Bachelor Thesis
in Information Systems and Management

Comprehensive Overview of Scene Text Spotting Methods with Deep Learning

Johannes Reichle
Matriculation No. 04797218

Supervisor	Prof. Dr. Rainer Schmidt
Date of Submission	XX.XX.2022

Declaration

I hereby certify that I have written this bachelor thesis on my own and that I have not used any sources or aids other than those indicated.

Munich, the XX.XX.2022

. . . .
Johannes Reichle

Abstract

Here abstract for Bachelor Thesis.

Keywords: Deep Learning, Scene Text Spotting, Overview

Contents

List of Figures

List of Tables

Abbreviations

CNN Convolutional Neural Networks

DL Deep Learning

DNN Deep Neural Network

GD Gradient Descent

ML Machine Learning

MLP Multi Layer Perceptron

MLS Machine Learning System

NN Neural Network

OCR Optical Character Recognition

RNN Recurrent Neural Networks

STD Scene Text Detection

STR Scene Text Recognition

STS Scene Text Spotting

Notation

Calculus

$\frac{\delta y}{\delta x}$	Partial derivative of y with respect to x
$\nabla_{\mathbf{x}} y$	Gradient of y with respect to **x**

Datasets

$\mathbb{X}$	A set of training examples
$\mathbf{x}^{(i)}$	The i-th example (input) from a dataset
$y^{(i)}$ or $\mathbf{y}^{(i)}$	The target associated with $\mathbf{x}^{(i)}$
X	The $m \times n$ matrix with input example $\mathbf{x}^{(i)}$ in row $X_{u,:}$

Numbers and Arrays

A	Matrix
a	Scalar
A	Tensor
a	Vector

Other

$\|\mathbf{x}\|$	L^2 norm of **x**
$\mathbb{R}$	Real numbers
$f(\mathbf{x}; \boldsymbol{\theta})$	A function of **x** parametrized by $\boldsymbol{\theta}$

Chapter 1

Introduction

1.1 Motivation

Optical Character Recognition (OCR) is the concept of extracting typed, handwritten or printed text from an image (Zhao et al., 2020). Techniques for this concept have improved a lot due to the advances in the field of Deep Learning (DL) (Zhao et al., 2020). When compared to traditional methods DL improves automation, effectiveness and generalization (Chen et al., 2021). DL is a technology based on Neural Networks (NNs) where data is processed in multiple layers to extract complex features to solve a given problem (Shrestha and Mahmood, 2019). DL has only caught on in the recent years as the big computational cost has been met by improvement in computer hardware as well as in automatic feature learning (Ponti et al., 2017; Chen et al., 2021). Applying these new capabilities and finding the right solution in the space of DL for the use case of extracting information of labels is the focus of this thesis. This is an interesting task as performance of OCR systems in natural scenes is still challenging (Zhao et al., 2020; Chen et al., 2021). Such scenes entail natural scenes captured by a camera (Chen et al., 2021; Baek et al., 2019). Factors such as complex backgrounds, noise, perspective and variability in fonts, colors and sizes, of scene texts complicate the process (Hu et al., 2020b; Chen et al., 2021; Baek et al., 2019).

1.2 Problem description

Technicians in the field work with different equipment. It is useful to digitize the lables of such equipment, to keep an overview over the inventory (Abramowicz and Corchuelo, 2019). The goal of this thesis is to find a solution which simplifies the digitization of equipment labels. The research question guiding the process is most crucial: Which state of the art DL approaches for scene text OCR are viable for the use case of extracting textual label data from images. The definition of the viability of an approach has to be determined for this. What qualities such as detecting alpha-numeric strings or suitability despite inadequate image conditions must a solution have (Ghosh et al., 2017; Hu et al., 2020b)?

It is difficult to assess how well a DL approach performs before it has been implemented and tested on the specific problem or dataset (Arpteg et al., 2018). Therefore, multiple promissing approaches that can be implemented and experimented with must be identified and compared. The reasearch and discussion of techniques from end-to-end scene Scene Text Spotting (STS) to dividing the pipeline into Scene Text Detection (STD) and Scene Text Recognition (STR) is centered Chen et al. (2021) around the requirements which are given by the problem.

The article Ashmore et al. (2021) defines four phases of the Machine Learning (ML) lifecycle, namely, Data Management, Model Learing, Model Verification and Model Deployment. Only the substage Model Selection from Model Learning will only be looked at in the scope of this thesis. Other aspects such as data analysis, implementation, training, deployment and maintenance of a solution in a production environment shall not be performed.

1.3 Methodology

The methodology of this thesis can be labeled as a literature review (Snyder, 2019; Torraco, 2005). The goal is to provide an overview over current DL pipelines and models that can help in choosing which to implement and test to solve the specific problem defined in Section 1.2 and more detailed in Chapter 3.

The research question guiding the process is most crucial: Which state of the art DL approaches for scene text OCR are viable for the use case of extracting textual label data from images. In order to improve the validity for the subsequent analysis, the problem is dissected further. This includes analysing the specific use case as well as researching which qualities have been identified as generally critical for scene text approaches. The qualities are taken from literature which covers ML in general to literature which covers OCR under challenging scene text conditions.

The literature is identified through searching in reputable journals. All research after 2017 which pertains to extracting scene text is regarded as relevant. Standard OCR solutions may not hold validity in practice, as the image and text conditions can vary in the defined problem (Chen et al., 2021). An important criteria is that the paper contributes to the ML model. This extends to the whole pipeline from preprocessing an image to the final result of the model. Concludent to the distinction in Section 1.2, contributions to other stages in the ML lifecycle are not examined. Therefore, keywords for the search include: Deep Learning, Optical Character Recognition, Text Detection, Text Recognition, Text Spotting, Scene Text.

The identified literature is synthesized into an overview over the most common approaches. This includes listing important factors for DL such as the number of parameters, or which type of layers are used to achieve success. The overview will be organized into the categories for the ML pipeline, such as End-to-End solutions as in Xing et al. (2019) or a split into Text Detection and Text Recognition as in Yang et al. (2021); Chen and Li (2018).

In the analysis possibly viable approaches are compared with the qualities defined in Chapter 3. The approaches are analysed in detail in regards to commonalities as well as differences and the possible effect on the feasibility. The analysis thus shows which approaches are worthwhile to apply the whole ML lifecycle to.

1.4 Expected results

In addition to a deeper understanding of the problem and its detailed definition, the literature review lays the foundation for finding the right approach for the extraction of textual information from images with equipment labels through literature review. In the subsequent analysis different approaches are highlighted for their theoretical fit as a solution.

In the following, the structure of this thesis is listed and each chapter's expected result is detailed along with its benefits for the overall objective of producing an overview of state of the art scene text OCR relevant for the problem described in Section 1.2. comprehension of the following chapters is gathered. This includes general principles of DL and by extension ML but also of OCR. In Chapter 3 the problem from Section 1.2 is addressed in more detail. The result shall be a firm understanding of qualities that a solution must possess. These requirements are the point of focus for the further examination of techniques. After laying the foundation, in Chapter 4 current research in regards to the identified requirements is examined. The resulting overview can be viewed as a basis for a decision when it comes implementing a practical solution. Therefore, it enables the discussion in Chapter 5. Here not only the results and the availability of a solution but also the methodology of this work are assessed critically. The conclusion is a summary of the results compared to the expected results detailed in this chapter as well as an outlook for further research into the topic.

Chapter 2

Theoretical Foundation

This chapter succinctly describes principles which build the foundation for later chapters. Only the most relevant topics are touched upon, necessary the details are explained in later chapters. The mathematics that makes the techniques possible is not explained in depths as it would otherwise exceed the scope of this work. Whenever possible heavy mathamatical notation is omitted if it does not aid the understanding of the reader.

2.1 Machine Learning

In order to grasp DL, a solid understanding of ML has to be developed first (Goodfellow et al., 2016). This is because DL is a subfield of ML (Chauhan and Singh, 2018). The most well known definition for ML comes from Mitchell (1997): 'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, improves with experience E'.

The task that the Machine Learning System (MLS) learns to perform, can range from approximating a function (e.g. regression — $f : \mathbb{R}^n \to \mathbb{R}^l$, classification — $f : \mathbb{R}^n \to \{1, \ldots, k\}$) to optaining a different representation for the data that has beneficial properties for further processing but preserves as much information as possible (e.g. PCA for compression) (Goodfellow et al., 2016). Note that the learning itself is not the task but merely the process of improving on performing the task (Goodfellow et al., 2016). One of the most well known ML algorithms is Linear Regression. In the following the algorithm is used as an example for explaining ML principles. As the name implies, Linear Regression is used to predict a value $\hat{y} \in \mathbb{R}$ given the input vector $\mathbf{x} \in \mathbb{R}^n$ which is made up of the features x_i. The goal is to approximate the ground truth y. Linear is derived from the underlying model shown in

Equation 2.1:

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^T \cdot \mathbf{x} + b = \sum_{i=1}^{n} w_i x_i + b = \hat{y} \tag{2.1}$$

The scalar product of the weights $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{x}$ is added to the bias term $b \in \mathbb{R}$. Both $\mathbf{w}, b$ are parameters that are learned by the model in order to optimize the approximation (Goodfellow et al., 2016).

The performance of a model measures how well the task can be completed. Depending on the task of the MLS, different quantitative measures are used. The metric Mean Squared Error (see Equation 2.2) can be used for Linear Regression.

$$MSE = \frac{1}{m} \|(\hat{\mathbf{y}} - \mathbf{y})\|^2 = \frac{1}{m} \sum_{i=1}^{m} \left((\mathbf{w}^T \mathbf{x}^{(i)} + b) - y^{(i)}\right)^2 \tag{2.2}$$

Here m denotes the number of examples $\mathbf{x}^{(i)}$ with the associated targets $y^{(i)}$, used to calculate the error (Géron, 2017; Goodfellow et al., 2016). The goal is to minimize the generalization error which measures the expected performance on previously unseen input (Géron, 2017). For this the test set is used, once the model has been trained. The test set is a part of the available data (Géron, 2017; Goodfellow et al., 2016). The generalization error can be divided into three components. The bias error arises from simplifying assumptions for the model, the variance error measures the variation in the model outcome depending on the data used for training. Both these errors are influenced by the model's capacity which is why the relationship between them is call the Bias/Variance tradeoff. Lastly the irreducible error stems from not having measured all data as well as the variation in real data and cannot be reduced (Ashmore et al., 2021; James et al., 2013; Géron, 2017).

The experience part of ML depicts the process where the algorithm is 'experiencing' the training dataset $\mathbb{X}$ and is learning important properties of the dataset. In general there are two paradigms for training: supervised and unsupervised (Goodfellow et al., 2016). Linear Regression is an example for supervised learning, as the model is using the ground truth value to learn approximating $y^{(i)}$ for the associated input $\mathbf{x}^{(i)}$ (Alzubi et al., 2018; Goodfellow et al., 2016). For unsupervised learning on the other hand the algorithm is not directed to predict a target value but to learn properties about the data and to leverage them for representation tasks like compressing or denoising the data (Goodfellow et al., 2016; Géron, 2017). In most cases training can be described as an optimization problem, i.e. as minimizing a function — the so called objective or loss function (Goodfellow et al., 2016). The MSE introduced earlier can be used for Linear Regression (see Equation 2.3). This objective function has properties which make it suitable for models which have linear output (Goodfellow et al., 2016).

$$\min_{\mathbf{w},b} MSE(\mathbf{w}, b) \tag{2.3}$$

Note that for minimization the MSE is a function of $\mathbf{w}, b$ and not of $\mathbf{x}$, in terms of predicting a value the MSE is a function of $\mathbf{x}$ parametrized by $\mathbf{w}, b$ (see Equation 2.3). In Equation 2.1 $\mathbf{w}$,b are parameters that have to be learned in order to minimize the generalization error (James et al., 2013; Géron, 2017). For other tasks such as binary classification, the metric (e.g. F_1-Score) and the objective function (binary cross entropy loss) are different (Géron, 2017; Ho and Wookey, 2020). For optimization the Gradient Descent (GD) algorithm is prevalent, especially in the subfield of DL. As the name suggests, the gradient is used to iteratively update the parameters $\mathbf{w}, b$ to arrive at a minimum of the objective function (see Equation 2.4 and 2.5) (Géron, 2017).

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon \cdot \nabla_{\mathbf{w}} MSE(\mathbf{w}, b) = \mathbf{w} - \frac{2\epsilon}{m} \mathbb{X}^T (\mathbb{X}\mathbf{w} + b - \mathbf{y}) \tag{2.4}$$

$$b \leftarrow b - \epsilon \cdot \frac{\delta}{\delta b} MSE(\mathbf{w}, b) = b - \frac{2\epsilon}{m} (\mathbb{X}\mathbf{w} + b - \mathbf{y}) \tag{2.5}$$

The learning rate constant ϵ can be adjusted to speed up or slow down the 'steps' which can have different effects on the convergence (Goodfellow et al., 2016). There are more sophisticated variations of the GD algorithm which are more suited for practical application (e.g. RMSProp, Adam) (Géron, 2017). Note that the process minimizes the test error with the test set $\mathbb{X}$. The effect on the generalization error depends on model capacity which is the space of functions the model enables (Goodfellow et al., 2016). Linear Regression has the capacity to fit data with a linear relationship between features and ground truth. If the underlying relationship is more complicated, the model can only underfit the data (model bias) (Goodfellow et al., 2016). Polynomial Regression has more capacity for example. Say the real relationship between features and ground truth now actually is linear; the Polynomial Regression model can overfit for statistical outliers in the test set which is why in this case the model with the lower capacity can achieve a lower generalization error (Géron, 2017). Therefore, it is important to improve the bias/variance tradeoff. Aside from model selection, there are different techniques use for this (e.g. Regularization) (Goodfellow et al., 2016).

2.2 Deep Learning

In DL Deep Neural Networks (DNNs) are leveraged to automatically learn new representations of data through multiple layers of abstraction. This makes

DNNs powerful function approximators (Goodfellow et al., 2016). In the this section the basics of NNs are explained and popular basic architectures thereof are introduced.

The most basic NN is called a feedforward NN or Multi Layer Perceptron (MLP) where the information only flows in one direction (in contrast to Recurrent Neural Networkss (RNNs)) (Goodfellow et al., 2016). The network is made up of artificial neurons. These neurons are arranged as a directed acyclic graph with multiple so called layers (Goodfellow et al., 2016). The first layer which receives the input features $\mathbf{x}$ is called the input layer, the last layer which outputs the final estimation of y or $\mathbf{y}$ is called the output layer, all layers in between are called the hidden layers (Shrestha and Mahmood, 2019). The structure with which the NN is build in terms of how many layers, how many neurons in each layer and how they are connected, is called architecture (Goodfellow et al., 2016). A neuron, the basic building block of NNs, receives input from neurons in the previous layer and calculates a single value which is propagated to neurons in the following layer (Shrestha and Mahmood, 2019). The value is calculated by feeding the received information into a Linear Regression model (see Equation 2.1). The resulting value is an fed into an activation function g which introduces nonlinearity, to allow more complicated transformations of information and representation (Goodfellow et al., 2016). Popular activation functions include ReLU, tanh, sigmoid (Shrestha and Mahmood, 2019).

$$ReLU(x) = \max(0, x) \tag{2.6}$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.7}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

Note that for e.g. regression, the output layer can omit the activation function (Goodfellow et al., 2016). The calculation of the prediction is basically a concatenation of the functions defined by the neurons, the process of which is called forwardpropagation (Goodfellow et al., 2016).

$$\hat{y} = f(f(\mathbf{x}; \boldsymbol{\theta}^{(1)}; \boldsymbol{\theta}^{(2)})) \tag{2.9}$$

Here f represents the function which is performed by the neurons (linear regression + activation). $\boldsymbol{\theta}^{(i)}$ in Equation 2.9 stands for the parameters in layer i with $T^{(i)}_{j,:}$ being the parameters the $j-th$ neuron in that layer (Goodfellow et al., 2016). For simplification both $\mathbf{w}, b$ are refered to as $\boldsymbol{\theta}$. The term DNN comes from adding many hidden layers to the NN (Shrestha and Mahmood, 2019). This allows for a more complicated function and better developed features or representations that are extracted from the input feature vector $\mathbf{x}$ (Oyedotun et al., 2015). The DNN can be trained as a whole, thus making feature engineering redundant in contrast to normal ML algorithms (Arpteg et al., 2018). The training algorithm is called backpropagation. It makes use of the chain rule of calculus which allows the algorithm to propagate the training error that is calculated through the objective function in conjunction with the output of forwardpropagation (Goodfellow et al., 2016). The propagation is performed from the output layer back to the first layer.

2.3 Scene Text Spotting

1. def
2. little history
3. need for STS / difference OCR
4. evaluation metric and matching prediction to ground truth
5. common data sets

Chapter 3

Problem analysis

This chapter entails an analysis of the problem which is the research question's foundation. It is crucial, as the quality of requirements ultimately determines the quality of the overview and subsequent analysis.

Requirements for a software system that involves ML and thus DL differs from the traditional approach. The data-driven software components are not entirely defined by the programmer but are influenced by data. The system acts with dependency on the test data (Siebert et al., 2021). This poses a challenge in determining requirements and measuring quality of results (Nakamichi et al., 2020). Instead of categorizing functional and non-functional requirements, like for traditional software projects (Zowghi et al., 2014), qualities that a MLS must possess are defined.

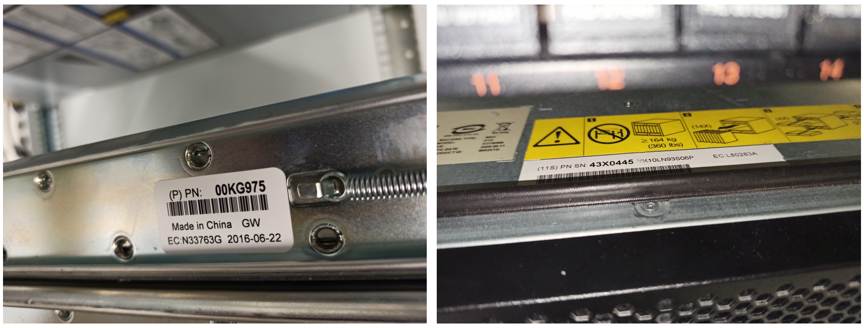
3.1 Use Case

The problem can be depicted by a use case. This use case sets the foundation for determining requirements for an approach because qualities derive from the intended purpose of use (Siebert et al., 2021). Table 3.1 gives an overview over the relevant properties that can be derived from the use case. For this

Offline Capabilities	Perform extraction process offline
Alphanumeric recognition	Recognize alphanumeric strings such as serial numbers
Semantics retention	Retain semantics given implicitly be space, strucure and rotation of text in labels

Table 3.1: Qualities specific to use case — exclusion criterias

thesis, the basic use case is as follows: A technician takes a photo of a device label with his smart phone. For this the technician is situated in locations like a cable shaft. Due to this, there's no internet availability. The process from taking the image to storing the extracted text safely must work offline. The resulting image contains printed textual information which must be extraced by an application on the smart phone. Space and structure of this information can vary from label to label (see figure 3.1). The text, spacing and structure carries semantic information which can be important for later processing in the scope of a business process (Chen et al., 2021). The goal is to extract the text and preserve semantics that are implicitly provided through structure and space. This means text and the respective coordinates, height, width and a possible rotation angle must be output as the result (Yang et al., 2021). Those values can then be transformed into other formats such as JSON or HTML as needed. In addition to this, the labels can contain arbitrary alphanumeric



(a) Positive example (b) Negative example

Figure 3.1: Examples for label images

strings such as serial numbers (see figure 3.1). This results in the requirement that the DL model has to be able to recognize sequences that are not part of a predefined lexicon (Ghosh et al., 2017). The qualities for the MLS that can be derived directly from the use case (see table 3.1) can be regarded as excluding criterias, because an approach that does not possess the qualities in question, cannot be regarded as viable for the use case.

3.2 Quality Identification

In the article Ashmore et al. (2021) the qualities are identified and assigned to different challenges in regards to working with MLS: Development Challenges, Production Challenges, Organizational Challenges. Because the only the Model Selection substage of the lifecycle is performed, the challenges and their qualities are not relevant for this thesis, as they concern the operational aspect of MLSs.

In Nakamichi et al. (2020); Siebert et al. (2021) systematic approaches for identification and documentation of qualities are detailed. In MLSs various entities interact to in order to produce the desired functionality. The paper Nakamichi et al. (2020) suggests that in order to adequately evaluate the qualities, it is essential to not only consider the model but the entire MLS. These entities are data, model, environment, system/infrastructure (Nakamichi et al., 2020; Siebert et al., 2021). The article Siebert et al. (2021) differentiates between system and infrastrucure. The infrastrucure represents given hardware and available libraries, whereas the system depicts the software that surrounds the model in the runtime environment. The data view pertains to the quality of development and runtime data (Siebert et al., 2021). The model consists of subcomponents organized in directed acyclc graph building a pipeline. This directed acyclic graph depicts everything from processing the images to the extracted information (Siebert et al., 2021). The environment entity covers the external aspects to the MLS which may interact with it (Siebert et al., 2021). In the scope of this work the environment entails mostly the conditions in which images are taken. For this thesis the entities data and system cannot be regarded as given. The entities environment and infrastrucure are only losely defined through the use case. That is why the systematic approaches cannot be performed in the scope of this thesis. For example Siebert et al. (2021) proposes to follow the systematic CRISP-DM approach of identifying qualities. It cannot be performed due to the lack of data and the other entities. Instead many qualities that are highlighted by research are taken into account along with critical qualities (offline capabilities, alphanumeric recognition, semantic retention) that are directly derived from the use case. The Table 3.2 (p. 17) lists all qualities that pertain to the model entity. Different qualities are *grouped together* for their similarities. Because of their properties they can be evaluated jointly. When it comes to documenting the identified qualities, both Nakamichi et al. (2020) and Siebert et al. (2021) define a meta model for qualities that combines qualities with measurement methods and values and assignes them to an entity of the MLS. The implementation and testing phase are not performed in the scope of this thesis and the difficulty in assessing the performance ahead of those phases, prevents the evaluation of measurements. Additionally, experimental results from literature can only be compared as long as factors such as hardware, platform, source code, configuration and dataset are uniform (Arpteg et al., 2018). Comparing models through results of diferent papers is troublesome, because different papers might use different evaluation and testing environments (Baek et al., 2019). This applies to studies that present an overview such as Chen et al. (2021); Long et al. (2021). These studies can only be regarded as guiding values because the performance for a specific dataset cannot be predicted without testing on it Arpteg et al. (2018). That's why targets for measurements are not defined, as evaluation would only deliver a false sense of certainty.

Qualitiy	**Source(s)**
Appropriateness	
Appropriateness	Siebert et al. (2021)
Suitability	Siebert et al. (2021)
Model Fitness — Quality of Output Data	Nakamichi et al. (2020)
Performance	
Performance	Ashmore et al. (2021); Vogelsang and Borg (2019)
Accuracy	Nakamichi et al. (2020)
Model Fitness — Degree of Correctness	Nakamichi et al. (2020); Zhang et al. (2020)
Development correctness	Siebert et al. (2021)
Robustness	
Robustness	Ashmore et al. (2021); Hu et al. (2020a); Siebert et al. (2021)
Robustness Against Change of Input Data	Nakamichi et al. (2020)
Robustness Against Noise Data	Nakamichi et al. (2020)
Relevance / bias-variance tradeoff	Siebert et al. (2021); Zhang et al. (2020)
Trained Model Generalization Performance Appropriateness	Nakamichi et al. (2020)
Reusability	Ashmore et al. (2021)
Interpretability	
Interpretability	Ashmore et al. (2021); Siebert et al. (2021); Zhang et al. (2020)
Understandability	Nakamichi et al. (2020)
Transparency	Arpteg et al. (2018)
Model Explainability	Vogelsang and Borg (2019)
Comprehensibility	Ashmore et al. (2021)
Comprehensiveness	Ashmore et al. (2021)
Fairness	
Fairness	Siebert et al. (2021); Zhang et al. (2020)
Freedom from Discrimination	Vogelsang and Borg (2019)
Performance Efficiency	
Resource Utilization	Siebert et al. (2021); Nakamichi et al. (2020)
Execution efficiency	Siebert et al. (2021)
Temporal Performance	Nakamichi et al. (2020)

Table 3.2: MLS qualities identified for model entity through literature

3.3 Quality Relevancy

In addition to the qualities that arise directly from the use case, literature reveals a number of common qualities in regards to MLS (see Table 3.3), some of which can be regarded as relevant and other do not hold any relevance for the specific use case. The qualities are taken from literature which covers ML in general to literature which covers scene text OCR. Only qualities that concern the model will be looked at, as the model is the focus of this thesis. The qualities may however be influenced by other entities.

The appropriateness quality refers to the ability to perform the type of task that is required by the use case (Siebert et al., 2021; Nakamichi et al., 2020). For this thesis this applies to scene text OCR models. Additionally, the properties which are derived from the use case (see Table 3.1), can be grouped under this quality.

Relevant	**Irrelevant**
Appropriateness	Fairness
Performance	Interpretability
Robustness	Reusability
Performance efficiency	

Table 3.3: Condensed Qualities for model entity

'An ML model is performant if it operates as expected according to a measure (or set of measures) that captures relevant characteristics of the model output' (Ashmore et al., 2021). For the performance quality, a measure is chosen depending on the type of task to be solved (Siebert et al., 2021). The F-Score is an example for a metric that is used to compare different models Chen et al. (2021); Long et al. (2021). Performance is usually measured with a test dataset that is independent from training and validating a model in order to approximate the generalization performance Goodfellow et al. (2016); Nakamichi et al. (2020).

The robustness of a model concerns environmental uncertainty Ashmore et al. (2021). Due to the uncontrolled environment in the practical aspect of taking the images on-site beneficial image properties can not be guaranteed (Chen et al., 2021). Robust text extraction can be influenced by factors such as complex backgrounds, text form (text rotation, font variability, arrangement), image noise (lighting conditions, blur, interference and low resolution) and access (perspective, shape of text) (Oyedotun et al., 2015; Ghosh et al., 2017; Chen et al., 2021). Therefore, these properties have to be accounted for when determining the viability for an approach. Some of these factors do not change the expected prediction (noise), others do (text form)Hu et al. (2020a). An example for bad image quality in regards to OCR can be seen in figure 3.1(b).

Performance efficiency addresses time and resource utilization when the model is in use. This does not involve the training phase but the execution or prediction (Siebert et al., 2021). The efficiency refers to low latency needs and to minimizing resource needs such as memory usage or power consumption (Nakamichi et al., 2020; Siebert et al., 2021; Sourvanos and Tsatiris, 2018). This quality is especially important for usage on mobile devices in conjunction with DNN (Sourvanos and Tsatiris, 2018; Niu et al., 2019). Note that performance efficiency is heaviliy influenced by the infrastrucure (Nakamichi et al., 2020; Siebert et al., 2021). Because the efficiency needs fall mostly on the model, it is categorized as such and thus deemed relevant in the scope of this thesis.

The first quality often found in research that is not relevant for the use case is fairness. A fair model is free from discrimination bias. For ML this is can be a big problem, since discrimination can not only be influenced through explicit programming in terms of the model but also through implicit knowledge from the data (Vogelsang and Borg, 2019). For the use case however no relevance is attached. The model can either recognize the text or it fails the task.

The interpretability of a model helps to justify the output (Ashmore et al., 2021). The interpretability is twofold: explain what the model has learned, explain how a model given the input comes to the output (Vogelsang and Borg, 2019). This can be challenging for two reasons. ML models used can be complex in terms of size and structure (Ashmore et al., 2021). Modular processing pipelines are continuously replaced with end-to-end models which facilitates the tradeoff between interpretability and performance Arpteg et al. (2018).

Another quality for a ML model refers to how well a model intended for one task can be reused for another related task. This can be beneficial because transfer learning can speed up the training, thus reducing training cost (Ashmore et al., 2021). Reusability is not relavant in the scope of this work as it targets the training phase of the ML lifecycle.

Chapter 4

Technique Overview

Show pipelines and notable advances along with properties

4.1 Different pipeline Frameworks

Scene Text Detection and Recognition Long et al. (2021) Deep learning → frees researchers from hand-crafting features, simplifies pipeline, improves performance

Difference: two-part pipeline vs end-to-end

- two-part / Detection & Recognition: detector passes cropped image to recognizer
- end-to-end: detector passes cropped feature maps to recognizer → end-to-end training

Text detection and localization can be generalized under object detection — generally: one-staged or two staged methods scene text detection algorithms often inspired bz object detectors — difference: text is homogeneous as a whole and characterized bz its locality stages of development

- early DL approaches: long and slow pipelines, design methodology is bottom up
- methods inspired by object detection:
  - modifying region proposal and bounding box regression modules to localize text directliy
  - consist of stacked convolutional layers that encode image into feature maps

  - spatial location at the feature maps corresponds to region of input image
  - feature maps are fed into classifier for prediction of existance and localization

  see p.5 / p.165 for specific papers

- Methods based on Sub-text components
  - any part of a text insance is still text → only predict sub-text components and then assemble into a text instance
  - use NN to predict local attributes or segments → postprocessing for re-construction
  - in comparison to early DL-approaches: rely more on NN and shorter pipelines
  - different approaches
    * pixel-level: learn dense prediction map — does pixel belong to any text instances
    * component-level: predict local region of text instance (overlapping one or more characters)
    * character-level: learn segmentation map for character centers and links between them, centers and links predicted in form of gaussian heat map, problem: requires iterative weak supervision, but real-world datasets rarely equipped with character-level labels

    sub-text components:
    better flexibility and generalization over shapes and aspect ratios
    drawback: module or post-processing step used to group segments into text instances may be vulnerable to noise and the efficiency

Recognition — Text transcribing and converting into linguistic symbols use CNNs to encode images into feature space different approaches in text content decoding module: challenge: represent oriented characters and curved text that are distributed over a 2-dimensional space (rather than 1-dim/horizontal) in order to fit decoding modules (whose decodes require 1-dimensional inputs)

- Connectionist Temporal Classification (CTC)
  - input images are viewed as sequence of vertical pixel frames

  - network outputs per-frame prediciton $\rightarrow$ probability distribution of label types for each frame
  - CTC-rule is applied to edit per-frame prediction to a text string
  - training end-to-end: sum of negative log propability of all possible per-frame predictions that generate target sequence by CTC-rules
  - less dependant on language models and has better character to pixel alignment

- Encoder-Decoder Framework
  - encoder RNN reads input sequence and passes final latent state to decoder RNN which generates output in auto-regressive way
  - often combined with attention mechanism
  - decoder is an implicit language model: can incorporate more linguistic priors

- adaptions for irregular text recognition
  - Spatial Transformer Networks (predict text bounding polyglons with fc-layers for thin-plate-spline transformations to rectify irregular input into more canonical form) $\rightarrow$ Sequence Recognition Network
  - ...

- other methods
  - perform word recognition by classifying image into pre-defined set of vocabulary
  - improve occlusion cases: transformer-based semantic reasoning module

evaluation of recognition methods falls behind the time robustness of recognition when cropped with slightly differend bounding box is seldom verified

End-to-end system also known as text spotting systems, profiting from idea of designing differentiable computation graphs

- Two-step pipeline:
  - recent work goes away from character level and towards word or line level
  - first generate proposal using text detection model, thenrecognize using text recognition model

  - detected words are cropped from the image → detection and recognition are two separate steps

- Two-stage pipeline
  - more recent
  - cropp feature maps instead of images and feed to recognition modules

- One-Stage Pipeline:
  - predict character, text bounding boxes and character type segmentation maps in parallel
  - text bb are used to group character boxes to form final word transcription results

Challenges in input preprocessing for mobile OCR applications Sourvanos and Tsatiris (2018) from 2018 → outdated

- Acquistion: obtaining image — digitization, binarization, compression
- Preprocessing: enhancing image quality — noise removal, skew removal, thinning, morphological operations
- Segmentation: separating structural elements — implicit and explicit segmentation
- Feature extraction: generating salient features — geometrical, statistical
- Classification: categorizing individual characters to their respective classes — clustering, neural networks, bayesian models, etc.
- Post-processing: improving and filtering — contextual approaches, multiple classifiers, dictionary based approaches

Text Recognition in the Wild: A Survey (Chen et al., 2021) various stages of OCR:

- text localization: localize text components, group into candidate text regions with as little background as possible, DNN
- text verification: verify text candidate regions as text or non-text, filter false-positives, CNN

- text detection: determine whether text is present using localization and verification procedures, basis for end-to-end, can be regression or segmentation based
- text segmentation: most challenging, includes text line (splitting a region of multiple text lines into subregion of single text lines) and character segmentation (separating text instance into single characters, typically used in earlier approaches)
- text recognition: translates cropped text instance image into target string sequence, basis for end-to-end, DL encoder-decoder frameworks
- end-to-end-system: given scene text image → convert all text regions into target string sequences, includes detectoin, recognition and post-processing, can be seen as indipendent subproblems but also joint by sharing information

text enhancement: recover degraded text, improve text resolution, remove distortions, remove background → reduce difficulty of recognition

Cropped Scene Text Image Recognition

- Main categories: Segmentation-based — Segmentation-free

  Segmentation-based

  - Approach: locate position of character, apply chlassifier to recognize each character, group characters into text lines
  - substeps: image preprocessing, character segmentation, character recognition
  - lexicon methods: query time linearly depends on size of lexicon → impractical — solve: higher-order statistical language models
  - lexicon-free methods: leverage more data and more complex NN
  - Shortcommings: require accurate character detection (hard), fail to model contextual information beyond individual characters → poor word-level results during training

  Segmentation-free

  - approach: recognize text line as a whole and focus on mapping entire text instance image into target string sequence
  - stages:

  * Preprocessing: improve image quality
    background removal: separate text from complex backgrounds
    super-resolution: TextSR can output plausible high-resolution image
    rectification: remove distortion (perspective and curving shape) STN, TPS
  * Feature extraction: various CNN backbone networks (VGG, ResNet, Binary Conv)
    maps image to representation that reflects attributes relevant for OCR, while suppressing irrelevant features
    Deeper and more advanced extractor better, but higher performance cost
  * Sequence Modeling: RNN, CNN, Transformer
    bridge between visual features and predictions, capture contextual informtion within sequence of characters, helpful as it is more stable than treating each symbol independently
    often BiLSTM because of ability to capture long-range dependencies problem: computationally expensive, time consumine, gradient vanishing/exploding deep one dimensional CNN instead of BiLSTM
    transformer structure
  * Prediction stage: estimate target string from features
    CTC attention-based CRNN also an option but more $2017 - 2018$ Look into: aggregation cross-entropy functiona → replace CTC and attention mechanism

Other approaches

–

CTC

- for training RNNs to label unsegmented sequences directly
- transcription layer converts input features by CNNs or RNNs into target string sequence by calculating conditional probability
- efficiently sum over all possible input-output sequence alignments and allow classifier to be trained without any prior alignment between input and target sequences
- is adapted to forward-backward algorithm for efficient computation
- problem:

  - produces highly peaky and overconfident distributions → overfitting → regularization to enhance generalization and exploration capabilities
  - large computational cost for long text sequences
  - can hardly be applied to two-dimensional prediction problems (text distributed in a spatial structure)

Attention-Based

- for STR often combined with RNN structure
- learns alignment between input instance image and output text sequences by referring to the history of the target characters and the encoded feature vectors
- outperform CTC in decoding because of ability to focus in informative areas
- can be extended to complex 2D prediction problems
- higher accuracy on isolated word recognition but perform worse on sentence tasks when compared to CTC
- problems
  - attention modul for label alignment → storage and compuation
  - attention drift phenomenon: for long text sequences, attention mechanism is difficult to train from scratch owing to misalignment between input instance image and output text sequences
  - current research mainly focuses on few character categories (chinese might be bad)

End-to-end Systems

- directly convert all text regions into string sequences
- includes text detection, text recognition and postprocessing
- reasons for end-to-end
  - real-time and efficient
  - errors can accumulate in cascade way for detection and recognition, end-to-end can prevend accumulation during training

  - end-to-end system can jointly be optimized
  - easier to maintain and adapt to new domains
  - competitive performance with faster interference and smaller storage requirements

What is wrong with scene text recognition model comparison Baek et al. (2019) Four stages derived from existing STR-Models — **only recognition**

- Transformation: normalize input image → Spatial Transormer Network
  - transform input image X into $\tilde{X}$
  - if curved and tilted texts / other diverse shapes are forwarded unaltered, feature extraction needs to learn an invariant representation
  - thin-plate spline transformatino (variant of spatial transformation network) → smooth spline interpolation between set of fiducial points
- Feature extraction: map input image to representation that focuses on relevant attributes, while suppressing irrelevant features
  - use Convolutional Neural Networks (CNN) to abstract image $\tilde{X}$ to output visual feature map $V = \{v_i\}, i = i, \ldots, I$
    I is number of columns in feature map, each column has a corresponding distinguishable receptive field along the horizontal line of $\tilde{X}$
  - important architectures: VGG, RCNN, ResNet
- Sequence Modeling: capture contextual information within sequence of characters
  - reshape extracted featured to be a sequence of features V (each column)
  - use Bidirectional LSTM → sequence $H = Seq.(V)$
  - this stage is optional
- Prediction: estimate output character sequence
  - use H to predict a sequence of characters $Y = y_1, y_2, \ldots, y_n$
  - options: Connectionist temporal classification or attention-based sequence prediction

Tradeoff:

- accuracy-speed
- accuracy-memory

Some part for which datasets have which characteristics? see Long et al. (2021)

4.2 Techniques/Modules for Improvement

text enhancement: Chen et al. (2021) model pruning: Niu et al. (2019) integer inference: Ignatov et al. (2019)

Chapter 5

Discussion

5.1 Analysis

try to find top $3-5$
Recognition

- Ability to cope with 2d text: CTC has problems, Attention/Encoder-Decoder based can be extended to work
- CTC prone to overfitting
- Attention has problems with long sequences

5.2 Reflection

Challenges DL (Arpteg et al., 2018) Note that actual experiments with models have to be done Problem: different papers have different components → Hardware, Platform, Source Code, Configuration → studies can't really be compared

'A major challenge in developing DL systems is the difficulties in estimating the results before a system has been trained and tested.' (Arpteg et al., 2018)

Threats to validity!

Problem: for practice all entities of MLS should be inspected

5.3 Outlook

What to do next: next steps Data Collection, Data Cleaning, Data Labeling, Model Training, Model Evaluation, Model Deployment, Model Monitoring Watanabe et al. (2019)

Chapter 6

Conclusion

Bibliography

Witold Abramowicz and Rafael Corchuelo, editors. *Business Information Systems: 22nd International Conference, BIS 2019, Seville, Spain, June 26–28, 2019, Proceedings, Part II*, volume 354 of *Lecture Notes in Business Information Processing*. Springer International Publishing, Cham, 2019. ISBN 978-3-030-20481-5 978-3-030-20482-2. doi: 10.1007/978-3-030-20482-2. URL http://link.springer.com/10.1007/978-3-030-20482-2.

Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, 1142:012012, November 2018. ISSN 1742-6588, 1742-6596. doi: 10.1088/1742-6596/1142/1/012012. URL https://iopscience.iop.org/article/10.1088/1742-6596/1142/1/012012.

Anders Arpteg, Björn Brinne, Luka Crnkovic-Friis, and Jan Bosch. Software Engineering Challenges of Deep Learning. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 50–59, August 2018. doi: 10.1109/SEAA.2018.00018.

Rob Ashmore, Radu Calinescu, and Colin Paterson. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. *ACM Computing Surveys*, 54(5):1–39, June 2021. ISSN 0360-0300, 1557-7341. doi: 10.1145/3453444. URL https://dl.acm.org/doi/10.1145/3453444.

Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4714–4722, Seoul, Korea (South), October 2019. IEEE. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.00481. URL https://ieeexplore.ieee.org/document/9010273/.

Nitin Kumar Chauhan and Krishna Singh. A Review on Conventional Machine Learning vs Deep Learning. In *2018 International Conference on Com-*

puting, Power and Communication Technologies (GUCON), pages 347–352, September 2018. doi: 10.1109/GUCON.2018.8675097.

Lei Chen and Shaobin Li. Improvement Research and Application of Text Recognition Algorithm Based on CRNN. In *Proceedings of the 2018 International Conference on Signal Processing and Machine Learning - SPML '18*, pages 166–170, Shanghai, China, 2018. ACM Press. ISBN 978-1-4503-6605-2. doi: 10.1145/3297067.3297073. URL `http://dl.acm.org/citation.cfm?doid=3297067.3297073`.

Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text Recognition in the Wild: A Survey. *ACM Computing Surveys*, 54(2): 1–35, April 2021. ISSN 0360-0300, 1557-7341. doi: 10.1145/3440756. URL `https://dl.acm.org/doi/10.1145/3440756`.

Suman K. Ghosh, Ernest Valveny, and Andrew D. Bagdanov. Visual Attention Models for Scene Text Recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 943–948, November 2017. doi: 10.1109/ICDAR.2017.158. ISSN: 2379-2140.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, 2016. ISBN 978-0-262-03561-3.

Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Beijing ; Boston, first edition edition, 2017. ISBN 978-1-4919-6229-9. OCLC: ocn953432302.

Yaoshiang Ho and Samuel Wookey. The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. *IEEE Access*, 8:4806–4813, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2962617. Conference Name: IEEE Access.

Boyue Caroline Hu, Rick Salay, Krzysztof Czarnecki, Mona Rahimi, Gehan Selim, and Marsha Chechik. Towards Requirements Specification for Machine-learned Perception Based on Human Performance. In *2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 48–51, September 2020a. doi: 10.1109/AIRE51212.2020.00014.

Wenyang Hu, Xiaocong Cai, Jun Hou, Shuai Yi, and Zhiping Lin. GTC: Guided Training of CTC Towards Efficient and Accurate Scene Text Recognition. *arXiv:2002.01276 [cs, eess]*, February 2020b. URL `http://arxiv.org/abs/2002.01276`. arXiv: 2002.01276.

Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. AI Benchmark: All About Deep Learning on Smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635, October 2019. doi: 10.1109/ICCVW.2019.00447. ISSN: 2473-9944.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, editors. *An introduction to statistical learning: with applications in R*. Number 103 in Springer texts in statistics. Springer, New York, 2013. ISBN 978-1-4614-7137-0. OCLC: ocn828488009.

Shangbang Long, Xin He, and Cong Yao. Scene Text Detection and Recognition: The Deep Learning Era. *International Journal of Computer Vision*, 129(1):161–184, January 2021. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-020-01369-0. URL `https://link.springer.com/10.1007/s11263-020-01369-0`.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill, New York, 1997. ISBN 978-0-07-042807-2.

Koji Nakamichi, Kyoko Ohashi, Isao Namba, Rieko Yamamoto, Mikio Aoyama, Lisa Joeckel, Julien Siebert, and Jens Heidrich. Requirements-Driven Method to Determine Quality Characteristics and Measurements for Machine Learning Software and Its Evaluation. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 260–270, August 2020. doi: 10.1109/RE48521.2020.00036. ISSN: 2332-6441.

Wei Niu, Xiaolong Ma, Yanzhi Wang, and Bin Ren. 26ms Inference Time for ResNet-50: Towards Real-Time Execution of all DNNs on Smartphone. *arXiv:1905.00571 [cs, stat]*, May 2019. URL `http://arxiv.org/abs/1905.00571`. arXiv: 1905.00571.

Oyebade K. Oyedotun, Ebenezer O. Olaniyi, and Adnan Khashman. Deep Learning in Character Recognition Considering Pattern Invariance Constraints. *International Journal of Intelligent Systems and Applications*, 7(7): 1–10, June 2015. ISSN 2074904X, 20749058. doi: 10.5815/ijisa.2015.07.01. URL `http://www.mecs-press.org/ijisa/ijisa-v7-n7/v7n7-1.html`.

Moacir Antonelli Ponti, Leonardo Sampaio Ferraz Ribeiro, Tiago Santana Nazare, Tu Bui, and John Collomosse. Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask. In *2017*

30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), pages 17–41, October 2017. doi: 10.1109/SIBGRAPI-T.2017.12. ISSN: 2474-0705.

Ajay Shrestha and Ausif Mahmood. Review of Deep Learning Algorithms and Architectures. *IEEE Access*, 7:53040–53065, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2912200. Conference Name: IEEE Access.

Julien Siebert, Lisa Joeckel, Jens Heidrich, Adam Trendowicz, Koji Nakamichi, Kyoko Ohashi, Isao Namba, Rieko Yamamoto, and Mikio Aoyama. Construction of a quality model for machine learning systems. *Software Quality Journal*, June 2021. ISSN 0963-9314, 1573-1367. doi: 10.1007/s11219-021-09557-y. URL `https://link.springer.com/10.1007/s11219-021-09557-y`.

Hannah Snyder. Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, 104:333–339, November 2019. ISSN 0148-2963. doi: 10.1016/j.jbusres.2019.07.039. URL `http://www.sciencedirect.com/science/article/pii/S0148296319304564`.

Nikolaos Sourvanos and Georgios Tsatiris. Challenges in Input Preprocessing for Mobile OCR Applications: A Realistic Testing Scenario. In *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–5, July 2018. doi: 10.1109/IISA.2018.8633688.

Richard J. Torraco. Writing Integrative Literature Reviews: Guidelines and Examples. *Human Resource Development Review*, 4(3):356–367, September 2005. ISSN 1534-4843. doi: 10.1177/1534484305278283. URL `https://doi.org/10.1177/1534484305278283`. Publisher: SAGE Publications.

Andreas Vogelsang and Markus Borg. Requirements Engineering for Machine Learning: Perspectives from Data Scientists. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 245–251, September 2019. doi: 10.1109/REW.2019.00050.

Yasuhiro Watanabe, Hironori Washizaki, Kazunori Sakamoto, Daisuke Saito, Kiyoshi Honda, Naohiko Tsuda, Yoshiaki Fukazawa, and Nobukazu Yoshioka. Preliminary Systematic Literature Review of Machine Learning System Development Process. *arXiv:1910.05528 [cs]*, October 2019. URL `http://arxiv.org/abs/1910.05528`. arXiv: 1910.05528.

Linjie Xing, Zhi Tian, Weilin Huang, and Matthew Scott. Convolutional Character Networks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9125–9135, Seoul, Korea (South), October 2019.

IEEE. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.00922. URL https://ieeexplore.ieee.org/document/9010699/.

Xue Yang, Xiaojiang Yang, Jirui Yang, Qi Ming, Wentao Wang, Qi Tian, and Junchi Yan. Learning High-Precision Bounding Box for Rotated Object Detection via Kullback-Leibler Divergence. *arXiv:2106.01883 [cs]*, October 2021. URL http://arxiv.org/abs/2106.01883. arXiv: 2106.01883.

Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering*, pages 1–1, 2020. ISSN 1939-3520. doi: 10.1109/TSE.2019.2962027. Conference Name: IEEE Transactions on Software Engineering.

Zhenyao Zhao, Min Jiang, Shihui Guo, Zhenzhong Wang, Fei Chao, and Kay Chen Tan. Improving Deep Learning based Optical Character Recognition via Neural Architecture Search. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7, July 2020. doi: 10.1109/CEC48606.2020.9185798.

Didar Zowghi, Zhi Jin, Simone Diniz Junqueira Barbosa, Phoebe Chen, Alfredo Cuzzocrea, Xiaoyong Du, Joaquim Filipe, Orhun Kara, Igor Kotenko, Krishna M. Sivalingam, Dominik Ślęzak, Takashi Washio, and Xiaokang Yang, editors. *Requirements Engineering*, volume 432 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-43609-7 978-3-662-43610-3. doi: 10.1007/978-3-662-43610-3. URL http://link.springer.com/10.1007/978-3-662-43610-3.

Appendix A

Litaratur Qualities

Qualitiy	Source(s)
Relevancy	Ashmore et al. (2021)
Currentness	Siebert et al. (2021)
Completeness	Ashmore et al. (2021); Vogelsang and Borg (2019); Siebert et al. (2021)
Balancedness	Ashmore et al. (2021); Siebert et al. (2021)
Consistency	Vogelsang and Borg (2019)
Intra-Consistency	Siebert et al. (2021)
Inter-Consistency	Siebert et al. (2021)
Accuracy	Ashmore et al. (2021)
Absence of bias	Siebert et al. (2021)
Correctness	Vogelsang and Borg (2019)
Data Representativeness	Nakamichi et al. (2020); Siebert et al. (2021)
Suitability of Training Data	Nakamichi et al. (2020)
Test Dataset Creating Appropriateness	Nakamichi et al. (2020)
Independence of Train and Test Data	Nakamichi et al. (2020); Siebert et al. (2021)

Table A.1: Machine Learning System qualities identified for data entity through literature

Qualitiy	Source(s)
Capacity of Data Storage	Nakamichi et al. (2020)
Infrastructure suitability	Siebert et al. (2021)
Deployment Fit-for-Purpose	Ashmore et al. (2021)
Training Process Appropriateness	Nakamichi et al. (2020)
Training efficiency	Siebert et al. (2021)

Table A.2: Machine Learning System qualities identified for infrastrucure entity through literature

Qualitiy	**Source(s)**
Coverage of Usage Environment	Nakamichi et al. (2020)
Coverage of Operation Environment	Nakamichi et al. (2020)
Scope compliance	Siebert et al. (2021)
Social impact	Siebert et al. (2021)
Environmental Impact of training process	Siebert et al. (2021)
Contextual Relevancy	Ashmore et al. (2021)

Table A.3: Machine Learning System qualities identified for environment entity through literature

Qualitiy	**Source(s)**
Suitability of Input Data Quality Maintenance	Nakamichi et al. (2020)
Quality Maintenance for Test Data Appropriateness	Nakamichi et al. (2020)
Security and Privacy Assurance	Nakamichi et al. (2020); Zhang et al. (2020)
Troubleshooting	Arpteg et al. (2018)
Easiness of Resource Update	Nakamichi et al. (2020)
Easiness of Software Update	Nakamichi et al. (2020)
Easiness of System Status Analysis	Nakamichi et al. (2020)
Runtime correctness	Siebert et al. (2021)
Legal and Regularity Requirements	Vogelsang and Borg (2019)
Effectiveness of output supervision	Siebert et al. (2021)
Efficiency of output supervision	Siebert et al. (2021)
Appropriateness of Operation Maintenance	Nakamichi et al. (2020)
Deployment Tolerability	Ashmore et al. (2021)
Deployment Adaptability	Ashmore et al. (2021)

Table A.4: Machine Learning System qualities identified for system entity through literature