University of West Bohemia
Faculty of Applied Sciences

Department of Cybernetics

Optical character recognition using deep learning

Bc. Pavel Andrlík

Advisor: Ing. Marek Hrúz Ph.D.

Pilsen, 2022

Západočeská univerzita
Fakulta aplikovaných věd

Katedra Kybernetiky

Čtení textů pomocí metod hlubokého učení

Bc. Pavel Andrlík

Školitel: Ing. Marek Hrúz Ph.D.

Plzeň, 2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

Studijní program: Aplikované vědy a informatika
Forma studia: Prezenční
Obor/kombinace: Kybernetika a řídicí technika (KŘTnp)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení:	**Bc. Pavel ANDRLÍK**
Osobní číslo:	**A20N0023P**
Adresa:	Litohlavy 188, Litohlavy, 33701 Rokycany 1, Česká republika
Téma práce:	Čtení textů pomocí metod hlubokého učení.
Téma práce anglicky:	Optical character recognition using deep learning.
Vedoucí práce:	Ing. Marek Hrúz, Ph.D. Výzkumný program 1

Zásady pro vypracování:

1. Seznamte se s implementací vhodného sytému pro automatické čtení textů založeném na hlubokých neuronových sítích.
2. Připravte trénovací, validační a testovací sadu počitačově generovaných obrázků s textem.
3. Natrénujte a otestujte vybraný systém na těchto datech.
4. Vyhodnoťte výsledky a navrněte případná zlepšení.

Seznam doporučené literatury:

Bušta, M., Patel, Y., & Matas, J. (2018, December). E2e-mlt-an unconstrained end-to-end method for multi-language scene text. In *Asian Conference on Computer Vision* (pp. 127-143). Springer, Cham.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum:

Declaration

I hereby declare that this master's thesis is completely my own work and that I used only the cited sources.

Plzeň, 23th May 2022

Pavel Andrlík

Abstract

This diploma thesis deals with the problem of optical character recognition (OCR) using neural networks. I am focusing on improving text detection and OCR by fine-tuning an E2E-MLT scene text detector by training it on synthetic data which emulates real data. The model was fine-tuned on several datasets with synthetically generated data and real data, then the models were tested on one synthetic and two real datasets, one with the majority of the wild text, the second with the majority of TV news imprinted text. On the dataset with majority of TV news imprinted texts the fine-tuned models achieved improvement by decreasing character error rate from 52% to 31.6% word error rate and from 56.5% to 22%. It was also experimentally discovered that training models on synthetic data simulating real TV news images deteriorate detection and reading model capability on wild text data.

Keywords

neural network, optical character recognition, scene text detector, deep learning, data generating

Abstrakt

Tato diplomová práce pojednává o problému optického rozpoznávání znaků při použití neuronových sítí. Zaměřuji se na zlepšení detekce a rozpoznávání textu pomocí dotrénování E2E-MLT scénového detektoru textu tak, že ho trénuji na umělých datech, která napodobují reálná data. Model byl dotrénováván na několika datasetech obsahujících uměle generovaná a reálná data, poté byly vybrány nejlepší modely a otestovány na jednom umělém a dvou reálných datasetech, jeden s převahou divokého textu, druhý s většinou textu vtištěného televizním zpravodajstvím. Na datasetu s většinout digitálně vložených textů bylo dosaženo zlepšení snížením chybovosti znaků z 52% na 31.6% a chybovosti slov z 56.5% na 22%. Během experimentů bylo také zjištěno, že trénování modelů na umělých datech simulující skutečné obrázky ze zpravodajství zhoršuje schopnost sítě detekovat a číst reálné divoké texty.

Klíčová slova

neuronová síť, optické rozpoznávání znaků, detektor textu ve scéně, hluboké učení, generování dat

Contents

1 Introduction

1.1 Optical Character Recognition

Optical character recognition, otherwise OCR, is one of the most important and most commonly used branch of computer vision. OCR consists of two main parts which are text area detection and text reading. The goal of the part of text area detection is to make a polygon or most often a square around the detected text. The Detected text could be a letter, word, or a whole sentence, but it is mostly a word. The main task of the second part, the part of text reading, is to correctly classify each letter and in the case of reading texts also correct mistakes such as letter duplicity or superfluous spaces. The texts could be standardized special fonts for official documents, handwritten scripts, normal or special fonts at various sizes or graphical text as on advertising banners or logotypes.

Figure 1: An Example of E2E-MLT [6] results on the ICDAR MLT 2017 data-set [34].

In OCR reading, the quality of input data is crucial. It is one of the main parameters which determine reading success rate, therefore preprocessing is commonly used. In preprocessing we adjust brightness, contrast, focus or zoom to achieve the possibly best quality, the biggest contrast of letters and background and the most similar size of letters.

Preprocessing with high quality is not always possible. In daily routine e.g. when using a camera on a mobile phone, we never have ideal or often even a good light condition, a perfect focus, the biggest contrast or same size of text with the same font. Due to the real conditions when the read text is very diverse, systems like template matching cannot be used.

After the big boom of neural networks around 2012 after AlexNet [25] was introduced, the stance on OCR changed. Neural networks are based on big training data sets which quality directly influences obtained results and reached success rate. Data sets could be created in two ways, first is by annotating a huge amount of images by hand, which requires a lot of money and a lot of time, the second one is by creating a synthetic data generator. The goal is to create such a generator that the generated images will be indistinguishable from reality.

1.2 Motivation

As mentioned in the Section 1.1, in deep learning, the quality of data is one of the main criteria which determine the later quality and precision of OCR reading. In my bachelor's work [1], I have created a program that generates synthetic data imitating the television news. My motivation in my diploma thesis is to verify that the generator of synthetic data could replace real data or even overcome the real data by the generator's ability of producing almost infinite amount high-quality images.

Figure 2: An illustration of real image (left) and synthetic image (right) with same text generated with AITGM module [2].

The scientific motivation is to determine which way leads to better results of reading digitally generated text in television. Whether a better way is to invest time or money to annotate big amount of real data with only limited volume, or whether it is better to invest time or money to creating or improving synthetic data generator. The final statement is going to be proved by experiments containing graphs and result tables.

2 Related work

Optical character recognition with the task to detect and correctly read all texts in the image is one of the most important and mostly used thing in computer vision. This topic has been addressed by researchers since first computers in 1960. Over the years, methods and algorithms have been improving from a single template matching working with one font to a deep neural network providing very good recognition even for handwritten fonts with outstanding results. Namely, the end-to-end systems were an improvement because in training the input is image and the output is detected text. There is no need to make preprocessing or to detect text before its reading separately, only to have a dataset with annotated images, which makes it powerful.

2.1 Optical Character Recognition by traditional approaches

A first method for reading or most likely assigning character on paper to a letter in char-set was pattern matching in 1960. Over the years until 1980's, OCR were improved but the main idea of approach to problem remained still. The main idea was to first make a preprocessing in the form of noise reduction, thresholding and segmentation. Then for feature extraction, three main methods were used. Those were a) Points distribution, b) Transformations and series expansions and last c) Structural analysis meaning geometric and topological structures of a symbol. After feature extraction was made a deciding was next step. There were three main used classifiers and these were the minimum distance classifier, statistical classifiers as e.g. Bayes classifier and neural networks. Neural networks were described as a non-perspective method, because they have limited predictability and generality which looks amusing in today's optic. Systems based on mentioned principles were used with technique of Magnetic Ink mostly in bank sector or at state authorities due to its high price [12].

Around the year 2000 as computing power increases, new methods of feature extraction and classification have appeared. For the feature extraction, I would like to mention a representatives as Histogram of gradients and Scale-invariant Fourier transform. For the group of classifiers the mains are AdaBoost [9] and Support vector machines [44].

2.2 Text detection and recognition using Neural Networks

There are three branch of computer vision reading using neural networks. These are **Text detection** [24, 22, 17, 45, 49], **Text recognition** [27, 42] and **End-to-end methods** [5, 28, 30]. Representatives of each branches will be presented in further subsections.
In branch of text detection and text, recognition are almost all systems designed in the way that there is a regressor as the main part, it is always some type of Neural Network, and auxiliary algorithms provide sorting, filtering, thresholding and NMS (Non-Maximum Suppression). In addition, there may be another neural network for geometric transformation of parameters or pseudo-parameters such as feature vectors to position parameters etc.
End-to-end methods in text reading are characterized by addressing both text spotting and text recognition consequently without the need for any third part intervention. They can also

use the same neural network for text detection and recognition, respectively same features from various layers. Architecture could be complemented by other neural network layers such as mostly used convolutional or fully connected layers.

2.2.1 Reading Text in the Wild with Convolutional Neural Networks from Jaderberg et al.

Jaderberg et al. [22] designed a six-stage method where horizontal-boxes are the first stage predicted by combining Aggregate Channel Feature Detector [11] and Edge Boxes [50]. Then at the second stage they used Random Forest [4] classifier for reducing proposals, more precisely to decrease false-positive detections. The next two steps are using CNN for refining bounding box proposals estimates and performing text recognition for refined text regions. Detections are merged by proximity and recognized text. Also, a score is assigned to each merged region. The last stage is thresholding for the final text spotting result by the assigned rating. The disadvantage of this approach is that recognized words have to be from dictionary so it is not a lexicon-free system and word probability is gained from word distribution which depends on training data. As a final product is text box with recognised word.

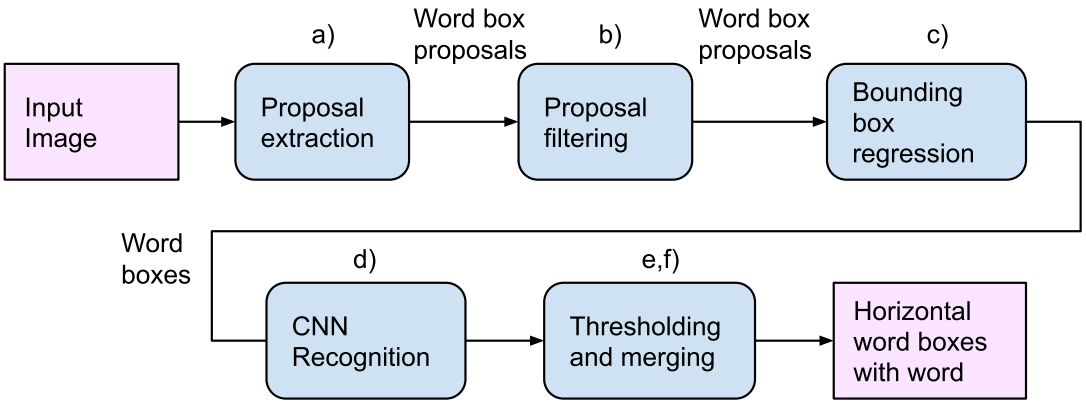


Figure 3: Text spotting and reading pipeline proposed by Jaderberg et al. [22].

In more detail works proposed model illustrated in the Figure 3 as follows: a) A combination of Edge Boxes and Aggregate Channel Feature Detector. b) Filtering by Random Forest classifier. c) A CNN is used to refine bounding boxes position. d) A VGG-16 [43] was retrained to estimate probability that the cropped region defined by bounding box belongs to class, where number of classes is 90 000 words in English dictionary. e) Merging refined detections by proximity and recognised words with adding score. Score is maximal probability mixed with probability of word in the word distribution. f) Thresholding by score obtained in previous step.

2.2.2 Synthetic Data for Text Localisation in Natural Images from Gupta et al.

Gupta et al. [17] proposed a YOLO object detector [37] like the network with a fully-convolutional regression network and trained it on synthetic data. Architecture of the network was inspired by VGG-16 [43] which means the usage of several small dense filters however, Gupta et al. found out that a smaller model is as good and even more efficient for text. As predictors they implemented seven 5×5 linear filters each for regressing one of seven object pose parameters **p**. **P** is composed of four position parameters, two rotation parameters and one object presence confidence. They also implemented multi-scale detection because of limited receptive field arising from anchor box sizes and the inability of spotting large texts. The input image is resized down by scales 1, $1/2, 1/4, 1/8$. Final detections are obtained by suppressing lower score detections with a condition of their overlapping. The advantage of this model is that it has 30 times less parameters then YOLO [37] network and its are 45 times smaller.

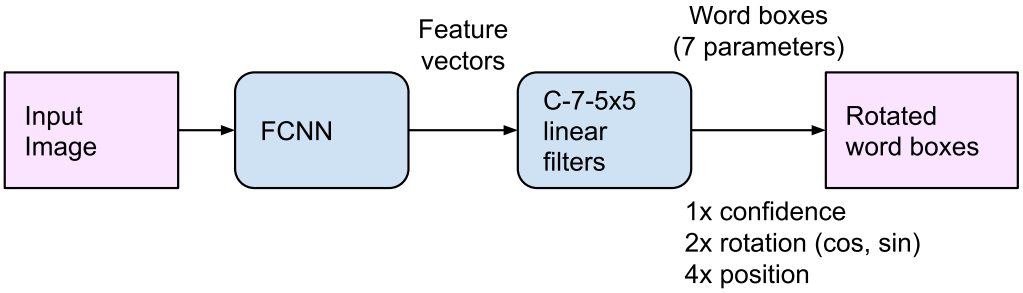


Figure 4: Text spotting pipeline proposed by Gupta et al. [17]. A fully-convolution neural network (FCNN) composed of nine convolutional layers with ReLU and sometimes followed by max-pooling. Object pose parameters (position, rotation) are *(x-u,y-v,w,h,cosΘ, sinΘ)*.

2.2.3 Detecting Text in Natural Image with Connectionist Text Proposal Network from Tian et al.

Tian et al. [45] introduced an idea of using Connectionist Text Proposal Network (CTPN) supplemented by a Bi-Directional Long short-term memory network (BLSTM) [16, 41]. First a VGG-16 [43] is used to generate feature maps which feeds BLSTM. After that fine-scaled output from BLSTM is attached to the fully-connected layer working as an transformation of feature vector into pose coordinated. Because of output of BLSTM which is k-sequential vertically oriented thin rectangles are pose parameters in shape $2k$ vertical coordinates, $2k$ scores and k side-refinement where k is number of vertical thin rectangles. Width of thin rectangles is fixed to 16px.

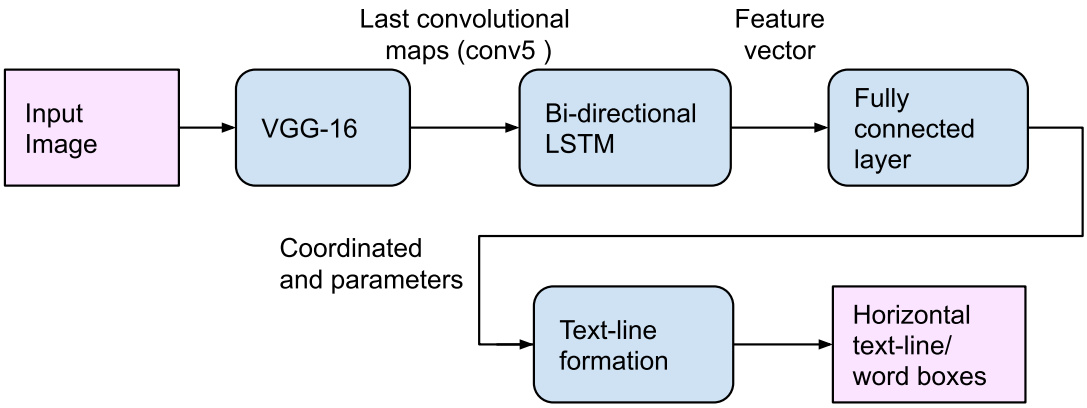


Figure 5: Detecting Text in Natural Image with Connectionist Text Proposal Network from Tian et al. [45].

In more detail, VGG-16 [43] last layer (conv5) is densely slid by a 3×3 spatial windows. Windows are recurrently connected by row to a Bi-Directional LSTM [16, 41] which afterward provides a feature vector for a fully connected layer. Output values of fully connected layer contains these parameters: coordinates of thin vertically oriented rectangles, text or no-text score, and side-refinement. The final area with detected text is defined by the amount of thin rectangles.

2.2.4 EAST: An Efficient and Accurate Scene Text Detector by Zhou et al.

In 2017 Zhou et al. presented EAST - An Efficient and Accurate Scene Text Detector [49] which uses U-shape [40] network design. U-shape designed network is the key component of their approach because then they can eliminate intermediate steps such as word partition, candidate proposal and text region formation. Post-processing steps are only thresholding followed by Non-maximum suppression (NMS) on predicted geometric shapes. The interesting part is NMS because they do not use the standard version because of complexity $O(n^2)$ and instead a Locality-Aware NMS is used with a best case complexity $O(n)$. Due to often highly correlated near by pixels can be geometries merged row by row and iteratively merge the current geometries with the last merged, all happens in same row. In article EAST is stated that in practice, the algorithm runs sufficiently fast as long as the locality assumption holds.

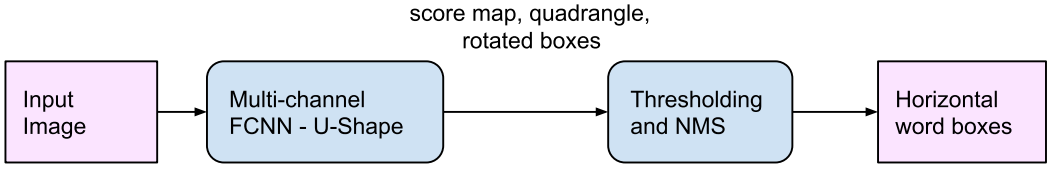


Figure 6: EAST: An Efficient and Accurate Scene Text Detector by Zhou et al. [49].

Network has four convolution stages followed by four unpool stages where in each unpool stage is merged unpooling features with features from convolution branch of same stage. Thresholding is done on score map which is one of predicted channels and it just consider score over predefined threshold as valid.

2.2.5 An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition from Shi et al.

A lexicon-free text recognition network was presented in 2016 by Shi et al. [42]. The proposed network is composed of the convolutional part, where are extracted features from image followed by mapping to sequences. Feature sequences then continue to a deep Bi-Directional LSTM [16, 41] which provides character sequence for Connectionist Temporal Classification (CTC) [15]. CTC is a transcription layer where duplicities, inner word spaces etc. are deleted. Inner word spaces, duplicities or miss-classified characters of the same letter could occurred because feature sequence which can be represented as a rectangle in an original image covers only half of the letter and so the letter is split into two or more rectangles. They also proposed a lexicon-based transcription in the form of method added after CTC where a word from a dictionary is chosen by its minimum distance to generated CTC string.

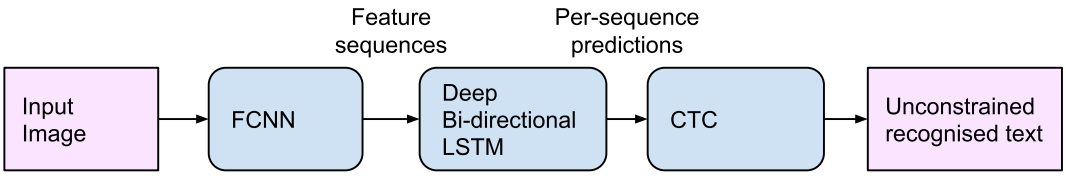


Figure 7: An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition from Shi et al. [42].

FCNN in Figure 7 composes of five convolutions each followed by max-pooling and then three convolution layers are used each with batch normalization. Feature maps are then mapped to feature sequences that represent the receptive field in form of thin rectangles vertically oriented. These continue into Bi-directional LSTM and result in character sequence as one character for one feature sequence.

2.2.6 Recursive Recurrent Nets with Attention Modeling for OCR in the Wild from Lee et al.

Lee et al. [27] introduced in 2016 a recursive recurrent neural network model and showed an approach of recursive CNN complemented by recurrent NN in the form of soft-attention mechanism. For obtaining the text from image are firstly used four recursive convolution

layers provides feature extraction. Then feature maps are transformed by two fully connected layers and brought into a recurrent networks. Soft-attention mechanism decode image features into output characters concerning for implicitly learned and saved character-level language model. In this form can model provides a lexicon-free image to text transcription or rather a recognition however, there are still included language statistics inside soft-attention layers.

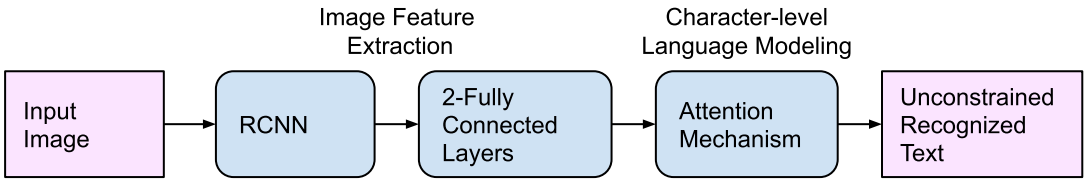


Figure 8: Recursive Recurrent Nets with Attention Modeling for OCR in the Wild from Lee et al. [27].

Recursive CNN (RCNN) is specific by sharing weights across layers and keeping the same number of channels. A recurrent network with a soft-attention mechanism implicitly contains learned language statistics which helps or forces to generate meaningful results without the need for a lexicon.

2.2.7 Towards End-to-end Text Spotting with Convolutional Recurrent Neural Networks from Li et al.

Li et al. [28] created a complex end-to-end model for text spotting and recognition. He makes use of convolution layers for feature extraction in the first place, then a Text Proposal Network make a list of text region proposals which is fed into Region Feature Encoder (RFE). RFE provides convolutional features conversion into fixed-length representation because of following Text Detection Network (TDN) which is a multi-layer perceptron (MLP) giving textness score and calculating bounding box offsets. MLP output is bring back (dashed line in Figure 9) into RFE for again computing fixed-length bounding boxes provided by TDN. In the last step Text Recognition Network (TRN) recognizes characters in the bounding boxes with usage of attention to focus in "time steps" on specific slide of given bounding box. During a time step is generated one character.

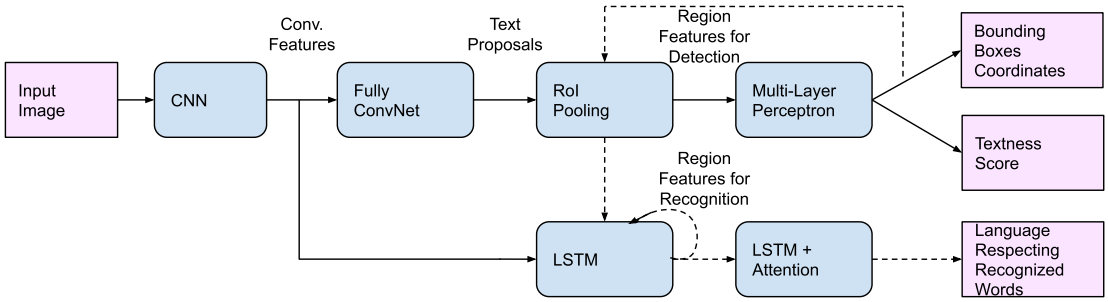


Figure 9: Towards End-to-end Text Spotting with Convolutional Recurrent Neural Networks from Li et al. [28].

While describing model more precisely, CNN is modified VGG-16 [43] by removing fully connected layers and keeping only 1st, 2nd and 4th max-pooling layers so down sampling ratio is changed from $1/32$ to $1/8$. Fully ConvNet extracts convolutional features and provides set of bounding boxes and coordinates offset. RoI pooling is spatial max-pooling trying keep ratio between width and length at rate $2w{:}h$ which is be beneficial for characters like "i" or "l". In Multi-layer perceptron part are two fully-connected layers with 2048 neurons followed by two parallel layers as is shown in figure. Standard LSTM [20] is accompanied by attention mechanism because in each time step attention generates heat map in form of sliding window.

2.2.8 Deep Text Spotter: An End-to-End Trainable Scene Text Localization and Recognition Framework from Bušta et al.

Bušta et al. [5] introduced an approach based on fully convolutional network (FCNN) with adapting the YOLOv2 [38] architecture because if its accuracy and significantly lower complexity than the standard VGG16 [43] architecture. As in YOLOv2 and Faster R-CNN [39] they use a Region Proposal Network (RPN) for region proposals generation and in addition a rotation θ is added as a crucial parameter for recognition. Proposals are filtered by its score r_p that give the probability that the region contains text. Because regions have different scales and rotations a bilinear sampling is used instead of classical RoI pooling. Bilinear sampling allows network normalise rotation and size with respect to original aspect ratio and positioning with giving to this approach an advantage. Finally a CTC layer [15] give a conditional probability distribution over labeled sequences for choosing the most probable characters sequence.

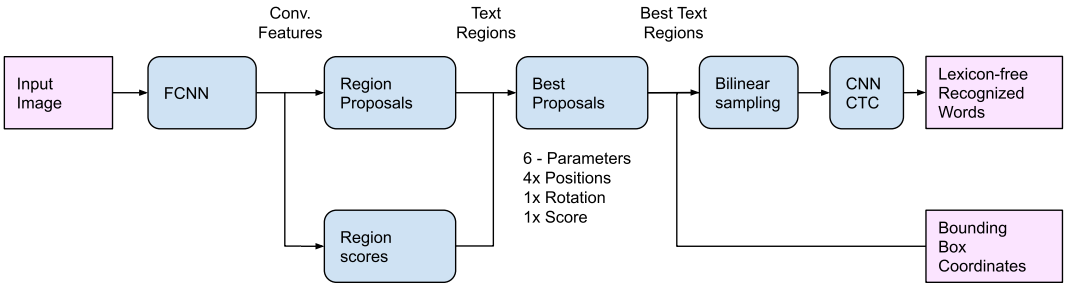


Figure 10: Deep Text Spotter: An End-to-End Trainable Scene Text Localization and Recognition Framework from Bušta et al. [5].

Adopted YOLOv2 network has deleted fully-connected layers so it is fully convolutional. Region proposal is a Region Proposal Network (RPN) supplemented with rotation r_θ. Best proposals box means selecting regions with score higher than 0.1. Bilinear sampling normalise rotation and scale with leaving original aspect and position unchanged.

2.2.9 FOTS: Fast Oriented Text Spotting with a Unified Network from Liu et al.

FOTS: Fast Oriented Text Spotting with a Unified Network is a system from Liu et al. [30] presented in 2018. Liu makes use of U-shaped [40] convolutional network for generating feature maps in scale $1/4$ considering input image by upscaling features from $1/32$ ratio followed by two parallel branches, text detection and text recognition. In text detection branch one convolution layer is applied to generate per pixel text probability and distances to all sides for each positive sample. Feature map is then processed with thresholding and NMS and rectangle areas with text are generated. If a quadrangle as in [49] have been used then in my point of view they could have better straightened detected text which is obvious in original article [30] on their Figure 4 at top right bbox with text "TEMT". In text recognition pipeline RoI rotate is fed with predicted bounding boxes and feature map in scale $1/4$ for obtaining axis-aligned feature maps with fixed feature high and unchanged aspect ratio. Then by a sequence of VGG-like [43] sequential convolutions with pooling, one bi-directional LSTM [16, 41] and fully-connected layer and finally CTC decoder [15].

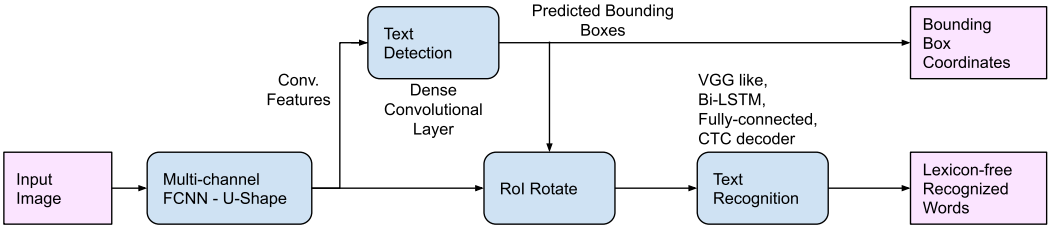


Figure 11: FOTS: Fast Oriented Text Spotting with a Unified Network from Liu et al. [30].

More precisely described Figure 11, FCNN encodes features to $1/32$ scale and then upscale back to $1/4$ size of input image by bilinear upsampling. Text Detection contains one deep convolution layer with six channel providing per-pixel text detection in one channel. For each positive sample are computed distances to all sides (4 channels) and the last channel predicts the orientation of the related bounding box. Then thresholding and NMS is applied. RoI rotate make use of affine transformation for axis-aligned straightening with fix height and unchanged aspect ratio.

3 E2E-MLT

An Unconstrained End-to-End Method for Multi-Language Scene Text or shortly E2E-MLT [6] was presented in 2018 by researchers from ČVUT in Prage, namely Michal Bušta, Yash Patel and Jiří Matas. They drew inspiration from modern growing cities where the same information can be written in several languages and different scripts like Latin, Chinese or Cyrillic alphabet, all together at one poster as in Figure 1. Even though deep learning helped a lot in text detection [17, 22, 24, 49, 45] and text recognition [22, 42, 27], there still exists a branch, a multi-language scene texts, where methods of text detection or text recognition as previously mentioned or other more complex methods as [28, 5, 30] fall short because of the following aspects. A) only English texts are trained and evaluated, so if multiple language detection and recognition in the same scene are needed, there have to be used more networks which could lead to mistakes, B) text localization and recognition are not solved together as a dependent but as two independent problems [22, 17] which is a mistake e.g. for Chinese or Japanese scripts as they are often vertically oriented, C) most of the existing OCR systems are insufficient for highly rotated texts.
One of the challenges of multi-language scene text is the fact that it is not sufficient amount of publicly available training data for deep neural networks.

Figure 12: Multi-language text detection and recognition demonstration by E2E-MLT [6].

Because of all reasons stated above Bušta et al. proposed a single end-to-end trainable fully convolutional network with shared convolutional layers for both tasks, scene text detection and text recognition with script identification. The method can recognize 7 500 characters in the original version and 8 400 in the newer one, it doesn´t use any fix dictionary and it solves all the listed lack previous solutions.
This method was chosen because it was trained primarily on synthetic dataset and network shown to be good at generalising and ability of learning from synthetic data. Next reason is that network is able to recognise Czech characters without any changes however, it has never seen Czech characters before so it has to be retrained or more likely only fine-tuned.

Model could be different from model described in article [6] because Bušta made newer one after article was released, so a newer changed model is used in this work.

3.1 Overall architecture

The architecture of this method is different from the others [28, 30, 5] because it is based on a single fully convolutional network. The network has three parts, see Figure 13, the first part is a base image processing followed by the two parallel branches, the text localisation branch and the text recognition branch. The first part contains four segments each made from 2D 3×3 convolution (Conv2D), Instance Normalisation (IN) [46, 8] and Concatenated Leaky Rectified Linear Unit (CLReLU) [32] and provides dimension reduction with feature extraction to $1/4$ size of height and width compared to the input image, with 64 channels. As the backbone of the text localisation branch is used ResNet-34 [19] with Feature Pyramid Network (FPN) object detector [29] as a core. First layer of ResNet-34 was replaced by 3×3 convolutions with stride 2 because the text itself is small compared to the whole image area, so 7×7 convolution window is too big. Because high resolution consumes a lot of memory, the proposed detector works on a scale $1/4$ of the input image same as it does FPN object detector [29]. Stride is a shift of convolution window and it determines the size of next layer by relation $1/stride$.

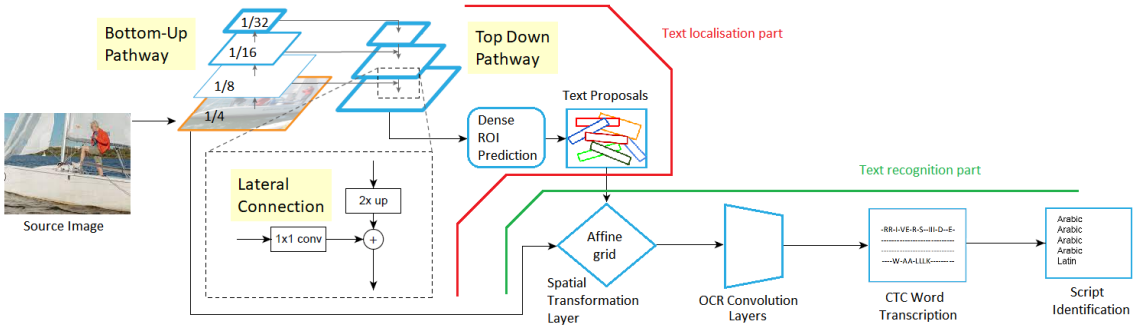


Figure 13: E2E-MLT model architecture overview: FPN [29] architecture is used to generate dense text proposals by extracting information from shared layers. Next OCR branch takes cropped feature regions according to the text proposals and generates a character sequence or remove the proposal as non-text.

The architecture of E2E-MLT model is graphically illustrated in Figure 13 concerning to all used techniques. All parts of the network are presented in more detail in further chapters 3.2, 3.3, starting with the initial part different from ResNet-34 [19] in Table 1. The initial part is important because it is advantageous to work with lower resolution due to memory consumption. This part is also interesting by the fact, that channel increase is not made by convolution but by CLReLU function defined in Eqn. 3.

Initial part			
Type	Channels	Size/Stride	Dimension
Conv2D, IN, CLReLU	16	3 × 3/1	W × H
Conv2D, IN, CLReLU	32	3 × 3/2	W/2 × H/2
Conv2D, IN, CLReLU	64	3 × 3/1	W/2 × H/2
Conv2D, IN, CLReLU	64	3 × 3/2	W/4 × H/4
Text localisation part			
Type	Channels	Size/Stride	Dimension
ResNet block ×3	64	3 × 3/1	W/4 × H/4
ResNet block ×4	128	3 × 3/2	W/8 × H/8
ResNet block ×6	256	3 × 3/2	W/16 × H/16
ResNet block ×4	512	3 × 3/2	W/32 × H/32
Dropout (0.2)			
FPN lateral con	256	1 × 1/1	W/16 × H/16
FPN lateral con	256	1 × 1/1	W/8 × H/8
FPN lateral con	256	1 × 1/1	W/4 × H/4
Dropout (0.2)			
Conv2D	7	1 × 1/1	
Text recognition part			
Type	Channels	Size/Stride	Dimension
Conv2D, IN, LReLU	128	3 × 3/1	W/4 × 10
Conv2D, IN	128	3 × 3/1	W/4 × 10
Conv2D, LReLU	128	3 × 3/1	W/4 × 10
maxpool		2 × 1/2×1	W/4 × 5
Conv2D, IN, LReLU	256	3 × 3/1	W/4 × 5
Conv2D, LReLU	256	3 × 3/1	W/4 × 5
Conv2D, LReLU	256	3 × 3/1	W/4 × 5
Conv2D, LReLU	256	3 × 3/1	W/4 × 5
Conv2D, LReLU	256	3 × 3/1	W/4 × 5
maxpool		2 × 1/2×1	W/4 × 2
Conv2D, IN, LReLU	256	2 × 3/1	W/4 × 2
Dropout (0.2)			
Conv2D	$\lvert\hat{\mathcal{A}}\rvert = 8400$	1 × 1/1	W/4 × 1

Table 1: Layer by layer E2E-MLT architecture is based on a fully convolutional network (FCN). W and H represent the width and height of the original image and the dimension column shows the size of the feature map after each layer. IN - Instance Normalisation [46, 8], CLReLU - Concatenated Leaku ReLU [32] as Eqn. 3. ResNet block - ResNet-34 [19] convolutional block, FPN - Feature Pyramid Network [29] object detector with lateral connection in detail depicted on Figure 13.

Concatenated Leaky Rectified Linear Unit - CLReLU
In the late 1960s has appeared a new activation function with development of visual feature extraction which was called Rectified Linear unit (ReLU). ReLU is defined as a positive part of its argument, mathematically as:

$$f(x) = x^+ = max(0, x) \tag{1}$$

and it is sometimes also called a ramp function. This function shows as advantageous because it solves many issues and it helps training and improve performance as experimentally proved [14]. The main issues in training which appeared during time are a) gradient vanishing, b) gradient explosion and c) slow training mainly because symmetric activation functions cause training stagnation. After that in 2014 Bing et al. [48] did an empirical study of ReLU and its modifications as **Leaky ReLU**:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \frac{x}{a} & \text{if } x < 0 \end{cases} = max(0, x) + min(0, x)a_i \tag{2}$$

where a is the fix parameter and determines a negative slope of function. Some small negative slope brings an experimentally proved additional advantage in form of better results when training deep neural networks.

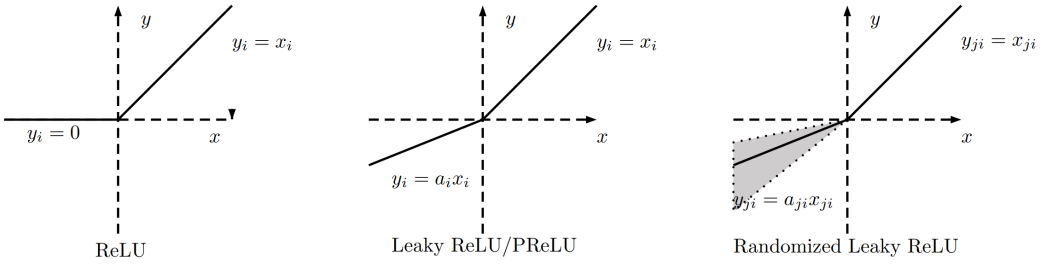


Figure 14: Illustration of ReLU, LReLU, PReLU and RReLU [48]. In LReLU a_i is a fixed constant but in PReLU it is by learning trained parameter. RReLU samples a_{i_j} as a variable in a uniform random distribution in given range.

Parametric ReLU (PReLU) was presented by He et al. [18] in 2015 and defined same as ReLU only with the small change that parameter a is not a fix number but it is a learned parameter during backpropagation training. Also a version **Randomised Leaky ReLU** was presented but it has not achieved any bigger successes. Only difference against Leaky ReLU is that a_{i_j} is a random number generated from the uniform distribution *U(l,u)*.

Concatenated Leaky ReLU
Concatenation is in this model used for channel multiplication. The number of channels is multiplied by 2 and it is created as a concatenation of tensors x and *-x*.
The whole formula for the activation function used in the model (Table1) in the initial part is:

$$CLReLU = LeakyReLU(InstanceNorm2D(Concatenation(x, -x))) \tag{3}$$

3.2 Text localisation

When the input image with text scene passes through the initial and then through localization part of the network, on the output, appears seven per-pixel text scores and geometries. These seven scores are: **a)** text/no-text confidence score $r_p \in (0, 1)$, **b,c,d,e)** distances to the **b)** *right*, **c)** *left*, **d)** *bottom* and **e)** *top* sides of the bounding box containing this pixel, **f)** angle of rotation r_θ.
For localisation and recognition is used same joint loss function which consists of four mutually independent loss functions.
Those are:

$$L_{final} = L_{geo} + \lambda_1 L_{angle} + \lambda_2 L_{dice} + \lambda_3 L_{CTC} \tag{4}$$

In original article, Bušta et al. [6] used $\lambda_1 = \lambda_2 = \lambda_3 = 1$, but for fine tuning in this master's thesis have been values changed to $\lambda_1 = 2$, $\lambda_2 = 0.5$ and $\lambda_3 = 1$.
L_{geo}: Loss function for geometrical shape which is invariant to text regions of different scales. An IoU loss proposed in EAST from Zhou et al [49] is used,

$$L_{geo} = L_{IoU} = -logIoU(\hat{R}, R^*) = -log\frac{|\hat{R} \cap R^*|}{|\hat{R} \cup R^*|} \tag{5}$$

where $\hat{R}$ is estimated rectangle and R^* is ground truth box.
L_{angle}: Angle loss is computed as mean squared error (MSE) of $sin(r_\theta) + cos(r_\theta)$ and their ground truth sine and cosine values. Combination of sine and cosine is used because there exists a discontinuity or rather jump of argument around angles where phase shifts and using a combination of sine and cosine eliminate the issue, because by switching to sine and cosine we can reach a continuous transition everywhere.
L_{dice}: As in [33], a dice loss is proposed to be a loss function for pixel-wise prediction maps or in other words for segmentation. Dice score or sometimes called F1 score is similar to IoU score function, it has same field of values in interval [0,1], but Dice is more soft when penalising mislabelings which is convenient because it helps with imbalance between area of text region and area of non-text region which is basically the rest of the image, so the difference in size is not negligible. Dice loss is defined as:

$$L_{dice} = 1 - \frac{2TP}{2TP + FN + FP} = 1 - \frac{\sum_i^N r_{p_i} g_i}{\sum_i^N r_{p_i}^2 + \sum_i^N g_i^2}, \tag{6}$$

TP represents true positive, FN false negative and FP false positive. Summations is over all points which means over all feature points r_{p_i} same as over all ground truth points g_i, i is pixel index.
L_{CTC}: CTC loss works on the word level and it is further explained in Section 3.3.2.

Last step of text localization is Dense ROI predictions which are generated by two steps: a) filtering feature maps by confidence score r_p through threshold, when only those with r_p higher than threshold value are kept, b) merging points in filtered maps into boxes with Locality-Aware NMS proposed in EAST [49]. Locality-Aware NMS is based on prerequisite that near by points are often highly correlated so they do not have to be passed with complexity $O(n^2)$ but could be passed in the best case of $O(n)$ when using merging row by row and in the same row merging currently encountered element with the last merged one.

Predicted text is bordered not by rectangles but by quadrangles which could be little baffling because loss function for geometry work with representation of rectangle L_{AABB}. The key idea behind it is that a maximal rectangle is cropped from quadrangle and used for loss function as an argument. The prerequisite is that there would not be many or none texts with highly trapezoid shapes.

3.2.1 U-Net vs Feature Pyramid Network

In 2015 was presented U-Net [40] as an improvement to fully convolution networks [31] for image segmentation which used simple down-sampling by convolutional layers and then up-sampling by learned deconvolution filters. U-Net architecture adopted the same principle of convolution compression into feature vectors followed by decompression creating segmentation maps, but with added skip connections.

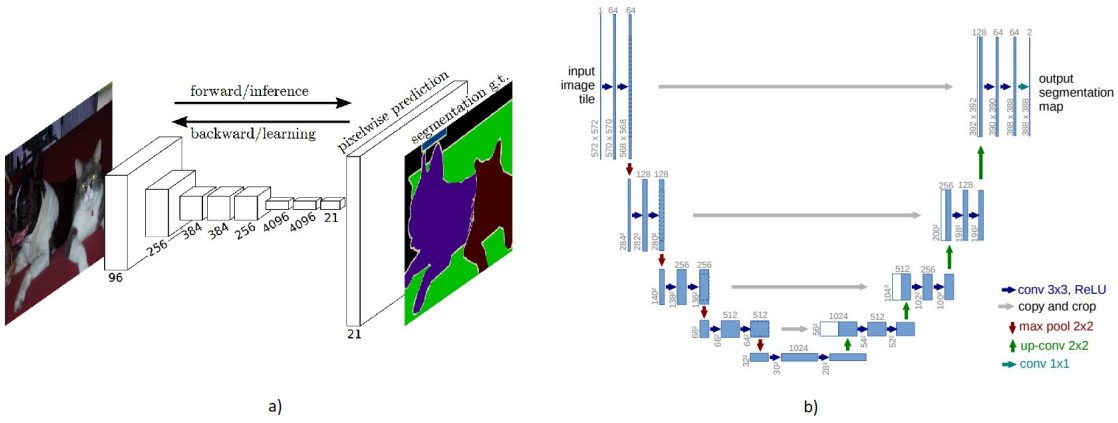


Figure 15: Two important segmentation models. a) Fully convolution network without skip connection [31] b) U-Net architecture using skip connection [40].

U-Net domain is mainly the medical field where was proved by many experiments and works that a piece of extra information from skip connection significantly improves segmentation results. Skip connection in U-Net is created as a simple concatenation of the feature map from contracting part as the next channel to feature maps on the expansive path. The disadvantage of skip connection in the form of a concatenation of high- resolution feature maps to the feature maps on the extensive path and after that processing by 3×3 convolution is that skip connection can only transfer information for segmentation because high resolution feature maps are firm and so the information has to be added and processed to the output segmentation branch in one step by mentioned 3×3 convolution. Another disadvantage is that U-Net like architectures predicts only at the last stage.

In 2016 was presented Feature Pyramid Networks (FPN) for Object Detection [29] which outperform both [31, 40] not only in segmentation tasks but also in the number of implementation options because it can be set specifically with regarding to solved job. It is almost

always used as a module for more complex neural networks because it can hardly ever stand alone due to its properties. In the original article was experimentally proved on many neural network models e.g. Fast R-CNN [13] and Faster R-CNN [39].

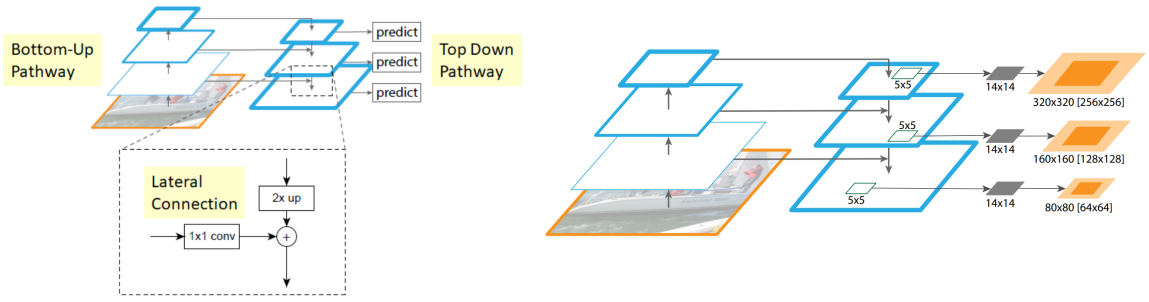


Figure 16: Feature Pyramid Network (FPN) [29]. Left: part focused on lateral connection, an alternative to skip connection. Right: An illustration of the advantage multiple level prediction which reduce no. of multi-scale anchors.

The Architecture of FPN is based on a pyramid scheme with two pyramids, one for feature extraction and second for creating a segmentation map. The first one for feature extraction contains multiple levels of the contracting path which provide feature extraction into smaller feature maps but with the increased number of channels. Feature map size reduction is made by convolution with stride 2 followed by Batch Normalisation and Leaky ReLU. The second part, the expansive path, provides extraction of compressed feature maps with multiple channels into larger maps with fewer channels. Larger feature maps are obtained by bilinear interpolation to higher resolution followed by convolution.

Other methods for upscaling are using deconvolution by trained filters such as [31]. In U-Net is upscaling made similarly as in FPN by bilinear upsampling with convolution. Different approaches were presented in [35] where the first one is unpooling, an operation that places values on the pixel with the same index as where it was before pooling and the second one is deconvolution as shown in Figure 17 also used in [31]. Another option was introduced in DeepLab [7] called atrous convolution by its inspiration from an "algorithm à trous" [21].

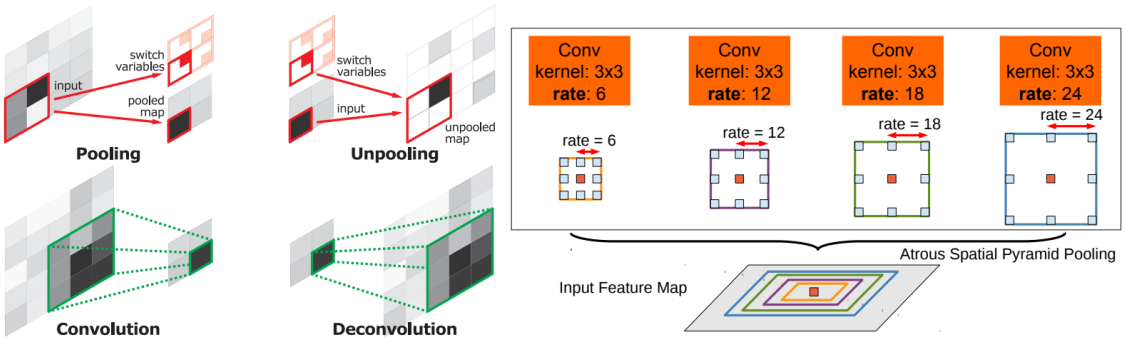


Figure 17: Illustration of the unpooling and deconvolution from [35] with classical pooling and convolution on the left side and atrous convolution from DeepLab [7] on the right side.

Same levels of pyramids are connected by lateral connection which is an alternative to skip connection used in U-Net [40] etc. The name skip connection came into use because in graphical representation of the model several layers were skipped. But it is different with the FPN because model was proposed as bent and its graphical representation is always drawn as up and down the path so the name lateral connection has been chosen since the connection going from side of one pyramid to the side of second one. Connection is not a simple summation or concatenation as in U-Net but there is added a 1×1 convolution which can transform feature maps from contracting path to almost anything. It was proved by Lin et al. [29] that this type of bridging provide better results for object detection.

3.3 Text recognition

The text recognition branch or in other words OCR part consists of a several convolutional layers followed by Instance Normalisation and Leaky ReLU as described in Table 1. Word boxes predicted from localisation part are compared with ground truth boxes and when the IoU is higher than 0.9, the detected text is used for OCR training. Then the parameters for spatial transformer layer [23] are estimated from selected predictions and the spatial transform is applied. Spatial transformation provides not only rotation but also a scale transformation. The affine grid transformation is applied to the input image and the rotated and scaled image part with detected text is than bring on the input of the initial network part (Tab. 1) to compute features again. Therefore the number of channels goes from 64 to 128 in the first OCR layer. Feature from the rotated image will be different than feature from the original image because convolution is only translation invariant, not rotation invariant. An idea for rotating input image with a forward pass and not only rotate features may be that is faster to rotate input image with only three channels and pass it forward through network than rotate features with 256 channels, however Mr. Bušta have not stated the true reason in his article [6]. During image transformation a bilinear interpolation is used to get smoothed image without misalignment pixels and with fixed height but variable width. Words text regions retain its aspect ratio after rotation and scale transform.

After spatial transformer the OCR model or in other words the part of the text recognition described in Table 1 obtain feature tensor with variable-width and fix height and outputs a

matrix containing estimated characters where probabilities of characters are recomputed by log-SoftMax due to CTC loss implementation which is in PyTorch [36] and log-SoftMax is required because of numerical stability. Size of the input feature vector is $\frac{W}{4} \times 40 \times C$ and and size of the output matrix is $\frac{W}{4} \times |\hat{\mathcal{A}}|$ where:
$W = \frac{w \times H'}{h}$
w, h = width and height of text region
H' = fixed height of transformed text region, by Bušta et al. choose 40
C = number of channels
$|\hat{\mathcal{A}}|$ = number of output characters, log-Softmax with 8400 characters.
Than an algorithm similar to CTC [15] transcripts the output matrix into the output text and concurrently during training CTC compute loss function which is used for training the OCR part. For transcription of network output during experiments in original article [6] was used greedy decoding. Other methods of decoding could be language respecting techniques however those could be language dependent which rule out generalisation, some of methods are e.g. language models [22] or attention mechanisms [27, 28]. Due to using CTC decoding is the presented network lexicon-free, language independent and generalized.

3.3.1 Spatial transformer

Spatial transformer [23] is composed of two main steps, a) estimation of the transformation parameters and b) feature map or in general matrix transformation. Parameters for 2D affine transformation, which is sufficient because transformed object is input image, are define by the affine transformation matrix A_θ:

$$A_\theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} = \begin{bmatrix} s_x cos(\alpha) & -s_y sin(\alpha) & t_x \\ s_x sin(\alpha) & s_y cos(\alpha) & t_y \end{bmatrix} \tag{7}$$

Pointwise transformation is then defined by an equation:

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{8}$$

where (x_i^s, y_i^s) is the source map, (x_i^t, y_i^t) are the target coordinates of a rectangular grid, α is the angle predicted from text detection part and $\mathcal{T}_\theta$ is a 2D affine transformation. An illustration of described process is bellow in the Figure 18.

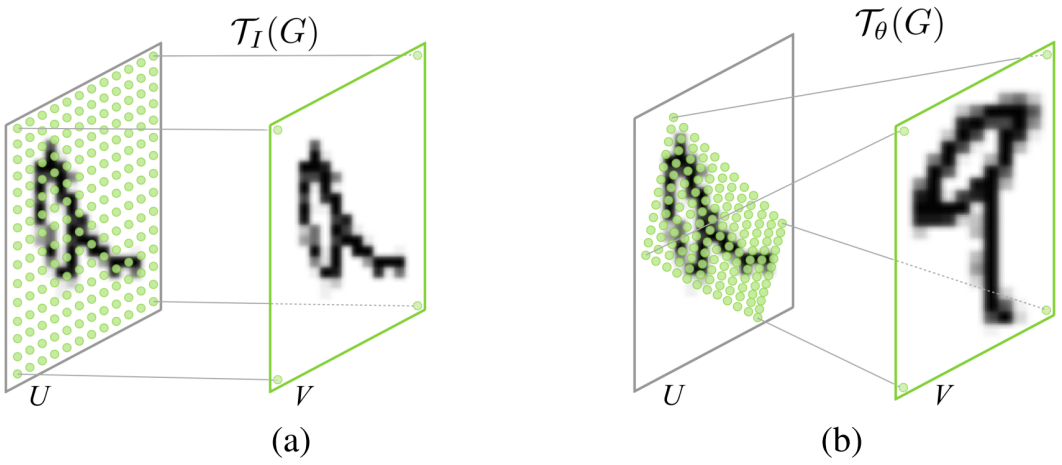


Figure 18: Illustration of an affine transformation fo 2D image by applying the parameterised sampling grid. a) Transformation by sampling grid with identity transformation parameters I. b) Transformation by warped regular grid and with affine transformation $\mathcal{T}_\theta(G)$. [23]

In the Figure 18 in part b) we can see that for one pixel in transformed grid V is not a whole single corresponding pixel but there is multiple pixels in the U grid belonging into a one V pixel. This challenge is solved by bilinear interpolation which mean multiple pixels are combined by average into a single one, this process with a grid is called a bilinear sampling.

3.3.2 CTC - Connectionist Temporal Classification, CTC Loss

CTC [15] was presented in 2006 by Graves et al. as an alternative to the Hidden Markov Model (HMM) for neural networks as output processing. Besides output processing CTC also provides a loss function that is used for backpropagation training.

To clearly explain how CTC works and why is it used for neural networks working with sequential pieces of information is necessary to explain what the last layer of OCR does and what is the other unused way. The last layer is as a sliding windows where the number of windows is determined by the number of recognized characters. Each window is represented as a channel and this window slides over the last feature map with dimension $2 \times \textit{text width}$ by one pixel or by one-time step. Each time step has its own column with responses of windows so multiple time steps create a matrix where are responses of characters over time (moving windows).

$$LogSoftmax(x_i) = log\left(\frac{exp(x_i)}{\sum_j exp(x_j)}\right) \tag{9}$$

On this matrix is applied log-SoftMax (Eqn. 9) function which makes some kind of normalization according to sum not necessarily equal to one. The matrix in this form is convenient because it can be used for back propagation training and also for finding the most probable

text. Windows are hardly ever wider than recognized characters so a window can see the same character for several consequential time steps. Therefore there has to be a symbol to tell where the space between two characters is almost always an underscore "_".

Decoding path is now easy to perform since decoding is simply finding the highest value in each column as illustrated in the Figure 20 in the right bottom corner and then making two simple operations defined by following rules in this order.
1) **Duplicate deleting** - If there are the same characters next to each other, delete one of them.
2) **Blank deleting** - If there is a "blank" character "_", delete it.

The loss function is created a little bit differently because no character is deleted and it is not about choosing the best values or neither the worst. As a first step for obtaining loss function all paths which are after decoding same as ground truth label are detected. This step ensures that all possible alignments are taken in account. Than all possible alignments of input to ground truth target are summed up and reversed since loss function cannot be positive. This process is also illustrated in the Figure 20 with the difference that we do not have to make a logarithm because it has been done in the last NN layer. Logarithm is done because loss function has to be derivable so it can be back-propagated.

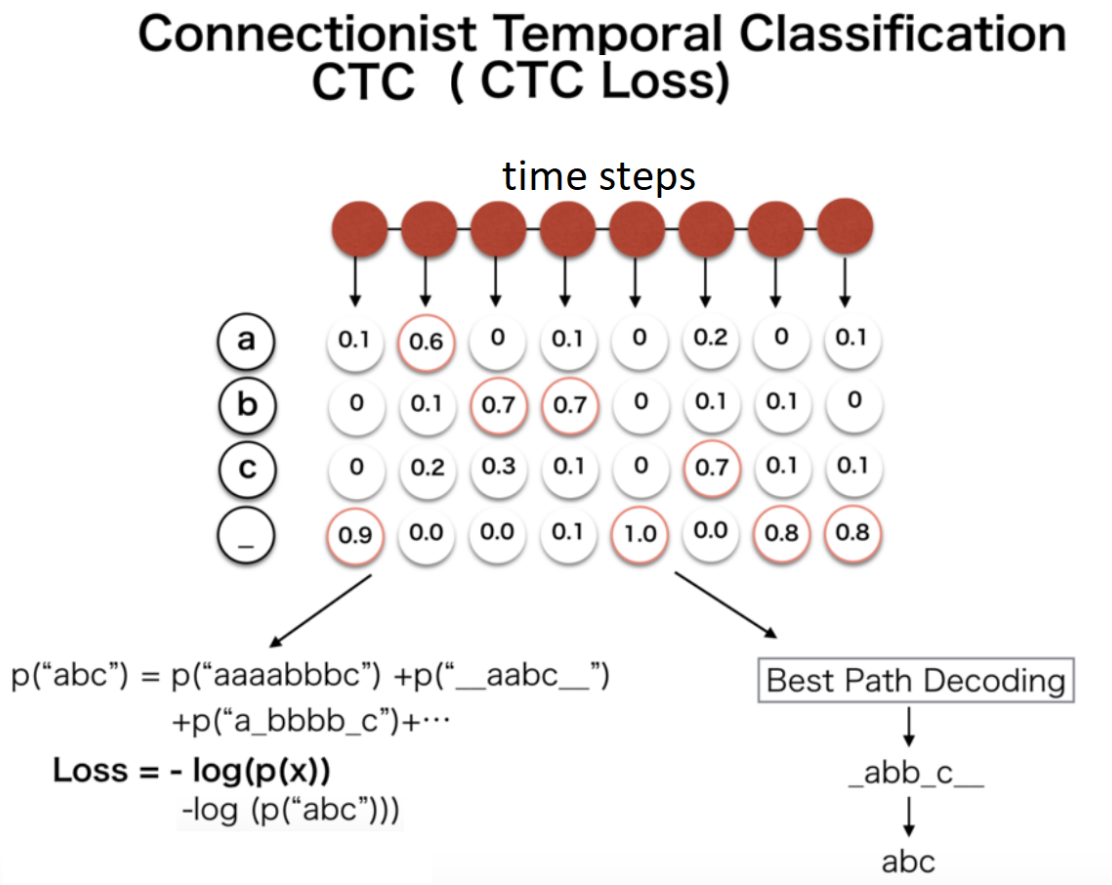


Figure 19: CTC algorithm with loss computation in the left bottom corner and best path decoding in the right bottom corner. Algorithm works with a probability matrix on the input where on the horizontal axis there are time steps and on the vertical axis there are possible characters. The decoding process generates a text which should in the best case match the real text. The loss generates a negative logarithm of sum of probabilities of all possible alignments of the input to the target hence the positions of the characters are not needed since all alignments are taken into account. The negative logarithm is necessary because the loss function cannot be positive and it has to be derivable because of back-propagation.

3.4 Training

Original network have been trained by Bušta et al. in an end-to-end way, it means both parts OCR and localisation part were trained together. As a training dataset were used several united datasets, namely ICDAR 2015, ICDAR RCTW 2017, ICDAR RCC-MLT 2017 and Synthetic multi-language dataset created by framework from Gupta et al. [17]. Whilst the largest ICDAR dataset RCC-MLT 2017 contains 7200 training images and 1800 validation images, neural networks need hundreds of thousands of images to be trained properly. Therefore a synthetic dataset contains 245 000 images based on 8000 backgrounds from [17]. Both the ICDAR RCC-MLT 2017 dataset and the synthetic language dataset are six-language.

Synthetic text is generated into the image by the logic that in real world texts are laying in the uniformly structured and well-defined area with same color. So on the background image is applied several methods providing prospective segmented region where can be imprinted text with respect to region lean and orientation. Imprinted are same six languages as in ICDAR RCC-MLT 2017: Arabic, Bangla, Chinese, Japanese, Korean, Latin. Annotation contains word level bounding boxes, character level bounding boxes, text transcription and script class.

As a training optimizer was used Adam [26] with the parameters setting from PyTorch implementation [36] except base learning rate which was set to $lr = 0.0001$. Joint loss function Eqn. 4 is used for both, text localisation and recognition.

(a) Arabic Scene Text (b) Bangla Scene Text (c) Chinese Scene Text

(d) Japanese Scene Text (e) Korean Scene Text (f) Latin Scene Text

Figure 20: Synthetic data generated with adjusted framework [17].

4 Experiments

E2E-MLT dataset contains only English words specifically text in wild, so for the purpose of this diploma thesis which is detecting and recognizing texts from the news broadcasting had to be network retrained or more likely only fine tuned. Experiments was divided into several parts, where the two main parts are in the first part was network trained by synthetic data (Section 4.1) generated by [3, 2] and in the second part a real annotated data from the news broadcasting provided by the university (University of West Bohemia, Faculty of Applied Sciences, Department of Cybernetics) were used, both for fine-tuning the original network for the purpose of contribution comparison.

		model		
		original	synthetic	real
data	synthetic	test1	test2	test3
	real	test4	test5	test6

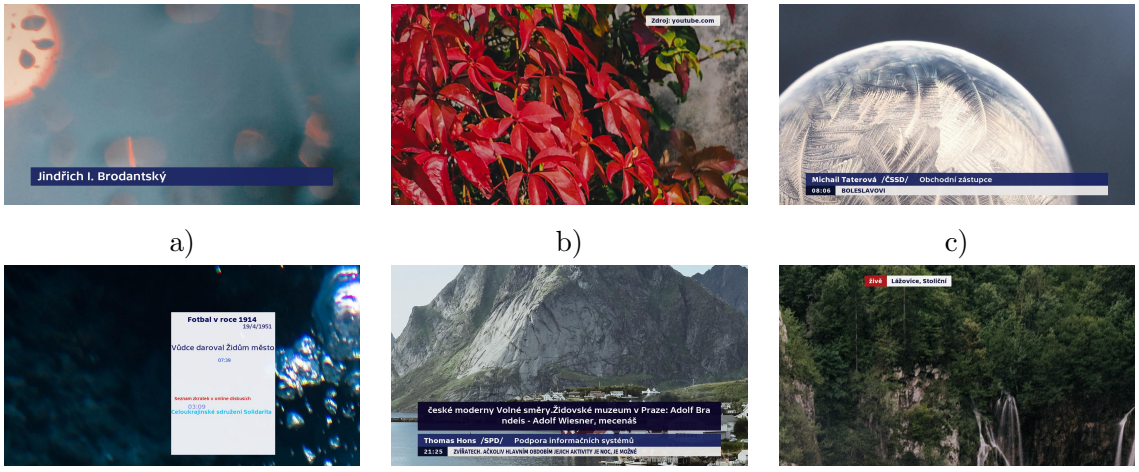
Table 2: Table of experiments by their category.

During experiments were evaluated a) **IoU per image**, b) **precision**, c) **recall**, d) **loss** provided by joint loss function (Eqn. 4), e) **CER** and f) **WER** (all described in section 4.3).

Training parameters of Adam remained unchanged with the respect to previous original training. It is $lr = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon =$ 1e-08, weight_decay $= 0$. Network has been trained on CESNET clusters, mainly on Adan cluster with 2x 16-core Xeon processor and 2x nVidia Tesla T4 16GB.

4.1 Synthetic data

Synthetically generated data is used because of amount of data needed for neural network training. Unlike the real annotated data where can be mistakes caused by annotators, synthetic data can contain only systematic mistakes caused by its creator and those are often revealed and corrected. Data generator [3] uses [2] as a core for generating synthetic images in five patters and one random text. Generator is created in an effort to copy TV news broadcasting which correspond with a goal to generate images unrecognizable from real ones. Patterns are characterised by same positions for semitransparent background and same starting pixel positions in patterns. Basic elements which can generate templates by its combinations are **a)** time, **b)** name and optionally politic party, **c)** job, **d)** report title, **e)** live, **f)** source, **g)** crawl text, **h)** subtitles and **i)** uncategorisable descriptive text. From these are assembled patterns shown in table bellow (Tab. 3).



a) b) c)

d) e) f)

Table 3: Six representatives of possible templates containing elements according to previous paragraph. Individual representatives contains a) report title, b) source, c) time, name, job, political party, crawl text, d) uncategorisable random wild text, e) time, name, job, political party, crawl text, subtitles, f) live.

Distribution of generated templates is set to $16.\overline{6}\%$ alias $1/6$ where the specific case is template e) according to Tab. 3. If the pattern e) is chosen for generating, basic element is subtitles and there are added elements from template c) in 30% cases. Each time is text for imprinting chosen from text files, where there are individual files for almost any part of elements so the whole generator is very flexible.

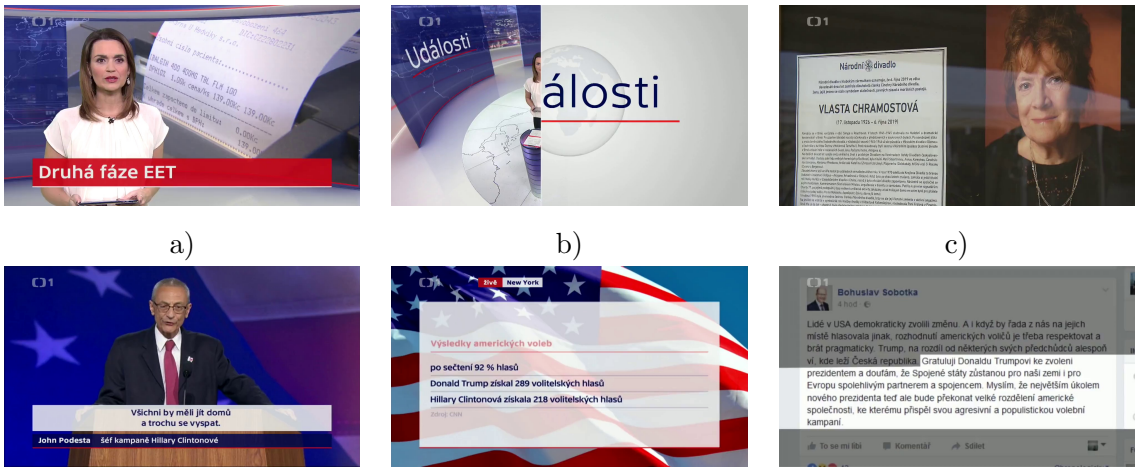
An extra setting is hidden in YAML configuration file which is published and fully set for CT news broadcasting with respect to observed attributes. By adjusting deep setting in configuration file can be generator easily reconstructed for different news broadcast, however it is not designed for the general public, but for machine learning experts and programmers.

Applied font is TV Sans Screen which is non-public font and it is also a private property of Czech Television loaned for this experiment. Font has its three variants called "Bold", "Medium" and "Regular". Text for generating is split into several files according to what is it used for, e.g. first names, surnames, street names, town, web domains etc. From non-public database were obtained names of all streets and towns in Czech Republic. From a public data were compiled surnames, web domains, jobs etc. and the large files with wild and random text or with titles were acquired from Czech Wikipedia dumps.

For training were generated two datasets. The first one is smaller and is composed of 4000 train images, 1000 validation images and 1000 test images. The second one is larger with 16000 train images while validation and test images remained same. Two differently sized datasets were created because it was desirable to ascertain how much is larger dataset significant for recognition.

4.2 Real data

Real annotated data are rare, therefore available amount of data is only 594 images which were split into 400 images for training and 194 images for testing. In synthetic data was background without text, so inside generated image was only imprinted text. In real data is inside images not only text from (Czech Television) CT but also text from real scenes, also called a wild text. This text is not always annotated as it is sometimes hardly visible even for human, however some text are not annotated because annotater appraises text as too small for reading or hardly detectable. From reasons stated above is real data annotation accuracy lower than at synthetic ones.



a) b) c)

d) e) f)

Table 4: Real data demonstration. a) Illustration of report title template however with multiple bevelled real annotated texts, b) Image with half annotation, annotated is only text on the left but recognized will be both text, c) Image where is a lot of wild text but at least 20% is annotated, d) Image also same as template "e" in Fig. 3, e) A representative of uncategorisable image, simulated in synth data by random wild text, f) Different representative of wild text but well annotated which were not simulated in synthetic data.

As could be seen in Table 4 in the picture "c" and as is described in caption, not negligible part of real images has imperfect of more likely incomplete annotation, so because of that is not possible compute precision accurately, and also because of that is recall only indicative. To moderate the impact of images with mostly wild text were from group of testing images separated the minority part of wild text images in quantity of 40 samples. Test group of real data is then divided into two groups with 154 images where is dominant text imprinted by CT, and 40 images where is dominant wild text. Annotations were created in specialized software independent on this work and were made by several different annotators.

4.3 Evaluation metrics

For the purpose of measuring important characteristics of fine-tuned model and comparing it against the original model were chose following metrics normalized to interval [0,1].

- Intersection over Union (IoU)
- Precision
- Recall
- Levenshtein distance
- Character error rate (CER)
- Word error rate (WER)

Annotations and predictions for each image contains list of word boxes defined by four points and their text. Both are not sorted in any way so prediction can not be aligned to annotation just by succession. It cannot be neither assigned by sorting only one, ground truth (GT) boxes or predictions, and then choosing second for pair, because there could emerge situations described in Fig. 21 where multiple GT boxes fits more predictions or where is GT box interspersed into more predictions. In both situations will assigning not be the possibly best one because searching the best assignment is a problem of searching global maximum in matrix of all possible combinations, an $n \times n$ cost matrix. This problem is called **Linear Sum Assignment**.

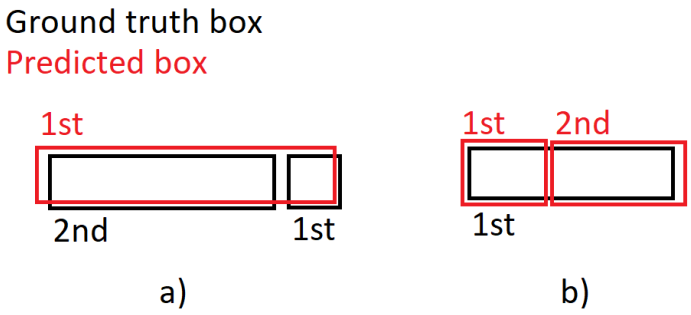


Figure 21: Illustration of two basic situations which have to be taken into consideration. a) a single prediction for multiple ground truth boxes, b) multiple predictions for a single ground truth box.

The algorithm solving a task of an $n \times n$ cost matrix also known as minimum weight matching in bipartite graphs is a "Hungarian method" developed in 1955 with complexity of $O(n^4)$. An improved method with complexity $O(n^3)$ is called Jonker-Volgenant algorithm [10] and it is also implemented in Python SciPy library [47] since version 1.4.0.

4.3.1 Intersection over Union - IoU

IoU was used in two cases, the first one was to compute overlap for assigning predictions to ground truth boxes where assignment is subsequently realized via linear sum assignment.

Computation is done simply on pixel level. Second case was computing IoU for whole image and several situations which could possible happened had to be taken in account while proposing counting method. An illustration in the Figure 21 shows the situation one-to-many and for that reason had to be counting done by uniting all GT boxes, then all predictions and from those two then computing IoU. In this work is uniting implemented as NumPy 2D array. In mathematical formulation is IoU computed as:

$$IoU_{per\_img} = \frac{(\cup \text{ GTs}) \cap (\cup \text{ preds.})}{(\cup \text{ GTs}) \cup (\cup \text{ preds.})}. \tag{10}$$

4.3.2 Precision and recall

First of all have to be defined four term, "true positives", "false negatives", "false positives", "true negatives" with following meaning:

1. true positive (TP) - elements correctly detected
2. false negative (FN) - non-detected elements which should be detected
3. false positive (FP) - elements incorrectly detected
4. true negative (TN) - elements which were not detected and should not be detected

Precision and recall are both metrics for measuring quality of detection and classification. Precision is defined as a ratio between true positives and sum of true positives and false negatives elements. In other words precision metric is a number which says what is the ratio of correct detections to all detections. In the following experiments where elements are GT boxes and predictions could be definition of precision reformulated into

$$Precision = \frac{TP}{TP + FP} = \frac{\#correct\ detections}{\#predictions} \tag{11}$$

where # is an abbreviation for number of.
Recall is measuring the second missing part of quality of the detection which is how many of relevant elements were detected. It is a number telling the ratio of correctly detected elements to all correct elements. Using this formulation can be recall written in mathematical form as

$$Recall = \frac{TP}{TP + FN} = \frac{\#correct\ detection}{\#GTs}. \tag{12}$$

In both, as a "correct detection" is considered a detection with IoU higher than parameter τ. By setting different level of tau, quality of detection could be observed from descending values of precision and recall. The relation between recall or precision and *tau* is as follows, the slower precision or recall decrease when *tau* is increasing, the better is detection.

4.3.3 CER and WER

Character error rate (CER) and word error rate (WER) are both metrics which allows to ascertain how good is character recognition. After matching by linear sum assignment is on

each pair of GT box and prediction applied Levenshtein (LV) distance which is an metric for measuring edit distance between two word sequences. Because Levenshtein distance is used for computing CER and we wanted CER to be between 0 and 1 to indicate error rate of reading, formula for Levenshtein distance had to be adjusted as follows:

$$LV\ dist. = min(LV(prediction, GT), GT). \tag{13}$$

Normalization of LV by length of GT is not done because it is required CER to be weighted by GT length. Due to that fact is CER normalized by sum of all lengths of GTs so each error has same weight independent on word length. Formula for CER is then:

$$CER = \frac{\sum_i LV dist.}{\sum_i length(GT_i)} = \frac{\sum_i min(LV(prediction_i, GT_i), GT_i)}{\sum_i length(GT_i)}. \tag{14}$$

In algorithm for computing CER are under-detections and over-detections also penalized. Because of over-detections could CER grow up over 1, so it could be taken as an indicator of that.

WER is then computed as a count of all words where is Levenshtein distance between prediction and GT text is bigger than 0.

$$WER = \frac{\sum_i \begin{cases} 0 & \text{if } CER_i = 0 \\ 1 & \text{if } CER_i > 0 \end{cases}}{\#GTs} \tag{15}$$

4.4 Results

Validation of training on synthetic dataset - 4000 images

As a first step the original model was trained on synthetic dataset with 4000 training images and validated on dataset with 1000 synthetic images. Training was performed for 30 epochs with parameters as described in Section 4.

After training, models were validated by criteria described in Section 4.3 and by joint loss function L_{final} Eq. 4 with following results.

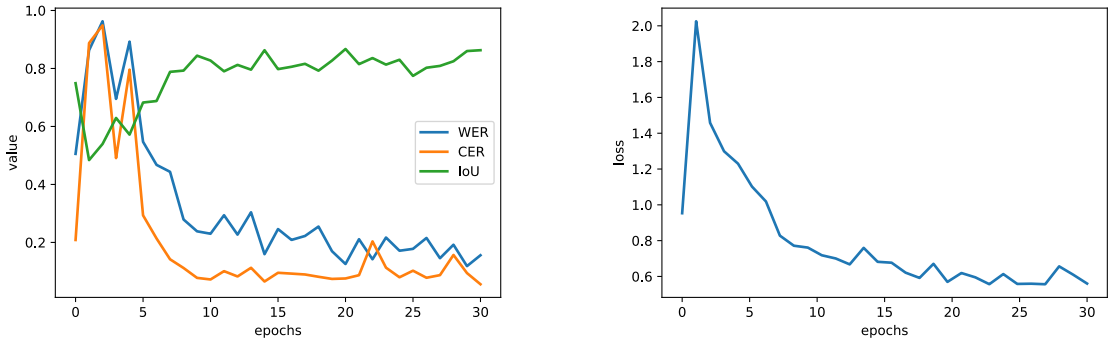


Table 5: Validation curves of model: trained on 4k synth. dataset, validated on 1k synth. dataset.

The best model by CER is the last one (epoch 30) with:

Train data	Val. data	Epoch	CER	WER	IoU
4k synth.	1k synth.	30	0.05575	0.15571	0.86298

Table 6: Validation results on synthetic dataset of model trained at 4k synthetic dataset.

CER and WER are figures for evaluating recognition or reading, however for evaluating detection serves precision and recall figure, where as a correct detection is considered the ones with IoU higher than set threshold τ (tau). The more area under curve is, the better is detection and the more confident the detector is. For the purpose of this work cannot be one value, neither recall or precision, bigger to the detriment of the second one, they should be in equilibrium.

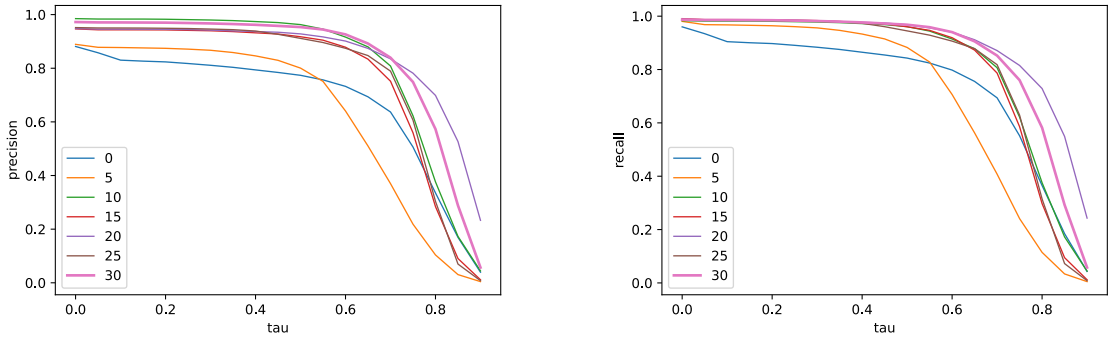


Table 7: Precision and recall curves of model: trained on 4k synth. dataset, validated on 1k synth. dataset.

In the graph are not all model because it will then be labyrinthine and also because important is trend of training, which is obvious from the selection. The best model chose from CER-WER figure in Tab. 5 is in the figure in Tab. 7 highlighted.

Validation of training on synthetic dataset - 16000 images

Next a 16000 training set of synthetic data was used. By this dataset was trained the original model and also the best model from training by 4000 synthetic dataset. Training original model by second four times larger dataset was performed because it was required to determine whether we can reach better results with bigger dataset or whether have synthetic data limited information value. Retraining the best model from previous 4k dataset was done for finding out if new synthetic dataset can ameliorate results.

With experiences from previous training and because of very long training time was training only for 26 epochs. Original model retrained by bigger dataset reached better results shown in Table 8.

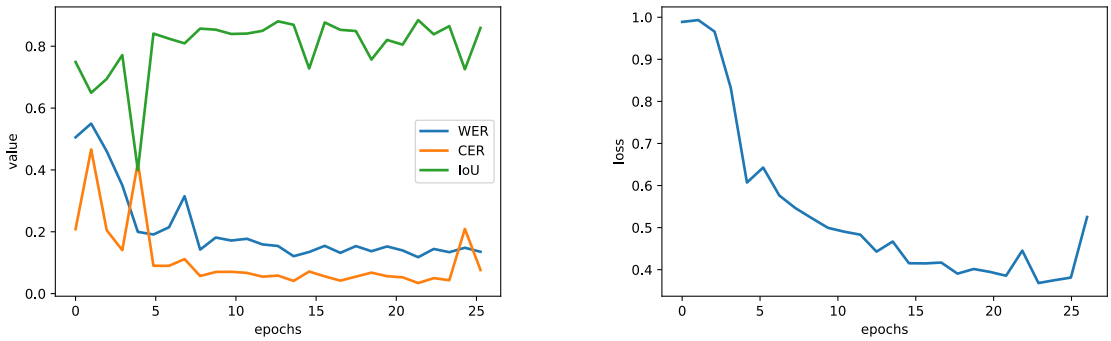


Table 8: Validation curves of model: trained on 16k synth. dataset, validated on 1k synth. dataset.

The best model is model after epoch 22 with results:

Train data	Val. data	Epoch	CER	WER	IoU
16k synth.	1k synth.	22	0.03416	0.11775	0.88463

Table 9: Validation results on synthetic dataset of model trained at 16k synthetic dataset.

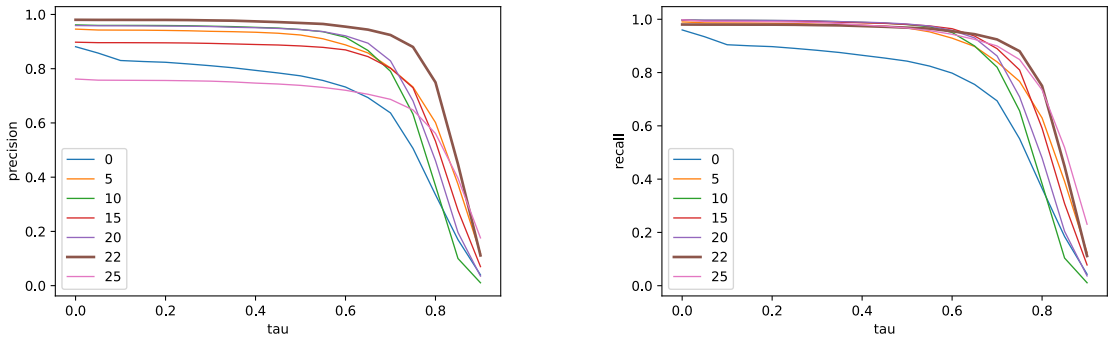


Table 10: Precision and recall curves of model: trained on 16k synth. dataset, validated on 1k synth. dataset.

In the Figures in Tab. 10 is clearly visible that the model of epoch 22 best by CER and WER is also very successful in detection confidence which confirms it was chosen correctly and that CER and WER are correlated with precision and recall through convolution neural network functionality.

Validation of training on synthetic dataset - 4000+16000 images

As a last model trained on synthetic dataset with 16000 samples was the best model pre-trained on 4000 synthetic dataset (Table 6).

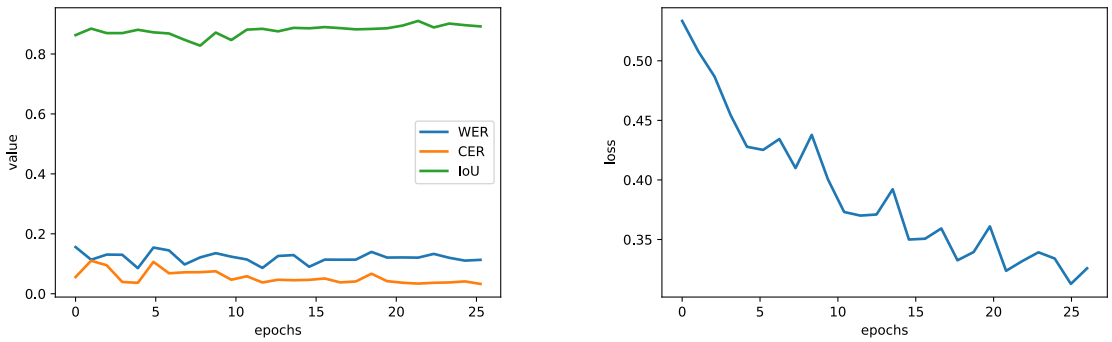


Table 11: Validation curves of model: trained firstly on 4k and then on 16k synth. dataset, validated on 1k synth. dataset.

The best model is model after epoch 4 because of the best WER and CER very similar to others with results:

Train data	Val. data	Epoch	CER	WER	IoU
4k,16k synth.	1k synth.	4	0.03642	0.08534	0.88080

Table 12: Validation CER, WER, IoU of model: trained firstly on 4k and then on 16k synthetic dataset, validated on 1k synth. dataset.

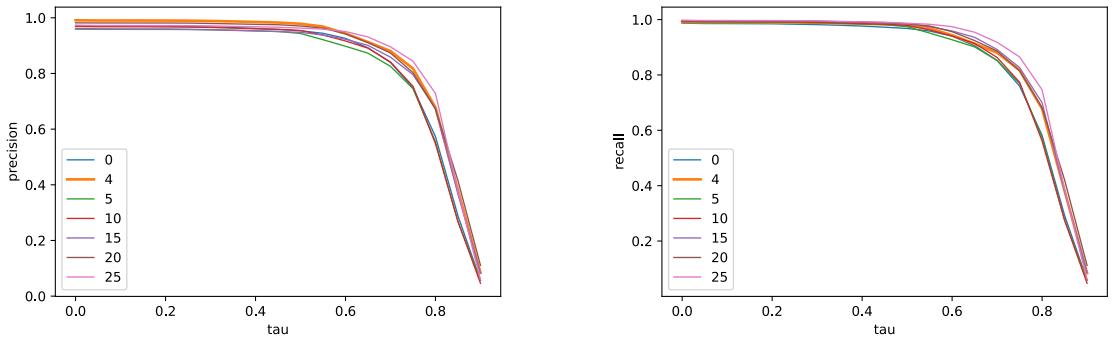


Table 13: Precision and recall curves of model: trained firstly on 4k and then on 16k synth. dataset, validated on 1k synth. dataset.

By plotting precision and recall in the Table 13 could be seen that training with more synthetic data cannot improve detection confidence and only recognition part of model is improved slightly.

Validation of training on real dataset - 400 images

After the training of models by synthetic data was done, the original model was trained using real data. Because of the real data content, meaning that sometimes is missing annotation of text and that some images are unsuitable for training because of texts inside, and because of small amount of real data was expected that training by real images will not be successful. That was proved by experiment where model from training were evaluated on synthetic dataset as models before for choosing the best one for later testing.

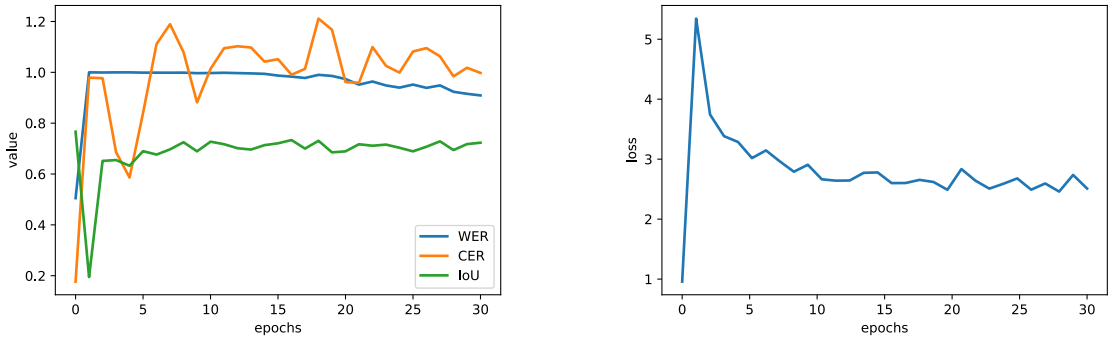


Table 14: Validation curves of model: trained on 400 real images dataset, validated on 1k synth. dataset.

The best model is the one after epoch 4 because of CER is almost two times better than a middle value, however results are much worse than the original model which confirm the supposition that training by the real data we have will not be good.

Train data	Val. data	Epoch	CER	WER	IoU
400 real	1k synth.	4	0.58614	0.99987	0.63254

Table 15: Validation CER, WER, IoU of model: trained on 400 real images dataset, validated on 1k synth. dataset.

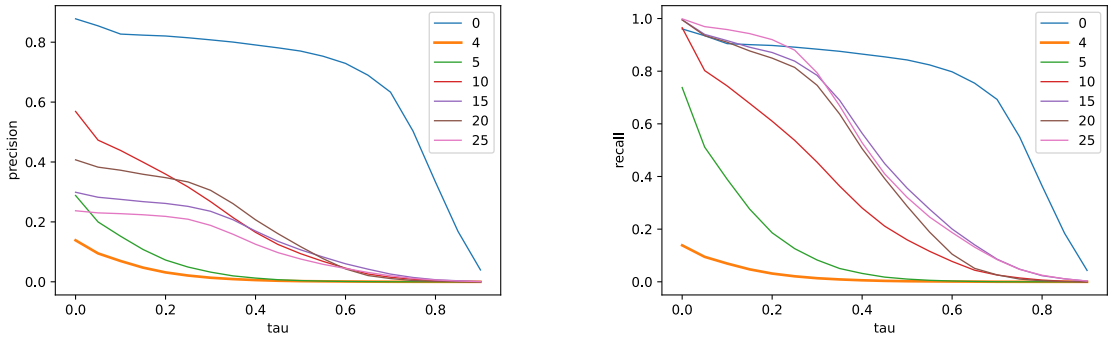


Table 16: Precision and recall curves of model: trained on 400 real images dataset, validated on 1k synth. dataset.

Testing

For testing were real dataset divided in two because real data contains images with more or less wild and generated text, so two groups were created on the bases of text type majority. Besides the real data was models tested also on synthetic dataset with 1000 images. Precision and recall values in Tables 17, 19 and 21 are the maximal values which is if τ is equal to zero.

Testing on 1000 synthetic dataset

Model No.	Train data	Epoch	CER	WER	IoU	Precision	Recall
1	original	0	0.21016	0.50615	0.74510	0.87534	0.96138
2	4k synth.	30	0.07618	0.17342	0.85360	0.96758	0.98442
3	16k synth.	22	**0.04445**	0.13442	**0.87628**	0.96946	**0.99751**
4	4+16k synth.	4	0.06513	**0.08821**	0.86579	**0.97337**	0.99070
5	400 real.	4	0.59558	1.0	0.61643	0.14958	0.48992

Table 17: Testing best models by validating on 1000 synthetic dataset.

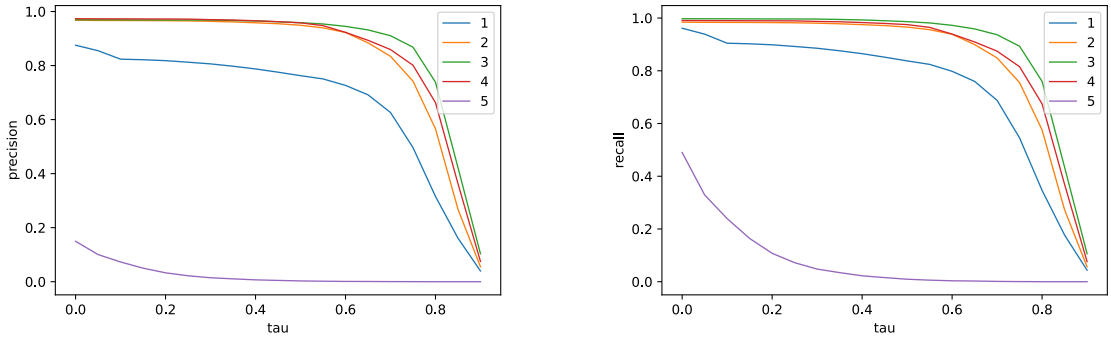


Table 18: Precision and recall curves of best models tested on 1000 synthetic dataset. Legend is derived from Table 17.

Testing on real dataset with majority of imprinted texts

Model No.	Train data	Epoch	CER	WER	IoU	Precision	Recall
1	original	0	0.52002	0.56612	**0.66255**	0.64049	**0.92124**
2	4k synth.	30	0.35155	0.26448	0.65248	0.67710	0.91604
3	16k synth.	22	0.36453	**0.22288**	0.66044	0.67896	0.91604
4	4+16k synth.	4	**0.31623**	0.22511	0.66181	**0.73246**	0.91530
5	400 real.	4	0.94696	1.0	0.48533	0.38942	0.63447

Table 19: Testing best models by validating on real dataset with majority of imprinted texts.

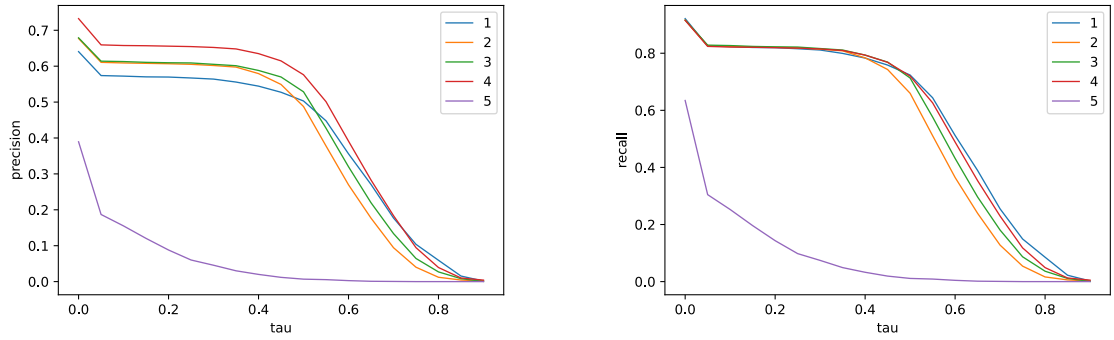


Table 20: Precision and recall curves of best models tested on real dataset with majority of imprinted texts. Legend is derived from Table 19.

Testing on real dataset with majority of wild texts

Model No.	Train data	Epoch	CER	WER	IoU	Precision	Recall
1	original	0	2.01185	**0.57264**	**0.55706**	0.33673	**0.98717**
2	4k synth.	30	1.64962	0.69658	0.50639	0.33984	0.96581
3	16k synth.	22	2.06962	0.58974	0.49636	0.29268	0.97435
4	4+16k synth.	4	2.18962	0.63247	0.49751	0.28719	0.94871
5	400 real.	4	**0.75259**	1.0	0.36846	**0.48571**	0.58119

Table 21: Testing best models by validating on real dataset with majority of wild texts.

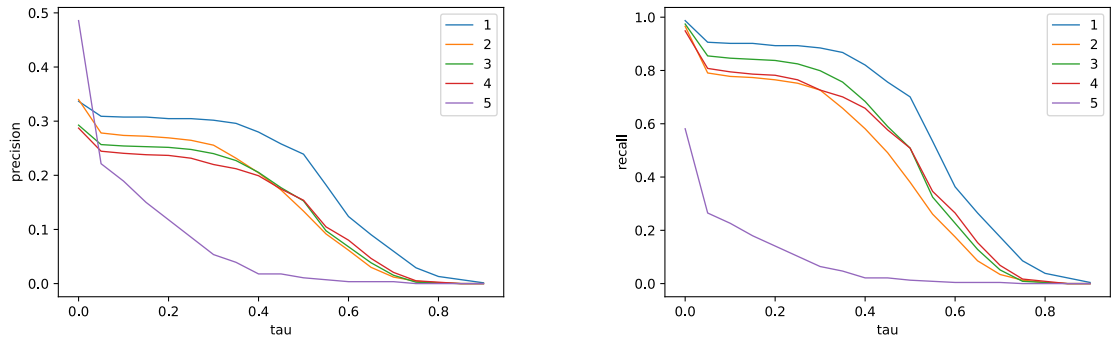


Table 22: Precision and recall curves of best models tested on real dataset with majority of wild texts. Legend is derived from Table 21.

From figures and tables above is evident that models trained on synthetic data reached better results than original model on real images where is majority of imprinted texts. On the images where is wild text was the best model the original one even if CER is high. CER is high because of many over-detection, however not all are over-detections because of some

non-annotated text. Low CER value of model 5 is caused because model tend to under detect and sometimes it hit the most clear letters, so there is not penalization for many fail detections. However it causes that none word is correctly recognized. Low precision of model 5 has same explanation as low CER and at recall is clearly visible that only 58% of boxes with text are hit while other models have from 94% to almost 99%. In the Table 21 is also visible that model 5 hit GT boxes only by a small area because with growing τ is recall and precision rapidly decreasing.

Model trained on real data failed totally in all testings. I blame from such a failure annotations of real data which were not in the top quality for training and the amount of available real data.

5 Conclusion

The objective of this diploma thesis was to discover what is better for training neural network which detects text area in the image and then recognizes text inside. Whether it is better to invest the time in the synthetic data generators or whether it is better to invest the time and money into annotating the real images. Another task was to examine the behavior of E2E-MLT [6] architecture on synthetic data and on Czech characters for future possible enhancement of the architecture.

I have discovered that synthetic data which simulate real image data from TV news broadcasting improved the accuracy of text detection and also text recognition. On the dataset composed of real images where TV news imprinted text is a majority was an improvement of the best model trained on synthetic data against original model is following; In character error rate parameter (CER) there is a 20% absolute improvement reached with the best model by decreasing CER from 52% error rate to 31.6% error rate against the original model. In the word error rate parameter (WER) there is an absolute improvement of decreasing WER of the best model by 34% from 56.5% error rate to 22% error rate. By observing recognized data by hand it was discovered that the most mistakes are in wild text, while in the imprinted text the error is rare.

The next finding was that the training of the model on synthetic data simulating TV news broadcasting harms the detection and the recognition accuracy on wild data. The result of training the original model with the real annotated images was that model deteriorates rapidly and the number of detections decreases. Knowledge consequent from that result is that it is required to use real data for training detection and recognition. The data need to be well-annotated without missing text annotations. Also images should contain well readable words which means that words can be read without guessing them by knowing the real words. Namely smudgy words which can be read by humans by guessing from the context of the real word, words going over the edge where only a half is readable etc. Special attention should be paid to text annotation which should contain information about box or quadrangle rotation, because if the text is rotated from 90 degrees to 270 degrees, there does not exist a way for an untrained network to recognize that the text is inverted and it could deteriorate the results.

From observing behaviour of the current architecture I propose an enhancement of the architecture by changing the number of recognized characters. They are determined by the number of channels in the last layer of text recognition part. Otherwise, the architecture is suitable for training with synthetic data. The model is suitable for recognizing wild text as well as for detecting and recognizing TV news imprinted text.

The results of this thesis build on my bachelor thesis and are the culmination of several years collaboration on a project with Czech Television and the Faculty of Applied Sciences. My work has brought significant improvements to the reading of digitally produced television texts and the trained model is going to replace the one previously used.

5.1 Future work

The future work for improving results on real images where the wild text is side by side with the well-formed imprinted text could be reached by improving the synthetic data generator. The generator used in this work simulates TV news well formed digitally added text, but it does not generate rotated or deformed wild texts. First variant would be to generate a single synthetic dataset with both wild texts generated by adjusted generator from E2E-MLT and TV news texts, and then train the model. In the case that the model would not be capable to learn both text types with sufficient accuracy then the second variant would be to train two separate models, one for TV news text and second for wild texts.

The next improvement could be changing the number of channels in the last layer of text recognition part in the network architecture because now there are 8400 possible recognizable characters. Reducing this number to only Latin characters could help because sometimes network detects a part of a building with windows as Chinese or other exotic characters.

References

[1] Pavel Andrlík et al. Generování obrazových dat pro účely trénování hlubokých neuronových sítí. 2020.

[2] Pavel Andrlík. Tv-news-text-generator, github.com/andrlikp/tv-news-text-generator. 2020.

[3] Pavel Andrlík. Image-scene-text-data-generator, github.com/andrlikp/image-scene-text-data-generator. 2022.

[4] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. Ieee, 2007.

[5] Michal Busta, Lukas Neumann, and Jiri Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proceedings of the IEEE international conference on computer vision*, pages 2204–2212, 2017.

[6] Michal Bušta, Yash Patel, and Jiri Matas. E2e-mlt-an unconstrained end-to-end method for multi-language scene text. In *Asian Conference on Computer Vision*, pages 127–143. Springer, 2018.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[8] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.

[9] X Chen and Alan Yuille. Adaboost learning for detecting and recognizing text.

[10] David F. Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, 2016.

[11] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1532–1545, 2014.

[12] Line Eikvil. Optical character recognition. *citeseer. ist. psu. edu/142042. html*, 26, 1993.

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[14] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[15] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[16] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052. IEEE, 2005.

[17] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2315–2324, 2016.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer, 1990.

[22] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International journal of computer vision*, 116(1):1–20, 2016.

[23] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[24] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep features for text spotting. In *European conference on computer vision*, pages 512–528. Springer, 2014.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2231–2239, 2016.

[28] Hui Li, Peng Wang, and Chunhua Shen. Towards end-to-end text spotting with convolutional recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5238–5246, 2017.

[29] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[30] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5676–5685, 2018.

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[32] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[33] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[34] Nibal Nayef, Fei Yin, Imen Bizid, Hyunsoo Choi, Yuan Feng, Dimosthenis Karatzas, Zhenbo Luo, Umapada Pal, Christophe Rigaud, Joseph Chazalon, et al. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1454–1459. IEEE, 2017.

[35] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[38] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[41] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[42] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[44] Pongsametrey Sok and Nguonly Taing. Support vector machine (svm) based classifier for khmer printed character-set recognition. In *Signal and information processing association annual summit and conference (APSIPA), 2014 Asia-Pacific*, pages 1–9. IEEE, 2014.

[45] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *European conference on computer vision*, pages 56–72. Springer, 2016.

[46] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[47] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[48] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[49] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.

[50] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.