M.Sc.Thesis
Master of Science in Engineering

Machine Learning for Automated Decision Making under European General Data Protection Regulation

- A Case Study of Vehicle Registration at SKAT

Anders Launer Bæk

Kongens Lyngby 2019

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

M.Sc.Thesis
Master of Science in Engineering

Machine Learning for Automated Decision Making under European General Data Protection Regulation

- A Case Study of Vehicle Registration at SKAT

Author:
Anders Launer Bæk

Supervisors:
Line Katrine Harder Clemmensen, Associate Professor at DTU
Rasmus Moseholm, Senior Data Scientist at UFST

Start date: 1st of August 2018
Submission date: 1st of February 2019

Anders Launer Bæk
Kongens Lyngby, January 28, 2019

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Dedicated to my beloved daughter Agnes, my soulmate Laura and my family.

(this page is intentionally left blank)

Summary

The objective of this graduation thesis entitled Machine Learning for Automated Decision Making under European General Data Protection Regulation - A Case Study of Vehicle Registration at SKAT, is to analyse the end-to-end process of the manual validation task for vehicle registration documents (VRD). The task in hand is to develop and present a suitable pipeline able to perform a validation task by the usage of document classification and extraction of text sequences within VRDs. The classification and the extraction are referred to as the first stage and the second stage of the suggested pipeline respectively.

This thesis investigates several usable models for the classification and text extraction. The performances for these tasks have each been evaluated with a single metric; the Cohen's Kappa metric which is applicable for imbalanced class distributions and the Levenshtein Distance metric, which is useful for describing the accuracy of a predicted sequence. Frameworks for interpreting artificial neural network models have been studied in order to develop a GDPR compliant pipeline.

The achieved results of this thesis demonstrate the utility of transfer learning for deep neural networks. This is useful when the number of training samples is small and when the pipeline requires the strong prediction power of deep neural networks. The neural network architecture for document classification is based upon the convolutional neural network blocks of the VVG16 model with a small non-linear classifier on top. The model for text extraction is composed of a few convolutional neural network blocks, recurrent neural network blocks and a non-linear classifier block.

Finally an investigation of the assigned probabilities of the correct predictions and of the wrong predictions have been examined in order to determine the thresholds of controlling an appropriate trade-off between accepted misclassifications and manual processing.

The mentioned models, the appropriate pre-processing methods and a business logic module have been composed into a recommended technical pipeline, which is part of the entire end-to-end automatic system.

The recommended technical pipeline is accessible here: Github repository [1].

Acknowledgements

First and foremost I wish to express a special gratitude to my research supervisor, Associate Professor Line Katrine Harder Clemmensen. Her guidance and useful critique has made me increasingly self-reflective throughout the research period. Her ability to supervise, to be constructive and her ongoing positive attitude at every meeting has encouraged me to enjoy the ride of composing a thesis of which I am grateful.

I would also like to acknowledge the support provided by the study group, where discussions and exchanges of professional experiences (not to mention a weekly safe space with cake) has helped me to move forward and kept my spirit high.

Advice and support given by technical supervisor Rasmus Moseholm has been greatly appreciated. His skills on technical difficulties and his encouragement on given challenges has provided me with very valuable feedback and pushed my limits. Rasmus has always been accessible and whenever I needed help, his guidance has been present and always friendly.
I would like to thank the general support and sparring provided by my colleagues in UFST. Especially Senior Data Scientist Ulrik Lindved Clausen has been standing by me and encouraged me to think with a holistic view on solving emerged difficulties.

Finally, I wish to express my profound gratitude to my Laura and to the rest of my family whom have supported, provided feedback and encouraged me to be diligent throughout my years of study. This accomplishment would not have been possible for me without their presence and them being who they are.

Thank you.

Anders Launer Bæk

Nomenclature

ANN Artificial Neural Network

CNN Convolutional Neural Network

CRNN Convolutional Recurrent Neural Network

CTC Connectionist Temporal Classification

CV Cross-validation

ED Edit Distance

EDA Exploratory Data Analysis

EEA European Economic Area

GDPR General Data Protection Regulation

Grad-CAM Gradient-weighted Class Activation Mapping

GRU Gated Recurrent Unit

HOG Histogram of Orientated Gradients

LD Levenshtein Distance

LER Label Error Rate

LIME Local Interpretable Model-agnostic Explanations

LP License Plate

LSTM Long Short-Term Memory

MOTOR Motorstyrelsen

PCA Principal Component Analysis

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

RPA Robotics Process Automation

TIC Technical Inspection Center

UFST Udviklings- og Forenklingsstyrelsen

VIN Vehicle Identification Number

VRD Vehicle Registration Document

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Using machine learning and robotics to undertake monotonous tasks hereby providing supported or fully intelligent data driven decisions and relocating resources is a high priority of the Udviklings- og Forenklingsstyrelsen (UFST).

This case study is a joint venture between UFST and Motorstyrelsen (MOTOR) and is a real world challenge, which at present stage is solved by a manual workflow. The thesis deals with an automation of an end-to-end process for vehicle registration document (VRD) validation from vehicles imported to Denmark.

A successful technical pipeline will be able to extract features of interest from the VRDs in order to automate the end-to-end process and provide a final recommendation for a technical General Data Protection Regulation (GDPR) compliant flow.

1.1 Research Problem

The main objective of this case study is to research and develop an appropriate technical pipeline capable of extracting relevant features from the VRDs. Several models will be trained, assessed and evaluated. The technical pipeline should be able to return three features from a VRD (figure 1.1). These features are mentioned in section 1.2.

The following research question has been carried out to secure a holistic view of this particular case study:

Is machine learning able to undertake the process of VRD validation and to what extent?

The above mentioned research question is supported by the following design questions:

- Are there any models used prior to this study recognized to solve similar tasks?
- Which of the models in the technical pipeline yields best performance?
- Which of the models in the technical pipeline yields best ability of interpretation?
- Is the final technical pipeline GDPR compliant?

1.2 Project Scope

The project scope deals with the input and the required output for a complete automation of the manual validation task.

The bullets below and figure 1.1 provide a description of the input and the output of the technical pipeline. By extracting these three features it is possible to undertake an automatic validation of the incoming VRDs.

- Input: Scanned VRDs in PDF format
- Outputs:
  1) Two-letter country code (ISO 3166-1 alpha-2 format)
  2) License plate (LP)
  3) Vehicle Identification Number (VIN)
  4) *New Danish license plate*
  5) *Initial date of registration in Denmark*[1]

Output 4) and output 5) do not appear in the original content of the VRD and are therefore excluded from the scope of this project. It is possible to extract output 4) and output 5) from the internal data warehouse by knowing output 3).

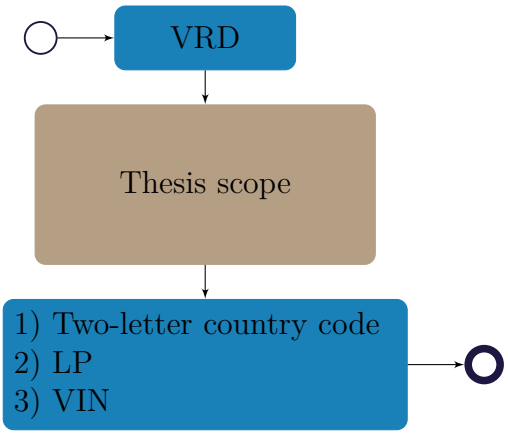


Figure 1.1: This illustration clarifies the scope of this thesis. The technical pipeline should be able to process and extract the following three features: 1) Two-letter country code, 2) LP and 3) VIN of VRDs from EU and EEA countries.

[1]Vehicles can be re-registered or issued a new license plate after the first registration in Denmark.

1.2.1 Demands and Delimitations

The following demands and delimitations have been agreed upon in order to limit the scope of this thesis:

- VRDs from outside EU and European Economic Area (EEA) countries need an origin classification in order to be dispatched to the manual queue. The majority (95%) of the VRDs are from EU and EEA countries.
- The selected technical pipeline and models need to fulfil an ability of interpretation in order to be GDPR compliant.
- Deployment of the technical pipeline is outside the scope of this thesis.
- The technical pipeline must be written in the language `R` due to the deployment platform.
- The technical pipeline must limit the number of additional `R-packages` and use `Packrat` (dependency management system for R) in order to be reproducible, tracking the dependencies and lowering the technical depth.

1.3 Research Methodology

This thesis encounter a problem orientated task which qualifies this research as a *quantitative case study* and will end up making a final recommendation for solving this task.

Several framework exists, which can be used as guidance in order to complete the tasks within the time limit. There are great similarities between software and model development and it is therefore attractive to benefit from the past two decades of experience within software development. The main objective of agile frameworks is to provide instant and continuous value for the end user through a flexible development process.

During the past decade such agile frameworks have been adapted into the area of machine learning. The Cross-Industry Standard Process for Data Mining (CRISP-DM) has proved its capabilities and the framework is illustrated in figure 1.2.

Central to the CRISP-DM framework is the importance of an iterative process and the interrelation and dependencies between each of its six phases. The sequence between each of the phases is not strict and most importantly is the ability to switch between phases as new knowledge is gained.

The deployment to production phase is outside the scope of this case study.

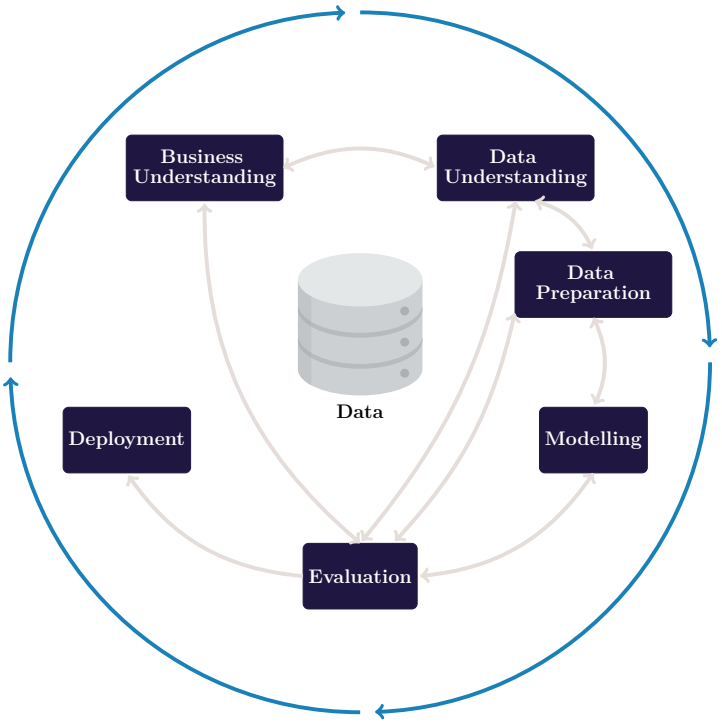


Figure 1.2: The CRISP-DM framework is illustrating the iterative approach for data mining. This approach has been used to master the iterative research process. This illustration is inspired by [2, fig. 2].

CHAPTER 2

Vehicle Registration Documents

This chapter provides an introduction to the case of interest and its entire end-to-end flow. A preview of the data foundation will be presented followed by an exploratory data analysis providing the essential insight of the data foundation.

2.1 The Case of Interest

Figure 2.1 illustrates a flow of the end-to-end process for a VRD when a vehicle is imported to Denmark. The diagram has been divided into four different colours. Each colour symbolize one in four involved parties: The technical inspection center (TIC) and the vehicle owner, MOTOR, Captia and Post & Arkiv (see top left corner).

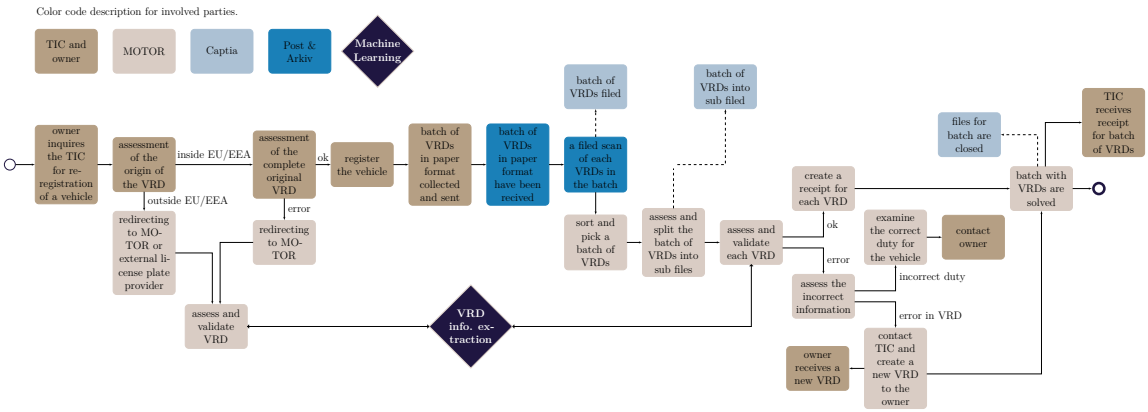


Figure 2.1: This illustration shows the entire end-to-end process for a VRD. Different colours have been applied in order to identify the involved parties and their actions within this process. The rhombus illustrates the connections to the scope of this thesis.

The purpose of the end-to-end process is to validate the registration done by the TIC. The scope of the thesis is centered around the manual validation processes done by the MOTOR division.

The validation process The first interaction is dependent on the origin of the VRD. The TIC makes an assessment of the origin of the VRD and if the origin is outside EU and EEA countries, the owner will be directed to the MOTOR office for manual processing. If the origin of the VRD is inside EU and EEA countries, the TIC will make a registration and collect the VRD. The VRDs from each TIC will be mailed once a month to the Post & Arkiv division. A case officer from the Post & Arkiv division will manually create a batch file with separate scans of sub-files for each of the VRDs and eventually upload them to Captia. The filed VRDs are now ready for manual processing and validation by employees of the MOTOR division.

The final step of the process is to create a receipt including the references of the VRDs. The receipt is created manually, based on known business roles, and needs to be returned to the TIC when the VRDs have been processed and validated. The creation of the receipts are outside the scope of this thesis but due to the known business roles, this flow can easily be done by a robotic process automation (RPA) task.

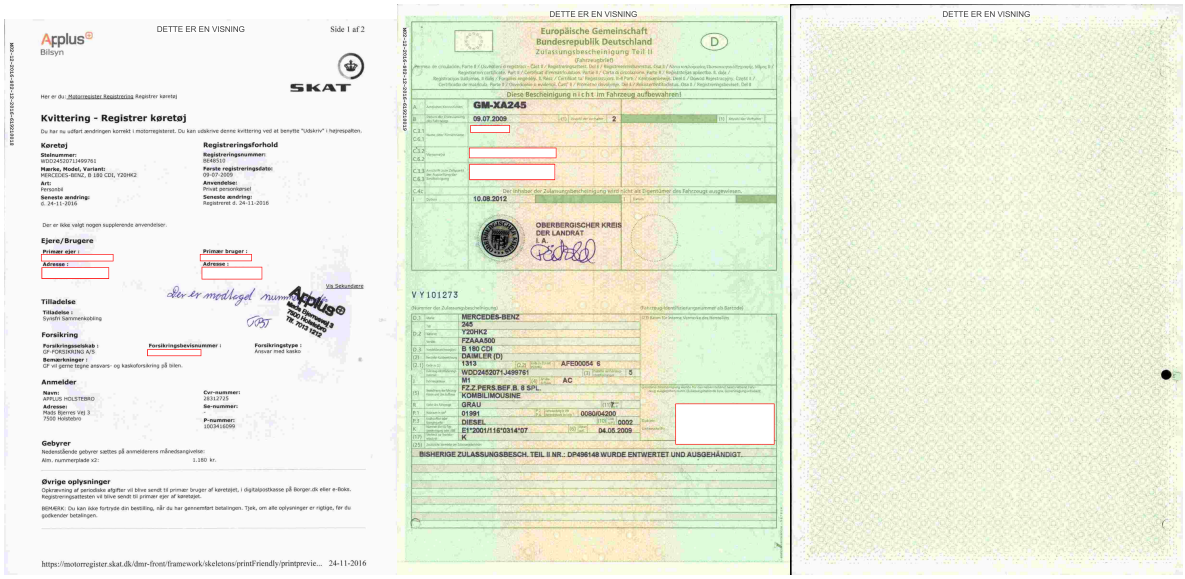
2.2 Data Foundation

A VRD file is a multi-paged PDF file including two forms of documents; Data from the TIC and data w.r.t. the vehicle. It has been chosen to divide the VRDs into single paged images in order to perform a classification of the origin, a text extraction of the LP and of the VIN sequences denoted as 1), 2) and 3) in figure 1.1 respectively. The reason for dividing the VRDs into single paged images enables the methods and properties of image classification.

Figure 2.2 provides an example of a German VRD. This will be used as an on-going example throughout this report. There are a similar examples of a Swedish VRD in appendix A. The illustrations of the German VRD and the Swedish VRD should provide an understanding and an interpretation of how the different algorithms and methods are working on the VRDs.

The sub-captions in figure 2.2 report the page index and the re-assigned class. The class is provided by the two-letter country code[1].

[1]ISO 3166-1 alpha-2 format.



(a) Page 1. Class META. (b) Page 2. Class DEs. (c) Page 3. Class DE.

(d) Page 4. Class DE. (e) Page 5. Class DE. (f) Page 6. Class META.

Figure 2.2: These six images constitutes the pages of a raw German VRD. There are remarkable differences in format of the pages. (a) and (f) illustrate the pages which do not include any original content of the registration document. These pages are referred to as the META class.

2.3 Exploratory Data Analysis

The exploratory data analysis (EDA) is an approach to gain knowledge about the data foundation. The first step of the EDA is to present various summary statistics of the origins of the accessible VRDs.

There are a total of 4,490 accessible VRDs which correspond to 23,715 single paged images in this project. These accessible VRDs constitute with 4,490 of the 24,075 (19%) possible VRDs in Captia. The process of data collection ended the 1st of October and was challenged by prolonged struggles of getting the VRDs out from Captia. The accessible VRDs only include samples from EU and EEA countries. Hence further description of VRDs from countries outside the EU and EEA region will not be included in this project.

2.3.1 The Origins of the VRDs

Figure 2.3 illustrates the distribution of the origin of the VRDs. The histogram reports a clearly imbalanced distribution of the origins of the VRDs. The imbalanced distribution needs to be considered in the modelling phase.

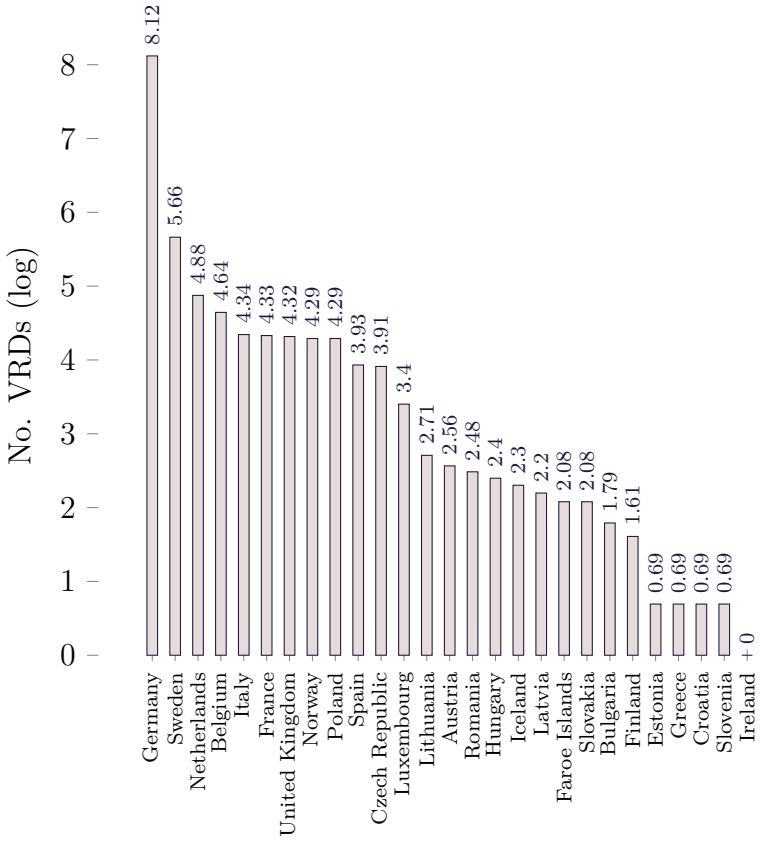


Figure 2.3: The histogram illustrates the distribution of the origins of the VRDs. All origins are inside the EU and EEA countries and the majority of the VRDs originates from Germany. Notice the second axis is presented in log-scale.

Table 2.1 provides an overview of the most frequent VRD origins. The table is grouped into the top 5 countries and has an aggregated row with the remaining countries.

Figure 2.4 illustrates a distribution of the number of pages in all of the accessible VRDs. Table 2.2 provides the summary statistics of this distribution.

There are few noticeable trends in the histogram and in the summary statistics in figure 2.4 and table 2.2 respectively. 9% of the VRDs exists of only a single page and 3% VRDs consisting of nine pages or more do not contribute the normal trend. Due to these abnormal trends the samples have been manually inspected and the following knowledge has been obtained:

- Of the VRDs consisting of only a single page $\approx$ 60% belongs to the META class.

Pos.	Country Name	Class	No. VRDs	Pct. (%)
1	Germany	DE (and DEs)	3,356	75
2	Sweden	SE	288	6
3	Netherlands	OUT	131	3
4	Belgium	OUT	104	2
5	Italy	OUT	77	2
-	Other EU and EEA countries	OUT	534	12

Table 2.1: The table shows the contributions from the main countries. The last row shows the aggregated contribution of the countries inside EU and EEA but outside the top 5. The class column denotes the categories used in the document classification.

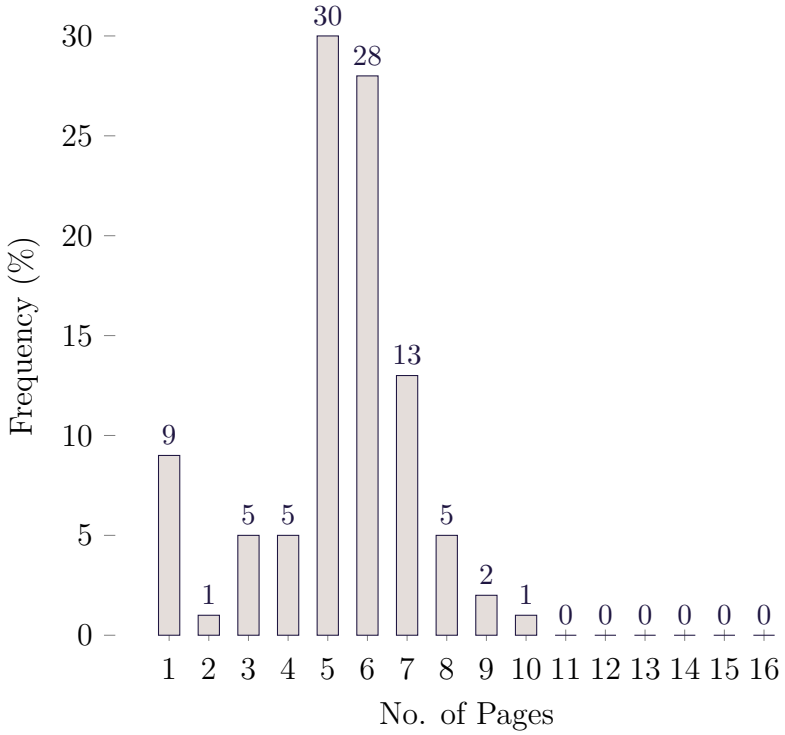


Figure 2.4: This table shows the distribution for the numbers of pages within the VRDs.

Statistics	Min.	25%	Median	Mean	75%	Max.
No. of Pages	1	5	5	5.28	6	16

Table 2.2: A summary statistic for the number of pages within the VRDs.

The META class is defined as a page in the VRD, which do not include any original content of a registration document.

- Of the VRDs consisting of nine pages or more, several VRDs include multiple registration documents. This is typically caused by an annulled registration document together with the present and valid registration document. Other abnormal VRDs include an unusual high number of META pages.

It has been chosen to include the above mentioned abnormal VRDs as they still contribute to the actual data foundation.

2.3.1.1 Re-assignment of the Labelling of the Single Page Classes

It was initial requested by the UFST to classify only the German VRDs. The German VRDs constitutes 69% of the VRDs in Captia. The page illustrated in figure 2.2b is the most important German page as it includes the LP and VIN sequences. This page is referred to as the DEs class and the sequences within the page are necessary to find in order to extract the features of interest.

A second round of manual inspection has been carried out for all of the 23,715 single paged images in this project. The purpose were to identify the pages of interest (the DEs class) and to improve the quality of the labels. The focus areas for the manual inspection were to label all pages:

- All images which are identical to the image in figure 2.2b are labelled as the DEs class.
- All images which do not include any original content of a vehicle registration document are labelled as the META class similar to the examples in figure 2.2a and in figure 2.2f.

The rest of the images keep their labelling from the processed and validated classes from Captia.

The manual re-labelling made it clear that the META pages were not unique for each of the VRD files. The META pages were duplicated in multiple VRDs from the same TIC, which is critical for making independent data partitions for the model training. It has therefore been chosen to exclude all pages labelled with the META class and to re-assign the class of VRDs outside the top 2 in table 2.1 of the most frequent origins. The reason for this choice is due to relatively few samples from these origins. Table 2.3 shows the distribution of the origin of the VRDs, when these are divided into single paged images. The class of each image is reflecting the (re-assigned) origin of the VRD.

Class	N	Pct. (%)
DE	9,755	41
DEs	3,040	13
OUT	2,307	10
SE	986	4

Table 2.3: This table shows the class distribution of the single page images after the manual inspection and re-assignment of the class labels. There are 4 classes with a total of 16,088 single paged images. N is reporting the number of samples within each class.

2.3.2 License Plates and Vehicle Identification Number

The table 2.4, table 2.5 and table 2.6 provide statistics of the reported character sequences of the license plates (LP) and of the vehicle identification numbers (VIN) for the VRDs from Germany.

It is interesting to analyse the pattern of the LP and VIN sequences because it provides an insight of the quality of the validated sequences from Captia. It also provides an understanding of the structure of the sequences.

According to ISO 3779 the length of the VIN character sequence must constitute of 17 characters and must not include the letters I, O and Q in order to avoid misreading the characters as 1 (one) or 0 (zero) respectively, applying to vehicles registered after 1981. These rules have to be implemented in the business logic module (figure 4.1).

Table 2.4 provides statistics of the length of the LP and the VIN sequences. The VIN sequences with a length different than 17 do not follow the ISO 3779 standard. 188 sequences have been manually inspected as they deviate from this standard. The manual inspection pinpoints two observations:

- The VIN sequences having a length less than 17 characters contain errors in the raw VRD and are therefore excluded.
- The VIN sequences having a length above 17 characters contain a single non-latin letter. This character will be removed during the pre-processing step.

The pattern of a German LP is characterized by one to three letters of prefix code mapping the city or region. These are followed by one to two random characters and one to four random digits in order to provide a unique LP for the vehicle in the region. The combination of these characters defines a valid range of three to nine characters for a German LP.

Table 2.4 reports three LP samples with a length outside the above mentioned valid range for a LP sequence. The samples have been manually inspected. All three samples contain errors in the raw VRD and these are therefore excluded from table 2.5 and table 2.6 and from the data foundation of this thesis.

Length	LP Pattern	VIN Pattern
1	1	0
2	0	0
3	1	5
4	15	2
5	98	4
6	578	3
7	1,179	13
8	1,140	6
9	342	24
10	0	8
11	0	9
12	0	10
13	0	6
14	0	6
15	0	0
16	0	0
17	2	3,238
18	0	22

Table 2.4: This table reports the length of the LP and the VIN character sequences. The table is only representing the VRDs from Germany. It is possible to highlight outliers deviating from the respective standard for each of the LP and VIN sequences. These samples have been manually inspected.

The strutural analysis of the sequences Table 2.5 and table 2.6 provide the result of a strutural analysis of the sequences. The sequences are summarized by three specific types of characters as mentioned below:

- X denotes a letter
- 9 denotes a digit
- S denotes a symbol

The integer within the parentheses is referring the to number of continuous recurrence of the specific type of character.

A few samples of the LP patterns (table 2.5) have been manually inspected. The validated sequences in Captia have been compared to the sequences of the raw data material. There is a tendency of a disconnected approach for processing the symbols ”-” and ” ” (space). These symbols are only frequently present within the VRDs but not in Captia or vice versa. It has therefore been chosen to remove these symbols in the business logic module in order to keep a continuous approach.

Pos.	LP Pattern	Length	N	Pct. (%)
1	X(4)9(4)	8	440	13
2	X(3)9(4)	7	424	13
3	X(4)9(3)	7	421	13
4	X(2)S(1)X(2)9(3)	8	249	8
5	X(2)S(1)X(2)9(4)	9	188	6
6	X(3)9(3)	6	185	6
7	X(4)9(2)	6	175	5
8	X(5)9(3)	8	170	5
9	X(3)S(1)X(2)9(3)	9	112	3
10	X(2)S(1)X(2)9(2)	7	106	3

Table 2.5: A structural analysis of the LP sequences for VRDs from Germany. There are three specific types of characters: X denotes a letter, 9 denotes a digit and S denotes a symbol. The integer within the parentheses is reporting the number of continuous recurrence of the type of character. This table reports the top 10 strutural patterns.

Pos.	VIN Pattern	Length	N	Pct. (%)
1	X(3)9(7)X(1)9(6)	17	763	23
2	X(6)9(1)X(1)9(1)X(2)9(6)	17	441	14
3	X(6)9(1)X(4)9(6)	17	276	8
4	X(5)9(5)X(2)9(5)	17	205	6
5	X(3)9(1)X(1)9(5)X(1)9(6)	17	198	6
6	X(5)9(5)X(1)9(6)	17	150	5
7	X(2)9(1)X(9)9(5)	17	115	4
8	X(6)9(1)X(2)9(1)X(1)9(6)	17	74	2
9	X(5)9(6)X(1)9(5)	17	55	2
10	X(2)9(1)X(2)9(4)X(1)9(7)	17	52	2

Table 2.6: A structural analysis of the VIN sequences for VRDs from Germany. There are three specific types of characters: X denotes a letter, 9 denotes a digit and S denotes a symbol. The integer within the parentheses is reporting the number of continuous recurrence of the type of character. This table reports the top 10 strutural patterns.

Key points The main conclusions to draw is that there are strutural patterns within the character sequences which needs to be modelled.

The EDA provides an insight of the single paged class distributions of the VRDs. The distribution is highly imbalanced and this needs to be considered during the development of the document classification model.

CHAPTER 3

Related Literature

This chapter introduces a selection of related or published papers which have been used as inspiration in order to design the modules for the technical pipeline presented in chapter 4. This chapter has been divided into three sections, each of which with a special focus. During the initial phase of this project, research have been studied across several domains including; document classification, text classification, frameworks able to interpret complex models and end-to-end pipelines for machine learning tasks.

3.1 Machine Learning Models for Production

There are a lot of ideas for best practices in developing machine learning models to production. The book by Ng [3] introduces a technical strategy for developing machine learning models to solve real life problems.

These methods have lead to an insight of a practical development process. The development phase of this project has been structured according to the suggested practical guidelines of [3]. The book covers all facets of the development cycle and pin points the essential methods from data collection and data pre-processing to enable fast development iterations. This is done by using a single metric for model evaluation and using error analysis to learn insight of the data foundation and the complex models.

These guidelines form a fundamental knowledge of developing a successful machine learning project, which is why the literature is used for this project.

3.2 First and Second Stage Models

The task of the first stage model is to classify the origin of an input document.

A local normalized descriptor such as the Histogram of Oriented Gradient (HOG) is presented by Dalal and Triggs [4]. It represents a high performing descriptor for visual object recognition due to its properties. Since the structural document layout of the classes have low within class variance it is possible to apply the properties of the HOG descriptor, which is composed by simple mathematics.

The HOG descriptor (in combination with an appropriate classifier) has been chosen as baseline reference in this project, due to its simple mathematics.

The article by Kang, Kumar, Ye, *et al.* [5] expounds an architecture of convolutional neural network (CNN) to learn the structures of the document layout instead of relying on handcrafted features with a classification algorithm. Their results of the CNN model clearly outperforms handcrafted feature models. The CNN architecture of this project is motivated by the performance documented in [5].

The task of the second stage model is to perform text recognition of an input image. The following two approaches have achieved state of the art results in their developed decade respectively.

The initial choice for solving the text recognition task was to apply the well acknowledged Tesseract optical character recognition (OCR) engine originally developed by Smith [6] and presented at The Annual Test of OCR Accuracy in 1995. It is one of the most recognized models within the field of text recognition and in this project the framework has been applied as baseline reference.

A more modern approach for performing text recognition is presented in the article by Shi, Bai, and Yao [7], which introduce a novel convolutional recurrent neural network architecture (CRNN). The performances achieves state of the art results for text recognition. Their demonstrated text recognition case is similar to the task in hand of the second stage in this project. The CRNN architecture of this project are inspired by [7] and is studied in further details in section 6.2.2.

3.3 Model Interpretations

The ability to interpret complex models is the key to trust their predictions and to be GDPR compliant. This section introduces two frameworks that are able to provide an explanation for a produced prediction.

The versatile LIME framework suggested by Ribeiro, Singh, and Guestrin [8] is able to produce an explanation for any prediction. Despite its drawbacks of setting the required hyperparameter to obtain an explanation, the framework can be used for the suggested models in the first stage module as it is possible through trial and error to find the optimal hyperparameters.

The secondary Grad-CAM framework is specialized to convolutional neural network models and it provides a discriminative localization of the most important spatial perceptrons. This localization can be interpreted as a visual explanation for a given class. The framework is suggested by Selvaraju, Das, Vedantam, *et al.* [9].

The two above mentioned frameworks have different properties, however they both demonstrate a great visual explanation in their respective papers, which is why the

methods have been chosen to transfer to the domain of this project as an explanation of the model interpretation. Both of these are further described in section 5.7.2 and section 5.7.3 respectively.

CHAPTER 4

The Technical Pipeline

This chapter will introduce the proposed technical pipeline of this project. The pipeline exists of four modules and provides an output composed of three different predictions with a VRD given as input. Figure 4.1 illustrates the technical pipeline highlighted in the brown box.

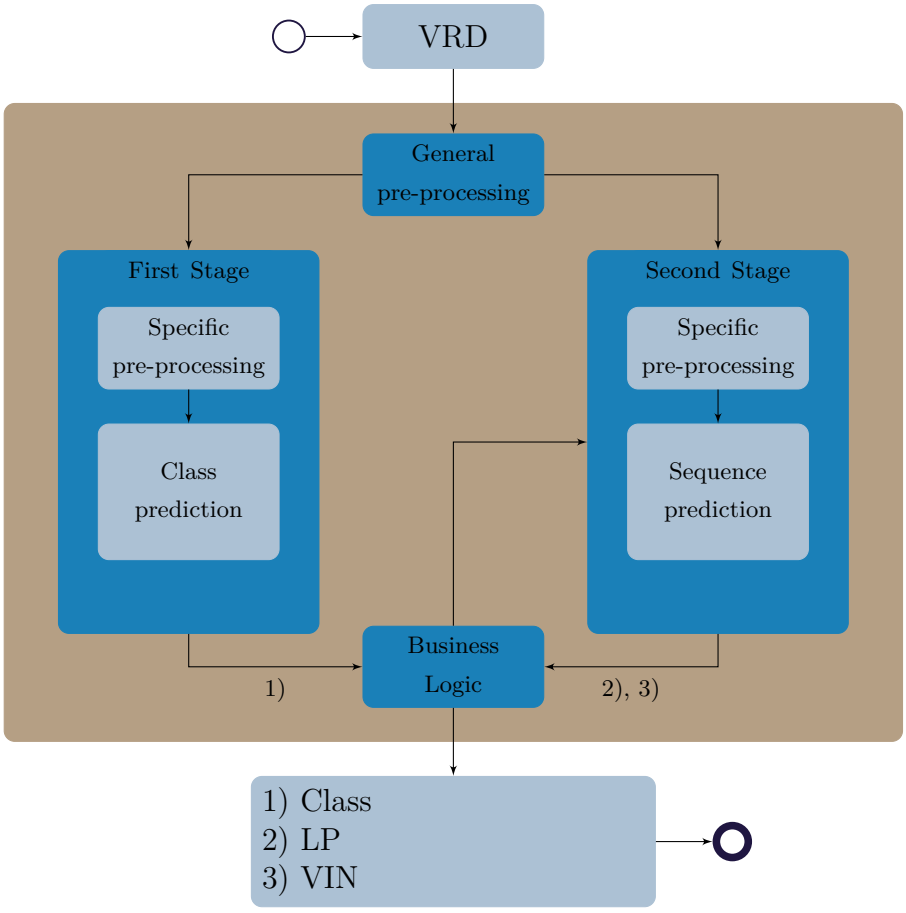


Figure 4.1: The figure shows an overview of the proposed technical pipeline. The pipeline has been divided into two stages; a general pre-processing module and a business logic module which is able to initialize the prediction of 2) and 3) based upon the classification in 1). The name of 1) has been changed from ”Two-letter country code” (figure 1.1) to ”Class” in order to reflect the labelling presented in table 2.3.

The pipeline contains four modules; The first stage and the second stage modules will be described further in section 4.2 and in section 4.3 respectively. Whereas the general pre-processing module and the business logic module will be described further in section 4.1 and in section 4.4 respectively.

4.1 General Pre-processing Module

The first step of the technical pipeline is to perform a general pre-processing of the VRD from a PDF format into an `imagemagick` object. The general pre-processing module includes the following methods and is carried out in sequential order for each of the pages within the VRD:

1. Cropping the meta text located in top and in the left hand-side of the raw VRD, see images in figure 2.2.

2. Determine the rotation of each of the pages in the VRD in order. The determined rotations are used to deskew each of the pages and hereby obtain an important horizontal alignment for the specific pre-processing steps in the first stage and in the second stage.
   The rotation is computed by a line detection method based upon the Hough Transformation, see section 5.1.

3. Find the smallest possible rectangular cropping area which includes the deskewed VRD content. This step includes the following steps:

   a) Apply a median filter with a radius of 5. The filter removes the noise within neighbouring pixel values. The median filter is more robust to the noise within the neighbouring pixels compared to the mean filter [10, sec. 5.1].

   b) Convert the page from colour to gray-scale and estimate the Otsu's threshold. The method of the Otsu's threshold is explained and illustrated in section 5.2.

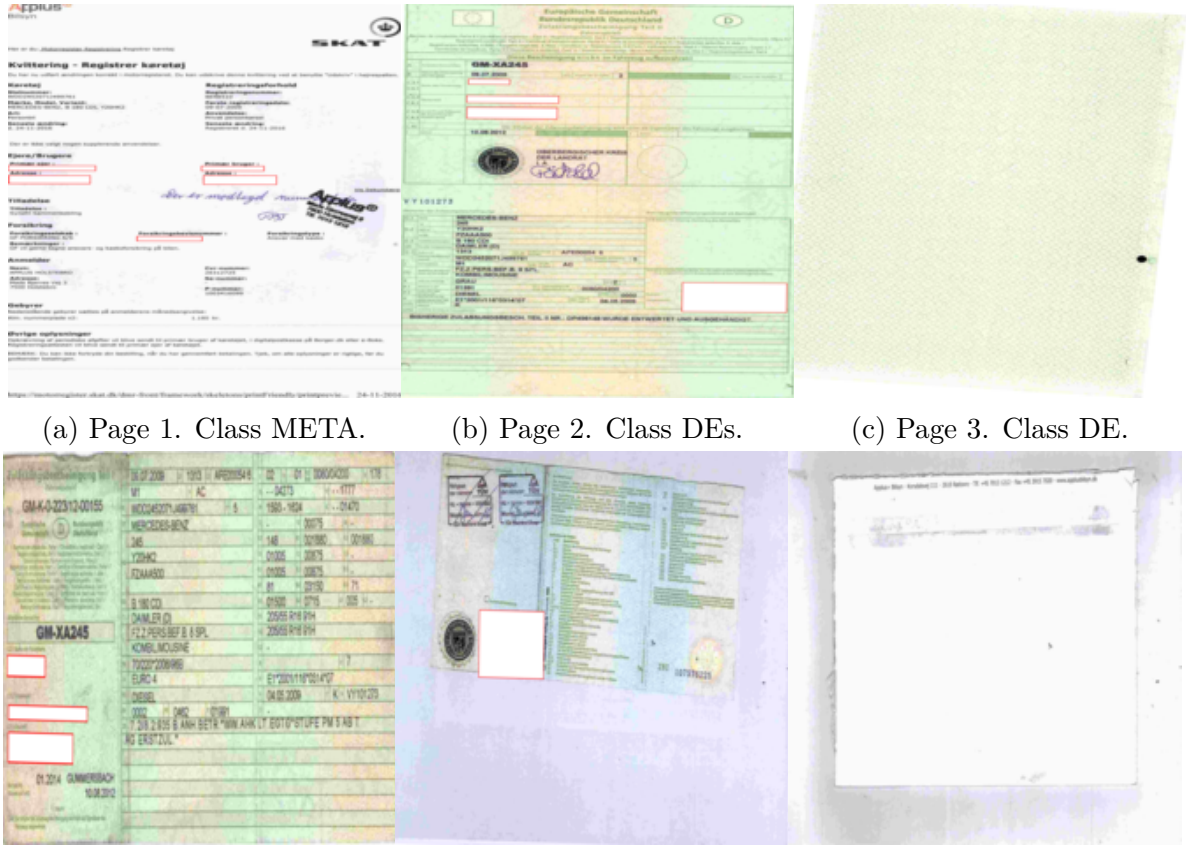
   c) From the Otsu's threshold, it is possible to create an binary image with a foreground, assumed to be the VRD, and a background, assumed to be the surrounding noise from the scans. The coordinates of all the pixels with a value below the threshold (the foreground) are used to find the smallest possible rectangle which determines the cropped region of the VRD.

4. The updated `imagemagick` pointer is returned to the first stage and to the second stage modules.

4.2 The First Stage Module

The first stage module exists of two steps. First step is to apply a specific pre-processing according to the input properties of the classification model. The next step is to predict the class of each of the pages in the VRD.

Figure 4.2 illustrates the pre-processed input pages on the example with the German VRD (see the raw VRD images in figure 2.2).



(a) Page 1. Class META. (b) Page 2. Class DEs. (c) Page 3. Class DE.

(d) Page 4. Class DE. (e) Page 5. Class DE. (f) Page 6. Class META.

Figure 4.2: These six images have been general pre-processed and specific pre-processed according to the input properties of the document classification model. (c), (e) and (f) illustrate examples of images, which are not perfectly pre-processed. However, (b) is the image of interest and has been pre-processed to a satisfying degree.

There will be developed two models to perform the classification of the images. Both models will be evaluated and compared with same metric, equation 6.1, in order to find the most suitable model for this classification task. The theory of the baseline model is described in section 5.3 and the theory of the components used to compose the competitive model is described in section 5.5. Their implementations are further described in section 6.1.1 and in section 6.1.2.

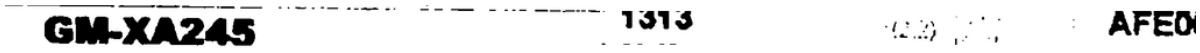
4.3 Second Stage Module

The second stage is activated by the business logic module if one or multiple pages in the VRD are classified as the DEs class (figure 4.2b).

Similar to the first stage module, the second stage module contains of two steps where the first step is to crop two areas of interest and then produce the predictions (and a decoding) of the text sequences within these cropped areas.

The cropping areas of the LP position and of the VIN position are a crucial part in the second stage. The chosen approach to crop these areas is to use fixed cropping areas w.r.t. the relative dimensions of the general pre-processed page. These two cropping areas are based upon 25 random general pre-processed samples from the training distribution of the pages corresponding to the DEs class. The dimensions of the pages and the coordinates of the areas of interest have been noted and scaled according to the dimensions. Lastly the average is found. This results in four numbers for each of the cropping areas, which are relative to the dimensions of the page. These four numbers correspond to the height and width of the area and the upper left starting coordinate within this area.

The four numbers reflect the entire text field within the VRD and it has been chosen to perform a further crop in order to zoom in on the text of interest. The new cropping is conceptually similar to the approach described in the specific pre-processing in the first stage module. The only difference between these cropping approaches is the extra added padding of 4 pixels in the second stage specific pre-processing. The cropped areas of the example in figure 2.2b are illustrated in figure 4.3.



(a) A tolerable crop of the LP field. (b) A miscrop of the VIN field.

Figure 4.3: These two images have been general pre-processed and specific pre-processed according to the input properties of the prediction model in the second stage module. Notice that there are two separate images with the same width and neither of the crops are perfect. (a) contains a horizontal line which indicates an offset in the crop region but the sequence of interest (GM-XA245) is suitable. (b) entails a vertical offset.

The cropped area of the LP in figure 4.3a seems to be acceptable for the model prediction. The cropped area for the VIN in figure 4.3b, is slightly off and does not contain the VIN text of interest. The VIN field in a German VRD is roughly twice as narrow as the

LP field and is hereby much more sensitive to misalignments. The misalignment of the initial crop in the general pre-preprocessing module is an inappropriate drawback of the fixed cropping approach.

In order to find the most suitable approach for the LP prediction and for the VIN prediction, there will be applied two models both of which will be evaluated and compared on same metric (equation 6.2). The implementation of the baseline model is described in section 6.2.1 and the implementation of the competitive model is described in section 6.2.2. The theory of the components used to compose the competitive model is described in section 5.5.

4.4 Business Logic Module

The business logic module is illustrated in figure 4.1 and contains two major properties. The actions of this module depend upon the input from the first stage module. The actions are explained in the bullets below:

- If the output (1)) from the first stage module does not contain an origin classification of the DEs class, the module will return the page-wise predicted origin and its belonging probability. It returns `NA` for the LP field and VIN field and likewise `NA` for their belonging probabilities.
- If the output (1)) from the first stage module does contain a single or multiple origin classified as the DEs class, the module returns the page indices to the second stage module and waits for the predicted LP (2)) text and VIN (3)) text and their belonging probabilities.
  The outputs from the second stage module are post-processed in order to return valid sequneces, which only contain uppercase characters from the chosen vocabulary and a compliant VIN string due to its previous mentioned standard. E.g. if the predicted VIN character string includes characters such as ”I”, ”O” or ”Q” they will be converted to ”1” or ”0” respectively.

At the current state the business logic module does not include pre-defined business rules of threshold requirements for valid predictions. The option for thresholding are further discussed in chapter 8, table 8.1a and table 8.1b for the first stage predictions and for the second stage predictions respectively.

CHAPTER 5

Theory

This section will provide the technical and theoretical foundation needed to gain an understanding of the models within the first stage module and the second stage module of the technical pipeline illustrated in figure 4.1.

The first method in section 5.1 is introducing the Hough Tranform which is used to determine the rotation of an image. The method of the Otsu's Threshold is determining the threshold value. This is separating the intensity values of a gray-scale image between its two bimodal distributions (section 5.2). Both of these methods are required for pre-processing each of the single paged images within the VRD.

Section 5.3 introduces the foundation of the histogram of orientated gradients descriptor, which is a method for local feature engineering of an image. This section introduces the basis of a tree based classifier. The properties of the Random Forest algorithm is described in section 5.4.

The main attention is devoted to section 5.5 which describes the following: the foundation of an artificial neural network, two special cases of the artificial neural network; a convolutional neural network and a recurrent neural network and the principle of transfer learning and techniques of regularization for these networks.

Section 5.6 introduces the concepts of bias and variance which are essential for evaluating the performance of the models.

Finally, section 5.7 introduces a framework for interpretation of the developed models which is a crucial aspect of being GDPR compliant in production.

5.1 The Hough Transform

The Hough Transform is a powerful tool for feature extraction and line detection. The properties of the Hough Transform expands to fitting multiple sinusoidal curves into the Hough Space. It is possible to estimate a rotation in an image by fitting multiple sinusoidal curves w.r.t. edge pixels determined by e.g. a Canny filter.

Recall the formula for a normal line, equation 5.1.

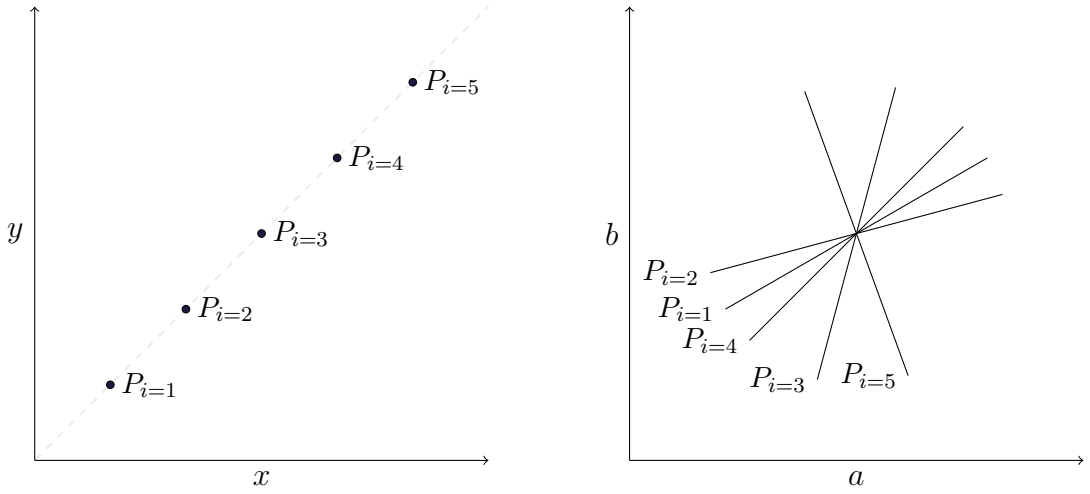
$$y_i = ax_i + b \tag{5.1}$$

where the linear connection between x_i and y_i, in edge pixel i, is derived by its slope a and its intercept b.

It is possible to rewrite equation 5.1 into b and a space, equation 5.2.

$$b = -x_i a + y_i \tag{5.2}$$

Figure 5.1 illustrates the concept of the Hough Transform. There are five edge pixels illustrated in figure 5.1a and their corresponding mapping into the b and a Hough Space in figure 5.1b.



(a) Five edge pixels.

(b) Their mapping into Hough Space.

Figure 5.1: These two figures illustrates the concept of the Hough Transform. The illustrations are inspired by [11].

The lines in figure 5.1b will create an intercept somewhere in the Hough Space (b and a). The intercept in the Hough Space is a direct mapping of the slope (a) and the intercept (b). In other words, each of the lines will vote for multiple values of a and b. The value of a and the value b with the highest aggregated vote describes the slope and the intercept of the line, which lies on the edge pixels in figure 5.1a.

Edge pixels There exists multiple methods for creating edge detection. The method which is suitable for the Hough Transform is the Canny kernel method [12]. The end result of the Canny method is a binary image with the intensity values $\in \{0, 255\}$. The high intensity pixels denotes an edge pixel.

The method is composed of four steps[1]:

[1]These four steps have been rewritten from [12].

1. Convolute an appropriate Gaussian blur in order to reduce the prominent details of the gray-scale image. The Gaussian blur takes two parameters; r and σ which denotes the radius and standard deviation of the filter respectively.

2. Compute the gradients for each of the pixels using the derivative of the previous convoluted image. Formulas of the magnitude and direction of the gradients are provided respectively in equation 5.11 and equation 5.10.

3. Assess each of the magnitudes of the gradients and assign the pixel as an edge-pixel if its magnitude are larger than the magnitude of its two neighbours in the gradient direction.

4. Reduce the number of weak edge-pixels by the hysteresis threshold. The hysteresis is defined by two values L and S, which denotes the intensity value respectively for a weak and a strong edge.

When combining these four stages the Canny method has very little noise interference compared to the Sobel kernel (equation 5.8).

The Canny method is called from the `magic R-package` with the following parameters: $r = 5$, $\sigma = 1.4$, $L = 10\%$ and $S = 40\%$. The percentage values, L and S, are relative values compared to the highest intensity pixel value within the image.

The parameters provide the following characteristics:

- r defines the number of neighbouring pixels to consider in the Gaussian blur and σ denotes the strengths of the blurring. High values of r and σ result in very low noise interference and a large reduction in prominent details.

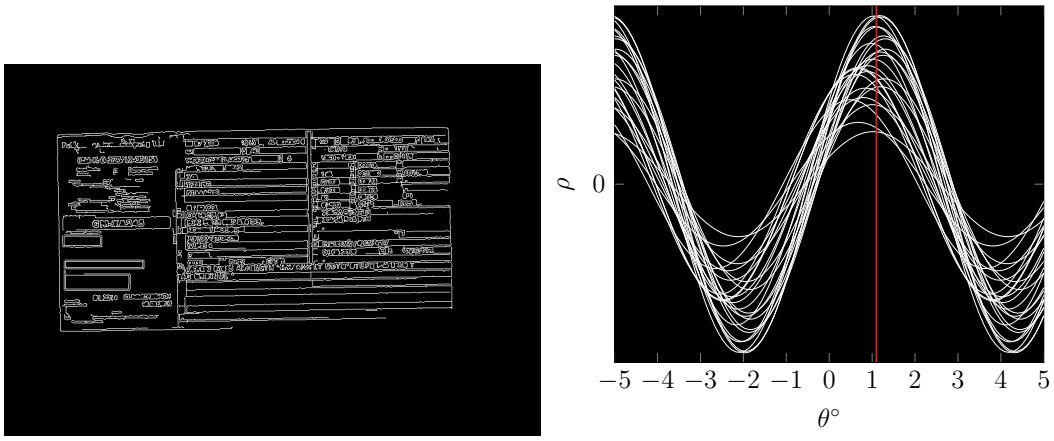
- L and S define the hysteresis threshold for a weak and a strong edge-pixel. An increase in the difference between L and S results in fewer pixel marked edges which demands a larger difference in the gradients of the neighbouring pixels. More pixels will be marked as edges if the values of L and S are increased.

Finding the right hyperparameters of r, σ, L and S have been done using trial and error for a small diverse set of VRD samples.

Polar form The normal equation for a line can also be written on polar form. This solves the issue of values approaching infinity. Equation 5.3 denotes the polar form of the normal equation.

$$\rho = x_i \cos(\theta) + y_i \sin(\theta) \tag{5.3}$$

The Hough Space, in regard to equation 5.3, is defined by θ and ρ. θ is quantified and defined for a specific interval in degrees. Figure 5.2 provides a similar illustration as figure 5.1 by using the polar form of the line equation.



(a) Edge pixels computed by the Canny filter.

(b) A random sample of 25 edge pixels and their respective sinusoidal curves.

Figure 5.2: (a) illustrates the edge points created by a Canny filter. (b) illustrates 25 random samples of edge pixels and their respective curves. The red line illustrates the value of θ which archives most of the votes and represents the most likely rotation of the image in (a).

As used in the figure 5.1, a similar voting mechanism can be transferred to the sinusoidal curves by extrapolating each of the edge pixels w.r.t. θ. The highest number of curve-crossings is obtained at $\theta \approx 1°$ which is the estimated rotation of deskewing the image.

5.2 Otsu's Threshold

The Otsu's threshold is an algorithm which provides an automatic threshold value between two classes within a gray-scale image, $f(x, y)$. The algorithm assumes there is a valley $(k*)$ between the two classes and the intensity values of the classes can be described with a bimodal distribution.

The method of selecting an adequate threshold is an essential task of separating an object from its background. This is needed for cropping the background noise from a single paged VRD (example in figure 2.2d).

Equation 5.4 and equation 5.5 define the maximization problem, which find the optimal value $k*$ where the variance between the two classes are maximized [13].

$$\sigma_B^2(k*) = \max_{1 \leq k < L} \frac{(\mu(L)\,\omega(k) - \mu(k))^2}{\omega(k)\,(1 - \omega(k))} \tag{5.4}$$

$$\begin{aligned} \mu(k) &= \sum_{i=1}^{k} ip_i \\ \mu(L) &= \sum_{i=1}^{L} ip_i \\ \omega(k) &= \sum_{i=1}^{k} p_i \\ p_i &= \frac{n_i}{N}, \quad p_i \geq 0 \quad and \quad \sum_{i=1}^{L} p_i = 1 \end{aligned} \tag{5.5}$$

where ω denotes the probabilities of the class occurrence and μ denotes the mean level of the two classes. k is a vector of unique intensity values in $f(x, y)$ and L denotes the highest intensity value in $f(x, y)$. p denotes a normalized probability distribution where n_i reflects the number of pixels having an intensity value k. N is the total number of pixels in the gray-scale image. Equation 5.4 to equation 5.5 are provided by [13, eq. 1-19].

A simple solution to find $k*$ is to perform a sequential search over k where a bin-count n_i is computed and normalized by its probability distribution p_i for each of the intensity values. The initialization of ω and μ will step through all values of k and return the value $k*$, which maximizes σ_B^2 from equation 5.4.

Figure 5.3 illustrates a gray-scale histogram of figure 2.2d and its optimal Otsu's threshold value $k*$.

The gray-scale histogram in figure 5.3 does not have a clear bimodal distribution which separates the objects from the background noise. However, the optimized Otsu's threshold $k*$ for figure 2.2d does provide a clear separation of the two classes, which makes the cropping succeeded in figure 4.2d.

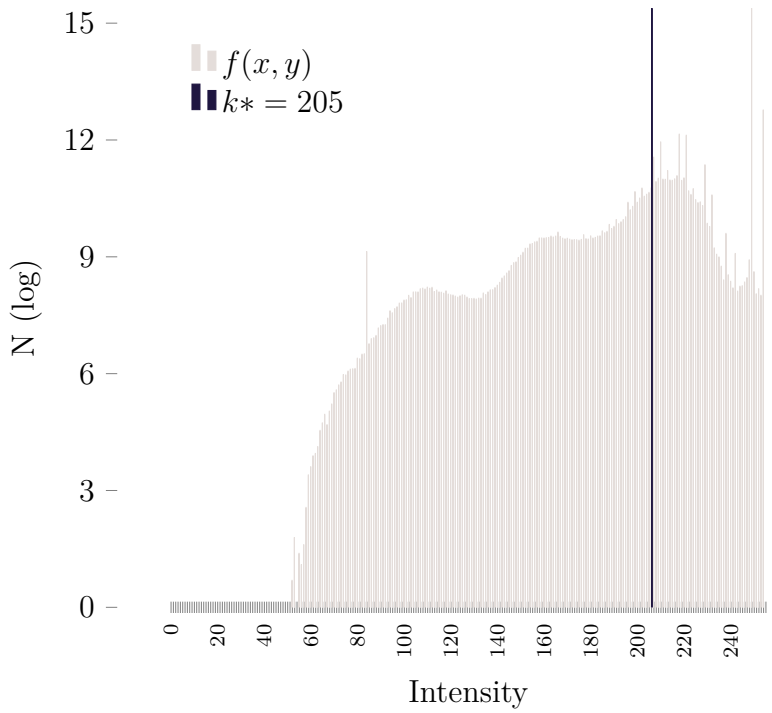


Figure 5.3: This figure illustrates the gray-scale intensities of the image ($f(x, y)$) in figure 2.2d. The single bin $k*$ reports the optimal Otsu's threshold which should separate the foreground from the background.

5.3 Histogram of Orientated Gradients

An image can be interpret by global or local features. A global feature would e.g. be to create statistics for each of the pixels within the image. This could be to average the colour intensity in each of its channels and represent these features by histogram bins. A local feature approach would be to create multiple histograms for the given patches of the image and hereby keep the spacial information within the image. The overall aim of the feature engineering is to create features which are unique, leading to an unambiguous match for multiple images within the same class.

The Histogram of Orientated Gradients (HOG) descriptor is an algorithm which creates representation in an image based upon its local accumulated gradient structures. The local patches, also called *cells*, can be either rectangular or radial. The features are calculated for each cell and then aggregated in local spatial histograms. The rectangular cell are preferred due to the properties of the coming mentioned edge detection filters [4].

The first step of the HOG-descriptor is to compute the gradients in each of the pixels, equation 5.6, where the images are denoted as $f(x, y)$. A difference in the gradients in the neighbouring pixels leads to an identification of an edge.

$$\overrightarrow{G}\left(g_x, g_y\right) = \nabla f\left(x, y\right) \tag{5.6}$$

Since $f\left(x, y\right)$ is not a continues function it is then possible to approximate the pixel gradient bot directions, x and y, by considering the directional neighbouring pixel as in equation 5.7 [10, eq. 5.10 & 5.11].

$$\begin{aligned} g_x\left(x, y\right) &\approx f\left(x+1, y\right) - f\left(x-1, y\right) \\ g_y\left(x, y\right) &\approx f\left(x, y+1\right) - f\left(x, y-1\right) \end{aligned} \tag{5.7}$$

The directional pixel gradient is similar to convolution of a 1D-mask $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ in both directions. However the approach can be very sensitive to noise in the image by only considering the latter and feature pixel. The most common approaches for edge detection is to convolute (equation 5.9) a Prewitt kernel or a Sobel kernel. The directional 3×3 Sobel kernels are represented in equation 5.8 and are less sensitive to pixel noise [10, chap. 5].

$$v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.8}$$

Equation 5.9 denoted one of many ways to apply a kernel on a image.

$$\begin{aligned} g_x\left(x, y\right) &= \sum h \odot f\left(\left(x - r_1\right) : \left(x + r_1\right), \left(y - r_2\right) : \left(y + r_2\right)\right) \\ g_y\left(x, y\right) &= \sum v \odot f\left(\left(x - r_1\right) : \left(x + r_1\right), \left(y - r_2\right) : \left(y + r_2\right)\right) \end{aligned} \tag{5.9}$$

where $\odot$ is element-wise multiplication, r_1 and r_2 is the radius of the kernel, $r_1 = r_2 = 1$ for the Sobel kernel in equation 5.8. $f\left(\cdot\right)$ defines a sub-region of images where the convolution is applied, see an illustration of convolutions in figure 5.13. The convolution uses at stride of 1 and a specified padding scheme. The choice of padding is effecting the kernel when it is operating on the edges of $f\left(\cdot\right)$.

The orientation of the pixel-wise tangent-plane can be computed by inverse tangent function of the directional pixel-wise gradients, estimated in equation 5.7. Equation 5.10

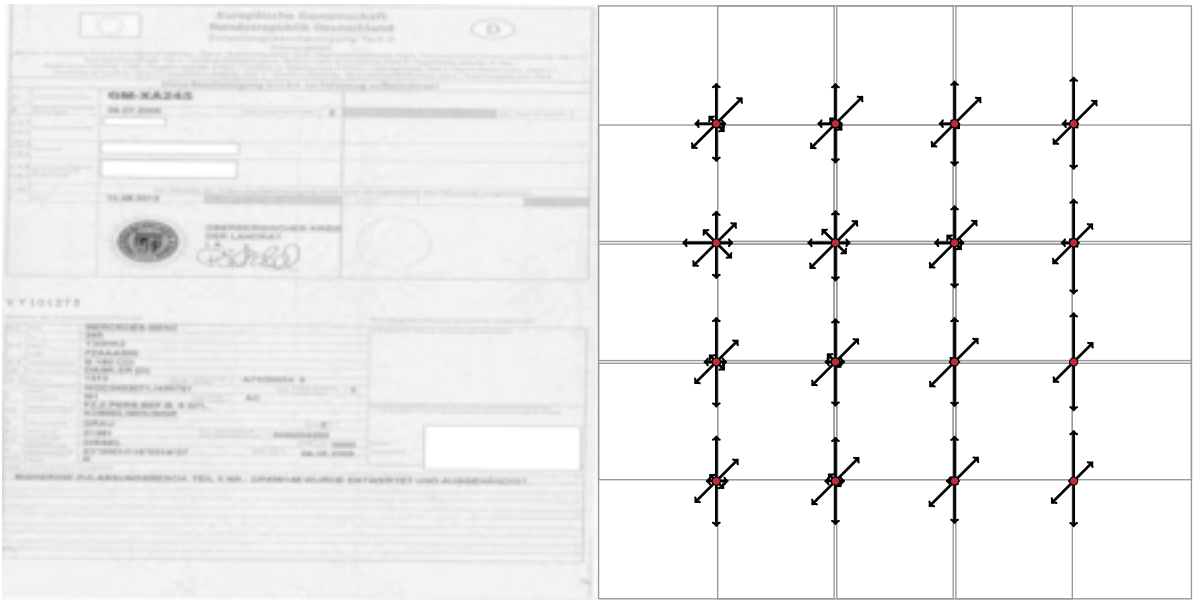
$$angle\left(x, y\right) = \arctan\left(g_y\left(x, y\right), g_x\left(x, y\right)\right) \tag{5.10}$$

The magnitude of the directional pixel-wise gradients is representing the presence or the absence of an edge, equation 5.11.

$$magnitude\left(x, y\right) = \sqrt{g_x\left(x, y\right)^2 + g_x\left(x, y\right)^2} \tag{5.11}$$

The final step of the HOG-descriptor is to accumulate the pixel-wise magnitudes to the appropriate bin interval for each local spatial region determined by the dimensions of the images and the number of cells. The number of bins and the number of cells are a design choice which will be defined in section 6.1.1.

Figure 5.4 illustrates the computed HOG-descriptor features for a pre-processed German VRD image. An equivalent illustration for a Swedish VRD image is provided in figure A.3. The red dots are representing the center of each of the cells. Each cell is sharing half of its pixels with one or multiple other cells. The overlapping cells are recommended by [4] in order to achieve a good classification performance.



(a) A pre-processed image of figure 2.2b. (b) Computed HOG descriptor of figure 5.4a.

Figure 5.4: This figure shows an illustration of the main trends of direction and magnitude for each of the local spatial regions. The image is divided into a 4×4 overlapping cell grid. The center of each cell is represented by a red dot. There are 8 accumulated orientation bins in each of the local cells. The magnitudes of these bins are defined as the HOG descriptors.

The main trend of the features in figure 5.4 are the vertical magnitudes. The magnitude is mapping the accumulated gradients, by recalling the properties of the HOG-descriptor. This identifies the most conspicuous edges in the pre-processed image which are the vertical edges.

The pattern of the engineered HOG features for the German VRD image in figure 5.4b and the pattern of the HOG features for the Swedish VRD image in figure A.3b have visual differentiable cells, however there seems to be a correlation between these features. An aggregated summary of the correlation matrix, table 5.1, includes the upper triangle elements, excluding the diagonal elements, of the correlation matrix.

Table 5.1 reveals the coefficients with a strong pairwise collinearity. This property needs

$\lvert\rho\rvert$	N	Pct. (%)
0	709	9
0.1	1,227	15
0.2	752	9
0.3	718	9
0.4	1,061	13
0.5	1,295	16
0.6	1,020	13
0.7	840	10
0.8	314	4
0.9	153	2
1	39	0

Table 5.1: A summary statistics of the correlation values between the HOG descriptors, $|\rho|$. The table only includes elements of the upper triangle, excluding the diagonal elements of the 128×128 correlation matrix. The absolute elements ($|\rho|$) are rounded to one decimal and then aggregated to report the percentage distribution.

to be eliminated. The strong pairwise collinearity, $|\rho|$ close to 1, means that the pairwise features can be created by a linear combination of each other. A unique estimation of the model coefficients does not exist for these features[2].

There are multiple approaches for solving this collinearity issue. The main idea is to create a projection of the features into a lower dimension, called a latent variable space, and then optimize the classifier for this projection. The projection mechanism is included in some classifiers whereas other classifiers need a preprocessed input, which can be performed e.g. by the principal component analysis (PCA).

5.4 A Tree Based Classification Algorithm

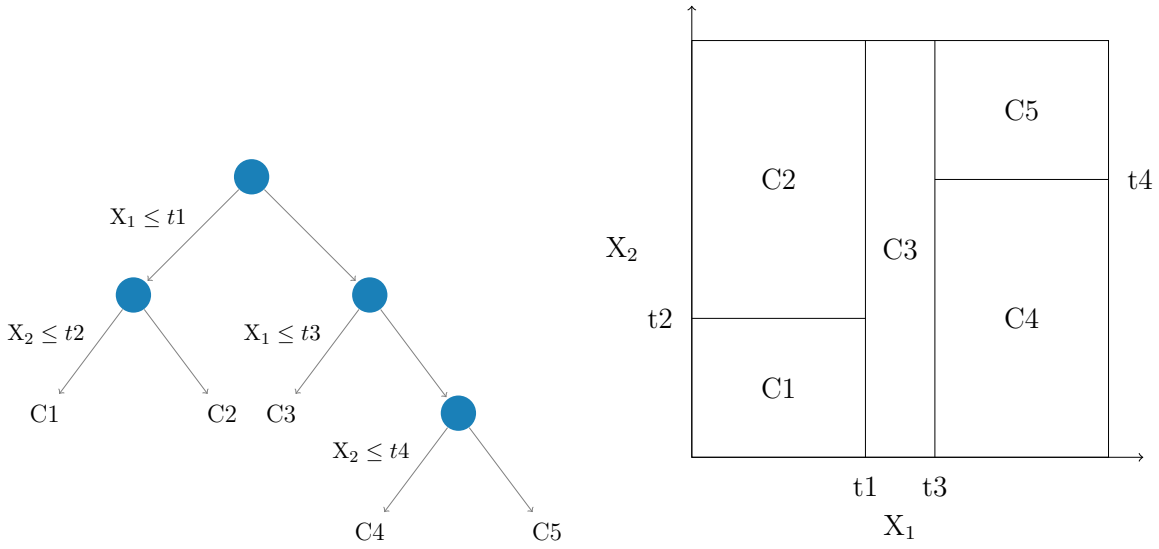
The next section deals with a classifier, which is capable of handling correlation features and producing a prediction of the origin of the VRD.

The classification and regression tree (CART) is the foundation for a decision tree approach. A rule based hierarchy of splits will be created during training and these rules are acting as the a high dimensional decision boundaries. The decision tree is interpretable and can provide an origin classification from a high dimensional feature space computed by the HOG-descriptor algorithm.

A decision tree is conceptually simple and yet very powerful. The main idea behind a decision tree is to divide the feature space into boxes of linear decision lines. Figure 5.5

[2] "When some predictors are linear combinations of others, then X^TX is singular, and there is (exact) collinearity. In this case there is no unique estimate of β. When X^TX is close to singular, there is collinearity (some texts call it multicollinearity)." [14, p. 64].

illustrates the properties of a decision tree created on synthetic data.



(a) A single decision tree. (b) A 2D-grid with decision boundaries.

Figure 5.5: These figure illustrate a decision tree and its learned decision boundaries. The illustrations are inspired by [15, fig. 9.2]

The synthetic data includes two features X_1 and X_2, and five classes $C_1, ..., C_5$. The number of features are kept low for illustration purposes. The synthetic data requires four splits $t1, ..., t4$ in order to separate the data.

A node m (the blue circle in figure 5.5a) includes N_m observations in the region R_m where the probability of each class k in the node m is mathematically determined by equation 5.12. When there is only one observation left in a node, the name will change from node to a *pure* node, also called a leaf.

$$\hat{p}\left(k, m\right) = \frac{1}{N_m} \sum_{x_i \in R_m} \begin{cases} 1 & y_i = k \\ 0 & y_i \neq k \end{cases} \tag{5.12}$$

The majority class for an arbitrary node m is given by an evaluating of the proportion of each class k (equation 5.12). The majority class is the one maximizing equation 5.13.

$$k\left(m\right) = \arg\max_k \left(\hat{p}\left(k, m\right)\right) \tag{5.13}$$

The most important part of the decision tree is the splitting procedure. The following three methods can be used to measure the node impurity. A low impurity means that each distribution of the classes in a node is highly skewed towards a single class k. A high impurity is caused by an equal distribution in each of the classes in a node.

- Gini index: $\sum_{k=1}^{K} \hat{p}\,(k,m)\,(1-\hat{p}\,(k,m))$
- Cross entropy: $-\sum_{k=1}^{K} \hat{p}\,(k,m)\log\left(\hat{p}\,(k,m)\right)$
- Misclassification: $1-\hat{p}\,(k,m)$

The misclassification measure is used as a value for tuning the hyperparameters and not as a value for performing splits in order to grow the tree. The preferable split is the one providing a pure node when multiple nodes obtain the same impurity measures [15, chap. 9.2].

5.4.1 The Random Forest Classifier

The Random Forest classifier is a supervised ensemble method for classification (and regression) [16]. The algorithm is a refinement of the bagged decision tree algorithm which originates from a single decision tree algorithm, as described above.

The following two main properties of the algorithm are the foundation for its predictive power:

- Bagged trees: The concept of bagging is to average the prediction over a large collection of trees based on bootstrap samples[3]. This bootstrap aggregation will lower the bias of the averaged prediction, [15, chap. 8.7].
- Decorrelation of trees: The Random Forest tries to reduce the variance by decorrelating the trees in the forest. The decorrelation is achieved by sampling a random subset of feature candidates for a current split. The impurity measure will only be computed for these candidates.

The following heuristics for classification are denoted from [15, chap. 15.3]:

- The default value of m is $\left\lfloor\sqrt{p}\right\rfloor$ where p is the number of candidate variables in each split.
- A minimum node size of one.

Figure 5.6 illustrates an artificial example of a single tree in the Random Forest. There are three classes ({ DE, META, SE }, $K = 3$), four observations, $p = 128$ features, and three nodes $m = 3$ in this example. The randomly sampled features ({ X_1,X_2,...,X_{128} }) are reported on the r.h.s. of the nodes.

[3]Bootstrap samples constitute from a re-sampling approach where a number of samples are drawn with replacement. The out of bag (OOB) samples are samples not present in the bootstrapping pool but in the sample population.

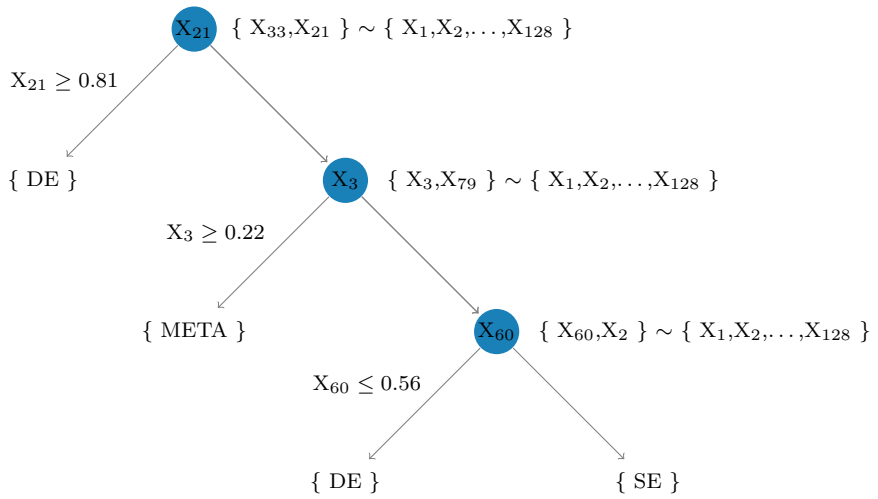


Figure 5.6: This figure shows a conceptual illustration of a single tree in the Random Forest. The candidate variables are able to compute the splitting criterion which is unlike circumstances for the splitting criterion of the single decision tree.

The two main differences between the single tree in the Random Forest in figure 5.6 and the decision tree in figure 5.5a is the proportion of possible feature candidates to be used in the split-criterion and in the samples which are used to grow the tree.

5.5 Artificial Neural Network

This section introduces the essential steps for composing, learning and making the mapping between the input and output for an ANN model, which will be able to generalize to unknown data.

An artificial neural network (ANN) is an old modelling technique inspired by the flow of processing biological neural signals. The first research of ANN models was introduced by McCulloch and Pitts [17] in the 1940s and during the last two decades, the research within the deep ANN architectures have exploded as the computational power has increased.

Suppose we want to create a model that can estimate the probabilities of a two class behaviour. The behaviours are described by three input features (x_i). The output probabilities are denoted by two respective output values y_i. Figure 5.7 provides an illustration of a simple feedforward ANN model. The blue circles are called perceptrons and are illustrated in detail in figure 5.8. Each layer in the model includes a bias perceptron overlaid with a $b_j^{(i)}$. The bias perceptron does only contribute with an adjustment to the perceptrons in the following layer.

A layer in an ANN is defined by a group of perceptrons. An ANN is composed of multiple layers which are typically arranged in a hierarchy structure. The activations to

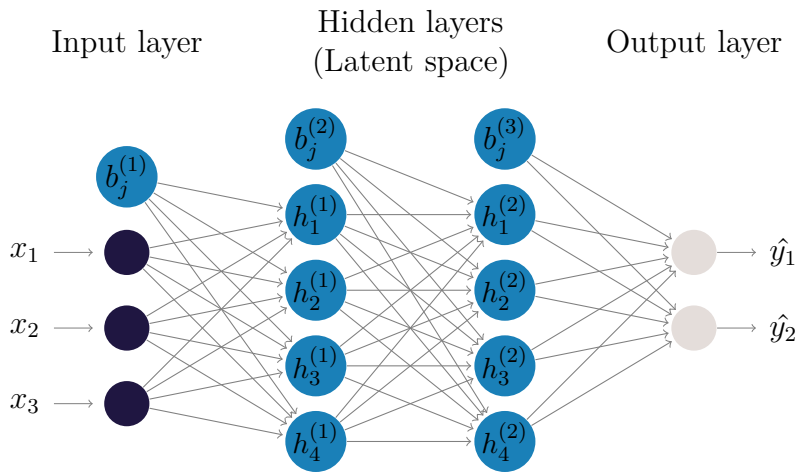


Figure 5.7: This figure illustrates a feedforward ANN. The model is composed of two latent layers ($h_\cdot^{(1)}$ and $h_\cdot^{(2)}$) as well as an input layer ($x_\cdot$) and an output layer ($\hat{y}_\cdot$). The blue perceptrons overlaid with $b_j^{(i)}$ are denoting the bias weights. The illustration has been inspired by [18].

a perceptron in a layer have been processed by the previous layer and its activations are fed into its up-coming layer. The level of feature abstraction is increasing throughout the layers in the hierarchy structure until it reaches the output layer, [19, sec. 6.4].

The ANN architecture illustrated in figure 5.7, is existing of two layers with four perceptrons in each layer. The number of layers and the number of perceptrons within each layer is an architectural design choice. The complexity of the ANN is increasing when stacking more layers and adding more perceptrons to each of the layers. The architectural design must be found by domain knowledge and trial and error, while tracking the performance of the validation set.

The mathematical description of the ANN in figure 5.7 is denoted in equation 5.14; a modified version of [19, eq. 6.40-41].

$$\begin{aligned} h_j^{(1)} &= f_j^{(1)}\left(b_j^{(1)} + \sum_i w_{i,j}^{(1)} \cdot x_i\right) \\ h_j^{(2)} &= f_j^{(2)}\left(b_j^{(2)} + \sum_i w_{i,j}^{(2)} \cdot h_j^{(1)}\right) \\ \hat{y}_j &= f_j^{(3)}\left(b_j^{(3)} + \sum_i w_{i,j}^{(3)} \cdot h_j^{(2)}\right) \end{aligned} \tag{5.14}$$

where x is the input and y is the output of the ANN model. $h_j^{(1)}$ and $h_j^{(2)}$ are representing the activations from the first and the second layer in the latent space. i is the number of outputs from the perceptrons in the previous layer and j is the number of perceptrons

in the current layer. $f_j^{(i)}(\cdot)$ is the chosen activation function for each of the layers. $b_j^{(i)}$ is the bias to each of the perceptrons j.

The reason for using an ANN model architecture to estimate the behaviour compared to a simple logistic regression model, is due to the complex properties of its latent space. The latent space will be learned through training (see section 5.5.4) and it provides the mapping; the approximated model, which describes the behaviour for a given set of input features. When the behaviour is not defined by a psychical known formula, the features to the parameterized logistic model needs to be hand engineered. The process of hand engineering input features requires a sufficient level of domain knowledge.

The learning process in an ANN model can be described as an automated form of feature engineering. The strength of the ANN model is its ability of creating the mapping between the inputs and the output by learning this mapping directly from underlying and non-observable structures of the input features.

The following sections will describe the details of a single perceptron, various activation functions, the technique of learning the latent space and several regularization approaches for an ANN model.

5.5.1 The Perceptron

A single artificial neuron, called a perceptron, is the foundation of an ANN model. The mathematical function of a perceptron is shown in equation 5.15 and a visual illustration is given in figure 5.8.

$$\begin{aligned} \hat{y} &= f\left(b + \sum_i^p w_i x_i\right) \\ &= f\left(w \odot x + b\right) \end{aligned} \tag{5.15}$$

where x is the input features, p is the number of outputs from the perceptrons in the previous layer, $\sum_i^p w_i x_i + b$ is determining the linear combinations of the inputs and can be expressed as the dot product $w \odot x + b$, b is the bias and $f(\cdot)$ is a non-linear activation function which transforms the activation of the perceptron to the non-linear output y.

5.5.2 Activation Functions

The importance of applying a non-linear activation function $f(\cdot)$ to the linear combination of $w \odot x + b$ in the perceptron, is to introduce non-linearities into the ANN model. These non-linearities are a requirement if the ANN model should be able to model non-linear behaviour. Hence, if the activation function is linear: $f(x) = x$ then by composing multiple perceptrons, the result will be a linear output [20].

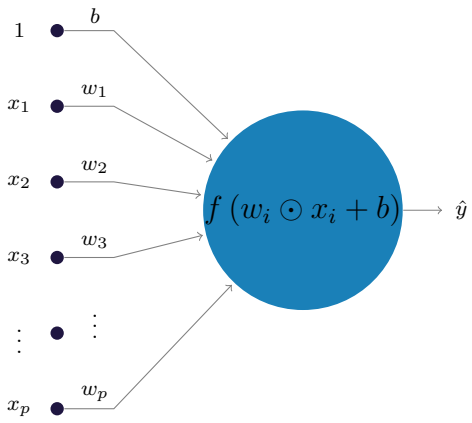


Figure 5.8: A close-up illustration of a single perceptron. The pair-wise inputs x_i and weights w_i, its bias weight b. An activation function provides the output of the elementwise multiplication and addition $\hat{y}$.

Equation 5.16 and equation 5.17 denote two of the most commonly used activation functions, each of which with different activation properties.

- The sigmoid activation function produces an activation value of the Bernoulli distribution in the range $\in]0, 1[$. This activation saturates when $w \odot x + b$ approaches $\pm\infty$ which makes the activation insensitive to changes of the perceptron activations of the previous layers [19, chap. 3.10]. The sigmoid activation function and its derivative are given in equation 5.16.

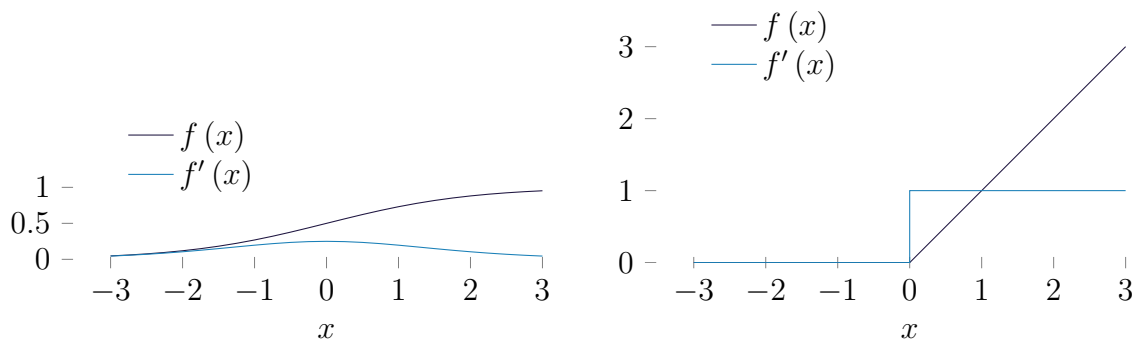
$$\begin{aligned} f(x) &= \frac{1}{1+e^{-x}} \\ f'(x) &= f(x)\,(1-f(x)) \end{aligned} \tag{5.16}$$

- The rectified linear unit (ReLU) activation function is the recommended choice for modern ANNs [19, p. 173]. The ReLU activation function and its derivative are given in equation 5.17.

$$\begin{aligned} f(x) &= \max\{0,\, x\} \\ f'(x) &= \begin{cases} 1 & , x > 0 \\ 0 & , \text{otherwise} \end{cases} \end{aligned} \tag{5.17}$$

  The piecewise linear activation function preserves the properties of a normal linear model which can easily be learned by gradient optimization [19, chap. 6.1].

Figure 5.9 illustrates both of the above mentioned activation functions.



(a) The sigmoid activation function.

(b) The ReLU activation function.

Figure 5.9: This figure provides visual illustrations of the sigmoid activation function and the ReLU activation functions in equation 5.16 and equation 5.17 respectively.

5.5.3 Derive Class Probabilities

It is possible to apply the softmax activation, which is a generalization of the sigmoid activation, when the modelling extends to predict multiple classes k.

$$S_k(y) = \frac{e^{y_k}}{\sum_{i=i}^{k} e^{y_i}} \tag{5.18}$$

The softmax function (equation 5.18) creates the mapping from the output of the second layer in the latent space (figure 5.7) to the estimated probabilities for each of the two classes [19, sec. 6.2].

5.5.4 Learning the Latent Space Directly from Data

There are 46 edges between the perceptrons in figure 5.7. Each of the edge illustrates a weight which needs to be learned in order to perform the desired output. The learning of these weights will be described in this section.

The weights require an initial value before running a forwardpass for a given set of input features. It is recommended that the initial values of the weights are either initialized to follow a uniform distribution or a normal distribution, with the range and the standard deviation given in equation 5.19 and equation 5.20 respectively, [21]–[23].

$$\mathcal{U}\left(-\sqrt{\frac{6}{j+i}}, \sqrt{\frac{6}{j+i}}\right) \tag{5.19}$$

$$\mathcal{N}\left(0, \sqrt{\frac{2}{j+i}}\right) \tag{5.20}$$

where i and j refer to the same definition as in equation 5.14.

The Xavier initialization method keeps a similar value of the gradients in all layers, [21]. The learning procedures throughout this thesis are supervised, meaning there exists a known target for given input features. The learning of the weights in the latent space is performed w.r.t. an appropriate loss function.

The objective of the learning procedure is to minimize the loss function $\mathcal{L}$ given in equation 5.22 and in equation 5.24.

5.5.4.1 The Cross-Entropy Loss

The most common loss function for classification is defined by the categorical cross-entropy function given in equation 5.21, [15, eq. 11.10]. The cross-entropy can be interpret as a distance measure between the estimated probability of the model and the one-hot encoded true target value, L.

$$\mathrm{CE}_i(\theta) = -\sum_j \log\left(S_{i,j}(\cdot)\right) L_{i,j} \tag{5.21}$$

where CE is short for cross-entropy and $S_{i,j}(\cdot)$ (equation 5.18) is the calculated softmax value for each class j for observation i. $L_{i,j}$ is a one-hot encoded matrix where the positional elements for the true class y_i are 1. The overall loss is averaged in equation 5.22 w.r.t. the current weights θ.

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_i^N \mathrm{CE}_i(\theta) \tag{5.22}$$

where N is the total number of samples used in the evaluation of $\mathcal{L}(\theta)$.

The loss function is non-negative and its value approaches 0 when the model iteratively learns the underlying structure of the latent space.

5.5.4.2 The Connectionist Temporal Classification Loss

The connectionist temporal classification (CTC) method has been developed by Graves, Fernández, Gomez, *et al.* [24] for tackling real-world sequence learning tasks. This section introduces an overview of the CTC loss function. Further technical insight is referred to [24].

The CTC function The main objective of the CTC function is to solve the alignment of the mapping between the input X and the output label Y, where the length of both X and Y can variate for each sample. Figure 6.1 illustrates the inner-workings of a CRNN model and the specific position and duty of the CTC function.

Suppose X is given in figure 4.3a and its label is Y = GMXA245. The mapping between X and Y is achieved by maximizing the conditional probability denoted in equation 5.23, [25].

$$Y^* = \underset{Y}{\operatorname{argmax}}\, p\left(Y|X\right) \tag{5.23}$$

where $p(Y|X)$ is the conditional probability. Y^* is the optimal output of the CRNN model, which is a tensor with the following dimensions: $N \times T \times L$. N is the size of the batch, T is the number of time steps and L is the length of the character vocabulary plus one. This extra token is a "blank" which is used in the decoding procedure.

The loss of equation 5.23 is obtained by minimizing the negative log-likelihood in equation 5.24.

$$\mathcal{L}\left(\theta\right) = \sum_{i}^{N} -\log p\left(Y_i|X_i\right) \tag{5.24}$$

where θ defines the parameters of the CRNN model. N is the total number of samples. $p(Y|X)$ is the conditional probability of the input X and the output Y.

As demonstrated in [24], [25], the loss function in equation 5.24 is differentiable w.r.t. the output per time step. It is possible to perform back-propagation and apply a proper gradient decent method to learn the optimal model parameters.

The decoding procedure There exist several methods to decode the target label from Y given X. A simple heuristic is to choose the most likely output character per time step, also known as the best path. The mathematical notion for this is given in equation 5.25, [25].

$$A^* = \underset{A}{\operatorname{argmax}} \prod_{t=1}^{T} p_t\left(a_t|X\right) \tag{5.25}$$

where t is the time step and T is the number of time steps. a_t is a sliced probability vector of A. A^* contains the most likely character per time step. Repeated characters have to be collapsed and the "blank" tokens are removed from A^* in order to get Y. This method is further described in [24].

It is possible to extent the decoding produce to a beam search which enables the model to return two or more likely predictions for a given X. However, this simple heuristic will be used as the starting point.

5.5.4.3 Gradient Decent Method

The approach to minimize the loss function, e.g. equation 5.22, can be achieved by optimizing the weights θ using the principle of gradient decent. Recall, how the gradients

illustrate the curvature of the loss function for a given set of parameters θ. Figure 5.10 shows an illustration of the loss function as a function of θ.

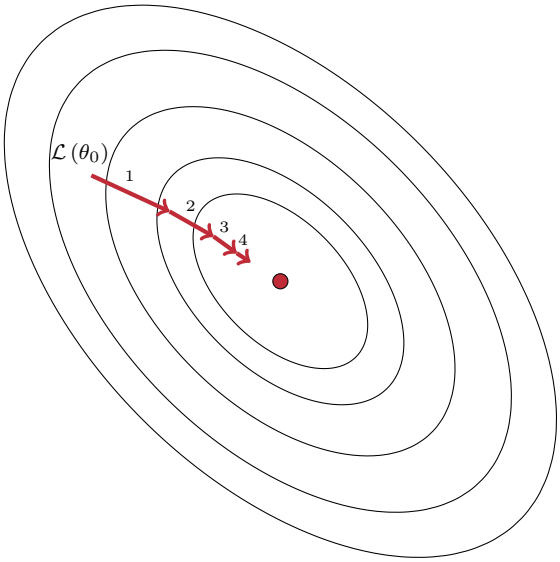


Figure 5.10: This figure shows four iterations of the gradient decent method. The step size of the change in parameter value is decreasing as the method approaches the minima of the loss function.

The gradient decent method is an iterative process and the process is stopped when the θ values converges, hence a minima of the loss function is achieved. Figure 5.10 illustrates four iterations of the gradient decent method. Each of these four steps can be described by equation 5.26.

$$\theta_{j,i+1} = \theta_{j,i} - \eta \frac{\partial \mathcal{L}(\theta_{j,i})}{\partial \theta_{j,i}} \tag{5.26}$$

where η determines the step in the negated direction in the gradient landscape. This parameter is also known as the learning rate. j defines the variable and i denotes the iteration of the gradient decent method.

The gradient decent method will converge and the $\mathcal{L}(\theta)$ is zero if equation 5.26 gets stuck in a local minima or a local saddle point. Other issues such as noise gradients occurs when the learning produce uses mini-batch (section 6.3). As the simple illustration in figure 5.10 shows, the loss function can be difficult to optimize as the number of parameters explode in more complex ANN architectures.

Multiple gradient decent methods (optimizers) exist each of which with different properties. This project uses the Adam optimizer which is created by Kingma and Ba [26]. The name of the Adam optimizer is derived from ”adaptive moment estimation” and is suggested as the default gradient decent method by [22], [27]. The suggested initial values for Adam are given in equation 5.27.

$$\begin{aligned} \eta &= 5e^{-4} \\ \beta_1 &= 0.9 \\ \beta_2 &= 0.999 \end{aligned} \tag{5.27}$$

The main properties of the Adam optimizer are:

- An individual adaptive learning rate for each of the weights determined by estimates of its respective first order moment and second order moment of its gradient.
- A bias correction term which estimates an unbiased value of the first and second order moments prior to the update step. This step will minimize the artefact of the estimate of the second order moment in the first iteration. The initial value of second order moment causes a huge updating step, which is reduced by the bias correction.

For further properties of the Adam optimizer, please consult [26].

5.5.4.4 Back-propagation

The gradient decent method is responsible for the iterative learning process. This method relies on the back-propagation method calculating the gradient of θ_j w.r.t. the backward propagated error $\mathcal{L}(\theta)$. Figure 5.11 provides a simple ANN model and the gradients of θ_1 and θ_2 are given w.r.t. $\mathcal{L}(\theta)$ in equation 5.28, [20].

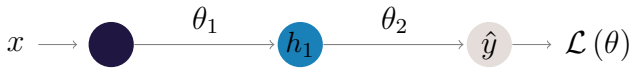


Figure 5.11: This figure illustrates an ANN. These notations are used to demonstrate the back-propagation method reported in equation 5.28.

$$\begin{aligned} \frac{\partial \mathcal{L}(\theta)}{\partial \theta_2} &= \frac{\partial \mathcal{L}(\theta)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_2} \\ \frac{\partial \mathcal{L}(\theta)}{\partial \theta_1} &= \frac{\partial \mathcal{L}(\theta)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial h_1}{\partial \theta_1} \end{aligned} \tag{5.28}$$

The back-propagation is recursively applying the chain rule of calculus [19, chap. 6.5]. It is possible to define the gradient w.r.t. all the θ weights by recursively applying the chain rule, equation 5.28.

5.5.4.5 When to Stop the Learning Process

The learning process is normally predefined to N_{epoch} iterations of a gradient decent method. The ANN model is learning a better representation of its latent space, which results in a decrease of the loss $\mathcal{L}$ for each iteration. When letting a gradient decent method iterate endlessly, the loss of the training set will approach zero. However, the loss of the validation set will start to increase again at some point. Figure 5.12 illustrates a synthetic learning procedure iterating for $N_{\text{epoch}} = 20$ epochs.

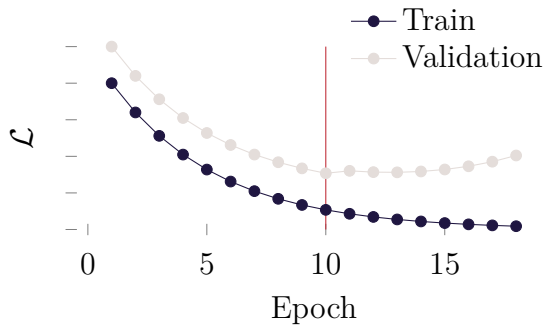


Figure 5.12: This figure shows a synthetic illustration of the training loss and the validation loss. The vertical red line denotes the ideal stopping point for the training procedure as the variation between the learning curves increase.

The vertical red line in figure 5.12, illustrates the transition between underfitting and overfitting the training set and the validation set. See further descriptions of training terminologies in section 6.3.

Key points To this point the takeaways from section 5.5:

- An ANN model is composed of hundreds of thousands of millions perceptrons, all of which are divided into multiple hierarchical layers.
- The single perceptron computes a dot-product; is adding a bias and is performing a non-linear activation. The ReLU function is the recommended activation function for modern ANNs.
- The learning procedure needs to combine the gradient decent method and the back-propagation concept. The latter mentioned concept is able to compute the derivative of a given model parameter in order to update the parameter.
- The learning procedure needs to be controlled by a validation set in order to avoid overfitting, see illustration in figure 5.12. However, section 5.5.8 will introduce several regularization approaches, to pursue a better ability of generalization of the ANN model.

5.5.5 A Convolutional Layer

A convolutional neural network (CNN) layer is a specific type of ANN which are employing the mathematical *convolution* operation within one of its layers [19, chap. 9]. An example of this convolution operation with a flipped kernel is given in equation 5.29 ([19, eq. 9.6]) and visual illustrations are provided in figure 5.13.

$$\begin{aligned} S(i, j) &= \sum_m \sum_n I(i+m, j+n) \cdot K(m, n) \\ &= (I \odot K)(i, j) \end{aligned} \tag{5.29}$$

where $S(\cdot)$ (in CNN terminology) is referred to as the feature map, I denotes the input image or a previous computed feature map and K is the kernel which includes learnable weights. The main objective of the kernel K, is to learn an appropriate feature extractor, which can detect meaningful features such as edges, textures and shapes.

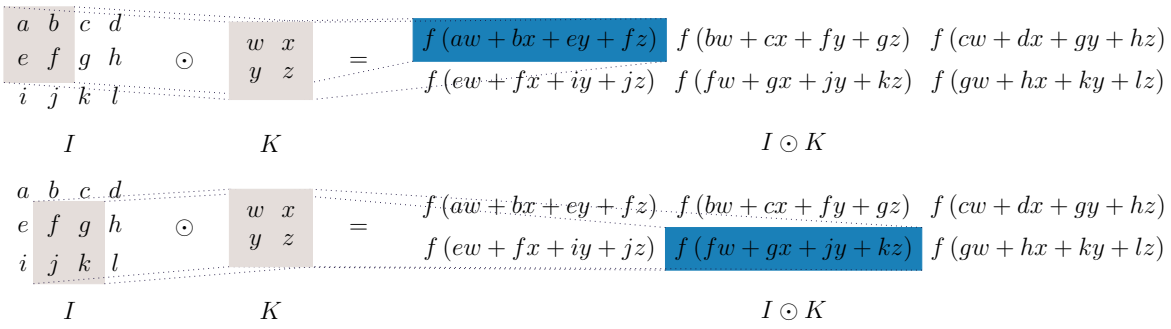


Figure 5.13: This figure illustrates a 2D convolution operation of a $4 \times 3 \times 1$ channel image I with a unit stride and no padding. The kernel K is 2×2 which creates a reduced feature map with the dimensions $3 \times 2 \times 1$. This figure is inspired by [19, fig. 9.1].

The definitions of I and K are referring to the above mentioned and $f(\cdot)$ is the default ReLU activation function defined in equation 5.17.

When a kernel is flipped as in equation 5.29, the correct name of the illustrated operation is denoted as the cross-correlation operation. However, the gradient decent method will learn the appropriate weights of the kernel whether the kernel is defined by the convolution operation or by the cross-correlation operation. The convention of the convolution operation is used for both operations throughout this project.

Figure 5.14 shows a single CNN layer. The single layer is composed of three different sub-layers which are; a convolution layer, a non-linear activation layer and a pooling layer.

Goodfellow, Bengio, and Courville mentioned three important leverages of the convolution operation, which are applied in the first sub-layer of a CNN layer. A short description for each of the properties is described below, [19, chap. 9.2].

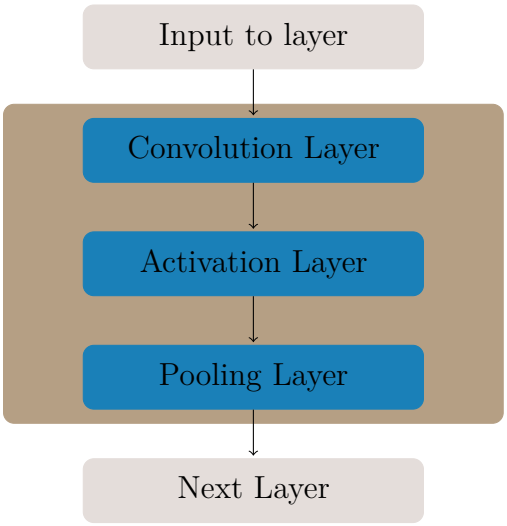


Figure 5.14: This figure illustrates the three sub-layers in a CNN layer. The illustration is inspired by [19, fig. 9.7].

- Sparse interactions: In the CNN terminology, the sparsity means a lack of interaction between each of the input perceptrons to each of the output perceptrons. This is caused by the convolution operation which computes the feature map due to the properties and dimensions of the kernel convolution (K) w.r.t. the input I. The reduction in dimensionality entails a smaller memory footprint and requires fewer operations in order to compute a new feature map $S(\cdot)$.
- Parameter sharing: As illustrated in figure 5.13, the weights in K are shared for each location of the convolution operation in I.
- Equivariant representations: The article by Lenc and Vedaldi [28] provides an excellent description of the properties of an equivariant function. Hence, the representation will capture the features of an object and if the object is transformed, e.g. rotated 90°, the feature map will reflect the same transformation.

Pooling The final sub-layer in figure 5.14 is the pooling layer. The property of the pooling operation is to reduce the spatial dimension of the feature map in the second sub-layer (figure 5.14) and become invariant to small differences in the feature map produced by the non-linear activations.

The most common pooling function performs a max-pooling operation, where the maximum value, within the determined area of neighbouring pixels, is returned. Figure 5.15 illustrates the max-pooling operation.

It is possible to perform pooling over the spatial activations in order to be invariant to small differences in the feature map. To perform pooling over a set of feature maps the model will learn which of the transformations it needs to become invariant to [19, chap. 9.3].

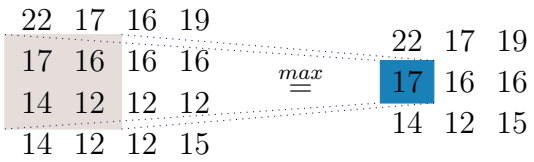


Figure 5.15: This figure illustrates the max pooling operation with unit stride and no padding. The values to the left are synthetic non-linear activation values and the values to the right are produced by the max pooling operation.

Depth, stride and padding The spatial arrangement of the output (figure 5.14) is determined by the following hyperparameters [29]:

- Depth is determined by the number of kernels applied.
- Stride defines the shift of the kernel or of the pooling operation. The shift is determined by the number of pixels. A unit stride is defined by: $s = 1$
- Padding is an essential mechanism which concatenates synthetic zero-valued pixels at the borders of input. The number of zero-valued pixels are dependent of the size of the kernel. Suppose an image is convoluted by a kernel of the size 3×3 with unit stride and we want to preserve the size of the output. This is obtained by using half padding $p = \left\lfloor \frac{k}{2} \right\rfloor = 1$, where k is the diagonal length of the kernel and p is the number of concatenated zero-valued pixels.

The hyperparameters of the CNN block in the first stage model of this project, illustrated in figure 5.14, have a kernel size of 3×3, half padding and a unit stride. The pooling operation uses a size of 2 and a stride of 2, which will half the spatial dimension of the output w.r.t. the input.

Key points The takeaways from section 5.5.5 are:

- The convolution operation uses parameter sharing, introducing sparse interactions and equivalent representations, all of which provide the ability to learn multiple meaningful feature detectors, which are applied to multiple spatial patches of the input.
- The pooling operation introduces the ability to become invariant to small differences within the activations of the feature maps.

5.5.6 A Recurrent Layer

A recurrent neural network (RNN) layer is a special case of the simple feedforward ANN, illustrated in figure 5.7 and equation 5.14, specialized for sequence problems.

A RNN layer is composed of a special kind of perceptrons which computes its current state dependent on the state from previous time step(s). The current state $h^{(t)}$ can be expressed according to equation 5.30 which denotes the mathematical notion of a recurrent perceptron, [19, eq. 10.8].

$$h^{(t)} = g\left(Wx^{(t)} + Uh^{(t-1)} + b\right) \tag{5.30}$$

where g is a hyperbolic tangent activation function, which squeezed the activation between ± 1, b is a bias vector, W and U are corresponding to the external and internal weight matrices of the perceptron.

The estimated output at a current time step t can be derived according to equation 5.31.

$$\hat{y}^{(t)} = S_k\left(Vh^{(t)} + c\right) \tag{5.31}$$

where $S_k(\cdot)$ is the softmax activation function from equation 5.18, V is a weight matrix and c is a bias vector.

Figure 5.16 shows an unfolded illustration of the recurrent perceptron across time. The function to determine the new state of the perceptron $(t+1)$ is calculated by an update of equation 5.30 and the estimated output from $t+1$ is calculated by equation 5.31. In these two equations it is notable that the recurrent perceptron uses parameter sharing throughout the unfolded sequence, which is a useful property for predicting variable sequences.

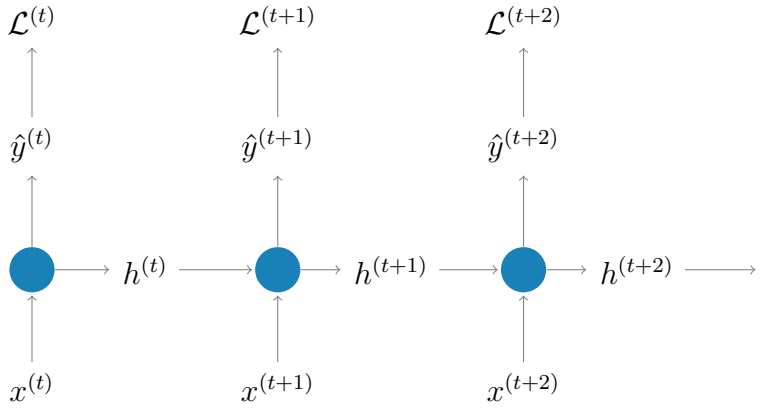


Figure 5.16: This figure illustrates a single recurrent perceptron unfolded across time t. The provided update of equation 5.30 is calculated for each time step. The weight matrices W, U and V and the bias vectors b and c are shared across time.

By unfolding the single recurrent perceptron across time, is it possible to see how the current state and output is produced w.r.t. the past sequence of states and inputs. Hereby is a recurrent layer able to learn to predict the future from the past.

Bidirectional RNNs A unidirectional RNN has a causal structure, which means it is only using the previous steps to predict the present step. The bidirectional RNN structure is allowing the output unit to predict the present step based upon past and future information. The bidirectional structure is used in the application where predicting the present step may depend on the entire input sequence [19, sec. 10.3].

Back-propagation The technique for learning the parameters W, U, V, b and c is similar to the mentioned back-propagation method (section 5.5.4.3) despite the extra time dimension. As illustrated in figure 5.16, it is possible to produce an estimated output $\hat{y^{(t)}}$ for each time step and hereby obtaining a loss for each of the time steps. The total loss is simply the summed loss across time.

Equation 5.32 shows the derivative of W w.r.t. its loss $\mathcal{L}$ for $t = 2$.

$$\frac{\partial \mathcal{L}^{(2)}}{\partial W} = \frac{\partial \mathcal{L}^{(2)}}{\partial \hat{y}^{(2)}} \frac{\partial \hat{y}^{(2)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial W} \tag{5.32}$$

However, it is not possible to treat the final term as constant since $h^{(2)}$ depends on its previous state (equation 5.30). The final term has to be expanded to include the previous time steps. The contributions of the earlier time steps to the error in $t = 2$ are achieved by aggregating the contributions.

Equation 5.33 shows the derivative of W w.r.t. its loss $\mathcal{L}$ for t as a general sum, [20].

$$\frac{\partial \mathcal{L}^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial \mathcal{L}^{(t)}}{\partial \hat{y}^{(t)}} \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(k)}} \frac{\partial h^{(k)}}{\partial W} \tag{5.33}$$

The recurrent perceptrons can be hard to train when the sequences become larger than the concept of vanishing gradients may occur. The third term (equation 5.33) expands into a product of t terms and due to the activation function, used to update the state of the perceptron, the terms consist of small numbers less than one. By multiplying these numbers the phenomenon of the vanishing gradients may occur. In the high-level abstraction, the phenomenon limits the ability of the perceptron to learn long term dependencies within the sequences, due to a gradient close to zero [20].

It is possible to increase the complexity of the architecture of the recurrent perceptron and hereby create a more suitable flow of the gradient which will prevent the phenomenon of vanishing gradients.

An enhancement of the architecture of the recurrent perceptron is to apply a gated recurrent perceptron [19, sec. 10.10]. The above mentioned increased complexity can be obtained by the gated recurrent unit (GRU) or by the Long short-term memory (LSTM) unit for improving the flow of the gradients. The architecture of the GRU is simpler and contains fewer model parameters compared to the LSTM unit. Since the amount of training samples in this project is limited, the GRU is the preferred choice.

5.5.6.1 Gated Recurrent Unit

The main concept of a GRU is its ability to create short cuts to the previous states. This creates a better flow of the gradients and hereby the likelihood of learning the long term dependencies.

The update function ($h^{(t)}$) of a GRU given in equation 5.34 is composed of an interpolation between the previous state and a candidate of the present state.

$$h^{(t)} = u^{(t)}\tilde{h}^{(t)} + (1 - u^{(t)})h^{(t-1)} \tag{5.34}$$

where the update gate $u^{(t)}$ determines how much the unit needs to update its current state w.r.t. the past and the present candidate activation $\tilde{h}^{(t)}$. The candidate activation $\tilde{h}^{(t)}$ is similar to the update function in equation 5.30 and beyond this includes a reset gate $r^{(t)}$. The candidate activation, the update gate and the reset gate are given in equation 5.35, equation 5.36 and equation 5.37 respectively.

$$\tilde{h}^{(t)} = g\left(Wx^{(t)} + U(r^{(t)} \odot h^{(t-1)}) + b\right) \tag{5.35}$$

$$u^{(t)} = f\left(W_u x^{(t)} + U_u h^{(t-1)} + b_u\right) \tag{5.36}$$

$$r^{(t)} = f\left(W_r x^{(t)} + U_r h^{(t-1)} + b_r\right) \tag{5.37}$$

where $f(\cdot)$ is the sigmoid activation function from equation 5.16. $g(\cdot)$ is the hyperbolic tangent activation function. W, U, W_u, U_u, W_r, U_r are weight matrices and b, b_u and b_r are bias vectors.

The main property of the reset gate ($r^{(t)}$) is to learn when the captured long term dependencies are not relevant. When its value is approaching zero $r^{(t)}$ allows the unit to forget the captured long term dependencies [30].

Key points An essential difference between the recurrent perceptron and the GRU is the feature of combining the knowledge of the previously time with new information by an interpolation of these. This extents the ability of accumulating information and learning long term dependencies.

Properly the most important feature of the GRU, compared to the recurrent perceptron, is its ability of creating short cuts which can by-pass multiple of the previous temporal steps and hereby prevent the phenomenon of vanishing gradients.

5.5.7 Transfer Learning

The concept of transfer learning refers to transferring the parameters of a model, which have been properly learned so its mapping between the inputs and outputs entail a high

performance and a low generalization error for a given situation. When transferring the learned parameters it is assumed that the learn mapping is relevant to explain the variation for the new situation. If this assumption is valid, the learned parameters can be applied to another different situation as an initialization of parameters [19, sec. 15.2].

5.5.7.1 ImageNet Data Set

The ImageNet Large Scale Visual Recognition Challenge [31], often referred to as the (ILSVRC), is a data set used as a benchmark for testing new CNN architectures. The ImageNet consists of 1,000 various separate categories of objects which we encounter in our daily lives. This data set is divided into three partitions, each of which includes $\approx 1.2M$, $50K$ and $100K$ for the training set, validation set and test set respectively. The diversity of objects within the data set forces the feature extraction, which primarily is carried out in the CNN layers of the ANN model, to look for appropriate generalizable features, such as edges, textures and shapes. The feature extraction in the early layers are more generic whereas the extraction in later layers is more general for the specific data set.

The concept of transfer learning is highly applicable for models trained on the ImageNet data set, as the CNN layers are performing general feature extraction.

5.5.7.2 Approaches to train the network further

Several approaches exist to train the ANN model further. It has been initialized with parameter values found by the ImageNet, some of which are highlighted in the bullets below [32].

- Freeze the transferred initialized parameter values of the CNN layers from ImageNet and only train the last few fully connected layers of the ANN model until it converges. This approach has been applied to the ANN model in the first stage module, since the data set are relative small and prone to overfitting.

- It is possible to start with a few epochs of the mentioned method above and then unfreeze some CNN layers, decrease the learning rate of the optimizer and continue training until it converges. This approach is typically known as ”fine tuning” the parameters of the model.

- A third approach would be to start the training from the transferred initialized parameter with a sightly lower learning rate and then train until the model converges. The reason for applying the lower learning is due to the assumption of high similarity between each of the situations and also to avoid a huge distortion of parameters in the landscape of the gradients.

The input images need to be preprocessed prior to feeding the pre-trained model weights. The preprocessing used in the ImageNet application is to zero-center by the mean pixel in each of its three channels. The mean pixel values are given in equation 5.38 [33].

$$\begin{aligned} \mathrm{R} &= 123.680 \\ \mathrm{G} &= 116.779 \\ \mathrm{B} &= 103.939 \end{aligned} \tag{5.38}$$

Key points It is often not possible to train an ANN model, where all of the parameters initialized, from scratch and hereby obtain a great performance on the validation set.

Deeper ANN models are often composed of several million parameters and the amount of data is often limited to a few thousand samples. The training of the model then tends to overfit the training set and the performance of the model does not generalize to unseen data.

5.5.8 Regularization of ANN Models

The essential approach for improving the ability of generalization of an ANN model is to apply regularization techniques. The regularization techniques for an ANN model differs from techniques used in e.g. simple linear models, but the main objective remains similar. The ANN needs to perform on the validation set. The most common regularization techniques are mentioned in the bullets below. A thorough description of the dropout method will be given, since the method is applied during this thesis.

- The dropout method: The main idea is to temporarily randomly remove a percentage (p) of the perceptrons in a layer for each training step. The effect of stochasticity within the ANN is achieved by randomly removing another p of the perceptrons in the next training iteration and hereby lowering the generalization error [34].
- Data argumentation: The objective of data argumentation is to increase the amount of training data. Techniques of increasing the amount of data in an image setting could be; to create a horizontal flip of an image, cropping different patch sizes of an image, perform colour adjustments, change the sharpness or the brightness of an image [22].
- Batch normalization: The idea of using batch normalization after a fully connected layer is to reduce the change in magnitude of the activation value, also called the covariance shift, from each of the perceptrons. The batch normalization makes sure that the activations does not saturate, which enables the possibility of using a higher learning rate (η) and implies less consideration of the initialization of the weights [35].

The dropout method The definition of the dropout method is to temporarily remove all incoming and outgoing activations of a perceptron; both hidden and visible perceptrons. This method is recommended as a regularization technique in the article by Srivastava, Hinton, Krizhevsky, *et al.* [34]. A visual illustration of the dropout method is shown in figure 5.17. The illustration shows temporarily removed perceptrons within the fully connected hidden layers with a dropout proportion of $p^{(1)} = 25\%$ and $p^{(2)} = 50\%$. The dropout probability p is a hyperparameter, which can be found w.r.t. the validation set. A default value of $p = 50\%$ is suggested by [34].

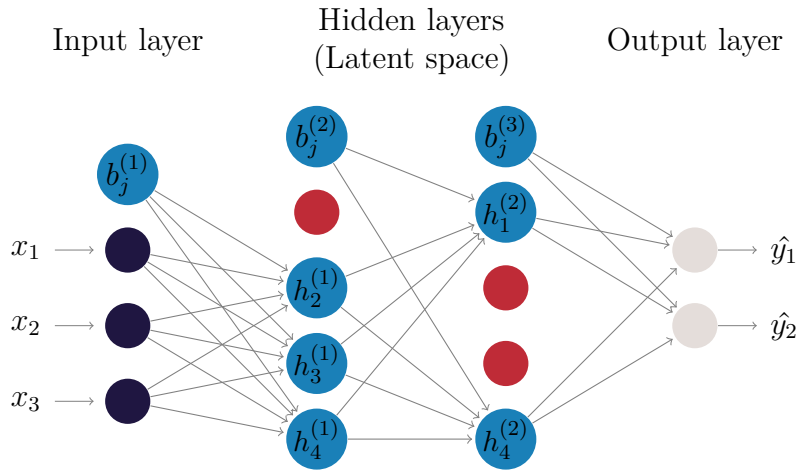


Figure 5.17: This figure illustrates the dropout regularization method. The red perceptrons are randomly are removed from the ANN model temporarily for the current training step. The dropout method makes the ANN model robust and prevents overfitting.

The dropout method is reducing the number of active perceptrons in the latent space of the ANN. It turns the interpretation of the ANN model into an ensemble of ANN models, which are sharing weights by randomly dropping the perceptrons during the training time.

All of the perceptrons are active during test time of the model but the activations need to be scaled by p in order to achieve the initial ANN model trained without dropout [22], [34].

5.6 Bias and Variance

Bias and variance are the two major sources of error contribution in machine learning algorithms. Figure 5.18 provides an illustration of the four combinatorial situations of bias and variance.

Suppose a machine learning algorithm tries to predict a continuous response y, marked by the vertical line in figure 5.18. An ideal prediction of the continuous response y is obtained by a low bias and a low variance. When the predicted value ($\hat{y}$) is off-set

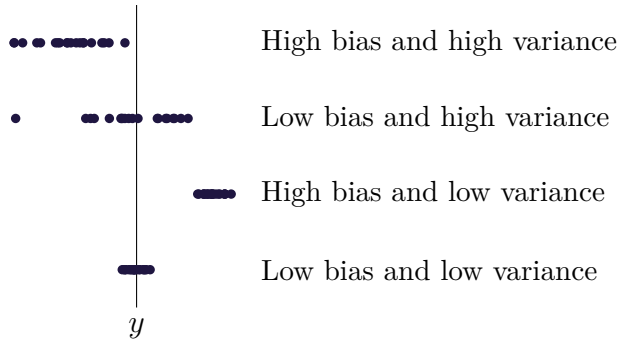


Figure 5.18: This figure illustrates the concept of bias and variance. y denotes the true value and the dots are denoting observed samples. The illustration is inspired by [36, lecture notes].

compared to the true value (y) the model has high bias. High variance in the prediction provides information of the uncertainty of the predicted response $\hat{y}$. Understanding the concept of bias and variance can help to decide which tactics to apply when improving the performance of the model.

Suppose a synthetic requirement from the UFST division is to obtain a maximum classification error rate of 4%. The error rate of the first model iteration is 11% and 15% for the training and validation set respectively (described in section 6.3.1).

The proper way to informally connect the concept of bias and variance to the values of error rates are mentioned below [3]:

- Bias is the error rate of the training set (11%).
- Variance: The difference between the error rate of the validation set (15%) and the error rate of the training set (11%) is 4%, which is interpret as the generalization error.

The concept of bias and variance will be used later in the error analysis in chapter 7 and possible model enhancements will be discussed in order satisfy the synthetic maximum error rate of 4%.

5.7 Interpretation of an ANN Model

This section introduces two frameworks which can be used in the interpretation of the structure of an ANN model and hereby obtain an insight of the inner-workings of the model. The first presented framework is called LIME which is a versatile framework able to explain a prediction of any predictive model. The LIME framework is further explained in section 5.7.2. The second presented framework is called Grad-CAM and is further explained in section 5.7.3.

The ability to interpret the inner-workings of a complex ANN model is essential for trusting the model. If the end user is able to the interpret the structure of the model, it is more likely that the end user will trust its predictions and base further actions upon its predictions [8], [9].

The main focus of this section is directed to the interpretation of the first stage CNN model (see section 6.1.2), but similar frameworks can easily be applied to the second stage model, as demonstrated in work by Wojna, Gorban, Lee, *et al.* [37] as well as other types of models.

The two main drivers for the ability to interpret a complex ANN model are:

- to evaluate their plausibility and understand their local decision boundaries which are derived in order to produce certain predictions for certain given inputs. This is essential for the trust of the model.
- to be compliant with the General Data Protection Regulation (GDPR) utilized as law in the European Union of May 25th 2018 [38].

The guidelines of "how to be" GDPR compliant are described in the up-coming sections as well as the mentioned frameworks for evaluating, and for achieving an understanding of the decision boundaries of an complex ANN model.

5.7.1 To be GDPR Compliant

The GDPR is an enactment to ensure and to protect the processes of treating personal information regardless of nationality or demographics to uphold the civic rights[38].

The following three legislations are the matter in question of applying personal information within the life cycle of model development, the model deployment process and the model production process. The bullets below must be fulfilled in the technical pipeline in order to be GDPR compliant.

- It is forbidden to develop predictive models which are based upon human discriminational features such as racial origin, ethnic origin, political opinions, religious beliefs and etc. according to GDPR article 9 [38].
- According to the GDPR article 13.2.f it is a requirement that automated decisions are derived by a fair and a transparent process. This introduces the "right to an explanation" of a prediction and a meaningful insight of the inner-working logic of the model.
- The GDPR article 17.2.f describes the "right to be forgotten" which refers to erasing all personal information of a subject of interest. The "right to be forgotten"

will hereby also influence the trained models and hereby raise a demand for re-training the model on the new data foundation without the subject of interest.

The data foundation of this project includes data with personal information and since the primary objective of this project is to develop an automatic decision-making process for production, the models within this pipeline are attached under the GDPR.
The two VRDs in figure 2.2 and figure A.1 include personal information (these are censured by the white boxes with red edges), which can be used to direct or indirect identification. Since the input data to the models of the technical pipeline are images and some of the images contains personal information, the entire pipeline is affected by the strict regulations in the GDPR article 13.2.f and the GDPR article 17.2.f [38].

5.7.2 Local Interpretable Model-agnostic Explanations

The local interpretable model-agnostic explanations (LIME) framework has been introduced by Ribeiro, Singh, and Guestrin [8]. The goal of the LIME framework is to derive an interpretable model which is able to explain the complex model locally. In other words, the LIME framework attempts to explain a prediction, determined by a complex model, by locally approximating an interpretable model.

The assumption The following assumption has been made in order to explain a prediction by a local interpretable linear model: A complex model has linear decision boundaries on its local scale ([8, fig. 3]) and it is hereby it is possible to use a linear model for the explanation [39].

In order to make the explanation, the input data has to be permuted in order to approximate the local decision boundaries and the linear model. The permutation is depended on the type of input data. The rest of this section will be focusing on images as input data, introducing the concept of super-pixels demonstrated on few examples. Please consult [8] for further technical details.

The proper explanation, when having images as input data, would be to highlight areas supporting the prediction. The supporting areas should give an intuition of the reason for the prediction. The segmentation of these areas are determined by its defined super-pixels.

Super-pixels A super-pixel is defining a set of an arbitrary pixel area in the image where the arbitrary pixel area is constituted of a high similarity. The super-pixels are performing a segmentation of the image based upon the defined hyperparameters for the super-pixels. It is essential to find the ”correct and meaningful” set of hyperparameters to get a reasonable segmentation by the super-pixels.

The explanation The explanation depends on a few hyperparameters. The article ([8]) and their Github repository does not provide a heuristic for these hyperparameters. The bullets below introduce the parameters used in the explanation:

- The number of super-pixels per image (n_{sp}) determine the number of separate areas within the image, which correspond to the length of the binary vector used in the linear model.
- A weighting parameter (w) determines the trade-off between the locality of the super-pixels and the colour of the super-pixels. A high value of w causes the super-pixels to be more compact with a rectangular shape. A low value of w causes the super-pixels to follow the spatial structure and the colour of the image.
- The number of super-pixels (n_{expo}) to include in the explanation is referring to the most prominent super-pixels in the linear model with the highest contribution to the prediction. These prominent super-pixels are found by feature selection.

Figure 5.19 illustrates the effect on the explanation for three different hyperparameter settings. It has been chosen to include the same number of explanatory super-pixels in all three examples in order to limit the parameter space. The illustrations in subfigure 5.19a to subfigure 5.19c shows the super-pixels delimited by the black lines. The three explanations are given in subfigure 5.19a to subfigure 5.19c.

Key Points Figure 5.19 demonstrates how the change in the hyperparameter settings are affecting the explanation for the same input image. It is impossible to state whether these three examples illustrate one of the most correct interpretations of the prediction produced. However, the LIME framework does agree on the left upper most super-pixels to be prominent for an explanation of the prediction, regardless of the hyperparameters for these three examples in general.

It is important to notice, that the LIME interpretation only provides an approximated insight of the complex model. It is possible to apply other frameworks such as a Saliency Maps, class activation maps [40] or gradient class activation maps [9]. The latter two are class discriminative and hereby preferable. Gradient class activation maps do not require a change in the model architecture and is hereby more applicable than the class activation maps.

5.7.3 Grad-CAM

The Grad-CAM method has been proposed by Selvaraju, Das, Vedantam, *et al.* [9] and is a generalization of the discriminative class activation map (CAM) as proposed by Zhou, Khosla, Lapedriza, *et al.* [40]. Both of these methods can be used for visual explanations by evaluating the perceptrons in a certain layer of a CNN model.

(a) $n_{sp} = 20$ and $w = 25$. (b) $n_{sp} = 20$ and $w = 10$. (c) $n_{sp} = 30$ and $w = 25$.

(d) $n_{sp} = 20$ and $w = 25$. (e) $n_{sp} = 20$ and $w = 10$. (f) $n_{sp} = 30$ and $w = 25$.

Figure 5.19: This figure shows three examples of visual explanations using three different hyperparameter settings. Common for all examples are the number of super-pixels to include in each explanation $n_{expo} = 5$ and the class to be explained. The first example, in figure 5.19a and in figure 5.19d, uses $n_{sp} = 20$ and $w = 25$. The second example and the third example illustrate the effect of changing the n_{sp} and w separately in figure 5.19b and 5.19e and in figure 5.19c and 5.19f respectively.

Grad-CAM is short for Gradient-weighted Class Activation Mapping and the explanation L^c for the class c is given in equation 5.39 and in equation 5.40, [9, eq. 1-2].

$$L^c = f\left(\sum_k \alpha_k^c A^k\right) \tag{5.39}$$

where L^c is a linear weighted combination of the feature maps A^k and the weights α_k^c. The linear combination is activated by $f(\cdot)$ which is the ReLU activation function given in equation 5.17. This activation ensures the perceptrons with a positive influence on c is represented in L^c. Equation 5.40 denotes the formula for α_k^c, which is determined by the score y^c of the class c produced by a forward pass through the CNN model and the feature maps A^k from an CCN block.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{GAP} \frac{\partial y^c}{\partial A_{ij}^k} \tag{5.40}$$

where c is denoting the target class. Z is the product of the dimensions i and j of the feature map A^k where k is a specific map. A^k is typically determined as the latest convolutional layer in a CNN model, since this is the layer preserving the spatial information. y^c represents the score of class c before entering the softmax activation function. This score is produced by a forward feed for a given input. GAP is short for global average pooling and is used to average the gradients calculated in the back-propagation. The average pooling makes the estimation of α_k^c less prone to noise of the gradients [9].

Figure 5.20 provides an illustration of L^c for c = DEs where y^c is determined by the input image in figure 4.2b. α_k^c is calculated w.r.t. A^k being the block5_conv3 layer of the optimized CNN model described in section 6.1.2. The heat map given by L^c have been rescaled and placed above the input image. High intensity values in the heat-map are reflecting areas which contribute to the class prediction c positively.

Figure 5.20: This figure illustrates the discriminative class location based upon the learned layers of the CNN model. The locations with high intensity pixels are referring to areas of the image which are effecting the prediction of the class of interest positively.

Key points The visual explanation (L^c) for c = DEs produced by the Grad-CAM method in figure 5.20, provides a unique insight of the most contributing area of the prediction produced by the optimized CNN model.

This explanation can be used to validate the explanations produced by the LIME framework, since the explanation of L^c is included in all three explanations in figure 5.19.

CHAPTER 6
Model Implementation

This chapter is composed of three sections. The two first sections are introducing the considerations and implementations of the models for the first and second stage modules in the technical pipeline. Both the sections are introducing a baseline model, which is used for a baseline reference, and are introducing specific types of competitive artificial neural network models.

The third section introduces practical methodologies of the model training including; the partition of data sets and the reasons for choosing a single metric for model evaluation.

An overall requirement of the model implementation is the restriction to the `R` programming language due to UFST's production environment. The neural network models have been developed through the highlevel `Keras` API with the `Tensorflow` framework as backend.

It has been chosen to make the code of the recommended technical pipeline and the object of the models open sourced. The pipeline has been structured as a `R-package` which is reached here: Github repository [1].

6.1 First Stage Models

The primary task of the first stage module is to classify the class of each of the pages within a VRD. The resolution of a VRD document is typically thousand $\times$ thousand pixels and it therefore has to be downscaled in order to be computationally efficient [5]. The pre-processing module of the first stage is downscaling and resizing each of the pages within the VRD to a dimension of $224 \times 224 \times 3$. This input dimension is used in [41].

The proposed baseline model is the HOG Descriptor in combination with the Random Forest classifier introduced in section 6.1.1. The motivation for using CNNs to predict the class of a document was presented in [5]. Section 6.1.2 describes the suggested CNN architecture of this project.

6.1.1 The HOG Descriptor and a Random Forest

It has been chosen to use the HOG descriptor compared to the more advanced and licensed SIFT descriptor presented by Lowe [42], as the descriptor is not required to provide features which are invariant to geometric transformations, because of the low variance of the structural document layout within the classes of interest.

The HOG descriptor has been implemented as a `R-function` in order to improve the understanding of the inner-working of this descriptor. The implementation is attached in section B.1. The HOG descriptor takes two parameters as input: *cells* and *bins* are determining the number of the uniform distributed cells. The number of histogram-bins are determining the resolution orientation of the calculated gradients. The number of bins and the number of cells are a design choice. A resolution of $bins = 8$ should be sufficient to capture the main tends of the gradients in each of the cells, since the basis structure of the documents, without the text, is existing of vertical and horizontal edges. The number of *cells* have been chosen to four in order to limit the number of total calculated features. This hyperparameter setting produces 128 features per page of the VRD.

The Random Forest as a classifier The chosen classifier for determining the class, for a given set of HOG descriptors, is the Random Forest algorithm. The implementation of the algorithm is developed by [16] and is available as a `R-package` called `randomForest`. The reason for this choice of classifier is because of the well-acknowlegded prediction power and few hyperparameters. The `mtry` and `ntree` are determining the number of random sampled candidate variables to include in its split criterion and the number of trees to grow in the forest respectively.

The optimal set of hyperparameters is achieved by repeated cross-validation (CV), which is further described in section 6.3. The chosen hyperparameter set achieves the highest average $\lfloor \kappa \rfloor$ metric and has the lowest standard deviation of this metric. The optimal hyperparameters found with grid-search are presented below:

- The optimal: `ntrees`=900, grid range: 500 to 2,500 by 200.
- The optimal: `mtry`=16, grid range: 10 to 20 by 1.

The grid-search of the hyperparameters creates 121 combinations. The explored grid-search metrics have been found by a 5-fold CV with 10 repeats. An overview of the highest average κ metrics of the combinations is presented in table C.1.

6.1.2 CNN Models

The suggested CNN architecture in [5] is composed of three blocks of convolution, a non-linear activation and a max-pooling operations similar to the block illustrated in

figure 5.14. Is it required to train this CNN architecture from scratch since [5] has not published their model parameters.

Due to the limited amount of samples for this project, it has been decided to find similar pre-trained CNN architectures and use transfer learning in order to limit the amount of trainable parameters.

Simonyan and Zisserman [41] presented their CNN architectures at the ILSVRC in 2014, which were quite different from top-performing architectures in the previous years. The core reason to apply their proposed CNN architecture to this project is because of their design of many small receptive fields (the size of the kernel of the convolution operations) throughout the CNN blocks of the network. It is possible to obtain a 7×7 receptive field by stacking three small 3×3 receptive fields with a stride of 1 and without spatial max-pooling in between. This introduces three non-linear activation layers instead of one in the same receptive field, which highly improves the desired discriminative power of architecture.

The model parameters trained on the ImageNet dataset are publicly available through the `Keras` framework. The top layers of the below mentioned CNN models have been trained with the `Adam optimizer` with the following non-default parameters: $lr = 5e^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Table 6.1 presents the layers of the chosen architecture.

The idea of transfer learning has been applied and the layers of the feature descriptor are identical to the layers proposed by [41]. The parameters from these layers has been frozen during training in order to preserve their learned knowledge. This reduces the amount of trainable parameters from 16,3M to 1,6M of the proposed CNN model.

The layers of the classifier have been designed for this project specifically. The classifier can be interpreted as 64 independent non-linear models, each of which are activated by a ReLU function to achieve non-linearity. These outputs are appropriately weighted and ReLU activated before the class-wise probabilities can be derived by the softmax function (equation 5.18).

The optimized CNN model The suggested optimized version of the initial CNN model has been achieved during evaluation of the learning curves and the error analysis after the first modelling iteration.

The goal of the changes from the initial to the optimized CNN architecture was to add dropout layers between layer 20 and layer 21, layer 21 and layer 22. This optimized architecture is mentioned as the optimized CNN model.

	Layer	Desc.	Output Dim.	No. param.
Feature Descriptor	1	InputLayer	224x224x3	0
	2	Conv2D	224x224x64	1,792
	3	Conv2D	224x224x64	36,928
	4	MaxPooling2D	112x112x64	0
	5	Conv2D	112x112x128	73,856
	6	Conv2D	112x112x128	147,584
	7	MaxPooling2D	56x56x128	0
	8	Conv2D	56x56x256	295,168
	9	Conv2D	56x56x256	590,080
	10	Conv2D	56x56x256	590,080
	11	MaxPooling2D	28x28x256	0
	12	Conv2D	28x28x512	1,180,160
	13	Conv2D	28x28x512	2,359,808
	14	Conv2D	28x28x512	2,359,808
	15	MaxPooling2D	14x14x512	0
	16	Conv2D	14x14x512	2,359,808
	17	Conv2D	14x14x512	2,359,808
	18	Conv2D	14x14x512	2,359,808
	19	MaxPooling2D	7x7x512	0
Classifier	20	Flatten	25088	0
	21	FullyConnected	64	1,605,696
	22	Softmax	4	260

Table 6.1: This table presents the initial CNN architecture used in this project. The CNN architecture is composed of feature descriptor layers and of classifier layers. The parameters in the descriptor layers are kept frozen during the training phases which decrease the number of trainable parameters by a factor of 10. The outputs from the Conv2D layers are activated by the ReLU function (equation 5.17). The kernel of the MaxPooling2D layers is 2×2 which halves the spatial dimension.

6.2 Second Stage Models

The task of the second stage module is to predict the VIN and LP sequences within their respective text fields. The resolution of these text fields has been fixed to a dimension of $512 \times 256 \times 1$ pixels illustrating a gray-scale image. This input dimension has been applied for the models in the second stage module.

The Tesseract OCR Engine proposed as the baseline model is briefly described in section 6.2.1. Section 6.2.2 introduces the suggested convolutional recurrent neural network (CRNN) model architecture for this project inspired by the presented architecture in [7].

6.2.1 The Tesseract OCR Engine

The Tesseract OCR Engine is used as the baseline model for this project. The Tesseract framework achieved state of the art results when presented at the The Fourth Annual Test of OCR Accuracy [43]. The framework holds a documented performance of 60% for automatic number plate recognition [44] which is conceptually similar to the prediction of text sequences in this thesis.

The Tesseract framework is acknowledged as a black-box. This report is referring to [6], [45] for further details of the inner-workings of the Tesseract OCR Engine.

The performance metrics of the baseline model have been achieved based upon version 3.04.00 combined with the highlevel `tesseract R-package`. The options for the OCR engine have been set to accept only characters from the defined vocabulary. The performance metrics of the OCR engine are reported in table 7.7.

6.2.2 CRNN Models

This section introduces the architecture of the suggested CRNN model. According to [7] a CRNN can be used to perform text recognition in images. Figure 6.1 illustrates the conceptual overview of the CRNN structure. This structure is composed of three parts; a CNN block applied as a feature descriptor, a RNN block used to learn the latent structures of the VIN and LP sequences and a classifier block, which outputs a softmax activated 2D tensor. The softmax tensor is containing the per time step probabilities for each of the characters in the vocabulary plus an extra ”blank” token.

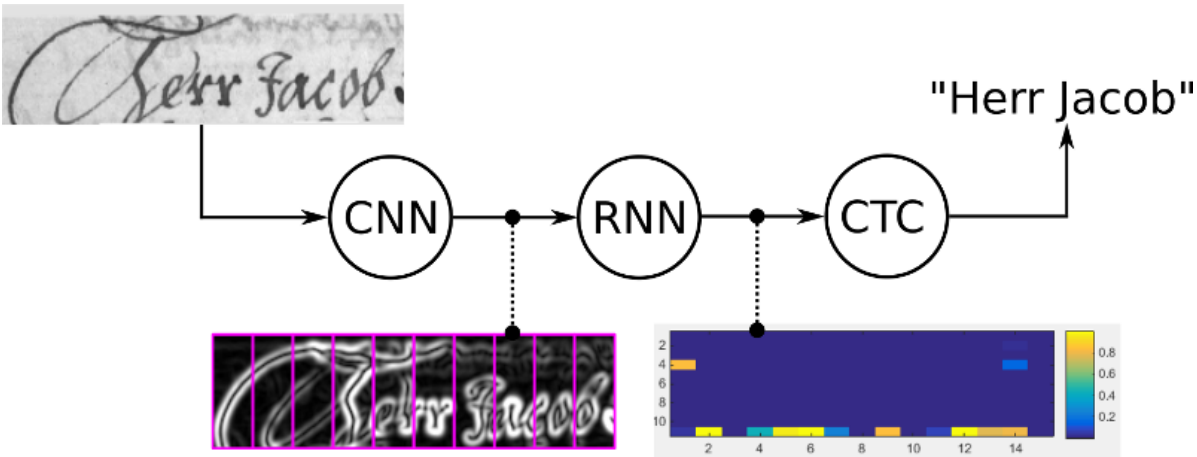


Figure 6.1: A conceptual overview of a CRNN model for the recognition of handwriting. This illustration is taken from [46].

Table 6.2 presents the layers of the chosen CRNN architecture for this project. The architecture deviates from the architecture which is proposed in [7]. A end-to-end implementation of a similar CRNN architecture with the CTC loss is presented by [47]. It

has been chosen to follow their architecture due to their reported ED metric of a similar text classification task, the possibility of applying transfer learning to the parameters of the CNN block and because by following their implementation the amount of possible implementation errors are limited.

	Layer	Desc.	Output Dim.	No. param.
CNN	1	InputLayer	512x256x1	0
	2	Conv2D	512x256x16	160
	3	MaxPooling2D	256x128x16	0
	4	Conv2D	256x128x16	2,320
	5	MaxPooling2D	128x64x16	0
RNN	6	Reshape	128x1024	0
	7	FullyConnected	128x32	32,800
	8	GRU, GRU	128x256, 128x256	221,952, 221,952
	9	AddingLayer	128x256	0
	10	GRU, GRU	128x256, 128x256	393,984, 393,984
	11	ConcatenateLayer	128x512	0
Cl.	12	FullyConnected	128x38	19,494
	13	Softmax	128x38	0

Table 6.2: This table presents the initial CRNN architecture used in this project. There are two GRU units presented in layer 8 and in layer 10 and they are placed in parallel. The final classifier block (Cl.) illustrates the layers of the classifier and is producing the softmax activated 2D tensor.

The initial CRNN architecture is composed of 1,3M model parameters which all are trainable. The Conv2D layers use 16 kernels with a size of 3×3. The kernel is using a padding and a stride of 1. This preserves the spatial dimension of its input layer. The MaxPooling2D layers use a kernel size of 2×2 which halves the spatial dimension of its input layer.

It has been possible to apply transfer learning for the CNN block of the suggested CRNN model since this block is identical to [47] and the model parameters are publicly available. The model has been trained for two iterations with two architectures for 50 and 25 epochs respectively.

The `Adam optimizer` has been applied to train the CRNN model with a $lr = 1e^{-3}$, $\beta_1 = 0.9$ and $beta_2 = 0.999$ for the first iteration and a decreased $lr = 5e^{-4}$ for the second iteration. The reason for applying a higher initial learning rate is due to small dissimilarities between the data foundation used in this project and in [47].

The optimized CRNN model The suggested changes to the initial model architecture were to add dropout layers between layer 11 and layer 12 in order to regularize the model. The suggestion is based upon the achieved knowledge of the learning curves

and of the error analysis produced by the initial CRNN model. However, the preferable choice would be to increase the quality and amount of training samples.

The new CRNN model architecture is from this stage forward referred to as the optimized CRNN model.

6.3 Model Training

The main objective in the model training is to optimize models with an ability to generalize to an independent validation set. The data samples are normally divided into three partitions; a training set, a validation set and a test set, which are described in further details in this section.

There are two general approaches for training models; an explicit approach and an implicit approach, both of which have been applied during this project.

The explicit approach The characteristics of the explicit approach are to train parametric models over a fixed well-known parameter space. When the parameter space is low it is computational efficient to apply the cross-validation (CV) technique. The CV technique uses three data partitions in order to provide an estimate of the generalization performance of the model. The estimates of the generalization performances are based upon several k-folds over the training data, see figure 6.2. k-fold is used to derive a more accurate estimate of the generalization performance.

Figure 6.2 provides an illustration of the CV technique with a 5-fold structure where the number of repeats are one. The idea of CV is to partition the training set into new training sets and validation sets. The shuffled partitions can be created by sampling the indices to obtain a structure similar to the pattern in figure 6.2.

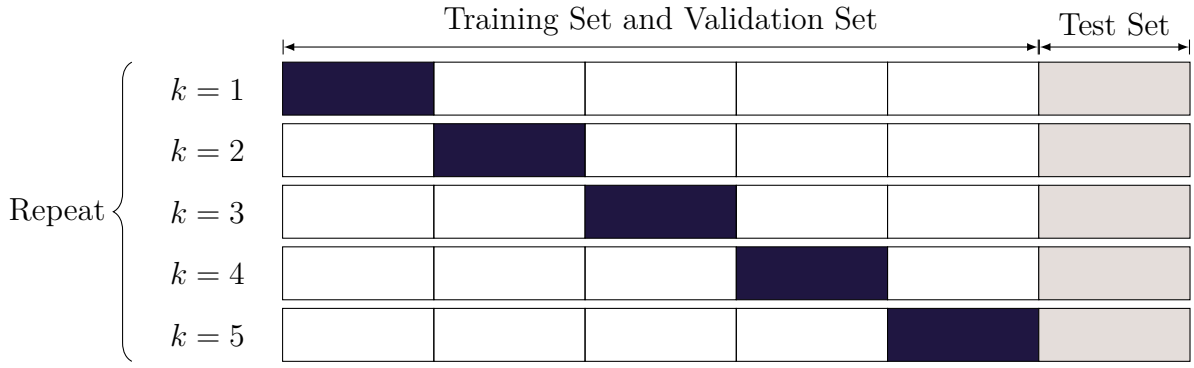


Figure 6.2: This figure illustrates the process of CV technique. The blocked dark boxes within the training set and validation set illustrate the validation partition for the given fold. The test set remains independent throughout the CV process.

The implicit approach The characteristics of the implicit approach applies to learning a latent non-designable parameter space. ANNs are examples of models which exist of these latent structures with millions of parameters. Applying the CV technique and closed form optimization is not computational feasible.

An evaluation of the model for all training samples is often not physically possible due to memory restrictions of the hardware. There are three possible learning procedures for ANN models:

- Online learning: propagating a single training pair through the network.
- Mini-batch learning: propagating a subset of training pairs through the network.
- Full-batch learning: propagating the entire set of training pairs through the network.

Common for all three approaches is to compute the gradients based upon the loss of forwarded training samples and apply the back-propagation method in order to update the parameters according to the approximated landscape of gradients for the given training samples.

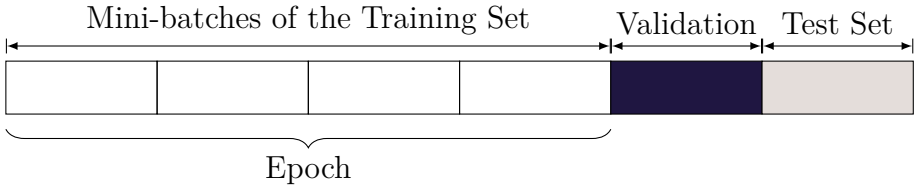


Figure 6.3: This figure illustrates the process of learning the parameters in an ANN. The blocked validation box and the test box remain independent throughout the learning process.

Model training In order to clarify the two model training approaches used in this project, the CV technique with repeats (figure 6.2) is applied to find the optimal parameters for the baseline model in the first stage. The secondary CNN and CRNN models in the first stage and in the second stage are using the mentioned learning procedure in figure 6.3.

A stratified down-sampling approach has been applied during training of the first stage models. It has been chosen to apply down-sampling in order to overcome the imbalanced prevalence of the classes and hereby not introduce an effective prior (bias) to the models.

6.3.1 Training Set, Validation Set and Test Set

The model training and model assessments will be performed on three different partitions of samples in the data set:

- A training set: The samples used for model training.
- A validation set (also called a development set in the deep learning community or a hold-out set in the CV framework): The samples which are unseen in the model training are used for making decisions regarding the parameters of the model; used for a feature selection and for the error analysis.
- A test set: Samples which are completely unseen during the training procedure. Hereby it is possible to get an independent estimation of the model performance.

The above described terminology will be used throughout this thesis.

The samples will be partitioned into; a training set, a validation set and a test set. The samples assigned to each of the partitions remain identical throughout the modelling of the first and second stage models. Table 6.3 reports an overview of the number of samples within each set for both the first stage models and for the second stage models.

	First stage	Second stage	Pct. (%)
Training Set	10,294	3,892	64
Validation Set	2,571	972	16
Test Set	3,223	1,216	20

Table 6.3: This table reports the number of samples in each of the data partitions. There are a total of 16,088 sample images for the first stage models and a total of 6,080 sample images (with text sequences) for the second stage models. The indices of the training samples, validation samples and test samples are identical for both the first stage and the second stage models. The percentage size of the training set and the validation set is 80% and the size of the test set is 20% of the total number of samples.

An important property of the data partitions is the requirement to contain the same relative class distributions. This will provide fairness among the data partitions and create the foundation of the fair evaluation [3].

6.3.2 A Single-number Metric

It has been chosen to use a single-number evaluation metric during the model iteration phase. The single-number metric will be used for each of the models in the first stage. This allows a fair comparison between different versions of each of the models. The strategy of using a single-number metric accelerate the ability to evaluate, distinguish and rank the possible models, hereby being able to choose the best performing model according to the selected single-number evaluation metric. The single-number evaluation metric is recommended by [3, chap. 8].

There will be used two different metrics for the first stage models and second stage models.

First stage metric The accuracy metric is not a suitable metric due to its properties. This metric performs badly when the class distribution is imbalanced (table 2.3). It is possible to use a weighted accuracy metric or to apply the Cohen's Kappa (κ) which is a coefficient accounting for the imbalance in the class distribution.

The Cohen's Kappa, onward mentioned as Kappa or κ, provides a ratio between 0 and 1. However this coefficient can be negative if the performance of the classification is worse than random. The κ metric is useful for comparison of the performance of multiple classes when an imbalanced class distribution occur [48]. Equation 6.1 denotes the formula of the κ metric.

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{6.1}$$

where p_o is the observed accuracy and p_e is the probability for a random classification.

The imbalanced class distribution will be ignored and the κ metric is therefore a suitable evaluation metric for assessments across the models. However, it should be mentioned that κ is threshold independent. This means it is not possible to change the cut-off values and hereby optimize the model for the business roles within the model iteration phase.

Table 6.4 provides a heuristic for the strength of the model compared to random. The heuristics have been denoted in the work by [49], [50].

Strength of the Model	κ (%)
Worse than random	< 0
Poor	0 to 20
Fair	21 to 40
Moderate	41 to 60
Good	61 to 80
Very Good	81 to 100

Table 6.4: This table provides a heuristic for the strength of the model compared to random. The heuristic does not provide a misclassification rate but can be used to evaluate the performance of the model compared to random guessing.

The κ metric will provide a per class metric and these metrics need to be combined to a single-number metric. The macro-averaging method and the micro-averaging method will produce two slightly different numbers.

- The macro-averaging method computes a κ metric independent for each class and then average these metrics. This provides an equally weighting of the per class metrics.

- The micro-averaging method computes a single κ metric on the aggregated contributions for each class.

The single-number metric is computed by the micro-averaging method due to its robustness for an imbalanced class distribution.

Second stage metric There will be used a different metric for model development and a difference metric for the final model assessment in the second stage module. The development metric presented in equation 6.2 and used in [24] is used to measure the rate of transcription mistakes. This metric is useful for tracking the improvement of the predictor during training and it provides a measure of the difference between two sequences of characters. However, the final model assessment needs to reflect a rate of true or false predictions.

The label error rate (LER) is the normalized Levenshtein distance (LD), also known as the edit distance (ED). The edit distance measures the distance between the two sequences s and t.

$$\mathrm{LER} = \frac{1}{N} \sum_{k} \mathrm{ED}_{(s_k, t_k)} \tag{6.2}$$

where N is the number of samples, k is a single sample and ED is the edit distance presented in equation 6.3. ED entails a recurrence which can be solved by a dynamic programming approach [51, sec. 2.1] and [52].

$$\mathrm{ED}_{s,t}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min = \begin{cases} \mathrm{ED}_{s,t}(i-1,j) + 1 \\ \mathrm{ED}_{s,t}(i,j-1) + 1 \\ \mathrm{ED}_{s,t}(i-1,j-1) + \begin{cases} 0 & \text{if } s_i = t_j \\ 1 & \text{otherwise} \end{cases} \end{cases} & \text{otherwise} \end{cases} \tag{6.3}$$

where i and j are single characters in s and t respectively. The edit distance exist of three inner elements which corresponds to a deletion step, an inserting step and a matching or mismatching step.

It is important to notice, that the LER does not reflect a metric w.r.t. the numbers of characters within the truth label. It is hereby not impossible to compare the metrics between the LP predictions and the VIN predictions.

CHAPTER 7

Evaluation of the Models

This chapter presents the achieved performance metrics for the developed first stage and second stage models. Through this chapter, various metrics will be applied to evaluate the models. Each section will introduce the metric of interest.

The first two sections are dealing with the developed first stage models and the developed second stage models. The structure of these sections are similar and both sections starts with a presentation of their baseline results. An illustration of the learning curves from the iterative training phase are presented and the findings from the error analysis used to optimize the next iteration of the models are shown. The main objective of the error analysis is to gain insight of the residual patterns, [3, sec. 14-17]. The performance metrics are eventually reported.

The data samples used for model development and model evaluating have been pre-processed according to earlier mentioned procedures in chapter 4. The data partitions defined in section 6.3.1 are kept identical during training of the first stage models and the second stage models.

For the second stage models, the partitions have been subsetted in order to contain only the DEs class and the samples have then been flatten, since each sample requires two predictions.

The final section will introduce the expected end-to-end performance of the technical pipeline.

7.1 The First Stage Models

This section is reporting the performances of the first stage models. HOG Descriptor and Random Forest classifier is used as the baseline results and the development process of the CNN models are described in section 6.1.1 and in section 6.1.2 respectively.

The evaluation metric for these models is the κ metric defined in equation 6.1. This metric provides a per-class measure which is combined to a single-number metric measure by the micro-averaging method. The single-number κ metric is used in the model development process of the first stage models due to its property of robustness for an imbalanced class distribution.

The target labels of all single paged VRD images have been manually inspected and corrected in order to obtain the best possible quality. The initial inspection phase has been double checked by trained model inspection. The initial CNN model (section 6.1.2) have been trained and its errors have been analysed in order to reduce the number of manual mislabelled targets. All performance metrics in the tables in section 7.1 are based upon the second phase model inspected target labels. However, it is notable that there are spotted mislabelled targets values in a later phase of this project, which are changed prior to the next model iteration. This can possibly reduce the number of misclassifications of the baseline model.

7.1.1 HOG Descriptor and Random Forest

The HOG descriptor and Random Forest classifier are described in section 6.1.1 and used as the baseline reference. Table 7.1 is reporting the confusion matrix of the test set and table 7.2 is reporting the single-number κ performance metrics for each of the data partitions.

		Predicted				Per-class
		DE	DEs	SE	OUT	κ (%)
Class	DE	1,775	8	4	169	84.4091
	DEs	4	601	0	3	98.0602
	SE	1	0	192	4	95.7237
	OUT	25	7	5	425	86.5066

Table 7.1: This table presents the confusion matrix and the per-class κ metrics for the test set. The classifier holds a very good performance according to the κ heuristics compared to random (table 6.4).

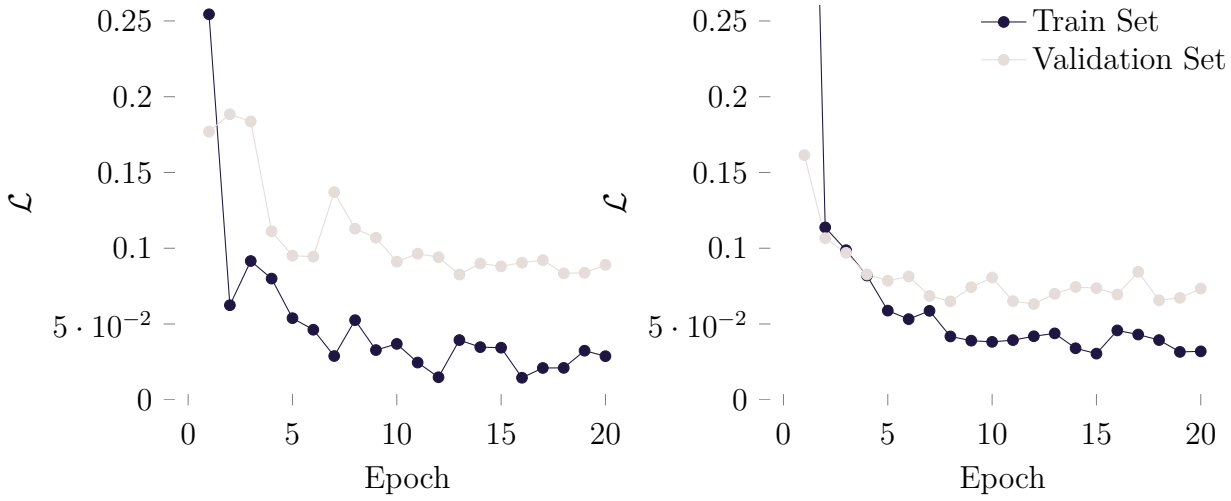
	κ (%)	Error Rate (%)
Train Set Validation Set	91.9564	4.7726
Test Set	87.9765	7.1362

Table 7.2: This table reports the performance metrics of the training set, the validation set and test set. The Random Forest classifier has been re-trained on the samples from the training set and the validation set. The presented metrics are based upon the optimized CV hyperparameters for the Random Forest classifier.

The derived confusion matrix from the test set (table 7.1) reports a materially worse ability of classification power for the DE class and for the OUT class. By combining this with the differences between the two single-number κ metrics at $\approx 4\%$, it is possible to conclude that the baseline model has a lack of complexity and it does not generalize to the test set.

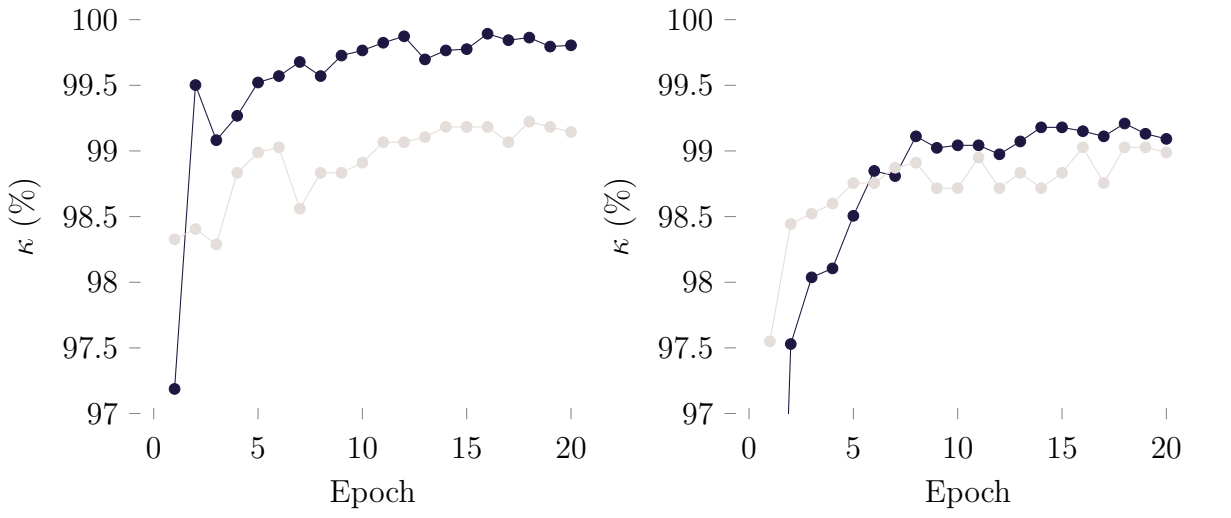
7.1.2 CNN Models

The pattern of the initial CNN model (section 6.1.2) is illustrated by the learning curves in figure 7.1a and figure 7.1c. The gap between the learning curves of the train set and validation set (figure 7.1c) is characterized as a model with low bias and high variance, since its performance is failing to generalize to the validation set.



(a) First iteration.

(b) Second iteration.



(c) First iteration.

(d) Second iteration.

Figure 7.1: This figure visualizes the learning curves of the cross-entropy loss ($\mathcal{L}$) and of the single-number κ metric for each of the training epochs. (a) and (c), (b) and (d) correspond to the first and second training iteration respectively. The legends in the upper right corner are applicable to all four subfigures.

As the model has low bias, according to the observations from figure 7.1c, it is assumed

that the complexity of the model is sufficient. It is therefore suggested to regularize the CNN model in order to reduce the variance and improve the ability of generalization to the validation set.

The manual inspections of the misclassifications for the two CNN models are reported in table 7.3. It is possible to differentiate the behaviour between the two models by considering the number of misclassification of the "An old German VRD" category of the error descriptions. This number is increased for the optimized CNN model which means that suggested regularizing is limiting the amount of overfitting and improving its ability of generalization to the validation set.

Error Desc.	N	Pct. (%)
Wrong prediction	10	45
An old German VRD	7	32
Wrong label from Captia	4	18
Wrong label from annotation	1	5

(a) First iteration.

Error Desc.	N	Pct. (%)
An old German VRD	13	50
Wrong prediction	5	19
Wrong label from annotation	4	15
Wrong label from Captia	4	15

(b) Second iteration.

Table 7.3: These tables report the complete statistics of error predictions of the validation set. N is referring to the number of samples for a given error description and the third column is reflecting the percentage distribution. The predictions with the error description "Wrong label from Captia" or "Wrong label from annotation" has been mislabelled and these 13 wrong predicted samples, in (a) and (b), have actually been predicted correctly.

Table 7.4a and table 7.4b are reporting the confusion matrices of the initial CNN model and of the optimized CNN model respectively.

The suggested regularization indicate an improvement of the model's ability to generalize in figure 7.1. However, there is an increase of the number of misclassifications between table 7.4a and table 7.4b.

The 13 misclassifications, with the error description "An old German VRD" in table 7.3b, constitutes with a large proportion of the predicted OUT class in table 7.4b. The decreased amount of misclassifications in this table is referring to the effect of the regularization.

The effect of regularization can be measured by the decrease in variance between the performance metrics for the train set and the validation set. Table 7.5 reports the metrics of the two models and variance between the κ metrics is decreased by $\approx 6\%$.

		Predicted				Per-class
		DE	DEs	SE	OUT	κ (%)
Class	DE	1,546	0	1	12	98.5447
	DEs	0	486	0	0	100.0000
	SE	0	0	157	0	100.0000
	OUT	9	0	0	360	95.7434

(a) First iteration.

		Predicted				Per-class
		DE	DEs	SE	OUT	κ (%)
Class	DE	1,536	1	1	21	97.4381
	DEs	0	486	0	0	100.0000
	SE	0	0	157	0	100.0000
	OUT	3	0	0	366	98.5882

(b) Second iteration.

Table 7.4: These tables report the confusion matrices and the per-class κ metrics for the validation set. The overall performances for the two classifiers are very good according to the κ heuristics compared to random.

	κ (%)	Error Rate (%)
Train Set	99.7117	0.1651
Validation Set	98.5066	0.8557
Test Set	98.4293	0.8998

(a) First iteration.

	κ (%)	Error Rate (%)
Train Set	99.3738	0.3594
Validation Set	98.2439	1.0113
Test Set	98.0619	1.1170

(b) Second iteration.

Table 7.5: These tables report the performance metrics for each of the data partitions for the first (a) and the second (b) training iteration. The Error Rates are reflecting the percentage number of wrongly classified samples.

Key points Table 7.6 reports the confusion matrix for the test set and for the optimized CNN model.

		Predicted				Per-class
		DE	DEs	SE	OUT	κ (%)
Class	DE	1,923	0	3	30	97.0726
	DEs	0	608	0	0	100.0000
	SE	0	0	196	1	99.1192
	OUT	2	0	0	460	99.2489

Table 7.6: This table reports the confusion matrix and the per-class κ metrics for the test set. The overall performances of the final CNN classifier is very good according to the κ heuristics compared to random.

It can be concluded that the optimized CNN model is better than its initial version and it out-performs the baseline model. This conclusion is based upon comparing the per-class κ metrics between table 7.1 and table 7.4, and by comparing the single κ metrics between table 7.2 and table 7.5. The optimized CNN model is therefore the recommended choice for the first stage model.

7.2 The Second Stage Models

This section is reporting the performances of the second stage models. The Tesseract OCR engine is used as the baseline result and the implementation of the CRNN models are described in section 6.2.1 and in section 6.2.2 respectively.

The evaluation metric for these models is the LER metric defined in equation 6.2. This metric gives an average number of wrongly predicted characters per sample for each of the three data partitions. The table 7.7 to table 7.9 includes the LER metric and a percentage error rate. The LER metric will be used in the model development and in the evaluation process, as this number is more informative when measuring the development progress because each sample is constituted of multiple characters.

Notice It is hereby given that the field of the LP and the field of the VIN will be treated equally. It may influence the interpretability of the LER performance metrics.

The tables presented in table 7.7 to table 7.11 are reflecting the performances after manually checking (and updating) each of the fields of interest for each sample. During the manual inspection of the first iteration error analysis, a mislabelling rate of $\approx 10\%$ between the image field and the target were discovered. Further $\approx 19\%$ of the samples were visually not identifiable and $\approx 14\%$ contains issues with the crop of the text fields. These targets have been changed to ”blank”.

7.2.1 The Tesseract OCR Engine

Table 7.7 reports the performance metrics of for each of the data partitions.

	LER	Error Rate (%)
Train Set	7.9561	87.4358
Validation Set	8.0761	88.0658
Test Set	7.9095	87.0066

Table 7.7: This table reports two performance metrics for each of the data partitions. The metrics are similar across the data partitions, which can be interpret as a quality of the samples as being equally distributed. The Error Rates are reflecting the percentage number of wrongly classified samples.

It is possible to observe that these metrics are equal across the three data sets which confirms an equal quality of the samples within each of the sets. The average LER metric for each of the three partitions are ≈ 8 characters. No further reflections are carried out since the LER metric for the validation set is used as a reference for the development process of the CRNN models.

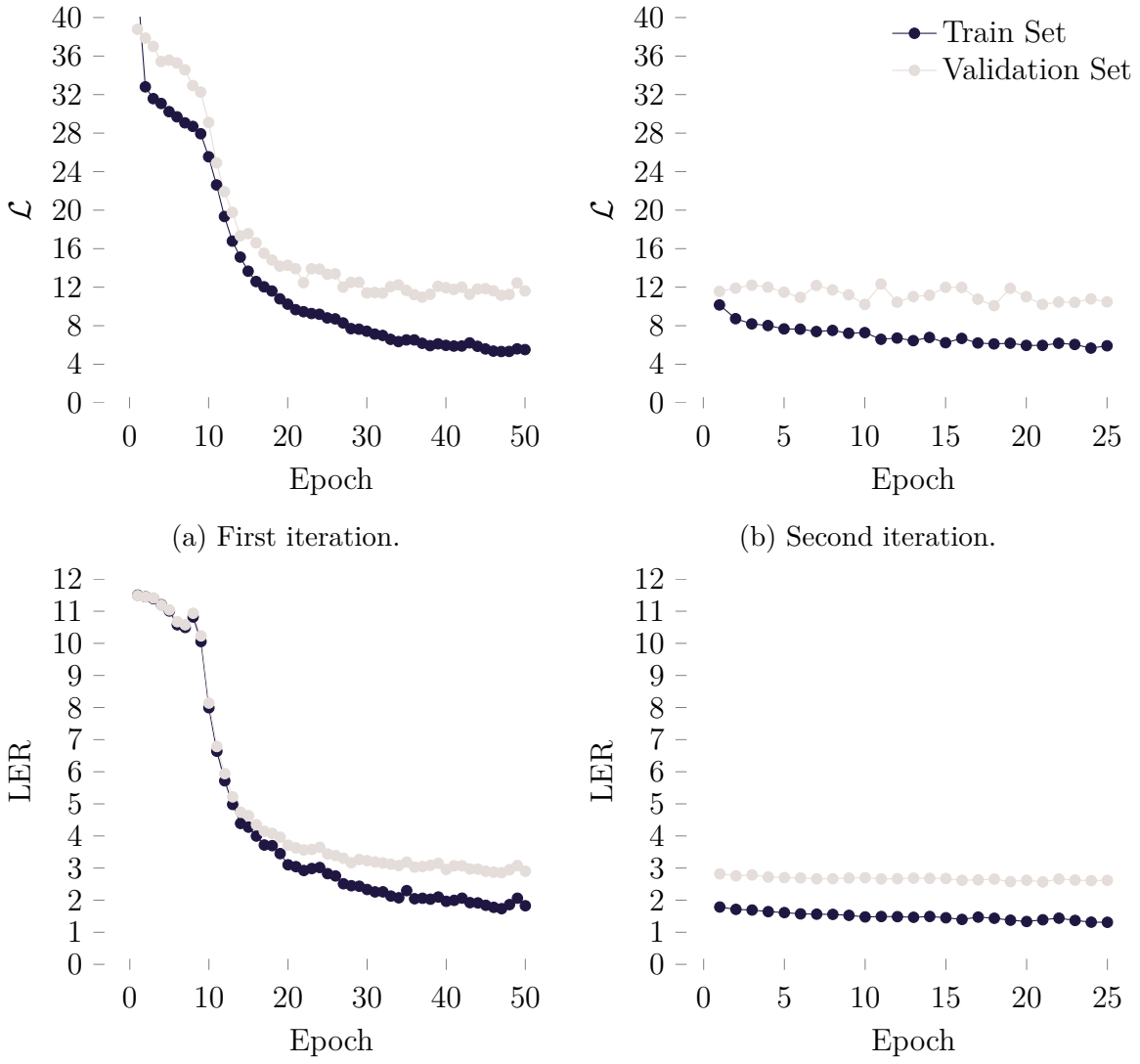
7.2.2 CRNN Models

As for the CNN model, the learning curves illustrated in figure 7.2a and figure 7.2c, provide an interpretable insight of the performance of the model during training phase. From epoch 20 in the first iteration it is possible to spot an increase in the gab between the train set and the validation set for both its loss curves and its performance metrics. At epoch 50 it is possible to conclude that the model has a bias with an average of LER ≈ 2 and a variance of LER ≈ 1. The model has memorized the samples in the training set to such degree that it makes its ability to generalize to the validation set poor.

The proper course of action for penalizing the model is either to add/create more high quality data or to apply dropout layers which will regularize the model. The first option is not prioritized due to the strict time limit of this project but is the preferable cause of action due to the limited amount of training samples. The optimized CRNN model with dropout layers is described in section 6.2.2.

The second training iteration has been executed for 25 epochs. The validation curves in figure 7.2b and in figure 7.2d denote a decreasing trend meaning the model is still learning and better performance metrics may be obtained. Due to the time limit it has been chosen not to continue for further epochs, since the major improvement of the CRNN model is restricted by the quality of the data.

Table 7.8 reports the findings of the error analysis after the first and second iteration. It is important to notice, that table 7.8a and table 7.8b are not directly comparable, since these are based upon 100 random sampled mispredictions. It has been chosen to group



(a) First iteration.

(b) Second iteration.

(c) First iteration.

(d) Second iteration.

Figure 7.2: This figure visualizes the learning curves of the CTC loss ($\mathcal{L}$) for the CRNN models and their respective LER metrics for each of the training epochs. (a) and (c), (b) and (d) correspond to the first and second training iteration respectively. It is notable to mention, that the loss curve of the validation set in the second iteration (b) is still decreasing after its training phase has ended. This means the learning of the model is not yet converged and its performance can be increased. The legends in the upper right corner are applicable to all four subfigures.

the findings of the error analysis into the LP text field and the VIN text field in table 7.8, since these fields have different phenomenons.

Text Field	Error Desc.	N	Pct. (%)
LP	Wrong prediction	42	35
LP	Visual impossible	10	8
LP	Bad crop	3	2
VIN	Visual impossible	25	21
VIN	Wrong prediction	21	17
VIN	Bad crop	20	17

(a) First iteration.

Text Field	Error Desc.	N	Pct. (%)
LP	Wrong prediction	49	40
LP	Visual impossible	10	8
LP	Bad crop	3	2
VIN	Visual impossible	22	18
VIN	Bad crop	19	16
VIN	Wrong prediction	19	16

(b) Second iteration.

Table 7.8: These tables report the statistics of 100 random sampled error predictions within the validation set. N is referring to the numbers of samples for a given error description and fourth column is reflecting the percentage distribution. The error analysis in (a) and (b) can therefore not be compared directly. A single sample may contain one or multiple error descriptions, which is why N does not add to the 100 random samples in (a) and (b). The percentage column describes the distribution across the unfolded error descriptions.

The most prominent finding after the first training iteration is in regard to the quality of the cropped text fields. By aggregating the ”Wrong prediction” category in table 7.8a, it is possible to conclude that 48% of the mispredicted samples are caused by poor data quality.

The root cause of the poor data quality of the cropped text fields is a combination of the variating quality of the input VRDs and the defined pre-processing steps defined in the technical pipeline. The proper approach for solving this issue is to re-engineer the pre-processing steps of the pipeline. The re-engineering is requiring a new iteration, which the time limit does not accommodate. The suggested improvement to the second training iteration is therefore achieved by regularizing the CRNN model and possibly decrease a proportion of the 52% misclassified samples.

A general pattern for misclassified characters is observed for the following characters: 6 and 9, N and M, W and V, I and T, E and F.

Another important observation between the ”Wrong predictions” category of the LP and of the VIN in table 7.8a is their respective misclassification rate. An explanation could

be the different scales of the LP text due to the length of the text. This will be further discussed in chapter 8.

A prevailing experience from the second iteration of the error analysis is the mispredictions of only one character. A majority of the observed single character mispredictions are constituted by visually similar characters such as; W and V, O and Q, E and F, I and L, S and 5, M and W, Z and 2.

It possible to get an insight of the poor quality of the pre-processed text fields in the presented figures in appendix D.

Table 7.9 is reporting the two performance tables for the first and the second training iteration. According to the LER metric, the optimized model has increased its variance by 21% but decreased its bias by 28%. This indicates a tendency of overfitting in the training set and hereby a lack of generalization. However, the error rate of the training set has improved by 27%. Both the LER metric and the error rate of the validation set are improved for the optimized CRNN model, see the difference between table 7.9a and table 7.9b.

	LER	Error Rate (%)
Train Set	1.8255	63.8489
Validation Set	2.9012	80.0412
Test Set	3.0058	78.6184

(a) First iteration.

	LER	Error Rate (%)
Train Set	1.3104	46.5057
Validation Set	2.6173	72.8395
Test Set	2.6571	74.2599

(b) Second iteration.

Table 7.9: These tables report the performance metrics for each of the data partitions for the first (a) and the second (b) training iteration. The Error Rates are reflecting the percentage number of wrongly classified samples.

The overall performance of the optimized CRNN model is better compared to its baseline model and the initial CRNN model. However, the performance of the model may be enhanced as pinpointed by its learnings curves in figure 7.2d. Other possible improvements are discussed in chapter 8.

7.3 The End-to-End Performance

The performance of the technical pipeline is achieved by using the best performing first stage model and second stage model. The overall performance accuracy of the test set is summarized in table 7.10.

Predicted	N	Rate (%)
Correct	2,605	80.8253
Wrong	618	19.1747

Table 7.10: This table reports the end-to-end performance of the technical pipeline on the test set. All three predictions have to be accurate per sample in order to obtain a correct sample. The Rate is reporting the percentage distribution for the correct and wrong predictions.

Table 7.11 combines the statistics of three predictions per sample; one from the first stage and two from the second stage. If the first stage predicts the DEs class, then the second stage will predict the LP sequence and the VIN sequence from the VRD. The table has converted the `NA` output from technical pipeline to "-" for other classes than DEs for both the LP sequence and the VIN sequence. The table has been divided into two parts where the first four rows are representing the performances for both the first stage and the second stage. The last two rows are representing the performance of the first stage predictions.

Class	LP	VIN	Normal (%)	Grouped (%)
Correct	Correct	Correct	0.8067	4.2763
Correct	Wrong	Correct	5.8331	30.9211
Correct	Correct	Wrong	1.5824	8.3882
Correct	Wrong	Wrong	10.6423	56.4145
Correct	-	-	80.0186	98.6233
Wrong	-	-	1.1170	1.3767

Table 7.11: This table reports the end-to-end performance of the technical pipeline. The table is divided into two parts: the first four rows deal with the predicted sample from both the first and the second stage. The last two rows are representing the predicted samples of the first stage, hence "-" for LP and VIN.
The reported percentage numbers in the fourth column and in the fifth column report the percentage distribution of the three predictions. The percentage numbers in the fourth column is calculated w.r.t. all samples in the test set. The percentage numbers in the fifth column is calculated w.r.t. the number of DEs samples and the number of samples for the remaining classes.

It has to be mentioned, that the first stage model does not misclassify any pages of the important samples of the DEs class, which means the second stage will be activated properly. It is notable that the performance of the second stage model is remarkably better for predicting VIN sequences compared to the LP sequences.

Key points The expected overall performance of the technical pipeline is reported in table 7.10. The table reports an expected prediction accuracy of 81% for the technical pipeline. The expected prediction accuracy of the first stage model is 99%, presented in table 7.5b. The expected prediction accuracy, when both LP and VIN are accurate per sample, for the second stage model is 4% (first row and fifth column in table 7.11).

CHAPTER 8

Discussion

This chapter introduces the discussion of this thesis. The achieved knowledge and reflections of the most important aspects of this project, together with an outlook on the reliability of the presented assumptions throughout this thesis will be discussed. In general, the established reflections deal with the achieved performance metrics presented in chapter 7. Most reflections are made in regard to the second stage module due to its considerably lower prediction performance for all of the three proposed models.

The reflections of the suggested visual explanation frameworks and an analysis of the assigned probabilities of the suggested models are discussed in order to access the applicability for production of this technical pipeline.

Other important issues related to deep ANN models such as adversarial noise amongst others are important issues but these are not investigated nor further discussed.

8.1 The Data Foundation

A perfect data foundation is hard to achieve and the data foundation for this project has been far from ideal. The number of training samples; the manual scanning of VRDs into PDFs and the wrong labelling from Captia are amongst the aspects of the data foundation which may have affected the outcome of the model training. The following paragraphs highlights three major difficulties with the data foundation.

A large data foundation is a vital part for model training of the ANN models. The process of data collection was time consuming and in order to be in progress with the project and eliminate possible delays, the data collection was set to expired at the 1st of October. It was only possible to access 19% of the total numbers of VRDs in Captia at this time and by receiving the remaining numbers of VRDs the pipeline, especially the CRNN model, could have been improved.

During the process of the EDA it was discovered that the META class pages of a VRD are duplicated and present in multiple VRDs (section 2.3.1.1). This causes a potential of having a single unique page represented in two or more of the data partitions, which leads to a depended validation set or a depended test set. The evaluation of the model is therefore biased. To eliminate this evaluation bias, the single pages assigned to the META class have been excluded which causes a reduction in the number of training

samples and a lack of capability to classify the META class pages. However, it is assumed that the META class pages within a VRD are classified as the OUT class. The accuracy of this assumption has not been further tested.

The reported LD metric of 2.6173 of the validation set in table 7.9b of the CRNN model is high compared to the reported metric of the similar CRNN architecture in [47]. The amount of training samples has to be appraised against the amount of trainable parameters in the CRNN model which may describe the reason for its performance. The learning curves presented in figure 7.2b and figure 7.2d may indicate the similar phenomenon of a complexity of the CRNN model which is too high compared to the number of training samples. The performance of the model does not generalize properly even though the CRNN architecture has been regularized to the second training iteration.

8.2 GDPR Compliance

The suggested pipeline is not yet GDPR compliant due to the lack of demonstrated interpretability of the CRNN model. However, it is possible to visually explain the discriminative locations of an image by applying the Grad-CAM method, which is contributing to the character prediction in each of the time steps. A similar visual explanation is performed in [37].

The visual explanations of the CNN model demonstrated in figure 5.19 and in figure 5.20 highlight areas which are unique for the DEs class. The derived explanations of the LIME framework use a chosen hyperparameter setting which includes five super-pixels in its derived sparse linear model. Since it is impossible to find the optimal hyperparameters which will generalize across the entire data foundation, it may be more rewarding to visualize the activated perceptrons in each of the layers during a forwardpass. Thereby it is possible to get an understanding of the different feature descriptors for different levels of abstraction.

Beyond this, it may gain extra insight to visualize the activations which are contradicting the prediction. This is done by multiplying the estimation of the gradient by minus one in equation 5.40 in the Grad-CAM method.

8.3 The Suggested Technical Pipeline for Production

To assess the quality of the suggested pipeline, the overall end-to-end accuracy rate is one method to estimate the performance in production. The end-to-end accuracy rate is 81% for the test set as reported in table 7.10.

Table 7.11 reports a detailed view of the performances of the three predictions. The percentage distribution for each of the models can be summarized into accuracy rates. The first stage model achieves 99% and the second stage model achieves 4% (both sequences of the LP and of the VIN have to be accurate per sample).

The accuracy rate is satisfying for the first stage model. The second stage model yields an accuracy rate which is not suitable to production. The error analysis of the initial CRNN model concluded that the root cause of the low prediction power was caused by the poor quality of the data. In 48% of the mispredicted samples the sentences were visually impossible to distinguish (figure D.1e) or the cropped text fields were offset during the second stage pre-processing step (figure D.1b).

8.3.1 Possible inappropriate design choices of the pre-processing modules

Bottlenecks within the technical pipeline It is important to limit the duration of the processing time in production in order to enhance the general efficiency. The duration[1] of processing the German VRD (figure 2.2) is 3:4 minutes and the duration of processing the Swedish VRD (figure A.1) is 3:11 minutes.

It is remarkable that the duration time is lower for the German VRD than for the Swedish VRD, as the German VRD involves both the first stage and the second stage model whereas the Swedish VRD only involves the first stage model. It can therefore be excluded, that there should be a difference when running the first stage model alone compared to running both models in succession. Hence is the time-consuming part present for both the German VRD and the Swedish VRD.

The durations of processing the VRDs are interfered as long API responses. It is therefore interesting to know where the bottlenecks are present in the end-to-end pipeline. The bottlenecks are profiled with the `profvis` `R-package` and 92% of the duration is within the pre-processing modules of the pipeline. The bullets below show the most time-consuming operations in the pre-processing modules:

- The general pre-processing function `image_preprocess()` is the most time-consuming part. The sub-process of deskewing each of the single pages is composed by several functions: a 25% downscale of the spatial dimension, a canny filter and a Hough Transform, which all are used to determine the rotation. The process is succeeded by deskewing the high-resolution single page image.

- The initial crop is analysed according to a binary gray-scale image and is the second most time-consuming part. The methods used for analyzing the initial

[1] Timed on a MacBook Pro (Retina, Mid 2012) - 2,7 GHz Intel Core i7.

crop of the binary image contribute with a prominent proportion in the general pre-processing function. This analysis relies on a median filter for noise reduction and the Otsu's Threshold method (section 5.2) which separates the foreground from the background of the image.

A remarkable reduction in the duration of processing a VRD can be achieved by analyzing the need for deskewing each image in the VRD. The initial reason for deskewing each of the images was to obtain a better crop, essential to the properties of the HOG descriptor. However, if the need for deskewing is not present, which it should not be since rotational transformation is normal in the data argumentation for CNN models, it is preferable to apply deskewing only for images processed by the second stage prior to the crop of the text fields. This will lead to a more computational efficient deskewing approach and hereby a faster end-to-end process of the VRDs.

Cropping of the text fields in the second stage The experience from the error analysis indicates a high proportion of wrongly cropped text fields. The core reason for this is due to the quality of the input data. The VRDs are manually scanned which introduce stochasticity in the input data. These perturbations are propagated and this may be increased during the two pre-processing steps, causing an offset of the pre-defined coordinates. The coordinates determining the cropping areas of the two text fields are derived by an analyzation of 25 random samples from the training set and may not generalize to the entire data foundation, hereby cropping incorrectly.

An alternative approach based upon the accumulated knowledge would be to crop a region around the field of interest, as proposed in the following steps:

1. Define two kernels, e.g. the Sobel kernels equation 5.8, detecting the vertical lines and the horizontal lines of this cropped region. This adds the images together to achieve rectangles and perform the analysis of rotation.
2. Deskew and crop a smaller region around the field of interest.
3. Locate all the rectangles, ideally just one, and extract the coordinates of the rectangle.

This approach would be more robust to perturbations of the scanned VRDs.

The effect of normalizing the characters of the sequences The experience from the second iteration of the CRNN error analysis leads to findings of a higher LD metric of the LP sequences compared to the VIN sequences where both text fields have been cropped perfectly without any of the surrounding edges. This metric is also causing a lower accuracy rate for the LP sequences compared to the VIN sequences (table 7.11).

This experience is conflicting with the assumption of considering the text fields identically during the pre-processing step. For two perfectly cropped text fields, the size of

the LP sequence will be larger compared to the VIN sequence which is bound by the horizontal dimension due to twice as many characters.

It would be interesting to measure the effect of the predictions, both the normalized size of the characters and the different sizes of the characters within the extracted sequences.

8.3.2 The Possibility of Multiple Text Decodings

A proportion of the misclassified text sequences experienced in the second iteration of the CRNN error analysis (table 7.8b) reveals a LD metric of one. This means a single character in the sequence is incorrect or missing in the prediction. Figure D.1a provides an illustration of a wrongly classified character in a VIN sequence. The third character is predicted as ”1” but is labelled as ”T”.

The assumed strategy is to decode the most likely path through the predicted two-dimensional softmax tensor (equation 5.25), however this has to be validated. It would be interesting to analyse the LD metric for each of the top three most likely sequences from the predicted softmax tensor. If this analysis documents an improved LD metric for one of the sequences, it may be considered to return all of these extra sequences to the end user. These likely sequences can be decoded by applying a CTC beam search method.

8.4 The Measure of Uncertainty

The two proposed models are composed of ANN architectures. Their latent parameter space have been optimized so its derived probability for a class is maximized when the particular class is given as input.

The ANN models as they are, do not provide a measure of uncertainty for a given prediction which raise the question of whether to trust their predictions. In order to gain trust of a prediction, when a visual explanation is not usable e.g. for automatic decision making processes, a measure of uncertainty along with its prediction must be provided.

One approach to achieve the measure of uncertainty would be to perform dropout operations during the forwardpass within the network and repeat this process several times. The variation of the assigned class-wise probabilities produced in each forwardpass can be used as the measure of uncertainty for a given prediction. This approach is suggested by Gal and Ghahramani [53]. The method of using dropout operations in the forwardpass to represent the model uncertainty is suggested to the future work. However, it is possible to gain an insight of the confident of a prediction by analyzing the assigned probabilities for correct and wrong predictions respectively.

Thresholding Table 8.1 reports two sub-tables which refer to the captured assigned probabilities of the predictions in the test set for the optimized CNN model and the optimized CRNN model respectively.

Predicted	Min	2.5%	$\hat{\mu}_B$	97.5%	Max
Correct	0.6256	0.9990	0.9993	0.9996	1
Wrong	0.4822	0.8852	0.8894	0.8936	1

(a) First stage model.

Predicted	Min	2.5%	$\hat{\mu}_B$	97.5%	Max
Correct	0.0056	0.2735	0.2786	0.2838	0.9399
Wrong	0	0.1109	0.1145	0.1182	0.7583

(b) Second stage model.

Table 8.1: This table provides an analysis of the assigned probabilities for each prediction performed on the test set. The min and max columns reports their respective lowest and highest assigned probabilities. $\hat{\mu}_B$, 2.5% and 97.5% present the approximated mean and percentiles of the bootstrap distribution. This distribution is approximated by 5000 samples.

The sub-tables in table 8.1 represent the approximated bootstrap distributions ($\hat{\mu}_B$, 2.5% and 97.5%) of the assigned probabilities for correct or wrong predictions for both of the suggested models respectively.

It is possible to derive a clear separation between the distribution for a correct prediction and of a wrong prediction when looking at table 8.1a.

However, there are 25 samples which are wrongly misclassified with an assigned probability above the 97.5% percentile (0.8936). These have been manually inspected and the majority of the 25 samples have wrong labels and are therefore classified correctly. This means the reported maximum assigned probabilities for a wrong prediction can not be trusted. It is possible to rectify these labels and hereby achieve a better insight for determining the threshold. This has not been prioritized due to limit of time.

The assigned probabilities of the second stage model are derived from the CTC decoding. It is possible to derive a clear separation between the correct and the wrong predictions of the second stage model, see table 8.1b, as of the first stage model.

The provided insight of these approximated bootstrap distributions can be used for determining two thresholds which ascertain "the trust" of the predictions. These thresholds should be determined by domain experts which are able to evaluate and quantify cost and trade-off between accepted misclassifications and manual processing. The defined thresholds can easily be included in the business logic module and hereby improve "the trust" to the end-to-end suggested pipeline.

CHAPTER 9

Conclusion

Throughout this thesis several models have been developed and their performances compared.

Machine learning is able to undertake the process of VRD validation with an end-to-end accuracy of 81% (table 7.10) for the recommend technical pipeline provided from [1].

This thesis presents a pipeline able to process the required information in order to automate the manual validation task of German vehicle registration documents.

Various methods and approaches have been studied during the literature review. Multiple approaches exist for solving challenges such as document classification and prediction of text sequences. Each of these have been used as an inspirational source to compose the modules of the pipeline presented in this thesis.

The proposed optimized CNN and CRNN architectures yield better performances for each of the two domains compared to their respective baseline models. The document classification model and the text prediction model achieves an accuracy of 99% (table 7.5b) and of 26% (table 7.9b) respectively on the test set. The text prediction model yields better prediction power for VIN sequences compared to LP sequences (table 7.11).

The performed error analysis pinpointed the root cause of the low performance of text extraction, which is the quality of the cropped text fields. The quality of $\approx$ 50% of the samples in the error analysis samples were either offset, which entails partly or completely wrong cropped areas, or inappropriately pre-processed, which entails the appearance of the sequence to be visually impossible to distinguish. However, for perfectly cropped fields the appearance of a LP sequence and a VIN sequence is deviating, which could explain the relatively higher experienced LD metric for the LP sequences. The performance cannot be enhanced with the chosen second stage pre-processing approach due to the data quality of the cropped text fields.

It is difficult to define the best ability of interpretation due to the very high complexity of the models. This thesis provides an example of model interpretation by two visual explanation frameworks. These interpretations are only demonstrated for the document classification model (CNN) in figure 5.19 and figure 5.20. However, it is likewise possible to apply the Grad-CAM method to the text extraction model (CRNN). Since the ability of interpretation is only demonstrated for the document classification model, the entire end-to-end technical pipeline is only partly GDPR compliant.

The recommended pipeline is presented in figure 4.1. Further descriptions of the pipeline and the current pre-processing steps are given in chapter 4. It is recommended to apply the optimized CNN model for document classification and the optimized CRNN model for text prediction. However, is it also recommended to determine and implement two appropriate thresholds for both of the models before using the pipeline in production (section 8.4).

CHAPTER 10
Future Perspectives

This chapter presents the recommended improvements for the suggested technical pipeline. The recommendations for future perspectives are prioritized w.r.t. the improvements which has the highest influence of the overall end-to-end accuracy pipeline.

A re-engineering of the pre-processing modules One of the key reasons for the low achieved performance metric of the CRNN model is related to the quality of its training data. The quality the cropped text fields may be achieved by re-engineering the pre-processing steps in the second stage. The new approach should analyse the relation between the sizes of the cropped sequences and its relative LD metric.

It is likewise suggested to analyse the requirement of rotation detection for all pages. The pipeline may benefit especially the end-to-the processing time, if the rotation detection is only executed for pages of the DEs class.

More data Gathering more data, create synthesized data and perform data argumentation will improve the ratio between the number of training samples for each of the models w.r.t. the number of trainable parameters.

Analysis of the uncertainty measure To find a proper approach to estimate a measure of uncertainty for each prediction, one approach would be to gray-out random super-pixels and evaluate each of the permutations. Another approach to introduce stochasticity would be to dropout perceptrons during the forwardpass [53], [54]. This approach can be used to represent a measure of uncertainty for a given ANN model.

A visual explanation of the CRNN model It is a requirement to interpret the inner-working of the CRNN model in order to achieve an end-to-end pipeline which is GDPR compliant. It is suggested to apply the Grad-CAM method per time step.

Bibliography

[1] A. L. Bæk. (2019). Github repository: The technical pipeline, [Online]. Available: https://github.com/anderslaunerbaek/thesis_pipe (visited on January 6, 2019).

[2] S. H. Sastry and M. S. P. Babu, "Implementation of CRISP methodology for ERP systems", *CoRR*, volume abs/1312.2065, 2013. arXiv: 1312.2065. [Online]. Available: http://arxiv.org/abs/1312.2065.

[3] A. Ng. (2017). Machine learning yearning, [Online]. Available: http://www.mlyearning.org (visited on Sep. 19, 2018).

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[5] L. Kang, J. Kumar, P. Ye, Y. Li, and D. Doermann, "Convolutional neural networks for document image classification", in *2014 22nd International Conference on Pattern Recognition*, 2014, pages 3168–3172. DOI: 10.1109/ICPR.2014.546.

[6] R. Smith, "An overview of the tesseract ocr engine", in *Proc. 9th IEEE Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 2007, pages 629–633.

[7] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition", *CoRR*, volume abs/1507.05717, 2015. arXiv: 1507.05717. [Online]. Available: http://arxiv.org/abs/1507.05717.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier", *CoRR*, volume abs/1602.04938, 2016. arXiv: 1602.04938. [Online]. Available: http://arxiv.org/abs/1602.04938.

[9] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization", *CoRR*, volume abs/1610.02391, 2016. arXiv: 1610.02391. [Online]. Available: http://arxiv.org/abs/1610.02391.

[10] R. R. Paulsen and T. B. Moeslund, *Introduction to Medical Image Analysis*. Kgs. Lyngby: DTU Compute, Technical University of Denmark, 2015, ISBN: 9788764309492.

[11] U. CRCV. (2015). Lecture 17 -hough transform- 2014, [Online]. Available: https://youtu.be/tAGiK1Ev6Ng?t=717 (visited on October 31, 2018).

[12] L. Ding and A. Goshtasby, "On the canny edge detector", *Pattern Recognition*, volume 34, pages 721–725, 2001.

[13] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, volume 9, number 1, pages 62–66, 1979, ISSN: 0018-9472. DOI: 10.1109/TSMC.1979.4310076.

[14] H. Madsen and P. Thyregod, *Introduction to General and Generalized Linear Models*. CRC Press, 2011, page. 64, ISBN: 978-1-4398-9114-8.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2017, ISBN: 9780387848570.

[16] L. Breiman, "Random forests", *Machine Learning*, volume 45, number 1, pages 5–32, 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324.

[17] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biology*, volume 52, pages 99–115, 1990. DOI: 10.1007/BF02459570.

[18] K. M. Fauske. (December 2006). IP over Avian Carriers with Quality of Service, http://www.texample.net, [Online]. Available: http://www.texample.net/media/tikz/examples/TEX/neural-network.tex (visited on August 9, 2018).

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[20] Massachusetts Institute of Technology. (2018). 6.s191: Introduction to deep learning - an introductory course on deep learning algorithms and their applications, [Online]. Available: http://introtodeeplearning.com (visited on November 1, 2018).

[21] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[22] Stanford School of Engineering. (2018). Cs230: Deep learning, [Online]. Available: https://web.stanford.edu/class/cs230/ (visited on November 1, 2018).

[23] TensorFlom™. (2018). Tf.contrib.layers.xavier_initializer, [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/contrib/layers/xavier_initializer (visited on November 1, 2018).

[24] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks", in *Proceedings of the 23rd International Conference on Machine Learning*, series ICML '06, Pittsburgh, Pennsylvania, USA: ACM, 2006, pages 369–376, ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143891. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143891.

[25] A. Hannun, "Sequence modeling with ctc", *Distill*, 2017, https://distill.pub/2017/ctc. DOI: 10.23915/distill.00008.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *CoRR*, volume abs/1412.6980, 2014. arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.

[27] S. Ruder, "An overview of gradient descent optimization algorithms", *CoRR*, volume abs/1609.04747, 2016. arXiv: 1609.04747. [Online]. Available: http://arxiv.org/abs/1609.04747.

[28] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pages 991–999. DOI: 10.1109/CVPR.2015.7298701.

[29] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning", *ArXiv e-prints*, March 2016. arXiv: 1603.07285 [stat.ML].

[30] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", *CoRR*, volume abs/1412.3555, 2014. arXiv: 1412.3555. [Online]. Available: http://arxiv.org/abs/1412.3555.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge", *International Journal of Computer Vision (IJCV)*, volume 115, number 3, pages 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[32] Stanford School of Engineering. (2018). Cs231n: Convolutional neural networks for visual recognition. Course notes, module 2: http://cs231n.github.io/transfer-learning/, [Online]. Available: http://cs231n.stanford.edu (visited on December 17, 2018).

[33] TensorFlow™. (2017). Preprocesses a image tensor as a numpy array. GIT SHA: f675c12, [Online]. Available: https://github.com/tensorflow/tensorflow/blob/r1.5/tensorflow/python/keras/_impl/keras/applications/imagenet_utils.py (visited on January 2, 2019).

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, volume 15, pages 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *CoRR*, volume abs/1502.03167, 2015. arXiv: 1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167.

[36] L. Arvastson and L. Clemmensen. (). Computational data analysis introduction, [Online]. Available: http://kurser.dtu.dk/course/2018-2019/02582 (visited on October 4, 2018).

[37] Z. Wojna, A. N. Gorban, D. Lee, K. Murphy, Q. Yu, Y. Li, and J. Ibarz, "Attention-based extraction of structured information from street view imagery", *CoRR*, volume abs/1704.03549, 2017. arXiv: 1704.03549. [Online]. Available: http://arxiv.org/abs/1704.03549.

[38] Den Europæiske Unions Tidende. (2016). Europa-parlamentets og rådets forordning (eu) 2016/679 af 27. april 2016 om beskyttelse af fysiske personer i forbindelse med behandling af personoplysninger og om fri udveksling af sådanne oplysninger og om ophævelse af direktiv 95/46/ef (generel forordning om databeskyttelse). PDF: https://eur-lex.europa.eu/legal-content/DA/TXT/PDF/?uri=CELEX:32016R0679&from=DA, [Online]. Available: https://eur-lex.europa.eu/legal-content/DA/TXT/?uri=CELEX:32016R0679 (visited on December 27, 2018).

[39] T. L. Pedersen and M. Benesty. (2016). Understanding lime, [Online]. Available: https://github.com/thomasp85/lime/blob/master/vignettes/Understanding_lime.Rmd (visited on December 27, 2018).

[40] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization", *CoRR*, volume abs/1512.04150, 2015. arXiv: 1512.04150. [Online]. Available: http://arxiv.org/abs/1512.04150.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *CoRR*, volume abs/1409.1556, 2014. arXiv: 1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556.

[42] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, 2004.

[43] S. V. Rice, F. R. Jenkins, and T. A. Nartker, *The fourth annual test of ocr accuracy*, 1995.

[44] A. S. Agbemenu, J. Yankey, and E. O. Addo, "An automatic number plate recognition system using opencv and tesseract ocr engine", *International Journal of Computer Applications*, volume 180, number 43, pages 1–5, 2018. DOI: 10.5120/ijca2018917150.

[45] tesseract-ocr. (2019). Tesseract open source ocr engine (main repository), [Online]. Available: https://github.com/tesseract-ocr/tesseract (visited on January 7, 2019).

[46] H. Scheidl. (2018). An intuitive explanation of connectionist temporal classification: Text recognition with the connectionist temporal classification (ctc) loss and decoding operation, [Online]. Available: https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c (visited on January 3, 2019).

[47] keras team. (2018). Image_ocr.py. GIT SHA: 96c3c19, [Online]. Available: https://github.com/keras-team/keras/blob/master/examples/image_ocr.py (visited on January 7, 2019).

[48] J. Cohen, "A coefficient of agreement for nominal scales", *Educational and Psychological Measurement*, volume 20, number 1, pages 37–46, 1960.

[49] D. G. Altman, *Practical Statistics for Medical Research*. Chapman and Hall, 1991, page 404, ISBN: 0412276305.

[50] S. Simon. (). What is a kappa coefficient? (cohen's kappa), [Online]. Available: http://www.pmean.com/definitions/kappa.htm (visited on Sep. 27, 2018).

[51] H. Hyyroe. (2018). A bit-vector algorithm for computing levenshtein and damerau edit distances, [Online]. Available: https://pdfs.semanticscholar.org/813e/26d8920d17c2afac6bf5a15c537b067a128a.pdf (visited on December 12, 2018).

[52] Python Advanced Course Topics. (2018). Levenshtein distance, [Online]. Available: https://www.python-course.eu/levenshtein_distance.php (visited on December 12, 2018).

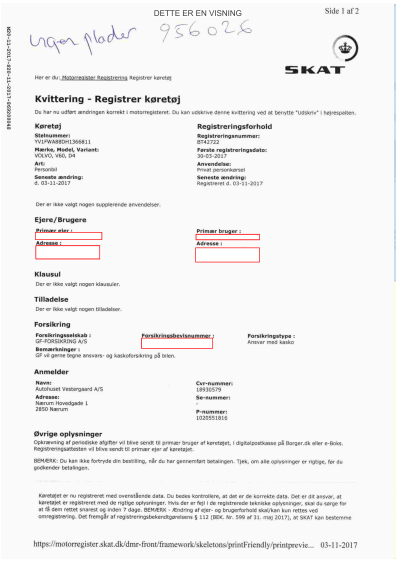
[53] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning", in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., series Proceedings of Machine Learning Research, volume 48, New York, New York, USA: PMLR, 2016, pages 1050–1059. [Online]. Available: http://proceedings.mlr.press/v48/gal16.html.

[54] MIT 6.S191. (2018). Deep learning limitations and new frontiers, [Online]. Available: https://youtu.be/l_yWLAQg7LU?t=1251 (visited on January 10, 2019).

APPENDIX A

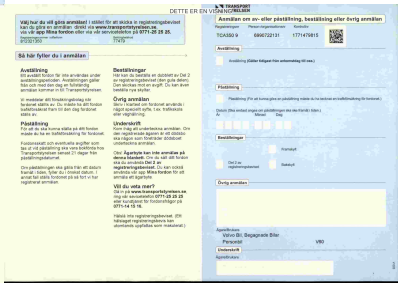
Methods on a Swedish VRD

This chapter illustrates the applied methods on Swedish VRD similar to the applied methods for a German VRD which are included in the report. The visualizations illustrated in this section shows how the methods are working on a VRD with a different origin. All figures in this section are showcasing a VRD from Sweden.



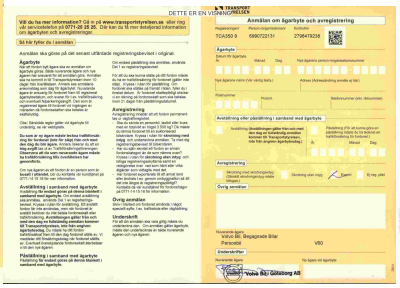
(a) Page 1. Class META.

(b) Page 2. Class SE.



(c) Page 3. Class SE.

(d) Page 4. Class SE.



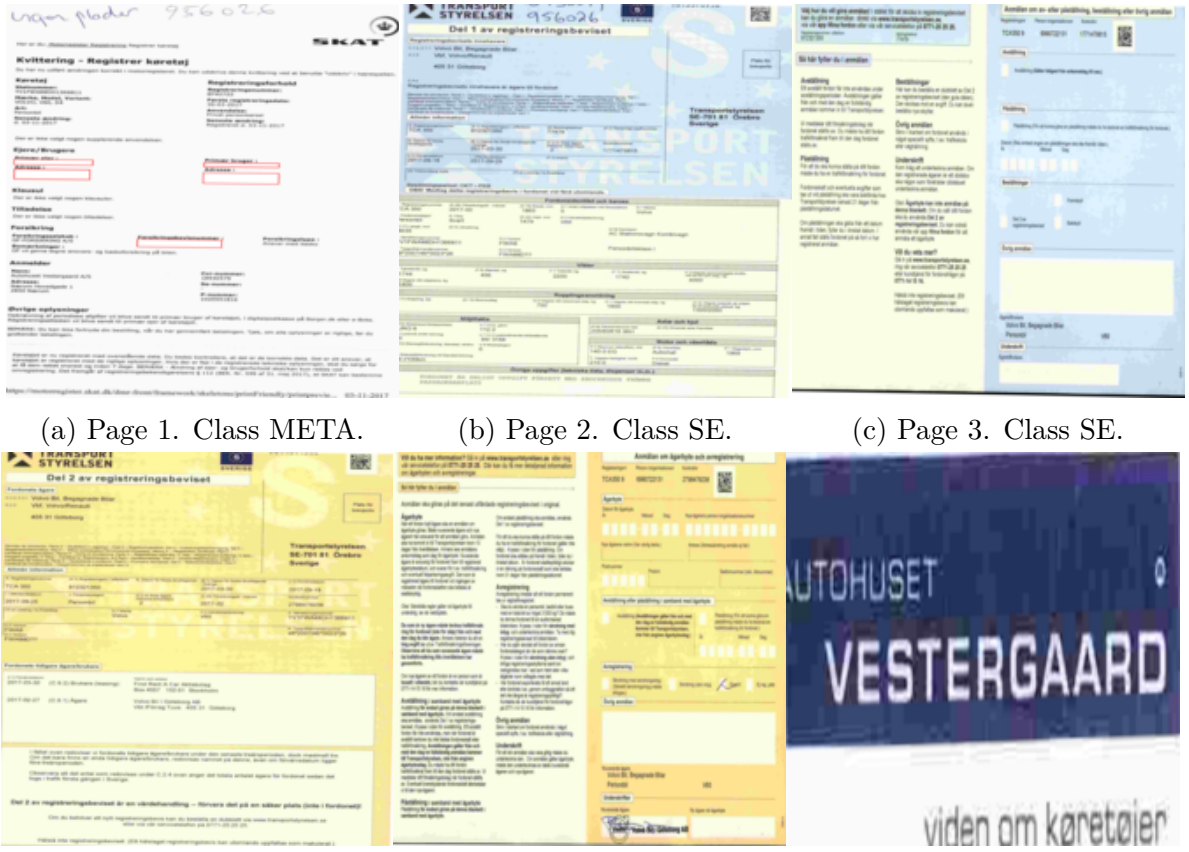
(e) Page 5. Class SE.

(f) Page 6. Class META.

Figure A.1: These six images shows each of the pages from a raw Swedish VRD.

A.1 Example of a Pre-processed VRD

Figure A.2 is illustrating the first staged pre-processed Swedish VRD. The original VRD is given in figure A.1.



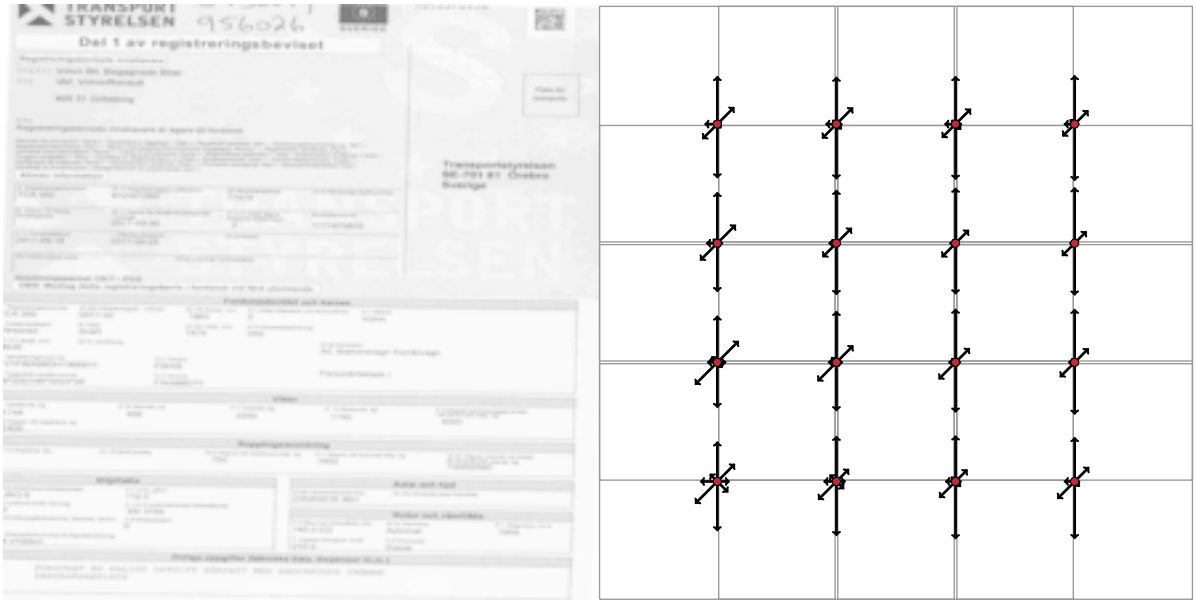
(a) Page 1. Class META. (b) Page 2. Class SE. (c) Page 3. Class SE.

(d) Page 4. Class SE. (e) Page 5. Class SE. (f) Page 6. Class META.

Figure A.2: These six images have been pre-processed. The original VRD images are illustrated in figure A.1.

A.2 Example of the Determined HOG Features

Figure A.3 provides an illustration of the computed HOG descriptor for a single page in a VRD from Sweden.

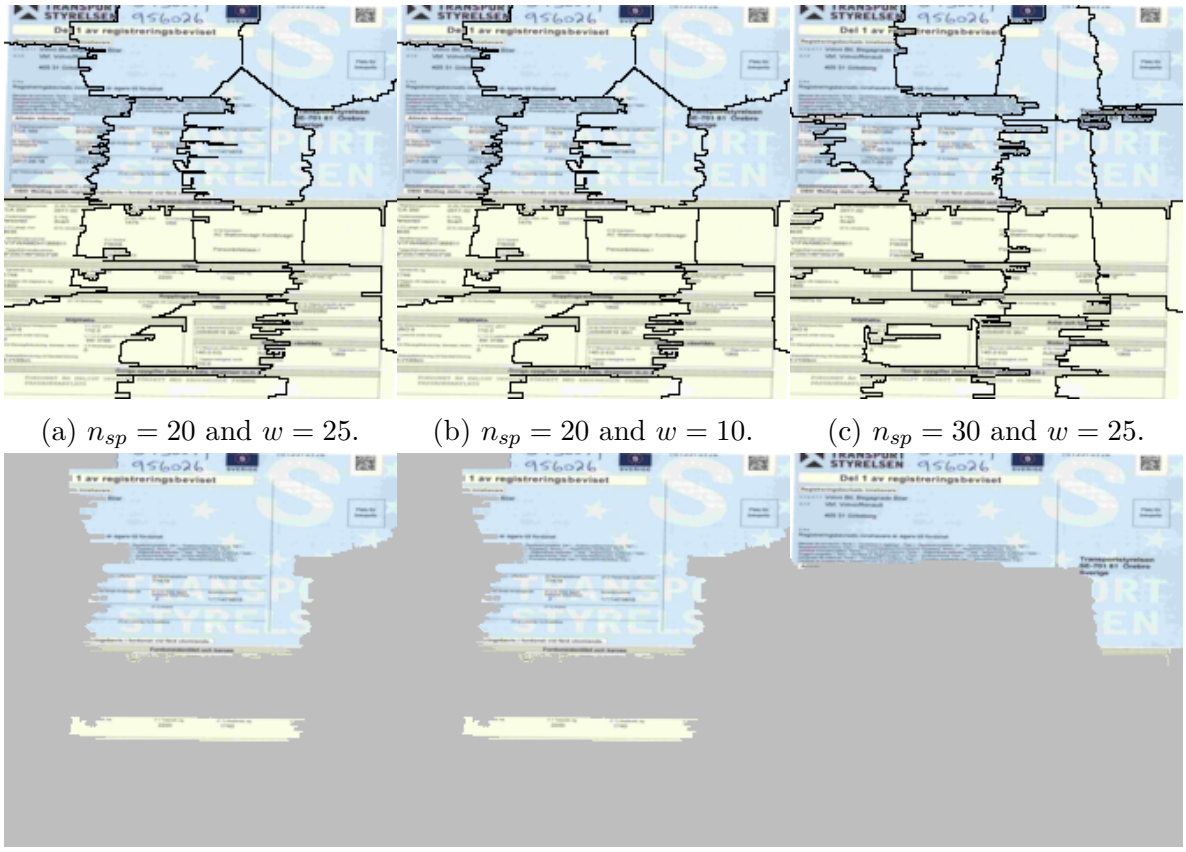


(a) A pre-processed image of figure A.1b. (b) Computed HOG descriptor, fig. A.3a.

Figure A.3: This figure shows an illustration of the main trends of direction and magnitude for each of the local spatial regions.

A.3 Example of a LIME Explanation

Figure A.4 illustrates three hyperparameter settings for a single page of a Swedish VRD similar to figure 5.19.



(a) $n_{sp} = 20$ and $w = 25$. (b) $n_{sp} = 20$ and $w = 10$. (c) $n_{sp} = 30$ and $w = 25$.

(d) $n_{sp} = 20$ and $w = 25$. (e) $n_{sp} = 20$ and $w = 10$. (f) $n_{sp} = 30$ and $w = 25$.

Figure A.4: This figure shows three examples of visual explanations using three different hyperparameter settings. Common for all examples are the number of super-pixels to include in each explanation, $n_{expo} = 5$ and the class to be explained. The first example in figure A.4a and in figure A.4d, uses $n_{sp} = 20$ and $w = 25$.

A.4 Example of a Grad-CAM Explanation

Figure A.5 illustrates the Grad-CAM explanation on a single page in a VRD from Sweden similar to figure 5.20.

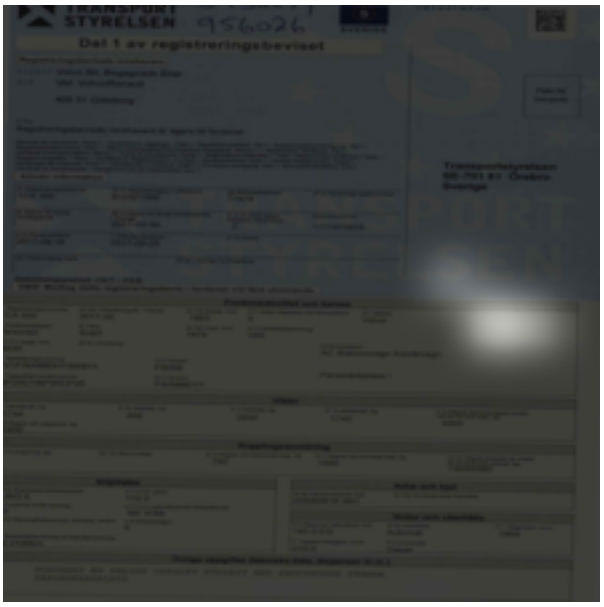


Figure A.5: This figure illustrates the discriminative class location based upon the layers of the CNN model. Locations with high intensity pixels are referring to areas of the image which are positively effecting the prediction of the class of interest.

APPENDIX B

Algorithms

B.1 The HOG Algorithm

Algorithm 1 reports the approach for calculating the HOG-features. This procedure has been implemented in plain `R`. Further performance enhancements can be achieved by a C++implementation.

input : An 3D-array of gray-scale images (img) of size $N \times rows \times cols$.
output : An 2D-array of size $N \times (cell^2 \cdot n_{bins})$ containing the HOG features per img.
parameter : $cell = 4$: the number of HOG windows per img.
parameter : $n_{bins} = 8$: the number of histogram bins in a HOG cell.

// initialize the dimensions and step sizes in x and y direction:
$N, rows, cols = dim(img)$
$step_x = floor(cols/(cell + 1))$
$step_y = floor(rows/(cell + 1))$

for $i \leftarrow 1$ **to** N **do**
  // calculate gradient in x and y direction:
  $\overrightarrow{G_i}(g_x, g_y) = \nabla img_i(g_x, g_y)$ ▷ eq. 5.7.

  // calculate elementwise angles and magnitudes of the gradients:
  $angles(x, y) = \text{atan2}(g_y(x, y), g_x(x, y))$
  $magnit(x, y) = \sqrt{g_y(x, y)^2 + g_x(x, y)^2}$ ▷ eq. 5.10 and eq. 5.11.

  // initialize HOG feature vector and *count* variable
  $H_i = \{x_1, x_2, ..., x_{cell^2 \cdot n_{bins}}\}$
  $count = 0$

  // loop each HOG window and calculate features
  for $n \leftarrow 0$ **to** $cell - 1$ **do**
    for $m \leftarrow 0$ **to** $cell - 1$ **do**
      $count+ = 1$
      // subset the *angles* and *magnit* matrices for the given HOG window and flatten as vectors. initialize the bin_{count} variable and a sub HOG feature vector.
      $H_{i_{sub}} = \{x_1, x_2, ..., x_{n_{bins}}\}$
      // assembling the histogram for the given HOG cell
      for $ang \leftarrow -\pi + 2 * \pi/n_bins$ **to** π **by** $2 * \pi/n_{bins}$ **do**
        $bin_{count}+ = 1$
        // find indices which satisfy the following: $idx = ang > angles_{sub}$
        $angles_{sub}(idx) = 100$
        $H_{i_{sub}}(bin_{count}) + = magnit_{sub}(idx)$
      end
      // norm and return features for the given HOG window
      $H_i(cell) = \|H_{i_{sub}}\|^2$
    end
  end
end
return H

Algorithm 1: This algorithm illustrates the approach for calculating the HOG-features. This procedure has been implemented in plain `R`.

APPENDIX C

Modelling

C.1 An Overview of the CV Grid-Search

The first row of table C.1 presents the highest average $\lfloor\kappa\rfloor$ metric which are having the lowest standard deviation of this metric.

The optimal hyperparameters found with CV and grid-search are presented here:

- The optimal: `ntrees`=900, grid range: 500 to 2,500 by 200.
- The optimal: `mtry`=16, grid range: 10 to 20 by 1.

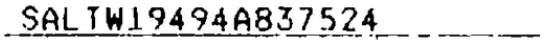
Position	*ntree*	*mtry*	κ (%)	κ SD (%)	Error Rate (%)
1	900	16	85.1473	0.7158	8.3583
2	2100	20	85.201	0.7223	8.3311
3	1300	17	85.1839	0.7224	8.3397
4	1100	20	85.1759	0.7247	8.3451
5	900	13	85.0392	0.7273	8.4143
6	2100	19	85.1741	0.73	8.3467
7	1500	20	85.1031	0.7304	8.3848
8	1700	19	85.2099	0.7316	8.3265
9	1100	19	85.1206	0.7342	8.3747
10	1500	18	85.1927	0.739	8.3335
⋮	⋮	⋮	⋮	⋮	⋮
121	500	12	84.952	0.9371	8.4586

Table C.1: This table presents the average κ metric for each of the unique hyperparameter sets. The table has been arranged according to the highest $\lfloor\kappa\rfloor$ with the lowest standard deviation.

APPENDIX D

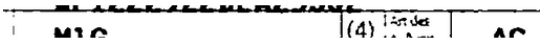
Error Analysis

Figure D.1 illustrates six cropped text fields which demonstrates the poor data foundation for the second stage models. The presented text fields are captured during the initial error analysis. All images have the same dimension $512 \times 256 \times 1$.



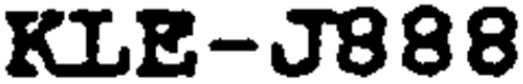
Predicted: SAL1W19494A837524
Label: SALTW19494A837524
LD: 1

(a) VIN, index 958.



Predicted: WW1ADZZ901010G911
Label: WP1ZZZ92ZBLA25362
LD: 13

(b) VIN, index 1536.

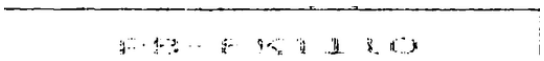


Predicted: MIM888
Label: KLEJ888
LD: 4

(c) LP, index 3359.

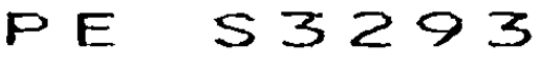
Predicted: WHHSJ12222
Label: HHGM2323
LD: 6

(d) LP, index 2343.



Predicted: VH1011
Label: PBFK1110
LD: 6

(e) LP, index 2121.



Predicted: PES293
Label: PES3293
LD: 1

(f) LP, index 2109.

Figure D.1: This figure illustrates samples of six misclassified text sequences. The correct label, the predicted text sequence and the LD metric are reported in the lower left corner in each of the images.

{2.7182818284}ϑωερτυθιοπσδφγηξκλ