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

In accordance with §16 para. 10 APO in conjunction with §35 para. 7 RaPO:
I hereby declare that I have written this bachelor thesis independently, have not submitted it for examination purposes elsewhere, have not used any sources or aids other than those indicated, and have marked verbatim and analogous quotations as such.

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

AED Average Edit Distance

AP Average Precision

CNN Convolutional Neural Network

DL Deep Learning

DNN Deep Neural Network

GD Gradient Descent

IOU Intersection-over-Union

LSTM Long Short Term Memory

ML Machine Learning

MLP Multi Layer Perceptron

MLS Machine Learning System

NED Normalized Edit Distance

NN Neural Network

OCR Optical Character Recognition

RNN Recurrent Neural Network

STD Scene Text Detection

STR Scene Text Recognition

STS Scene Text Spotting

Notation

Calculus

$\frac{\delta \mathbf{y}}{\delta \mathbf{x}}$ Jacobian matrix $\mathbf{J} \in \mathbb{R}^{n \times m}$ of $f : \mathbb{R}^n \to \mathbb{R}^m$

$\frac{\delta y}{\delta x}$ Partial derivative of y with respect to x

$\frac{dy}{dx}$ Derivative of y with respect to x

$\nabla_{\mathbf{x}} y$ or $\frac{\delta y}{\delta \mathbf{x}}$ Gradient of y with respect to $\mathbf{x}$

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

$\mathbf{A}$ Tensor

$\mathbf{a}$ Vector

Other

$\|\mathbf{x}\|$ L^2 norm of $\mathbf{x}$

$\mathbb{R}$ Real numbers

$f(\mathbf{x}; \boldsymbol{\theta})$ A function of $\mathbf{x}$ 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, Cai, Hou, Yi and Lin, 2020; Chen et al., 2021; Baek et al., 2019). In these conditions OCR is known as Scene Text Spotting (STS) (Long et al., 2021).

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 create an overview over

possible DL techniques that facilitates finding a solution for the process of digitization. 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 taken in real world conditions?

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, Cai, Hou, Yi and Lin, 2020)?

It is difficult to assess how well a DL approach performs before it has been implemented and tested on the specific problem or representative dataset (Arpteg et al., 2018). This justifies the need to create an overview rather than pointing out a single approach which is deemed the most promissing. In order to create the overview the necessary steps in the process of STS need to be highlighted, from preprocessing to classifying the identified text (Long et al., 2021; Sourvanos and Tsatiris, 2018). The ways in which the respective issues for the steps are solved need to be identified from literature, listed and explained alongside.

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. Goodfellow et al. (2016) states: 'Nearly all deep learning algorithms can be described as particular instances of a fairly simple recipe: combine a specification of a dataset, a cost function, an optimization procedure and a model.' Other aspects such as data analysis, implementation, training, deployment and maintenance of a solution in a production environment shall not be performed. Based on this theses, further further verification, implementation and testing can then be performed. The overview and subsequent analysis thereof creates a foundation for finding the right solution, it does however not contain any claims about the degree of goodness or about the certainty of solving the given problem.

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 the Google Scholar database. A criterion for further examination is an appropriate amount of citations for the piece of literature in question. Additionally, literature is selected through citations for and by literature which has already been identified as important. All research after 2018 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 criterion 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.

The identified literature is synthesized into an overview over the most common approaches. This includes listing important factors for DL such as the type and size of NN or the used objective function. The overview will be organized according to a taxonomy which facilitates the clarity and comprehensibility.

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. Chapter 2 lays the foundation for later chapters. This includes general principles of DL and by extension ML but also of STS. 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 is 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 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 mathematical notation is omitted if it does not aid the understanding of the reader.

2.1 Machine Learning

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 obtaining 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 L (Goodfellow et al., 2016). The MSE introduced earlier can be used for Linear Regression (see Equation 2.3). This

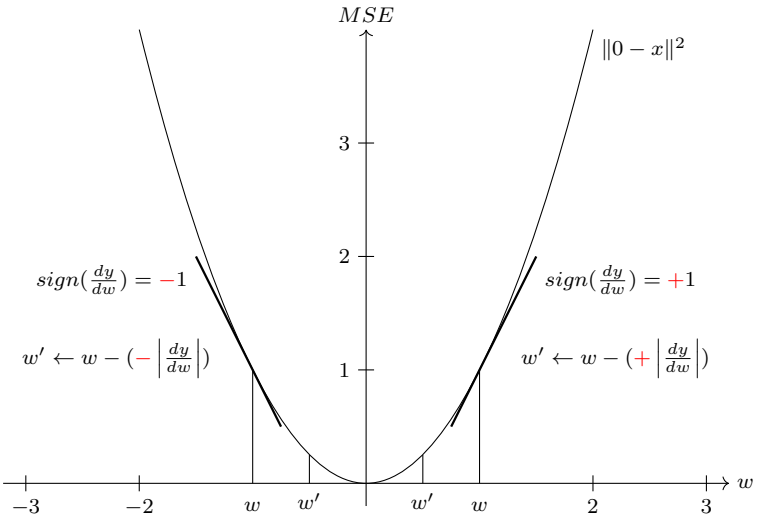


Figure 2.1: Gradient descent for 1-dimensional objective function (Goodfellow et al., 2016)

objective function has properties which make it suitable for models which have linear output (Goodfellow et al., 2016).

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

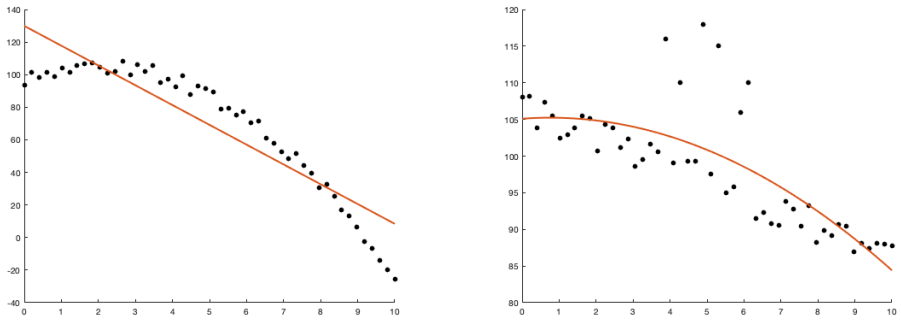
Note that for minimization the MSE is a function of $\mathbf{w}, b$ and not of $\mathbf{x}$, in terms of predicting a value the MSE is a function of $\mathbf{x}$ parametrized by $\mathbf{w}, b$ (see Equation 2.3). In Equation 2.1 $\mathbf{w}$,b are parameters that have to be learned in order to minimize 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 training 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 used to prevent overfitting (Regularization) (Goodfellow et al., 2016).



(a) Linear Regression underfit (b) Polynomial Regression overfit

Figure 2.2: Regression with linear and polynomial model

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 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 Networks (RNNs) or Transformers) (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 $\hat{y}$ or $\hat{\mathbf{y}}$ 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). The number of layers d is refered to as depths, whereas the dimensionality of those layers is called the width w (Goodfellow et al., 2016). A neuron, the basic building block of NNs, re-

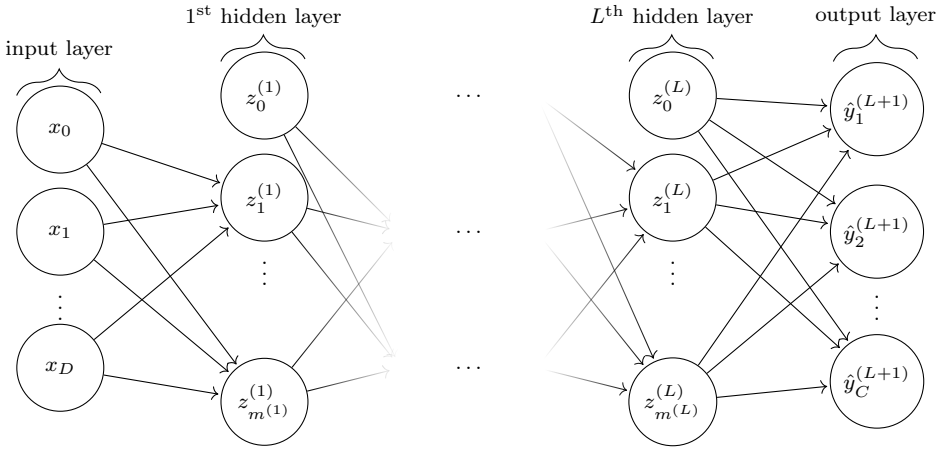


Figure 2.3: Network graph of a $(L+1)$-layer perceptron with D input units and C output units. The l^{th} hidden layer contains $m^{(l)}$ hidden units (Chauhan and Singh, 2018; Goodfellow et al., 2016).

ceives 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 fed into an activation function g which introduces nonlinearity, to allow more complicated transformations of information and representation (Goodfellow et al., 2016).

$$f(\mathbf{x};\boldsymbol{\theta}) = g(\boldsymbol{\theta}\mathbf{x}) = \mathbf{z} \tag{2.6}$$

Here f denotes the function which is performed by a layer of neurons (linearity + activation). The parameters of the individual neurons are grouped together to $\boldsymbol{\theta}$ ($\boldsymbol{\theta}_{:,0}$ equals 1 for the bias term). Popular activation functions include ReLU, tanh, sigmoid (σ) and softmax (Shrestha and Mahmood, 2019).

$$ReLU(x) = \max(0, x) \tag{2.7}$$

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp x + \exp -x} \tag{2.8}$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.9}$$

$$softmax(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.10}$$

While ReLU is the prevalent function for hidden layers, sigmoid (softmax) activation functions, used for the output layer, are used to generate a bernoulli (multinouli) distribution which is useful for classification tasks (Goodfellow et al., 2016). tanh is often used in RNNs like in Sherstinsky (2020); Greff et al. (2017). 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 layers and their neurons, the process of which is called forwardpropagation (Ponti et al., 2017; Goodfellow et al., 2016).

$$\hat{y} = f(\dots f(f(\mathbf{x}; \boldsymbol{\theta}^{(1)}); \boldsymbol{\theta}^{(2)}) \dots; \boldsymbol{\theta}^{(d)}) \tag{2.11}$$

$\boldsymbol{\theta}^{(i)}$ in Equation 2.11 stands for the parameters in layer i with $\boldsymbol{\theta}^{(i)}_{j,:}$ being the parameters the j-th neuron in that layer (Goodfellow et al., 2016). The forwardpropagation can also be described by a computational graph (see Figure 2.3) (Goodfellow et al., 2016).

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. The training error is calculated through the objective function and is propagated in conjunction with the output of forwardpropagation on each neuron (Goodfellow et al., 2016). For this the chain rule of calculus can be used to modularly, recursively propagate the loss backwards to use GD (see Figure 2.4). The upstream gradient that is coming from neurons in the next

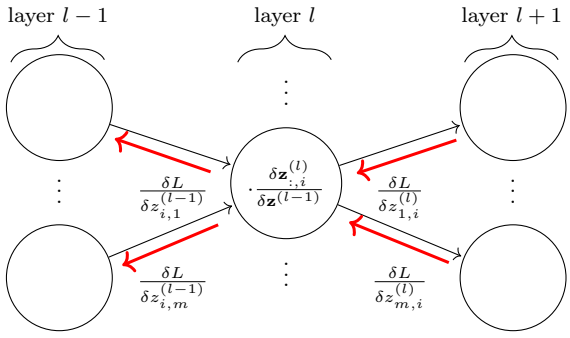


Figure 2.4: Reverse traversing the network's computation graph, $\cdot \frac{\delta \mathbf{z}^{(l)}_{:,i}}{\delta \mathbf{w}^{(l)}_i}$ is used for updating the neurons parameters

layer is multiplied with the jacobian matrix of the current neuron to produce the downstream gradient that is then used by the preceding layer (Boué, 2018; Goodfellow et al., 2016).

$$\frac{\delta L}{\delta \mathbf{w}} = \frac{\delta L}{\delta \mathbf{z}} \frac{\delta \mathbf{z}}{\delta \mathbf{w}} \tag{2.12}$$

$$\frac{\delta L}{\delta \mathbf{x}} = \frac{\delta L}{\delta \mathbf{z}} \frac{\delta \mathbf{z}}{\delta \mathbf{x}} \tag{2.13}$$

The result of Equation 2.12 is used to update the neuron's weights $\mathbf{w}$ while the result of Equation 2.13 is used for further propagation (Boué, 2018). This calculation is performed until the first layer of the computational graph is reached (Goodfellow et al., 2016). Note that the algorithm can be performed with tensors of arbitrary dimensionality (Goodfellow et al., 2016).

2.3 Convoluational Neural Nets

Convolutional Neural Networks (CNNs) are a type of NN that is also acyclic, like MLPs (Chauhan and Singh, 2018). CNNs are specialized to process a grid of values $\mathbf{X}$ like an image (Goodfellow et al., 2016). CNNs are extensively used in computer vision (Chauhan and Singh, 2018). They consist of a variety of components: fully connected layer, activation function, convolutional layer, pooling layer (Chauhan and Singh, 2018; Ponti et al., 2017). The fully connected layers are the layers that make up MLPs (Ponti et al., 2017).

A convolutional layer has multiple filters which consist of multiple kernels (Chauhan and Singh, 2018). For multi layer input, with d so called channels, a filter has the same amount of kernels as there are channels (d) (Ponti et al., 2017). A kernel is a $n \times n$ square matrix made up of learnable parameters, so a filter is a tensor $n \times n \times d$. The convolution operation is the

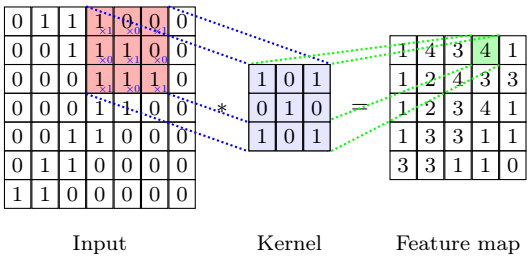


Figure 2.5: Convolution operation of a single kernel (Chauhan and Singh, 2018)

elementwise multiplication between the filter and overlapping $n \times n$ subspace

of the input (see Figure 2.5) (Ponti et al., 2017). The convolution operation is performed for every space in the input, spaces can be skipped if stride is introduced (Ponti et al., 2017). Often zero-padding is used to preserve the dimensions height and width between input and output of the layer (Ponti et al., 2017). The number of filters a convolutional layer applies is equal to the output channels that the layer has which are often called feature maps (Ponti et al., 2017). The result of a convolutional layer is usually fed into a activation function to introduce nonlinearity like with fully connected layers. The activation is performed on every element in the output und preserves the dimensionality (Ponti et al., 2017). In the explained scenario, the filter is slid across a 2d-surface to perform convolution, note that this can be restricted to 1d (for e.g. audio), or extended to 3d (for e.g. CT scans) (Goodfellow et al., 2016).

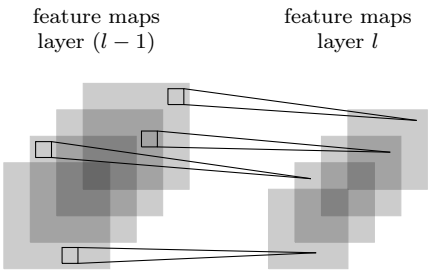


Figure 2.6: Pooling layer

Pooling layers are used to reduce the spatial dimension (i.e. downsampling) of feature maps (Ponti et al., 2017). Pooling layers preserve the number of channels, as the operation is performed to each channel separately (see Figure 2.7) (Chauhan and Singh, 2018). Much like with convolutions (in 2d

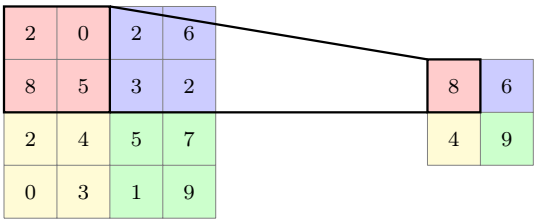


Figure 2.7: 2×2 max pooling operation with stride 2 (Chauhan and Singh, 2018)

scenario), the pooling operation is slid across the height and weight dimen-

sions of the channels (possibly with stride) and the pooling operation is performed (Ponti et al., 2017; Chauhan and Singh, 2018). The most popular kind of pooling is maxpooling (Ponti et al., 2017). For the operation only the maximum value of the current subspace of the current channel is taken (Chauhan and Singh, 2018).

When it comes to deep CNNs, the problem of vanishing/exploding gradients occurs during backpropagation. ResNet introduced residual blocks with skip connections which pass the gradient to later layers to bypass vanishing/exploding (He et al., 2015). ResNet and other popular CNN architectures

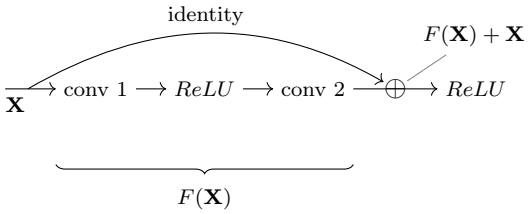


Figure 2.8: Residual block module with skip connection (He et al., 2015)

are explained in Chapter 4.

2.4 Recurrent Neural Nets

RNNs are another popular category of NN which are used for processing sequential input, like text and speech (Chauhan and Singh, 2018). Figure 2.9 shows a simple RNN. The defining element is the recurrent connection from

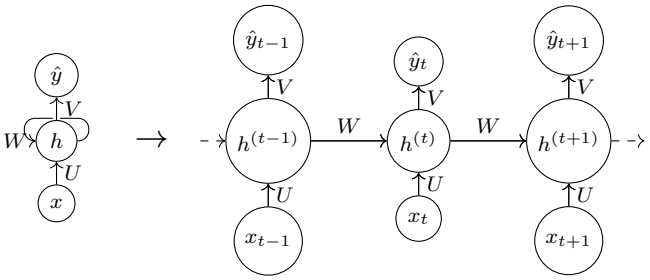


Figure 2.9: Recurrent neural net unrolling (Goodfellow et al., 2016)

node h to itself (Goodfellow et al., 2016). A sequence of inputs $\mathbf{x}$ is iteratively fed into the RNN. The current layer of the network takes $x^{(t)}$ and $h^{(t-1)}$ and produces $h^{(t)}$ (see Equation 2.14). Additionally, $h^{(t)}$ is used to calculate the

output $\hat{y}^{(t)}$ (see Equation 2.15) and it is handed to the next layer (Goodfellow et al., 2016).

$$h^{(t)} = \tanh(b + Wh^{(t-1)} + Ux^{(t)}) \tag{2.14}$$

$$\hat{y}^{(t)} = softmax(c + Vh^{(t)}) \tag{2.15}$$

Note that the connection(s) between hidden layers can differ depending on the type of RNN used (Goodfellow et al., 2016). During an execution run, all unrolled layers share the same weights (U, V, W) (Chauhan and Singh, 2018). The example RNN produced a output sequence the same length as the input sequence, however, this does not have to be the case for RNNs (Goodfellow et al., 2016). Both the input and the output can either be a sequence of variable length or a vector of fixed length (Goodfellow et al., 2016). Optimization of the parameters of RNNs is performed with backpropagation through time (Sherstinsky, 2020). That is because the unrolled network can be seen as a deep feed forward network (Chauhan and Singh, 2018).

The Long Short Term Memory (LSTM) network was introduced by Hochreiter and Schmidhuber (1997) and modified by Gers et al. (1999) to improve longer storing of information from earlier layers (Chauhan and Singh, 2018). The LSTM also improves upon the vanishing/exploding gradients problem associated with increasingly deep NNs (Sherstinsky, 2020). A so called cell is shown in Figure 2.10, it shows the basic structure which represents the recurrent building block of LSTMs (Goodfellow et al., 2016). The three gates

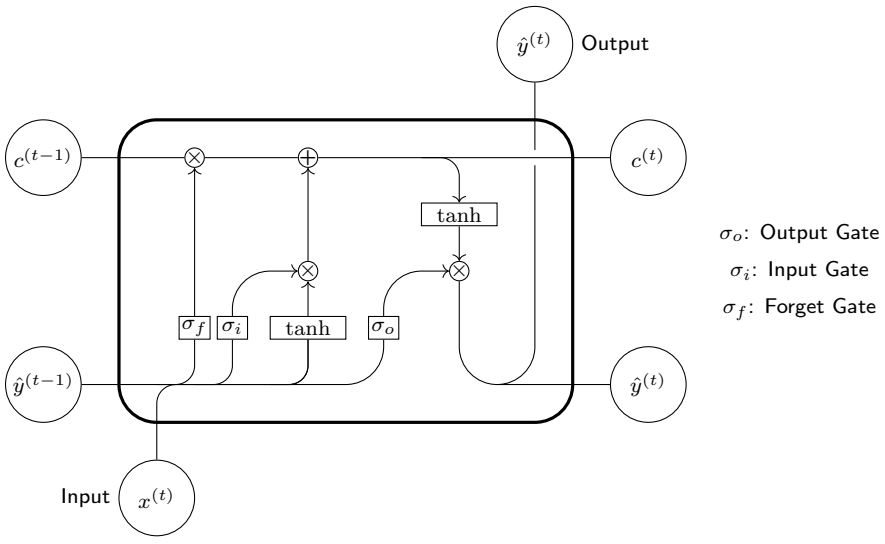


Figure 2.10: Long short term memory cell (Goodfellow et al., 2016; Yu et al., 2019)

(input, forget, output) help make the LSTM a widely used NN model. The gates calculate weights between 0 and 1 (thus the use of σ) that are used to

control the flow of information (Goodfellow et al., 2016). The forget gate is part of a self loop which helps to accumulate information longer, the input gate helps to focus on the relevant characteristics of the input information and the output gate removes irrelevant information for the output as well as the next cell (Goodfellow et al., 2016).

2.5 Transformers

Of the most popular basic strucures for NNs, the transformers are the newest innovation. The paper from Vaswani et al. (2017) introduced the transfomer architecture which is build for processing sequence data and Dosovitskiy et al. (2021) then introduced the vision transformer (ViT). Explaining the inner workings of transformers in detail exceeds the scope of this thesis. In the following the basic building blocks and their effects are described with the example of translating sentences. Figure 2.11 shows a transformer network at the abstraction level of components and their alignment to build the network. The network is made up of multiple multiple encoder and multiple decoder

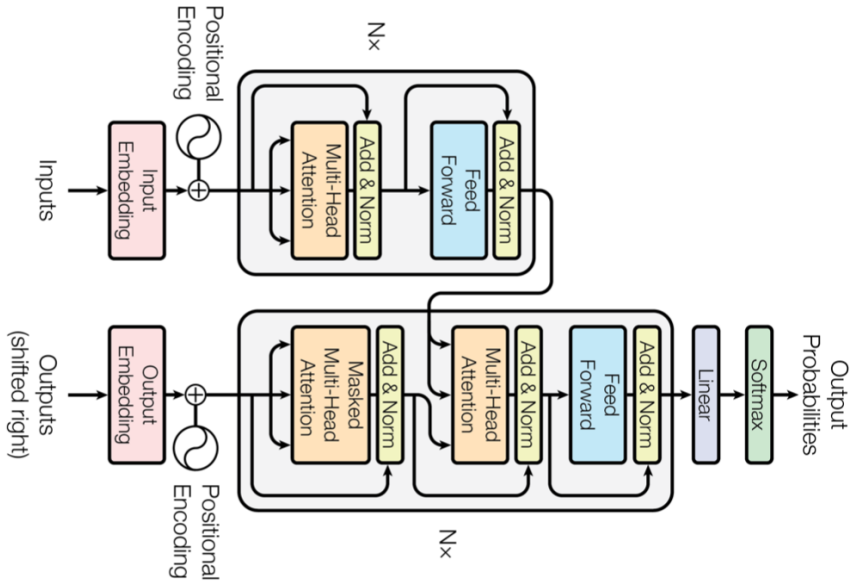


Figure 2.11: Transformer network (Vaswani et al., 2017)

blocks (Vaswani et al., 2017). The blocks are preceded by embedding and positional modules (Vaswani et al., 2017). The embedding modules map a word to a vector, also interpreted as a point in space, where similar words are close to each other (Vaswani et al., 2017). An example for a embedding

space is presented by Pennington et al. (2014). Because the placing of a word in a sentence determines different meaning or context, positional encoding is added to the word embedding (Vaswani et al., 2017). The encoder and decoder blocks contain multi-head attention modules and feedforward networks. They also contain skip connections and layer normalization which helps with training the network and to keep it stable (Vaswani et al., 2017). A multi-head attention block calculates which words (by now embedded) of the input are important for each other word. For n words the results are n vectors $\in \mathbb{R}^n$. The attention is calculated by a scaled dot product (see Figure 2.12) (Vaswani et al., 2017). This is calculated multiple times (hence the term multi-head)

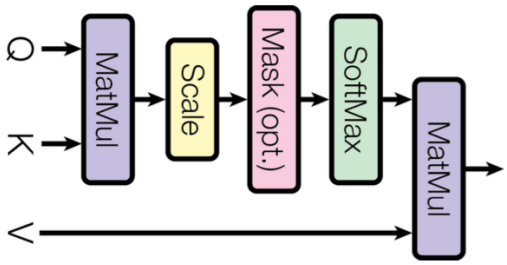


Figure 2.12: Scaled dot product used to compute attention vectors (Vaswani et al., 2017)

and the results are merged with weighted average, leading to an attention vector for each word in the sentence (Vaswani et al., 2017). Each encoder block has one multi-head attention module which calculates the needed attention between each word in the original language. Each decoder has two multi-head attention modules (Vaswani et al., 2017). One is masked (only already predicted words are used for calculation, others are masked/0), it calculates the attention between translated words. The second one calculates the attention between each original and translated word. Finally the feedforward layers help to transform the attention vectors into digestible form for the next encoder/decoder block (Vaswani et al., 2017). The result of the nx encoder blocks are n vectors (one for each word). They are fed into the second attention module for the nx decoder blocks (Vaswani et al., 2017). The result of the nx decoder blocks is fed into a linear layer and a softmax layer afterwards. This creates the prediction, for the next translated word. For the next iteration (to generate next translated word) the previous translated words are encoded like the original words and fed into the first attention module of the nx decode blocks. New translated words are generated with this iteration until until the end of sentence token is generated (Vaswani et al., 2017).

2.6 Scene Text Spotting

OCR is the concept of extracting typed, handwritten or printed text from an image (Zhao et al., 2020). Achieving satisfactory 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). The difficulties arise from diviersity and variability of text, complexity and interference from backgrounds and imperfect imaging conditions. In these conditions OCR is known as STS (Long et al., 2021). Before the advent of DL, researchers in the field had to hand-craft features (Long et al., 2021). DL automates the feature generation process with its representation and learning capabilities (Long et al., 2021; Goodfellow et al., 2016). Because of this, DL methods are the prefered tools for performing STS (Long et al., 2021). OCR and STS are often divided into two subcategories (Scene) Text Detection and (Scene) Text Recognition (Zhao et al., 2020; Long et al., 2021; Chen et al., 2021). For Scene Text Detection (STD) the task is to localize text instances in the image, whereas the Scene Text Recognition (STR) task is to recognize/categorize text from already cropped images (Chen et al., 2021). Note that a system which performs both STR and STD in one continuous pipeline are called end-to-end approaches (Chen et al., 2021).

To assess the performance of the developed approaches, the right evaluation metrics have too be used. The popular protocols Precision, Recall and the F_1-Score are used for comparison among approaches for STD (Long et al., 2021). The metrics are derived with values from the confusion matrix (see Tabel 2.1) (Davis and Goadrich, 2006).

		Ground Truth	
		positive	negative
Prediction	positive	True Positive	False Positive
	negative	False Negative	True Negative

Table 2.1: Confusion Matrix

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2.16}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.17}$$

$$F_1\text{-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.18}$$

The difference of metrics for the task manifests itself in the way the values of the confusion matrix are calculated (Long et al., 2021). Note that the

tradeoff between False Positives and Fales Negatives manifests itself in the Precision-versus-Recall curve (Su et al., 2015). F_1-Score is also refered to as the harmonic mean (between Precision and Recall) (He et al., 2018). STD differs mostly in the way the protocols match the prediction to the ground truth (Long et al., 2021). Detectors have multiple predictors which regress the placing and sizing of bounding boxes. More information on this will follow in Chapter 4. Matching is the process of assigning a bounding box prediction to the ground truth, like in e.g. Liu et al. (2016); Liao et al. (2018). The PASCAL approach defines the Intersection-over-Union (IOU) (see Equation 2.19).

$$IOU = \frac{\text{area of intersection between truth and prediction}}{\text{area of union between truth and prediction}} \tag{2.19}$$

For PASCAL, the prediction will be matched, if the IOU value is larger than a threshhold (Long et al., 2021). A match is considered a True Positive, the other values are assigned accordingly (Sun et al., 2019). Other evaluation approaches are mostly based on IOU, e.g. MSRA-TD 500 evaluates the rotation from the bounding box, compared to the truth in addition to the IOU threshhold (Long et al., 2021). Long et al. (2021) argues that researchers in the field of STD should consider Average Precision (AP) as the main evaluation protocol rather than F_1-Score. According to Su et al. (2015), AP can be considered the area under the Precision-versus-Recall curve. F_1-Score on the other hand only considers singular instances on that curve (Long et al., 2021) and is sensitive to the tradeoff while AP is invariant to it (Shi et al., 2017). For STR the evaluation can be based on character-level or word-level. There is no need to match ground truth to prediction, as the image is already cropped (Long et al., 2021). The equations 2.20 and 2.21 show metrics based on word level (Chen et al., 2021).

$$\text{Word Recognition Accuracy} = \frac{\text{correctly recognized words}}{\text{total words}} \tag{2.20}$$

$$\text{Word Error Rate} = 1 - \text{Word Recognition Accuracy} \tag{2.21}$$

An example for a character-based metric would be 1−NED where Normalized Edit Distance (NED) calculates the distance between prediction and ground truth (see Equation 2.22).

$$\text{NED} = \frac{1}{N} \sum_{i=1}^{N} \frac{D(s_i, \hat{s}_i)}{\max(l_i, \hat{l}_i)} \tag{2.22}$$

D denotes the Levenshtein distance, s denotes the text, l denotes the text length and N is the total number of text lines (Shi et al., 2017). For STR NED is used

over the whole dataset (Karatzas et al., 2013). STS is oriented to both STD and STR. The prediction has to be matched to the ground truth like for STD (Long et al., 2021). For comparing predictions and matched the respective ground truths that have been matched, NED is used (Chen et al., 2021). For end-to-end recognition (Karatzas et al., 2013, 2015), the main evaluation protocols that are used include Precision, Recall, F_1-Score and Average Edit Distance (AED) (Chen et al., 2021). A sample is considered a True positive if the NED distance between predictions and matched ground truths is equal to 0 (Sun et al., 2019) (on sample can have multiple text instances). AED is the sum of NED values devided by the number of pictures (Chen et al., 2021). Note that competitions often define their own variants of the metrics, e.g. He et al. (2018); Shi et al. (2017). Case sensitivity and matching criteria are examples for changing properties of metrics.

To compare approaches with these metrics benchmark datasets are used which have different characteristics. Table 2.2 lists a couple of influencial

Dataset (year)	**STD**	**STR**	**Text Orientation**	**Characteristics**
ICDAR (2013)	✓	✓	Horizontal	—
IIIT 5K-Word (2012)		✓	Horizontal	Cropped, variance in font, color, size and noise (Long et al., 2021)
ICDAR (2015)	✓	✓	Multi-oriented	Low resolution, small text instances (Liao et al., 2020)
MSRA-TD500 (2012)	✓		Multi-Oriented	Extreme aspect ratios (Liao et al., 2020)
ICDAR MLT (2017)	✓	✓	Curved	Multilingual (Long et al., 2021)
SCUT CTW1500 (2017)	✓		Curved	—
Total-Text (2017)	✓	✓	Curved	—

Table 2.2: Benchmark datasets and their properties

benchmark datasets along with their key properties. ICDAR (2013) references the Focused Scene Text dataset (Karatzas et al., 2013) and ICDAR 2015 to the Incidental Text Competition dataset (Karatzas et al., 2015). The second and third column indicate whether the dataset provides annotations for the tasks. The Text Orientation column specifies the most complicated orientation that is present in the dataset (Curved $\subset$ Multi-oriented $\subset$ Horizontal).

(a) ICDAR (2013) (Karatzas et al., 2013)

(b) ICDAR (2015) (Karatzas et al., 2015)

(c) ICDAR MLT (2017) (Nayef et al., 2017)

(d) IIIT 5k Word (2012) (Mishra et al., 2012)

(e) MSRA-TD500 (Cong Yao et al., 2012)

(f) Total Text (2017) (Ch'ng and Chan, 2017)

(g) SCUT CTW1500 (2017) (Yuliang et al., 2017)

Figure 2.13: Benchmark Data Set Examples

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

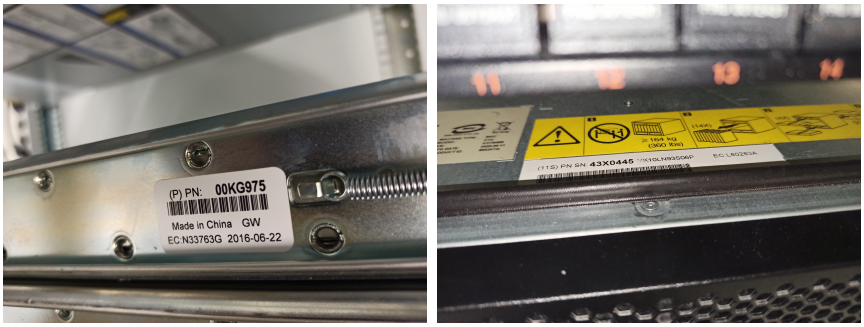
The problem can be depicted by a use case. This use case sets the foundation for determining requirements for an approach because qualities derive from the intended purpose of use (Siebert et al., 2021). Table 3.1 gives an overview over the relevant properties that can be derived from the use case. For 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 does not have to be oriented horizontally. However, it is not curved. 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 strings such



(a) Positive example (b) Negative example

Figure 3.1: Examples for label images

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. derived from the use case. The Table 3.2 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, Salay, Czarnecki, Rahimi, Selim and Chechik (2020); 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	Nakamichi et al. (2020)
Appropriateness	
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

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 3.2), some of which can be regarded as relevant and other do not hold any relevance for the specific use case (see Table 3.3). 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, Salay, Czarnecki, Rahimi, Selim and Chechik (2020). An example for bad image quality in regards to STS can be seen in figure 3.1(b). Note that the datasets introduced in Section 2.6 include the challenging image properties, as STS is defined with robustness in mind. Therefore, the difference performance and robustness is not clear cut for STS.

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

This part of the thesis constitutes the main part. The objective create an overview of techniques in the field which may be used to solve the problem detailed in Chapter 3. First the search for literature which lays the foundation for the subsequent sections, is documented. The a taxonomy is introduced which facilitates the analization and comparison of techniques. Lastly, the advances in the field are placed in the correct position and analyzed.

4.1 Literature Search

This section documents the search for literature which provides the content for the subsequent overview. For this, important pipelines and notable advances along with their properties are researched, analysed and presented. The literature is identified through searching in the Google Scholar database. The search is executed with keywords such as but not only: Deep Learning, Text Detection, Text Recognition, Text Spotting, Scene Text, Pipeline. A criterion for further examination is an appropriate amount of citations for the piece of literature in question. Additionally, literature is selected through citations for and by literature which has already been identified as important. All research after 2018 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). The delimination from Section 1.2 of course holds for this chapter and only literature which concerns advances for the DL model will be regarded as important for the scope of this thesis. This extends to the whole pipeline from preprocessing an image to the final result of the model.

4.2 Taxonomy of Pipeline Steps

In order to create the overview the necessary steps in the process of STS need to be highlighted, from preprocessing to classifying the identified text (Long et al., 2021; Sourvanos and Tsatiris, 2018). The ways in which the respective issues for the steps are solved are identified from literature, listed and explained alongside.

- Bestandteile nicht zwangsweise sequentiell
- Tiefe: Pipeline Steps → possible solutions → research points (only mention) e.g.: Sequence Modeling → CNN, RNN, Transformer → ...
- Lexicon Free Models!
- Trend: detection moves towards segmentation, recognition away
- for feature extraction:
  - architectures ResNet
    - aggressive downsampling
    - 3×3 best kernel size
    - Gradient saving → Res block, Res Bottleneck, Res Grouped
    - batch normalization

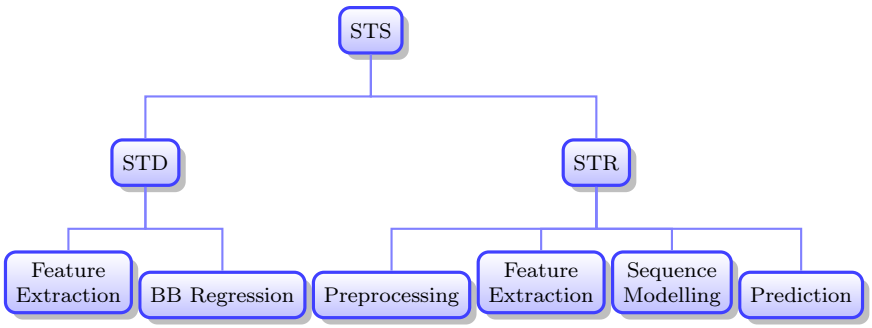


Figure 4.1: Pipeline steps with increasing granularity

Same pipeline for Baek et al. (2019)

- For Detection: describe how BB is assigned to ground truth while training & testing

An accurate Segmentation-Based ... (Liu et al., 2020)

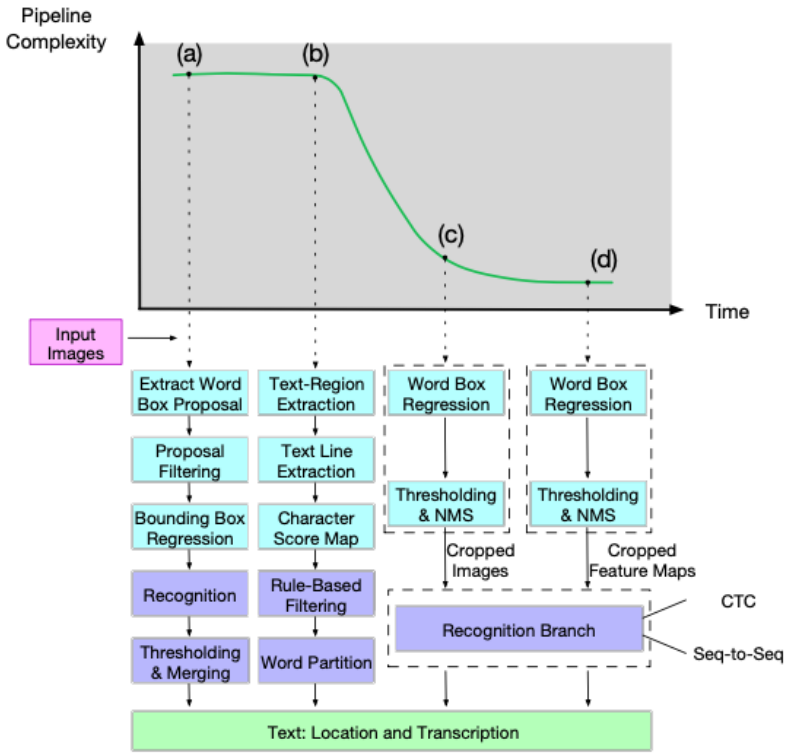


Fig. 3 Illustrations of representative scene text detection and recognition system pipelines. **a** Jaderberg et al. (2016) and **b** Yao et al. (2016) are representative multi-step methods. **c**, **d** are simplified pipeline. In **c**, detectors and recognizers are separate. In **d**, the detectors pass cropped feature maps to recognizers, which allows end-to-end training

Figure 4.2: Pipeline changes (Long et al., 2021)

- For detection there's two categories: detection-based, segmentation based → latter is more getting more popular
  - detection-based: adapt general object detection framework → directly regress quadrangles → problem with arbitrary shapes and small texts
  - segmentation-based: use pixel-wise segmentation to segment text areas → use segment text areas to extract text instances by post-processing
- look into paper for overview of text detection
- Text Detection is type of object detection

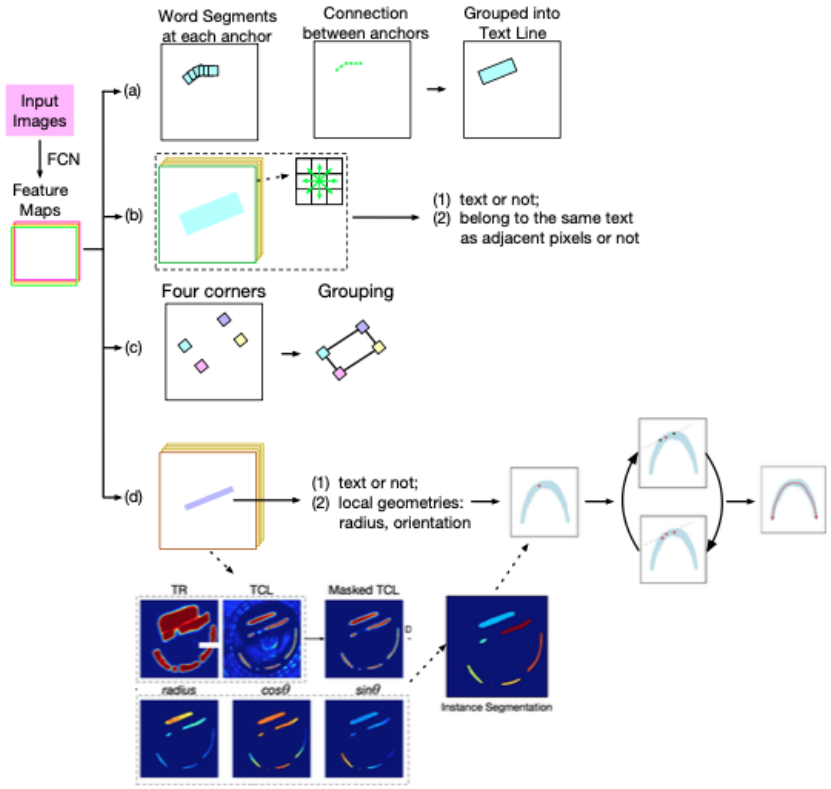


Figure 4.3: STD pipelines (Long et al., 2021)

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:

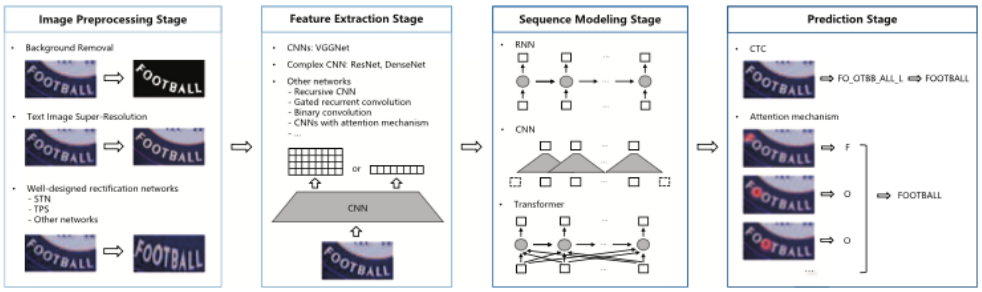


Figure 4.4: STR pipeline (Chen et al., 2021)

  - 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 → 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) → 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
  → regression based 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

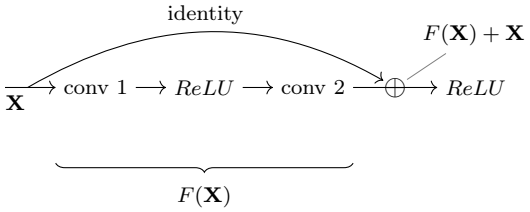
  - approach: recognize text line as a whole and focus on mapping entire text instance image into target string sequence
  - stages:
    * Preprocessing: improve image quality
      background removal: separate text from complex backgrounds
      super-resolution: TextSR can output plausible high-resolution 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

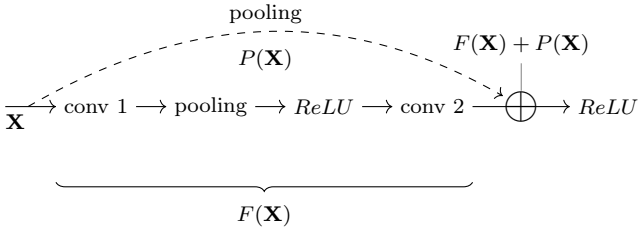
  –

CTC

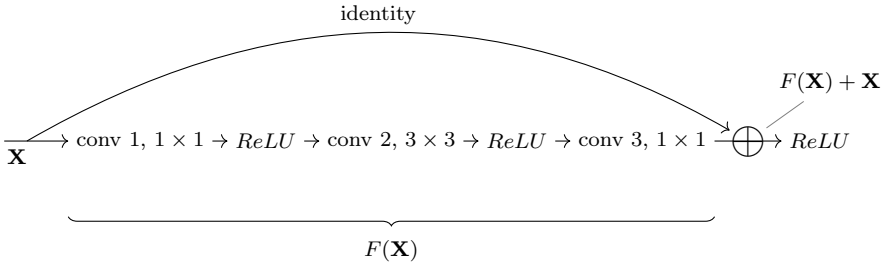
- for training RNNs to label unsegmented sequences directly



(a) Residual block



(b) Residual block with pooling



(c) Residual bottleneck block

Figure 4.5: Modules introduced with ResNet

- 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 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)

4.3 State of the Art Methods

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

Chapter 5

Discussion

5.1 Analysis

Plan

- Go down hierarchy and compare most important advances

For comparison: which Benchmark Dataset fits the problem the best?

- Liao et al. (2020):
  - Rotated ICDAR 2013 (changed normal icdar): rotation robustness
  - Total-Text: shape ropustness
  - MSRA-TD500: aspect ratio ropustness
- Yang et al. (2021):commonly used for oriented text: ICDAR2015, ICDAR2017 MLT, MSRA-TD500

Important points:

- ResNet is pretty much the Feature Extractor Benchmark

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

Threats to validity!

- Arpteg et al. (2018): different papers have different components → Hardware, Platform, Source Code, Configuration → studies can't really be compared
- Arpteg et al. (2018): 'A major challenge in developing DL systems is the difficulties in estimating the results before a system has been trained and tested.'
- Long et al. (2021): different interpretations of metrics (matching for STD, word/char for STR)
- Siebert et al. (2021); Nakamichi et al. (2020): all entities of MLS should be inspected when developing a solution
- Baek et al. (2019): different papers use different evaluation and testing environments
- Baek et al. (2019): different papers use different subsets of the same dataset → discrepancies in performance
- Long and Yao (2020): half of the widely adopted benchmark datasets have imperfect annotations → ignoring case-sensitivities and punctuations, and provide new annotations for those datasets
- Chen et al. (2021): inconsistency of datasets, priors and testing environments make comparison difficult

5.3 Outlook

- next steps to practically solve problem → Data Collection, Data Cleaning, Data Labeling, Model Training, Model Evaluation, Model Deployment, Model Monitoring (Watanabe et al., 2019)
- Use Neural Architecture Search to automatically find right feature extractor (Zhao et al., 2020)
- Siebert et al. (2021); Nakamichi et al. (2020): build system around model → e.g. supervision mechanism
- Shi et al. (2017); He et al. (2018): adjust field to better metrics for evaluation

- Long et al. (2021): general trend to move towards simpler,shorter pipeline

Chapter 6

Conclusion

Bibliography

Abramowicz, W. and Corchuelo, R., eds (2019), *Business Information Systems: 22nd International Conference, BIS 2019, Seville, Spain, June 26–28, 2019, Proceedings, Part II*, Vol. 354 of *Lecture Notes in Business Information Processing*, Springer International Publishing, Cham.
URL: *http://link.springer.com/10.1007/978-3-030-20482-2*

Alzubi, J., Nayyar, A. and Kumar, A. (2018), 'Machine Learning from Theory to Algorithms: An Overview', *Journal of Physics: Conference Series* **1142**, 012012.
URL: *https://iopscience.iop.org/article/10.1088/1742-6596/1142/1/012012*

Arpteg, A., Brinne, B., Crnkovic-Friis, L. and Bosch, J. (2018), Software Engineering Challenges of Deep Learning, *in* '2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)', pp. 50–59.

Ashmore, R., Calinescu, R. and Paterson, C. (2021), 'Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges', *ACM Computing Surveys* **54**(5), 1–39.
URL: *https://dl.acm.org/doi/10.1145/3453444*

Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S. J. and Lee, H. (2019), What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis, *in* '2019 IEEE/CVF International Conference on Computer Vision (ICCV)', IEEE, Seoul, Korea (South), pp. 4714–4722.
URL: *https://ieeexplore.ieee.org/document/9010273/*

Boué, L. (2018), 'Deep learning for pedestrians: backpropagation in CNNs', *arXiv:1811.11987 [cs, stat]* . arXiv: 1811.11987.
URL: *http://arxiv.org/abs/1811.11987*

Chauhan, N. K. and Singh, K. (2018), A Review on Conventional Machine Learning vs Deep Learning, *in* '2018 International Conference on Computing, Power and Communication Technologies (GUCON)', pp. 347–352.

Chen, X., Jin, L., Zhu, Y., Luo, C. and Wang, T. (2021), 'Text Recognition in the Wild: A Survey', *ACM Computing Surveys* **54**(2), 1–35.
URL: *https://dl.acm.org/doi/10.1145/3440756*

Ch'ng, C. K. and Chan, C. S. (2017), Total-Text: A Comprehensive Dataset for Scene Text Detection and Recognition, *in* '2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)', Vol. 01, pp. 935–942. ISSN: 2379-2140.

Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma and Zhuowen Tu (2012), Detecting texts of arbitrary orientations in natural images, *in* '2012 IEEE Conference on Computer Vision and Pattern Recognition', IEEE, Providence, RI, pp. 1083–1090.
URL: *http://ieeexplore.ieee.org/document/6247787/*

Davis, J. and Goadrich, M. (2006), The relationship between Precision-Recall and ROC curves, *in* 'Proceedings of the 23rd international conference on Machine learning - ICML '06', ACM Press, Pittsburgh, Pennsylvania, pp. 233–240.
URL: *http://portal.acm.org/citation.cfm?doid=1143844.1143874*

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. (2021), 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale', *arXiv:2010.11929 [cs]* . arXiv: 2010.11929.
URL: *http://arxiv.org/abs/2010.11929*

Gers, F. A., Schmidhuber, J. and Cummins, F. (1999), 'Learning to Forget: Continual Prediction with LSTM', *Neural Computation* **12**, 2451–2471.

Ghosh, S. K., Valveny, E. and Bagdanov, A. D. (2017), Visual Attention Models for Scene Text Recognition, *in* '2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)', Vol. 01, pp. 943–948. ISSN: 2379-2140.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep learning*, Adaptive computation and machine learning, The MIT Press, Cambridge, Massachusetts.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. and Schmidhuber, J. (2017), 'LSTM: A Search Space Odyssey', *IEEE Transactions on Neural Networks and Learning Systems* **28**(10), 2222–2232. arXiv: 1503.04069.
URL: *http://arxiv.org/abs/1503.04069*

Géron, A. (2017), *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*, first edition edn, O'Reilly Media, Beijing ; Boston. OCLC: ocn953432302.

He, K., Zhang, X., Ren, S. and Sun, J. (2015), 'Deep Residual Learning for Image Recognition', *arXiv:1512.03385 [cs]* . arXiv: 1512.03385.
URL: *http://arxiv.org/abs/1512.03385*

He, M., Liu, Y., Yang, Z., Zhang, S., Luo, C., Gao, F., Zheng, Q., Wang, Y., Zhang, X. and Jin, L. (2018), ICPR2018 Contest on Robust Reading for Multi-Type Web Images, *in* '2018 24th International Conference on Pattern Recognition (ICPR)', pp. 7–12. ISSN: 1051-4651.

Ho, Y. and Wookey, S. (2020), 'The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling', *IEEE Access* **8**, 4806–4813. Conference Name: IEEE Access.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long Short-Term Memory', *Neural Computation* **9**(8), 1735–1780. Conference Name: Neural Computation.

Hu, B. C., Salay, R., Czarnecki, K., Rahimi, M., Selim, G. and Chechik, M. (2020), Towards Requirements Specification for Machine-learned Perception Based on Human Performance, *in* '2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)', pp. 48–51.

Hu, W., Cai, X., Hou, J., Yi, S. and Lin, Z. (2020), 'GTC: Guided Training of CTC Towards Efficient and Accurate Scene Text Recognition', *arXiv:2002.01276 [cs, eess]* . arXiv: 2002.01276.
URL: *http://arxiv.org/abs/2002.01276*

Ignatov, A., Timofte, R., Kulik, A., Yang, S., Wang, K., Baum, F., Wu, M., Xu, L. and Van Gool, L. (2019), AI Benchmark: All About Deep Learning on Smartphones in 2019, *in* '2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)', pp. 3617–3635. ISSN: 2473-9944.

James, G., Witten, D., Hastie, T. and Tibshirani, R., eds (2013), *An introduction to statistical learning: with applications in R*, number 103 *in* 'Springer texts in statistics', Springer, New York. OCLC: ocn828488009.

Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V. R., Lu, S., Shafait, F., Uchida, S. and Valveny, E. (2015), ICDAR 2015 competition on Robust Reading, *in* '2015 13th International Conference on Document Analysis and Recognition (ICDAR)', pp. 1156–1160.

Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Bigorda, L. G. i., Mestre, S. R., Mas, J., Mota, D. F., Almazàn, J. A. and de las Heras, L. P. (2013), ICDAR 2013 Robust Reading Competition, *in* '2013 12th International Conference on Document Analysis and Recognition', pp. 1484–1493. ISSN: 2379-2140.

Liao, M., Pang, G., Huang, J., Hassner, T. and Bai, X. (2020), 'Mask TextSpotter v3: Segmentation Proposal Network for Robust Scene Text Spotting', *arXiv:2007.09482 [cs]* . arXiv: 2007.09482.
URL: *http://arxiv.org/abs/2007.09482*

Liao, M., Shi, B. and Bai, X. (2018), 'TextBoxes++: A Single-Shot Oriented Scene Text Detector', *IEEE Transactions on Image Processing* **27**(8), 3676–3690. arXiv: 1801.02765.
URL: *http://arxiv.org/abs/1801.02765*

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016), SSD: Single Shot MultiBox Detector, *in* B. Leibe, J. Matas, N. Sebe and M. Welling, eds, 'Computer Vision – ECCV 2016', Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 21–37.

Liu, X., Zhou, G., Zhang, R. and Wei, X. (2020), An Accurate Segmentation-Based Scene Text Detector with Context Attention and Repulsive Text Border, *in* '2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)', IEEE, Seattle, WA, USA, pp. 2344–2352.
URL: *https://ieeexplore.ieee.org/document/9151062/*

Long, S., He, X. and Yao, C. (2021), 'Scene Text Detection and Recognition: The Deep Learning Era', *International Journal of Computer Vision* **129**(1), 161–184.
URL: *https://link.springer.com/10.1007/s11263-020-01369-0*

Long, S. and Yao, C. (2020), 'UnrealText: Synthesizing Realistic Scene Text Images from the Unreal World', *arXiv:2003.10608 [cs]* . arXiv: 2003.10608.
URL: *http://arxiv.org/abs/2003.10608*

Mishra, A., Alahari, K. and Jawahar, C. (2012), Scene Text Recognition using Higher Order Language Priors, *in* 'Procedings of the British Machine Vision Conference 2012', British Machine Vision Association, Surrey, pp. 127.1–127.11.
URL: *http://www.bmva.org/bmvc/2012/BMVC/paper127/index.html*

Mitchell, T. M. (1997), *Machine Learning*, McGraw-Hill series in computer science, McGraw-Hill, New York.

Nakamichi, K., Ohashi, K., Namba, I., Yamamoto, R., Aoyama, M., Joeckel, L., Siebert, J. and Heidrich, J. (2020), 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)', pp. 260–270. ISSN: 2332-6441.

Nayef, N., Yin, F., Bizid, I., Choi, H., Feng, Y., Karatzas, D., Luo, Z., Pal, U., Rigaud, C., Chazalon, J., Khlif, W., Luqman, M. M., Burie, J.-C., Liu, C.-l. and Ogier, J.-M. (2017), ICDAR2017 Robust Reading Challenge on Multi-Lingual Scene Text Detection and Script Identification - RRC-MLT, *in* '2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)', Vol. 01, pp. 1454–1459. ISSN: 2379-2140.

Niu, W., Ma, X., Wang, Y. and Ren, B. (2019), '26ms Inference Time for ResNet-50: Towards Real-Time Execution of all DNNs on Smartphone', *arXiv:1905.00571 [cs, stat]* . arXiv: 1905.00571.
URL: *http://arxiv.org/abs/1905.00571*

Oyedotun, O. K., Olaniyi, E. O. and Khashman, A. (2015), 'Deep Learning in Character Recognition Considering Pattern Invariance Constraints', *International Journal of Intelligent Systems and Applications* **7**(7), 1–10.
URL: *http://www.mecs-press.org/ijisa/ijisa-v7-n7/v7n7-1.html*

Pennington, J., Socher, R. and Manning, C. (2014), GloVe: Global Vectors for Word Representation, *in* 'Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)', Association for Computational Linguistics, Doha, Qatar, pp. 1532–1543.
URL: *https://aclanthology.org/D14-1162*

Ponti, M. A., Ribeiro, L. S. F., Nazare, T. S., Bui, T. and Collomosse, J. (2017), 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)', pp. 17–41. ISSN: 2474-0705.

Sherstinsky, A. (2020), 'Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network', *Physica D: Nonlinear Phenomena* **404**, 132306. arXiv: 1808.03314.
URL: *http://arxiv.org/abs/1808.03314*

Shi, B., Yao, C., Liao, M., Yang, M., Xu, P., Cui, L., Belongie, S., Lu, S. and Bai, X. (2017), ICDAR2017 Competition on Reading Chinese Text in the

Wild (RCTW-17), *in* '2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)', Vol. 01, pp. 1429–1434. ISSN: 2379-2140.

Shrestha, A. and Mahmood, A. (2019), 'Review of Deep Learning Algorithms and Architectures', *IEEE Access* **7**, 53040–53065. Conference Name: IEEE Access.

Siebert, J., Joeckel, L., Heidrich, J., Trendowicz, A., Nakamichi, K., Ohashi, K., Namba, I., Yamamoto, R. and Aoyama, M. (2021), 'Construction of a quality model for machine learning systems', *Software Quality Journal* .
URL: *https://link.springer.com/10.1007/s11219-021-09557-y*

Snyder, H. (2019), 'Literature review as a research methodology: An overview and guidelines', *Journal of Business Research* **104**, 333–339.
URL: *http://www.sciencedirect.com/science/article/pii/S0148296319304564*

Sourvanos, N. and Tsatiris, G. (2018), Challenges in Input Preprocessing for Mobile OCR Applications: A Realistic Testing Scenario, *in* '2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)', pp. 1–5.

Su, W., Yuan, Y. and Zhu, M. (2015), A Relationship between the Average Precision and the Area Under the ROC Curve, *in* 'Proceedings of the 2015 International Conference on The Theory of Information Retrieval', ACM, Northampton Massachusetts USA, pp. 349–352.
URL: *https://dl.acm.org/doi/10.1145/2808194.2809481*

Sun, Y., Ni, Z., Chng, C.-K., Liu, Y., Luo, C., Ng, C. C., Han, J., Ding, E., Liu, J., Karatzas, D., Chan, C. S. and Jin, L. (2019), ICDAR 2019 Competition on Large-Scale Street View Text with Partial Labeling - RRC-LSVT, *in* '2019 International Conference on Document Analysis and Recognition (ICDAR)', pp. 1557–1562. ISSN: 2379-2140.

Torraco, R. J. (2005), 'Writing Integrative Literature Reviews: Guidelines and Examples', *Human Resource Development Review* **4**(3), 356–367. Publisher: SAGE Publications.
URL: *https://doi.org/10.1177/1534484305278283*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017), Attention is All you Need, *in* 'Advances in Neural Information Processing Systems', Vol. 30, Curran Associates, Inc.

URL: *https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html*

Vogelsang, A. and Borg, M. (2019), Requirements Engineering for Machine Learning: Perspectives from Data Scientists, *in* '2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)', pp. 245–251.

Watanabe, Y., Washizaki, H., Sakamoto, K., Saito, D., Honda, K., Tsuda, N., Fukazawa, Y. and Yoshioka, N. (2019), 'Preliminary Systematic Literature Review of Machine Learning System Development Process', *arXiv:1910.05528 [cs]* . arXiv: 1910.05528.
URL: *http://arxiv.org/abs/1910.05528*

Yang, X., Yang, X., Yang, J., Ming, Q., Wang, W., Tian, Q. and Yan, J. (2021), 'Learning High-Precision Bounding Box for Rotated Object Detection via Kullback-Leibler Divergence', *arXiv:2106.01883 [cs]* . arXiv: 2106.01883.
URL: *http://arxiv.org/abs/2106.01883*

Yu, Y., Si, X., Hu, C. and Zhang, J. (2019), 'A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures', *Neural Computation* **31**(7), 1235–1270.
URL: *https://direct.mit.edu/neco/article/31/7/1235-1270/8500*

Yuliang, L., Lianwen, J., Shuaitao, Z. and Sheng, Z. (2017), 'Detecting Curve Text in the Wild: New Dataset and New Solution', *arXiv:1712.02170 [cs]* . arXiv: 1712.02170 version: 1.
URL: *http://arxiv.org/abs/1712.02170*

Zhang, J. M., Harman, M., Ma, L. and Liu, Y. (2020), 'Machine Learning Testing: Survey, Landscapes and Horizons', *IEEE Transactions on Software Engineering* pp. 1–1. Conference Name: IEEE Transactions on Software Engineering.

Zhao, Z., Jiang, M., Guo, S., Wang, Z., Chao, F. and Tan, K. C. (2020), Improving Deep Learning based Optical Character Recognition via Neural Architecture Search, *in* '2020 IEEE Congress on Evolutionary Computation (CEC)', pp. 1–7.

Zowghi, D., Jin, Z., Junqueira Barbosa, S. D., Chen, P., Cuzzocrea, A., Du, X., Filipe, J., Kara, O., Kotenko, I., Sivalingam, K. M., Ślęzak, D., Washio, T. and Yang, X., eds (2014), *Requirements Engineering*, Vol. 432 of *Communications in Computer and Information Science*, Springer Berlin Heidelberg,

Berlin, Heidelberg.
URL: *http://link.springer.com/10.1007/978-3-662-43610-3*

Appendix A

Litaratur Qualities

The following tables show qualities that where identified in literature. The qualities are categorized by the MLS entities defined in Siebert et al. (2021). The model entity is the focus of this work and is thus discussed in Chapter 3.

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: MLS qualities identified for data entity

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: MLS qualities identified for infrastrucure entity

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: MLS qualities identified for environment entity

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: MLS qualities identified for system entity