MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Software Engineering

Mixed Intelligence for Lost Property Offices

By: Paul Puntschart, BSc

Student Number: 1710299039

Supervisors: FH-Prof. Dipl.-Ing. Dr. Robert Pucher
Dipl.-Ing. Dr. Christian Kenngott

Vienna, May 3, 2019

FH University of Applied Sciences TECHNIKUM WIEN

Declaration

Vienna, May 3, 2019 Signature

Kurzfassung

TensorFlow ist eine Open-Source-Softwarebibliothek für numerische Hochleistungsberechnungen und bietet starke Unterstützung für maschinelles Lernen und künstliche neuronale Netzwerke. In dieser Studie wurde TensorFlow zur Objekterkennung und Bildklassifizierung genutzt. Ein spezielles künstliches neuronales Netzwerk, das Faltung unterstützt, wurde trainiert.

Das Ergebnis dieser Studie ist eine Software-Bibliothek zur unterstützenden Verwaltung von Fundbüros, die Aufbewahrungszeiten verringern und die Rückgabequote von Fundgegenständen erhöhen soll. Für digitale Bilder von Fundgegenständen können diese erkannt und nach Gegenstandskategorie bestimmt werden. Weiters werden die dominanten Farben als Hauptfarbe und Nebenfarbe ermittelt. Im letzten Schritt werden die ermittelten Informationen als Beschreibungstext zurückgegeben.

Ein selbst erstellter Trainingsdatensatz von 1736 hochauflösenden Bildern wurde verwendet, um ein künstliches neuronales Netzwerk zu trainieren. Insgesamt wurden 7 verschiedenen Gegenstandskategorien verwendet. Diese Bilder wurden unter Laborbedingungen erstellt und bezüglich der Invarianz in Bezug auf Winkel, Abstand, Position, Rotation und Beleuchtung optimiert. Ziel dieser Studie war es, eine Genauigkeit von 90 Prozent bei der Gegenstandsklassifizierung zu erreichen. Dieses Ziel wurde erfolgreich erfüllt. Darüber hinaus kann die Anwendung Gegenstandsfarben beschreiben und unterstützt die Einführung neuer Gegenstandskategorien durch die Verwendung einer Strategie namens Mixed Intelligence.

Die vorliegende Studie soll dabei helfen die Qualität von Gegenstandsbeschreibungen zu optimieren.

Schlagworte: Bildklassifizierung, künstliche Gegenstandsbeschreibung, Bilderkennung, künstliches neuronales Netzwerk, tensorflow, Farbbestimmung, künstliche Intelligenz, Mixed Intelligence, tiefes künstliches neuronales Netzwerk, maschinelles Lernen

Abstract

TensorFlow is an open source software library for high-performance numerical computation, which comes with strong support for machine learning and deep learning. In this study, TensorFlow was used for object detection and image classification. A deep convolutional artificial neural network was trained.

The result of this study is a software library “Third Ai” for lost property offices. This software library provides following three steps. Firstly, given an image showing one lost property item, the library classifies the lost property item per name. Secondly, dominant colours are calculated as major and minor colour for the detected item. In the last step all findings are returned, each in a well-formatted and human-readable result text.

A self-made training data set of 1736 high-resolution images showing 7 different lost property item classes was used to train an artificial neural network. These images were created under laboratory conditions and optimised for invariance regarding variations of angle, distance, position, rotation, illumination and background. The goal of this study was to reach an accuracy of 90 per cent for item classification, which was fulfilled successfully. Furthermore, the software library can describe item colours and supports the training of new item classes due to the use of a strategy called Mixed Intelligence.

It is hoped this study will inform practitioners about how to optimise item classification results on their models.

Keywords: image classification, item classification, image recognition, neural network, tensorflow, colour classification, artificial intelligence, mixed intelligence, deep learning, machine learning

Acknowledgements

I would first like to express my very profound gratitude to my partner and to my parents for providing me with unfailing support and awesome encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you my dearest Alina, Mum, Dad and Eel.

I would also like to thank my Technical Supervisor Christian Kenngott as well as the Technical Domain Specialist Georg Müller from RUBICON IT GmbH and my Academic Supervisor Robert Pucher from Fachhochschule Technikum Wien. Thank you for your awesome support.

Finally, I would like to joke some guys from my hometown who once told me that I will never make it so far. Who is the master now?

Paul Puntschart

Contents

1 Introduction

This thesis is written by Paul Puntschart who is interested in artificial intelligence and developed the software library “Third Ai” which is a solution to identify and describe lost property items for and in lost property offices by images of items captured by smartphones or other cameras. Lost property offices are institutions in Austria, where people can hand over items which they found as lost property. People can search for their lost property over a public website (in Austria this is https://www.fundamt.gv.at/) and so get their lost property back. RUBICON IT GmbH was interested in such a software library because they wanted to use artificial intelligence to improve the rate of lost properties that found their way back to their owner.

The focus of this chapter is to explain the motivation behind the thesis, followed by aimed goals, used methods of development and involved persons and institutions. At first, the section “Problem” gives an overview of the problematic situation of lost property offices; lost property officers have a hard job because it is a very subjective one. The problem of describing an item is not only bound to the lost property domain. Subjective descriptions are a general problem since humans describe things. Second, the section “Goal” explains how we may get rid of subjective descriptions or at least variable descriptions by committing this problem to a computer. Next, the section “Methods” shows the approach of reaching the goal in a capacity of 60 person days (480 working hours). Finally, the section “Team” is a detailed listing of all people involved in this master project.

1.1 Problem

One problem about lost property office item registrations is that it takes long times and high effort to fill out forms about lost property items. Lost property officers have to classify lost property with given catalogues and detailed textual descriptions. If we would speed up item registrations by one minute and if in average one registration is done every minute, we can save 24 hours of work per day. The problem directly includes another one, because to create lost property item descriptions is a highly subjective issue. However, these item descriptions are essential to get our lost property back.

The main reasons for variability in item descriptions are knowledge, vocabulary and colour perception. An excellent example to make this clear is the item shown in Figure 1: Image of a Lost Property Item, and the question “What is shown in this image?”.

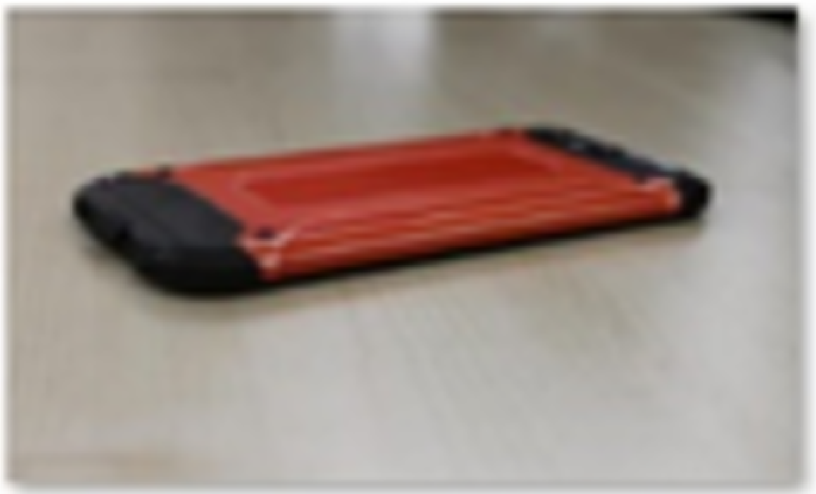


Figure 1: Image of a Lost Property Item

This item can been described as smartphone, mobile phone, cell phone, Handy, calculator, case or electronic thing, with a colour description widely ranging from orange, scarlet, over red to black also including all kind of mixtures of them. Some even say the item is green, due to colour blindness.

Although it is also a correct answer, hardly anyone mentiones the wooden desk, which is the main item on this image regarding the pixel area per item. That shows that only lost property domain knowledge and problem understanding can solve the issue of finding the focus and describe items so that others would recognise them just by their description.

To solve the question above, Figure 1 shows (in the authors words) a “black smartphone in a red cover laying on a wooden desk”. In this thesis we want to ask the computer what item is shown and how it looks like, and so eliminate the human subjectivity.

1.2 Goal

This goal description was created in the first week of work and has not changed since then.

The goal of this thesis is to develop a software library which can output item descriptions for given images showing lost property items. Item descriptions shall include an item class name and item colour information. The subtask of item classification has to reach an accuracy of 90 per cent. The accuracy is defined as how many items were classified correctly depending on their item class.

The following Figure 2: Topic Description of the Preliminary Project Proposal shows the original topic description, written by the Technical Supervisor Christian Kenngott and Paul Puntschart.

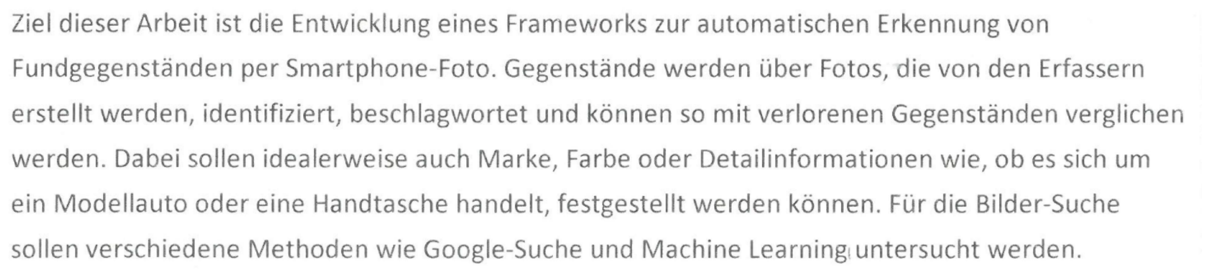

Ziel dieser Arbeit ist die Entwicklung eines Frameworks zur automatischen Erkennung von Fundgegenständen per Smartphone-Foto. Gegenstände werden über Fotos, die von den Erfassern erstellt werden, identifiziert, beschlagwortet und können so mit verlorenen Gegenständen verglichen werden. Dabei sollen idealerweise auch Marke, Farbe oder Detailinformationen wie, ob es sich um ein Modellauto oder eine Handtasche handelt, festgestellt werden können. Für die Bilder-Suche sollen verschiedene Methoden wie Google-Suche und Machine Learning untersucht werden.

Figure 2: Topic Description of the Preliminary Project Proposal

1.3 Methods

For this master project, a custom development model was developed which fits the personal approach of solving problems. The structure of this thesis is also based on that model.

There was a meeting held every two weeks between the Technical Supervisor Christian Kenngott, the Technical Domain Specialist Georg Müller and the author of this thesis to discuss progression, results and future steps (mentioned persons are described in the section “Team”). This way a very agile development process was possible. Presentation slides were used in every meeting to support a fast understanding of the master project status.

The master project status was also submitted to the Academic Supervisor Robert Pucher every week by excerpts of a project diary, written daily during the implementation phase of the master project.

1.3.1 Development Model

The development model starts with a proof of concept to get a clue what to expect and to assess what is coming up. The next step is to perform iterative prototype development and so collect knowledge and practice. In the final stage, the plan is to develop a stable release candidate which includes all previous collected proven knowledge.

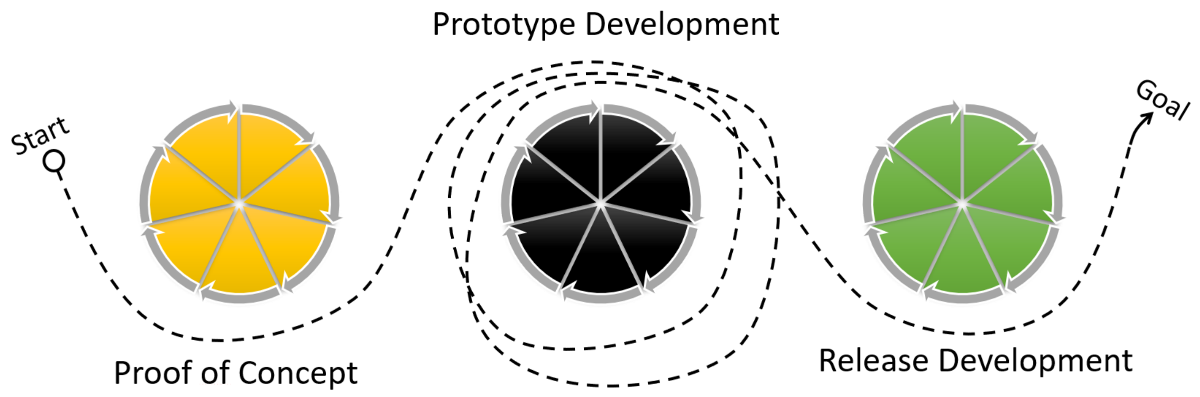


Figure 3: The Development Model

Figure 3: The Development Model shows the timeline which connects the proof of concept, prototype development and release development. The timeline in the figure is the dashed line between *Start* and *Goal*. It passes the *Proof of Concept*, then loops around *Prototype Development* because several prototypes are expected and finally passes the *Release Development*.

1.3.2 Iterative Prototype Model

The previous section describes the development model. One stage of it is prototype development, which is also the largest one in the aspect of time consumption. The following iterative prototype model was developed to fit the specific needs of working with artificial intelligence.

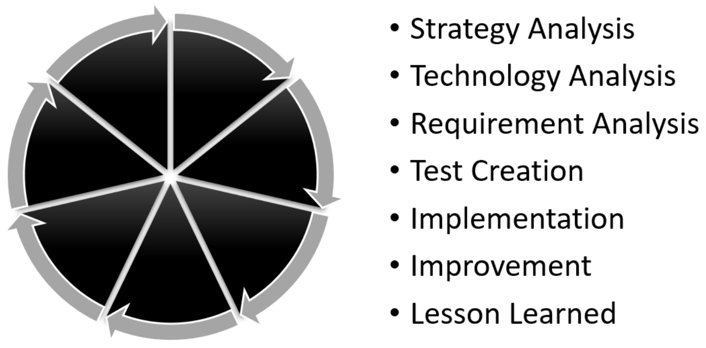


Figure 4: The Iterative Prototype Model

Figure 4: The Iterative Prototype Model shows the seven stages of a typical prototype development iteration. The purpose of each iteration is to cover a specific topic to build up practical and empirical knowledge. Different topics require different strategies and different technologies, so the stages Strategy Analysis an Technology Analysis are treated first.

1.3.3 Investment of Working Hours

57 working days á 8 hours were invested in the implementation phase of this master project. Working hours were spread over 6 months in total. The following chart in Figure 5: Investment of Working Hours by Date shows the distribution of working hours over the implementation phase of 6 months.

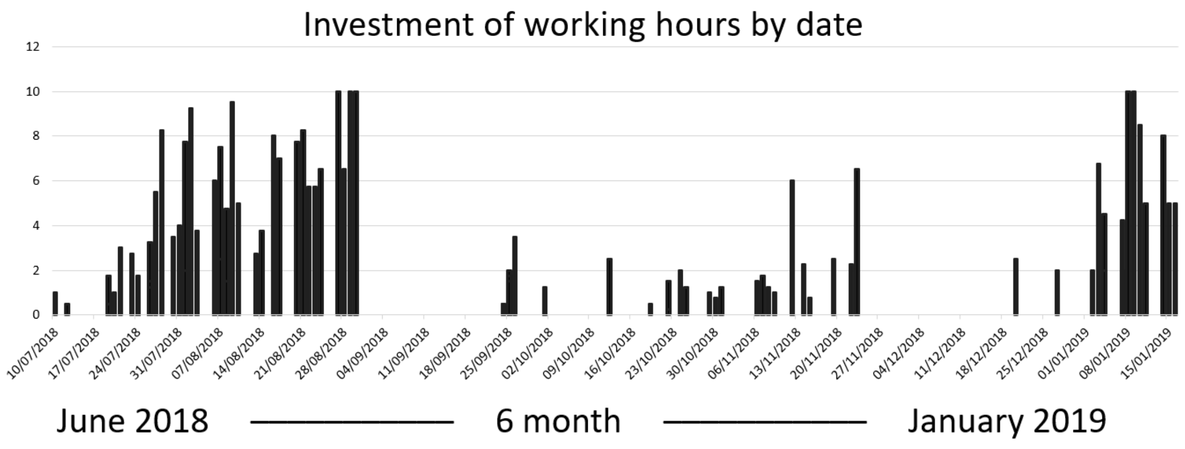


Figure 5: Investment of Working Hours by Date

The working hours investment is interrupted by 2 periods of holidays. The project was proposed in June 2018 and the implementation phase finished in January 2019. All findings were documented during the implementation and summarised in this thesis until May 2019. The developed solution “Third Ai”, which is the heart of the implementation, was implemented in just 7 working days in January 2019.

1.4 Team

This section shows all persons and institutions involved in the master project and how they are connected with each other. In total 5 persons and 3 institutions are involved.

The following Table 1 shows all persons involved in the project. The abbreviations are used to describe their connections in Figure 6: Team Organigram.

Name	Abbreviation	Role
Paul Puntschart	PP	Student
Christian Kenngott	CKE	Technical Supervisor
Georg Müller	GM	Technical Domain Specialist
Robert Pucher	RP	Academic Supervisor
Alessandra Pieroni	UM-P	Academic Reviewer

Table 1: Table of Persons

The following Table 2: Table of Institutions shows all institutions involved in the project, the full name, address and contact information.

Institution	Address	Contact
Fachhochschule Technikum Wien	Höchstädtplatz 6 1200 Wien Austria	Email: info@technikum-wien.at Phone: +43 1 333 40 77-0
RUBICON IT GmbH	Werdertorgasse 14 1010 Wien Austria	Email: office@rubicon.eu Phone: +43 1 533 25 55-0
Università degli Studi Guglielmo Marconi	Via Plinio, 44 - 00193 Roma Italia	Fax +39 06 37725212 E-mail: info@unimarconi.it Tel +39 06 377251

Table 2: Table of Institutions

The following organigram about the team, Figure 6: Team Organigram, shows the connections between persons and institutions.

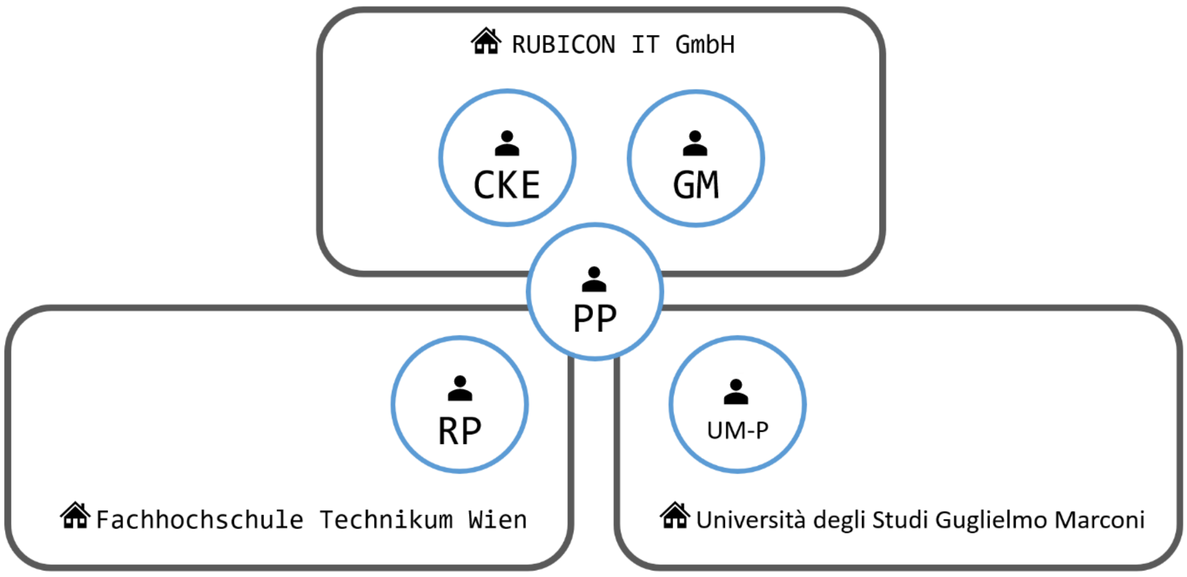

RUBICON IT GmbH
CKE
GM
PP
RP
UM-P
Fachhochschule Technikum Wien
Università degli Studi Guglielmo Marconi

Figure 6: Team Organigram

The master thesis is written by the student Paul Puntschart (PP). Technical Supervisor Christian Kenngott (CKE) and Technical Domain Specialist Georg Müller (GM) are working at the company Rubicon and so in direct contact to Paul Puntschart, who is also working at RUBICON besides studying at the Fachhochschule Technikum Wien and the Università degli Studi Guglielmo Marconi.

The master thesis is judged by Academic Supervisor Robert Pucher (RP), Technical Supervisor Christian Kenngott (CKE) and Alessandra Pieroni (UM-P).

2 Basics

This chapter references work which can be used as basic knowledge for this thesis. It covers artificial intelligence and colour theory, each in a separate section. The first section is about references to artificial intelligence literature, containing definitions, problems and explanations. The second section introduces references to colour definitions and colour theories.

2.1 Artificial Intelligence

In a research project proposal "A proposal for the Dartmouth summer research project on artificial intelligence" in 1955 [1], the work of a study about artificial intelligence is described as "the study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.". In this proposal, 7 aspects of artificial intelligence problems are listed:

- Automatic Computers
- How Can a Computer be Programmed to Use a Language
- Neuron Nets
- Self-Improvement
- Abstractions
- Randomness and Creativity

In "Artificial intelligence: The very idea" [2] Haugeland describes the work with artificial intelligence as "the exciting new effort to make computers think...machines with minds, in the full and literal sense.". Haugeland also says "the very idea of AI may or may not be successful: we can only find out by trying".

In "Artificial Intelligence Third Edition" [3], artificial intelligence is described as "the study of how to make computers do things at which, at the moment, people are better.". This means that artificial intelligence is dependent of the current state of computer science. Artificial intelligence has to perform tasks that a human would be considered intelligent to perform it. To let a computer act in a human way is a goal we want to achieve because this way we can use science to let computers do work for us.

2.2 Colour Theory

In “Color Measurement of Segmented Printed Fabric Patterns in Lab Color Space from RGB Digital Images” [4], the Lab colour space and the RGB colour space are introduced. The RGB colour space is a cube with red, green, blue, cyan, magenta, yellow, black and white corners.

The book “Zur Farblehre” [5] (engl. Theory of Colours) by Johann Wolfgang von Goethe is about his theory of colours. Goethe based his theory on the eye´s experience of colours. He researched the nature of colours and how colours are perceived by humans. The book was published in 1810 in German and translated, “Goethe’s Theory of Colours: Translated from the German; with Notes by Charles Lock Eastlake” [6], to English (with added notes) in 1840 by Charles Lock Eastlake. The original version contains the following six parts (chapters), the translated version has an extra chapter with notes.

The first part is about physiological colours, including effects of light and darkness on the eye, lights, shadows and subjective halos. Physiological colours are the foundation of the whole doctrine. The second part is about physical colours; colours which are produced by material mediums. The third part is about chemical colours; colours which we can produce or fix in certain bodies. Black and white are described as chemical colours. Part IV is about general characteristics of colours and the general nature of colour appearance. The last chapters, part V and part VI are about relations to other pursuits and effects of colours with reference to moral associations.

“Opticks” [7] is a book by Isaac Newton, published in 1730. Newton based his theory on the mathematical understanding of optics and found that white light is a composition of individual colours and already contains all other colours. The following description is about the fourth edition of “Opticks”.

The first book of opticks is divided into the chapters “Definitions”, “Axioms” and “Propositions”. Book 2 adds more propositions and observations. The third book adds more observations, queries and quests.

3 Related Work

This chapter references related work. TensorFlow and ColorMine were used in this work to fullfil the thesis goal. TensorFlow is often used to train neural networks; in this work, TensorFlow was used for object detection and image classification. ColorMine is a proven colour distance calculator.

3.1 TensorFlow

TensorFlow [8] is an open source software library for high-performance numerical computation, which comes with strong support for machine learning and deep learning. In comparison to this thesis, in “Greedy Algorithm Based Deep Learning Strategy for User Behavior Prediction and Decision Making Support”, TensorFlow is used to suggest a deep learning strategy for decision support, based on a greedy algorithm.

3.2 ColorMine

ColorMine [9] is an online delta-E calculator and can also be used as software library to calculate colour distances. In comparison to the used colour theory of the RGB colour space in this work, ColorMine supports the color spaces Rgb, Cmy, Cmyk, Hsl, Xyz, CIE-L*ab, CIE-Lch, and Yxy.

4 Proof of Concept

The first practical step of this master project was to perform a so-called proof of concept. The proof of concept is, in the manner of the chosen development model, see 1.3.1 Development Model, an effort to quickly determine the feasibility of aimed goals and helps to facilitate technology decisions. With a small set of the hardest requirements (an excerpt of the requirements of the thesis), misleading strategies or technologies can be rejected with small effort. In general, a proof of concept is an evidence that an idea, a plan or a project is likely to succeed. In this work it turned out not to be an evidence, however, it indicated the right direction and a possible solution, at least to meet the requirements described in the following section.

4.1 Requirements

This section shows the requirements for the proof of concept which led to the use of TensorFlow as technology. Requirements were developed to analyse the general topic of image classification with artificial neural networks.

- Setup TensorFlow (the graphical processor unit version of TensorFlow) and run a test to verify that the setup is working and faster than the CPU version
- Use an already proofed training set to train the computer in image classification
- Create a custom training set and evaluate the computer for image classification tasks

The first three requirements are preconditions for the last one. The last requirement is the most important one because the intention behind it is to decide for or against TensorFlow as technology.

4.2 Solution

The previous section showed the requirements for the proof of concept, and in each of the following subsections, the solution to every requirement is presented.

4.2.1 Setup TensorFlow-GPU

For this requirement Python and TensorFlow were installed on a laptop. The laptop is a special XMG setup with built-in desktop PC components. It took 8 hours to set up Python and Tensorflow.

Device Specification

The model of the laptop of the author is “XMG Zenith” with the following specification:

- Operation system: Microsoft Windows Version 10.0.17134.165
- Processor: Intel Core i7-8700 CPU @ 3.20 GHz
- Installed memory (RAM): 16 GB
- System type: 64-bit Operating System, x64-based processor
- Graphical processor unit: NVIDIA GeForce GTX 1080

The laptop device can optionally support a second graphical processor unit of NVIDIA GeForce GTX 1080. It was decided to upgrade the laptop only if out-of-memory errors would occur which did not happen.

Python

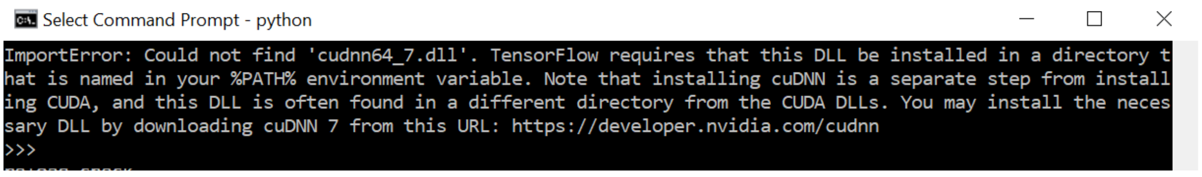
Python Version 3.6.6 can be downloaded from https://www.python.org/ftp/python/3.6.6/python-3.6.6-amd64.exe. It was installed under `C:\Program Files\Python36`. The checkbox “set the PATH environment variable” was enabled at the installation process.

Tensorflow

To install TensorFlow API 1.9 (with native pip), a command prompt was started with administrator rights and the appropriate pip3 install command was issued:

```
C:\> pip3 install --upgrade tensorflow-gpu
```

During the installation process, there were many warnings that `C:\program files\python36` is not set on PATH, so the device was rebooted. The installation guidelines can also be found on https://www.tensorflow.org/install/install_windows. The website https://developer.nvidia.com/cuda-90-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exelocal was visited and the legacy cuda version 9.0, namely “cuda_9.0.176_win10.exe” was downloaded and installed. It was necessary to edit the PATH environment variables from the path *9.2* to `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin`. After a device reboot following output was shown:

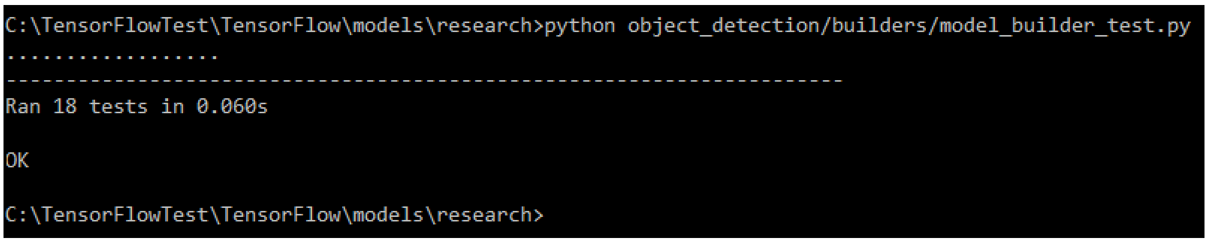


```
ImportError: Could not find 'cudnn64_7.dll'. TensorFlow requires that this DLL be installed in a directory t
hat is named in your %PATH% environment variable. Note that installing cuDNN is a separate step from install
ing CUDA, and this DLL is often found in a different directory from the CUDA DLLs. You may install the neces
sary DLL by downloading cuDNN 7 from this URL: https://developer.nvidia.com/cudnn
>>>
```

Figure 7: CUDA Driver Error Message

To get the missing “cudnn64_7.dll”, “cuDNN v7.1.4 (May 16, 2018), for CUDA 9.0” was needed, available from https://developer.nvidia.com/rdp/cudnn-download after registration. In the zipped download file, in the bin directory, “cudnn64_7.dll” and copied it to `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin` was found.

It was verified that TensorFlow had been installed correctly by issuing the following command:



```
C:\TensorFlowTest\TensorFlow\models\research>python object_detection/builders/model_builder_test.py
..................
----------------------------------------------------------------------
Ran 18 tests in 0.060s

OK

C:\TensorFlowTest\TensorFlow\models\research>
```

Figure 8: TensorFlow GPU Setup Test

As shown in the screenshot in Figure 8: TensorFlow GPU Setup Test, the model builder test ran 18 tests in 60 milliseconds successfully. This result enabled to go on with the next requirement.

4.2.2 Image Classification of Flowers

Tensorflow offers a tutorial in their guide page “How to Retrain an Image Classifier for New Categories” [11]. The guide uses an archive of creative-commons licensed flower photos to retrain a new flower class on a pre-trained neural network. The tutorial covers training, bottlenecks, training data visualisation and hyperparameters. There is also a section about how to train a custom image set. The guide was used to verify that TensorFlow is working properly and that the graphical processor unit of the laptop meets the minimum requirements to process training.

4.2.3 Image Classification of Border Collie Bitches

Alina Gaugg is the owner of two Border Collie bitches, Borderline Country Lordana “Minsk” and Borderline Country Tiara “Titos”. On her website https://www.mileysworld.at/ [12], Alina shows hundreds of images of the bitches in her online gallery. These images were used to train a convolutional artificial neural network to classify images showing Minsk or Titos. The following Figure 9: Borderline Country Lordana “Minsk” (left) and Borderline Country Tiara “Titos” (right) shows two images of the used training set.

Figure 9: Borderline Country Lordana “Minsk” (left) and Borderline Country Tiara “Titos” (right)

4.3 Results

This section shows the results of the proof of concept. Following subsections explain the evaluated accuracy of tested image classification tasks and why TensorFlow has been chosen as the technology for further work. The last subsection is a conclusion over all results.

4.3.1 Training Accuracy

TensorBoard was used to analyse the created model. “TensorBoard is a suite of web applications for inspecting and understanding your TensorFlow runs and graphs.” (see https://github.com/tensorflow/tensorboard/blob/master/README.md, [13]). With TensorBoard it was possible to watch accuracy and cross-entropy live at the training over training steps and so over time. The following figure shows both accuracies on the left and cross-entropy on the right diagram side over 50 000 training steps which were equivalent to 60 minutes in time.

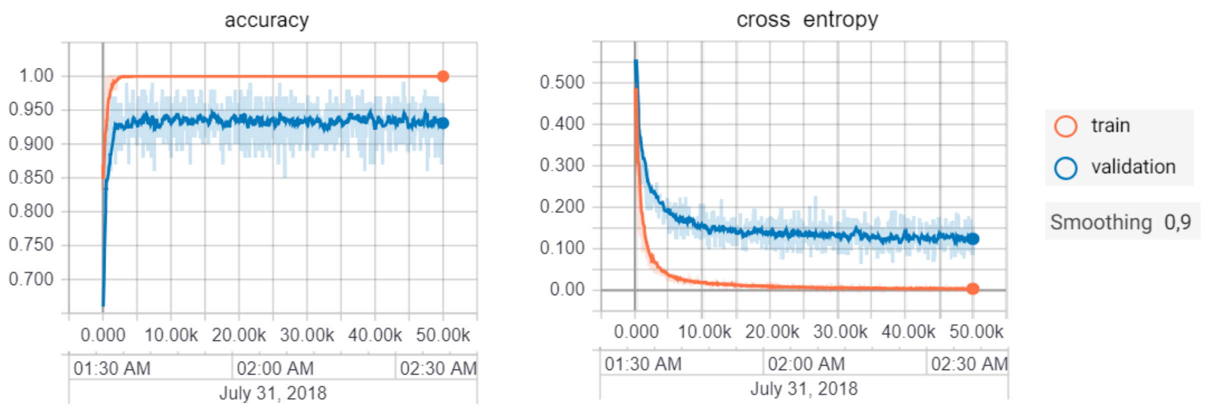


Figure 10: TensorBoard Training Graphs

In the accuracy diagram on the left, we can see a smoothed graph (the original graph is shown in brighter colour) representing the accuracy over time. We can say that with the chosen training parameters, the graph converges at 10.000 steps and shows no further improvement.

In the cross entropy diagram on the right, we can see that the cross-entropy is still improving after 50.000 training steps.

4.3.2 Sample Accuracy

Besides training accuracy and validation accuracy, it was decided to create a small sample with images the computer has never seen before and call that sample accuracy. Table 3 shows the results. The first row is the header row followed by a total of 9 rows representing the sample results. The headers are:

- Sample: The sample number
- First Guess: The computer´s guess which image class is shown in the given image. The score, $s1$, stands for the confidence of the computer´s decision.
- Second Guess: The computer´s second guess, which is the opposite of the first guess for binary classification with a score of $1 - s1$.
- Truth: The image class for each sample in truth.

Following Table 3: Proof of Concept - Sample Accuracy shows the result of 9 evaluated images.

Sample	**First Guess**	**Second Guess**	**Truth**
1	titos (score=0.89437)	minsk (score=0.10563)	titos
2	titos (score=0.99913)	minsk (score=0.00087)	titos
3	titos (score=0.81812)	minsk (score=0.18188)	minsk
4	minsk (score=0.93442)	titos (score=0.06558)	minsk
5	titos (score=0.97587)	minsk (score=0.02413)	titos
6	titos (score=0.90440)	minsk (score=0.09560)	titos
7	minsk (score=0.87146)	titos (score=0.12854)	minsk
8	minsk (score=0.81686)	titos (score=0.18314)	minsk
9	titos (score=0.81283)	minsk (score=0.18717)	minsk

Table 3: Proof of Concept - Sample Accuracy

The sample accuracy is 7 correct guesses out of 9 total guesses and so 77.7 per cent. Wrong guesses are marked orange in the figure. We can see that the computer is having issues to classify Minsk correctly. Only 3 out of 5 images of Minsk were decided correctly. However if the confidence is above 82 per cent, the guess is always correct here.

4.3.3 Tournament

To compare human and artificial skills, it was decided to organise a tournament. Seven persons, playing against the computer of the author, should solve the binary image classification task of deciding whether an image shows the students dog Minsk or his other dog Titos.

Boundary Conditions

The persons have 3 minutes time to learn how Minsk and Titos look like with a library of 100 images of each dog. These conditions are the same conditions that the computer have to learn. After the learning time, an image showing one of the bitches is presented and the probands have 3 seconds time to make a guess which one is shown. For every image that a person makes a guess, the computer is also making a guess. After an average of 5 tries, the winner is chosen as the one who can classify more images correctly. Draws are also counted.

Result

In all seven matches, the computer could not be defeated. Only one person was able to play a draw against the computer. The computer´s evaluation time for each image was below 2 seconds.

4.3.4 Conclusion

The previous subsections show that the training accuracy of 92.5 per cent.

The sample accuracy reached 77.7 per cent, and we can also see that if the computer´s confidence is over 82 per cent per guess, the decision is correct. For the sample accuracy of 77.7 per cent it is woth noting that for binary classifications we can calculate the probability that a random classifier has to guess 7 out of 9 classes correctly. The formula behind this is based on the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. With that in mind, we can calculate the probability as

$$P(X \geq k) = \sum_{i=k}^{n} \binom{n}{i} \cdot p^i \cdot (1-p)^{n-i}.$$

For our example, $P(X \geq k)$ for $n = 9$, $k = 7$ and $p = 0.5$, results in a probability of 9 per cent. This means that a sample size of 9 is not a reliable number. Higher sample sizes, for example the probability $P(X \geq k)$ for $n = 90$ and $k = 70$ which actually has the same accuracy, result in 0.0000057 per cent which is more than a million times less likely.

The tournament showed that no one of 7 persons were able to beat the computer.

Because of these evaluations, especially the 92.5 per cent training accuracy, it was decided to choose TensorFlow as the technology for the prototype development. The next planned step was to build a prototype for more than two image classes, create a training set with more than 100 images per class and then train the computer to classify images as explained in detail in the first section of the next chapter.

5 Prototype Development

Each prototype covers one topic. This strategy makes it possible to reach the thesis goal step by step. Unrewarding prototypes can be identified and rejected, prototypes with acceptable results can be used for the final development phase (see chapter 7 Release Development “Third Ai”). The last section in this chapter is a conclusion over all prototypes.

5.1 Prototype 1 - Image Classification

Prototype 1 is, like the work in the previous chapter, an experiment to analyse image classification. The used technology, once again, is TensorFlow. In comparison to the previous chapter, this prototype is already designed for the domain of a lost property office and the special needs that the domain comes with.

5.1.1 Training Set with 3 Classes

It was decided to use the following lost property item classes as training data set and 921 photos of these items were taken at the office workplace desk at RUBICON, each showing exactly one item (or just the empty table for the third class):

- Single keys, 480 images named “key”
- Smartphones, 267 images named “phone”
- Unknown (the empty table), 174 images named “unknown”

For all pictures, different angles, item rotations, item positions, distances and lights were used to make the neural network invariant against these variations. So the computer can not associate sunlight with smartphones for example. 14 different smartphone models and 23 different keys were used as items. The following Figure 11: Prototype 1 Training Set shows a small excerpt of the training set.

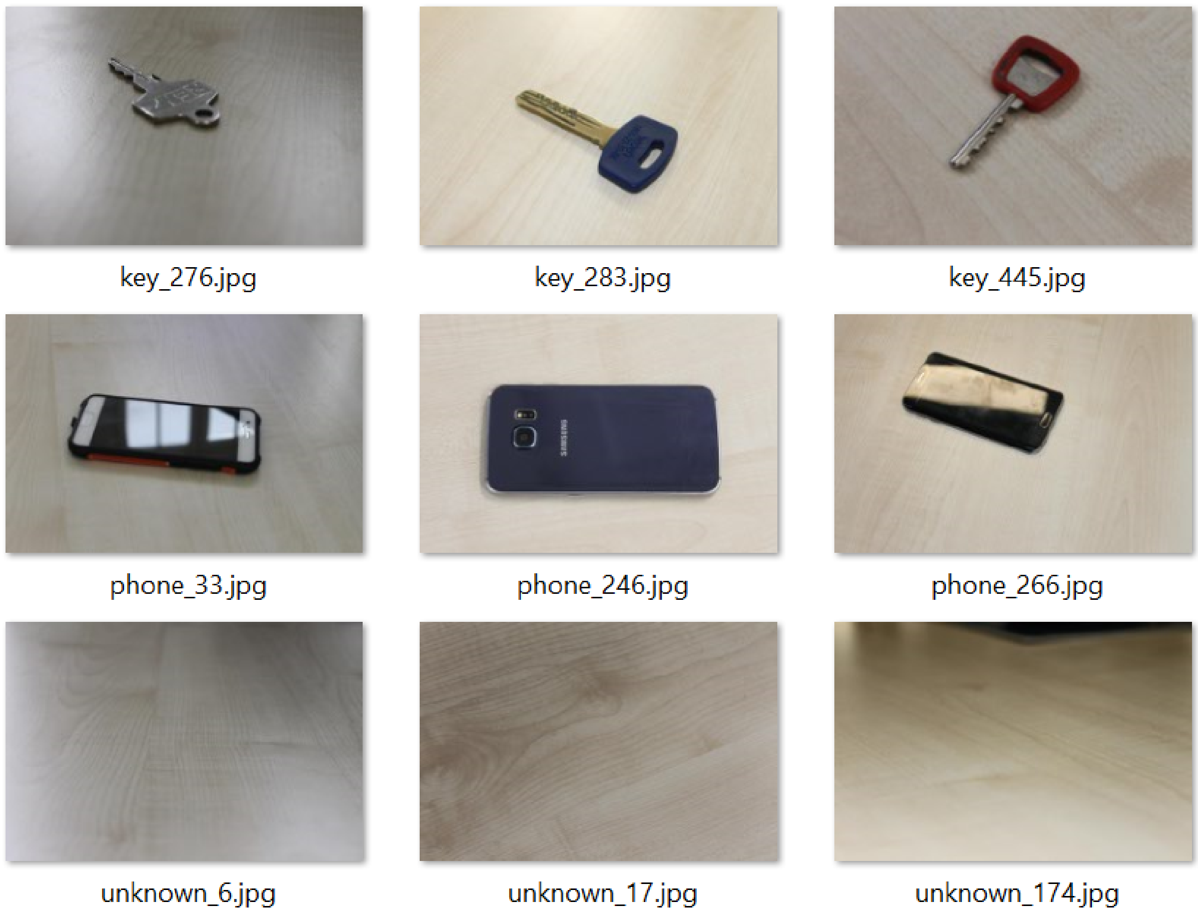


Figure 11: Prototype 1 Training Set

To use those images for training, it was needed to scale them to a dimension of 600 x 400 pixels. This limitation is a limitation of the used pre-trained convolutional artificial neural network. With a horizontal and vertical resolution of 96 dpi, an image took 19 kilobytes of disk space on average. The whole data set took below 20 megabytes in total.

The training set was then used to analyse different learning approaches, as documented in the following analysis sections.

5.1.2 Training with 3 Classes

The pre-trained convolutional artificial neural network model “SSD with Mobilenet v1” was used as model. The “ssd_mobilenet_v1_coco_2017_11_17” was used as fine tune checkpoint. The following parameters were used as training configuration.

```
train_config: {
  batch_size: 2
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.004
          decay_steps: 800720
          decay_factor: 0.95
        }
      }
      momentum_optimizer_value: 0.9
      decay: 0.9
      epsilon: 1.0
    }
  }
  fine_tune_checkpoint: "ssd_mobilenet_v1_coco_2017_11_17/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 200000
}
```

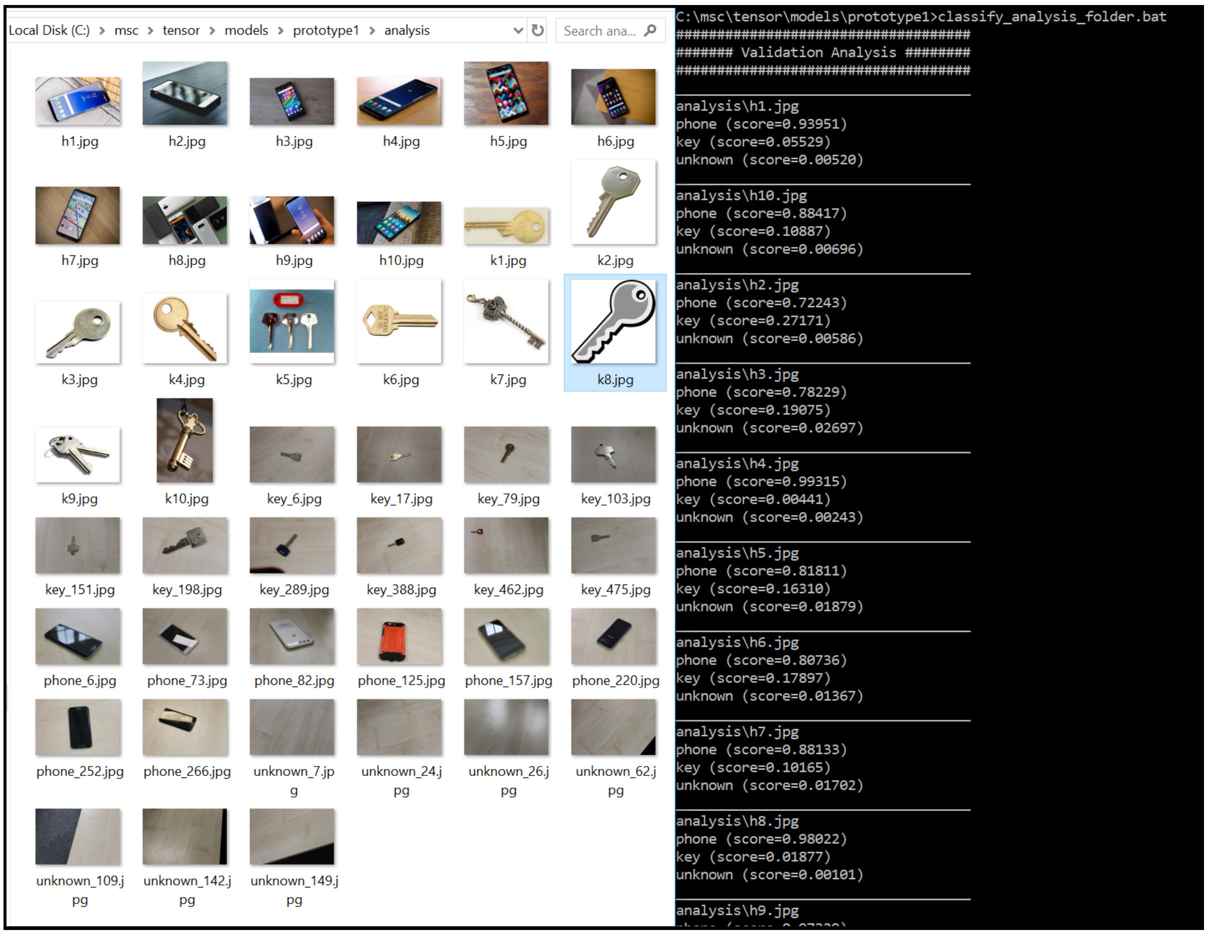
The model was only trained for 3 minutes. The, the training was manually stopped, because TensorBoard showed that the loss function was already converged.

5.1.3 Analysis with 3 Classes

I created a test set of 45 images that the model has never seen before to analyse the result accuracy. Parts of the test set were images I did not use for training, parts random images from the internet with random background, sometimes just white background. I developed following batch script to classify all files in that test set:

```
@echo off
echo ########################################
echo ####### Validation Analysis ########
echo ########################################
echo ____________________________________
for %%i in (analysis\*.jpg) do (
echo %%i
python -m scripts.label_image --graph=tf_files/retrained_graph.pb
 --image=C:\msc\tensor\models\prototype1\%%i
echo ____________________________________
)
pause
)
```

The following Figure 12: Prototype 1 Classification Analysis 1 Screenshot shows a screenshot of the folder, containing all 45 images, at the left side and the running batch script at the right.



```
C:\msc\tensor\models\prototype1>classify_analysis_folder.bat
####################################
####### Validation Analysis ########
####################################
____________________________________
analysis\h1.jpg
phone (score=0.93951)
key (score=0.05529)
unknown (score=0.00520)
____________________________________
analysis\h10.jpg
phone (score=0.88417)
key (score=0.10887)
unknown (score=0.00696)
____________________________________
analysis\h2.jpg
phone (score=0.72243)
key (score=0.27171)
unknown (score=0.00586)
____________________________________
analysis\h3.jpg
phone (score=0.78229)
key (score=0.19075)
unknown (score=0.02697)
____________________________________
analysis\h4.jpg
phone (score=0.99315)
key (score=0.00441)
unknown (score=0.00243)
____________________________________
analysis\h5.jpg
phone (score=0.81811)
key (score=0.16310)
unknown (score=0.01879)
____________________________________
analysis\h6.jpg
phone (score=0.80736)
key (score=0.17897)
unknown (score=0.01367)
____________________________________
analysis\h7.jpg
phone (score=0.88133)
key (score=0.10165)
unknown (score=0.01702)
____________________________________
analysis\h8.jpg
phone (score=0.98022)
key (score=0.01877)
unknown (score=0.00101)
____________________________________
analysis\h9.jpg
```

Figure 12: Prototype 1 Classification Analysis 1 Screenshot

I manually validated the result of all 45 image classifications. The result was that 100% (45 out of 45) images have been classified correctly. So the accuracy was 100 per cent.

5.1.4 Training Set with 4 Classes

I extended the training set with the class "Wallet" with 179 images. I decided to pick those classes, because they are mentioned under the top 10 of lost items in vienna [16].

5.1.5 Training with 4 Classes

The training was visualised with TensorBoard. The following Figure 13: Training Graphs in TensorBoard shows accuracy and cross entropy over 2000 training steps. The duration for 2000 training steps was 3 minutes.

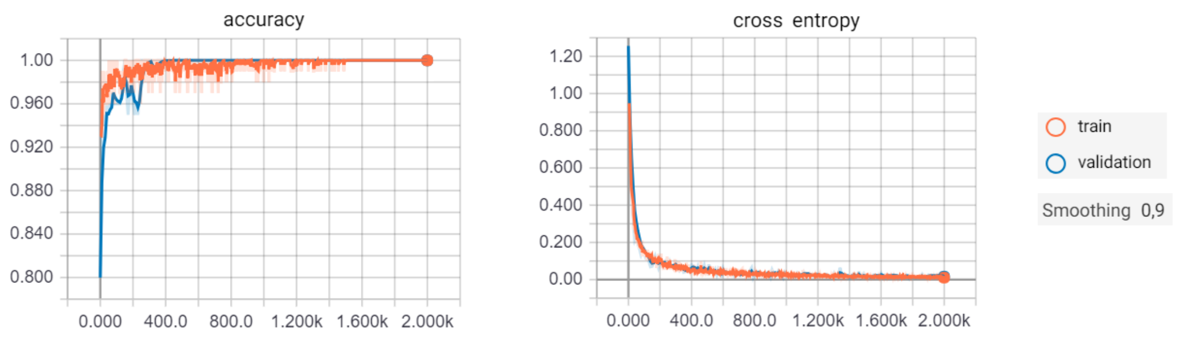
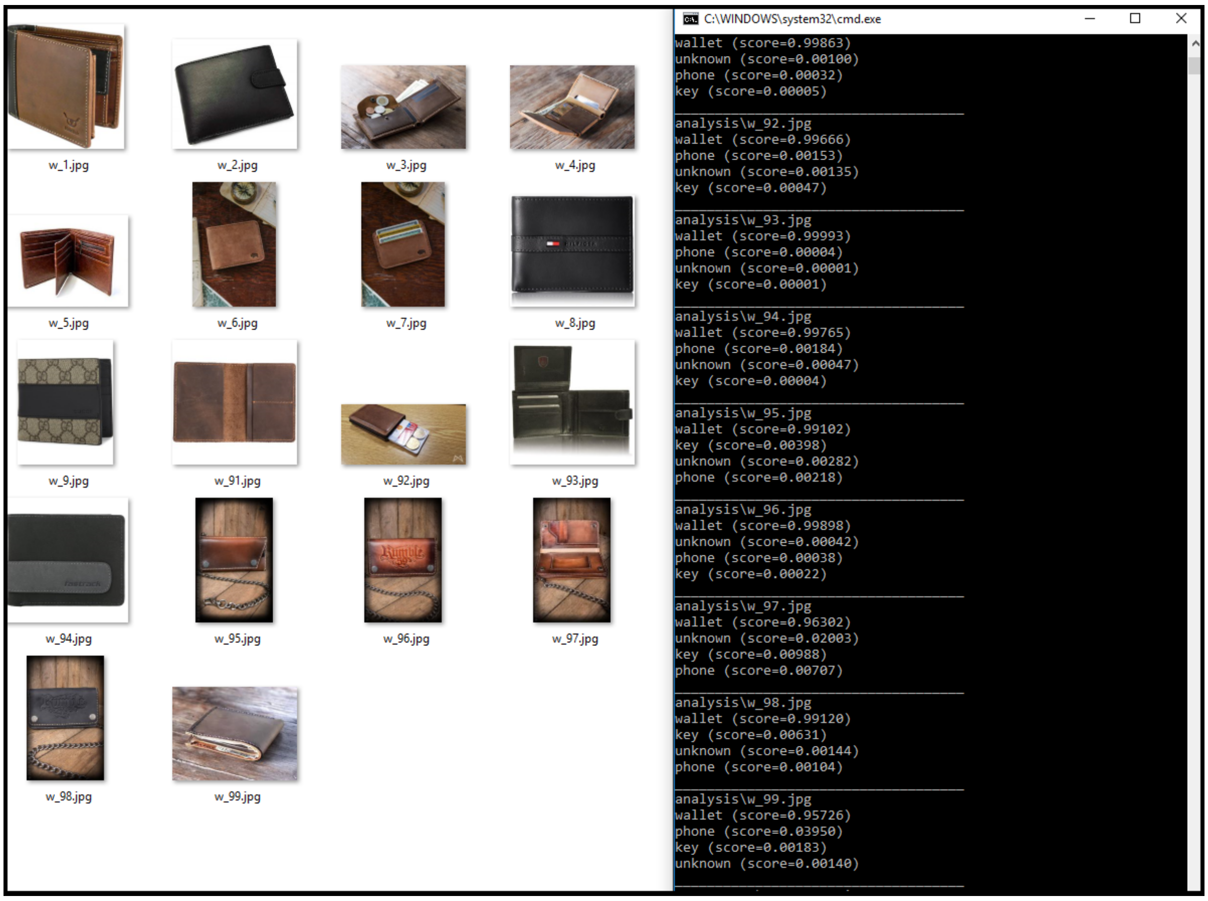


Figure 13: Training Graphs in TensorBoard

The accuracy for the training set reached 100 per cent which means that at the end of the training the computer was able to classify all tested images correctly. The cross entropy converged to 0.

5.1.6 Analysis with 4 Classes

I used the developed batch script again to classify a new test set containing 18 images of wallets. I used random images from the internet with random background, sometimes just white background, as sample. The following Figure 14: Prototype 1 Classification Analysis 2 Screenshot shows the test set containing all 18 images at the left side and the running batch script at the right.



```
wallet (score=0.99863)
unknown (score=0.00100)
phone (score=0.00032)
key (score=0.00005)
________________________________________
analysis\w_92.jpg
wallet (score=0.99666)
phone (score=0.00153)
unknown (score=0.00135)
key (score=0.00047)
________________________________________
analysis\w_93.jpg
wallet (score=0.99993)
phone (score=0.00004)
unknown (score=0.00001)
key (score=0.00001)
________________________________________
analysis\w_94.jpg
wallet (score=0.99765)
phone (score=0.00184)
unknown (score=0.00047)
key (score=0.00004)
________________________________________
analysis\w_95.jpg
wallet (score=0.99102)
key (score=0.00398)
unknown (score=0.00282)
phone (score=0.00218)
________________________________________
analysis\w_96.jpg
wallet (score=0.99898)
unknown (score=0.00042)
phone (score=0.00038)
key (score=0.00022)
________________________________________
analysis\w_97.jpg
wallet (score=0.96302)
unknown (score=0.02003)
key (score=0.00988)
phone (score=0.00707)
________________________________________
analysis\w_98.jpg
wallet (score=0.99120)
key (score=0.00631)
unknown (score=0.00144)
phone (score=0.00104)
________________________________________
analysis\w_99.jpg
wallet (score=0.95726)
phone (score=0.03950)
key (score=0.00183)
unknown (score=0.00140)
________________________________________
```

Figure 14: Prototype 1 Classification Analysis 2 Screenshot

Like in the first analysis iteration, the result was that 100% (18 out of 18) images of wallets have been classified correctly.

We can also see in the script part of the figure that with a score of above 99% as median the computer´s confidence is not even dropped by 1 per cent with other classes beneath the wallet class. The new trained wallet class reached the highest confidence over all classes.

5.2 Prototype 2 - Object Detection

The goal for this prototype is, as an extension to Prototype 1 from the previous section, to locate up to 3 items in a given image. Furthermore, detections shall be visualised in a human-readable format which is judged by the Technical Domain Specialist Georg Müller.

5.2.1 Requirements

This section shows the requirements for the second prototype.

- Prepare a training set containing 3 different item classes and their bounding boxes
- Train the computer to detect and classify items in given images
- Develop a program to visualise detections as images with labelled bounding boxes
- Evaluate the results and analyse edge cases

The following subsections show how those requirements were achieved.

5.2.2 Training Set

To train a neural network for object detection, it is necessary to prepare a training set that consists of images with labelled bounding boxes for each item in one image. Drawing bounding boxes for 642 items took 4 hours.

The following Table 4: Object Detection - Data Split Ratio shows the used split ratio for training data and test data. There may be an opportunity for improvement by choosing another split ratio or the so-called "k-fold cross-validation".

All Data		Training Data		Test Data	
642 items	100 %	578 items	90 %	64 items	10 %

Table 4: Object Detection - Data Split Ratio

The following Table 5: Object Detection - Class Split Ratio shows the total item counts per class and the distributions used for the training data and the test data.

All Data		Training Data		Test Data	
212 keys	33 %	191 keys	33 %	21 keys	33 %
262 phones	41 %	236 phones	41 %	26 phones	41 %
168 wallets	26 %	151 wallets	26 %	17 wallets	26 %

Table 5: Object Detection - Class Split Ratio

5.2.3 Training

The training was based on the pre-trained model "faster_rcnn_resnet101_coco_11_06_2017".

TensorBoard was used to visualise the training. The following Figure 15: Prototype 2 TensorBoard Analysis shows a screenshot with the training progression over 3 899 training steps for precision, recall, classification loss and total loss. The training was running for 7 hours and 17 minutes.

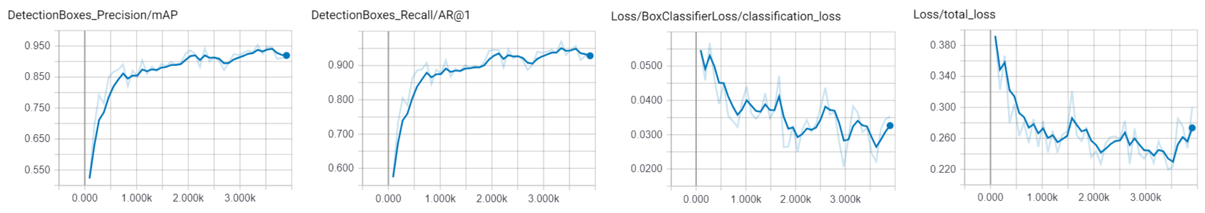


Figure 15: Prototype 2 TensorBoard Analysis

In the diagrams, we can see the smoothed graph (the original graphs are shown in brighter colour) representing the training progression. The best result has been reached at step number 3 621 after 6 hours and 45 minutes. The model was saved and the training stopped at this point as the main goal of this thesis is to break the 90 % line of accuracy which was already done here.

The following figure shows a comparison between the detection of the model and the so-called ground-truth (the handwritten bounding box) after 3 899 training steps.

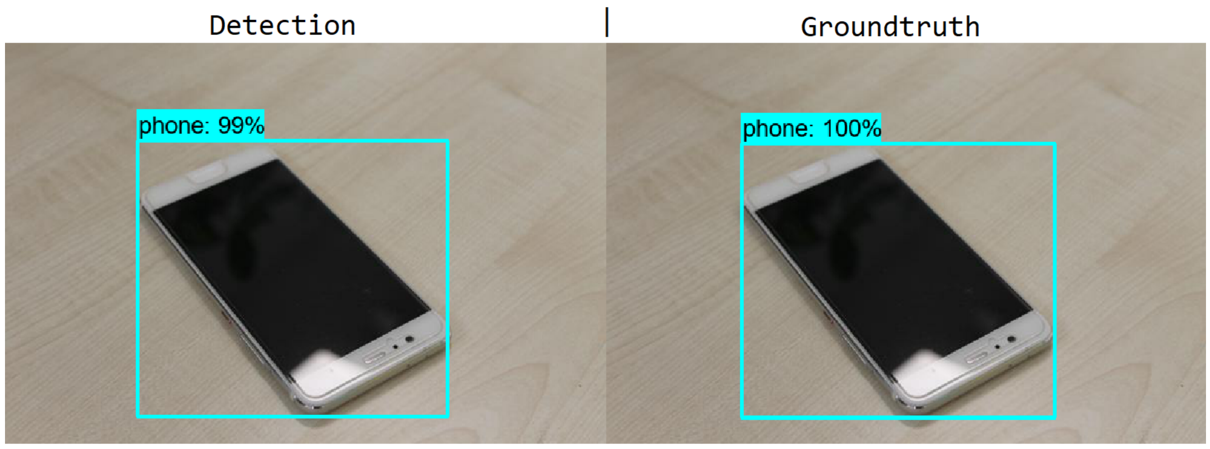


Figure 16: Prototype 2 TensorBoard Detection vs Groundtruth

We can see that the detection is 99 % IoU (intersection over union). There is no need to reach 100 % because the IoU of the autor´s handwritten bounding boxes is below 95 % due to shadows and drawing precision. For all detections a safety padding of 5 % did capture the entire area of the item.

5.2.4 Visualisation

A simple trick to visualise human readable and easy verifiable outputs is to colour the bounding boxes differently for each item class. TensorFlow comes with an opportunity to do that by adjusting the given python script “label_image.py”. A batch script was developed to analyse a given image by path and save a new labelled image at the same directory. The result looks like the following image in Figure 17: Prototype 2 Output.

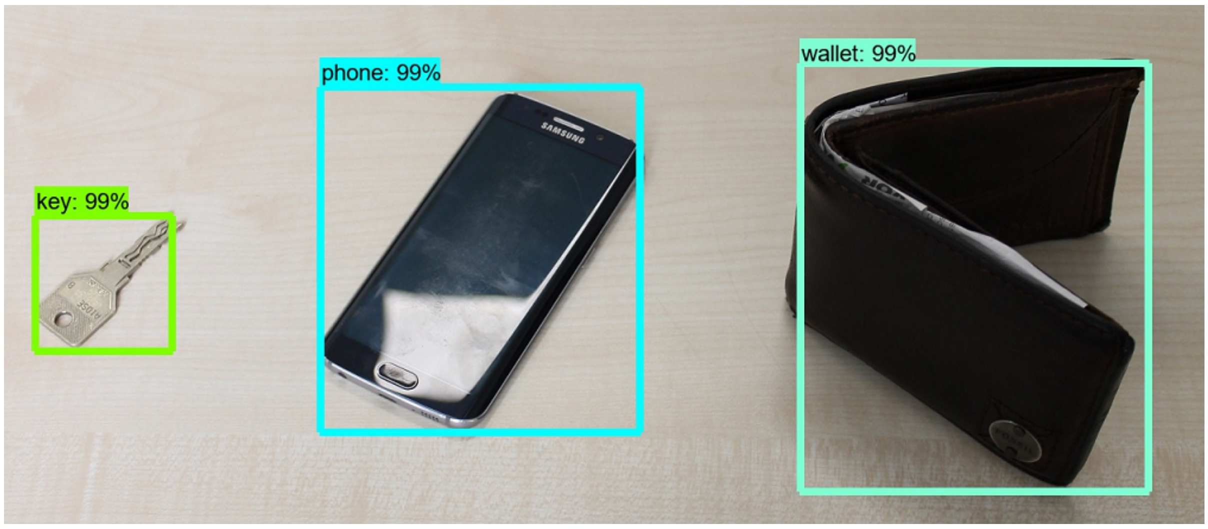


Figure 17: Prototype 2 Output

The resulting output shows the original image with bounding boxes around found items and a text label describing the item class and the score (confidence). The colours of the bounding boxes are representing the item class.

5.2.5 Results

All three trained item classes were analysed on different backgrounds to verify that the model is invariant against background variations in terms of structure, colour and different illuminations. The following Figure 18: Object Detection with Different Backgrounds shows the result of the trained model.

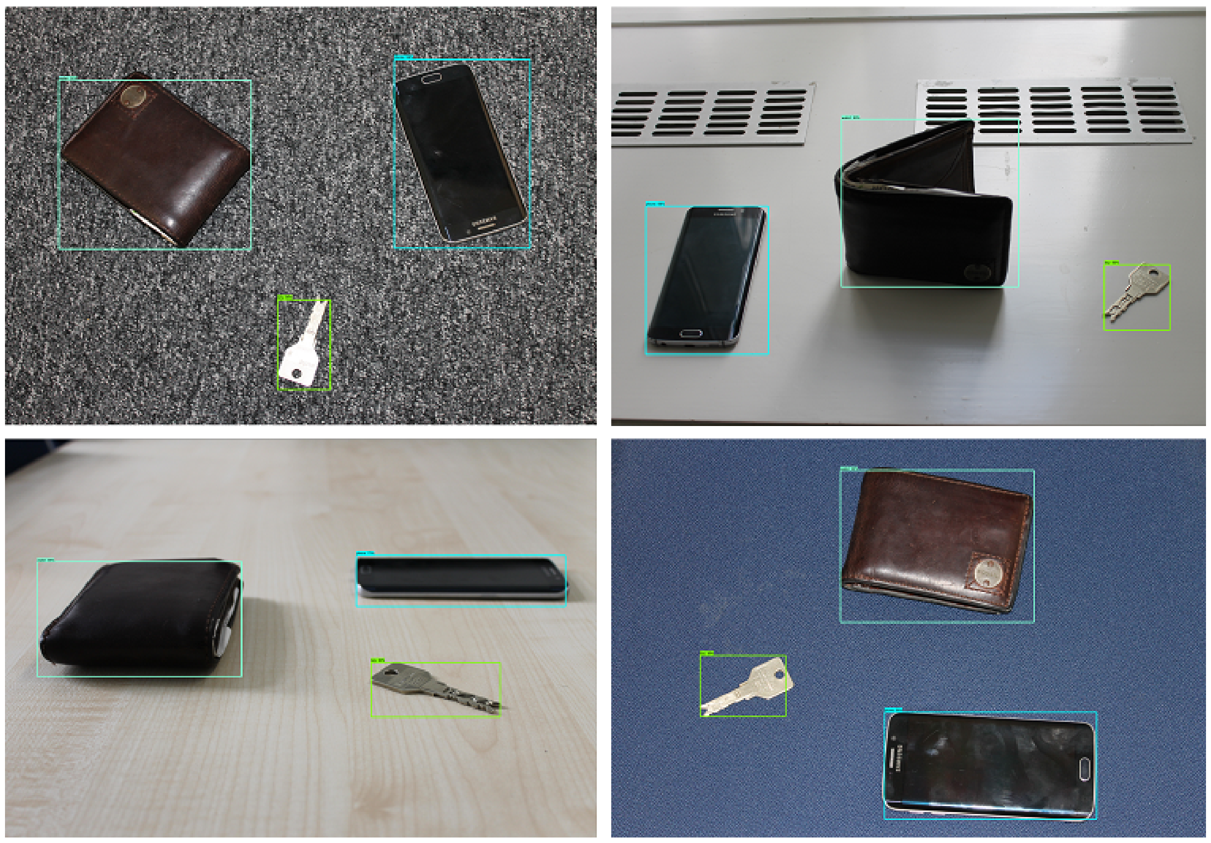


Figure 18: Object Detection with Different Backgrounds

The result was that all three items on all four images were classified correctly. For this figure it is to point out that the results show that the model achieved invariance against the shown backgrounds with black carpet in the top left image, a wooden desk in the bottom left image, a plain white wooden window ledge in the top right image and the blue seat cover in the bottom right image. The most significant IoU mismatch is the one of the detection of the wallet in the top left image with an IoU of 92 %. The lowest score that the model predicted for classification was for the smartphone in the bottom left image with a score of 75 %.

5.2.6 Edge Cases and Misclassifications

This subsection describes findings of edge cases and misclassifications which appeared during the result analysis. The purpose of this analysis is to prevent future pitfalls. Limitations can be considered to develop future requirements.

Abstract Paintings

Because we can hardly look inside neural networks, it was wondered which features the model needs to classify an item. It was decided to paint two keys with Microsoft Paint, besides some other shapes that a human would not declare as key such as triangles, stars and circles. The following Figure 19: Object Detection for Abstract Paintings shows the painting and the result that the model returns, the numbers on the axis represent pixels.

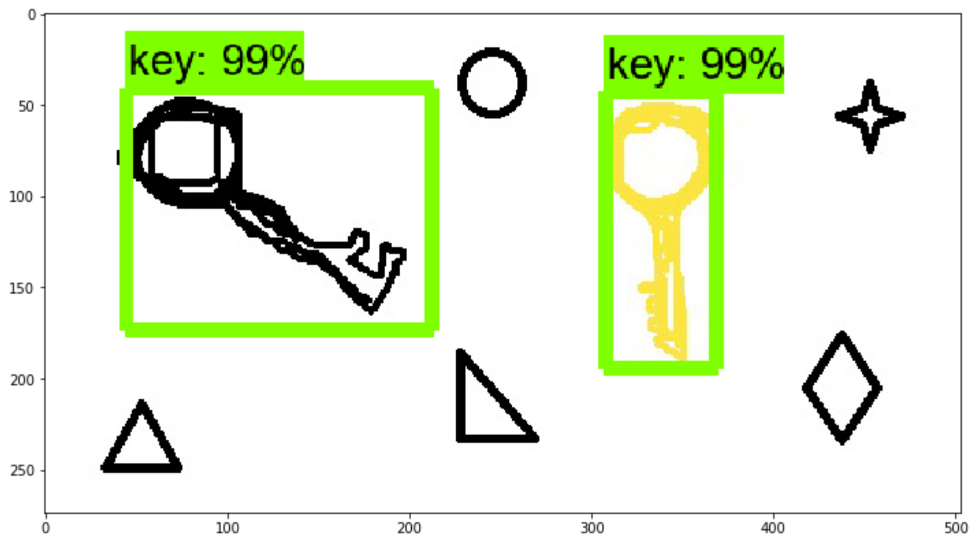


Figure 19: Object Detection for Abstract Paintings

Although this is a side effect of the training, it was not helpful to reach the goal of this thesis, so further analysis in that direction was rejected.

Overlapping Items

There is a big chance that keys were brought to the lost property office as a bunch of keys on a key ring. Since the model was only trained to detect single keys, it was tested what happens with a bunch of 7 keys. The following Figure 20: Object Detection for Overlapping Objects shows the result of the model.

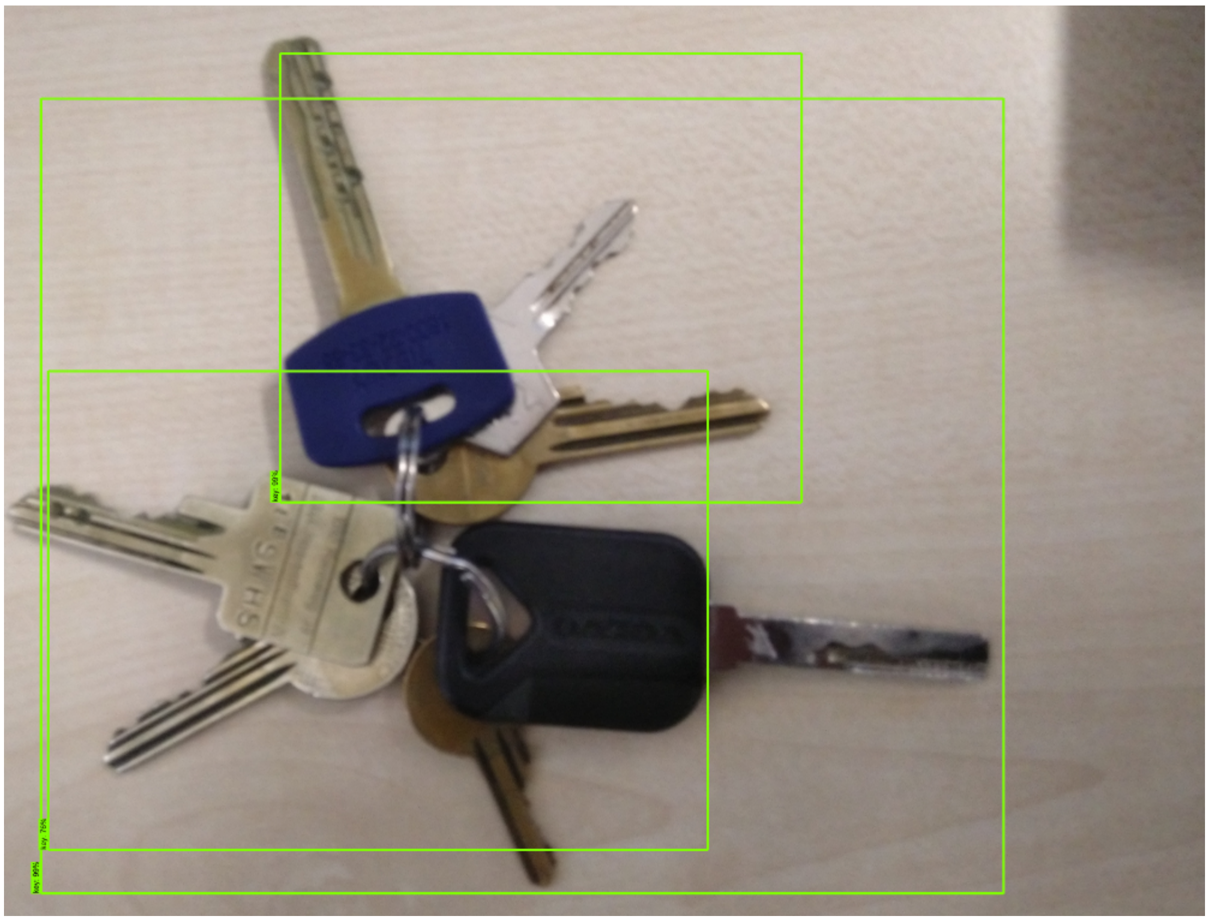


Figure 20: Object Detection for Overlapping Objects

Although this is a nice side effect of the training, it was not really helpful to reach the goal of this thesis, so further analysis in that direction was rejected.

Unknown Objects

The model was tested with objects that the model has never learned, to answer the question “Is my Border Collie either a key, a smartphone or a wallet?”. The following Figure 21: Object Detection for Unknown Objects shows two tested images.

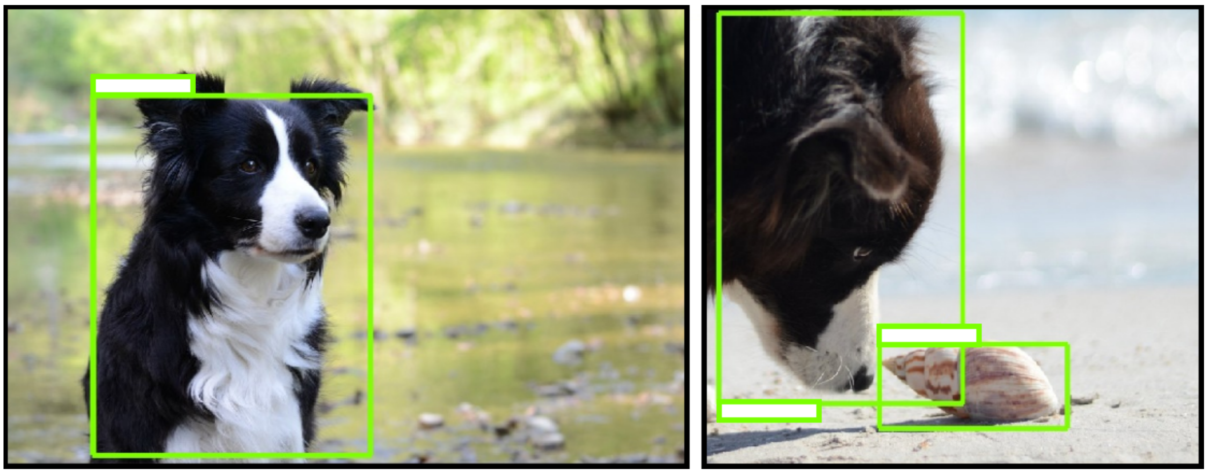


Figure 21: Object Detection for Unknown Objects

The model detected 3 keys in the image. None of the predicted bounding boxes does match one single key. The model has not learned to detect overlapping items.

Conclusion

Although this kind of analysis looks a bit weird at first sight, it is sometimes a big win to analyse misclassifications because we can find features that the model learned “by mistake”. We then have the opportunity to adjust the training set and explicitly train the model to classify unwanted objects for example as “unknown”. For the specific case of lost property offices, we might learn the model not to detect dogs, just in case a lost property officer think it is funny to take a picture of his dog he brought to work that day. We never know what the users are doing with our solution.

The example of the overlapping key led to following future strategies:

- Do not allow images with overlapping items
- Create an extra training set with a category “keyring”

With that in mind, the development of Prototype 2 finished and development continued with a prototype to recognise brands of already detected items, as described in the next section.

5.3 Prototype 3 - Subclassification

This section shows results of image subclassification and why it was rejected as thesis goal. Subclassification is to classify a feature of an already classified lost property item. To classify the brand of an image that has already been classified as a smartphone is such a subclassification. One limiting issue is the small resolution of a brand on an image because the area of a brand logo is relatively small compared to the item.

5.3.1 Goal

The goal of this prototype is to detect brands on already detected smartphones. The requirement of the prototype is that 70 % or more of tested brands shall be classified correctly.

5.3.2 Experiment

A model was trained with 297 images of a Samsung Galaxy S6 Edge device, photographed from multiple perspectives to detect the Samsung brand. All images of the training set were created with handwritten bounding boxes around the brand and labelled with “samsung”. The strategy for detection was to apply brand detection only on already detected and forwarded smartphone detections. This prototype and its goal has been rejected as “out of scope” by the Technical Domain Specialist Georg Müller because the trained model was not able to detect brands in more than 50 per cent of images of Samsung Galaxy S6 Edge devices. The reason of this detection issue could not be identified clearly. One finding was that the model was trained with small sized images (however the largest size which can be configured in the model configuration file). Small sized images have the problem of being pixelated, as shown in the following Figure 22.

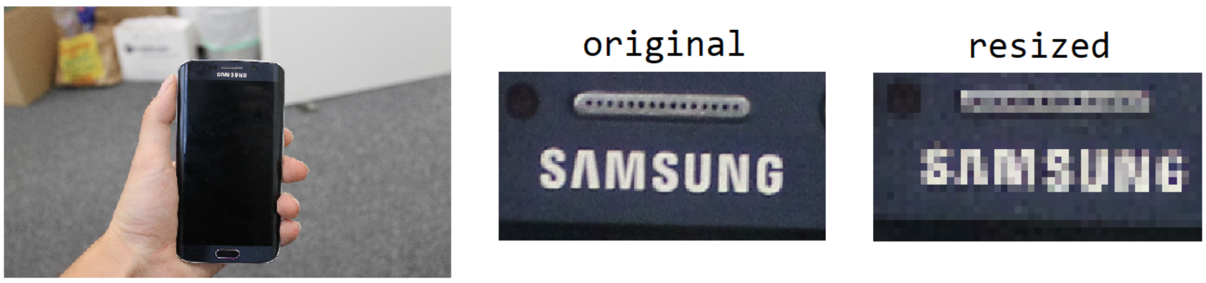


Figure 22: Prototype 3 - Brand Subclassification Pixelation

The image on the left shows the entire image with reduced size (for training). The image in the middle shows the contained brand in original quality. The third image shows the contained brand after the resizing step. We can see the pixelation in the third image. However, the third image shows a sample with better quality than the average quality of the training set.

5.4 Prototype 4 - Determine Dominant Colours

This section shows the progress of development of the “ColourExtractor”, a program to determine the major and minor colour of images of lost property items. The development was based multiple iterations which are represented by following subsections.

5.4.1 Iteration 1 - One Dominant Colour

Goal

The goal of this iteration is to determine the dominant colour of a given image.

Technology

The prototype is implemented as a .NET Console Application. The used image format is JPG.

Colour Palette

The RGB cube is divided into 3 x 3 x 3 = 27 clusters. All clusters have the same size. The centre of each cluster represents his cluster as RGB colour, therefor following colours could be chosen as dominant colour by the application:

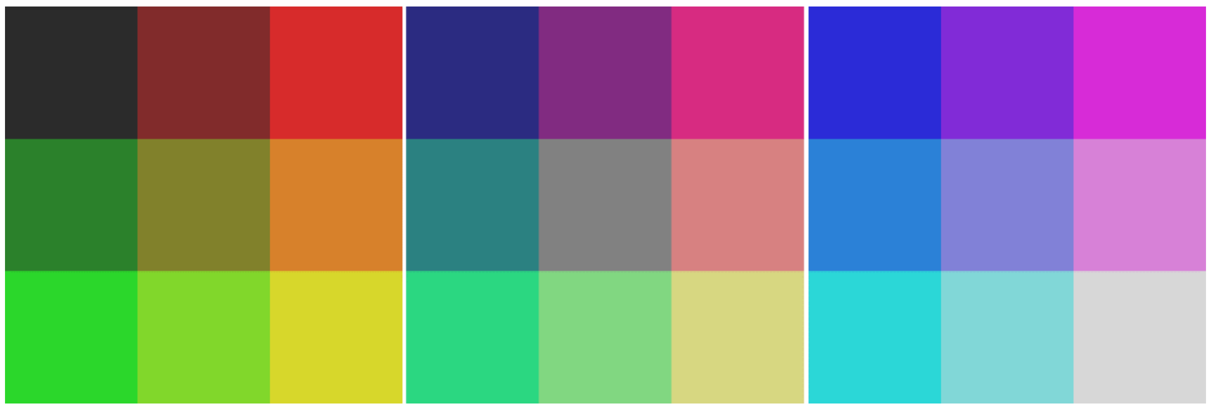


Figure 23: Prototype 4 Iteration 1 - Colour Palette

There is no white cluster because the centroid of the cluster in which white is contained is of RGB value (176, 176, 176) which is light grey. On the opposite side of the RGB cube, black is also represented as dark gray.

Area of Interest

The analysed area is defined by a quarter of the given image area, a rectangle centred in the middle with half the width and half the height of the image.

Algorithm

Each pixel in the area of interest is mapped to a cluster. The cluster which got most pixels mapped is chosen as the dominant cluster. The dominant colour is calculated as the centred RGB-value of the dominant cluster.

Application Runtimes

The total runtime of the application to analyse an image, determine the dominant cluster and colour and save the resulting image was below 100 milliseconds for the following environment:

- Given image size: 1000 x 1000 pixels
- Result image size: 100 x 100 pixels
- Tested samples: 25 images

An image recorded by a Samsung Galaxy S6 Edge device with a size of 4000 x 3000 pixels has also been evaluated and took 2.0 seconds of calculation time. As an improvement for large sized images a filter was developed which took only every fifth pixel. This improvement does not reduce the quality because the result converges to the unfiltered result; in the given case the mean of 600 000 equidistant RGB values in the area of interest converged against the original 3 000 000 RGB values for the tested image in an epsilon environment below 3 for the RGB distance of $d(a,b) := \sum_{i=1}^{n} |a_i - b_i|$.

Results

The following Figure 24 shows 5 sets (columns) of tested images and the corresponding result image from top to bottom.

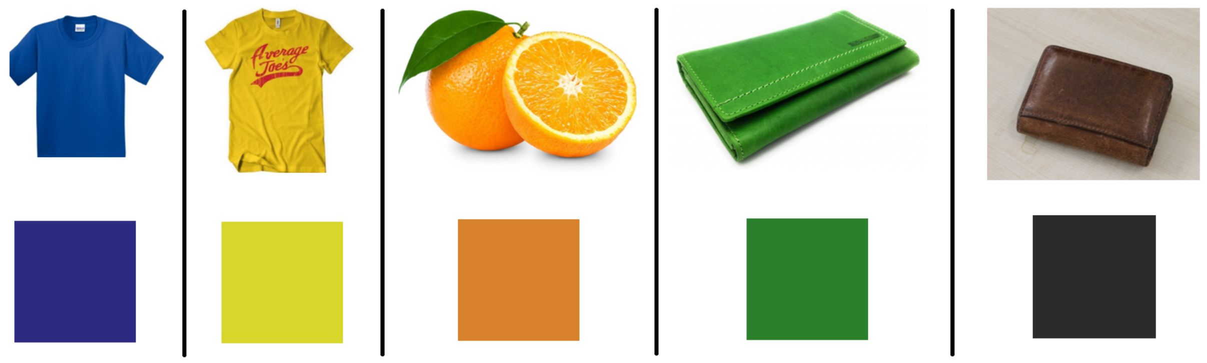


Figure 24: Prototype 4 Iteration 1 - Input / Output

We see that the brown wallet is determined as black; this is because the majority of pixels were mapped to black rather than the second colour in the colour palette as shown in Figure 23.

Conclusion

The first iteration has been designed as a base for the “ColourExtractor”, as it enables comparison of a test set to compare further version improvements.

5.4.2 Iteration 2 - Minor and Major Colour

Goal

The goal of this iteration is to determine the minor and major dominant colour of a given image.

Algorithm

This implementation extracts two dominant colours if the second one reaches more than 5 % area of the first one. The source code of the first iteration was extended by, at the final step of the algorithm, choosing the topmost two clusters.

Result

The output of this iteration has changed from a box, showing the one dominant colours, to a bar. The bar shows the major colour on the left side and the minor colour (if any) on the right side. The line between both areas represents the ratio of the cluster sizes. The minor colour area has a minimum size to improve the recognisability. The following Figure 25 shows a screenshot with tested images and the result of the application.

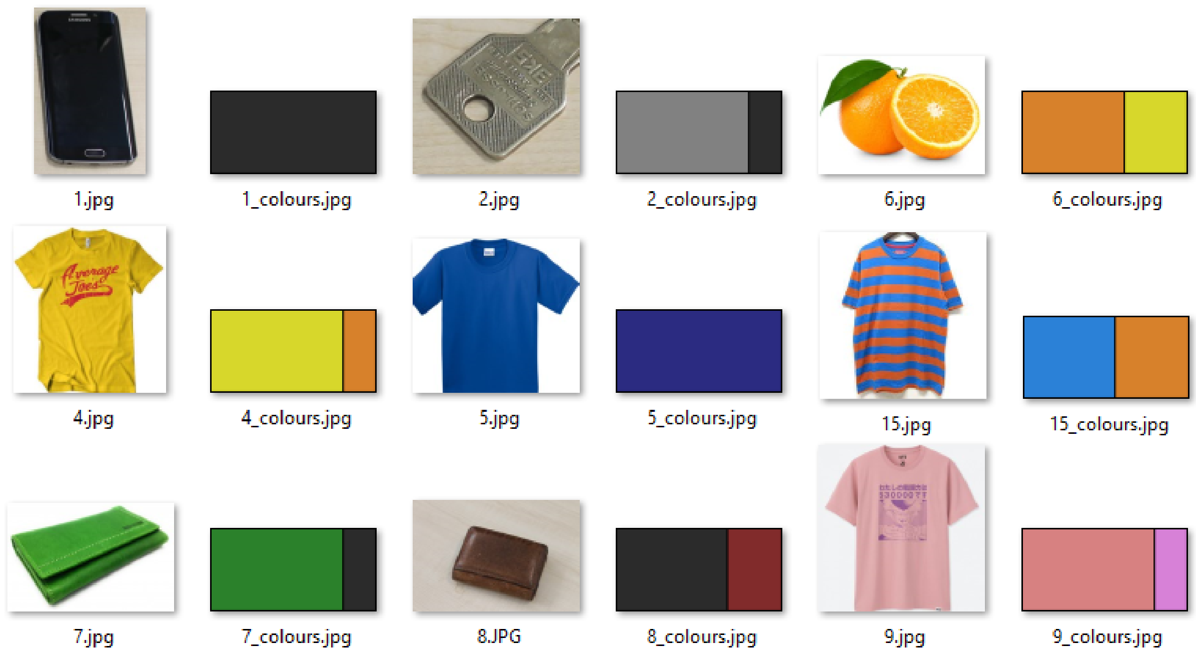


Figure 25: Prototype 4 Iteration 2 - Input / Output

25 images were tested. The algorithm works well for t-shirts because the output colours are understandable for humans, as decided at a presentation of the results in a meeting with Technical Domain Specialist Georg Müller. The blue t-shirt in the second row has only one colour as a result, while the output of the regularly striped t-shirt in the second row has two colours with a 50-50 ratio. One finding of the discussion of the presentation of these results was that

the output colours do not match the colour of the item because the palette is static. So it was decided to directly use RGB values as output instead of the static palette, which is content of the following subsection Iteration 3 - Improvements.

5.4.3 Iteration 3 - Improvements

Goal

The goal of this iteration is to determine the minor and major dominant colour of a given image as RGB values.

Algorithm

An algorithm was developed based on the K-Means Clustering Algorithm, inspired by “Least Squares Quantization in PCM” [14], which uses an algorithm to cluster data points. The developed algorithm is designed to find two clusters in the RGB colour space colour data of given “.jpg”-images. The algorithm consists of two looped steps:

- Assign each data point to the nearest cluster
- Update the centroid of each cluster

The two centroids converge to the minor and major RGB value. The developed program returns RGB values of both centroids.

Area of Interest

In this iteration, the area of interest was adjusted. Instead of a rectangle area of interest, the 1-norm was used. The 1-norm is defined as

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \ .$$

For a better fit on rectangular objects which are recorded from an angle of 45 degrees, the 1-norm was an improvement which doubles the area that a rectangle would have resulted in. The following Figure 26 shows the previous strategy compared to the new strategy.

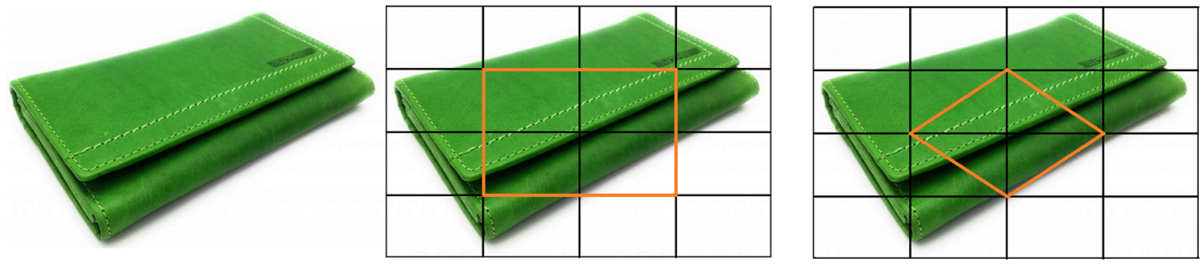


Figure 26: Prototype 4 Area of Interest

The original images are on the left side. In the second and third picture, the areas of interest are marked with orange lines. The strategy of iteration 1 is shown in the middle. The problem for rectangle areas is that background is included in the area, as seen in the second image here. The image on the right shows the new strategy with the 1-norm; every pixel in the area is content of the item and not content of the background. This strategy was tested with 42 images of smartphones, keys and wallets. The result was that 93 % of tested images with this strategy captured the area of items without touching the background. In the 3 remaining tested images, the area of background was below 3 %.

Output

The output was changed to show the original image on top of the colour bar containing the minor and major colour. Samples are shown in the next section.

Results

The result of this prototype is the console application “ColourExtractor” which takes the path to an image as input and saves a result image to the same parent directory location. Result images contain the original image and a colour bar which describes the minor and major colour of the shown item and the ratio between them.

The following Figure 27 is a screenshot showing 15 result images of the application. 44 images were analysed in total, containing partly pictures of random items taken at the RUBICON office and partly images from the internet.

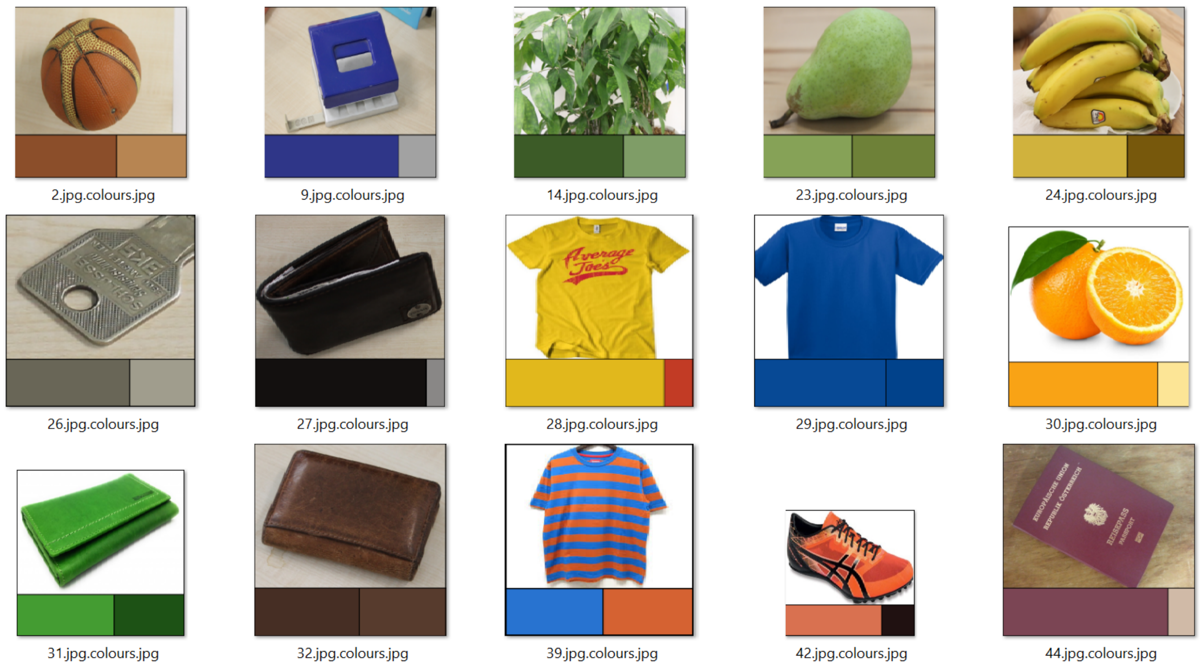


Figure 27: Prototype 4 - Colour Determination Results

After a presentation in a meeting with Technical Advisor Christian Kenngott and Technical Domain Specialist Georg Müller, these results were judged to be sufficiently good.

5.5 Prototype 5 - Material Classification

Material classification means to determine the material of lost property items just by a given image.

5.5.1 Idea

Following materials were identified in a brainstorming session as possible goals for material classification:

- Metal
- Wood
- Textile
- Plastic
- Leather
- Fur
- Glass

The idea is to improve colour predicates. For example, if we can determine that an item is made out of metal, we would exchange colour classification as follows:

- yellow -> gold
- grey -> silver
- brown -> bronze

Another example is if we can determine that an item is made out of wood, to map brown -> wood. For materials that can be determined as glass, the strategy is to deny colour information.

5.5.2 Laboratory Conditions

A training set could not be created because the following questions were not answered:

- What distance between the camera lens and item/material should be chosen?
- What size/resolution of training set images should be chosen?
- What sharpness of training set images should be chosen?
- How to train the features of glass?

5.5.3 Conclusion

Without training set and related literature about material classification, the business value was declared as too low against the cost of implementation in the course of a meeting with the Technical Domain Specialist Georg Müller.

5.6 Prototype 6 - More Item Classes

This prototype shows how the training set was extended from 4 to 7 item classes.

5.6.1 Goal

The goal of this prototype is to train a model to classify 7 different item classes. The model shall be trained with a pre-trained model as a base. The item classes are as follows:

- Wallet (wallets)
- Phone (mobile phones)
- Keyring (a keyring with more than one key on it)
- Key (single keys)
- Glasses (optical glasses and sunglasses)
- Clock (wristwatches)
- Empty (no item is shown)

The training accuracy must reach 90 per cent or more (the higher the better). This shall be validated by TensorBoard summaries.

5.6.2 Training Set

Three different meeting rooms at the office were used at three corporate events to extend the training set with images of keyrings, glasses and wristwatches. 1736 images were collected in total. All recorded items were made freely available by participants of the event. It was tried to record the same count of images for every item class. Since everyone has a smartphone at hand (company rule) but not everyone has glasses or a wristwatch, the counts differ in a range from 126 (glasses) to 476 (single key). The image counts are shown in following Figure 28.

Class Name	Count
key	476
phone	311
wallet	291
keyring	227
empty	159
clock	146
glasses	126

Total:	1736

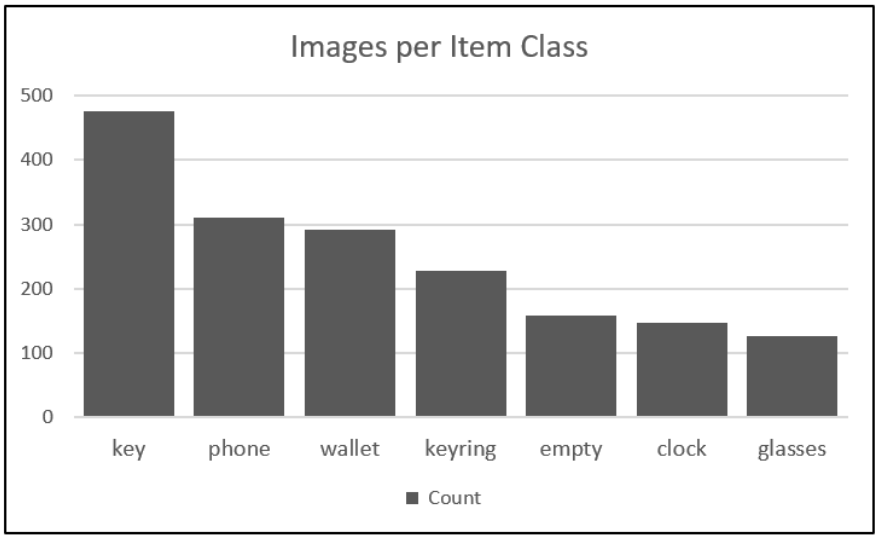


Figure 28: Prototype 6 - Item Classes

The training set was created in 3 hours with the following different devices:

- Nikon DSLR
- Samsung Galaxy S6 Edge
- Moto 4 Smartphone

For all pictures, as already tested in prototype 1, different angles, item rotations, item positions, distances and lights were used to make the neural network invariant against these variations. The following Figure 29 shows an excerpt of the training set with 3 images per item class:

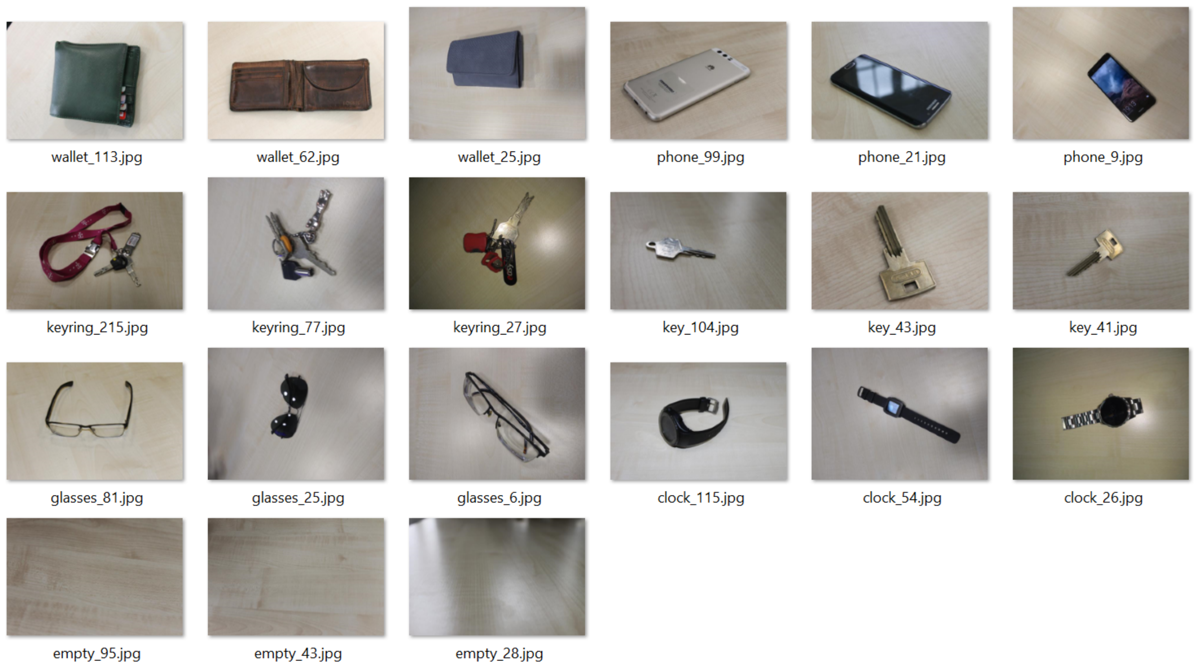


Figure 29: Prototype 6 - Training Set Excerpt

The figure shows the variations for each class. Besides the differences described above, following variations were trained per item class:

- Wallets: open/closed; different colours
- Phones: with/without reflexions; display on/off; with/without phone cover; different colours
- Keyrings: different cout of keys; with/without charms; with/without lace; different colours
- Keys: colours “gold”, “silver” or “’bronze’
- Glasses: open/closed; optical glasses and sunglasses (black or clear glass); different colours
- Clock: rolled in/out; digital/analog models; with/without reflexions; different colours
- Empty (no item is shown): with/without reflexions

5.6.3 Training

Inception v3 was used as a pre-trained model as a base (training checkpoint) for the training of this prototype. The training was running for 6 minutes and 43 seconds and included 5 000 training steps in total.

Following figures, Figure 30 and Figure 31 show screenshots from TensorBoard about accuracy and cross entropy over training steps. Graphs were smoothed with the TensorBoard option “Smoothing 0.9”. The brighter coloured graphs show the original values.

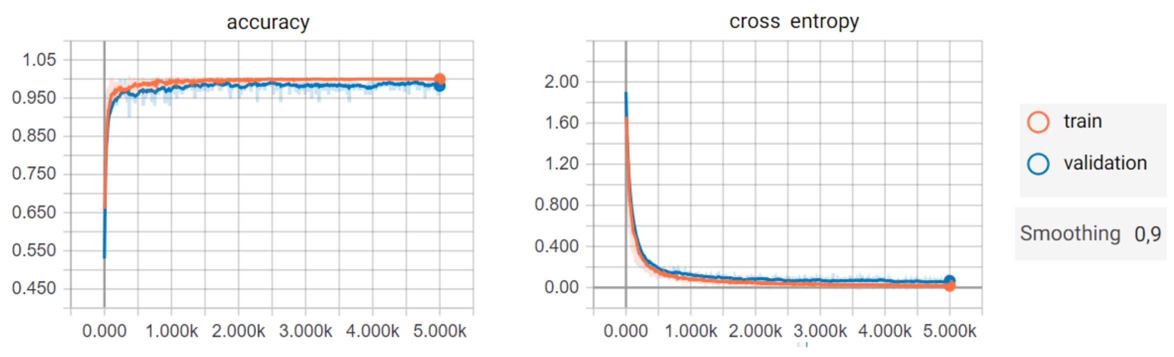


Figure 30: Prototype 6 TensorBoard Analysis

After 5 steps of training (below 1 second) and a training accuracy of 66 %, the first validation was logged with a validation accuracy of 53 %. The 90 % mark was reached after 80 training steps (6 seconds) for both training accuracy and validation accuracy. So, the major prototype goal was reached after 6 seconds. As the goal details say “90 % accuracy, the higher, the better”, the training was continued and only stopped after 5 000 training steps (6 minutes and 43 seconds) with a training accuracy of 100 % and a validation accuracy of 99 %.

The following Figure 31 shows the same data, but with the TensorBoard option “Ignore outliers in chart scaling” enabled.

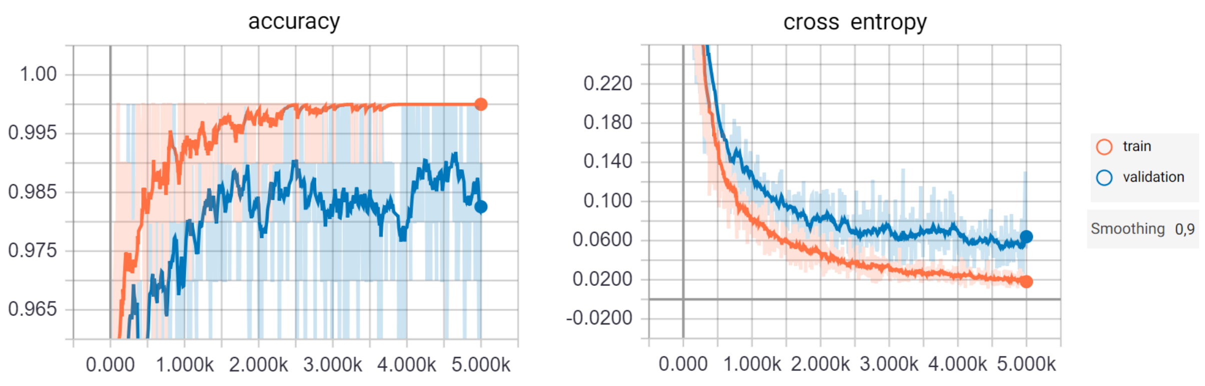


Figure 31: Prototype 6 TensorBoard Analysis

The scale is reduced to an interval of [0.965, 1] for accuracy and an interval of [0, 0.22] for cross entropy. So we can see the progression in detail.

5.6.4 Results

The model was tested with a special sample accuracy test, additionally to the TensorBoard analysis. The sample test set contained 7 item classes with 6 images per item class, 42 images in total. The following Figure 32 shows a screenshot of all tested images.

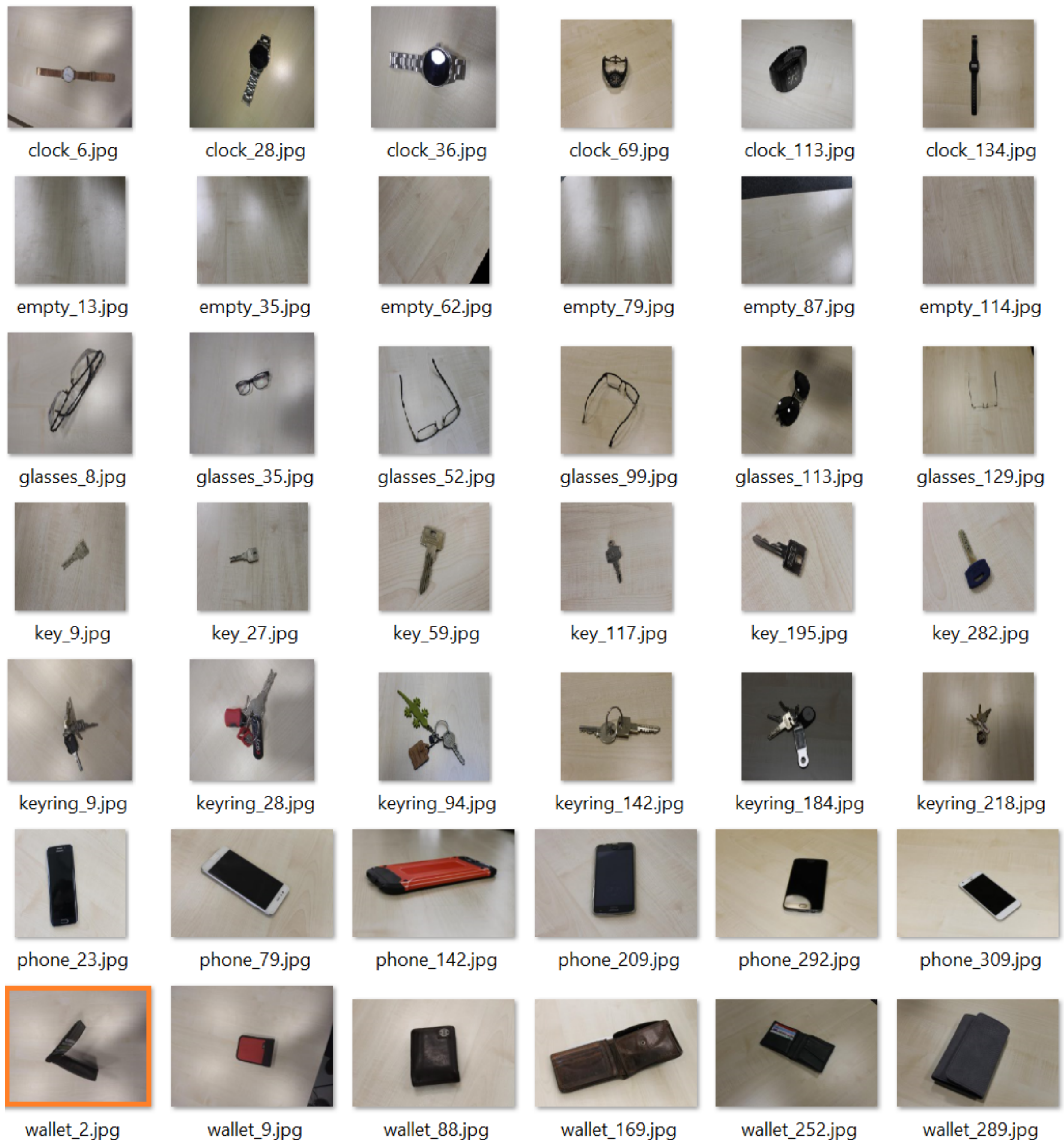


Figure 32: Prototype 6 Sample Accuracy Test Set

The result is that 41 of 42 images are classified correctly. This results in a sample accuracy of 97.6 per cent. In Figure 32 the one misclassified image is “wallet_2.jpg”, marked with an orange box. This wallet is classified as phone with a score (confidence) of 57 per cent, however the second guess of the model would have been correct with a score of 18 per cent.

The modal score of all 42 classifications is above 99 per cent. There are 4 outliers below 59 per cent (one of them is the misclassified wallet, the other three are classified correctly).

5.7 Conclusion

This is a conclusion over all prototypes of the prototype development phase.

The implementation of Prototype 1 shows that 4 lost property items can be successfully classified by images with up to 99 per cent of accuracy with a training duration of 3 minutes.

In Prototype 2, an object detection model was successfully trained to detect, plot and label smartphones, wallets and single keys. The model was strongly invariant against background variations in terms of structure, colour and different illuminations.

The experiment in Prototype 3 shows that subclassification (feature extraction on already classified lost property items) could not be applied without further work. The main problem found, was the fact that features of lost property items on images have a too small area to be recognised.

Prototype 4 and the successful implementation of colour information extraction can be used to determine the major and minor colour of lost property items. The algorithm works best for clothing and colourful items. Reflecting surfaces, concave items and items made of glass are problematic for proper colour descriptions. To execute a object detection iteration ahead of the colour detection is helpful to filter background.

The lesson learned on Prototype 5 is that material classification will not be part of this thesis, because no analysis could be performed due to missing data. However, material classification can be ssen as possibility of further research and future work.

Prototype 6 is an extension of Prototype 1 with more item classes. Up to 7 item classes can successfully be classified by the computer.

6 Mixed Intelligence

Mixed Intelligence is a strategy designed especially for this project. However, this strategy can be applied to many other artificial intelligence systems. A general problem with systems using artificial intelligence is that trained artificial neural networks do not keep up with the times. If we would have trained artificial neural networks to recognise mobile phones 20 years ago, how would they ever recognise mobile phones of nowadays? Mixed Intelligence is a strategy that solves exactly this problem. Human and machine go hand in hand.

6.1 General Idea

The general idea of Mixed Intelligence is to combine the advantages of Artificial Intelligence and Human Intelligence and let them work together to achieve goals that they would not be able to achieve on their own.

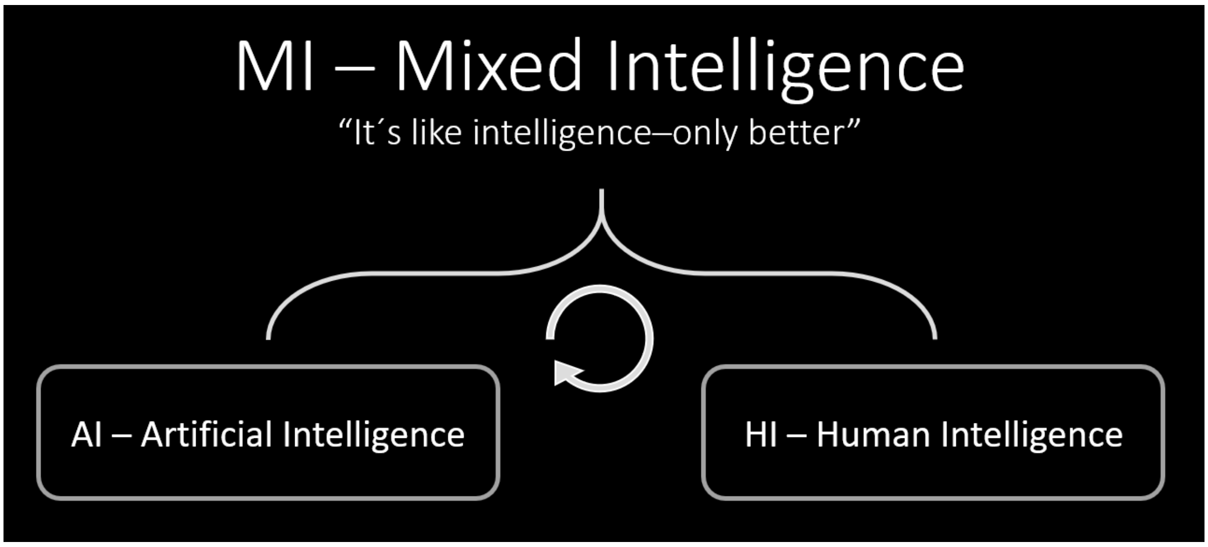

MI – Mixed Intelligence
"It´s like intelligence–only better"
AI – Artificial Intelligence
HI – Human Intelligence

Figure 33: Mixed Intelligence Overview

Figure 33: Mixed Intelligence Overview shows a representation of Mixed Intelligence. The round arrow in the middle symbolises the interplay between Artificial Intelligence and Human Intelligence which are the two components of Mixed Intelligence.

6.2 Mixed Intelligence for Lost Property Offices

With the strategy of Mixed Intelligence, the lost property item registration in lost property offices (by lost property officers) or in trains (by train conductors) can be realized as follows. The “actor” is a person who has to register a lost property item “item”. The “device” is either a computer or a mobile phone.

Firstly, with the device, an image of the item is made by the actor. This is an action that humans can perform with their hands. This action can be considered as Human Intelligence because the human brain controls the body. A computer without hands can not place items well yet.

Secondly, the computer comes to play. A trained artificial neural network classifies the image and returns textual and digital descriptions of the shown item. The task of description can be considered as Artificial Intelligence because of the used artificial neural network; furthermore, also a human would be considered intelligent to perform it (compare [3]). Artificial Intelligence does not have the problem of dictionary variety, which is the main advantage here.

Third, an algorithm determines colour information for the recognised item. All findings are returned and verified by the actor. Quick verifications are notThe actor has the possibility to correct the resulting descriptions if necessary. The item is finally registrated.

6.3 Advantages and Disadvantages

This section covers the identified advantages and disadvantages that are related explicitly to this thesis.

Identified advantages are:

- this strategy enables continuous learning of new item classes
- the training set grows per usage
- weak spots (classes) can be revealed and improved precisely

Identified disadvantages are

- human feedback interaction is needed (but is optional)
- object detection is not possible (users would have to draw bounding boxes)

7 Release Development “Third Ai”

This chapter shows how the thesis goal was finally reached. The developed solution combines all lessons learned from previous prototypes. The release development phase is the last phase of the used development model (see Development Model 1.3.1). The implementation part of the master project ends after this phase, and the developed solution is handed over to RUBICON in a final meeting with the Technical Advisor Christian Kenngott and the Technical Domain Specialist Georg Müller. Source code, documentation and artefacts related to the master project are handed over via a git repository. The master thesis is submitted to RUBICON after the final submission to Fachhochschule Technikum Wien.

7.1 Goals

The strategy of this summarising development phase is to combine all previously developed prototypes. The best prototypes are retained, unrewarding prototypes are rejected (decided by the Technical Advisor Christian Kenngott, the Technical Domain Specialist Georg Müller and the author Paul Puntschart).

The main goal is to provide a software library that can take an image as input. The library then classifies the given image and outputs human-readable text about shown lost property items. The release development shall be driven by a precending requirement engineering phase. Therefore, goal details are developed in the next section “Requirements”.

The following Table 6 shows which prototypes are retained or rejected, including a quick note from the author:

Prototype	**State**	**Note**
Prototype 1 - Image Classification	Retained	Main goal of this thesis; further extended in Prototype 6
Prototype 2 - Object Detection	Rejected	Rejected in favour of the Mixed Intelligence strategy
Prototype 3 - Subclassification	Rejected	Experiment did not show sufficient results
Prototype 4 - Determine Dominant Colours	Retained	Main goal of this thesis; adjusted to colour palette output
Prototype 5 - Material Classification	Rejected	Rejected because of too high implementation costs
Prototype 6 - More Item Classes	Retained	Main goal of this thesis; extension to Prototype 1

Table 6: Documentation of Retained and Rejected Prototypes

7.2 Requirements

This section shows the strategy to reach the thesis goals. The goal is derived into three levels of requirements, based on the Kano model [15]:

- Basic Requirements
- Performance Requirements
- Excitement Requirements

Each requirement is also linked to one of two tags:

- Application
- Function

An application requirement, e.g. “(Application) Support multiple items on one image” should be read as “The application must support multiple items on one image.”, while function requirements should be read as “The application must provide a function to . . . ”.

7.2.1 Basic Requirements

The basic requirements are as follows:

- (Application) Provide a user guide

- (Function) Receive information about the data set
    - Supported item classes
    - Number of images per item class

The user guide is necessary to set up and use the developed software library solution. Information about the data set are needed to maintain the training set.

7.2.2 Performance Requirements

There are two performance requirements:

- (Function) Generate a description for a given image regarding the item class
- (Function) Receive a description for a given image regarding item colours

The requirement of describing a given image has to reach a minimum accuracy of 90% to reach the thesis goal, the higher, the better.

7.2.3 Excitement Requirements

Following requirements are optional but highly valuable for future work:

- (Function) Process a given image for continuous training
    - Integrate an evaluated image to the data set
    - Trigger continuous training
- (Function) Receive information about evaluation statistics
    - Accuracy per item class
- (Application) Support multiple items shown on one image

7.2.4 Verification Strategy

To verify the result accuracy of classifying lost property items, a data set of 7 images per item class shall be analysed with the developed software library. Overall at least 90 % of the item class guesses shall be correct.

7.3 Technology

Following technologies are used for the released class library:

- TensorFlow: Image classification training

- Classification Model: The pre-trained model “inception-2015-12-05”
- ColorMine: Delta-E calculation with CIEDE2000
- .Net Solution: The Mixed Intelligence interface “Third Ai”
  - C# Class Library
  - Console Application
  - Test Project

7.4 Solution

The solution is a .Net Visual Studio Solution named “Third Ai”. It was developed in 7 days (56 hours).

7.4.1 ThirdAiLibrary Model

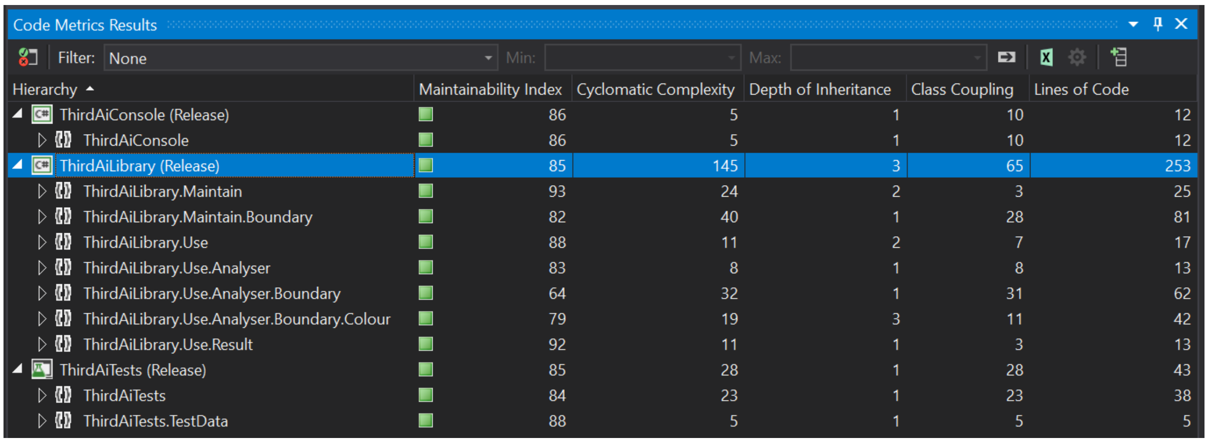
The ThirdAiLibrary is modelled with three components: Use, Maintain and Model. The component “Use” contains mainly classes that users directly need to use the library, such as a ClassAnalyser, a ColourAnalyser and a FileSystemScanner. The interface to use those classes is the class ThirdAi. ThirdAi contains the most important method “Describe”, which takes a path to an image as input and returns an object ItemDescription.

The component “Maintain” contains classes that administrators need for a training set insight. This component is also needed to extend the training set with new images for the existing item classes or new images for new item classes (the mixed intelligence strategy).

The component “Model” contains the trained artificial convolutional neural network and an interface to call the Python script “classify_image.py” with C# code.

7.4.2 Code Metrics

Code metrics of the solution “Third Ai” were analysed with Visual Studio. The following Figure 34 shows a screenshot of the results.



Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
ThirdAiConsole (Release)	🟩 86	5	1	10	12
ThirdAiConsole	🟩 86	5	1	10	12
ThirdAiLibrary (Release)	🟩 85	145	3	65	253
ThirdAiLibrary.Maintain	🟩 93	24	2	3	25
ThirdAiLibrary.Maintain.Boundary	🟩 82	40	1	28	81
ThirdAiLibrary.Use	🟩 88	11	2	7	17
ThirdAiLibrary.Use.Analyser	🟩 83	8	1	8	13
ThirdAiLibrary.Use.Analyser.Boundary	🟩 64	32	1	31	62
ThirdAiLibrary.Use.Analyser.Boundary.Colour	🟩 79	19	3	11	42
ThirdAiLibrary.Use.Result	🟩 92	11	1	3	13
ThirdAiTests (Release)	🟩 85	28	1	28	43
ThirdAiTests	🟩 84	23	1	23	38
ThirdAiTests.TestData	🟩 88	5	1	5	5

Figure 34: Visual Studio Code Metrics

The projects in the solution were judged with a maintainability index of 86 for the ThirdAiConsole and a maintainability index of 85 for both the ThirdAiLibrary and the ThirdAiTests.

7.4.3 Usage

This section shows how to use the developed solution with an image of a wallet as an example.

Wallet Example

Given the image of a wallet, see Figure 35, we want to use the developed solution “Third Ai” to determine which item is shown and how it looks like regarding the item colour.

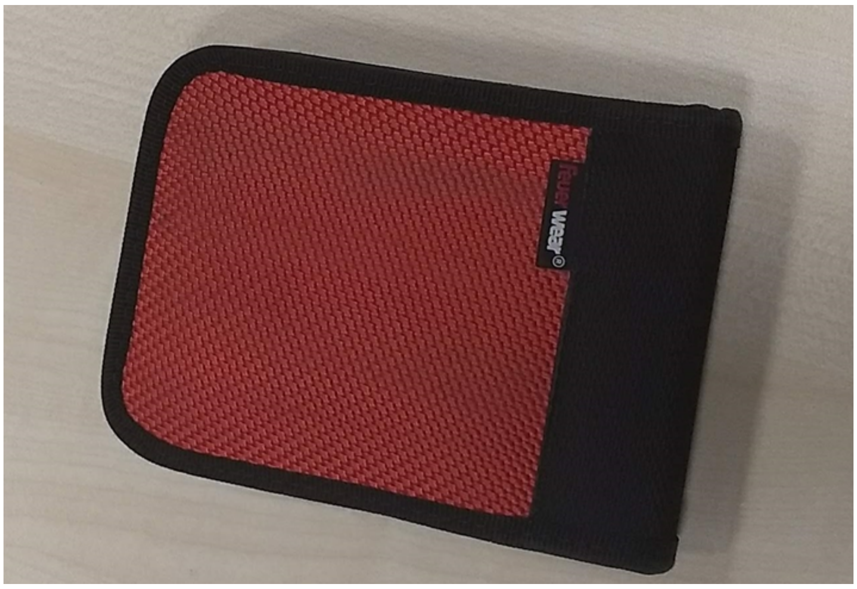


Figure 35: Image of a Wallet as Example

The following code snippet shows how to import and use the library in a plain created console application named “ThirdAiConsole”. All we need to know is the location of the image on the computer (pathToExampleImage in this code snippet).

```
using System;
using ThirdAiLibrary.Use;
namespace ThirdAiConsole
{
  public class Program
  {
    public static void Main (string[] args)
    {
      var pathToExampleImage = @"C:\tmp\wallet.jpg";
      var thirdAi = new ThirdAi ();
      Console.WriteLine (thirdAi.Describe(pathToExampleImage));
    }
  }
}
```

First, we import the library “ThirdAiLibrary.Use”. Second, we set up a variable pathToExampleImage with the location of the image on the computer. Third, we initialise a variable thirdAi with the class ThirdAi (see section ThirdAiLibrary Model 7.4.1). Finally, we call the method “Describe” with the image path as parameter and write the result to a console.

The program returns the following text:

```
The item class is ’wallet’.
The major colour is ’red’, the minor colour is ’black’.
```

8 Results

This chapter shows the results and features of the developed solution “Third Ai”. Firstly, the used item classes are explained. Second, the used item colours are shown. Third, the evaluated accuracy is tested against the thesis goal of reaching an accuracy of 90 per cent.

8.1 Item Classes

The solution “Third Ai” starts with a trained set of seven lost property item classes. The item classes are wallets, mobile phones, keyrings, single keys, glasses, wristwatches and also a class without item. Example images are shown in the following Figure 36: Item Class Examples.

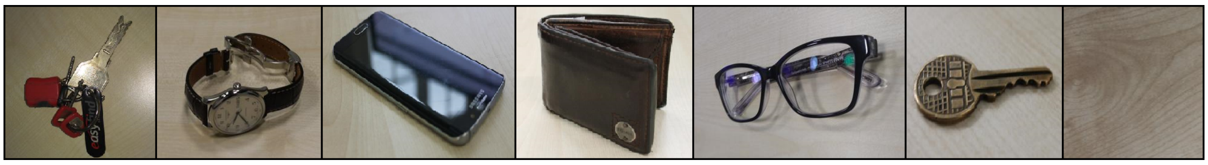


Figure 36: Item Class Examples

The developed solution provides the possibility to extend the item classes with new item classes. It is also possible to add correct classified images to the training set for further model training. So a lost property office can improve the power of the solution over time.

Following Table 7: Number of Object Classes per Development Phase summarises the progression of item classes in this thesis, regarding the task of classification.

Development Phase	**Object Classes**
Proof of Concept	2
Prototype 1 - Image Classification	4
Prototype 6 - More Item Classes	7
Release - Third Ai	7+

Table 7: Number of Object Classes per Development Phase

The number of item (object) classes is rated with 7+ for the developed solution “Third Ai”, because 7 classes are pre-trained and the number can be increased with the Mixed Intelligence strategy.

8.2 Item Colours

The solution “Third Ai” utilises a colour catalogue containing 13 colours. Following Figure 37 shows all of those colours.

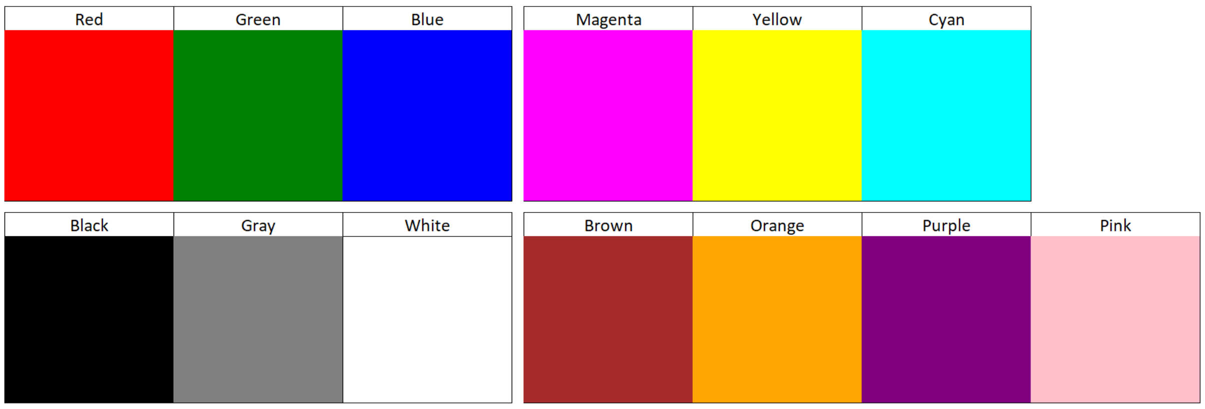


Figure 37: Colour Catalogue

Colours of lost property items are determined as major and minor colour. The major colour has more area in the image than the minor colour. If major and minor colour have the same catalogue colour, the result of “Third Ai” only shows the major colour.

The decision of which colour is chosen is based on the metric CIEDE2000. The nuget package ColorMine 1.1.3 (see [9]) is used to determine colour distances, and the nearest catalogue colour is taken as a result.

8.3 Accuracy

The accuracy of the solution “Third Ai” is defined as how many given images are classified correctly. The accuracy was measured as training accuracy and validation accuracy (evaluated by TensorBoard directly at the training time) as well as with a custom sample validation set. Achieved accuracies are shown in the following Table 8: Accuracy per Evaluation Type.

Evaluation Type	**Accuracy**
Training Accuracy	100 %
Validation Accuracy	99 %
Sample Accuracy	98 %

Table 8: Accuracy per Evaluation Type

The sample accuracy was used to verify the thesis goal of reaching an accuracy of 90 %. This goal was fulfilled successfully.

9 Conclusion

This thesis shows an approach to classify lost property items for lost property offices. Lost property items are classified via images. The sample accuracy (how many items are classified correctly) reached 98 %, which shows that image classification with TensorFlow is a possible approach for lost property offices. The following sections are conclusions about realised features and two possible ways to use the developed software library “Third Ai”.

9.1 Lost Property Item Description

Lost property items can be described via given images if we use an item class set of 7 item classes (containing an empty class with no item shown). Item description is a task that computers can do together with humans using a strategy called Mixed Intelligence (see chapter 6 Mixed Intelligence). With this approach, a computer can automatically generate pre-filled item description forms.

Forms can be reviewed by lost property officers to improve computer item class knowledge continuously. New item classes can be trained. The accuracy can be improved with new images of items without taking action as reviewer automatically.

9.2 Item Descriptions with Permanently Installed Cameras

A possible way to record images of lost property items is to use a permanently installed camera. Lost property officers could use a simple plain desk or a wooden desk as background of a recording station. An officer would have to position an item on that desk and push a button to capture a picture of the item. The so created image can be piped to “Third Ai” to get a text representation of item classification and colour determination results. This text can be reviewed directly at the station and optionally adjusted to fit the human evaluated ground-truth. If the description is reviewed, it can also be used to improve computer knowledge and accuracy.

9.3 Item Descriptions with Smartphones

This thesis shows that it is possible to record images with a smartphone and then to ask a sofware library which lost property item of a known item class set is shown. The resolution of images taken by smartphones nowadays is high enough (greater than the resolution of training data images of 600 x 400 pixels used in this project). The runtime of the developed software library function is below 2 seconds to classify a smartphone image.

The result of the library can be used to create a description of the lost property item, containing item class name probability distribution information and a colour description with major and minor colour chosen from a catalogue of 13 colours.

10 Discussion

To describe lost property items is a very personal issue and this thesis shows how this subjectivity can be reduced by working together with a computer. Using Mixed Intelligence is an innovative strategy that helps to keep the solution up to date and to improve the solution over time. Although the main goal of this thesis is reached successfully, there are some edge cases and limitations covered in the following section followed by another section about the business value of Mixed Intelligence and the developed software library “Third Ai”.

10.1 Edge Cases and Limitations

Edge cases are situations that appear with low propability. To handle them takes a lot of time but has only a small area of improvement. Identified edge cases are listed in this section. They were found and documented during the implementation of the master project.

10.1.1 Images with Multiple Items

The developed solution does not support images which show multiple lost property items on it. This decision was made to enable the strategy of Mixed Intelligence. However, with a given training set it is possible to support multiple items on one image. That training set would just have to contain metadata with labelled bounding boxes. In this project, it took 4 hours to draw bounding boxes for 642 items.

An experiment has shown that it is possible to use the training data set from Prototype 2 (see section Training Set 5.2.2)) to train the general task of detecting general objects. In a pipeline, detected objects can then be classified and described in an extra step. This would have the advantage that object detection and multiple items on one image can be combined with Mixed Intelligence. It was also found, during the task of drawing bounding boxes, that correcting a guessed object bounding box is a lot easier than drawing the bounding box from scratch.

10.1.2 Overlapping Items

In addition to the previous limitation, overlapping items are not supported by the developed solution “Third Ai”. As shown in section Edge Cases and Misclassifications 5.2.6, overlapping items can be a topic of interest for key bounds or a bag which is photographed with items found inside laying onto that bag.

10.1.3 Item Colour Issues

In some cases, item colours cannot be determined in a meaningful way. If lost property items are coloured with three or more main colours, the developed solution can only determine two of them. If we have an image of a reflecting item, for example, the turned-off display of a smartphone, the colours of the reflexion are detected instead of the colour of the item, which in this case would be a black display and not the plafond.

10.2 Business Value

The business value of Mitxed Intelligence for lost property offices can be estimated by answering following questions. However, in this work no estimation was performed.

- How much time can we save at lost property registrations?
- How much complexity can we reduce at lost property registrations?
- Do other competitors use artificial intelligence?

11 Future Work

This chapter is about the possibilities of future work based on this thesis. Each of the following sections covers a way to improve or extend the developed solution “Third Ai”. All those suggestions are related to the domain of lost property offices. However, one recommended future work not covered here is the use of Mixed Intelligence in other projects and domains.

11.1 More Item Classes

The developed solution “Third Ai” enabled the possibility to train new item classes. The Technical Domain Specialist Georg Müller rated following lost property items “of great value”:

- Items with a high finding rate
- Items with a high value (price)
- Items with a high personal value (e.g. a children´s toy)

Due to the probability of occurrence, items with a high finding rate will automatically become part of the training set over time. This is why the Mixed Intelligence strategy is so valuable. Other item classes should be added manually. The main question is who decides which items are of high personal value. This can may be determined with a survey at the process of handing over lost property items at lost property offices.

11.2 Higher Accuracy

The accuracy, which is already an accuracy of 98 %, can be approved. This can be achieved with different approaches. One way is to add more training set data to the developed software library. Another way is to optimise training parameters (hyperparameters like the learning rate, optimiser, etc.). Other models of deep convolutional neural networks were not tested in this project but may also lead to acceptable results.

11.3 More Item Features

An adequate extension of the developed description would be not only to classify item class names and to determine colours but also to collect other item features. This has already been tried in this project but without proper success. Additional item features would be features that support to identify lost property items, like brands, logos, damages or any feature that reduces abstraction and make an item description more concrete.

11.4 Software Integration

The developed solution “Third Ai” can be integrated into lost property software as is. However, it is recommended to use the latest versions of TensorFlow and modern hardware if possible. Research of artificial neural networks is progressing very fast these days, offering new approaches almost every day.

11.5 Retry of Rejected Prototypes

Some prototypes were not used yet. The developed solution does not include object detection, material classification or colour discription without a colour catalogue. To retry and solve rejected prototypes is a highly recomennded task to start from if the developed solution “Third Ai” is at hand.

Bibliography

[1] J. McCarthy, M. I. Minsky, N. Rochester, and C. E. Shannon. *A proposal for the Dartmouth summer research project on artificial intelligence,* 1955, John McCarthy (1996) [Online] Available: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html (Date last accessed April 28, 2019).

[2] J. Haugeland. *Artificial intelligence: The very idea,* MIT Press, Cambridge, MA, 1985.

[3] E. Rich, K. Knight, and S. B. Nair. *Artificial intelligence,* Tata McGraw-Hill, New Delhi, 3rd edition, 2009.

[4] C. Kumah, N. Zhang, R. K. Raji, and R. Pan. *Color Measurement of Segmented Printed Fabric Patterns in Lab Color Space from RGB Digital Images,* Journal of Textile Science and Technology, 5, 1-18., 2019. Available: https://doi.org/10.4236/jtst.2019.51001 (Date last accessed April 29, 2019).

[5] J. W. Goethe. *Zur Farbenlehre,* Cotta, Tübingen, Bd. 1, 1810.

[6] C. L. Eastlake. *Goethe's Theory of Colours: Translated from the German; with Notes by Charles Lock Eastlake,* John Murray, London, 1840.

[7] I. Newton. *Opticks: or, a treatise of the reflections, refractions, inflections and colours of light. The fourth edition, corrected. By Sir Isaac Newton,* Knt. London, Harvard, vol. 18, Printed for William Innys, 1730.

[8] TensorFlow [Online], Available: https://www.tensorflow.org/ (Date last accessed April 30, 2019).

[9] ColorMine [Online], Available: http://colormine.org/delta-e-calculator/Cie2000 (Date last accessed April 30, 2019).

[10] K. A. P. Perichappan. *Greedy Algorithm Based Deep Learning Strategy for User Behavior Prediction and Decision Making Support,* Journal of Computer and Communications, vol. 6, no. 6, 2018.

[11] How to Retrain an Image Classifier for New Categories, TensorFlow [Online], Available: https://www.tensorflow.org/hub/tutorials/image_retraining (Date last accessed April 30, 2019).

[12] Gallery of Borderline Country Lordana and Borderline Country Tiara, Alina Gaugg [Online], Available: https://www.mileysworld.at (Date last accessed April 30, 2019).

[13] TensorBoard [Online], Available: https://github.com/tensorflow/tensorboard/blob/master/README.md (Date last accessed April 30, 2019).

[14] S. P. Lloyd: *Least Squares Quantization in PCM,* Transactions on Information Theory, vol. 28, pp. 129–137, 1982.

[15] N. Kano, S. Nobuhiku, T. Fumio, and T. Shinichi. *Attractive Quality and Must-Be Quality,* Journal of the Japanese Society for Quality Control, Japan, 1984.

[16] Uber Lost Found Wien Skurrilsten Fundstuecke, HEUTE (March 16, 2018) [Online], Available: http://www.heute.at/oesterreich/wien/story/Uber-Lost-Found-Wien-Skurrilsten-Fundstuecke-40590983 (Date last accessed April 16, 2019).

List of Figures

List of Tables