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	07.03.2022

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 07.03.2022

.....
Johannes Reichle

Abstract

Here abstract for Bachelor Thesis.

Keywords: Deep Learning, Scene Text, Text Spotting, Overview

Contents

List of Figures

List of Tables

Abbreviations

AED Average Edit Distance

AP Average Precision

BB Bounding Box

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

NMS Non Maximum Suppression

NN Neural Network

OCR Optical Character Recognition

RNN Recurrent Neural Network

ROI Region of Interest

RPN Region Proposal 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

Digitization can be described as transforming analog information into a digital respresentation (Imgrund et al., 2018). Information systems in conjunction with digitization helps to optimize efficiency and productivity for business performance, as well as to reduce costs (Imgrund et al., 2018). This facilitates the growing need for automation (Imgrund et al., 2018). Additionally, a comprehensive information system with such digitized information allows a company to aggregate and share information to harness it (Goodhue et al., 1992). However, before reaping the rewards, the information must be digitized in the first place. Take technicians for example, who work in the field 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 automated process of digitizing such data is called Optical Character Recognition (OCR), 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). Applying these new capabilities and finding the right solution in the space of DL for the use case of extracting information of labels is the focus of this thesis. This is an interesting task as performance of OCR systems in natural scenes is still challenging (Zhao et al., 2020; Chen et al., 2021). Such scenes entail natural scenes captured by a camera (Chen et al., 2021; Baek et al., 2019). Factors such as complex backgrounds, noise, perspective and variability in fonts, colors and sizes, of scene texts complicate the process (Hu et al., 2020a; 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

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 must be determined for this. What qualities such as detecting alpha-numeric strings or suitability despite inadequate image conditions must a solution have (Ghosh et al., 2017; Hu et al., 2020a)?

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 techniques that can help in choosing which to implement and test to solve the specific problem of STS, defined in Section 1.2 and more detailed in Chapter 3.

The research question guiding the process is most crucial (Snyder, 2019): Which state of the art DL approaches for STS 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.

For a literature review, it is important to report how the information was found and synthesized (Torraco, 2005). Therefore, each section in the overview contains a paragraph about it. Before getting into current research level, a foundation of knowledge about the field must be layed. A taxonomy is created for this which is useful to classify and give context to innovations in the field. The partition of tasks and categorization of approaches is conducted according to information from overview literature such as Long et al. (2021); Chen et al. (2021); Cong et al. (2019) and difference of approaches that can be identified in research such as Qiao et al. (2021); Sheng et al. (2021); Liu et al. (2020); Deng et al. (2018). The taxonomy is determined according to the requirement for clarity of subsequent provision of current research.

The strategy for researching current research is most important for a literature review (Snyder, 2019). This includes determining databases and keywords that are used, as well as exclusion criteria that are enforced (Torraco, 2005). Innovations are explored 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 that is aligned according to the taxonomy.

In the analysis possibly viable approaches are compared with the qualities defined in Chapter 3. The comparison will be organized according the taxonomy which facilitates the clarity and comprehensibility. The comparison is arranged in hierarchical fashion: different pipeline categories, different categories for pipeline tasks and innovations for those tasks. 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

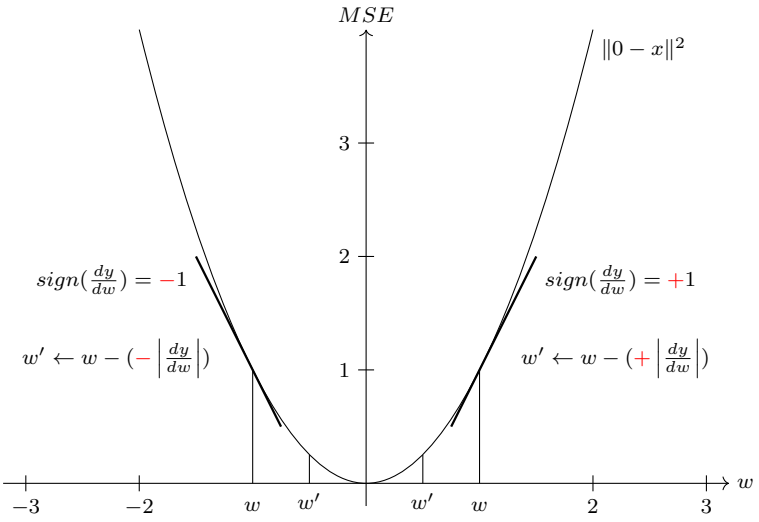


Figure 2.1: Gradient descent for 1-dimensional objective function (Goodfellow et al., 2016)

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

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

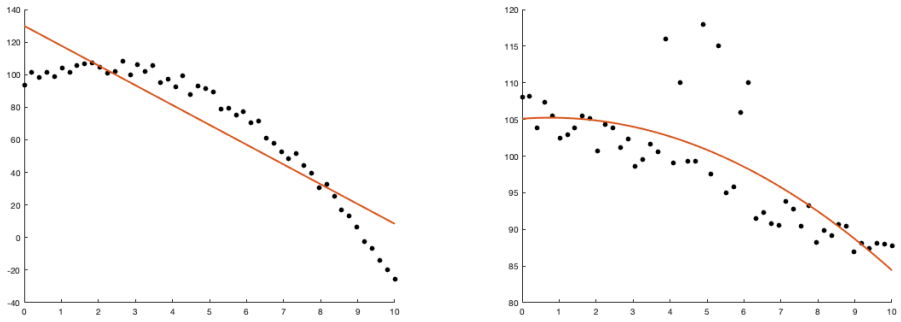
Note that for minimization the MSE is a function of $\mathbf{w}, b$ and not of $\mathbf{x}$, in terms of predicting a value the MSE is a function of $\mathbf{x}$ parametrized by $\mathbf{w}, b$ (see Equation 2.3). In Equation 2.1 $\mathbf{w}$,b are parameters that have to be learned in order to minimize 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). 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). In this section the basics of Neural Networks (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-

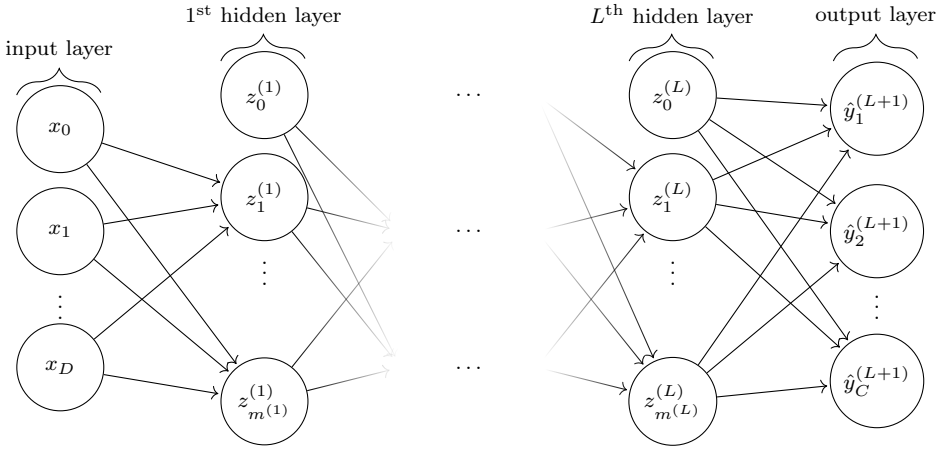


Figure 2.3: Network graph of a $(L+1)$-layer perceptron with D input units and C output units. The l^{th} hidden layer contains $m^{(l)}$ hidden units (Chauhan and Singh, 2018; Goodfellow et al., 2016).

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 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

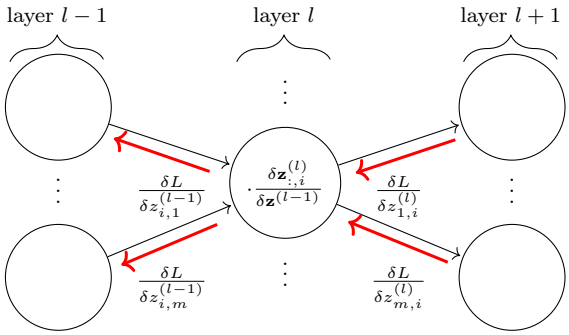


Figure 2.4: Reverse traversing the network's computation graph, $\cdot \frac{\delta \mathbf{z}_{:,i}^{(l)}}{\delta \mathbf{w}_i^{(l)}}$ is used for updating the neurons parameters

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). Note that the height and with are refered to as spatial dimensions and the depth is refered to as the channel dimension. 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 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 the convolution is usually fed into a ReLU activation function to introduce nonlinearity like with fully connected layers. The activation is performed on every element in the

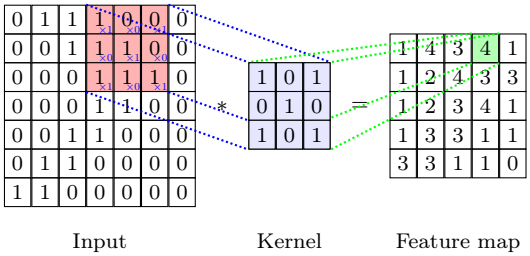


Figure 2.5: Convolution operation of a single kernel (Chauhan and Singh, 2018)

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).

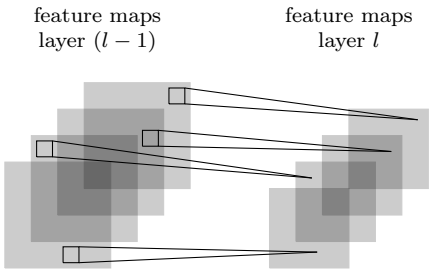


Figure 2.6: Pooling layers preserve channel dimenstions but downsample spatial dimensions

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

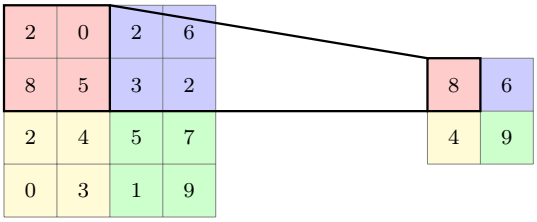


Figure 2.7: 2×2 max pooling operation with stride 2 (Chauhan and Singh, 2018)

ing/exploding (He et al., 2015).

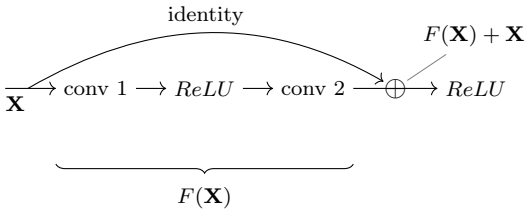


Figure 2.8: Residual block module with skip connection (He et al., 2015)

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

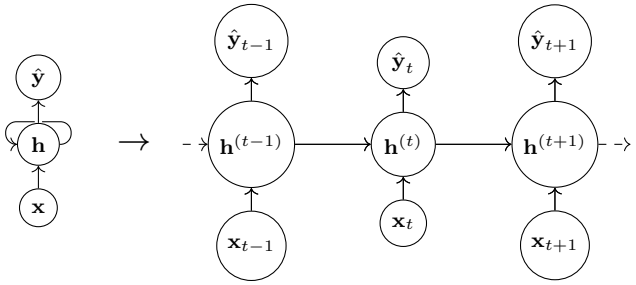


Figure 2.9: Recurrent neural net unrolling (Goodfellow et al., 2016)

node $\mathbf{h}$ 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 $\mathbf{x}^{(t)}$ and $\mathbf{h}^{(t-1)}$ and

produces $\mathbf{h}^{(t)}$ (see Equation 2.14). Additionally, $\mathbf{h}^{(t)}$ is used to calculate the output $\hat{\mathbf{y}}^{(t)}$ (see Equation 2.15) and it is handed to the next layer (Goodfellow et al., 2016). $\mathbf{h}^{(t)}$ is thoght of as a hidden state which stores information from previous inputs (Goodfellow et al., 2016).

$$\mathbf{h}^{(t)} = \tanh(b + W\mathbf{h}^{(t-1)} + U\mathbf{x}^{(t)}) \tag{2.14}$$

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

Note that the connection 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). This makes gradients from backpropagation vulnerable to exploding/vanishing gradients if the singular values of those weight matrices are > 1 or < 1 (Goodfellow et al., 2016). 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 (truncated) backpropagation through time (Sherstinsky, 2020).

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 gradients problem associated

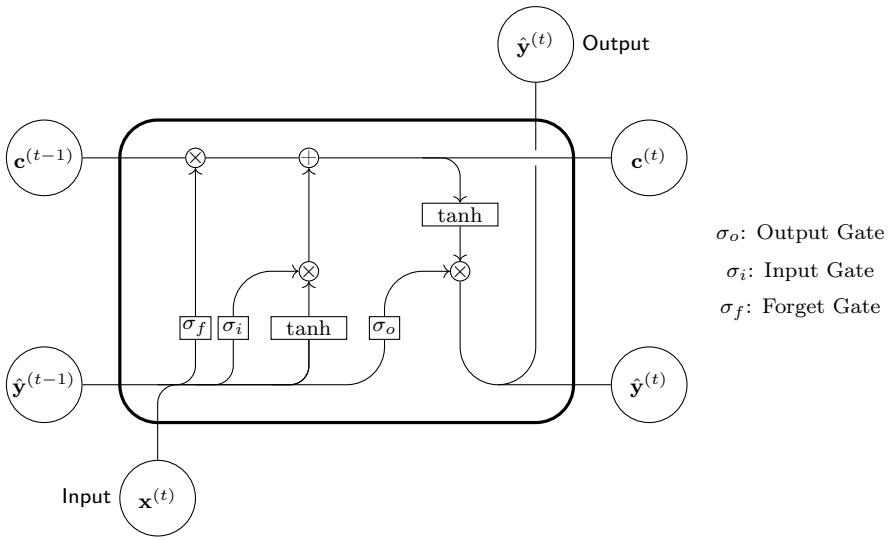


Figure 2.10: Long short term memory cell, merged lines stack vectors, split lines duplicate vector (Goodfellow et al., 2016; Yu et al., 2019)

with increasingly deep NNs (Sherstinsky, 2020) (clipping gradients is often

used to help with exploding gradients (Goodfellow et al., 2016)). 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 (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

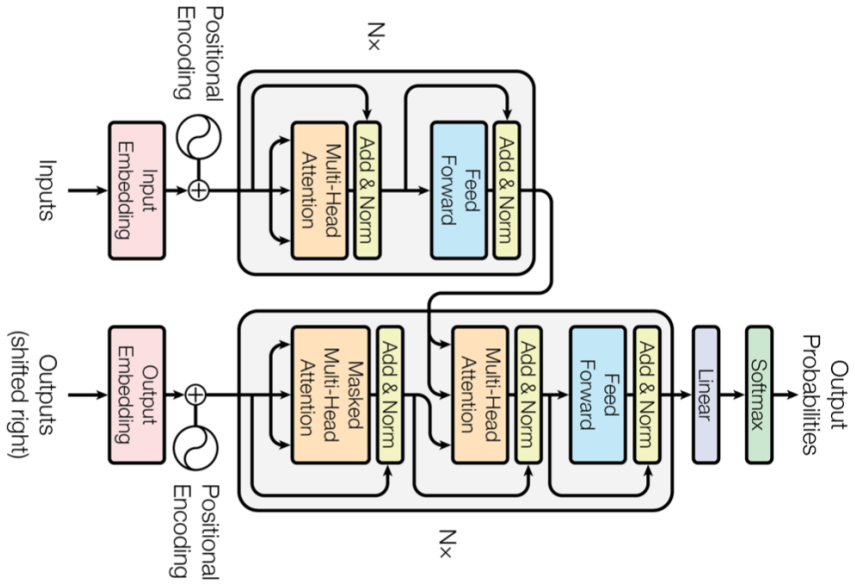


Figure 2.11: Transformer model architecture (Vaswani et al., 2017)

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 a sentence. 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 blocks (Nx) (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 can change the meaning, 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 (introduced with residual block (He et al., 2015)) and layer normalization (Vaswani et al., 2017) which helps with training the network and to keep it stable (Liu et al., 2021). A multi-head attention block calculates which words (by now embedded) of the input are important for each other word. The attention is calculated by a scaled dot product (see Figure 2.12) (Vaswani et al., 2017). This is calculated multiple times (hence the

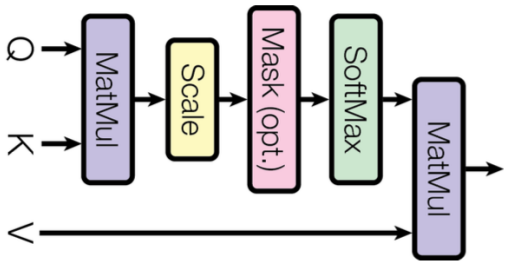


Figure 2.12: Scaled dot product used to compute attention vectors (Vaswani et al., 2017)

term multi-head) and the results are merged by concatenation and a subseding linear layer (Vaswani et al., 2017). This allows the model to use different representations of the information (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 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., 2018a). 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. (2018a). 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., 2017a). 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). Often lexicons which contain possible words, are used by STR. Testing as well as real world performance can depend strongly on these lexicons (Chen et al., 2021; 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).

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

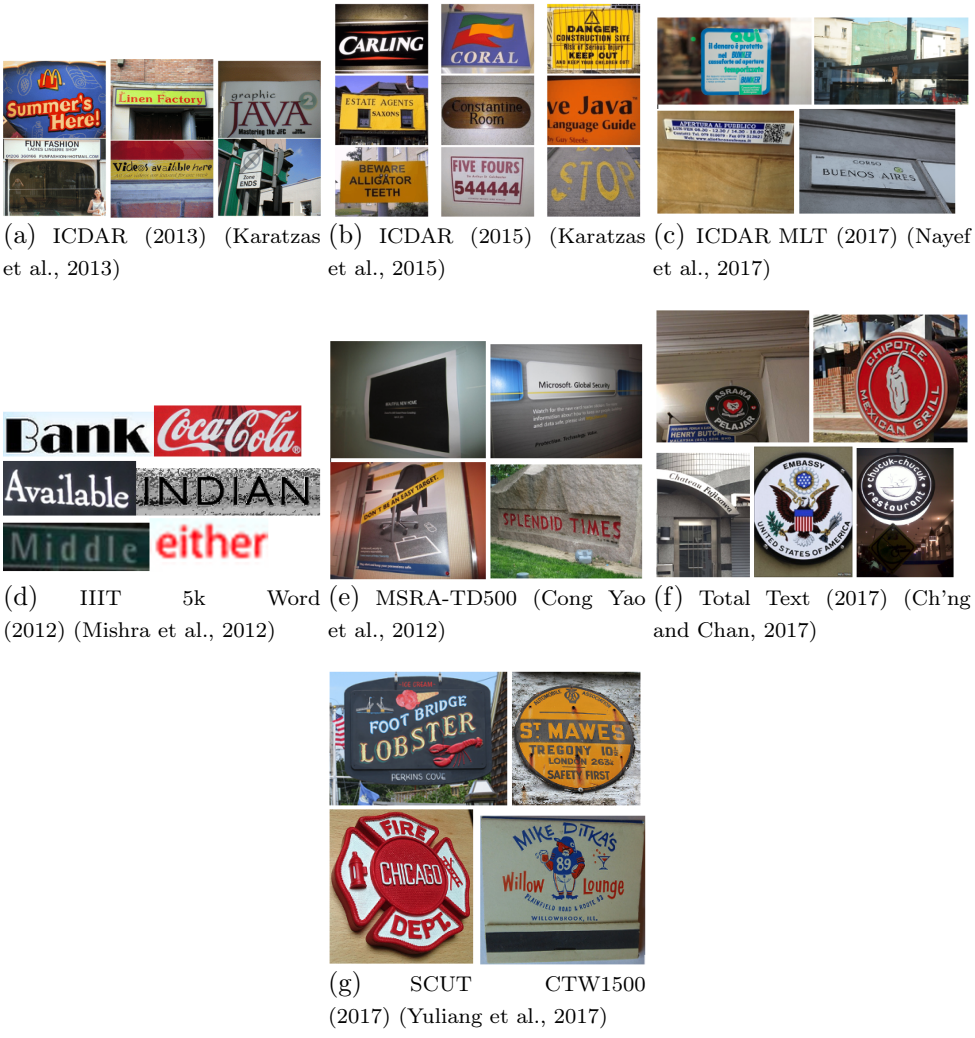
D denotes the Levenshtein distance, s denotes the text, l denotes the text length and N is the total number of text lines (Shi et al., 2017a). For STR NED is used over the whole dataset (Karatzas et al., 2013). STS is oriented to both STD and STR. The prediction has to be matched to the ground truth 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. (2018a); Shi et al. (2017a). 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

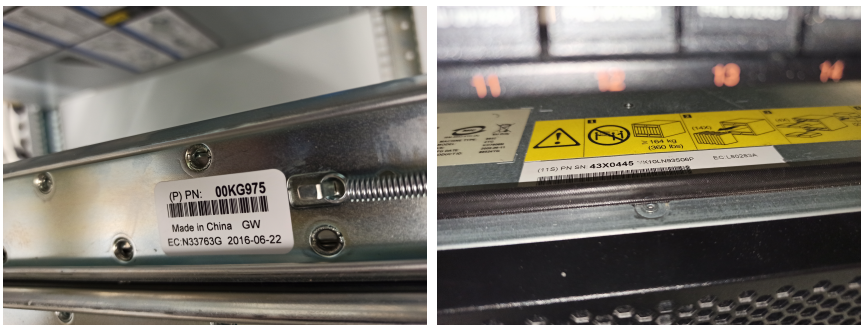
The problem can be depicted by a use case. This use case sets the foundation for determining requirements for an approach because qualities derive from the intended purpose of use (Siebert et al., 2021). Table 3.1 gives an overview over the relevant properties that can be derived from the use case. For 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 (see Figure (Nakamichi et al., 2020)). These entities are data, model, environment,

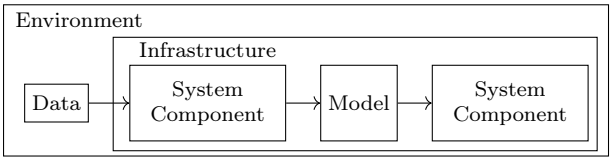


Figure 3.2: Machine learning system entities (Nakamichi et al., 2020)

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.

Quality	**Sources**
Appropriateness	
Appropriateness	Siebert et al. (2021)
Suitability	Siebert et al. (2021)
Model Fitness — Quality of Output Data	Nakamichi et al. (2020)
Performance	
Performance	Ashmore et al. (2021); Vogelsang and Borg (2019)
Accuracy	Nakamichi et al. (2020)
Model Fitness — Degree of Correctness	Nakamichi et al. (2020); Zhang et al. (2020)
Development correctness	Siebert et al. (2021)
Robustness	
Robustness	Ashmore et al. (2021); Hu et al. (2020b); 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.

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

Table 3.3: Condensed Qualities for model entity

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.

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

The robustness of a model concerns environmental uncertainty Ashmore et al. (2021). Due to the uncontrolled environment in the practical aspect of taking the images on-site beneficial image properties can not be guaranteed (Chen et al., 2021). Robust text extraction can be influenced by factors such as complex backgrounds, text form (text rotation, font variability, arrangement), image noise (lighting conditions, blur, interference and low resolution) and access (perspective, shape of text) (Oyedotun et al., 2015; Ghosh et al., 2017; Chen et al., 2021). Therefore, these properties have to be accounted for when determining the viability for an approach. Some of these factors do not change the expected prediction (noise), others do (text form)Hu et al. (2020b). 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. For example Karatzas et al. (2013, 2015); Ch'ng and Chan (2017) define their challenge with different image properties concerned with robustness. Additionally, STS is differenciated from OCR by solving more difficult reading problems with more complex image (Long et al., 2021; Hu et al., 2020a; Chen et al.,

2021; Baek et al., 2019) 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

The objective is to create an overview of techniques in the field which may be used to solve the problem detailed in Chapter 3. According to the methodology detailed in 1.3, first a taxonomy is introduced which facilitates the analysis and comparison of techniques. The subsequent search for literature which lays the foundation for an overview over current research is documented. Lastly, the advances in the field are placed in the correct position and analyzed.

4.1 Taxonomy of Pipeline Steps

Before getting into current research level, a base of knowledge about the field has to be layed. A taxonomy is created for this which is useful to classify and give context to innovations in the field. The partition of tasks and categorization of approaches is conducted according to overview literature such as Long et al. (2021); Chen et al. (2021); Cong et al. (2019) and related work sections of research in the field such as Qiao et al. (2021); Sheng et al. (2021); Liu et al. (2020); Deng et al. (2018). To create the overview the necessary steps in the process of STS needs 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. Figure 4.1 shows two pipelines categories for end to end STS approaches (Long et al., 2021). STD and STR only incorporate a part of STS, while end to end approaches. incorporate both STD and STR techniques to solve STS (Long et al., 2021). Therefore this section will first discuss the two parts, to then later combine them.

For STD two main categories of approaches can be identified: segmentation based and Bounding Box (BB) regression based (see Figure 4.2) (Long et al., 2021; Sheng et al., 2021; Liu et al., 2020). The regression based category draws heavy inspiration from the field of object detection (Long et al., 2021; Liu et al.,

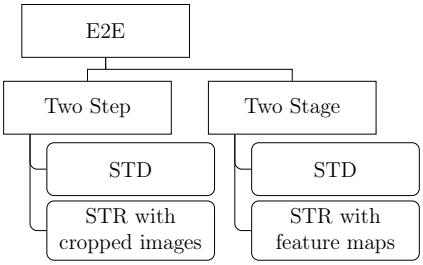


Figure 4.1: Different E2E Pipelines

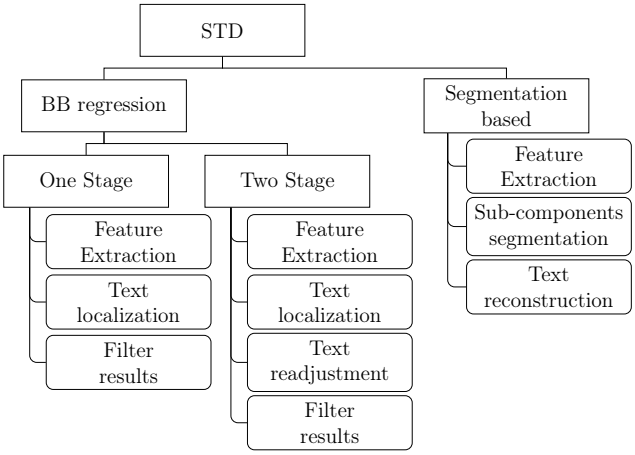


Figure 4.2: Different STD Pipelines

2020). This is only natural as text detection can be seen as a type of object detection (Liu et al., 2020; Long et al., 2021). For object detection inspired STD there are two methods: one stage and two stage (Long et al., 2021). Both localize text instances as a whole (in the form of a BB) (Long et al., 2021; Sheng et al., 2021). One stage approaches are modelled after Liu et al. (2016), Single Shot MultiBox Detector (SSD) and Redmon et al. (2016), You Only Look Once (YOLO). They have in comon that BBs are regressed once and not changed or optimized afterwards (Redmon et al., 2016; Liu et al., 2016), as opposed to the Region of Interest (ROI) based approach with two stages (Girshick et al., 2014). The basic approach is explained with the example of Liao et al. (2017) (see Figure 4.3) which is based on SSD. Note that the approach is modeled to recognize horizontal text instances (Liao et al., 2017). It uses 13 layer convolutional network inspired by the VGG architecture (blocks of: two or three 3×3 conv layers followed by a 2×2 max pooling layer with

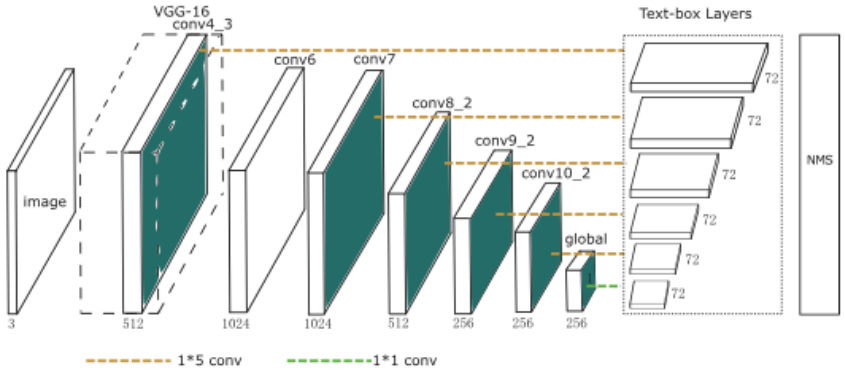


Figure 4.3: Example for a one stage, BB regression based STD architecture (Liao et al., 2017)

stride 2) for feature extraction (Liao et al., 2017; Simonyan and Zisserman, 2015). Note that the spatial padding added for convolution ensures that spatial dimensions are preserved (Simonyan and Zisserman, 2015). Afterwards come nine additional layers which continuously downsample, the output of six of them is separately used as feature maps for BB regression (Liao et al., 2017). The downsampling and BB regression for different layers helps detect text instances of different scales (Liu et al., 2016). Each spatial location on the feature map can be traced back to a region on the input image (Long et al., 2021). The BB regression is carried out by six text-box layers which predict how certain (c_1, c_2) the prediction is a text instance or background and where the text instance is (x, y, w, h). Note that the output is not the location of a BB but the offset to the respective anchor box (Liao et al., 2017; Long et al., 2021). Anchor boxes are predefined to give bias towards sizes and aspect ratios of text (Liao et al., 2017). The text-box layers are the difference to the SSD approach for normal object detection (Liao et al., 2017; Liu et al., 2016). These layers use 1×5 filters to adjust to larger aspect ratios (Liao et al., 2017). Each text-box layer has 72 filters (12 anchor boxes $\cdot$ 6 values per prediction), the filters are slided accross the input features generating 12 predicted BB per position (Liao et al., 2017). The BBs of all layers are then subjected to the process of Non Maximum Suppression (NMS) to filter out the best BB for each possible text instance (Liao et al., 2017). For this NMS is used: of all detections which overlap more than a threshhold ϕ only the with the highest confidence score (c) is kept (Hosang et al., 2017).

The R-CNN which builds the foundation for ROI based text detection, was introduced by Girshick et al. (2014) and improved by Girshick (2015) (Fast R-CNN), Ren et al. (2015) (Faster R-CNN) and He et al. (2018b) (Mask R-CNN). Note that 2-stage methods are fully differentiable and thus end to end trainable since Faster R-CNN (like 1-stage methods) (Ren et al., 2015; Long et al.,

2021). The two stages consist of: ROIs regression, BBs adjustments (Jiang et al., 2017; Ren et al., 2015). The STD approach introduced by Jiang et al. (2017) (see Figure 4.4) uses Faster R-CNN.Unlike the previous approach, the architecture is designed to detect multioriented text instances as such (Jiang et al., 2017; Liao et al., 2017). Like with the one stage approach, the two stage

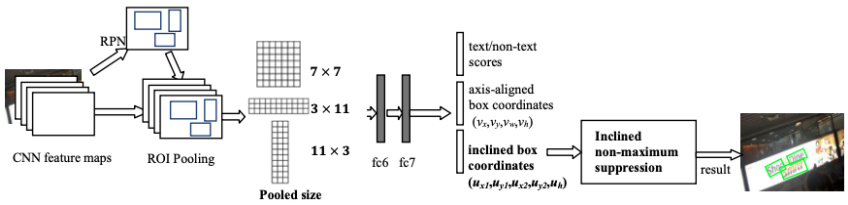


Figure 4.4: Example for a two stage, BB regression based STD architecture (Jiang et al., 2017)

approach starts with feature extraction from the image with a convolutional layers (Jiang et al., 2017). Feature extraction can be performed with the VGG architecture (Jiang et al., 2017), like in the previous approach. The generated feature map is used by a Region Proposal Network (RPN). Like with the previously explained approach, the feature maps are used to regress the offset respective to bounding boxes. The bounding boxes are still axis aligned at this point (Jiang et al., 2017). In contrast to the previous SSD based approach, only one feature map is used in conjunction with BB regression (Jiang et al., 2017). The RPN from Faster R-CNN is adjusted to use smaller scale anchor boxes to adapt to text (Jiang et al., 2017). Note that R-CNN and Fast-RCNN used the slower Sequential Search algorithm instead of an RPN (Girshick et al., 2014; Girshick, 2015). The resulting BBs are called ROIs (Ren et al., 2015; Jiang et al., 2017). They are used for ROI pooling in conjunction with the original feature maps. This layer uses max pooling to convert the spatial features corresponding to the location of the ROI to a small feature map (Girshick, 2015). In the case of this example, ROI pooling is used to create three feature maps with different aspect ratios ($7 \times 7, 3 \times 11, 11 \times 3$) which are concatenated for the next step (Jiang et al., 2017). The second stage is to predict a confidence score (text, background) for each ROI and to refine them by regressing values (x_1, y_1, x_2, y_2, h) that allow for inclined boxes to account for rotated text (Jiang et al., 2017). At last the resulting BBs are filtered by inclined NMS which is adjusted to the incline BB (Jiang et al., 2017).

The basis for the segmentation based methods is the fact that every part of the text instance can be used to verify that there is text (Long et al., 2021). Because of this, sub-text components can be detected separately and then used to re-construct a text instance (Long et al., 2021). Segmentation based methods can roughly be summed up in two categories: pixel based and component based (Long et al., 2021). Like with BB regression based methods, example

architectures are explained in order to describe their categories more clearly. The paper Deng et al. (2018) introduced a pixel based STD approach. Fig-

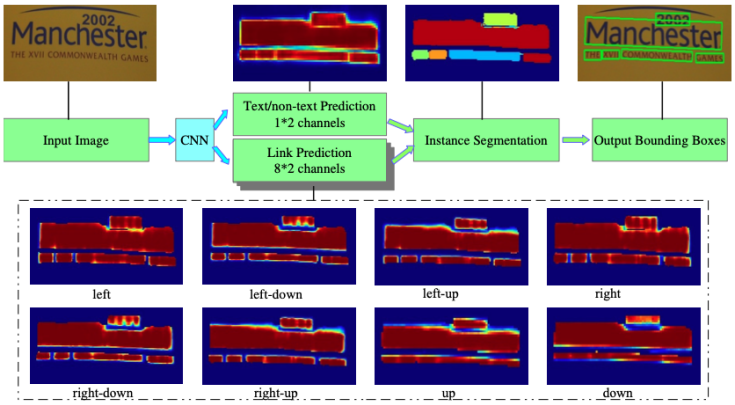


Figure 4.5: Example for a pixel, segmentation based STD architecture (Deng et al., 2018)

ure 4.5 shows the approch's architecture. Figure 4.6 shows the CNN structure for feature extraction (until conv5) which is basically VGG architecture with the fully connected layers exchanged with another convolutional stage (Deng et al., 2018). The feature extraction followed by two heads which either pre-

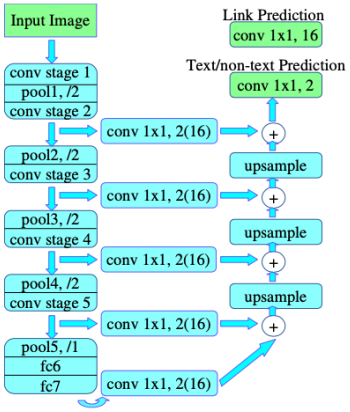


Figure 4.6: PixelNet CNN feature extractor with head structure for pixel segmentation

dict text/non-text or links (Deng et al., 2018). The structure which combines downsampled feature maps with later upsampled ones is inspired by Long et al. (2015). Continuous downsampling and and combining those layers with later, upsampled layers helps to combine coarse, higher level information with fine, lower level information (Long et al., 2015). The upsampling is performed with bilinear interpolation (Deng et al., 2018). Depending on which head is used, the 1×1 convolution layers either have 2 or $2 \cdot 8$ filters. Counted together,

the model has 18 output channels (Deng et al., 2018). The 1×1 convolution layers are also used for upsampling (deconvolution, see Noh et al. (2015); Long et al. (2015) for an in depth explanation) (Deng et al., 2018). The 2 filters are used to predict text/non-text for each pixel, while the other 16 filters predict the links (Deng et al., 2018). The text/non-text head essentially performs semantic segmentation, that is to categorize each pixel to its type (Deng et al., 2018). For every neighbor there is a negative and a positive score. A pixel has eight neighbors: left, left-down, left-up, right, right-down, right-up, up, down. Each of the $2 \cdot 8$ filters is responsiblde for link a neighbor (Deng et al., 2018). After both links and text/non-text pixels have been predicted, they are combined for instance segmentation (Deng et al., 2018). The link layers are used to indicate whether two text pixels are grouped together an thus belong to the same instance (Deng et al., 2018). The output is a dense prediction map with the same spatial structure as the input image (Deng et al., 2018). A bounding box can then be extracted by laying minimum area rectangles over the instances (Deng et al., 2018).

The second segmentation based category for STD segments components which are local regions of text that can overlap one or more characters (Long et al., 2021). The architecture (see Figure 4.7) from Shi et al. (2017b) is used

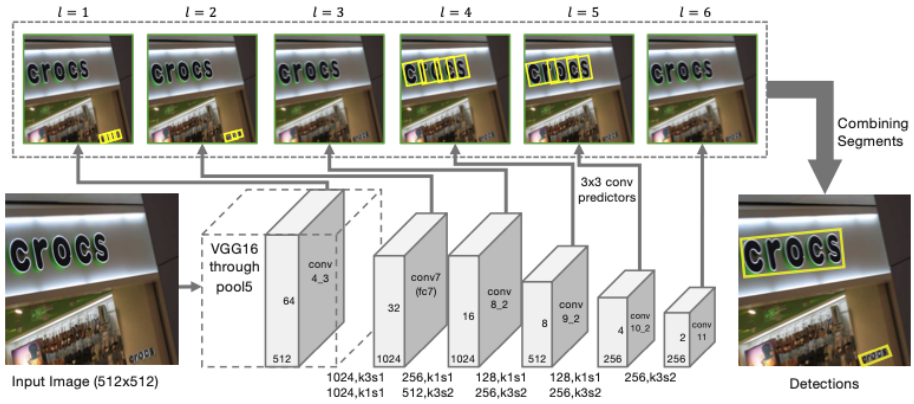


Figure 4.7: Example for a sub-component, segmentation based STD architecture (Shi et al., 2017b)

as an example for this category. Like the one-stage, BB regression based STD approach, the feature extraction CNN of this approach is taken from SSD and thus VGG, the difference is reflected in the prediction layers (Shi et al., 2017b; Liu et al., 2016; Simonyan and Zisserman, 2015). Instead of detecting whole BBs, the networks predicts both subcomponents and links at multiple scales (Shi et al., 2017b). The convolutional prediction is carried out with seven 3×3 filters followed by a softmax nonlinearity for normalization. The segments are given by the values $x_s, y_s, w_s, h_s, \theta_s$ which offset an anchor box as well as confidence scores c_1, c_2 (Shi et al., 2017b). Links are used to separate the

segments and are accordingly used to separate neraby words (Shi et al., 2017b). The prediction layer applies convolution, like with c_1, c_2 for every neighbor and cross layer neighbor a positive and a negative value is predicted (Shi et al., 2017b). Within layer links are predicted for the neighbors of a space in the

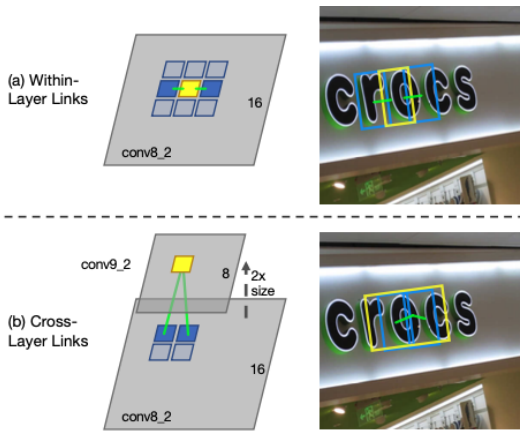


Figure 4.8: Visualization for prediction of links within and cross layers for segmentation based STD (Shi et al., 2017b)

predicted feature map (see Figure 4.8 (a), Equation 4.1) (Shi et al., 2017b).

$$\mathcal{N}^{w}_{s^{x,y,l}} = \frac{\{s^{(x',y',l)}\}_{x-1\leq x'\leq x+1, y-1\leq y'\leq y+1}}{s^{(x,y,l)}} \tag{4.1}$$

The cross layer links on the other hand are predicted by using the 4 cross layer neighors of the feature map of the preceding predictor (Shi et al., 2017b). (see Figure 4.8 (b), Equation 4.2) (Shi et al., 2017b).

$$\mathcal{N}^{c}_{s^{x,y,l}} = \{s^{(x',y',l-1)}\}_{2x\leq x'\leq 2x+1, 2y\leq y'\leq 2y+1} \tag{4.2}$$

These cross layer links are used to connect segments on different scales (Shi et al., 2017b). The network architecture designed so that the preceding feature map is twice the sice (spatial dimensions) of the current which is necessary to extract the right locations (Shi et al., 2017b). Before reconstruction the text instances, segments are filtered by their confidence scores (Shi et al., 2017b). To reconstruct, the predictions are taken as a graph: segments are nodes, links are edges. Depth-first search is applied to the graph to find connected components and thus text instances (Shi et al., 2017b).

For STD two main categories of approaches can be identified: segmentation based and BB regression based (see Figure 4.2) (Long et al., 2021; Sheng et al., 2021; Liu et al., 2020). For STR two main categories of approaches can be identified: segmentation based and segmentation less (Chen et al.,

2021). Segmentation less approaches can again be divided into CTC based and Encoder-Decoder based approaches (see Figure 4.9) (Long et al., 2021; Cong et al., 2019). Like with STD the different approaches are explained with

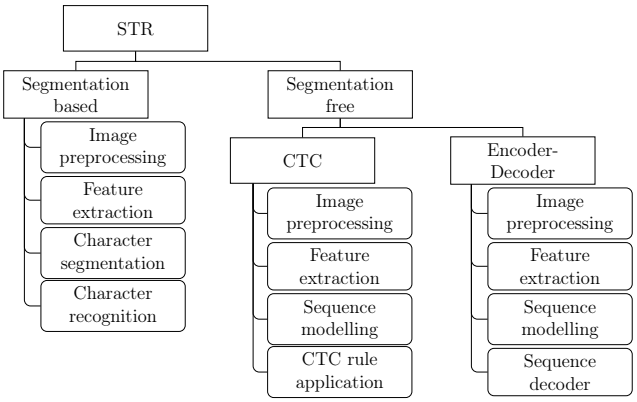


Figure 4.9: Different STR Pipelines

examples. For segmentation based STR the approach introduced by Liao et al. (2018b) (see Figure 4.10) is used.

Figure 4.10: Example for a segmentation based STR architecture (Liao et al., 2018b)

Segmentation less Liao et al. (2018b)

- Steps: Preprocessing, Feature Extraction, Sequence Modelling, Prediction
- preprocessing, text enhancement: remove distortions, background; improve resolution, recover degraded text
  - Spatial Transformer Networks (for distortions)
- feature extraction: encode image into feature space
  - ResNet
    - aggressive downsampling
    - 3×3 best kernel size
    - Gradient saving $\rightarrow$ Res block, Res Bottleneck, Res Grouped

    - batch normalization
    - 32,50,150,...
  - GoogLeNet

- Sequence → Encoder
- Prediction → Decoder (mention improvements made!)
  - CTC — Shi et al. (2017c),
  - Attention — Ghosh et al. (2017) (inspired by Bahdanau et al. (2016); Xu et al. (2016)) Difference Attention and Encoder-Decoder (Long)!

E2E

- trainable as one?
- Bestandteile nicht zwangsweise sequentiell
- two step or two stage pipeline

Task	Approach category		Identifying properties
STD	Seg free		Localize whole instances
		1-stage	direct BB regression
		2-stage	find ROIs, adjust ROIs for better fit
	Seg based		Localize sub text components to reconstruct instance
		Pixel-level	Pixel level segmentation
		Component-level	Sub-component level segmentation
STR	Seg based		Character segmentation and classification
	Seg free		Text instance recognition
		CTC based	
		Attention based	Attention Mechanism
E2E	2-step		
	Parallel		

Table 4.1: Tasks, method categories and identifying propertis

4.2 Literature Search

This section documents the search for literature which provides the content for the subsequent overview of innovation. For this, important DL techniques and notable advances along with their properties are researched and presented. For a literature review, it is important to report how the information was

found and synthesized (Torraco, 2005). The strategy for researching current research is most important for a literature review (Snyder, 2019). This includes databases and keywords that were used, as well as exclusion criteria that were enforced (Torraco, 2005). 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 architecture 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.

4.3 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

Go down hierarchy and compare most important advances

- Compare: two step, two stage
- Compare: innovations for specific categories

Comparing numbers on benchmarks and not looking at quanitative data would result in fallacy.

Task	Approach		Shortcommings
STD	Seg free		Curved text (Long et al., 2021) Higher aspect ratios (Long et al., 2021; Shi et al., 2017b) Text orientation (Shi et al., 2017b)
		1-stage	
		2-stage	
	Seg based		
		Pixel-level	
		Component-level	
		Character-level	
STR	Seg based		Accurate detection of individual characters is hard (Chen et al., 2021) Contextual information between character gets lost (Chen et al., 2021)
	Seg free		
		CTC based	
		Attention based	Sentence recognition
E2E	2-step		
	Parallel		

Table 5.1: Tasks, method categories and their shortcomings

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

Detection

- Long et al. (2018): many methods have strong assumption that text instances are in linear shape and therefore adopted simple representations (rectangles — axis aligned or rotated, quadrangles) → problem with irregular and curved text
- Long et al. (2018): some models are specifically designed for arbitrary shapes
- Shi et al. (2017b): modular approach instead of whole BBs → good with long aspect ratio and orientation because of flexibility

Recognition

- Ability to cope with 2d text: CTC has problems, Attention/Encoder-Decoder based can be extended to work
- Segmentation based
  - Xie et al. (2019): require separate training targets for each segment or time-step in the input sequence → inconvenient pre-segmentation and post-processing stages
- Sequence based
  - Xie et al. (2019): no need for separate training targets for each segment or time step
  - CTC
    - Chen et al. (2021):CTC prone to overfitting
    - Xie et al. (2019): implementation of forward-backward algorithm is complicated → large computation consumption

    - Xie et al. (2019): can hardly be applied to 2D prediction problems
  - Attention
    - Chen et al. (2021): good for isolated word recognition
    - Chen et al. (2021): Attention has problems with long sequences
    - Xie et al. (2019): relies on attention module for label alignment → large storage requirement and computation consumption
    - Xie et al. (2019): misalignment problem can confuse and mislead training problem, leading to degradation of accuracy
    - Xie et al. (2019): can be adopted to 2D prediction but memory and time consumption are to big then
  - look into Cong et al. (2019) for CTC — Attention comparison

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

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

Chapter 6

Conclusion

Bibliography

F. Imgrund, M. Fischer, C. Janiesch, and A. Winkelmann, *Approaching Digitalization with Business Process Management*, Mar. 2018.

D. L. Goodhue, M. D. Wybo, and L. J. Kirsch, "The Impact of Data Integration on the Costs and Benefits of Information Systems," *MIS Quarterly*, vol. 16, no. 3, p. 293, Sep. 1992. [Online]. Available: https://www.jstor.org/stable/249530?origin=crossref

W. Abramowicz and R. Corchuelo, Eds., *Business Information Systems: 22nd International Conference, BIS 2019, Seville, Spain, June 26–28, 2019, Proceedings, Part II*, ser. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2019, vol. 354. [Online]. Available: http://link.springer.com/10.1007/978-3-030-20482-2

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

X. Chen, L. Jin, Y. Zhu, C. Luo, and T. Wang, "Text Recognition in the Wild: A Survey," *ACM Computing Surveys*, vol. 54, no. 2, pp. 1–35, Apr. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3440756

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

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

S. Long, X. He, and C. Yao, "Scene Text Detection and Recognition: The Deep Learning Era," *International Journal of Computer Vision*, vol. 129, no. 1, pp. 161–184, Jan. 2021. [Online]. Available: https://link.springer.com/10.1007/s11263-020-01369-0

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

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

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

R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges," *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–39, Jun. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3453444

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

H. Snyder, "Literature review as a research methodology: An overview and guidelines," *Journal of Business Research*, vol. 104, pp. 333–339, Nov. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0148296319304564

R. J. Torraco, "Writing Integrative Literature Reviews: Guidelines and Examples," *Human Resource Development Review*, vol. 4, no. 3, pp. 356–367, Sep. 2005, publisher: SAGE Publications. [Online]. Available: https://doi.org/10.1177/1534484305278283

F. Cong, W. Hu, Q. Huo, and L. Guo, "A Comparative Study of Attention-Based Encoder-Decoder Approaches to Natural Scene Text Recognition," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, Sep. 2019, pp. 916–921, iSSN: 2379-2140.

L. Qiao, S. Tang, Z. Cheng, Y. Xu, Y. Niu, S. Pu, and F. Wu, "Text Perceptron: Towards End-to-End Arbitrary-Shaped Text Spotting," *arXiv:2002.06820 [cs]*, Oct. 2021, arXiv: 2002.06820. [Online]. Available: http://arxiv.org/abs/2002.06820

T. Sheng, J. Chen, and Z. Lian, "CentripetalText: An Efficient Text Instance Representation for Scene Text Detection," *arXiv:2107.05945 [cs]*, Oct. 2021, arXiv: 2107.05945. [Online]. Available: http://arxiv.org/abs/2107.05945

X. Liu, G. Zhou, R. Zhang, and X. Wei, "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)*. Seattle, WA, USA: IEEE, Jun. 2020, pp. 2344–2352. [Online]. Available: https://ieeexplore.ieee.org/document/9151062/

D. Deng, H. Liu, X. Li, and D. Cai, "PixelLink: Detecting Scene Text via Instance Segmentation," *arXiv:1801.01315 [cs]*, Jan. 2018, arXiv: 1801.01315. [Online]. Available: http://arxiv.org/abs/1801.01315

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

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

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

G. James, D. Witten, T. Hastie, and R. Tibshirani, Eds., *An introduction to statistical learning: with applications in R*, ser. Springer texts in statistics. New York: Springer, 2013, no. 103, oCLC: ocn828488009.

J. Alzubi, A. Nayyar, and A. Kumar, "Machine Learning from Theory to Algorithms: An Overview," *Journal of Physics: Conference Series*, vol. 1142, p. 012012, Nov. 2018. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1142/1/012012

Y. Ho and S. Wookey, "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020, conference Name: IEEE Access.

M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, "Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, Oct. 2017, pp. 17–41, iSSN: 2474-0705.

A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019, conference Name: IEEE Access.

A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar. 2020, arXiv: 1808.03314. [Online]. Available: http://arxiv.org/abs/1808.03314

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, arXiv: 1503.04069. [Online]. Available: http://arxiv.org/abs/1503.04069

O. K. Oyedotun, E. O. Olaniyi, and A. Khashman, "Deep Learning in Character Recognition Considering Pattern Invariance Constraints," *International Journal of Intelligent Systems and Applications*, vol. 7, no. 7, pp. 1–10, Jun. 2015. [Online]. Available: http://www.mecs-press.org/ijisa/ijisa-v7-n7/v7n7-1.html

L. Boué, "Deep learning for pedestrians: backpropagation in CNNs," *arXiv:1811.11987 [cs, stat]*, Nov. 2018, arXiv: 1811.11987. [Online]. Available: http://arxiv.org/abs/1811.11987

K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385

S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, conference Name: Neural Computation.

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

Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019. [Online]. Available: https://direct.mit.edu/neco/article/31/7/1235-1270/8500

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv:1706.03762 [cs]*, Dec. 2017, arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv:2010.11929 [cs]*, Jun. 2021, arXiv: 2010.11929. [Online]. Available: http://arxiv.org/abs/2010.11929

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

F. Liu, X. Ren, Z. Zhang, X. Sun, and Y. Zou, "Rethinking Skip Connection with Layer Normalization in Transformers and ResNets," *arXiv:2105.07205 [cs]*, May 2021, arXiv: 2105.07205. [Online]. Available: http://arxiv.org/abs/2105.07205

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

W. Su, Y. Yuan, and M. Zhu, "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*. Northampton Massachusetts USA: ACM, Sep. 2015, pp. 349–352. [Online]. Available: https://dl.acm.org/doi/10.1145/2808194.2809481

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

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe,

and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

M. Liao, B. Shi, and X. Bai, "TextBoxes++: A Single-Shot Oriented Scene Text Detector," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3676–3690, Aug. 2018, arXiv: 1801.02765. [Online]. Available: http://arxiv.org/abs/1801.02765

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

B. Shi, C. Yao, M. Liao, M. Yang, P. Xu, L. Cui, S. Belongie, S. Lu, and X. Bai, "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, Nov. 2017, pp. 1429–1434, iSSN: 2379-2140.

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

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

M. Liao, G. Pang, J. Huang, T. Hassner, and X. Bai, "Mask TextSpotter v3: Segmentation Proposal Network for Robust Scene Text Spotting," *arXiv:2007.09482 [cs]*, Jul. 2020, arXiv: 2007.09482. [Online]. Available: http://arxiv.org/abs/2007.09482

N. Nayef, F. Yin, I. Bizid, H. Choi, Y. Feng, D. Karatzas, Z. Luo, U. Pal, C. Rigaud, J. Chazalon, W. Khlif, M. M. Luqman, J.-C. Burie, C.-l. Liu, and J.-M. Ogier, "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, Nov. 2017, pp. 1454–1459, iSSN: 2379-2140.

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

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

C. K. Ch'ng and C. S. Chan, "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, Nov. 2017, pp. 935–942, iSSN: 2379-2140.

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

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

K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, L. Joeckel, J. Siebert, and J. Heidrich, "Requirements-Driven Method to Determine Quality Characteristics and Measurements for Machine Learning Software and Its Evaluation," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, Aug. 2020, pp. 260–270, iSSN: 2332-6441.

D. Zowghi, Z. Jin, S. D. Junqueira Barbosa, P. Chen, A. Cuzzocrea, X. Du, J. Filipe, O. Kara, I. Kotenko, K. M. Sivalingam, D. Ślęzak, T. Washio, and X. Yang, Eds., *Requirements Engineering*, ser. Communications in Computer and Information Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, vol. 432. [Online]. Available: http://link.springer.com/10.1007/978-3-662-43610-3

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

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

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

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

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

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016, pp. 779–788. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv:1311.2524 [cs]*, Oct. 2014, arXiv: 1311.2524. [Online]. Available: http://arxiv.org/abs/1311.2524

M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A Fast Text Detector with a Single Deep Neural Network," in *Thirty-First AAAI Conference on Artificial Intelligence*, Feb. 2017. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14202

K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015, arXiv: 1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556

J. Hosang, R. Benenson, and B. Schiele, "Learning Non-maximum Suppression," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 6469–6477. [Online]. Available: http://ieeexplore.ieee.org/document/8100168/

R. Girshick, "Fast R-CNN," *arXiv:1504.08083 [cs]*, Sep. 2015, arXiv: 1504.08083. [Online]. Available: http://arxiv.org/abs/1504.08083

S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv:1703.06870 [cs]*, Jan. 2018, arXiv: 1703.06870. [Online]. Available: http://arxiv.org/abs/1703.06870

Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo, "R2CNN: Rotational Region CNN for Orientation Robust Scene Text Detection," *arXiv:1706.09579 [cs]*, Jun. 2017, arXiv: 1706.09579. [Online]. Available: http://arxiv.org/abs/1706.09579

J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *arXiv:1411.4038 [cs]*, Mar. 2015, arXiv: 1411.4038. [Online]. Available: http://arxiv.org/abs/1411.4038

H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, Dec. 2015, pp. 1520–1528. [Online]. Available: http://ieeexplore.ieee.org/document/7410535/

B. Shi, X. Bai, and S. Belongie, "Detecting Oriented Text in Natural Images by Linking Segments," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 3482–3490. [Online]. Available: http://ieeexplore.ieee.org/document/8099854/

M. Liao, J. Zhang, Z. Wan, F. Xie, J. Liang, P. Lyu, C. Yao, and X. Bai, "Scene Text Recognition from Two-Dimensional Perspective," *arXiv:1809.06508 [cs]*, Nov. 2018, arXiv: 1809.06508. [Online]. Available: http://arxiv.org/abs/1809.06508

B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv:1409.0473 [cs, stat]*, May 2016, arXiv: 1409.0473. [Online]. Available: http://arxiv.org/abs/1409.0473

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," *arXiv:1502.03044 [cs]*, Apr. 2016, arXiv: 1502.03044. [Online]. Available: http://arxiv.org/abs/1502.03044

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

S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, "TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, vol. 11206, pp. 19–35, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-01216-8_2

Z. Xie, Y. Huang, Y. Zhu, L. Jin, Y. Liu, and L. Xie, "Aggregation Cross-Entropy for Sequence Recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, Jun. 2019, pp. 6531–6540. [Online]. Available: https://ieeexplore.ieee.org/document/8953200/

S. Long and C. Yao, "UnrealText: Synthesizing Realistic Scene Text Images from the Unreal World," *arXiv:2003.10608 [cs]*, Aug. 2020, arXiv: 2003.10608. [Online]. Available: http://arxiv.org/abs/2003.10608

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

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	**Sources**
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	**Sources**
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	**Sources**
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	**Sources**
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