**AI BASED CAR INSURANCE CLAIM ANALYSIS**

# LEARNING ALGORITHMS

### A PROJECT REPORT

***Submitted by***

### BOORANI P (921319205016)

### DARSHANAA M (921319205018)

### JEYASHRI J (921319205050)

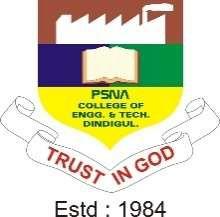
***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**



**PSNA COLLEGE OF ENGINEERING AND TECHNOLOGY**

(An Autonomous Institution Affiliated to Anna University, Chennai)

**DINDIGUL - 624622**

**APRIL 2023**

**PSNA COLLEGE OF ENGINEERING AND TECHNOLOGY**

(An Autonomous Institution Affiliated to Anna University, Chennai)

**DINDIGUL - 624622**

**BONAFIDE CERTIFICATE**

Certified that the project report **“AI Based Car Insurance Claim Analysis”** is a bonafide work of “**BOORANI P (921319205016), DARSHANAA M (921319205018), JEYASHRI J (921319205050**)**”** who carried out the project under my supervision.

**SIGNATURE                                                           SIGNATURE**

Dr .A.Vincent Antony Kumar.,M.E,Ph.D                 Dr.M.Anandaraj M.E.,Ph.D.

**HEAD OF THE DEPARTMENT                       SUPERVISOR**

Department of Information                                       Department of Information Technology                                             Technology

PSNA College of Engineering and                           PSNA College of Engineering and

Technology Dindigul-624622                                  TechnologyDindigul-624622

Submitted for the project viva voce examination held on -2023

**INTERNAL EXAMINER**                                         **EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

With warm hearts and immense pleasure, we thank the almighty for his grace and blessings which drove us to the successful completion of the project. We would like to express our gratitude towards our parents for their kind cooperation and encouragement which helped us in completion of this project.

We take this opportunity to express our sincere thanks to the respected Chairperson **Tmt.K. DHANALAKSHMI AMMAL,** who is the guiding light for all the activities in our college. We would like to express deep gratitude to our Pro Chairman **Rtn.Thiru R.S.K RAGURAAM D.A.E, M.COM** for his continuous support towards the development of the students.

We would like to thank our Principal **Dr.D. VASUDEVAN M.E, Ph.D.** for being a beacon in guiding every one of us and infusing us the strength and enthusiasm to work over successfully.

We express our sincere thanks and heartfelt gratitude to our respectful **Dr.A. VINCENT ANTONY KUMAR M.E., Ph.D.,** Professor and Head, Department of Information Technology for his valuable suggestions and continuous encouragement in the completion of the project work.

 This project would not have been possible without the motivation and guidance of our Project Guide **Dr.M.Anandaraj M.E,Ph.D.** Professor of the Department

# 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.

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.

In order to be able to develop such system, a suitable dataset was gathered. 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

# Table of Contents

**Chapter Title Page**

**Abstract i**

**List of Figures iv**

**List of Tables v**

**List of Abbreviation** **vi**

1. Introduction [1](#_bookmark0)

1.1 Context . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . [2](#_bookmark1)

1.2 Motivation and Objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . [2](#_bookmark2)

1.3 Dissertation Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . [3](#_bookmark3)

1. Related Work [5](#_bookmark4)

2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . [5](#_bookmark5)

2.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . [5](#_bookmark6)

2.2.1 Image based car damage detection . . . . . . . . . . . . . . . . . . . . . [5](#_bookmark7)

2.2.2 Image Classification . . . . . . . . . . . . . . . . . . . . . . . . . . . . [6](#_bookmark8)

2.2.3 Available Datasets [10](#_bookmark14)

* 1. Existing Technologies [11](#_bookmark15)
  2. Conclusions [12](#_bookmark17)

1. Automatic detection of damages in cars [13](#_bookmark18)
   1. Problem Definition [14](#_bookmark21)
      1. Damage Severity Detection [15](#_bookmark22)
      2. Damage Location Estimation [15](#_bookmark24)
   2. Proposed Solution [16](#_bookmark26)
      1. Distinguishing different types of damages [16](#_bookmark27)
      2. Locating the damages [16](#_bookmark28)
   3. Conclusions [17](#_bookmark31)
2. Implementation [19](#_bookmark32)
   1. Architectures [19](#_bookmark33)
      1. Google LeNet [19](#_bookmark34)
      2. Residual Network [20](#_bookmark36)
   2. Changes in Topology [21](#_bookmark38)
   3. Training Parameters [22](#_bookmark40)
   4. Data Augmentation [22](#_bookmark41)
      1. Mirroring [22](#_bookmark42)
      2. Random Cropping [23](#_bookmark43)
   5. Ensemble Methods [23](#_bookmark44)
      1. Boosting [23](#_bookmark45)
      2. Bagging [24](#_bookmark46)
   6. Dataset Imbalances [24](#_bookmark47)
      1. Random Oversampling [24](#_bookmark48)
      2. Random Under-sampling [25](#_bookmark49)
      3. Data Augmentation [25](#_bookmark50)
      4. Impact on performance [25](#_bookmark51)
   7. Implementation Details Summary [26](#_bookmark52)
3. Results and Outputs [27](#_bookmark53)
   1. Dataset [27](#_bookmark54)
   2. Damage Severity Detection [28](#_bookmark55)
      1. Google LeNet [28](#_bookmark56)
      2. Resnet [31](#_bookmark63)
      3. Human Performance [31](#_bookmark64)
   3. Damage Location Estimation [33](#_bookmark68)
      1. Google LeNet [33](#_bookmark69)
      2. Resnet [33](#_bookmark70)
   4. Chapter Conclusions [36](#_bookmark76)
4. Conclusions and Future Work [39](#_bookmark79)
   1. Conclusions [39](#_bookmark80)
   2. Future Work [40](#_bookmark81)

Appendix

A.Coding 42

B.Snapshot 43

References [44](#_bookmark82)

**List of Figures**

* 1. Evolution of error of the top performing system on two ILSVRC tasks [[RDS+15](#_bookmark106)] [8](#_bookmark11)
  2. Example of a section of a CNN learning a residual function *F*(*x*) instead of the function *O*(*x*) . . . . . . . . . . . . . . . . .. . …….[9](#_bookmark13)
  3. Image depicting a situation where the dent visible [13](#_bookmark19)
  4. Another image depicting a situation visible . [14](#_bookmark20)
  5. System for distinguishing different kinds of damages [17](#_bookmark29)
  6. System for distinguishing different kinds of damages [17](#_bookmark30)
  7. Inception block basic components [[LCC+14](#_bookmark101)] [20](#_bookmark35)
  8. Google LeNet [[LCC+14](#_bookmark101)] basic topology [21](#_bookmark37)
  9. Residual Network 152 layer [[HZRS15a](#_bookmark96)] basic topology [22](#_bookmark39)
  10. 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](#_bookmark60)

* 1. Confusion matrix for Google LeNet architecture [30](#_bookmark61)
  2. Confusion matrix for Google LeNet base models [32](#_bookmark65)
  3. Confusion matrix for Resnet architecture [32](#_bookmark67)
  4. Confusion matrix for Google LeNet architecture [34](#_bookmark72)
  5. Confusion matrix for Google LeNet ensemble [35](#_bookmark74)
  6. Confusion matrix for Google LeNet architecture [37](#_bookmark77)

# List of Tables

2.1 Main characteristic of some frameworks with support for CNN [12](#_bookmark16)

* 1. Damage severity scale [15](#_bookmark23)
  2. Damage location categories to be used [16](#_bookmark25)
  3. Dataset distribution across damage severity scale [29](#_bookmark58)
  4. Dataset distribution across damage location scale [29](#_bookmark59)
  5. Performance metrics [30](#_bookmark62)
  6. Performance metrics of 3 Google LeNet base models [32](#_bookmark66)
  7. Performance metrics for the Resnet architecture [34](#_bookmark71)
  8. Performance metrics for a single Google LeNet model [34](#_bookmark73)
  9. Performance metrics for an ensemble of base models [35](#_bookmark75)
  10. Performance metrics for a single Resnet50 model [37](#_bookmark78)

**Table of Abbreviation**

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 it’s 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.

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, favoring 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.

## 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.

## 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.

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 tough cars are very important to many people, they have been somehow neglected by the computer vision research community [[YLCLT15](#_bookmark114)].

Cars are very challenging to work with in computer vision. They present great variation in shape and form by slight variations of viewpoint [[YLCLT15](#_bookmark114)]. 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 of detecting damages in car images a very chal- lenging 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+13](#_bookmark98)]. 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+13](#_bookmark98)]. 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 tech- nology has endured tremendous changes in the last decade. We now have image classification systems that can allegedly outperform human beings [[HZRS15b](#_bookmark97)] in some tasks, a feat that could not be achieved by decades long research with traditional approaches.

## Dissertation Structure

In Chapter [2](#_bookmark4), 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](#_bookmark18), 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](#_bookmark32), implementation specific details are presented. Some

possible implementation related difficulties and constraints are also detailed here. In Chapter [5](#_bookmark53), the results and outputs obtained will be discussed and detailed. In Chapter [6](#_bookmark79), general conclusions are drawn and some hypothesis for possible future work are presented.

**Chapter 2**

# Related Work

## 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+09](#_bookmark88)] or Pascal [[EVGW+10](#_bookmark91)] and therefore, performance on those datasets will be used as benchmark for discussed methods when no further information is available.

## Related Work

### Image based car damage detection

The research found that addresses automatic damage detection in cars uses 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+13](#_bookmark98)]. Segmentation algorithms are then used to identify different areas of the car [[J+13](#_bookmark98)]. Edges are then extracted and the ones that are not predicted from the ground truth information are further processed [[J+13](#_bookmark98)]. This task alone of reflection detection, yields an MCC of no more than 0.34 [[J+13](#_bookmark98)]. 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+13](#_bookmark98)].

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+13](#_bookmark98)]. 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.

### Image Classification

Image classification is a research area where important progresses have been made in the last few years. Figure [2.1](#_bookmark11) shows the reported error of the winning teams on two major tasks on ILSVRC [[RDS+15](#_bookmark106)], an important computer vision competition.

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+09](#_bookmark88)] or PASCAL [[EVGW+10](#_bookmark91)]. ILSVRC [[RDS+15](#_bookmark106)] 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 an unconstrained problems like detecting damages in images of cars.

#### Traditional Approach

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.

Feature extractors and detectors are algorithms, of which SIFT [[Low99](#_bookmark102)] 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

Systems such as these were used for many years in real-world applications and important image classification contests such as Imagenet [[DDS+09](#_bookmark88)] whose top performing systems’ results can be seen on Figure [2.1](#_bookmark11). The teams whose results are shown in Figure [2.1](#_bookmark11) 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.

#### 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.

Since the introduction of the first CNNs in the ILSVRC [[RDS+15](#_bookmark106)] in 2012, all the winning teams for the classification task have used different types of CNNs [[KSH12](#_bookmark100), [LCC+14](#_bookmark101), [HZRS15a](#_bookmark96)]. CNNs systems now clearly outperform other image classification systems such as the ones de- scribed in Section [2.2.2.1](#_bookmark9). State of the art CNNs have even crossed the boundary of what is con- sidered to be the error rate for human beings in image classification tasks [[HZRS15b](#_bookmark97)].

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

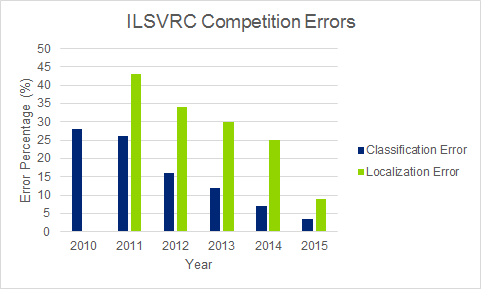


Figure 2.1: Evolution of error of the top performing system on two ILSVRC tasks [[RDS+15](#_bookmark106)]

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.

ReLUs or Rectified Linear Units, are also widely used in CNNs. ReLUs are units that apply a function similar to Equation [2.1](#_bookmark12). There are often alternatives for these types of units, Equation [2.2](#_bookmark12) and [2.3](#_bookmark12) show two of them.

|  |  |  |  |
| --- | --- | --- | --- |
| *ReLU* : | *f* (*x*) | = *max*(0*, x*) | (2.1) |
| *Tanh* : | *f* (*x*) | = *tanh*(*x*) | (2.2) |
| *Sigm* : | *f* (*x*) | = (1 + *e−x*)*−*1 | (2.3) |

Fully Connected Layers are used too, as in more standard NNs.

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](#_bookmark100)].

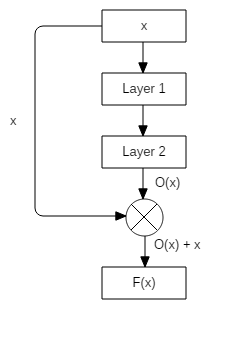


Figure 2.2: Example of a section of a CNN learning a residual function *F*(*x*) instead of the function

*O*(*x*)

Evidence shows that one way of improving a CNN performance is by stacking more layers and making the network effectively deeper [[SLJ+15](#_bookmark108), [SZ14](#_bookmark109)]. The problem that often arises is that as more layers are stacked, the performance saturates and eventually starts to decrease at some point [[SGS15](#_bookmark107), [HS15](#_bookmark95)]. To address this issue, the idea of deep residual networks has been proposed [[HZRS15a](#_bookmark96)]. 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](#_bookmark13). Evidence shows that CNNs train better and converge faster when the approximated function is in fact *F*(*x*), a residual function [[HZRS15a](#_bookmark96)]. This approach opens door for usage of even deeper and more powerful models.

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.

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

#### Image Classification on Cars

The most recent body of work in image classification aimed towards cars is also small [[YLCLT15](#_bookmark114), [ZTHW12](#_bookmark115)]. 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](#_bookmark114)], 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](#_bookmark114)] and pose estimation [[ZTHW12](#_bookmark115)] 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 orienta- tion of the photographed vehicle, ranges from 0.43 to 0.76 on 431 possible classes [[YLCLT15](#_bookmark114)]. 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](#_bookmark114)].

### Available Datasets

There are some publicly available datasets featuring cars [[YLCLT15](#_bookmark114), [OLP09](#_bookmark104), [GGA+12](#_bookmark93)]. 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.

## 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.

Torch [[tor](#_bookmark112)] is another option. It supports CNNs and CUDA. It has it’s 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](#_bookmark89)] 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.

Tensorflow [[tf](#_bookmark110)] 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](#_bookmark99)] is a framework that relies on a Theano or Tensorflow backend. It exposes a sligtly friendlier interface API that Theano or Tensorflow, and also offers a slightly better support for pretrained models. It lacks a CLI.

CNTK [[cnt](#_bookmark86)] 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. 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](#_bookmark16)

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.

## 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 perfor- mance in object recognition [[HZRS15b](#_bookmark97)] and recent developments allow it to take more advantage of deeper and more powerful architectures [[HZRS15a](#_bookmark96)]. The performances exhibited by modern CNNs [[RDS+15](#_bookmark106)] make it possible to apply them to problems in computer vision that were pre- viously 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.

**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 chal- lenges in computer vision. That is especially the case if the photographed objects exhibit highly reflective surfaces like cars do [[YLCLT15](#_bookmark114)]. 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](#_bookmark19) and [3.2](#_bookmark20). 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



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 lo- cate damages in uncontrolled images, taken in uncontrolled lightning conditions, viewpoints and distances.

## Problem Definition

The goal of this dissertation is to develop a system capable of automatically identifying and locat- ing 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.

A slight modification of framework proposed by the NSC will be used [[Cou83](#_bookmark87)], adapting it to the necessities of the system to be developed here.

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.

### 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](#_bookmark23) 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.

### 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.

In terms of damage location, the NSC framework will also be adapted. The different dam- age location categories to be used are shown on Table [3.2](#_bookmark25). The damage locations correspond to a simplification of the categorisation proposed by the NSC report [[Cou83](#_bookmark87)] 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

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.

## Proposed Solution

The two problems will be solved with the use of CNNs. The winner submission of the ILSVRC 2015 challenge [[HZRS15a](#_bookmark96)] (Residual Network) and the winner submission of the ILSVRC 2014 challenge [[LCC+14](#_bookmark101)] (Google LeNet) will be used to solve the two problems mentioned on Section

[3.1](#_bookmark21). The Residual Network was chosen because it is the current state of the art in image classification technology.

### Distinguishing different types of damages

Distinguishing between different kinds of damages is, as detailed on Section [3.1](#_bookmark21), 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](#_bookmark29). Again, both architectures will be compared here, both pre-trained on the Imagenet dataset [[DDS+09](#_bookmark88)] and fine-tuned on a 3000 image dataset featuring both damaged and undamaged cars.

### 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

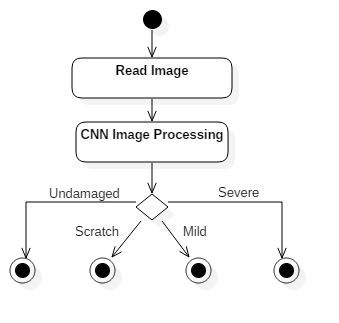


Figure 3.3: System for distinguishing different kinds of damages

in the car is shown on Figure [3.4](#_bookmark30). The system used will be trained as the ones in the previous problems.

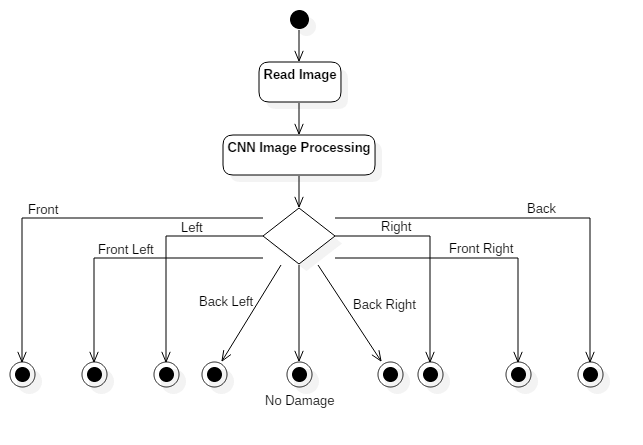


Figure 3.4: System for distinguishing different kinds of damages

## 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](#_bookmark96)] is the current state of the art in terms of reducing the error percentage in image classification tasks, the Google LeNet architecture [[LCC+14](#_bookmark101)] still is the fastest network to train with an error rate that resembles the state of the art performance.

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](#_bookmark18), 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](#_bookmark18). The system will be developed using the Caffe framework and pre-trained models will be used whenever possible as said on Section [2.3](#_bookmark15). For both tasks, more than one architecture will be used, in order to measure the performance of different architectures.

## Architectures

Two architectures are of great interest to this project. The Residual Net architecture [[HZRS15a](#_bookmark96)], 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+14](#_bookmark101)], that has a slightly larger error rate on the Imagenet [[DDS+09](#_bookmark88)] 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.

### 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](#_bookmark35).

Figure [4.2](#_bookmark37) for the sake of simplicity. In order to adapt this topology to the various tasks, The Inner Product layer, which is in fact a fully connected layer, whose purpose was detailed on

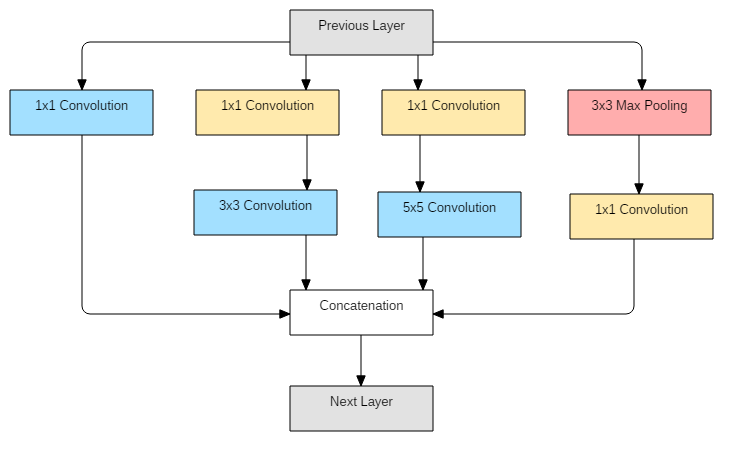


Figure 4.1: Inception block basic components [[LCC+14](#_bookmark101)]

Section [2.2.2.2](#_bookmark10), 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+15](#_bookmark106)] 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+14](#_bookmark101)], 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+14](#_bookmark101)]. 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.

### Residual Network

The Residual Network architecture, pioneered in the 2015 edition of the ILSVRC [[HZRS15a](#_bookmark96)], features the usage of residual blocks shown on Figure [2.2](#_bookmark13) and whose original topology is shown on Figure [4.3](#_bookmark39). The rationale behind residual blocks is explained in Section [2.2.2.2](#_bookmark10).



Figure 4.2: Google LeNet [[LCC+14](#_bookmark101)] basic topology

## Changes in Topology

Both detailed architectures were conceived with the Imagenet dataset [[DDS+09](#_bookmark88)] 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.

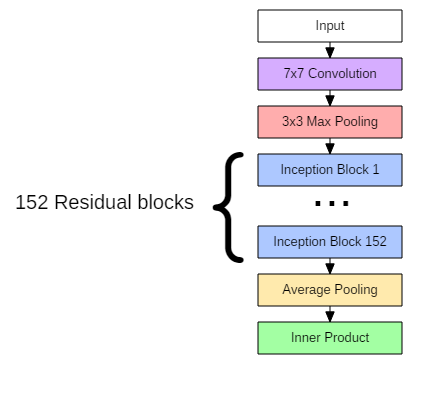


Figure 4.3: Residual Network 152 layer [[HZRS15a](#_bookmark96)] basic topology

Discarding the possibility of using pre-trained models would probably then be a bad decision.

## Training Parameters

Both the Google LeNet [[LCC+14](#_bookmark101)] and the Residual Net [[HZRS15a](#_bookmark96)] 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.

## 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.

### Mirroring

Mirroring is a technique that is used very often [[MDRM15](#_bookmark103)]. This technique cannot be used when the task performed by the model involves some kind of location .

task mentioned on Section [3.1.2](#_bookmark24) 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.

### Random Cropping

Other very popular technique, random cropping [[MDRM15](#_bookmark103)] 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. 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.

## 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.

### Boosting

Boosting, sequential ensemble, is a meta-algorithm typically used to correct bias in the predictions of models. 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 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](#_bookmark92)], Gradient Tree Boosting and XGBoost[[CG16](#_bookmark85)].

### Bagging

Bagging [[Bre96](#_bookmark83)], 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](#_bookmark100)] 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 sub models 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.

## Dataset Imbalances

As mentioned on Section [2.2.3](#_bookmark14), 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.

### 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.

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](#_bookmark90)], 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.

### Random Under-sampling

Similar in concept to the method presented in Section [4.6.1](#_bookmark48). 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 generalize 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.

### 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.

### Impact on performance

The impact of dataset imbalances on the overall system performances usually doesn’t exceed 5%, as reported in [[PBS15](#_bookmark105)], especially if the imbalance is not severe. SVM based classification systems are typically less prone to be affected by class imbalances [[PBS15](#_bookmark105)]. Tree based models were not mentioned in this discussion because they usually exhibit worse performance than SVMs or NNs in image classification tasks.

## Implementation Details Summary

Implementation wise, the modifications that are necessary to make to the original CNNs’ architec- ture 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 pa- rameters 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](#_bookmark114)] 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](#_bookmark47) 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](#_bookmark32), with the required modifications and additions, employing some of the techniques already mentioned, according to previously discussed criterions, are then presented and explained.

## Dataset

The dataset required for the system to be trained is composed of 15671 images labelled according to damage severity and 10016 labelled according the place of the damage.

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 bal- anced, with some under represented classes where damages are less prone to exist. This may be seen in Table [5.2](#_bookmark59).

These imbalances reinforce the need to use data balancing techniques mentioned and detailed on Section [4.6](#_bookmark47). The techniques employed where mostly Random Oversampling, described in Section [4.6.1](#_bookmark48). 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.

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.

## Damage Severity Detection

Both architectures, Google LeNet [[LCC+14](#_bookmark101)] and Resnet [[HZRS15a](#_bookmark96)] were tried on this problem. Google LeNet performed, as expected, slightly worse than the Resnet architecture, scoring a max- imum 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.

### Google LeNet

Google LeNet maximum score of 75% was achieved at iteration 1500. The confusion matrix may be seen in Figure [5.2](#_bookmark61). Some performance metrics such as precision, recall rate and f1-score are shown on Table [5.3](#_bookmark62).

A these numbers and matrix clearly show, the number of times the model classifies an undam- aged or scratched vehicle as having heavy damages is quite small. It is very difficult do detect damages that are not very evident, even to humans.

#### Ensemble

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](#_bookmark46). 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.

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% |
| **Total** | **15671** | **100**% |

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% |
| **Total** | **10016** | **100**% |

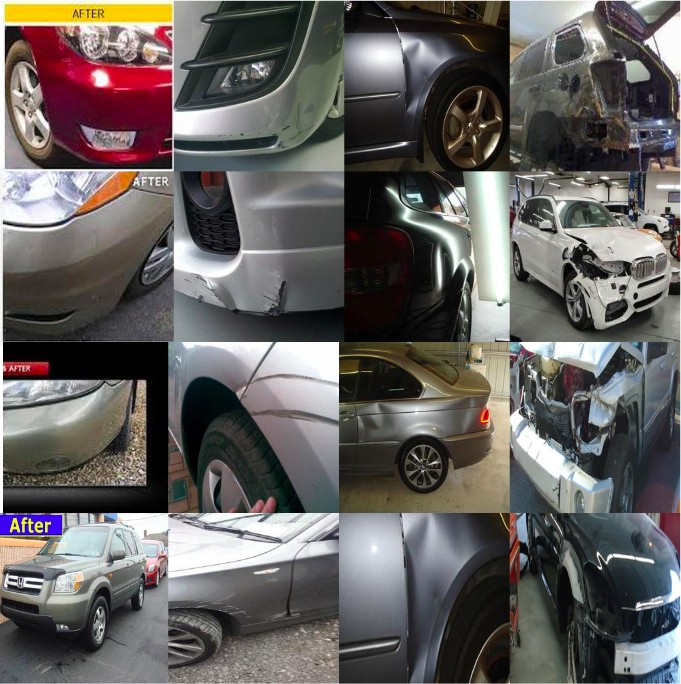


Figure 5.1: Some examples from the dataset. The first, second, third and fourth columns feature examples of undamaged cars, cars with scratches, cars with mild damages and cars with severe damages, respectively

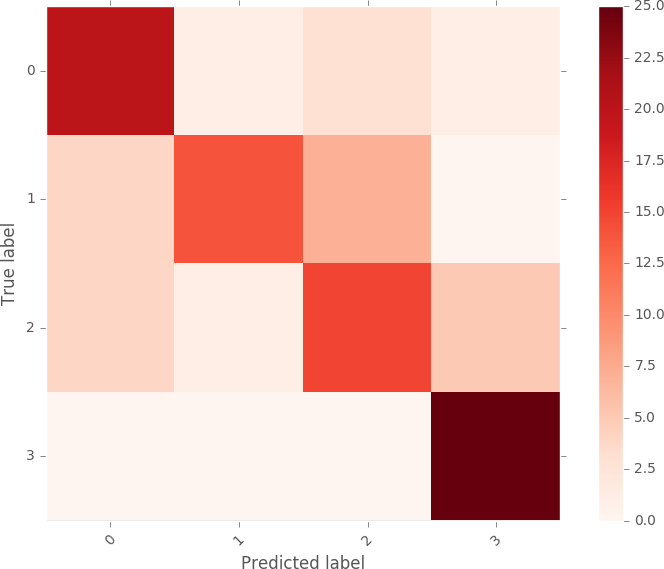


Figure 5.2: Confusion matrix for Google LeNet architecture

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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% |
| **Avg/Total** | **0.76** | **0.75** | **0.75** | **100**% |

Table 5.3: Performance metrics

The output of the mentioned ensemble, whose confusion matrix is shown in Figure [5.3](#_bookmark65), 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](#_bookmark18). 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 it’s 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](#_bookmark46). Table [5.4](#_bookmark66) 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 it’s performance, especially the "Paint Damage" class, that saw it’s recall rate raised from 52% to 72%.

### 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](#_bookmark67) and Table [5.5](#_bookmark71) show, the marks and error distribution are quite similar to the ones observed for the other architecture.

### 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

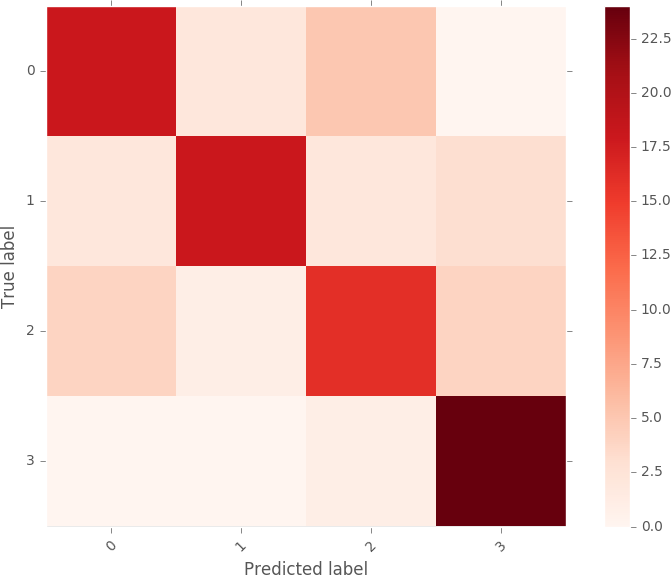


Figure 5.3: Confusion matrix for Google LeNet architecture ensemble of 3 base models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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% |
| **Avg/Total** | **0.76** | **0.76** | **0.76** | **100**% |

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

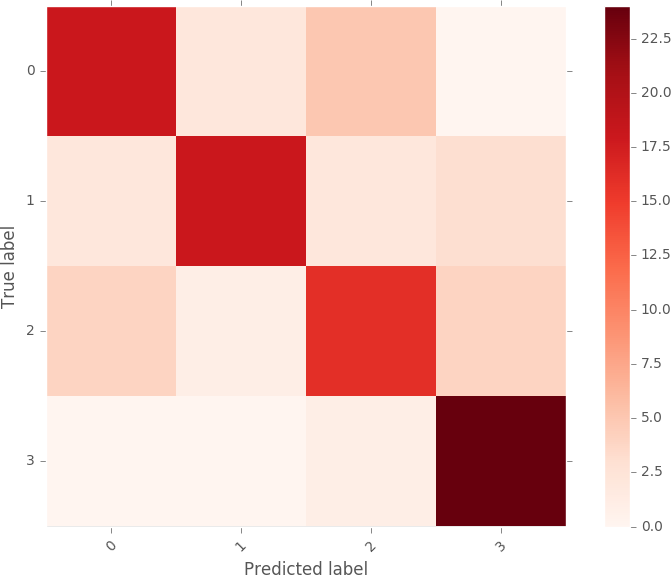


Figure 5.4: Confusion matrix for Resnet architecture

## 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](#_bookmark96)] and GoogleNet [[HZRS15a](#_bookmark96)] 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 Mirror- ing were also used in all models.

### Google LeNet

The Google LeNet architecture achieved 84% accuracy in this task. Other related metrics may be seen on Table [5.6](#_bookmark73). The confusion matrix is shown on Figure [5.5](#_bookmark72). This performance was achieved at around iteration 16000.

#### 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](#_bookmark57). 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.

### 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 confu- sion matrix is shown in Figure [5.7](#_bookmark77). 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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% |
| **Avg/Total** | **0.79** | **0.76** | **0.76** | **100**% |

Table 5.5: Performance metrics for the Resnet architecture

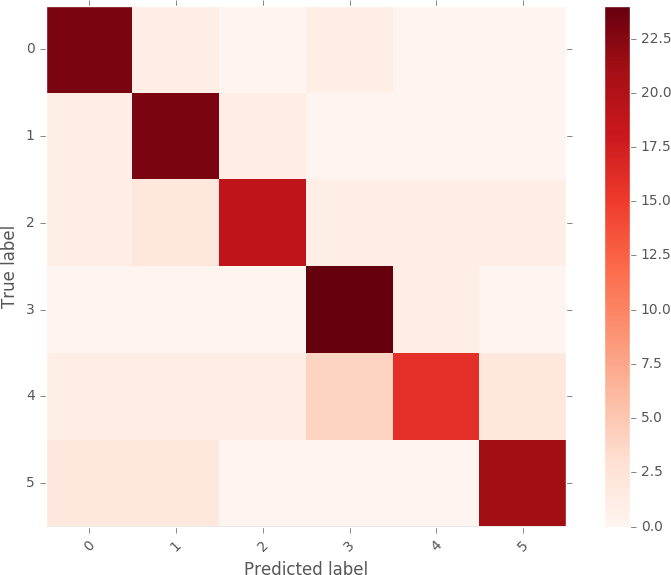


Figure 5.5: Confusion matrix for Google LeNet architecture

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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% |
| **Avg/Total** | **0.85** | **0.84** | **0.84** | **150**% |

Table 5.6: Performance metrics for a single Google LeNet model

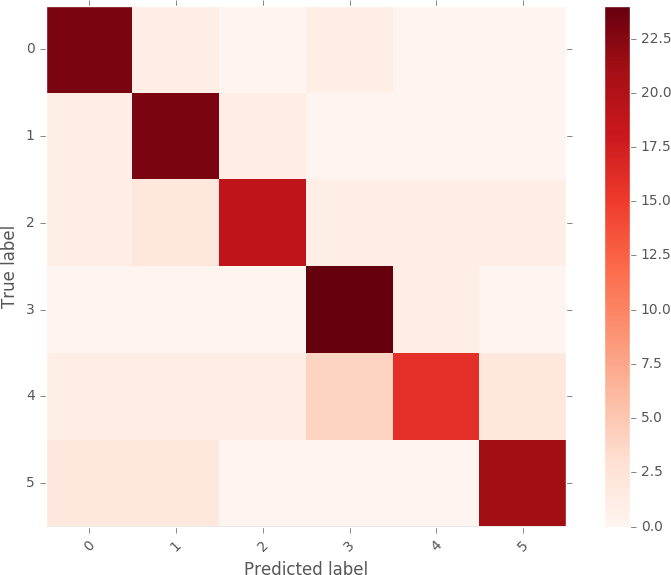


Figure 5.6: Confusion matrix for Google LeNet ensemble

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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% |
| **Avg/Total** | **0.86** | **0.85** | **0.85** | **150**% |

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

"Front-Side" and "Back-Side". This might again be due to the fact that those locations’ bound- aries 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.

## 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 ar- chitecture. 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 perfor- mance 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 con- sidered noisy because they has been labeled using subjective crietia where the frontier between two different classes is not always clear. The models proved effective in dealing with such noisy datasets as this ones. The wide range of condition in which the images were taken make present enormous challenges to computer vision systems. These obstacles were all successfully overcome by these models.

In this research proposal, a neural network-based solution for automobile detection will be used to address the issues of automotive damage analysis and position and severity prediction. This project does several tasks in one bundle. The method will unquestionably assist the insurance firms in conducting far more thorough and systematic analyses of the vehicle damage. Simply sending the system a photograph of the vehicle, it will evaluate it and determine whether there is damage of any type, where it is located, and how severe it is.

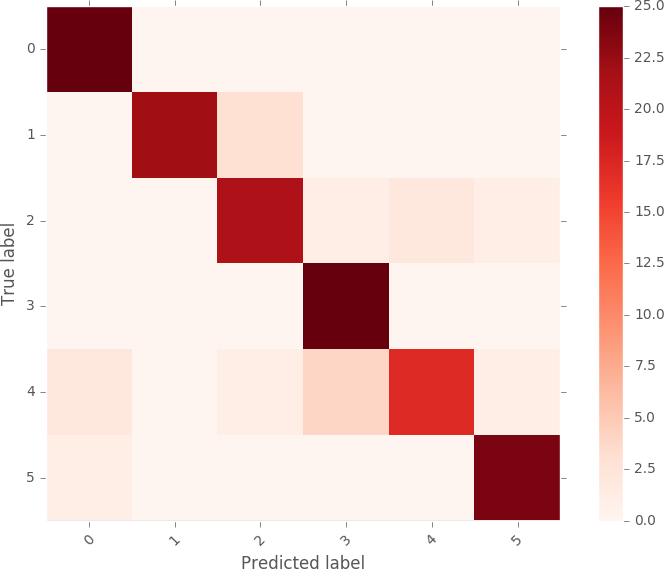


Figure 5.7: Confusion matrix for Google LeNet architecture

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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% |
| **Avg/Total** | **0.90** | **0.89** | **0.89** | **150**% |

Table 5.8: Performance metrics for a single Resnet50 model

In conclusion, a car damage analysis system using image processing and machine learning techniques has the potential to improve the accuracy and speed of damage assessment for vehicles. By automating the process of damage analysis, the system can provide more consistent and objective results, reducing the potential for human error and bias.

The system can have a wide range of applications in the insurance, automotive repair, law enforcement, and automotive manufacturing industries. It can help insurance companies to provide more accurate and efficient claims processing, repair shops to provide more accurate estimates and perform repairs more efficiently, law enforcement agencies to improve accident investigation and traffic safety, and automotive manufacturers to improve quality control and reduce defects.

The development of a car damage analysis system using image processing and machine learning techniques requires expertise in various fields, including computer vision, machine learning, and software development. However, with the right skills and resources, such a system can provide significant benefits and improve the efficiency and accuracy of damage assessment .

**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.

## Conclusions

Computer vision and, in particular, image classification are fields of study where major break- throughs 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.

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. Fur- thermore, 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

some of the images. While there are some images that clearly exhibit very mild or severe dam- ages, 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.

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.

## 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 mod- els 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 rel- atively 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 detec- tion. 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 ac- cidents.

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.

**Appendix**

**A . Coding**

from flask import Flask, render\_template, flash, request,session from cloudant.client import Cloudant import cv2 client = Cloudant.iam("eb55a2b7-ae45-4df8-8d1c-69c5229ffdbebluemix","YzG5FZg9Vs\_HScOBZaWyVXm7PpNjbPrmPaPMfHx7w3X9",connect= True) my\_database = client.create\_database("database-darshan") app = Flask(\_\_name\_\_) app.config.from\_object(\_\_name\_\_) app.config['SECRET\_KEY'] = '7d441f27d441f27567d441f2b6176a' @app.route("/") def homepage(): return render\_template('index.html') @app.route("/userhome") def userhome(): return render\_template('userhome.html') @app.route("/addamount") @app.route("/NewUser") def NewUser(): return render\_template('NewUser.html') @app.route("/user") def user(): return render\_template('user.html') @app.route("/newuse",methods=['GET','POST']) def newuse(): if request.method == 'POST': x = [x for x in request.form.values()] print(x) data = { '\_id': x[1], 'name': x[0], 'psw': x[2] } print(data) query = {'\_id': {'Seq': data['\_id']}} docs = my\_database.get\_query\_result(query)

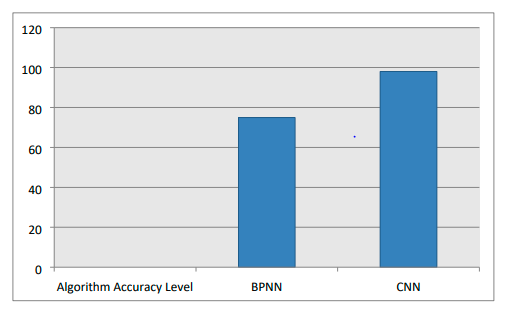
# warnings warnings.filterwarnings('ignore') import tensorflow as tf classifierLoad = tf.keras.models.load\_model('level.h5') import numpy as np from keras.preprocessing import image test\_image = image.load\_img('static/Out/Test1.jpg', target\_size=(200, 200)) img1 = cv2.imread('static/Out/Test1.jpg') # test\_image = image.img\_to\_array(test\_image) test\_image = np.expand\_dims(test\_image, axis=0)

# result = classifierLoad.predict(test\_image) result2 = '' if result[0][0] == 1: result2 = "minor" elif result[0][1] == 1: result2 = "moderate" elif result[0][2] == 1: result2 = "severe" if (result1 == "front" and result2 == "minor"): value = "3000 - 5000 INR" elif (result1 == "front" and result2 == "moderate"): value = "6000 8000 INR" elif (result1 == "front" and result2 == "severe"): value = "9000 11000 INR" elif (result1 == "rear" and result2 == "minor"):

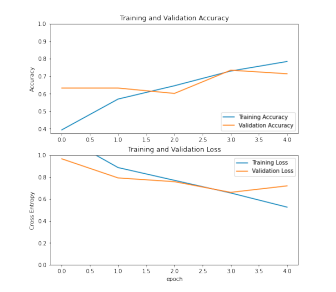
# value = "4000 - 6000 INR" elif (result1 == "rear" and result2 == "moderate"): value = "7000 9000 INR" elif (result1 == "rear" and result2 == "severe"): value = "11000 - 13000 INR" elif (result1 == "side" and result2 == "minor"): value = "6000 - 8000 INR" elif (result1 == "side" and result2 == "moderate"): value = "9000 - 11000 INR" elif (result1 == "side" and result2 == "severe"): value = "12000 - 15000 INR" else: value = "16000 - 50000 INR" return render\_template('userhome.html', prediction=value) if \_\_name\_\_ == '\_\_main\_\_': app.run(debug=True,use\_reloader=True)

**B. SNAPSHOT**

**B.1 PERFORMANCE METRICS**

****

# B.2 DAMAGE SEVERITY MODEL

****

# References

[1] Leo Breiman. Bagging predictors. *Machine learning*, 24(2).

[2] 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.

[3] Cntkwebsite. <https://github.com/Microsoft/CNTK/wiki>. Accessed: 2017-02-25.

[4] National Safety Council. Vehicle Damage Scale for Traffic Accident Inverstigators. Technical report, National Safety Council, 12 1983.

[5] 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.

[6] Deeplearning4jwebsite. [https://deeplearning4j.org](https://deeplearning4j.org/). Accessed: 2017- 02-25.

[7] 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.

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and An- drew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rec- tifiers: Surpassing human-level performance on imagenet classification. In *Pro- ceedings of the IEEE International Conference on Computer Vision*, pages 1026– 1034, 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information pro- cessing systems*, pages 1097–1105, 2012.

[12] 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.

[13] David G Lowe. Object recognition from local scale-invariant features. In *Com- puter vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[14] 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 Con- ference on*, pages 1–6. IEEE, 2015.

[15] Ronaldo C Prati, Gustavo EAPA Batista, and Diego F Silva. Class imbalance re- visited: a new experimental setup to assess

[16] Tensorflow website.[https://www.tensorflow.org](https://www.tensorflow.org/). Accessed: 2017-02- 25.

REFERENCES

[17] Theano website. <http://deeplearning.net/software/theano/>. Ac- cessed: 2017-02-25.

[18] Torch website.

[http://torch.ch](http://torch.ch/). Accessed: 2017-02-25.

[19] 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.

[20] 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.

[21] Zhaoxiang Zhang, Tieniu Tan, Kaiqi Huang, and Yunhong Wang. Three- dimensional deformable-model-based localization and recognition of road vehi- cles. *Image Processing, IEEE Transactions on*, 21(1):1–13, 2012.