CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Generalization abilities of scene text detection models

Analysis of the generalization abilities of scene text detection models EAST and DBnet, in the context of intelligence collection.

Master's thesis in Computer science and engineering

Andreas F.T. Gjesdal

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Generalization abilities of scene text detection models

Analysis of the generalization abilities of scene text detection models EAST and DBnet, in the context of intelligence collection.

Andreas F.T. Gjesdal

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Generalization abilities of scene text detection models
Analysis of the generalization abilities of scene text detection models EAST and DBnet, in the context of intelligence collection.
Andreas F. T. Gjesdal

Supervisors: Tobias Norlund, Department of Computer Science and Engineering
Richard Johansson, Department of Computer Science and Engineering

Advisors: Mats Kvarnström, Recorded Future
Anders Hansson, Recorded Future

Examiner: Richard Johansson, Department of Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2022

Generalization abilities of scene text detection models
Analysis of the generalization abilities of scene text detection models EAST and DBnet, in the context of intelligence collection.
Andreas F.T. Gjesdal
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The field of scene text detection has seen massive improvements in the last year with the introduction of models that are based on deep convolutional networks. State-of-the-art performance on certain benchmark datasets is getting close to human capabilities of detecting text. The question of how well these text detection models can generalize to detect text in images from different domains is of high interest for tasks where a high variety of images are included. This thesis performs an analysis of the generalization abilities of two scene text detection models, EAST and DBnet by training various instances of both models on different combinations of benchmark dataset and evaluating the performance on several datasets which are both used for training and unseen during training. The results show that both models are able to generalize the text detection to a certain degree with instances of both models achieving an average f1-score >0.6 on a selection of benchmark datasets. Both models also achieved f1-scores >0.6 on a set of images collected from social media provided by *Recorded Future* which was not used for training. Results from some of the easier benchmark datasets are, however, not indicative of performance in a highly varied domain. Finally, it was showed that both models perform quite well as classifiers of whether an image contains text or not.

Keywords: Scene text detection, computer vision, deep neural network, convolutional network.

Acknowledgements

I would like to thank *Recorded Future* and specifically Mats Kvarnström and Anders Hansson for giving me the opportunity to write this thesis and for help and fruitful discussions during my work.

Thanks also to my academic supervisors Tobias Norlund, for help and advice during my work, and to Richard Johansson for providing helpful feedback on my writing.

Big thanks to my parents for always encouraging and supporting me during my entire education and helping me keep my cool during my thesis work.

Thanks to my brilliant cousin, Sofia Thomaz, for providing me with illustrations that are included in this thesis.

Andreas F.T. Gjesdal, Gothenburg, August 2022

Contents

List of Figures

List of Tables

1
Introduction

Text is one of humankind's great creations. It is an immensely effective tool for communicating and storing information. The invention of the internet has dramatically increased the accessibility of much of the combined knowledge and information that exist in the world. Within a few seconds one can search for and find information about practically any question one can think of or current events. The internet space mainly consists of textual data, especially in earlier years. In addition to text, visual media has gradually become a bigger presence in the internet space over the last decade. The rise of popularity of smartphones, with a built-in camera and internet connection, and social media platforms where it is possible to share images and videos has fuelled the increase of visual media as a way to report information by anyone practically in real time.

Visual media can also contain textual information if there are text instances present in the images or videos. Text in visual media can be very easy to spot and read for the human eye, but as it is not stored as text it is not possible to extract the information digitally in a simple, efficient way and it is not searchable like textual data is. In this context it is interesting to be able to automatically detect, recognize and store textual information from images. The field of research working on this task is called *scene text recognition*. End-to-end scene recognition pipelines usually comprise of two main steps, *scene text detection* and *text recognition*. This thesis will focus on scene text detection and examine how it could work as a component in an end-to-end system for the task of extracting textual information from images collected on the internet in order to store it and make it searchable. The work will be conducted in collaboration with *Recorded Future*.

Recorded Future is the world's largest intelligence company with over 1400 customers from 66 different countries. They provide intelligence to their customers from a wide variety of sources on the internet. Recorded Future stores enormous amount of data in order to do this and an increasing proportion of the data comes in the form of visual media. It is a challenge that potentially interesting information from textual information inside visual media can not be analyzed in a traditional way like normal textual data. In this context, Recorded Future is looking into the possibilities and challenges of potentially implementing an in-house scene text detection pipeline. The work in this project is the first step of looking into this by focusing on the first step in an end-to-end pipeline, namely, the text detection step.

1.1 Scene text detection

Scene text detection and recognition is a popular and important research field within computer vision. There exist several review papers [15][13][18] and numerous different algorithms and approaches[31][10][2][5]. The task of detecting and recognizing text in images is particularly challenging due to complex backgrounds and high variability in text font, color, size, etc. Like in most computer vision fields, there has been a dramatic improvement in performance of scene text detection algorithms in the last decade which can be attributed to the introduction of deep learning based algorithms. This dramatic improvement can be exemplified by the results in the ICDAR 2015 competition where the winner, in 2015, had a f1-score of 0.49, while the best result registered to date has a f1-score of 0.92, and a multitude of methods in the high 0.8s[1]. It is often common to refer to the "scene text detection" field as before and after the "deep learning era".

Early text detection methods before the "deep learning era" mainly relied on handcrafted features from the images such as e.g. connected component analysis methods and stroke based methods. These methods perform reasonably well under certain circumstances but struggle with more challenging natural images with a higher variety in background, color and text size.

All of the methods that are state-of-the-art, or close, in terms of performance, are deep learning based methods. The existing methods can be divided into two main categories, *region proposal methods* and *segmentation methods*. Region proposal based methods produce rectangle bounding boxes with confidence scores. Many different cropped regions from the image are fed to the network and the network predicts whether the region that is fed is an appropriate bounding box for text instances in the image. This leads to quite heavy computation cost and the bounding box geometry is restricted to the different ways that the image is cropped and fed to the network.

Segmentation based models, on the other hand, predict text on a pixel level. They only need to pass the whole image through the network once and output a mapping of the image where each pixel is given a probability of containing text. This mapping is used to group pixels and produce bounding boxes that can have more flexible geometries than region proposal based methods.

Most papers presenting text detection algorithms present the results of algorithms which are trained and tested on specific benchmark datasets. These datasets have often been collected and presented with specific challenge in mind, such as multi-oriented text, incidental text and synthetic text overlaid on natural images. These

[1] https://rrc.cvc.uab.es/?ch=4com=evaluationtask=1

performances are used for showing the potential of different algorithms and comparing them, but do not necessarily reflect how well they generalize to target domains where we expect to encounter a variety of these instances. Good generalization abilities of text detection methods are valuable for real-world use-cases of scene text recognition such as instant translation and augmented computer vision e.g. in autonomous cars which should perform stable results under different conditions and perspectives. Examining the generalization abilities of scene text detection models is identified as an important area for future work by [13]. The generalization abilities of the models is also highly relevant for the task faced at Recorded Future where the images come from all kinds of different sources and are highly varied.

1.2 Problem statement

This project aims to examine the generalization abilities of two widely used scene text detection algorithms, EAST[31] and Differential Binarization(DBnet)[10]. Both algorithms are segmentation based deep-learning algorithms and both algorithms have also been implemented in open source projects which will be used to conduct this study[2,3]. In the literature, most algorithms are presented exclusively with results obtained on benchmark datasets. This only gives an idea of the potential of the algorithms but leaves the question of generalization abilities of the algorithms in the open. In many real life scenarios annotated data is hard to come by and can be expensive to produce and thus it is of interest to see how algorithms such as EAST and DBnet can perform on unknown target domains with training limited to openly available datasets. This project will examine how the methods perform in an unknown target domain with different data used for training. EAST and DBnet will be trained on various benchmark datasets and then each instance of the models will be evaluated on the testing dataset of all the datasets used in this study, as well as a datasets of Recorded Future's own data. This will provide insight into which of the existing benchmark dataset that result in the best model when we consider the ability of generalization and also if pooling existing datasets lead to improved generalization abilities.

Specifically, there are two use cases for the data provided by Recorded Future that will be examined. The first one is how well the algorithms detect the actual text instances present in the images. This examines how well the models can perform as a component of an end-to-end text recognition pipeline. For an end-to-end pipeline to perform well it is critical that the text is accurately detected. The second use case is how well the models perform as a classifier of whether an image contains text or not. I.e. if the models correctly can predict no bounding boxes in images which do not contain text and at least one bounding box in images which contain text. This can be valuable if the end-to-end pipeline is complex or expensive in order to filter out images which do not need to be analyzed.

[2] `https://github.com/argman/EAST`
[3] `https://github.com/open-mmlab/mmocr`

1.3 Related work

There is not much work published on the subject of generalization abilities of scene text detection models in the literature. There are a few models which have been presented with some limited analysis of their generalization abilities.

Baek et al. [2] present some analysis of the generalization abilities of their proposed scene text detector model, CRAFT. CRAFT is a segmentation model which detects text by help of weakly supervised character-level annotations. The model is first pre-trained on synthetically generated data before fine tuning on real data. The model produces state-of-the-art performances on several common benchmark datasets. The weakly supervised training process does not work with polygon annotations and thus the authors fine-tuned CRAFT on two benchmark datasets with rectangular annotations and evaluated them on two benchmark datasets, known for including curved text instances. CRAFT performed as good as, or better than, three methods that they compare themselves to in the paper. The performance of CRAFT on these two datasets is only a few precentage points behind the current state-of-the-art indicating that the model has good abilities to generalize to curved text.

Long et al. [14] present a similar study of generalization abilities of their method. The authors have trained their model and three other models[31][21][5] on the dataset "ICDAR2015" and evaluate the models on the dataset called TotalText. Their study show that TextSnake performs the best of the models they compare with, but the recall and precision is close to 20 percentage points worse than the model achieves when it is trained and evaluated on the same dataset, both for ICDAR2015 and TotalText. This drop in performance indicates that the model's abilities to generalize over different domains are quite limited and similar drop in performance was seen for all models.

End-to-end scene text recognition algorithms and scene text detection models are often either pretrained on synthetic data and only fine-tuned on the real data or based on convolutional backbone networks which are pretrained on ImageNet (an object detection dataset containing more than 14 million images). The reason for this is that deep learning models require a lot of data to train from scratch and there is limited real world datasets for scene text recognition available. Baek et al. [1] pooled all existing openly available real world datasets in order to see if an end-to-end scene text recognition algorithms could be trained exclusively on real world data. The results in the study show that a scene text recognition model trained on a pooled combination of all the real world produces state-of-the-art performance over six benchmark datasets. The study does not specify why only six datasets was used for evauation. This study show that pooling datasets can lead to improved generalization abilities and that it potentially can eliminate the need for pretraining

on synthetic datasets or larger object detection datasets.

1.4 Limitations

The scope of this project will be limited to examine the performance and generalization abilities of EAST and DBnet. Further work will be needed to examine whether the findings of this project can be generalized over other scene text detection algorithms. The models will be trained and studied as they are implemented in their respective open source project. The aim of this project is not to optimize the performance of the models but rather compare model instances under equal circumstances.

The scope of this project is also limited to mainly detect text written in Latin script. There are a few instances of text in other scripts in the training and evaluation data, but the overwhelming majority of text instances are in Latin script.

2

Theory

This chapter will explain the theory behind the work that has been performed in this project. Firstly, relevant concepts that are important to understand the models that are used are explained. Secondly, the two models, EAST and DBnet, are presented in detail. The last parts of this chapter will present the evaluation metrics that are used to evaluate the performance of the models and the challenges related to evaluating text detection performance.

2.1 Key concepts

This section will present and describe the concepts which are necessary to understand the methods and algorithms that are used in the work with this thesis.

2.1.1 Artificial neural network

Artificial neural networks (ANN) are mathematical computing systems that are designed with inspiration from how biological brains work. The research about ANNs is a subfield within machine learning. ANNs consist of nodes, also called neurons, that are connected and can be trained to learn and perform a wide variety of tasks. The concept of ANNs was presented in 1943 by Warren McCulloch and Walter Pitts [16]. Since then there has been a varying amount of interest in the subject, but the last decade has seen a great rise in use and research in the field. This is mainly due to the success of deep learning algorithms in a variety of tasks such as language translation, speech recognition and object detection. Most neural networks in use today are feed-forward networks, this means that the network is provided with input, which is propagated through the network to produce an output. The neurons are usually structured in layers and deep-learning is a term for neural networks with many layers. Deep-learning networks often have several million nodes and trainable parameters.

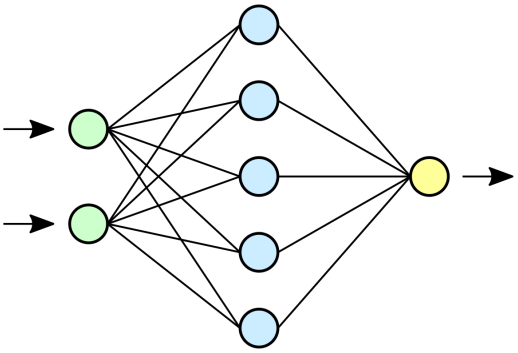

Figure 2.1: Example of a simple ANN.

In a feed-forward network connections between neurons are only one-way connections. Each connection between nodes is assigned a weight and each neuron is assigned a threshold. The information that is passed on to the receiver neuron in each connection is the value of the sender neuron multiplied by the weight. The values transmitted in all incoming connections is added and the threshold is subtracted from the sum. An activation function such as ReLu or Sigmoid is then applied to the resulting value to produce the value or state of the current neuron. The weights and thresholds are the trainable parameters that are determined by training the network.

2.1.2 Convolutional neural networks

There exist many different architectures for constructing ANNs. One of the most widely used architectures in tasks involving image analysis is convolution. Instead of each connection having an individual weight assigned to it the weights in convolutional networks are set in a kernel of given size and this kernel is applied to the layer. One example of this can be viewed in Figure 2.2. The advantage of this architecture is that it condenses information from neighbouring nodes and can thus capture local patterns in the image.

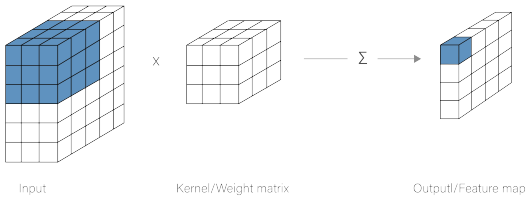

Figure 2.2: Example showing the convolution operation. The shaded nodes in the input are multiplied element-wise with the convolution kernel and the results are summed and passed on to the shaded output node.

When denoting convolutional layers they are typically referred to by the size of the spatial dimensions, i.e. height and width of the kernel. The kernel also has a third dimension which is the number of channels in the network. The kernel pictured in

figure 2.2 has spatial dimensions of 3, so it would be called a 3x3 convolution layer. However since the layer that is being performed convolution on has three channels the complete kernel size is 3x3x3. The output of this kernel is a layer with only 1 channel and the spatial dimensions are determined by the spatial dimensions of the kernel as well as the step size, i.e. how many nodes the kernel is moved before calculating the neighboring output nodes value. Typically a convolution layer will reduce the spatial dimensions of the output layer compared to the preceding layer, but this can be avoided by padding the preceding layer with zeros around the edges. The spatial dimensions of the output layer when the input layer and the kernel has quadratic spatial dimensions can be calculated by the formula $o = (i - k + 2p)/s + 1$ where i is the input layer size, k is the kernel size, p is the size of the zero padding around i and s is the step size.

2.1.2.1 1x1 convolution

The 1x1 convolution is a special case because it obviously does not use information from neighbouring nodes to learn geometric patterns in the data. It is used however to control the number of channels in the network. Most often it is used to reduce the number of channels in order to reduce the computational cost of a network. Since one kernel produces one output channel one can select the number of channels that is desired and use the same number of kernels to let the network itself reduce the channels. This way the network can learn the optimal way to condense the information in the original number of channels to the desired number of channels. The spatial dimensions remain unchanged.

2.1.2.2 Deformable convolution

Another special case of convolution is deformable convolution which was first introduced by [4]. Instead of having the kernel dimensions fixed only the number of nodes used in the kernel is fixed, and an offset determines which nodes are used for the convolution. Figure 2.3 shows how the deformable convolution works to make the convolution field more flexible. The offset is also a trainable parameter so that the network learns how to adapt the convolution field optimally during training. All the convolution layers in DBnet (described more in detail in section 2.3) are deformable convolution layers.

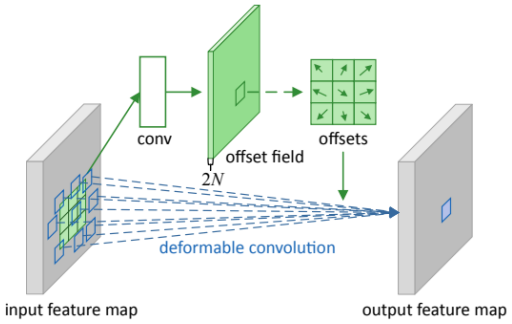

Figure 2.3: En example of how the deformable convolution have a more flexible convolution field. Taken from [4].

2.1.3 Pooling layers

When convolution layers extract features from data the resulting feature maps can be highly location dependent. The network can learn to associate features with specific locations in the image and this can lead to reduced performance. Ideally the network should capture high-level features so that the network is translation invariant. This means that the network should not use the position of text in order to predict it, but rather the inherent properties of text itself independent of position, orientation or size. One way to reduce the location-dependency is to implement pooling layers. Similarly to convolution layers, pooling layers also use kernels with specified sizes, such as 2×2 or 3×3. Instead of the kernel consisting of trainable weights, the kernel in pooling layers simple define the nodes on which to perform a pooling operation such as max-pooling or average pooling. For max-pooling operations the output value is simply the highest value of the nodes that are encompassed by the kernel. Average pooling outputs the average value of the nodes encompassed by the kernel.

2.1.4 Training ANNs

Training a neural network is, simply put, an optimization problem where the objective is to find the optimal weights and thresholds for the network for a given task. For complex networks with many trainable parameters the optimization problem gets very big and computationally heavy. In most use cases the training data is labelled with correct output such that the network is fed input data and the output of the network is compared to the correct solution. This comparison happens in the form of a loss function. The loss function is designed in a way that it is the lowest when the network always produces the correct output. The optimization problem is thus to minimize the loss function.

As mentioned previously, the parameters that are subject to the optimization are the weights and the threshold. This is done by calculating the gradient of the loss function with respect to each parameter iteratively backwards from the output, this is called *backpropagation*. A gradient descent algorithm is used to optimize the trainable parameters.

2.1.5 Segmentation networks

Instance segmentation networks are networks which perform pixel-wise predictions of some class/classes in an image. The field of ANNs that performs object detection mainly consists of networks which produce rectangular non-angled bounding boxes but this output geometry can in some cases be too restrictive. Segmentation networks on the other hand can produce predictions of arbitrary shapes since the prediction is performed on a pixel level. This is very common in medical image analysis. In the case of text detection this can also be beneficial as text can come in many orientations and can even be curved. The predictions of the networks comes in the form of a score map.
A score map is a pixel-wise mapping of the input data, this means that the output has the same spatial dimensions as the input, which indicates on a pixel level what class each given pixel belongs to. For the task of text detection the model performs a binary classification of text/non-text in the input image. The values in the score map take the form of a probability distribution. The value of each pixel corresponds to the predicted probability that the specific pixel is part of a text instance or not. Bounding boxes are extracted using regression on the score map and can be of arbitrary shapes. Some segmentation networks also output bounding box candidates for each pixel in separate output channels. This is the case for EAST (discussed more in detail in section 2.2) and in this case the score map must be viewed more as a probability that the candidate bounding box contains text. This output format also requires some post-processing to filter the bounding box candidates.

2.1.6 Feature extraction with deep neural networks

The success of deep learning based methods in many image analysis tasks is mainly due to the capability of deep convolutional networks to extract relevant features, such as patterns and correlations, from images. Most state-of-the-art models for image analysis rely on predefined deep, fully convolutional, networks to perform feature extraction. These networks usually have the same overlying structure and due to this, the models are compatible with several such feature extraction networks. The networks are structured into convolution blocks which can contain variable layers of convolution. This is very useful for modifying the models for use in different tasks. If performance is the main priority it is most likely better to implement a big, complex feature extraction network, while if speed is prioritized then a more lightweight feature extraction network is desirable.

One of the limiting factor for network depth has been what is known as the vanishing gradient problem. In essence, parameters that are many layers removed from the output would get an extremely small gradient during backpropagation and this would hinder the training. One solution to this problem is known as residual connections. Residual connections are connections in the network that skip layers using a simple identity mapping of the preceding layer. A layer which has inputs from the preceding convolution layer as well as a residual connection will receive an element wise addition of the two sources as its input. This connection can be seen in Figure

2.4 and it contributes to a shorter connection from the parameters in early layer to the output of the model.

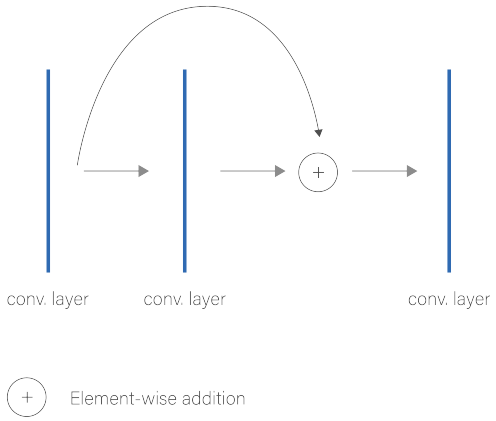

Figure 2.4: Example of a simple residual connection in a convolutional network. The layer to the right receives an elementwise addition of the output from the preceding layer and the output from the source of the residual connection.

Table 2.1: The structure of ResNet50 and PVANet. Convolutional layers are denoted as "$X \times X, C$", where X is the spatial dimension of the kernel, and C is the number of channels in the output.

	ResNet50	PVANet
Convolutional block 1	7×7,64,stride=2	7×7, 16
Conv. block 2	3×3, max pool, stride=2 $\begin{pmatrix} 1\times1,64 \\ 3\times3,64 \\ 1\times1,256 \end{pmatrix} \times 3$	3× 3, max pool, stride=2 $\begin{pmatrix} 1\times1,24 \\ 3\times3,24 \\ 1\times1,64 \end{pmatrix} \times3$
Conv. block 3	$\begin{pmatrix} 1\times1,128 \\ 3\times3,128 \\ 1\times1,512 \end{pmatrix} \times4$	$\begin{pmatrix} 1\times1,48 \\ 3\times3,48 \\ 1\times1,128 \end{pmatrix} \times4$
Conv. block 4	$\begin{pmatrix} 1\times1,256 \\ 3\times3,256 \\ 1\times1,1024 \end{pmatrix} \times6$	Inception block,256 ×4
Conv. block 5	$\begin{pmatrix} 1\times1,512 \\ 3\times3,512 \\ 1\times12048 \end{pmatrix} \times3$	Inception block, 384 ×4

2.1.6.1 Residual networks(ResNet)

ResNets presented by [6] is a widely used type of deep convolutional networks. There exist several variants with varying depth/number of layers. The models in this project are implemented with ResNet50 as the backbone feature extraction

network, ResNet50 is 50 layers deep. The network consists of 5 convolution blocks which are constructed using 3x3 bottleneck building blocks. In ResNet50 each of the bottleneck building block contain three convolutional layers; the middle layer is a 3x3 convolutional layer and there is a 1x1 convolutional layer before and after. The bottleneck building blocks can be seen in Figure 2.5. The first 1x1 convolution layer is used to reduce the dimensionality of the input to the 3x3 convolution by reducing the number of channels, and the final layer is used to restore the number of channels. The 3x3 convolutions are performed with zero padding (padding=1) and stride length 1 to maintain the feature map size within the convolution blocks. The full architecture of ResNet50 can be seen in the middle column in Table 2.1. The network also has residual connections passing over every two building blocks from convolution block 2 all the way to the final convolution layer. The first layer of 3x3 convolution in conv. block 3, 4 and 5 is performed with a stride length of 2 in order to reduce the feature map size by a factor of 2.

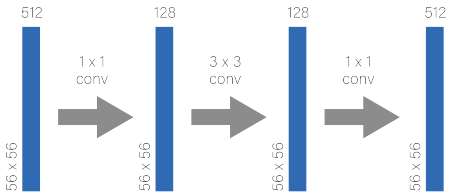

Figure 2.5: An illustration of how the bottleneck building blocks reduce the number of channels before the 3×3 convolution is applied. The building block depicted in this figure is a building block from convolution block 3, where the number above each layer shows the number of channels and the dimensions on the left of each layer is the spatial dimensions of the feature maps.

2.1.6.2 PVANet

PVANet presented by [8] was designed as a lightweight deep neural network to reduce the computational costs relative to the widely used ResNets [8]. The computational complexity is reduced by implementing more 1x1 convolution layers to keep the number of channels low. The corresponding loss in performance is counteracted by implementing more complex modules compared to the standard convolution layers in ResNets. These modules are called *Concatenated Rectified Linear Units* (C.relu) and *inception modules*. C.Relu layers are implemented in the early stages and inception modules in the deeper layers. C.Relu layers are taking advantage of an observation that nodes in early convolution layers tend to be paired, such that channel's output often is mirrored by a channel with the negation of that output[20]. Thus the number of convolution channels are halved and the output is concatenated by the negated output from the remaining channels. Implementing C.Relu layers in the first layers of a network has shown improved performance compared to normal Relu layers in other existing networks [20], and it reduces the computational cost. Inception modules compute convolutions with different kernel sizes in parallel in order to allow the network to learn spatial features at different scales in each layer

throughout the network [22][23]. The inception modules used in PVANet can be seen in Figure 2.6.

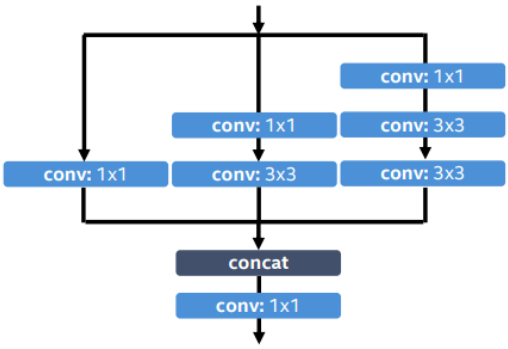

Figure 2.6: The inception blocks used in PVANet. Each path happens in parallel in terms of depth in the network, but the kernel sizes vary. Figure taken from [8].

2.1.7 Upscaling feature maps

The feature maps that are extracted by the convolutional networks have decreasing spatial dimensions. It is often necessary to increase the dimensions of a feature map, for example if two feature maps from subsequent convolution blocks are to be merged or concatenated. There are several ways to achieve this increase in spatial dimensions for a feature map, such as deconvolution and simply resizing the matrices.

Deconvolution or transpose convolution, as it is also called, is the transpose of the normal convolution operation. Each channel in the input is assigned a 2d-kernel and each node in the spatial dimension of the input is multiplied with the whole kernel corresponding to its channel. The resulting values in each channel are summed element-wise to collapse the number of channels to one and added to the output nodes corresponding to each node in the kernel to produce one output channel. This way the dimensions of the feature map is increased. Equation 2.1 show the resulting output dimensions, assuming square layers. o denotes the output dimension, s is the stride length, i is the input dimension, p is the padding and k is the kernel size.

$$o = s(i-1) - 2p + (k-1) + 1 \tag{2.1}$$

Another way to increase the spatial dimensionality of the feature map is by simply resizing it by interpolation. There are several ways to do this, such as Nearest-neighbor interpolation, bicubic interpolation and bilinear interpolation. Upscaling feature maps by interpolation can miss some information that the learned weights in deconvolution can pick up, but is also less computationally heavy as it does not introduce more learnable parameters in the model.

2.1.8 Hard negative mining

The balance between positive (text) and negative pixels (not text) in an image can be very skewed. In most scene images the negative pixels will severely outnumber the positive. This can lead to problems during training because the negative class will dominate and the model can learn to be biased towards predicting the dominating class. One way to deal with this is to simply design the loss function to focus more on the positive class (example of this in section 2.2.2). Another way to deal with this challenge is to implement *hard negative mining* (HNM). HNM is the process of resampling the classes to change the class balance by selecting the negative classes which are hardest to predict and discarding easy negatives. One way to implement HNM is to select a desired ratio between the positive and negative classes and calculate the loss for all pixels by propagating an image through the network. Initially the loss should be calculated for all pixels. If the number of negative pixels outnumber the number of negative pixels that are needed to achieve the desired ratio then only the negative pixels with the highest loss are kept. For example, if the desired ratio is 1:3 then the number of negative pixels that are kept is three times the number of positive pixels in the groundtruth. The selected pixels correspond to the nodes in the score map with highest score that are actually negative in the ground truth, and are by definition the negative pixels that the models had most trouble predicting correctly. Only the positive and the selected negative pixels are then used to calculate the gradients for the network to learn during training.

2.1.9 Non maximum suppression

In a network that outputs bounding box candidates, such as mentioned previously, it is very likely that many boxes which are highly correlated and, in essence, capturing the same class instance. Neighboring bounding box candidates are thus likely to encompass the same instances of the predicted class. Thus it is necessary to filter the candidates to obtain one bounding box per instance. Non maximum suppression is a widely used algorithm for this purpose. The idea is simple, select the candidate with the highest score and remove all boxes which overlap more than a certain threshold with the selected box. The procedure continues with the remaining boxes until no boxes are left. There exist several modifications to this idea in order to refine the filter process.

2.2 EAST

The EAST model for scene text detection was presented by Zhou et al. in 2017[31]. The goal of the authors was to construct a simpler and more efficient pipeline for text detection compared to other models at the time. The model is presented with two different bounding box geometries as output, rotated rectangles (RBOX) and quadrangles. Separate loss functions and output channels to each geometry are presented. In this project only the RBOX geometry was used.

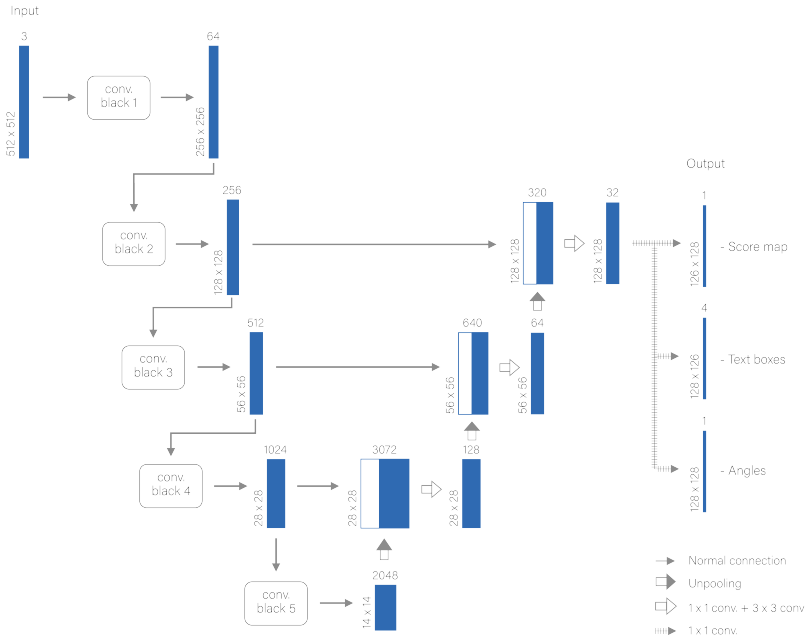

Figure 2.7: The complete architecture of EAST with ResNet50 as the feature extraction backbone. The blue rectangles correspond to the output at each stage in the network and the number above each output indicates the number of channels while the spatial dimensions are shown on the left.

2.2.1 Architecture

Due to the potential varying scale of text instances in images the authors adopted the idea of an U-shaped convolutional net presented in [19]. The network consists of a fully convolutional backbone network that performs feature extraction at different scales and a feature merging branch which merges the feature maps gradually up along the backbone network. This allows the network to identify text instances of different sizes. The architecture of the network can be seen in 2.7. The feature extraction stem can be seen on the left in the figure and the feature merging happens up along the feature extraction branch, producing an U-shape, as illustrated in the figure. There are some differences between the model presented in the original paper [31] and the implementation used in this project and they are described in detail in the following sections.

2.2.1.1 Backbone

The feature extraction network that is implemented in EAST as presented in the original paper [31] is PVANet. This is in line with the ambition stated by authors of creating and efficient model as PVANet is more lightweight and has fewer trainable parameters than ResNet50, while still comparable in terms of performance. The implementation of EAST that is used to carry out the work in this project uses ResNet50 as the feature extraction backbone. This change increases the computational costs of the model, but should not alter the performance much. This is acceptable as speed and efficiency is not the primary focus of this study.

2.2.1.2 Feature merging branch

The feature maps from convolution block 2-5 in the feature extraction network are extracted to be used in the feature merging branch. As mentioned previously, subsequent feature maps have dimensions (width and height) differing by a factor of 2 so in order to merge two feature maps the smaller one is upscaled to match the other. In the implementation of EAST used for this project the feature map are upscaled using bilinear interpolation. The upscaling operation used in the original model is not specified by the authors. After the lower/smaller layer is upscaled, the feature maps are concatenated. Then a 1x1 convolution layer is used to reduce the number of channels before a 3x3 convolution layer which preserves the size. This process is repeated starting with the feature map from convolution block 5 and finishing when all extracted feature maps are merged. The output from the feature merging branch is passed through one 3x3 convolution layer. The output from this layer is then used to produce the final output.

2.2.1.3 Output

The final output from the model consists of 6 channels. One channel represents a score map in the form of a pixel level prediction of where text is present in the text. Four channels indicating distances on a pixel-level to the four edges (top, bottom, left, right) if the given pixel is predicted to be inside of a bounding box and one channel indicating the angle of the bounding box. Each of the output channels are produced by applying a 1x1 convolution with one single kernel on the output from the feature merging branch.

2.2.2 Loss function

The loss function needs to address both the score map predictions as well as the geometry prediction. The loss function can be formulated as

$$L = L_s + \lambda_g L_g \tag{2.2}$$

where L_s represents the loss for the score map and L_g the loss for the geometry. λ_g is a hyper parameter that weights the two losses. In the original implementation of EAST the loss for the score map was calculated with class balanced cross entropy [31].

Class balanced cross entropy was first introduced in [27] and it is designed to tackle the class imbalance of negative and positive samples in edge detection tasks. Edge detection is similar to scene text detection in that both tasks are binary segmentation task where the negative class usually far outnumbers the positive. In other words, the number of pixels categorized as background in most cases far outnumber the number of pixels categorized as edge or text. Class imbalance can lead to difficulties in training because the algorithm can be biased towards the majority class if the imbalance is not addressed [7]. Class balanced cross entropy does this is by weighting the loss term corresponding to positive pixels in the ground truth by the

proportion of negative pixels in the ground truth and vice versa [27]. There are several loss functions designed to deal with this issue [11][27][17][28].

The implementation of EAST used in this project use the dice loss objective function instead of class balanced cross entropy. The dice loss is designed to maximize the Dice-Sørensen coefficient, also known as the F1-score, on a pixel level in the score map [17]. Since a high F1 score (close to 1) indicates a good match we subtract the F1 score from 1 to get a loss function that can be minimized during training. The loss function for the score map can be seen in eq. 2.3, where y_{gt} denotes the ground truth and y_{pred} denotes the predicted score map.

$$L_s = 1 - \frac{2\sum y_{gt} * y_{pred} + 1}{\sum y_{gt} + \sum y_{pred} + 1} \tag{2.3}$$

$$L_g = L_{AABB} + \lambda_\theta L_\theta \tag{2.4}$$

The loss function for the geometry, seen in eq. 2.4, consist of two terms, one for the bounding box overlap with ground truth, L_{AABB}, and one for the angle L_θ, weighted by the hyperparameter λ_θ. The loss function for the overlap is the IoU-loss, first presented in [29] and it can be seen in eq. 2.5. The intersection between the predicted and ground truth bounding box is calculated assuming that the bounding boxes are axis-aligned. The reason for this is that classical object detection only deals with non-angled bounding rectangles. EAST predicts rotated bounding boxes (RBOX) and thus the IoU loss is not completely accurate during training as the angle is not likely to be perfectly predicted. The authors still implement the classical IoU loss due to its simplicity and state that the approximation is close enough to produce gradients for the network to learn correctly. Equation 2.5 show the IoU loss function and the calculation for the intersection of the predicted and groundtruth bounding box can be seen in eq. 2.6.

$$L_{AABB} = -\ln \frac{intersection(pred, groundtruth)}{union(pred, groundtruth)} \tag{2.5}$$

$$\text{intersection}(p_i, g_i) = (min(p_l, g_l) + min(p_r, g_r)) * (min(p_u, g_u) + min(p_d, g_d)) \tag{2.6}$$

where p denotes the distance to all four edges of the predicted bounding box and g the distance to the edges of the groundtruth bounding box. The subscripts "l,r,u,d" represent left,right,up and down (lower) edges respectively.
The loss for the angle prediction output is calculated using the equation that can be seen in 2.7 where $\hat{\theta}$ is the predicted angle and θ^* is the groundtruth. The loss for the geometry of the bounding boxes, L_g, is multiplied with a mask of the positive and negative classes in the groundtruth, set to 1 and 0 respectively, so that the geometry regression only is performed on positive examples during training.

$$L_\theta(\hat{\theta}, \theta^*) = 1 - \cos(\hat{\theta} - \theta^*) \tag{2.7}$$

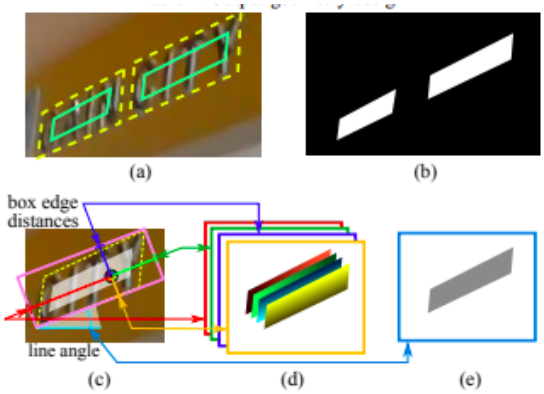

Figure 2.8: Label generating process for EAST:(a) Yellow dashed line show the bounding box and the solid, green line show the shrunk box. (b) Score map for the text; (c) The map showing the geometry for the rotated rectangle; (d) 4 channels of distances of each pixel to rectangle boundaries; (e) Rotation angle of rectangle.

2.2.3 Label generation for EAST

In order to calculate the loss it is necessary to generate the ground truth for the output layers. This includes the score map and the 5 geometry outputs. The score map is generated by shrinking the bounding boxes and setting all nodes inside the shrunk boxes to 1 and all other nodes to 0. The shrinking is done by moving each corner inwards along both outgoing edges. The corners are moved a factor of r of the length of the shortest outgoing edge. Since the bounding boxes are rotated rectangles, the shortest outgoing edge will have the same length for all corners. The authors of [31] have set $r = 0.3$ without providing any reason as to why. The shrunk bounding box and resulting score map can be seen in a) and b) in figure 2.8.

The geometry ground truths are generated by calculating the distance from each pixel inside the shrunk rectangle to the four edges. The edges, upper, lower, right, left, have corresponding output layers. The value of the nodes in the ground truth for these layers are set to the distance of the given node to the edge corresponding to the layer. This can be seen in c) and d) in figure 2.8. The angle map is produced by setting the nodes inside the shrunk rectangle to the angle of the lower edge relative to horizontal. Since the edges are denoted upper, lower, right and left the angle can only vary between -45°and 45°. The value for the angles is set in radians in the ground truth.

2.2.4 Post-processing with NMS

During inference, the output needs to be processed to produce the final bounding boxes. First the score map is filtered with a threshold of 0.8, this means that all geometries corresponding to pixels with a score below 0.8 in the score map are discarded. In dense examples this can amount to several thousand geometries that need to be filtered. The filtering of these geometries that are likely to be heavily overlap-

ping is done by a process that the authors call *Locality aware NMS* (LANMS). It is derived from the standard NMS algorithm, but is designed to be more efficient. The main idea behind the algorithm is the assumption that geometries at nearby pixels tend to be highly correlated [31]. Under this assumptions the geometries are merged iteratively in row first order if they overlap more than a given threshold. Standard NMS select geometries with the highest score and discards boxes that overlap the selected geometry. LANMS instead merges geometries that overlap by merging or averaging the geometries. The average is performed by weighting the geometries with their respective scores. If $a = \text{WeightedMerge}(g, p)$ then $a_i = V(g)g_i + V(p)p_i$ and $V(a) = V(g) + V(p)$ where a_i is one coordinate for the merged geometry and V(a) is the score for the merged geometry. To extract the actual geometry the coordinates of a is divided by the score $V(a)$. The authors claim that this averaging of geometries in addition to being more efficient also provides a stabilizing effect when feeding videos.

The final step of the post-processing is to perform standard NMS on the geometries resulting from the LANMS procedure. If the assumption that geometries at nearby pixels are highly correlated hold then LANMS will drastically reduce the number of geometries. Performing the more computationally expensive standard NMS algorithm on the reduced number of geometries will thus be less expensive than if one were to use it directly on the output from the thresholding.

2.3 Differential binarization: DBnet

The model that will be called *DBnet* in this project was presented in 2019 in [10]. The model is a segmentation-based network that directly outputs a single, pixel wise, score map. The novelty of DBnet is that the score map output is approximately binary due to a new module called *Differential binarization*. This module will be explained more in detail in 2.3.2. The model has shown state of the art results on benchmark datasets such as ICDAR2015 (f1-score of 0.873) and totaltext (f1-score of 0.847). The model is designed to be very efficient and the authors claim that the model also performs at state of the art speed with inference running at over 82 fps on a single GPU in some experiments [10].

2.3.1 Architecture

The architecture of DBnet is quite simple. It is based on a simple segmentation network and coupled with the differential binarization module. The feature extraction is performed by ResNet50 described in section 2.2.1.1 and the feature maps after each convolution block are extracted and merged together. Each convolution block scales down the feature maps by 1/2. The merging is performed by upscaling the feature map from convolution block 5, the last convolution block, in ResNet50 to match the scale of the preceding map, a 1x1 convolution to match the number of channels and performing an element wise sum on the two maps. The resulting map is merged with the preceding map in the same way up to the feature map that is produced from convolution block 2. In addition to the merging of all the

feature maps each merged feature map is passed through a convolution layer with 3x3 convolutions and upscaled to match the scale of feature map 2 before all maps are concatenated. This concatenated feature map is passed on to the differential binarization module which produces a score map as the final output of the network. The complete architecture of the model can be viewed in Figure 2.9.

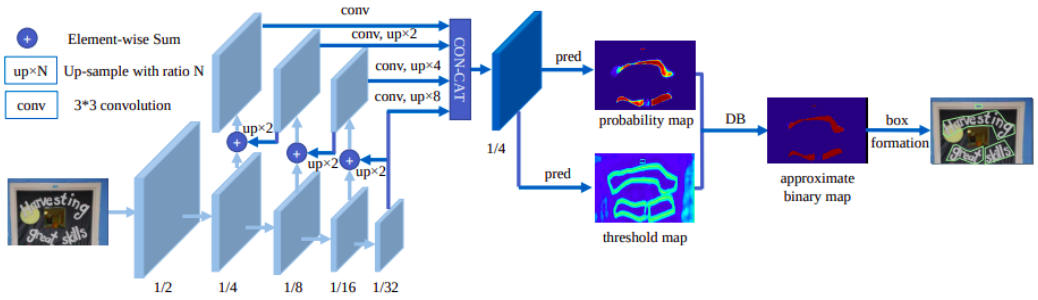

Figure 2.9: Illustration of the architecture of DBNet. "Pred" consists of one 3x3 convolution layer with stride 2 and two deconvolution layers with stride 2 so that the predictions match the dimensions of the input image. The fractions show the size of spatial dimensions of the layers in relation to the input dimensions. The figure is taken from [10].

2.3.2 Differential binarization module

In traditional segmentation-based methods the score map is processed with a binarization step before the bounding boxes/regions are found. The binarization step is simply to set pixels with values over a given threshold to 1, and 0 otherwise. This process is not differentiable and is thus performed on the score map that the network outputs. The threshold is a hyperparameter that is set manually. The differential binarization module is designed to approximate the binarization process and it is, as the name implies, differentiable so that it can be optimized along with the network during training. Instead of directly predicting a score map from the features the model predicts both a score map and a threshold map. The purpose of the threshold map is to predict the boundaries of the text instances. These two maps, the score map and the threshold map, are used with a specially designed activation function, seen in eq. 2.8 to produce a score map that is approximately binary. B is the approximate binary score map, P is the score map used in the DB-module and T is the threshold map. k is a constant that is used for amplifying the gradient.

$$\hat{B}_{i,j} = \frac{1}{1 + e^{-k(P_{i,j} - T_{i,j})}} \tag{2.8}$$

2.3.3 Loss function

The loss function to be minimized during training can be represented as a weighted sum of the loss for the score map, L_s, threshold map, L_t, and approximate binary map, L_b, seen in eq. 2.9. α and β are constants weighing the losses.

$$L = L_s + \alpha L_b + \beta L_t \tag{2.9}$$

Since the binary output map and the score map used in the DB-module in principle both are score maps predicting the same thing (text/no text) the loss function for the two is equal. The loss function used for the score maps is the binary cross-entropy loss with hard negative mining and it can be viewed in eq. 2.10.

$$L_s = L_b = -\sum_{i \in S_t} y_i \log(x_i) + (1 - y_i) \log(1 - x_i) \tag{2.10}$$

S_t is the sample set of positive and hard negative examples at a ratio of 1:3, y_i is the ground truth which is set to 1 for pixels in text instances and 0 else, and x_i is the predicted probability that pixel i is inside a text instance.

Using binary cross-entropy as the loss function is equivalent to maximizing the log-likelihood. Standard binary cross-entropy can sometimes run into issues when the classes are unbalanced and as mentioned previously this is often the case for scene text detection. In order to deal with this, hard negative mining is implemented.

$$L_t = \sum_{i \in R_d} |y_i^* - x_i^*| \tag{2.11}$$

Equation 2.11 show the loss function for the threshold map. R_d is the set of pixels inside the polygon which is used for creating the threshold map (explained further in the next section), y^* is the ground truth threshold map and x^* is the predicted threshold map. The loss function simply calculates the L1 distances between the ground truth and predictions inside the outer bound for the threshold for all text instances present in the sample.

2.3.4 Label generation

In order to calculate the loss it is necessary to generate the ground truth for the three maps (score, threshold and binary) using the known text instances in each sample. As discussed in the previous section, the score map and the approximately binary map essentially predicts the same thing, so the ground truth for these two maps are equal. All images are resized to the same size, this size can be selected before training is started and then has to be kept constant for training and inference. The score map has the same dimensions as the resized image. The bounding boxes are shrunk using the clipping algorithm presented by [24]. The offset is calculated using eq. 2.12 with the shrink ratio, r, set to 0.4. This value was found, empirically, to lead to the best performance by [10].

$$D = \frac{A(1 - r^2)}{L} \tag{2.12}$$

In eq. 2.12 A represents the area of the bounding box, L represents the perimeter and D represents the offset with which the bounding box is shrunk. All values have to be rounded to the nearest integer since we are dealing with a discrete space. All nodes corresponding to pixels inside the shrunk bounding boxes are set to 1 and

all other nodes are set to 0 for the ground truth labelling of the score maps. The threshold map is generated by dilating the bounding boxes with the same offset, D and setting all nodes in the area between the shrunk and dilated bounding box to 1 and the rest to 0.

The reason for shrinking the bounding boxes is to get better separation of text instances. Most datasets are annotated on a word level where the text instances can be very close together and in these cases the shrinking of the bounding boxes will help the network in separating text instances and improve the detection. During inference, the bounding boxes that are detected are dilated with the same ratio in order to get the correct scale and capture whole text instances.

2.4 Evaluation of text detection models

When evaluating any models it is important to select appropriate performance metrics for the task. The most common metrics used when evaluating scene text detection models are precision, recall and their harmonic mean, known as the f1-score [15][13]. These metrics are typically computed using the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). These values are found by matching predicted bounding boxes to ground truth.
Since the predicted area will never be a perfect replication of the ground truth it is necessary to use a method that defines criteria for matching a predicted bounding box with a ground truth bounding box. There exist several methods for matching predictions to ground truth that are very similar. Due to the similarity between scene text detection and the bigger field of classical object detection, much inspiration has been taken from the latter when it comes to methods for matching prediction to groundtruth. One such method which is widely used in object detection and scene text detection is called *Intersection over Union* (IoU).

2.4.1 Intersection over union

IoU is calculated by dividing the union of a prediction and ground truth box by the intersection of the two boxes. If the IoU is higher than a given threshold, typically 0.5, the prediction and ground truth are defined as matches. The classification of matches is binary, this means that a pair of prediction and ground truth boxes are either defined as a complete match or no match. This rather simple binary matching is suitable in most cases of classical object detection as semantically important information is not lost by detecting partial objects. In other words, a car is a car even if a portion of the car is not inside the predicted bounding box. The end goal in most cases where text detection is implemented is to recognize the text, and not just detect it. With this end goal in mind, the relaxed matching criteria of IoU might not be as suitable since missing portions of the text will inevitably lead to loss of semantically important information. Especially if the end-to-end pipeline feeds the

detected bounding boxes directly to a text recognition model without any steps in between. IoU will in this case give an inflated impression of the performance of text detection models compared to what can actually be achieved in an end-to-end text recognition pipeline.

2.4.2 Tightness Intersection over union

Tightness-aware IoU (TIoU) presented by [12] is designed to specifically deal with the limitations that IoU faces in the field of text detection. In particular it penalizes predictions which cut the text instance, it penalizes predictions which contain to much noise or non-text areas and it also does not match the predictions to ground truth in a binary way like IoU. Since the matching is not binary the recall and precision score is calculated in a different way than usual. The method calculates the recall and precision in a more targeted way than IoU and a pair of prediction and ground truth bounding boxes get assigned a separate recall and precision value.

2.4.2.1 TIoU-recall

The recall for a pair of prediction and ground truth bounding boxes in TIoU is targeted at representing the proportion of the ground truth that is captured by the prediction. The not-recalled area of a ground truth bonding box, G_i, by a prediction bounding box, D_j, is defined in eq. 2.13. $A(x)$ denotes the area of x.

$$C_t = A(G_i) - A(D_j \cap G_i), \quad C_t \in [0, A(G_i)] \tag{2.13}$$

The proportion of recall is defined as $f(C_t) = 1 - x$ where $x = C_t/A(G_i)$. The TIoU-recall score for the pair G_i and D_j is then calculated as seen in eq. 2.14.

$$TIoU_{recall} = \frac{A(G_i \cap D_j) * f(C_t)}{A(G_i \cup D_j)} \tag{2.14}$$

The TIoU-recall score is very similar to the classical IoU score, the only difference being the penalization factor $f(C_t)$ which is the proportion of the area of G_i that was captured by the prediction.

The recall metric is calculated directly using the TIou-recall score instead of classifying matches like normally. For ground truth instances with IoU score under 0.5 the $TIoU_{recall}$ is set to 0 and the rest are used directly to calculated the recall. The formula for calculating the recall metric is shown in eq. 2.15.

$$Recall_{TIoU} = \frac{\sum TIoU_{recall}}{Num_{gt}} \tag{2.15}$$

2.4.2.2 TIoU-precision

The TIoU-precision is targeted at penalizing detections which cover more than one text instance. If a detection of text overlaps with several text instances it can be difficult for a text recognition model to know what text is the target and it will often lead to inaccurate recognition of the text. Only the area of ground truth instances that are inside the detection but outside the target ground truth will be penalized. This area for a pair of predicted and ground truth instance, D_j and G_i respectively, is defined in eq. 2.16.

$$\begin{aligned} O_{i,j} =& A((G_1 \cap D_j - G_1 \cap D_j \cap G_i) \cup ...(G_{i-1} \cap D_j - G_{i-1} \cap D_j \cap G_i) \cup \\ & (G_{i+1} \cap D_j - G_{i+1} \cap D_j \cap G_i) \cup ...(G_n \cap D_j - G_n \cap D_j \cap G_i)) \end{aligned} \tag{2.16}$$

The proportion of intersection between the prediction and unwanted ground truths is then calculated as $h(O_{i,j}) = 1 - x$ where $x = O_{i,j}/A(D_j)$. The TIoU-precision score for the pair G_i and D_j is then calculated using the formula in eq. 2.17.

$$TIoU_{precision} = \frac{A(G_i \cap D_j) * h(O_{i,j})}{A(G_i \cup D_j)} \tag{2.17}$$

Once again it is clear that the precision score is the same as the standard IoU, except for the penalizing factor h. Similarly to the recall metric, the precision metric is calculated by setting the $TIoU_{precision}$ of detections with IoU lower than 0.5 to 0 and otherwise using the $TIoU_{precision}$ score directly to calculate the precision using the formula in eq. 2.18.

$$Precision_{TIoU} = \frac{\sum TIoU_{recall}}{Num_{det}} \tag{2.18}$$

2.4.2.3 TIoU f1-score

The f1-score which is the metric that is mainly reported when evaluating text detection models is calculated as the harmonic mean of the precision and recall. Equation 2.19 shows how the f1-score is defined for TIoU.

$$F1_{TIoU} = \frac{2 \cdot Recall_{TIoU} \cdot Precision_{TIoU}}{Recall_{TIoU} + Precision_{TIoU}} \tag{2.19}$$

2.4.3 Many-to-one and one-to-many

The evaluation protocols that have been presented in the previous sections only deal with so-called one-to-one (OO) matching. OO matching means that only pairs of predicted and ground truth bounding boxes are matched and it does not take into account that one ground truth instance can be predicted by multiple detections (one-to-many, OM) or opposite, that one detection spans multiple ground truth instances (many-to-one, MO). These situations where the detection is over or under segmented compared to the ground truth may very well be correct in terms of detecting the actual text, but will not (in most cases) be classified as correct with the matching protocols discussed in the previous sections. This issue can arise when

performing text detection because text instances often appear close to each other in images and can be difficult to separate correctly. Liou et al. [12] also notes that there are inconsistencies in the annotation of several benchmark datasets including the ICDAR 2015 dataset which is used in this project. The inconsistencies in annotation will also lead to inconsistencies when evaluating with OO methods.

Solutions to the problem of under- and over-segmentation in the results are evaluation protocols which include OM and MO matching. Liou et al. [12] present an attempt to deal with this by first evaluating the results on a line-level ground truth and removing the word level annotations for resulting matches before evaluating the remaining ground truth instances on a word level. This is specifically aimed at handling the flaws in the evaluation process that are introduced by inconsistent annotations. The significant downside of this solution is that line-level annotations are needed, and this has to be generated in many cases as most benchmark datasets in the field come with word-level annotations.

3
Data

This section will present the data that is used in this study. First the annotation format is explained before the datasets are presented. The data consists of 4 openly available, and widely used, benchmark datasets for scene text detection and a set of images collected from social media by Recorded Future.

3.1 Annotation format

There does not exist one standardized format for annotating text in images. Text can be defined on different levels such as character-level, word-level and line-level. When training and evaluating scene text detection models it is important that the text is consistently annotated. In this project the text is annotated on a word-level, meaning that every word counts as one instance of text. In other words, there is a one-to-one relation between words and bounding boxes. This is the most prevalent format for defining text in the field of scene text recognition.

The text instances are annotated with RBOX geometry (rotated rectangles). This means that every annotation consists of four vertices which form a rectangle. The coordinates of the vertices are defined in pixels with the origin located at the upper left corner of the image and the x-axis going horizontally to the right and the y-axis pointing downwards.

3.2 Benchmark datasets

In this section the four benchmark datasets that are used in this project are presented. Benchmark datasets are used for measuring performance of scene text detectors and models presented in the literature are always accompanied by results achieved on one or more of the openly available datasets. For keeping track of the training process the training sets are split into a set used for training (80%) and a validation set (20%).

3.2.1 ICDAR2013

ICDAR2013 was released for the second Robust Reading Competition in 2013[1]. The dataset consists of images with text instances that are mainly horizontal and in focus. Text instances in ICDAR2013 are annotated with horizontal rectangles, but the annotations are acceptable for RBOX geometry since the text instances are mainly horizontal and the angle can be set to 0. The dataset contains 229 images in the training set and 233 images in the test set. Since the number of images is so low, the training set is concatenated with the training set of ICDAR2015 during training.

3.2.2 ICDAR2015

ICDAR2015 was released for the fourth Robust Reading competition in 2015[2]. The competition is titled "Incidental scene text" and the images are captured with Google glasses and show every day scenes with incidental text instances. This means that the text is not in focus and can be tilted at an angle. The images were not taken in a way to improve the quality or positioning of the text instances. ICDAR2015 is annotated with RBOX geometry and consists of a training set with 1000 images and a test set with 500 images.

3.2.3 TotalText

TotalText was presented by [3] in 2017. The images have a higher frequency of curved and multi-oriented text instances than other benchmark datasets such as ICDAR and TotalText is thus considered a benchmark dataset for the ability to detect curved scene text. TotalText consists of a training set with 1255 images and a test set with 300 images. The dataset comes with two different annotation formats; polygon and non-angled rectangles. In order to use consistent annotation formats the polygon annotations are used to produce rotated rectangles by using the `minAreaRect()` function from openCV[3] which fits the rectangle with smallest possible area that covers the polygon.

3.2.4 COCO-text

COCO-text is one of the largest scene text detection datasets that are openly available. It was released in 2016 and presented by [25]. The images depict complex everyday scenes and the text is not in focus. COCO-text consists of a training set of 43686 images, a validation set of 10000 images and a test set of 10000 which is not publicly available. Since the test set is not available the validation set is used as a test set in this study. The training set is split into training and validation sets as described previously. The annotations of COCO-text are converted to RBOX format in the same way as TotalText.

[1] https://rrc.cvc.uab.es/?ch=2
[2] https://rrc.cvc.uab.es/?ch=4
[3] https://pypi.org/project/opencv-python/

3.3 Recorded Future image set

Recorded Future has provided a set of images from their data collection pipeline. The set consists of 1499 unique images which are collected from social media. There is very high variance in the kind of images such as screenshots, images of documents and natural scene images. In order to get some insight into the dataset it was decided to label the images into three classes; natural images, documents and mixed. Natural images are images that have variability in color, contrast, etc. If there is text in the images it is either scene text or text overlaid onto an image. Digitally created images which fit into these criteria are also considered natural images. Natural images which contain no text were additionally classified as "No text". Document images are screenshots containing mainly text, images of documents and similar. The main criteria is that the text is in focus and on a plain background. Finally the mixed class is defined as images with a combination of the two previous classes. One example of a mixed image is a screenshot of a twitter post consisting of a caption and a natural image. The distribution of the defined classes can be seen in table 3.1. The class of natural images is of most interest in this study as it is the domain of scene text detection. Detecting text on documents and focused text on a plain background is a considerably easier task.

Table 3.1: The distribution of images in the three classes "Natural images", "Document" and "Mixed" in the dataset provided by Recorded Future.

Class	# of images
Natural images	944(408 no text)
Document	358
Mixed	198

In addition to the images, Recorded Future has also provided annotations from Google vision's text detection service. Google visions text detection is absolute state-of-the-art and it produces great detections in most images. There are, however, some cases where it fails and therefore it can not be viewed as ground truth for the complete set. Instead the images classified as natural images are inspected visually and a subset of the images, where the annotations from google vision are good enough to be used as groundtruth, will be used for evaluating the model instances on. This subset will be referred to as *RecFut* and it consists of 230 images.

4
Methodology

In this chapter, the work that has been done in this study is described. The first section will describe the training process for both models and the subsequent chapters will describe how the models are evaluated to analyze the models and answer the problem statement.

4.1 Training process

When training a neural network model like EAST and DBNet there are many hyperparameters that need to be decided to define the training process. To achieve optimal performance it is common to perform a hyperparameter search to optimize these parameters, in this study however, the goal is not optimal performance but rather to analyze the generalization abilities and thus time will not be invested into this. In addition it is reasonable to believe that the people behind the implementation of the algorithms have performed some sort of hyperparameter optimization and thus the default settings are expected to be satisfactory. EAST and DBNet have different training processes, but the training process is equal for the different instances of the respective models. The training was conducted on one of Recorded Future's GPUs, a Nvidia GeForce RTX 2080.

4.1.1 Training parameters for EAST

EAST is implemented with a mini-batch gradient descent, meaning that the network is fed a batch of samples used for calculating the loss and updating the trainable parameters. The gradients that are calculated from the samples in the batches are added before being applied to the trainable parameters. The batch size is set to 10 samples and the input size of the samples is set to 512×512. In order to achieve this input size the samples consist of random portions of the images in the training set with the correct input size. The process of extracting a portion of the images is performed making sure that text instances in the images are not cut. Images which are smaller than 512×512 pixels are padded with zeros in order to get the correct input size.

The batches are loaded randomly from the training data and thus it is not possible to use epochs as a measure of training duration. One epoch is defined as one full rotation through the training set, meaning that each sample has been passed

through the model exactly once. When training EAST in this project, one rotation of the training data is defined as the number of batches needed to feed the network the same number of samples as one epoch (number of images in the training set). After each rotation of the training set, the loss is calculated for the validation set without calculating and applying the gradients to the network. Early-stopping is implemented such that if the validation loss does not improve during 50 rotations of the data, the training is suspended and the model with the best validation loss is kept. Due to limited resources and time in this project, the maximum number of batches is set to 100000, upon which training is stopped.

The network is optimized using the ADAM optimizer[9]. The learning rate is initially set to 1e-4 with an exponential decay rate of 0.94 every 5000 batches. The decay rate for the first and second moment is set to 0.9 and 0.999 respectively. The constants λ_g in eq. 2.2 is set to 100 and λ_θ in eq. 2.4 is set to 20.

4.1.2 Training parameters for DBNet

DBNet is also implemented with mini-batch gradient descent with a batch size of 10. However, instead of randomly selecting samples, the samples are divided into batches such that each sample is fed to the network exactly once before a new rotation of the data is started. The input size is set to 640 $\times$ 640 pixels and the images are processed in the same way as in EAST to achieve the input size.

Instead of evaluating the loss of the validation set after each epoch, the current model is used for inference on the validation set and the predictions are evaluated with IoU. Early-stopping is implemeted such that if the f1-score for IoU on the validation set does not improve over 50 epochs training is terminated and the model with the highest IoU is kept. Similarly to EAST, the maximum number of batches is set to 100000. The network is optimized using Stochastic Gradient Descent with an initial learning rate of 1e-3 and a weight decay of 5e-4 every 200 epochs, and momentum factor of 0.99.

The constant k in eq. 2.8 is set to 50 and the weighting constants α and β in eq. 2.9 are set to 5 and 10, respectively.

4.2 Analyzing generalization abilities

The main aim of this study is to analyze the generalization abilities of EAST and DBNet. Due to the limited amount of openly available data and the costs connected with producing annotated data it is of high interest to study how model instances trained on benchmark datasets are able to generalize to other domains. This will be studied by training instances of EAST and DBNet on different datasets and evaluating each instance on all the test sets from the benchmark datasets and RecFut.

One idea for potentially improving the generalization abilities using only openly available datasets which was presented by [13] is to pool existing datasets. Pooling datasets will increase the variation and amount of training data and the hypothesis is that it will improve the generalization abilities of the models. Therefore model instances will also be trained on pooled variations of the benchmark datasets which are used in this project. In total different instances of EAST will be trained on ICDAR (ICDAR2013/ICDAR2015 pooled together), TotalText, COCO, a pool of ICDAR and TotalText and a pool of ICDAR, TotalText and COCO. Due to formatting issues with the COCO dateset DBnet will only be trained on ICDAR, TotalText and the two datasets pooled together.

All of the trained instances are evaluated over all test sets i.e. ICDAR2013, ICDAR2015, TotalText, COCO and RecFut. The performance of the models is evaluated using both IoU and TIoU, and the metrics recall, precision and f1-score are calculated for both evaluation protocols. Evaluating all instances on all test sets will give an insight into how well performance is generalized to domains and data which the model instance has not encountered during training.

4.2.1 Macro-average of f1-scores

The current standard of reporting results on single benchmark datasets performed by a model instance which is specifically trained on the target domain show that the models are able to perform well in given domains, but they do not reflect how well one single model instance can handle text detection in a general setting with examples from different domains. In order to better report the generalization abilities of scene text detection models the f1-scores over the benchmark datasets are averaged to produce a single value. Instances of scene text detection models are evaluated on a selection of benchmark datasets and the macro-average of the f1-scores is calculated. The datasets that are used for calculating the macro-average score in this report are ICDAR2013, ICDAR2015, TotalText and COCO-text.

There are several benefits of calculating a macro-average of the f1-scores achieved by each model instance. The main benefit is that it illustrates the generalization abilities of scene text detection models better than the standard way of reporting results. Also, since it only uses openly available datasets it can be standardized to create a new benchmark, similarly to how the benchmark datasets are currently used separately to report results. If the field of scene text detection could agree on a selection of benchmark datasets to be used for the calculation of such a macro-average then it could become a valuable, standardized metric that can be used to compare the generalization abilities of different models.

4.3 Text/Non-text classifier

Another aim of this project is to analyze whether the scene text detection models can be used for classifying whether an image contains text or not. This is a simple binary classification task. In order to examine this question the model instances described in the previous section will be used to predict text in the images provided by Recorded Future. If a model predicts any text instances in an image then it will be classified as containing text (positive class) and if not then it is classified as not containing any text (negative class). Ground truth for this experiment will be defined using the classes described in Section 3.3. The images in the "document" and "mixed" class contain text by default and thus a detection of text in any of these images will count as a true positive result. In the class "natural image" there are both positive and negative samples, but the images that are in this class are also classified as whether they contain text or not. The class of "text" (positive) and "no text" (negative) will be used as ground truth.

The use case for an image classifier in the intelligence domain, where Recorded Future operates, would potentially be to decide whether the image should be sent to more detailed analysis in an end-to-end pipeline. It is crucial to avoid losing potential information that can be turned into valuable intelligence so the classifier must have a high recall. On the other hand the classifier must not be too "trigger happy" because there is a cost connected to analyzing images, and the main motivation for implementing the classifier in the first place is to save the cost of performing analysis on images which contain no information. The classifiers will therefore be evaluated with the metrics precision and recall, with an extra emphasis on the recall.

5
Results

In this chapter the results from the experiments that have been carried out are presented. In the first section the results from the experiment aiming to analyze the generalization abilities of EAST and DBnet, described in 4.2, are presented and discussed. The second section presents the results from the experiment where the models are used to classify images depending on whether there is text in the image or not. This experiment was described in 4.3.

5.1 Generalizing abilities

In the study of the generalization abilities of EAST and DBnet several instances of the models were trained on different data and evaluated over all the test sets in the study. Five instances of EAST and three instances of DBnet was evaluated on five different test sets. The results from this study can be seen in table 5.1, 5.2, 5.3 and 5.4.

5.1.1 EAST

Table 5.1 and 5.2 show the results for the instances of EAST with the IoU and TIoU evaluation protocol, respectively. The results show that the EAST model generally has higher precision than recall. Another result that stands out at first glance is that EAST, overall, perform better on the two ICDAR datasets better than the rest, having particular difficulty with COCO-text.

Looking at Table 5.1, one can see that the model trained on the pooled ICDAR datasets and the model trained on TotalText perform best on the ICDAR2015 and TotalText test set, respectively. The model instance which performed best on ICDAR2013 was trained on IDCAR and TotalText pooled together. This is reasonable as ICDAR2013 and TotalText are the two datasets which contained focused text, meaning that the images were captured with the text in mind. Thus the training set where ICDAR and TotalText are pooled together have the highest proportion of focused text where ICDAR2013 is included, since ICDAR2015 contain more than four times as many images as ICDAR2013. The instance trained on TotalText also has the second best performance on ICDAR2013 ahead of the instance which was only trained on ICDAR data. These finding show that EAST learns specific features for the different domains that is is trained on that might not be universal for text

in a general setting, and that one can not expect to get the same performance on unseen domain as it is possible to achieve on some of the commonly used benchmark datasets.

Table 5.1: Recall, precision and F1-score achieved by the different instances of the EAST model on the different datasets using the IOU protocol. I13=ICDAR2013, I15=ICDAR2015, T=TotalText, C=COCO-text

Test / Train	ICDAR 2013			ICDAR 2015			TotalText		
	R	P	F	R	P	F	R	P	F
I13/I15	0.69	**0.80**	0.740	**0.73**	**0.82**	**0.772**	0.47	0.60	0.524
T	0.72	0.76	0.741	0.63	0.63	0.627	**0.59**	**0.67**	**0.627**
C	0.54	0.58	0.561	0.57	0.65	0.608	0.43	0.45	0.436
I13/I15/T	**0.73**	0.80	**0.763**	0.65	0.75	0.696	0.58	0.64	0.609
I13/I15/T/C	0.69	0.70	0.698	0.69	0.76	0.724	0.57	0.57	0.568

	COCO-text			RecFut		
	R	P	F	R	P	F
I13/I15	0.31	0.53	0.387	0.56	**0.74**	**0.635**
T	0.24	0.37	0.290	0.50	0.65	0.560
C	0.31	0.46	0.365	0.45	0.57	0.504
I13/I15/T	0.26	0.38	0.305	0.50	0.65	0.562
I13/I15/T/C	**0.34**	**0.55**	**0.419**	**0.57**	0.69	0.625

The performance on the RecFut dataset which is not seen during training by any instances show that simply pooling together benchmark datasets did not clearly lead to performance gain, as the baseline instance trained only on the ICDAR datasets actually performed best overall. However, the instance which was trained on all the datasets pooled together performed better on COCO and TotalText while performing close to the baseline instance on the ICDAR and RecFut datasets. Especially when looking at recall, which is important in order to capture as much text as possible within reason. This is a small indication that pooling datasets together can lead to better generalization abilities, as expected.

Comparing the results in table 5.1 and 5.2 is is clear that the TIoU evaluation protocol is much stricter than IoU with all results about 10-25 percentage points lower. However it is also clear that the two protocols are highly correlated as the findings from table 5.1 also appear in table 5.2. The lower results are due to the penalizing factor and non-binary matching of TIoU. According to [12] these results are a better depiction of the performance one can get in an end-to-end pipeline if the output from the text detector are used directly for text recognition without any post-processing steps in between.

Table 5.2: Recall, precision and F1-score achieved by the different instances of the EAST model on the different datasets using the TIOU protocol. I13=ICDAR2013, I15 = ICDAR2015, T=totaltext, C=COCO-text,RF=recFut

Test / Train	ICDAR 2013			ICDAR 2015			TotalText		
	R	P	F	R	P	F	R	P	F
I13/I15	0.50	**0.64**	**0.557**	**0.47**	**0.59**	**0.522**	0.26	0.41	0.318
T	0.49	0.55	0.519	0.41	0.46	0.426	**0.38**	**0.48**	**0.423**
C	0.31	0.38	0.339	0.32	0.43	0.366	0.21	0.28	0.239
I13/I15/T	**0.51**	0.61	0.554	0.42	0.54	0.471	0.36	0.46	0.404
I13/I15/T/C	0.45	0.51	0.478	0.44	0.54	0.485	0.33	0.39	0.356

	COCO-text			RecFut		
	R	P	F	R	P	F
I13/I15	0.19	0.36	0.247	**0.36**	**0.52**	**0.422**
T	0.15	0.25	0.182	0.31	0.45	0.368
C	0.18	0.30	0.223	0.26	0.36	0.299
I13/I15/T	0.16	0.25	0.194	0.32	0.45	0.370
I13/I15/T/C	**0.21**	**0.38**	**0.270**	0.35	0.47	0.402

5.1.2 DBnet

Table 5.3 and 5.4 show the results for the instances of DBnet with the IoU and TIoU protocol, respectively. The results show that DBnet generally has higher recall than precision, meaning that DBnet is quite "trigger-happy" during inference. Similarly as for EAST, it is clear that DBnet performs better on the two ICDAR datasets than the rest, with the best instance achieving f1-scores of 0.750 and 0.761, while the best f1-score for TotalText, COCO and RecFut was 0.662, 0.453 and 0.648, respectively. The recall metric, however, does not vary as much, the best recall scores on the test sets all lie within 0.57-0.81. In fact, no instance achieved a recall below 0.5 on any of the datasets. Even on the COCO-text dataset which was not seen by any instance during training. This indicates that the ability to capture the text generalizes quite well, but also comes with many false positives.

There is no clear result showing that pooling data sets lead to improved generalization ablities as the best perfomance in f1-score on both COCO-text and RecFut was achieved by the baseline instance trained only on the ICDAR datasets. However, the instanced trained on both TotalText and ICDAR has better recall for both COCO-text and RecFut. In addition the instance trained on TotalText and ICDAR performs better on ICDAR2013 and much better on TotalText. This indicates that the model trained on pooled datasets has better generalization abilities and especially when it comes to curved text instances which there are many of in TotalText. This result is not very surprising given that there are practically no curved text instances in the ICDAR datasets.

Table 5.3: Recall, precision and F1-score achieved by the different instances of the DBNet model on the different datasets using the IOU protocol. I13=ICDAR2013, I15 = ICDAR2015, T=totaltext

Test / Train	ICDAR 2013			ICDAR 2015			TotalText		
	R	P	F	R	P	F	R	P	F
I13/I15	0.76	0.67	0.712	**0.81**	**0.72**	**0.761**	0.53	0.42	0.471
T	**0.79**	0.61	0.692	0.75	0.51	0.609	**0.77**	0.58	0.658
I13/I15/T	0.79	**0.71**	**0.750**	0.78	0.68	0.725	0.73	**0.61**	**0.662**
	COCO-text			RecFut					
	R	P	F	R	P	F			
I13/I15	0.54	**0.39**	**0.453**	0.66	**0.63**	**0.648**			
T	0.57	0.26	0.356	0.67	0.55	0.600			
I13/I15/T	**0.57**	0.27	0.365	**0.70**	0.59	0.640			

The results in table 5.4 show, once again, that the TIoU metric is much stricter than IoU. The performance according to the TIoU protocol is between 5 and 30 percentage points lower than for IoU. The results are, however, highly correlated and the same tendencies and indications of generalization abilities are seen in both table 5.3 and 5.4.

Table 5.4: Recall, precision and F1-score achieved by the different instances of the DBNet model on the different datasets using the TIoU protocol. I13=ICDAR2013, I15 = ICDAR2015, T=totaltext„RF=recFut

Test / Train	ICDAR 2013			ICDAR 2015			TotalText		
	R	P	F	R	P	F	R	P	F
I13/I15	0.49	0.51	0.501	0.51	**0.54**	**0.524**	0.27	0.29	0.282
T	**0.56**	0.46	0.506	**0.52**	0.38	0.437	**0.49**	0.43	**0.457**
I13/I15/T	0.53	**0.54**	**0.537**	0.51	0.50	0.509	0.43	**0.44**	0.436
	COCO-text			RecFut					
	R	P	F	R	P	F			
I13/I15	0.34	**0.29**	**0.315**	0.41	**0.45**	0.430			
T	0.38	0.19	0.249	0.43	0.38	0.403			
I13/I15/T	**0.39**	0.20	0.261	**0.44**	0.42	**0.432**			

5.1.3 Macro-average of f1-scores

The results for the macro-averages, calculated as described in Section 4.2.1, can be seen in Table 5.5. The best performing instance of EAST was the one which was trained using only the ICDAR datasets, closely followed by the instance that was trained on all benchmark datasets pooled together. The best performing instance of DBnet was the one which was trained on both ICDAR and TotalText, and this instance of DBnet performed the best of all instances of both EAST and DBnet. The two different models achieve similar results.

Table 5.5: The macro-average of the f1-scores for all model instances.

Model Instance	Macro-f1 (IoU)	Macro-f1 (TIoU)
EAST - I	0.612	0.419
EAST - T	0.569	0.384
EAST - C	0.495	0.293
EAST - I/T	0.587	0.399
EAST - I/T/C	0.607	0.399
DBnet - I	0.609	0.410
DBnet - T	0.583	0.410
DBnet - I/T	0.628	0.435

5.1.4 Comparison between EAST and DBnet

The results of EAST and DBnet are quite similar in terms of f1-score which is the main metric which is reported in the field of scene text detection. The best f1-score of both models are within 3.5 percentage points for all datasets for both IoU and TIoU. There is, however, a big difference behind the calculation of the f1-score for the two models. While EAST generally has higher precision than recall, the opposite is true for DBnet. One example from the results on RecFut with the IoU evaluation protocol reveals the difference: the highest precision achieved on RecFut by any instance of EAST was 0.74 while the corresponding best for DBnet was 0.63. For recall on the other hand, the highest score which was achieved by EAST was 0.57 while the corresponding value for DBnet was 0.70. This result is representative for the results in this experiment and it shows the importance of not only focusing on a single metric.

The tradeoff between precision and recall can generally be adjusted by tuning the threshold in the matching protocol and is therefore often not very meaningful to compare in detail. However, there are some clear differences between the two models' results which contribute to the different characteristics that are observed. This is discussed more in detail in Section 6.1

Both EAST and DBnet display some generalization abilities but they have completely different characteristics. The EAST instance which was trained on all the datasets used in this study pooled together is able to achieve a precision higher than 0.55 on all test sets while the recall varies from 0.34 to 0.76. The DBnet instance which was trained on ICDAR and TotalText, on the other hand, is able to achieve a recall above 0.57 for all datasets while the precision ranges from 0.27-0.71.

5.2 Text/Non-text classifier

In order to examine how the models perform, when used as a classifier to classify whether an image contains text or not, all instances were evaluated on the complete

dataset provided by Recorded Future as described in 4.3. The results from this experiment can be seen in table 5.6. Both models are able to effectively identify images which contain text, with DBnet having slightly higher recall than EAST. The higher recall comes at a cost of lower precision, and this combination is not surprising giving the finding in the previous experiment, which showed that DBnet is quite "trigger-happy" compared to EAST.

Table 5.6: The recall and precision results of the model instances when used as an image classifier with the classes text/no text on the complete dataset provided by Recorded Future.

Model instance	Recall	Precision
EAST - I13/I15	0.950	**0.883**
EAST - T	0.960	0.812
EAST - C	0.947	0.872
EAST - I13/I15/T	0.961	0.798
EAST I13/I15/T/C	0.951	**0.883**
DBnet - I13/I15	0.978	0.635
DBnet - T	**0.994**	0.688
DBnet - I13/I15/T	0.993	0.635

6
Discussion

In this chapter the results are discussed and potential directions for future work are presented.

6.1 Generalization abilities in natural images

The results show that both DBnet and EAST has some generalization abilities, although with quite different characteristics. While DBnet was able to generalize the recall to unseen domains reasonably well, EAST on the other hand was able to achieve higher precision for all domains. These characteristics are useful in different use cases, for instance in scene-text-to-speech or instant translation applications it is desirable to have high precision to avoid garbage output, such as texture being interpreted as text resulting in meaningless output. In the intelligence domain, however, it is desirable to extract as much information as possible and garbage output does not really affect the result that much. The reason that garbage output does not affect the results a lot is because meaningless words will generally not result in hits from other references or keywords and will just be ignored. In this case the results for DBnet are more promising due to the higher recall. The performance of both models on the RecFut datset are far inferior to the result achieved by Google Vision, which is used as the ground truth and thus, by definition, achieves 100% recall and precision. It is important to note that this result is heavily biased as the set was specifically selected using with the criteria of being able to use it as ground truth.

The evaluation protocol plays a big role in how the performance is evaluated. The results show that the TIoU protocol is significantly stricter than the IoU protocol. A visual inspection of the images with the predicted bounding boxes drawn on, gives an impression of better generalization and detection abilities than the evaluation metrics show. There are some reasons for why this mismatch might be true. First of all, MO- and OM-matches are not considered in the evaluation protocol used in this project. The performance of EAST is particularly negatively affected by this because is struggles more to segment the predicted text at a word-level. Figure 6.1 shows two images from RecFut with ground truth and prediction bounding boxes as well as the evaluation metrics of each image. Both images clearly show that EAST identifies the text quite well, but the segmentation of the predictions is not on a word-level and thus the performance is highly punished. Even though a majority

(a) IoU: P: 0.5, R: 0.286, F: 0.364
TIoU: P: 0.419, R: 0.222, F: 0.310

(b) IoU: P: 0.0. R: 0.0, F: 0.0
TIoU: P: 0.0, R: 0.0, F: 0.0

Figure 6.1: Two images from the set provided by Recorded Future which was collected on social media. Ground truth bounding boxes are drawn in green and predictions from the EAST instance which was trained on all the data are drawn in red.

of the words in both images are completely captured by bounding boxes the recall is only 0.286 and 0.0 with the IoU protocol. This shows the importance of using evaluation protocols which handle OM- and MO-matching when evaluating scene text detection models.

The reason for EAST's issues with segmenting the text on a word level could be caused by the post processing with LANMS. If the NMS is too agressive with merging bounding boxes and merges boxes which are in fact targeting two different text instances then this would cause the poor text segmentation that is observed. On the other hand, it might be possible to engineer some heuristics that can improve the text segmentation and refine the bounding boxes before they are fed to a text recognition model. One idea for this could be to crop a section of an image containing several bounding boxes and feeding the cropped portion of the image the the text detection model separately.

Visual inspection of the results from DBnet show that the issue of under segmenting the text instances is not a problem for DBnet. Figure 6.2 show the same images as Figure 6.1, but with the results from DBnet. It is clear that the text segmentation is much better fitted to the word level annotations that are used in this study and this is generally true for the predictions of all images in RecFut. One of the reasons that DBnet is able to segment the text so effectively is likely to be the Differential Binarization module. According to [10], the threshold map used in the module is

(a) IoU: P: 1.0, R: 1.0, F: 1.0
TIoU: P: 0.740, R: 0.617, F: 0.673

(b) IoU: P: 1.0 R: 1.0, F: 1.0
TIoU: P: 0.626, R: 0.547, F: 0.584

Figure 6.2: Two images from the set provided by Recorded Future which was collected on social media. Ground truth bounding boxes are drawn in green and predictions from DBnet trained on ICDAR and TotalText are drawn in red.

included specifically to help the model learn to predict precise boundaries for the text instances.

The results showed that DBnet is quite trigger-happy and produced many false positives. Examples of this are displayed in figure 6.3. Figure 6.3a shows that DBnet segments the actual text quite well, although not perfect, but also predicts some text in places where it looks nothing like text to the human eye. In figure 6.3b there is also a false positive, but in this case the soldiers wearing dark colors and standing in line against the lighter background somewhat resembles textual features. EAST on the other hand which achieved higher precision has much fewer false positives and most detections that are not classified as correct are due to wrong segmentation.

The examples in figure 6.3c and 6.3d also illustrate how EAST's issue of wrongful segmentation leads to performance loss with the evaluation protocol used in this study when text instances are close, such as in a sentence, but does not affect the performance of single words. In figure 6.3c the recall is only evaluated to 0.2 with the IoU protocol, meaning that only 1 out of 5 words is considered detected correctly, even though 4 words are completely covered by the union of the bounding boxes. In figure 6.3d where the words are more isolated each word is considered to be detected by the IoU metric as the segmentation is not an issue. After inspection of the results of the evaluation metrics and a visual inspection of the output from EAST it becomes apparent that EAST is better at detecting text than the metrics indicate, but it is not precise with the word level segmentation and thus is penalized

quite heavily by the evaluation metrics.

(a) IoU: P:0.3, R: 0.6, F: 0.4
TIoU: P: 0.218, R: 0.417, F: 0.286

(b) IoU: P:0.75 R: 1.0, F: 0.857
TIoU: P: 0.549, R: 0.583, F: 0.566

(c) IoU: P: 0.333 R: 0.2, F: 0.25
TIoU: P: 0.244, R: 0.143, F: 0.180

(d) IoU: P:1.0 R:1.0, F: 1.0
TIoU: P: 0.755, R: 0.705, F:0.729

Figure 6.3: Two images from the set provided by Recorded Future which was collected on social media. Ground truth bounding boxes are drawn in green and predictions are drawn in red. Image a) and b) have predictions from DBnet while c) and d) have predictions from EAST.

In an attempt to quantify the generalization abilities of the model instances, the average f1-score was calculated on the benchmark datasets. The results showed that the two methods, EAST and DBnet, performed equally in terms of generalization ability. The results also show that the methods have an ability to generalize their performance to different domains with most instances producing an average f1-score around 0.6. The small difference in performance indicates that the models are able to learn universal features that are inherent to text from different datasets. The models are able to generalize their abilities quite well, but the universal performance is obviously still inferior to human performance.

6.2 Pooling benchmark datasets

One hypothesis that was held going into the experiment was that it was possible to achieve better generalization abilities by pooling existing datasets together. The reasoning behind this hypothesis is that larger and more varied datasets will force the model to learn univeral features inherent to text and not make use of domain

specific features that can be helpful in single benchmark dataset. One example of this is curved text, if the training data contains no examples of curved text then the model can learn that text "always" is straight and use that feature to help detect text. This will lead to poor performance on curved text, and is not an issue for humans as we are generally able to detect text independent of the curvature. This exact effect was clear in the results of the experiment. The model instances which did not include TotalText in the training data performed considerably worse on the TotalText test set compared to instances which included TotalText in the training data. This effect was clear even for the EAST instance which was trained on all datasets, where TotalText consists of less than 3% of the images, when looking at the recall metric. The main reason why this effect on the generalization abilities for instances trained on data incuding TotalText did not show in the evaluation of the performance of RecFut is due to the biased nature of that dataset. Google vision's text detection algorithm does not perform nearly as well on curved text as on straight text and thus the RecFut dataset contains a lower proportion of curved text than the complete set of images provided by Recorded Future.

The average f1-scores do not really show evidence clear evidence in support of the hypothesis of pooling datasets to improve generalization abilities. The best performing instance of EAST is the one that was trained on only the ICDAR datasets. The overall best-performing instance was the DBnet trained on ICDAR and TotalText, but it was only slightly better than the DBnet model trained on only the ICDAR datasets(approx. two percentage points).

Overall the results show some signs that pooling together existing datasets can improve the generalization abilities, but the results were not clear-cut.

6.3 Image classification

The experiment where EAST an DBnet was used as image classifiers, showed promising results. All instances of both models produced a recall >94% showing that they are able to correctly classify most images which contain text. The differences in performance between the two models reflect the results that were observed in the previously discussed experiment. EAST has a higher precision than DBnet, which is consistent with the findings in the experiment were the generalization abilities were analyzed. The segmentation of the text is irrelevant in the classification task, and thus EAST's recall does not suffer as much from the segmentation issues that were previously observed compared to DBnet. Two of the DBnet instances have very high recall (>99%) which comes at the cost of significantly lower precision compared to EAST.

High recall is a desirable property of an image classifier in the intelligence business where Recorded Future operates. The value of implementing image classification in

order to determine which image should be analyzed more thoroughly depends on several factors. If high recall comes at the cost of low precision then the number of images that are actually "filtered out" can be quite low, depending on the proportion of images that don't contain text. In the set of 1499 images provided by Recorded Future approximately 27% contain no text and, using the model which achieved the highest recall (DBnet trained on TotalText), less than 11% of images were classified as no text. If the distribution of images in the set provided by Recorded Future is representative for all images that are stored, this means that the classification step only would be worthwhile if the cost of classifying all images is less than 11% of the cost of performing complete analysis of the images. In addition, there is a small risk of missing out on images which could contain valuable information that the complete analysis potentially could have detected.

6.4 Future work

The field of scene text detection needs more research focusing on generalization abilities of the models. The current standard in the field when presenting new models is to present results on a few selected benchmark datasets. This serves mainly as an indication of the potential of a model and can lead to cherry-picking datasets and authors can choose to only include competitive results and leave out unflattering results. Also it only shows how models perform when they are trained specifically for a given domain. In addition to reporting results on single datasets, the field should consider reporting average results like the macro-average that was calculated in this thesis. This is a way to quantify the generalization abilities of the models. If the field could agree on a selection of datasets to include for this purpose then it could serve as a standardized measure for the generalization abilities of scene text detection models.

Closely related to the issue of generalization abilities for scene text detection models is the problem of maximizing performance on a target domain that is different from the training domain. This is called *domain adaptation* and is a field of research within machine learning and computer vision. Domain adaptation has been applied with some success in the field of scene text detection, mainly aiming to use synthetic data for training models which can perform well on real world data sets [30],[26]. Domain adaptation is a very interesting area which can prove valuable in the context of scene text detection due to the difficulty in obtaining real world training data and the varied nature of the task.

Other future directions of work is to develop evaluation protocols which can better deal with MO and OM matching of ground truth and predicted bounding boxes in order to better reflect the true performance of a model. Under- or over-segmentation of the text by the detection model might not necessarily lead to loss of information as the text detection in almost all cases is the first step in an end-to-end pipeline. It is important that the evaluation reflects how the detections will perform in an end-

to-end pipeline. This is a difficult problem as it depends on the quality of the text recognition model and other potential steps/heuristics in between text detection and recognition so it is difficult to standardize an evaluation procedure that isolates the text detection step.

7
Conclusion

The thesis work presented in this report was conducted in collaboration with Recorded Future and the main focus was to analyze the generalization abilities of scene text detection algorithms. Scene text detection is the field of detecting text in images and for Recorded Future this is relevant as part of a potential pipeline that can extract intelligence information from text in images. Two experiments were performed to analyze the abilities of two selected scene text models, EAST and DBnet.

The first experiment consisted of training several instances of EAST and DBnet with different benchmark datasets and evaluating the performance of all instances on several benchmarks. The results show that both the models have some generalization abilities but they have very different characteristics. DBnet is able to achieve a recall over 0.5 for all datasets while the precision was less stable. EAST, on the other hand, achieved a precision of over 50% while the recall was more variable. Both models produced instances which achieved an average IoU f1-score above 0.6 which clearly indicates that they are able to generalize the text detection abilities to a certain extent. In order to better represent the generalization abilities of proposed scene text detection models, macro-averaging the f1-scores achieved by single instances over several datasets should be reported more frequently in the litterature. If the field "scene text detection) were able to agree on a set of benchmark datasets suitable for this metric, then it could become a benchmark in its own right.
Visual inspection of the results showed that EAST had issues segmenting the text on a word-level in sentences and this affected the recall negatively. This effect really illustrates the need for good evaluation protocols which handle MO- and OM-matching which was not the case in this study. Overall the results show that both models are able to generalize the text detection to universally be able to detect text to a certain degree, but the performance is still much lower than the capacity of humans.

The second experiment consisted of using EAST and DBnet as image classifiers to classify images as containing text or not. Both models achieved promising results, with recall well above 90% on the data set provided by Recorded Future. EAST has significantly higher precision and slightly lower recall than DBnet. These results correspond to the findings in the first experiment with regards to the different characteristics which were observed for the two models.

Bibliography

[1] Jeonghun Baek, Yusuke Matsui, and Kiyoharu Aizawa. What if we only use real datasets for scene text recognition? toward scene text recognition with fewer labels, 2021.

[2] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection, 2019.

[3] Chee Kheng Chng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition, 2017.

[4] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017.

[5] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. Pixellink: Detecting scene text via instance segmentation, 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Justin Johnson and Taghi Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6:27, 03 2019.

[8] Kye-Hyeon Kim, Sanghoon Hong, Byungseok Roh, Yeongjae Cheon, and Minje Park. Pvanet: Deep but lightweight neural networks for real-time object detection, 2016.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[10] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization, 2019.

[11] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2017.

[12] Yuliang Liu, Lianwen Jin, Zecheng Xie, Canjie Luo, Shuaitao Zhang, and Lele Xie. Tightness-aware evaluation protocol for scene text detection, 2019.

[13] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era, 2018.

[14] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes, 2018.

[15] Shilpa Mahajan and Rajneesh Rani. Text detection and localization in scene images: a broad review. *Artificial Intelligence Review*, 54, 08 2021.

[16] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[17] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation, 2016.

[18] Fatemeh Naiemi, Vahid Ghods, and Hassan Khalesi. Scene text detection and recognition: a survey. *Multimedia Tools and Applications*, 03 2022.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[20] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units, 2016.

[21] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments, 2017.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[24] Bala R. Vatti. A generic solution to polygon clipping. *Commun. ACM*, 35(7):56–63, jul 1992.

[25] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images, 2016.

[26] Weijia Wu, Ning Lu, and Enze Xie. Synthetic-to-real unsupervised domain adaptation for scene text detection in the wild, 2020.

[27] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015.

[28] Michael Yeung, Evis Sala, Carola-Bibiane Schönlieb, and Leonardo Rundo. Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation, 2021.

[29] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. UnitBox. In *Proceedings of the 24th ACM international conference on Multimedia*. ACM, oct 2016.

[30] Gangyan Zeng, Yuan Zhang, Yu Zhou, and Xiaomeng Yang. A cost-efficient framework for scene text detection in the wild. In Duc Nghia Pham, Thanaruk Theeramunkong, Guido Governatori, and Fenrong Liu, editors, *PRICAI 2021: Trends in Artificial Intelligence*, pages 139–153, Cham, 2021. Springer International Publishing.

[31] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: An efficient and accurate scene text detector, 2017.