applications.

Photographs taken in uncontrolled environments also present important and difficult challenges in computer vision. That is especially the case if the photographed objects exhibit highly reflective surfaces like cars do [YLCLT15]. Distinguishing dents from reflections is a difficult task. But there are situations when it's the reflection itself, that makes the dent visible as shown on Figures 3.1 and 3.2. It is obvious then that although reflections may be undesirable in some situations, they provide valuable visual clues that indicate the presence of minor car body damages that would otherwise go unnoticed.



Figure 3.1: Image depicting a situation where the reflection makes the dent visible

This document aims to document the development of a prototype of a system capable of identifying the type of damage in car images and also to give a rough estimate of the location of the



Figure 3.2: Another image depicting a situation where the reflection makes the dent visible

damage in the car's external surface. The system should be able to accurately distinguish and locate damages in uncontrolled images, taken in uncontrolled lightning conditions, viewpoints and distances.

### 3.1 Problem Definition

The goal of this dissertation is to develop a system capable of automatically identifying and locating damages in images of cars.

Automatic detection of car damages in images is a task that may be tackled using a wide variety of approaches. The proposed approach splits the general problem of identifying damages in two different problems. The first problem, consists of distinguishing different types of damages based on their severity. It is important to notice that distinguishing different kinds of damages requires the system to first be able to detect it's presence. The second task would be to locate the damages in the car's external surface by saying which areas are damaged. This last problem involves detecting the damage and being able to locate it in the car.

Both are multi-class classification tasks with a different number of classes.

These two distinct tasks, if successfully performed, may lay ground for more complex tasks such as car repair cost estimation, or other higher level tasks that may require a severity and damage locations estimation to be performed.

There are several available location and severity scales that might be used to distinguish different kind of damages. A slight modification of framework proposed by the NSC will be used [Cou83], adapting it to the necessities of the system to be developed here.

The NSC is a forum whose function is to advise and assist the US president on national security and foreign policies. It proposes, among other things, policies regarding traffic laws and car safety.

The mentioned framework sets parameters for assessing damage sustained by motor vehicles in traffic crashes. It establishes a scale for type of impact (direction) and severity of damages. The document describes sixteen different impact types, and seven different severity levels.

### 3.1.1 Damage Severity Detection

In the first task the number of classes is equal to the number of different classes of damages, plus one class to assign to the images where no damages are present.

In terms of damage severity scales, four different categories will be used corresponding to the absence of damages, instead of the 7 proposed by the NSC framework. Table 3.1 shows the four categories to be used.

Table 3.1: Damage severity scale

Category Name	Description
No Damage	The images shows no visible damages
Paint Damage	The image shows paint damages such as stains and scratches
Minor Damage	The image shows minor damages such as localised dents or broken headlights
Major Damage	The image shows major damages from big dents to widespread car destruction

This classification in four different classes is used because it is meaningful while not being too restrictive. A too restrictive scale would end up with classes that had very few samples assigned to them, making it very difficult to subsequently train the models.

### 3.1.2 Damage Location Estimation

For the second task at least two different approaches are possible for the formulation of the problem. The first and simplest approach consists of splitting the car in  $N$  different areas and assigning one class to each of them. Using this approach, assigning an image to a class  $C$  is equivalent to saying that area  $C$  is damaged. It is important to notice that this approach ignores the fact that more than one area may be damaged in a given picture, making it insufficient to assign each image to a single class in these terms. In the case of more than one area being damaged, it often happens that the damaged areas are contiguous. It is also important to notice that it is very difficult to have a significant number of images where more than two areas are visible and damaged at the same time. A possible approach to cover the situation of two contiguous damaged areas is to, in addition to the  $N$  classes corresponding to one of the  $N$  areas being damaged, create  $N$  more classes corresponding to the cases where any two contiguous areas damaged, making a total of  $2N$  different classes. The choosing of the approach to use will be done based on the incidence of instances where more than one area is damaged.

In terms of damage location, the NSC framework will also be adapted. The different damage location categories to be used are shown on Table 3.2. The damage locations correspond to a simplification of the categorisation proposed by the NSC report [Cou83] by removing some unnecessary categories and by merging others.

This categorisation provides a small enough number of categories to make most images clearly fall into one and only one of the categories, while also providing a big enough number of categories to make the categorisation useful for being used in a real world system. Here again, a scale with too many different classes might end up with few samples in some of the categories. A

Table 3.2: Damage location categories to be used

Category Name	Description
No Damage	No damage shown
Front	Front damage due to impact
Front Side	Front left or right corner damage
Back	Back damage
Back Side	Back left or right corner damage
Side	Right or left side damage in vicinity of passenger or driver compartment

drawback that would almost certainly badly affect the performance of the models trained with such a classification system.

## 3.2 Proposed Solution

CNNs have been consistently yielding good results in visual classification tasks and as such, applying them to the task of visually identifying damages in cars might be an interesting and promising approach to follow.

The two problems will be solved with the use of CNNs. The winner submission of the ILSVRC 2015 challenge [HZRS15a] (Residual Network) and the winner submission of the ILSVRC 2014 challenge [LCC<sup>+</sup>14] (Google LeNet) will be used to solve the two problems mentioned on Section 3.1. The Residual Network was chosen because it is the current state of the art in image classification technology. In spite of the Residual Networks better performance on big datasets like the Imagenet dataset [DDS<sup>+</sup>09], Google LeNet architecture still is the architecture that has the best error percentage per training time ratio, making it a very appealing architecture to be used in image classification problems, especially when the computational resources available are scarce, as it's the case. The architectures will have to be adapted to the specific problems. As far as possible, pre-trained models, trained on the Imagenet dataset will be used, allowing for the models to be fine-tuned on bigger datasets, like the one gathered.

### 3.2.1 Distinguishing different types of damages

Distinguishing between different kinds of damages is, as detailed on Section 3.1, a four class classification problem. An overview of a system to correctly classify the different kinds of damages may be seen on Figure 3.3. Again, both architectures will be compared here, both pre-trained on the Imagenet dataset [DDS<sup>+</sup>09] and fine-tuned on a 3000 image dataset featuring both damaged and undamaged cars.

### 3.2.2 Locating the damages

The damage location task, like the previous one, will be undergone by using variations of the architectures mentioned before. An activity diagram detailing the system to locate the damages

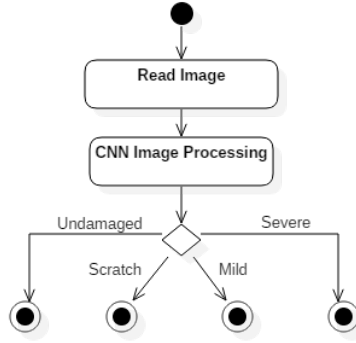


Figure 3.3: System for distinguishing different kinds of damages

in the car is shown on Figure 3.4. The system used will be trained as the ones in the previous problems.

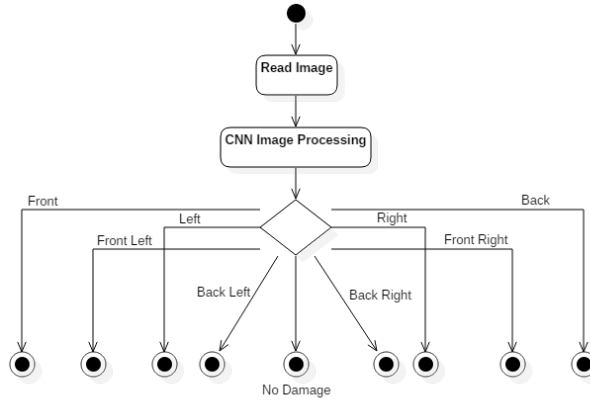


Figure 3.4: System for distinguishing different kinds of damages

### 3.3 Conclusions

The general problem addressed in this document will be divided in two simpler sub-problems addressed separately. Two different promising CNN architectures will be tried on both problems and performances will be compared. While the Residual Network architecture [HZRS15a] is the current state of the art in terms of reducing the error percentage in image classification tasks, the Google LeNet architecture [LCC<sup>+</sup>14] still is the fastest network to train with an error rate that resembles the state of the art performance. It is therefore interesting to compare the performance of both architectures. Both architectures will have to be adapted to perform the desired classification tasks, these adaptations will be detailed on Section 4. The two problems, while related in some

## Automatic detection of damages in cars

aspects, present different degrees of difficulty and tackle two different aspects important to any system intending to extract useful information from images of damaged cars.

## Chapter 4

# Implementation

The implementation of a solution for the problem of visually identifying damages in images of cars will consist, as mentioned in Section 3, in a system based on a CNN. The system, will be built with state of the art CNN technology, addressing both tasks mentioned in Section 3. The system will be developed using the Caffe framework and pre-trained models will be used whenever possible as said on Section 2.3. For both tasks, more than one architecture will be used, in order to measure the performance of different architectures.

### 4.1 Architectures

Two architectures are of great interest to this project. The Residual Net architecture [HZRS15a], that won the 2015 edition of ILSVRC, is in fact, the state of the art architecture for visual classification tasks. On the other hand, the Google LeNet architecture [LCC<sup>+</sup>14], that has a slightly larger error rate on the Imagenet [DDS<sup>+</sup>09] dataset, still is one of the fastest architectures to yield an acceptable error rate. Both are therefore interesting to be applied to this project for the reasons mentioned before.

#### 4.1.1 Google LeNet

The basic principle and innovation introduced by the Google LeNet architecture was the usage of the so called Inception blocks. These Inception blocks consist of a set of convolution layers whose output is fed to a concatenation layer. The basic topology of the inception blocks is shown on Figure 4.1.

The network overall topology is mainly made up of 22 Inception blocks stacked on top of each other, as shown on Figure 4.2. Some inception blocks have max pooling layers between them, but such details are omitted from Figure 4.2 for the sake of simplicity. In order to adapt this topology to the various tasks, the Inner Product layer, the last layer in the network, will have to be modified. The Inner Product layer, which is in fact a fully connected layer, whose purpose was detailed on

## Implementation

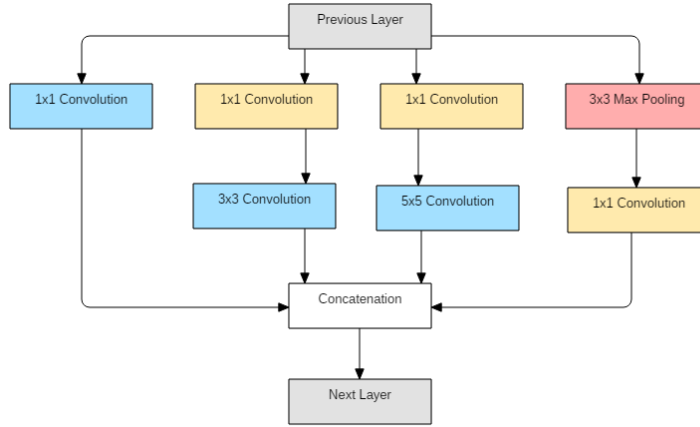


Figure 4.1: Inception block basic components [LCC<sup>+</sup>14]

Section 2.2.2.2, has one output per target class. It is clear then that a change in the number of classes requires a change in this layer. The original network has in fact 1000 outputs, one per each ILSVRC [RDS<sup>+</sup>15] class. This number of outputs will have to be reduced to adapt the network to the specific tasks.

The training procedure for this CNN, as show on the original paper [LCC<sup>+</sup>14], involves the usage of three different softmax layers with loss. The merits of this multiple loss approach is said to have a relatively minor impact on the overall accuracy of the net, around 0.5% [LCC<sup>+</sup>14]. It is also mentioned that the minor 0.5% increase in accuracy may be achieved by using only one auxiliary softmax layer with loss, instead of the original two. In spite of the minor benefits, and since these auxiliary softmax layers do not create any significant impact on the time required for the net to train, and for the sake of simplicity, they will be kept throughout this project.

### 4.1.2 Residual Network

The Residual Network architecture, pioneered in the 2015 edition of the ILSVRC [HZRS15a], features the usage of residual blocks shown on Figure 2.2 and whose original topology is shown on Figure 4.3. The rationale behind residual blocks is explained in Section 2.2.2.2. In fact the original paper [HZRS15a], mentions several variations of the architecture shown on Figure 4.3, where the only difference between them is the number of stacked residual blocks. The one presented here is in fact the top performing one.

In the case of this CNN, the recommended way of training detailed on the paper is to use a single softmax layer with loss. This is the procedure to be used in fine-tuning the network.

## Implementation

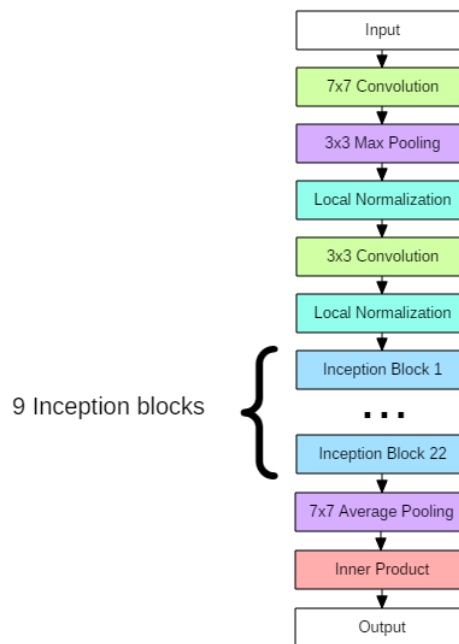


Figure 4.2: Google LeNet [LCC<sup>+</sup>14] basic topology

## 4.2 Changes in Topology

Both detailed architectures were conceived with the Imagenet dataset [DDS<sup>+</sup>09] in mind. They perform especially well on big datasets with a great number of classes such as the Imagenet dataset.

The characteristics of the tasks we're going to adapt the architectures to are quite different. The number of classes involved is typically smaller, by a factor of 100, at least, and the differences between the various classes are subtler.

These differences in problem characteristics might justify deeper changes in both architectures. As a rule of thumb, problems with less classes, and smaller training samples, often benefit from shallower and thinner architectures. Shallower and thinner architectures typically are faster to train and less prone to overfitting. These advantages come at the cost of a reduced representational power of the net. The decrease in the representational power of the network, materialised in the form of less extracted features from images, might not affect the performance of the system in the new task, especially if the number of classes is reduced too.

Making an architecture shallower or thinner has some associated problems. Such deep changes often make the usage of pre-trained models infeasible. This happens because the transfer of weights from the original model to the new one is not easy to do correctly, and not useful at all in some cases because some layers often depend on the previous one. Without a pre-trained model, the net will have to be trained from scratch, a complicated task to be carried out effectively with a relatively small dataset, like the one we have at our disposal. Deep pre-trained models, especially the ones trained on big datasets such as Imagenet, tend to grasp the feature extraction capabili-

## Implementation

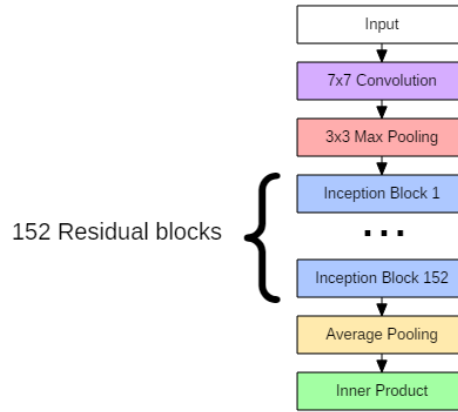


Figure 4.3: Residual Network 152 layer [HZRS15a] basic topology

ties that allow them to perform quite well in a wide variety of problems and it's use is therefore paramount to obtain good results when data scarcity is an issue. Discarding the possibility of using pre-trained models would probably then be a bad decision.

### 4.3 Training Parameters

Both the Google LeNet [LCC<sup>+</sup>14] and the Residual Net [HZRS15a] papers suggest the usage of SGD with mini-batches for training. This will be the preferred method used for fine-tuning the models. As to the specific parameters to be used in fine-tuning, the ones mentioned on the paper won't probably be adequate to be used for fine-tuning as they are especially suited for training both nets from scratch.

The best meta-parameters for training will have to be identified during the development of the project, possibly making use of parameters used in research where nets were fine-tuned, such as [YLCLT15], where the Google LeNet architecture was fine-tuned on a car related dataset.

### 4.4 Data Augmentation

It is often necessary to augment the dataset in order to obtain better results from DCNNs. These techniques are even more necessary when the datasets are small. Not all techniques can be applied to all situations and it's excessive use may even hurt the performance of the models.

#### 4.4.1 Mirroring

Mirroring is a technique that is used very often [MDRM15]. It consists on a random left-right flipping of the images presented to the model, making it possible for the dataset to be duplicated in size. This technique cannot be used when the task performed by the model involves some kind of location or classification according to sides (left or right classification). This is the case of the

task mentioned on Section 3.1.2 where left and right matters. This happens because an image labelled as exhibiting a damage in the right side of the car, actually has a damage on the left side of the vehicle if the image is flipped.

### 4.4.2 Random Cropping

Other very popular technique, random cropping [MDRM15] consists on feeding the models with randomly cropped patches, of fixed size, instead of feeding it the whole image. This allows for an augmentation of the number of samples available for training, many times fold. It is also very used, possibly in conjunction with mirroring to boost the performance of already trained models, by getting their prediction for a set of crops from the image to be classified, and averaging them, instead of taking into account only the classification for the whole image. Crop size may vary. It is important to find a size that is big enough to be classified, and small enough so that different crops are significantly different from one another. In this work, and taking into account that we used only pre-trained models, there wasn't much choice regarding crop sizes, as they had to be the same as the ones used by the original model. This is closely linked to the reason gave in Section 4.2 for not changing the topology of the original networks too much, because change in the crop size would imply a change in the width of the network.

## 4.5 Ensemble Methods

While there is a great diversity of ensemble methods, there are two major groups typically used in computer vision related problems. These are boosting, also known as sequential ensemble, and bagging, also known as parallel ensemble. Both have their benefits, their adequacy to the problems addressed in this document is discussed below.

### 4.5.1 Boosting

Boosting, sequential ensemble, is a meta-algorithm typically used to correct bias in the predictions of models. These bias and/or variance issues typically arise when the models are not able to grasp the complexity of the classified entities. This may happen when the model is not powerful enough to completely understand the domain they are modelling. The main idea behind boosting is that it is possible to develop a model arbitrarily well-correlated with the true classification (strong learner) by using a combinations of only slightly correlated classifiers (weak classifiers). There are several variations of boosting algorithms the most important of them being AdaBoost (Adaptive Boosting) [FSA99], Gradient Tree Boosting and XGBoost[CG16].

For boosting it is desirable to have an usually great number of fast to train, weak learners. DC-NNs are usually not fast to train and typically are not weak, highly biased learners and therefore, they are not very adequate to be used as base learners for Boosting models. This is the chief reason why this approach was not attempted in this work.

### 4.5.2 Bagging

Bagging [Bre96], also known as bootstrap aggregating, is another family of meta-algorithms that aimed at improving the performance of machine learning models. It is usually applied to reduce variance and increase stability of base learners. The method consists in applying the learning algorithm to each bootstrap sample, and then averaging the resulting prediction rules.

It might be argued that Dropout [KSH12] heavily used in the architectures we're trying to use is, in fact an ensemble technique, that like bagging aims at reducing the effects of overfitting. While the similarity between bagging and dropout is that they average submodels trained in different subsets of the data. Their mechanics, however, is completely different. While dropout is used to avoid overfitting, bagging is mainly used to reduce variance.

## 4.6 Dataset Imbalances

As mentioned on Section 2.2.3, no publicly available datasets, that I'm aware of, feature damaged and undamaged vehicles. It is therefore necessary to make such dataset. Classification systems, and CNNs in specific, generally depend on large datasets to be trained. The algorithms used for image classification tasks usually work better with balanced datasets. Here balancing means that the samples given to the model should have roughly the same number of examples of each class. This might be very difficult to achieve because certain situations are very rare. Lets take the system to identify the location of the damage as example. It's many times more likely to find images of accidents where the front of the vehicle is damaged, then to find situations where the side of the vehicle is damaged, simply because cars tend to move forward and not sideways. On the other hand, a sample that is balanced in terms of damaged/undamaged vehicles cannot be balanced if we're trying to distinguish different classes of damages and the classes involved are "undamaged vehicle", "scratch", "minor damage" and "major damage".

There are several ways to mitigate this inherent imbalance in the dataset as shown in [DPRGOK15, PBS15]. Some of them will be detailed in the present section, since some of them were used in this project. We will focus here in pre-processing techniques, rather than ensemble based techniques, mainly because pre-processing techniques can be more easily integrated in existing systems on a later development stage, and often can be used in conjunction with ensemble techniques.

### 4.6.1 Random Oversampling

One of the simplest strategies used to balance out datasets, and often one of the most effective, is to simply over-sample the classes with less instances. This is accomplished by sampling more than one time several randomly chosen instances of the less frequent classes.

The main problem with this approach is that it often leads to overfitting problems, especially in the case of extreme imbalances. This can be overcome, for the most part, with the usage of ensembles [DPRGOK15], by simply making sure oversampled instances are different for different

nets in the ensemble. It is an advantageous method, mainly because it allows for fairly good results with a simple approach.

### 4.6.2 Random Under-sampling

Similar in concept to the method presented in Section 4.6.1. Random Under-sampling consists in removing some instances of the classes that have more examples. This technique has the problem of potentially removing useful data that would help the system generalise better. This problem may again be mitigated by the use of ensembles where different instances are under-sampled in different nets inside the ensemble. This is too, a fairly simple method to implement that often yields good results.

### 4.6.3 Data Augmentation

Another more sophisticated method to balance the sample is to use data augmentation. The basic idea behind this strategy is to apply data augmentation techniques only to under-represented classes and not to over-represented classes.

This technique has the advantage of allowing a bigger number of examples per class than the number of examples of the least represented class, as opposed to what is possible using oversampling without ensembles. On the other hand, it is more difficult to implement, and often involves the necessity of performing data augmentation technique prior to feeding the sample to the model. If the sample is big, this might mean that more storage space is required to store the augmented sample. It may also lead to some form of overfitting that might be diminished by the usage of this technique in conjunction with ensembles.

### 4.6.4 Impact on performance

The impact of dataset imbalances on the overall system performances usually doesn't exceed 5%, as reported in [PBS15], especially if the imbalance is not severe. SVM based classification systems are typically less prone to be affected by class imbalances [PBS15]. This fact makes SVMs especially adequate for situations where the class imbalance is severe. Some research points at the fact that only about 30% of performance loss from class imbalance are possible to be recovered with such techniques [PBS15], which means that the performance increase due to usage of such techniques is no greater than 1,5%. Other research, which evaluates the usage of such techniques together with ensemble techniques, suggests that most of the performance loss may indeed be recovered, which means that more than 50% of the 5% may be recovered [DPRGOK15]. Tree based models were not mentioned in this discussion because they usually exhibit worse performance than SVMs or NNs in image classification tasks.

## 4.7 Implementation Details Summary

Implementation wise, the modifications that are necessary to make to the original CNNs' architecture are minor. Only the classifier part of the CNNs will have to be modified in order for the net to perform as expected. Deeper changes, although desirable in other contexts, will not be used here because they might make the usage of pre-trained models infeasible.

The articles that describe the Residual Net and Google LeNet architectures only mention parameters used to train the nets from scratch. To fine-tune the hyperparameters, the parameters are somehow different from the ones that have an optimal performance when training the same network from scratch. Papers such as [YLCLT15] will be used in order to find a starting point for the parameters to be used in the fine-tuning process.

Class imbalances will likely be a relevant setback and therefore, the usage of some techniques detailed on Section 4.6 will allow for imbalance issues to be mitigated.

## Chapter 5

# Results and Outputs

The results obtained using the techniques described in the previous chapter are detailed here. This chapter starts by briefly giving details of the dataset developed and used for training and testing the models developed. The results obtained when training the models described in 4, with the required modifications and additions, employing some of the techniques already mentioned, according to previously discussed criteria, are then presented and explained.

### 5.1 Dataset

The dataset required for the system to be trained is composed of 15671 images labelled according to damage severity and 10016 labelled according to the place of the damage. Most of the images are present in both datasets. The discrepancy in number of samples in each case is due to the fact that some images could be unmistakably labelled according to one scale and not the other. The dataset features both damaged and undamaged vehicles in various poses and lighting conditions.

The distribution of samples along the damage severity scale is quite unbalanced. This can be seen on Table 5.1. The chief reason for this imbalance is the fact that most images that could be found were of undamaged cars. The fact that it is very rare for cars only to get the paint peeled off, without any further damages also contributed to the imbalance of the dataset.

The distribution of samples according to the location of the damage is also a very imbalanced one, with more than 70% of images being of undamaged cars. Other classes are relatively balanced, with some under represented classes where damages are less prone to exist. This may be seen in Table 5.2.

These imbalances reinforce the need to use data balancing techniques mentioned and detailed on Section 4.6. The techniques employed were mostly Random Oversampling, described in Section 4.6.1. This particular technique was used, mainly because of its simplicity. While Random Under-sampling was also an option, with the same properties in what regards to simplicity of implementation, it would greatly reduce the size of the dataset given to the model for training.

Since the already relatively small size of the dataset was likely a problem for training deep models, oversampling ended up being used.

Random over-sample inevitably brings some problems of over-fitting with it. Ensembles, together with a slightly increased dropout rate were tried in order to minimise these effects. The results are shown on the next few sections.

Some examples gathered from the dataset may be seen in Figure 5.1 below. As this figure clearly shows, there are images with various poses, severity of damages and light conditions, making it a challenging dataset for computer vision tasks.

## 5.2 Damage Severity Detection

Both architectures, Google LeNet [LCC<sup>+</sup>14] and Resnet [HZRS15a] were tried on this problem. Google LeNet performed, as expected, slightly worse than the Resnet architecture, scoring a maximum of 75% accuracy in a validation set composed of 100 images not previously seen by the model. Resnet although more performant, took more time to train until over-fitting was reached.

Stochastic Gradient Descent was used to train all models. Random Cropping and Mirroring were also used in all models.

### 5.2.1 Google LeNet

Google LeNet maximum score of 75% was achieved at iteration 1500. The confusion matrix may be seen in Figure 5.2. Some performance metrics such as precision, recall rate and f1-score are shown on Table 5.3.

At these numbers and matrix clearly show, the number of times the model classifies an undamaged or scratched vehicle as having heavy damages is quite small. The reverse is also true. It is also clear that the model finds it very difficult to distinguish between cars with scratches or mild damages. This was expected for two reasons. Firstly scratches are often, but not always, accompanied by mild damages, and it is therefore very difficult to find examples of the one, that are not examples of the other. On the other hand, the poor quality, in terms of resolution, of the images gathered make it very difficult to detect damages that are not very evident, even to humans.

#### 5.2.1.1 Ensemble

In order to further improve on the results obtained by a single model, a small parallel ensemble was tried.

The ensemble was trained by dividing the available training samples by the number of base models, three in this case. This is what's commonly called Bootstrap Aggregating. Details on Bootstrap Aggregating are mentioned and discussed in Section 4.5.2. Since the models take nearly a week to train, only a small ensemble size of three base models was attempted. Also the relatively small size of the dataset was taken into account when deciding the number of individual base models to use.

## Results and Outputs

Table 5.1: Dataset distribution across damage severity scale

Category Name	Nr. of Images	Percentage
No Damage	7846	50%
Paint Damage	565	3.6%
Minor Damage	3021	19.6%
Major Damage	4239	27%
<b>Total</b>	<b>15671</b>	<b>100%</b>

Table 5.2: Dataset distribution across damage location scale

Category Name	Nr. of Images	Percentage
No Damage	7062	70.5%
Front	757	7.5%
Back	468	4.7%
Front-Side	733	7.3%
Side	488	4.8%
Back-Side	508	5%
<b>Total</b>	<b>10016</b>	<b>100%</b>

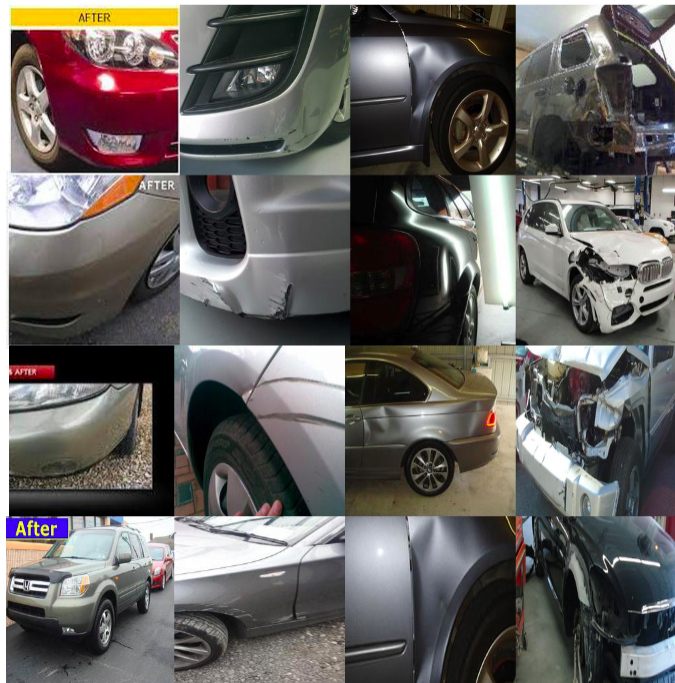


Figure 5.1: Some examples from the dataset. The first, second, third and fourth columns feature examples of undamaged cars, cars with scratches, cars with mild damages and cars with severe damages, respectively

## Results and Outputs

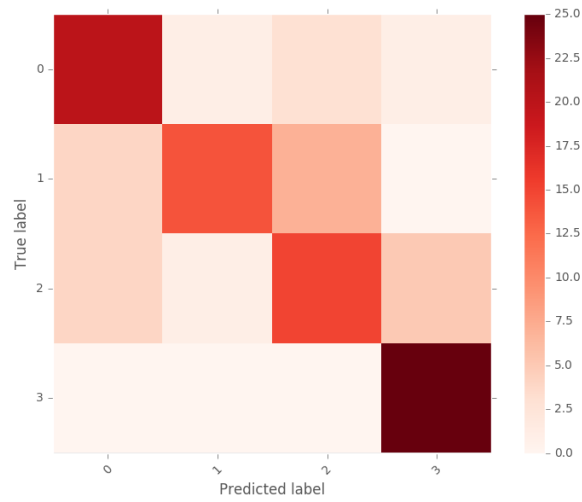


Figure 5.2: Confusion matrix for Google LeNet architecture

Table 5.3: Performance metrics

Category Name	Precision	Recall	F1-score	Support
No Damage	0.75	0.84	0.79	25%
Paint Damage	0.81	0.52	0.63	25%
Minor Damage	0.60	0.72	0.65	25%
Major Damage	0.88	0.92	0.90	25%
<b>Avg/Total</b>	<b>0.76</b>	<b>0.75</b>	<b>0.75</b>	<b>100%</b>

The output of the mentioned ensemble, whose confusion matrix is shown in Figure 5.3, was obtained by averaging the probabilities assigned to each category by each of the base models, and by then picking the highest one. The labels on the image represent the various classes of the scales described in chapter 3. The 2% boost in performance is in line with the performance increase expected with the use of small ensembles. The ensemble also exhibited a more stable behaviour in the sense that its accuracy didn't vary much as the number of training iterations progressed. Again this behaviour is the expected one when Bagging is used, as explained in Section 4.5.2. Table 5.4 also confirms the slight increase in performance. It is interesting to note the measurable decrease in the precision and recall rate for the "No Damage" category. All other classes improved its performance, especially the "Paint Damage" class, that saw its recall rate raised from 52% to 72%.

### 5.2.2 Resnet

The Resnet architecture performance for this task is, as expected slightly better than the ones reported by a Google LeNet. This happens although the net used is smaller than the top scoring model for the Imagenet dataset (ILSVRC 2015). The top performance achieved by this model was 76%. As the confusion matrix in Figure 5.4 and Table 5.5 show, the marks and error distribution are quite similar to the ones observed for the other architecture.

### 5.2.3 Human Performance

A small experiment was conducted in order to assert the performance of untrained human beings when performing the same task. This experiment lacks the rigour required to ascertain the true average human performance on this dataset and task. It merely serves the purpose of ascertaining what may be considered an acceptable mark for the computational models tested. This is important because the task is somewhat subjective, either because the dataset was labelled following no particular rule or exact criteria, or because the quality of the images make it very difficult, in some samples, to distinguish what damage really is in them. There really wasn't any reason why the labelling should have been done in a more rigorous manner, since one of the main goals of the present work was to ascertain if it was possible to use modern image classification technology to help perform the tasks related to damage estimation done by humans. And since this task is implicitly performed by humans, no particular rule or precise criteria, other than common sense was established.

The subjectiveness of the task hints at the fact that a performance far bellow 100% might actually be a very acceptable mark.

The results gathered by testing the performance of ten different human beings on the same data, against which the performance of the models was tested show that human accuracy lays somewhere in the interval encompassed between 75% and 81%. This interval's boundaries are the accuracies of the less and more accurate humans in the experiment.

## Results and Outputs

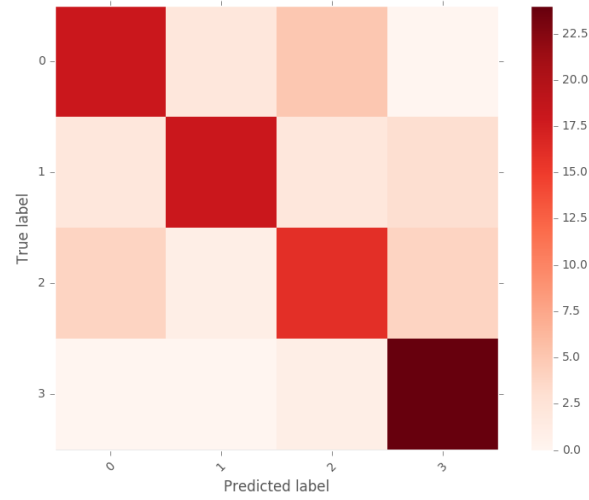


Figure 5.3: Confusion matrix for Google LeNet architecture ensemble of 3 base models

Table 5.4: Performance metrics for the ensemble of 3 Google LeNet base models

Category Name	Precision	Recall	F1-score	Support
No Damage	0.75	0.72	0.73	25%
Paint Damage	0.86	0.72	0.78	25%
Minor Damage	0.67	0.64	0.65	25%
Major Damage	0.77	0.96	0.86	25%
<b>Avg/Total</b>	<b>0.76</b>	<b>0.76</b>	<b>0.76</b>	<b>100%</b>

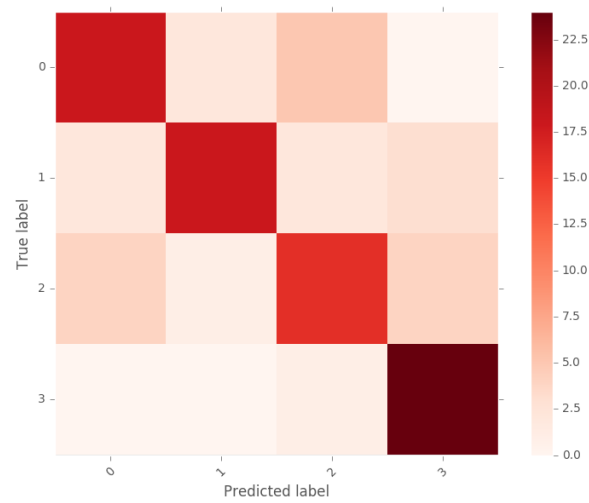


Figure 5.4: Confusion matrix for Resnet architecture

### 5.3 Damage Location Estimation

Both architectures were also tried in the damage location task. The task consists of classifying the images according to the approximate location of the damage exhibited. This task is again attempted with both the Resnet [HZRS15a] and GoogleNet [HZRS15a] architectures. A small ensemble was also tried with Google LeNet base models, in an attempt to further improve the results.

Stochastic Gradient Descent was also used to train all models. Random Cropping and Mirroring were also used in all models.

#### 5.3.1 Google LeNet

The Google LeNet architecture achieved 84% accuracy in this task. Other related metrics may be seen on Table 5.6. The confusion matrix is shown on Figure 5.5. This performance was achieved at around iteration 16000.

The data displayed here clearly shows that the model is capable of figuring out if damages are present or not. This shows that this task is not a mere pose estimation task. The model shows slightly worse performance on the classes labelled "Front Side" and "Back Side". This probably is due to an increased ambiguity that exists when classification in this part of the car happens. While the front of the car is hardly mistaken by its side, the boundary between what's considered "Front-Side" and "Side", for example, often depends on the vehicle shown. The slightly worse performance observed on the "Back-Side" class, when compared to others, probably is due to the lack of samples available for that types of damages.

##### 5.3.1.1 Ensemble

As expected, the ensemble performs slightly better than a single model. This behaviour was also observed in the previous task, whose results for the ensemble are presented on Section 5.2.1.1. The conclusions possible to derive from this experiment are also similar to the ones presented for the single model, in the previous section. The mark of 85% accuracy was the best possible to obtain with this ensemble.

#### 5.3.2 Resnet

The results obtained using the Resnet architecture were significantly better than the ones obtained with the Google LeNet architecture. A mark of 89% was achieved at iteration 16000. The confusion matrix is shown in Figure 5.7. It can be clearly seen that in the rare cases where the model misclassified the images, it classified them in an adjacent area, making it an acceptable error. Some of these errors might be due to damages where the location is not exactly clear. More parameters related to this model's performance may be seen in Table 5.8. From the table some conclusions may be drawn. It is clear that the classes that exhibit worse performance are the ones labelled

## Results and Outputs

Table 5.5: Performance metrics for the Resnet architecture

Category Name	Precision	Recall	F1-score	Support
No Damage	0.78	0.72	0.75	25%
Paint Damage	0.88	0.56	0.68	25%
Minor Damage	0.60	0.84	0.70	25%
Major Damage	0.88	0.92	0.90	25%
<b>Avg/Total</b>	<b>0.79</b>	<b>0.76</b>	<b>0.76</b>	<b>100%</b>

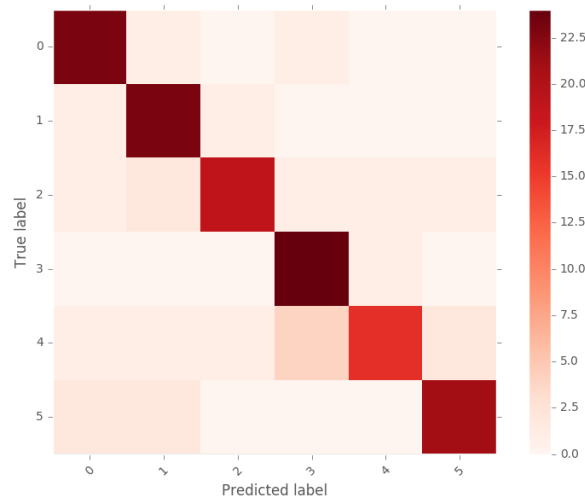


Figure 5.5: Confusion matrix for Google LeNet architecture

Table 5.6: Performance metrics for a single Google LeNet model

Category Name	Precision	Recall	F1-score	Support
No Damage	0.82	0.92	0.87	25%
Front Damage	0.79	0.92	0.85	25%
Front-Side Damage	0.90	0.76	0.83	25%
Side Damage	0.80	0.96	0.87	25%
Back-Side Damage	0.89	0.64	0.74	25%
Back Damage	0.88	0.84	0.86	25%
<b>Avg/Total</b>	<b>0.85</b>	<b>0.84</b>	<b>0.84</b>	<b>150%</b>

## Results and Outputs

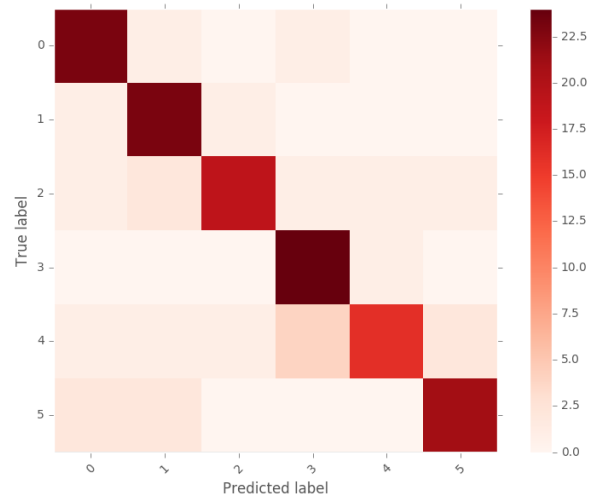


Figure 5.6: Confusion matrix for Google LeNet ensemble

Table 5.7: Performance metrics for an ensemble of three Google LeNet base models

Category Name	Precision	Recall	F1-score	Support
No Damage	0.84	0.84	0.84	25%
Front Damage	0.85	0.92	0.88	25%
Front-Side Damage	0.95	0.76	0.84	25%
Side Damage	0.86	0.96	0.91	25%
Back-Side Damage	0.73	0.76	0.75	25%
Back Damage	0.92	0.88	0.90	25%
<b>Avg/Total</b>	<b>0.86</b>	<b>0.85</b>	<b>0.85</b>	<b>150%</b>

"Front-Side" and "Back-Side". This might again be due to the fact that those locations' boundaries are not very clear in some vehicles, making it difficult to figure out which class to attribute to the image. Also, the "Back-Side" class exhibits a worse performance than the others. This might be due to the lack of samples featuring damages in this location.

### 5.4 Chapter Conclusions

It is clear that both architectures are capable of achieving very acceptable performances in both tasks. Ensembles were shown to slightly improve the performance of a single model of the same architecture. Under-represented classes often show poorer recall rates than the more common ones. The models, possibly aided by the balancing techniques employed were able to avoid great performance penalties stemming from the highly imbalanced datasets. The pre-trained models used were successfully fine-tuned to perform the tasks intended, showing that it is possible to achieve very interesting performances even with relatively small datasets. The datasets used should be considered noisy because they have been labeled using subjective criteria where the frontier between two different classes is not always clear. The models proved effective in dealing with such noisy datasets as these ones. The wide range of conditions in which the images were taken make present enormous challenges to computer vision systems. These obstacles were all successfully overcome by these models.

## Results and Outputs

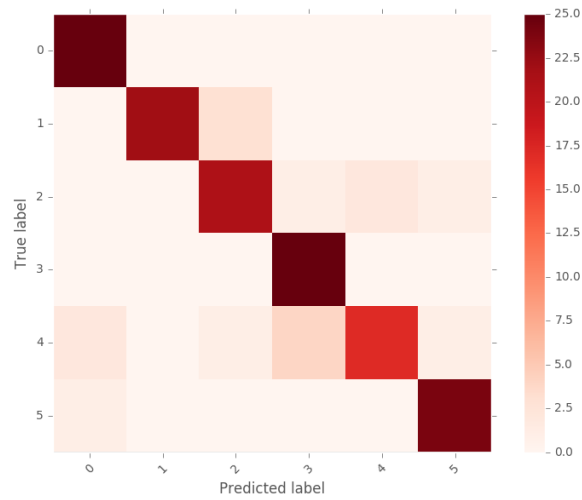


Figure 5.7: Confusion matrix for Google LeNet architecture

Table 5.8: Performance metrics for a single Resnet50 model

Category Name	Precision	Recall	F1-score	Support
No Damage	0.89	1.00	0.94	25%
Front Damage	1.00	0.88	0.94	25%
Front-Side Damage	0.84	0.84	0.84	25%
Side Damage	0.83	1.00	0.91	25%
Back-Side Damage	0.89	0.68	0.77	25%
Back Damage	0.92	0.96	0.94	25%
<b>Avg/Total</b>	<b>0.90</b>	<b>0.89</b>	<b>0.89</b>	<b>150%</b>

## Results and Outputs

## Chapter 6

# Conclusions and Future Work

In this section the conclusions derived from the work described in this document are presented. Some suggestions of future work is suggested.

### 6.1 Conclusions

Computer vision and, in particular, image classification are fields of study where major breakthroughs were achieved in the last few years. These breakthroughs came mainly, but not only, from the usage of Deep Convolutional Neural Networks. Some of these breakthroughs were used in this project. These breakthroughs were applied in damage severity and location estimation in vehicles. These same tasks, and others that depend on them, are nowadays performed manually, without any assistance from computer systems.

The results shown in this document lead to the conclusion that systems as the ones studied here, might be used to, at least help humans in these tasks.

The two somewhat related tasks were attempted with a distinct level of success, using different architectures and techniques.

The results, shown in Chapter 5 clearly show that DCNN can achieve very interesting accuracy marks on both tasks, even with small datasets and modest computational resources.

On the first task, the 76% accuracy mark achieved with a small ensemble, places this model in the same range of accuracy expected of untrained human beings. This is an encouraging mark obtained without a properly sized dataset available. With a more adequate dataset, results would likely be better and place this models on accuracy levels similar to trained human beings. Furthermore, bigger ensembles might bring even better results, but again, for that purpose, a better dataset would have to be gathered and more computational resources would be required. This mark of 76% may seem a little disappointing but a few remarks being should be made. The task is very subjective in the sense that it is not always clear, not even to humans, how to categorise

## Conclusions and Future Work

some of the images. While there are some images that clearly exhibit very mild or severe damages, there are some where the boundary between mild and severe damages blurs itself, and the classification is not clear. On the other hand, scratches, or peeled off paint images are difficult to find, and therefore the small number of available examples hurts the performances of the models. If not the first, at least the second problem is likely to be possibly mitigated by the usage of big datasets of images car insurance companies have access to. These big datasets alone might turn these modestly performing models used here in highly performant ones, perhaps reaching super human performance.

The second task is slightly different. The boundaries between the different classifications are clearer and therefore the expected results are also better than the ones expected for the first task. There isn't a total absence of ambiguity on the classification though. It is sometimes difficult to ascertain if the a damage is located on, for example, what's considered the side of the vehicle, or the "Front-Side" area instead. This is even more so, when we compare similar damages on cars that are very different from one another. Nevertheless the accuracy levels obtained, while still leaving room for improvement, clearly show that the models are totally capable of figuring out where the damages, if any, are located. The best accuracy mark of 89%, is an impressive mark for this task.

Results presented for both tasks ensure us that models like these ones are perfectly capable of distinguishing between different levels of damages and are also very capable of locating them. Being able to identify and locate damages is a crucial part of many other vehicle damage related tasks. It is therefore likely that models such as the ones used in this work might be successfully incorporated in systems capable of automatically estimating repair costs.

Scarcity of computational resources and time constraints made it impossible to further deepen the experiments shown in this document. The model's choice was often constrained by the scarce time available to train them, since bigger more powerful, more performant models are often slower to train.

## 6.2 Future Work

The work presented here merely lays ground for further, more detailed work in this field. The conclusions and results obtained allow us to be optimistic about the capabilities of these models in tasks related to car damage detection and classification. Further studies, relying on more computational power and bigger datasets might be desirable. This would make it possible to use even more powerful models to surpass human performance in related tasks. But the usage of more powerful models would be of no avail without access to a bigger, better dataset. Even with the relatively small models used here, a bigger, more diversified dataset, would likely allow the models to perform best.

This work also lays ground for the attempt of more complex tasks related to damage detection. These tasks might include repair cost estimation, determination of the driver at fault in car

## Conclusions and Future Work

accidents, automated damage conditions estimation, among other tasks performed around car accidents.

More rigorous estimation of human performance in these tasks might also be useful. What's an acceptable performance for a classification task often depends on the task itself. Therefore having a human performance baseline might be very useful in order to understand how good these models really are.

The usage of some alternative methods, possibly using these models only as feature extractors, might prove worthy. It often happens that DCNNs work very well with SVMs to achieve superior performance on image classification tasks.

## Conclusions and Future Work

# References

- [Bre96] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
- [caf] Caffe website. <http://caffe.berkeleyvision.org>. Accessed: 2017-02-25.
- [CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.
- [cnt] Cntk website. <https://github.com/Microsoft/CNTK/wiki>. Accessed: 2017-02-25.
- [Cou83] National Safety Council. Vehicle Damage Scale for Traffic Accident Investigators. Technical report, National Safety Council, 12 1983.
- [DDS<sup>+</sup>09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [dee] Deeplearning4j website. <https://deeplearning4j.org>. Accessed: 2017-02-25.
- [DPRGOK15] José F Díez-Pastor, Juan J Rodríguez, César I García-Osorio, and Ludmila I Kuncheva. Diversity techniques improve the performance of the best imbalance learning ensembles. *Information Sciences*, 325:98–117, 2015.
- [EVGW<sup>+</sup>10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [FSA99] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.
- [GGA<sup>+</sup>12] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and continuous pose estimation. *Image and Vision Computing*, 2012.
- [HBFS01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.
- [HS15] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5353–5360, 2015.

## REFERENCES

- [HZRS15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- [HZRS15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.
- [J<sup>+</sup>13] Srimal Jayawardena et al. Image based automatic vehicle damage detection. 2013.
- [ker] Keras website. <https://keras.io>. Accessed: 2017-02-25.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [LCC<sup>+</sup>14] Cewu Lu, Hao Chen, Qifeng Chen, Hei Law, Yao Xiao, and Chi-Keung Tang. 1-hkust: Object detection in ilsvrc 2014. *arXiv preprint arXiv:1409.6155*, 2014.
- [Low99] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- [MDRM15] Niall McLaughlin, Jesus Martinez Del Rincon, and Paul Miller. Data-augmentation for reducing dataset bias in person re-identification. In *Advanced Video and Signal Based Surveillance (AVSS), 2015 12th IEEE International Conference on*, pages 1–6. IEEE, 2015.
- [OLP09] M. Ozuysal, V. Lepetit, and P.Fua. Pose estimation for category specific multiview object localization. In *Conference on Computer Vision and Pattern Recognition*, Miami, FL, June 2009.
- [PBS15] Ronaldo C Prati, Gustavo EAPA Batista, and Diego F Silva. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, 45(1):247–270, 2015.
- [RDS<sup>+</sup>15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.
- [SLJ<sup>+</sup>15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- [SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [tf] Tensorflow website. <https://www.tensorflow.org>. Accessed: 2017-02-25.

## REFERENCES

- [The] Theano website. <http://deeplearning.net/software/theano/>. Accessed: 2017-02-25.
- [tor] Torch website. <http://torch.ch>. Accessed: 2017-02-25.
- [VGBP15] Phong D Vo, Alexandru Ginsca, Hervé Le Borgne, and Adrian Popescu. On deep representation learning from noisy web images. *arXiv preprint arXiv:1512.04785*, 2015.
- [YLCLT15] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3973–3981, 2015.
- [ZTHW12] Zhaoxiang Zhang, Tieniu Tan, Kaiqi Huang, and Yunhong Wang. Three-dimensional deformable-model-based localization and recognition of road vehicles. *Image Processing, IEEE Transactions on*, 21(1):1–13, 2012.