České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství

Obor: Aplikace informatiky v přírodních vědách

Optické rozpoznávání znaků na naskenovaných historických plakátech pomocí nejmodernějších metod

Optical Character Recognition on Scanned Historical Posters Using the State-of-the-Art Methods

VÝZKUMNÝ ÚKOL

Vypracoval:	Anna Gruberová
Vedoucí práce:	Ing. Adam Novozámský, Ph.D.
Rok:	2022

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství

Akademický rok 2021/2022

ZADÁNÍ VÝZKUMNÉHO ÚKOLU

Student:	**Bc. Anna Gruberová**
Studijní program:	**Aplikace informatiky v přírodních vědách**
Název práce česky:	**Optické rozpoznávání znaků na naskenovaných historických plakátech pomocí nejmodernějších metod**
Název práce anglicky:	**Optical Character Recognition on Scanned Historical Posters Using the State-of-the-Art Methods**

Pokyny pro vypracování:

1. Seznamte se s problematikou optického rozpoznávání znaků. Na základě rešerše vyberte několik metod, se kterými budete dále pracovat a vyhodnocovat úspěšnost jejich detekce.
2. Stáhněte několik volně dostupných datasetů, které jsou využívány v literatuře k porovnání jednotlivých metod na OCR. Dále vytvořte svůj vlastní dataset z obdržených dat.
3. Nastudujte techniky porovnání OCR výstupů s ground-truth.
4. U vybraných metod prostudujte jejich chování na jednotlivých datasetech při různém nastavení parametrů.
5. Navrhněte také možnosti filtrování výstupů jednotlivých metod za účelem snížení falešných detekcí.

Doporučená literatura:

[1] R. C. Gonzalez, R. E. Woods, Digital Image Processing (4th ed.). Pearson, 2018. ISBN 9353062985.
[2] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning. Cambridge, Massachusetts: The MIT Press, [2016]. ISBN 0262035618.
[3] SMITH, R. An Overview of the Tesseract OCR Engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2 [online]. IEEE, 2007, 2007, s. 629-633. ISBN 0-7695-2822-8. ISSN 1520-5363. Dostupné z: doi:10.1109/ICDAR.2007.4376991
[4] CHEN, Xiaoxue, et al. Text Recognition in the Wild. ACM Computing Surveys [online]. 2021, 54(2), 1-35 [cit. 2021-10-2]. ISSN 0360-0300. Dostupné z: doi:10.1145/3440756

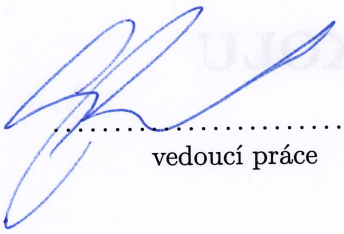
Jméno a pracoviště vedoucího práce:

Ing. Adam Novozámský, Ph.D.
Computer Vision Lab, Institute of Visual Computing & Human-Centered Technology,
TU Wien - Faculty of Informatics

Datum zadání výzkumného úkolu: 15. 10. 2021

Termín odevzdání výzkumného úkolu: 31. 8. 2022

V Praze dne 15. 10. 2021



.....
vedoucí práce

.....
vedoucí katedry

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracovala samostatně a použila jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ....

Anna Gruberová

Poděkování

.

Anna Gruberová

Název práce:

Optické rozpoznávání znaků na naskenovaných historických plakátech pomocí nej

Autor: Anna Gruberová

Obor: Aplikace informatiky v přírodních vědách

Druh práce: Výzkumný úkol

Vedoucí práce: Ing. Adam Novozámský, Ph.D.
Computer Vision Lab, Institute of Visual Computing & Human-Centered Technology, TU Wien - Faculty of Informatics

Konzultant: –

Abstrakt: .

Klíčová slova: .

Title:

Optical Character Recognition on Scanned Historical Posters Using the State-of-t

Author: Anna Gruberová

Abstract: .

Key words: .

Contents

Chapter 1

Optical Character Recognition

Optical character recognition (OCR) is a branch of digital image processing. Its aim is to detect and convert a text on an image into a machine-readable text. This discipline can be divided into three similar, yet different tasks: reading text on scanned printed documents, reading of handwritten texts and scene text recognition (also called text in the wild). The first task is very well developed, first successful results date back to the second half twentieth and were used in commercial sector [19]. If we assume the handwritten text is on scanned single colored paper or created using a digital pen and that it was written legibly and without omitting letters in words due to fast writing, then recognition is similar to printed documents. The last task – scene text recognition (STR) is the most challenging one. The main factors that make STR a more difficult task are listed below.[5, 21]

- Complex background: in scanned documents background is white and without a distinctive pattern (omitting lines is an easy preprocessing task), while in scene images there are objects that can be mistaken for letters.
- Text diversity: text can appear in various colors, fonts, sizes or orientations.
- Distortions: photographs often suffer from noise due to bad illumination, also from motion or out-of-focus blurring, perspective distortion due to the capturing angle. Other problems come from the insufficient resolution that might be set on the camera.

Apart from these main categories of image text data there exist born-digital images (e. g. web advertisements or any cases where text was digitally added on images or videos) and poster/newspaper images. These share with scene images the diversity of text but are free from visual distortions caused by cameras. Examples of different image data are in the chapter 4.

Digital reading of a text on an image consists of two main tasks – text detection and text recognition. Both processes are described in the next sections.

1.1 Text Detection

The first phase of digital text processing is to detect the text regions on the image. The goal is to determine a bounding box surrounding a group of letters – usually one word, but it can be also a text line consisting of few words. This box may be a bounding rectangle or a polygon which is more accurate for curved or skewed text. It is ideal when the bounding box contains the letters with as little background as possible. The methods store this information as a set of coordinates which unambiguously define the box, some methods produce cropped images based on the coordinates or a binary image with highlighted detected area.Methods used for text detection can be categorizes into formerly used classical machine learning methods and deep learning methods.

Classical Methods

Classical methods include connected component methods and sliding window methods. In the latter method an image is scanned with a moving window of certain size. In each position of the window features are computed, for example, standard deviation or histogram of oriented gradients and are compared with values known from training images with text. This approach does not have a good response for scene images, because many objects can be misunderstood as letters and vice versa. Connected component based methods extract from the image features like color, texture, edges or corners. Then they are classified either as text or non-text by a traditional classifier such as support vector machines, nearest neighbor or random forest. Detected letters are then combine into words or text lines if desired [21]. Classical methods are generally not very successful with scene images where most of the observed values are negatively influenced by distorting factors listed above. In recent years the development of neural networks enabled a new, efficient way of detecting text in images.

Deep Learning Methods

Deep learning methods are faster, more precise than the classical methods. They are automated and need less of human assistance therefore they can process bigger amounts of data. Another advantage is that algorithms can be generalized and used for other image detection tasks. Most of the state-of-the-art methods utilize convolutional neural network (CNN). Deep learning methods can be split into bounding-box regression based, segmentation based and hybrid methods according to the survey of Raisi et al. [21].

Bounding-box regression based methods treat text as an object and predict directly the bounding box around text. However, the bounding boxes have different aspect ratio than typical objects, because text is usually long and thin. Some methods decompose the text on smaller units and concentrate on distances between the units. These methods are hard to tune during training an usually fail on distinctly curved

text. Examples of bounding-box methods are TextBoxes [15], TextBoxes++ [14] or EAST [29]. Segmentation based methods investigate the image at pixel level. The image data are processed by a CNN which produces a segmentation map from which a bounding box is generated. An example of this method is PSENet [27]. Hybrid methods combine the features obtained by regression and the segmentation map from CNN. The segmentation methods tend to return false positive detection. They select pixels that are not text at the borders of letters or when there is a complicated background. Hybrid methods further process the results from segmentation and precisely detect text. Example methods are PMTD [16].

A popular method in recent years is Character Region Awareness for Text Detection – CRAFT. This method in contrast to the previously mentioned methods detects text based on character level rather than word (group of characters) level. CRAFT trains a CNN which produces two results for a character – a region score and an affinity score. "The region score represents the probability that the given pixel is the center of the character and the affinity score represents the center probability of the space between adjacent characters" [4, page 3]. Because this method goes over individual characters it performs very well on curved and deformed text and outperforms 15 popular detection methods according to the authors of CRAFT in their paper [4].

1.2 Text Recognition

Text recognition converts a detected text on an image to a string. Analogous to detection methods again recognition methods can be divided to classical methods and deep learning methods. Traditional methods work with image features such as histogram of gradients, features from SIFT[1], which are then classified by SVM or nearest neighbor algorithms. Moreover methods can be divided on segmentation-based and segmentation free methods. The former classify single characters gradually and then connect them into a word. The latter examine the text line as a whole, so it can use word neighborhood for contextual information. In the following subsection four stages of segmentation free methods are described.[5, 21]

Segmentation Free Methods

Segmentation free methods follow a pipeline with four stages: image preprocessing, feature representation, sequence modeling and prediction.

1. **Image Preprocessing Stage**: during this phase visual quality of the image is improved as much as possible. Because in scene images background is usually not a plain color, but a complex mixture of colors in various shape, the effort is to replace it and create optimally a binary image with one color for the foreground text and another for the background. It can be achieved by using

[1] Scale-Invariant Feature Transform

neural networks. If complete background removal is not easily achievable, basic image preprocessing such as noise removal are performed at least. Optional enhancement for scene images is to increase readability via superresolution, this removes noise caused by low resolution of the original image. If the text is perspectively distorted or curved there is an effort to straighten the text, this process is called rectification. It is a computationally expensive process, therefore it is not used very often.[5]

2. **Feature Representation Stage**: now it is necessary to extract features from the image data that are used as a representation of objects (text) in the images. For this purpose CNNs are widely used. For example namely these types VGGNet, ResNet, DenseNet, recurrent CNN (RCNN).

3. **Sequence Modeling Stage**: An optional step between final prediction and features. A contextual information is obtained via a recurrent neural network. Mostly one of two types called long short-term memory (LSTM) and bidirectional long short-term memory (BLSTM) is applied. The information is utilized during the last stage for predicting subsequent characters rather then predicting each character individually.

4. **Prediction**: in this last stage the model returns a target word or text line that was recognized. Two major technologies are connectionist temporal classification (CTC) and the attention mechanism. Because some letters span more than others it may happen that the letter is classified multiple times, to avoid this CTC introduces a blank symbol and inserts it among characters in various locations and creates character sequences. CTC then trains the network to maximize the probability distribution over all possible sequences. This approach is to a great extent dependent on a lexicon or language information about word formation. Attention-based methods utilizes RNNs [13]. "The attention mechanism learns the alignment between the input instance image and the output text sequences by referring to the history of the target characters and the encoded feature vectors" [5, page 12].

Classical Methods

Traditional methods work with image features such as histogram of gradients, features from SIFT[2], which are then classified by SVM or nearest neighbor algorithms. These methods

Deep Learning Methods

[2]Scale-Invariant Feature Transform

Chapter 2

Neural Networks

A perceptron unit is able to find a linear boundary between two separable classes. In n-dimensional space we talk about a separating hyperplane. The following equation

$$\boldsymbol{w}^\top \boldsymbol{x} + w_{n+1} = 0 \tag{2.1}$$

is a vector form of the hyperplane equation, where $\boldsymbol{w}$ is a weight, n-dimensional column vector, $\boldsymbol{x}$ is also n-dimensional vector and it contains the coordinates of a point, which is being classified, w_{n+1} is a bias. We can simplify the notation by creating a *n+1*-dimensional vectors: $\boldsymbol{x} = [x_1, \ldots, x_n, 1]^\top$ and $\boldsymbol{w} = [w_1, \ldots, w_n, w_{n+1}]^\top$. We want to find a weight coefficients that satisfies the following property:

$$\boldsymbol{w}^\top \boldsymbol{x} > 0 \qquad \boldsymbol{x} \in class_1 \tag{2.2}$$

$$\boldsymbol{w}^\top \boldsymbol{x} < 0 \qquad \boldsymbol{x} \in class_2 \tag{2.3}$$

Then the perceptron learning algortihm can be described as follows:

For any $\boldsymbol{x}(k)$, at step k:

1. If $\boldsymbol{x}(k) \in class_1$ and $\boldsymbol{w}^\top \boldsymbol{x} \leq 0$, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \alpha \boldsymbol{x}(k) \tag{2.4}$$

2. If $\boldsymbol{x}(k) \in class_2$ and $\boldsymbol{w}^\top \boldsymbol{x} \geq 0$, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \alpha \boldsymbol{x}(k) \tag{2.5}$$

3. Otherwise, let

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) \tag{2.6}$$

where $\boldsymbol{w}(1)$ is arbitrary and $\alpha > 0$ is a constant called learning rate. Sum of the products, $\sum_{k=1}^{n} w_k x + w_{n+1}$ is then passed through an activation function. In case of perceptron it is a threshold function returning either 1, when $\boldsymbol{x}$ belongs to $class_1$,

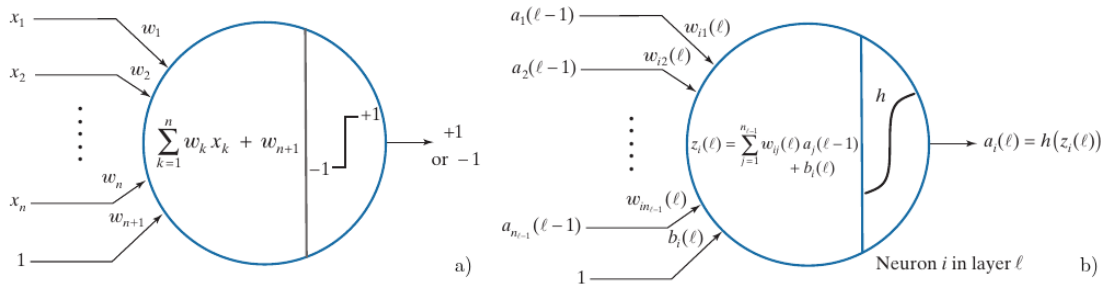


Figure 2.1: Model of a perceptron unit (a), model of an artificial neuron (b) with used operations. The letter h denotes an activation function, l denotes a particular layer in a multilayer network. The element on the right is sometimes called a sigmoid neuron.[8]

or -1 when it belongs to the second class $class_2$. In Figure 2.1.a. is shown a model of a perceptron.[21]

Linearly separable data are rather rare in real life problems. One possibilities is to use more perceptron units, however, the solution comes with neural networks and computing elements called artificial neurons. These elements are similar to perceptrons as they perform the same computations, but have a different attitude to processing the results. Schematics of a perceptron unit and an artificial neuron can be comapred in figure 2.1. The perceptron activation function is very insensitive to small signals which can lead to false results. If the activation function is changed from a hard threshold to smooth function, results are then handled more carefully. There are few commonly used smooth activation functions, such as sigmoid, hyperbolic tangent or ReLu (rectifier linear unit) function. In Figure are the equations and shapes of the mentioned functions.[8]

A generic neural network is depicted in Figure 2.2. By layer we understand a group of neurons, usually symbolized in a column. First layer contains the input vector $\boldsymbol{x}$, then every other layer contains the activation values of neurons in this layer. The conntecting lines between each two neurons signifies a fully connected neural network, where output of every neuron from one layer is used as input for neurons in the following layer. Values of initial layer are known and also are the last output values, all neurons (and layers) between first and last layer are therefore called hidden. When there are more then one hidden layer we talk about deep neural network. Usually the number of output neurons is equal to the number of observed classes. For the rest of this chapter we will assume that there are no loops in the network.[8]

A forward pass through neural network maps the values of vector $\boldsymbol{x}$ (the input layer) to the output layer, thus to determine the class of the vector $\boldsymbol{x}$. Steps of a forward pass can be written in matrix notation. This aproach is good for computing simultaneously with multiple input vectors and all neurons in one layer. The forward pass can be described in three steps:

1. Input

$$\boldsymbol{A}(1) = \boldsymbol{X} \tag{2.7}$$

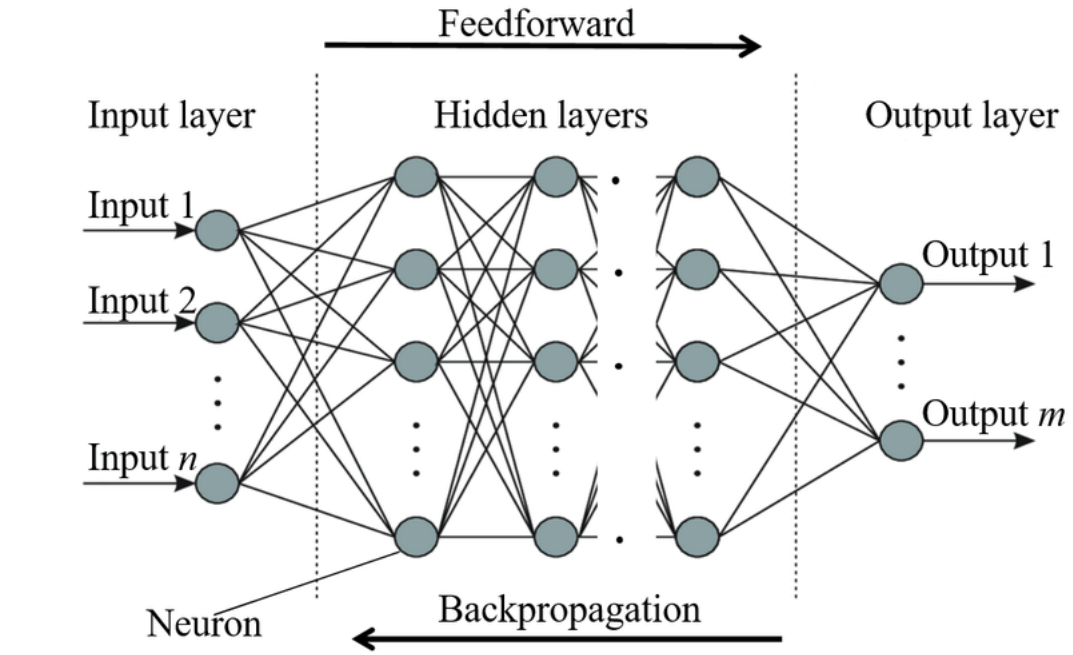


Figure 2.2: Fully connected neural network with processes.[1]

2. Feedforward step

$$\text{For } l = 2, \ldots, L \qquad \boldsymbol{Z}(l) = \boldsymbol{W}(l)\boldsymbol{A}(l-1) + \boldsymbol{B}(l),\ \boldsymbol{A}(l) = h(\boldsymbol{Z}(l)) \tag{2.8}$$

3. Output

$$\boldsymbol{A}(L) = h(\boldsymbol{Z}(L)) \tag{2.9}$$

Matrix $\boldsymbol{W}(l)$ contains all weight vectors of all nodes in one layer l, $\boldsymbol{X}$ contains input vectors, $\boldsymbol{A}(l)$ contains output values from layer l, $\boldsymbol{B}$ is the matrix of biases, $\boldsymbol{Z}(l)$ contains the net inputs to neurons in layer l, h is an activation function. This process of predicting works when we know the right values of weights and biases. These can be obtained by training a neural network via backpropagation.[8]

During training of neural network we work with data where it is known for each sample to which class it belongs. This means that we know the values of all otuput neurons in the net. However, we do not know the values of outputs of hidden neurons. A process called backpropagation is used to obtain information about hidden neurons. It can be devided into four steps. These steps are repeated until the value of cost function is reduced to a desired level. One repetition is called an epoch, thus the number of iterations is the number of epochs used for training the network. [8]

1. Input of data from training set.

$$\boldsymbol{A}(1) = \boldsymbol{X} \tag{2.10}$$

2. A forward pass to classify the data into class and determine the error of misidentified classes (sometimes this error function is called cost function) based on ground thruth from the training data.

$$\begin{aligned} \text{For } l = 2, \ldots, L \qquad & \boldsymbol{Z}(l) = \boldsymbol{W}(l)\boldsymbol{A}(l-1) + \boldsymbol{B}(l), \\ & \boldsymbol{A}(l) = h(\boldsymbol{Z}(l)), \\ & \boldsymbol{D}(L) = (\boldsymbol{A}(L) - \boldsymbol{R}) \odot h'(\boldsymbol{Z}(L)) \end{aligned} \tag{2.11}$$

3. A backward pass that sends the output error back through the network, where changes to update neuron paramters are computed.

$$\text{For } l = L-1, L-2, \ldots, 2 \qquad \boldsymbol{D}(l) = (\boldsymbol{W}^\top(l+1)\boldsymbol{D}(l+1) \odot h'(\boldsymbol{Z}(l)) \tag{2.12}$$

4. An update of weights and biases of neurons.

$$\begin{aligned} \text{For } l = 2, \ldots, L \qquad & \boldsymbol{W}(l) = \boldsymbol{W}(l) - \alpha \boldsymbol{D}(l)\boldsymbol{A}^t op(l-1), \\ & \boldsymbol{b}(l) = \boldsymbol{b}(l) - \alpha \sum_{k=1}^{n_p} \delta_k(l), \\ & \boldsymbol{D}(L) = (\boldsymbol{A}(L) - \boldsymbol{R}) \odot h'(\boldsymbol{Z}(L)), \\ & \boldsymbol{B}(l) \text{ consist of horizontally stacked vectors } \boldsymbol{b}(l) \\ & \text{for } n_p \text{ times}, \\ & \boldsymbol{\delta}_k(l) \text{ are the columns of matrix } \boldsymbol{D}(l), \end{aligned} \tag{2.13}$$

where $\boldsymbol{W}(l)$ is the matrix of weights of all nodes in one layer l for the inputs from $\boldsymbol{X}$, which contains multiple input vectors, $\boldsymbol{A}(l)$ contains output activation values from layer l, $\boldsymbol{B}$ are the biases, $\boldsymbol{Z}(l)$ contains the net inputs to neurons in layer l, $\boldsymbol{\delta}(l)$ tells us the rate of error change with respect to a change in the net input to any neuron in the network $\delta_k(l) =$, α is the learning rate used in training, h is an activation function. $\boldsymbol{W}(1)$ and $\boldsymbol{B}(1)$ are set as random small numbers when initializing the process.[8]

The error function for all output neurons for a single input $\boldsymbol{x}$ is defined as

$$E = \sum_{j=1}^{n_L} E_j = \frac{1}{2} \sum_{j=1}^{n_L} (r_j - a_j(L)) = \frac{1}{2} ||\boldsymbol{r} - \boldsymbol{a}(L)||^2 \tag{2.14}$$

where $\boldsymbol{r}$ is a desired response for a given input $\boldsymbol{x}$, $\boldsymbol{a}(L)$ is the output of last layer in the network, in the last term is used the notation of the Euclidan vector norm. The error for all input vectors[1] is equal to the sum of the individual errors. [8]

2.1 Convolutional Neural Networks

The procedures described in the previous part dealt only with the case where the input data are in the form of a vector. In optical character recognition we work

[1] a total network output error

with image data that are not primarily represented as vector but as a matrix of pixel values. The matrix can be linearized from 2D to 1D when indeces are mapped gradually. However, this approach does not consider spatial relationships that may be present among specific pixels. For examples edge or color similarities which are significant in text detection. Convolutional neural network (CNN) accepts 2D matrix as input and extracts features from the given image, these features are then fed to a classic fully connected neural network. A diagram describing a simple CNN is in Figure 2.3. We will discuss individual steps visible in this figure below.[8]

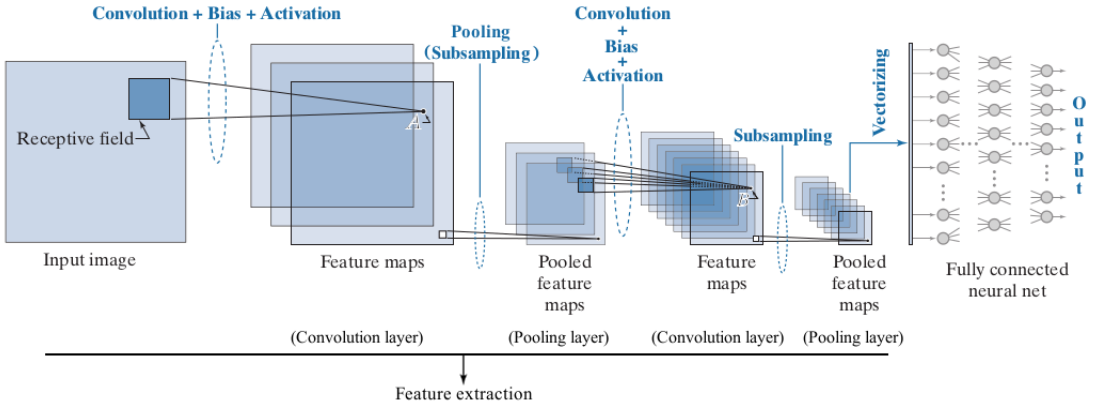


Figure 2.3: A simple CNN with LeNet architecture.[8, altered]

First, a region of of pixels of the input image, a receptive field, is selected. This field is moved over the image with a certain step. At each position a convolution is performed and values are stored in a 2D matrix. The size of the step determines the size of the resulting matrix (for example step of size two reduces the resolution of the image by one half). To each value obtained by convolution a bias is added and then is passed through an activation function. The new matrix thus obtained is called feature map. A single feature map is generated with same weights in convolution kernel and same bias, because this way a same feature is detected through the image. For another feature map different weight and bias is used. A group of feature maps in a CNN are called a convolutional layer. After the convolution step pooling is performed (sometimes called subsampling).[8]

Definition 1 *Let $f(x, y)$ be in image and w a kernel of size $m \times n$. Convolution with kernel w is denoted as $(w \star f)$ and defined as*

$$(w \star f)(x, y) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} w(u, v) f(x - u, y - v). \tag{2.15}$$

Pooling is responsible for reduction of output dimension and causes a translational invariance. It is done by dividing a feature map into 2×2 adjacent (non-overlapping) regions, the four values in this region are replaced by only one value. Common pooling methods are: average pooling, the values in region are replaced by the average of the values; max pooling, the values are replaced by the maximal value from the region; L_2 pooling, the final value is obtained by computing the Euclidan

norm of the four values. We obtain one matrix for each feature map from the convolution step. The bunch of matrices from the pooling step is called a pooling layer. The result of the last pooling layer is vectorized and sent to the fully connected neural network. The training procedure of the CNN is analogical to the training of a fully connected neural network. However, convolution is used instead of matrix multiplication and the output from fully connected network has to be converted into 2D matrix.[8]

Outline

- What is OCR
- Datasets (types(synthetic, photos, scaned documents),problems(languages, noise, nonhorizontal text))
- Text detection
  - Description
  - Methods (CRAFT)
- Text recognition
  - Description
  - Methods
- End-to-end systems (Annotating tool)

  Reading scanned documents
  - EasyOCR
  - keras-ocr
  - Tesseract (PyTesseract)
  - (Google Cloud Vision free) paid
  - (AWS Recognition) paid
  - (Kili) paid
- Results evaluation
  - Comparison of output and ground-truth
  - Bag of words
- Testing methods on free datasets

  Description of datasets
- Using methods on historical posters

  Description of dataset

Chapter 3

Software

3.1 Scene text detection

Methods

3.1.1 CRAFT

Character Region Awareness for Text Detection (CRAFT) is framework for scene text detection. It uses a Convolutional Neural Network. It performs good also on curved or differently deformed texts. Its methodology is to localize individual characters then characters belonging to the same word (based on distance) can be connected into word box or polygon. After that bounding box is created around it and output contains the rectangle coordinates.[4]

3.2 Scene text recognition

Methods

3.2.1 CRNN

Convolutional Recurrent Neural Network

3.3 End-to-end systems

End-to-end system when given an image with text it can both detect and recognize the text and produces a string output of the text. The supply of such tools is wide, ranging from open source libraries for various programming languages to commercial softwares with modern GUI. New methods are still being developed as there is always

space for improvement. New methods can come from commercial background or are developed for international OCR competitions. In the next section some of the free available tools are described.

3.3.1 Tesseract

Tesseract is an open source text recognition engine. It supports over 160 languages. Originally Tesseract was created by Hewlett-Packard in late 1980s, from 2006 it is developed and maintained by Google. As it does not have a built-in GUI direct use is via command line. However, there exist a significant number of GUIs for Linux, Windows, Mac for computer usage and also for Android and iOS to use on mobile phones and few online OCR services. Another way how to use the engine is via libraries for computer languages, namely for example they exist for Java called tess4j, Python called pytesseract, R, Ruby and others. [25]

Tesseract is mainly used as tool for recognizing documents (with both computer font text or hadwritten text). Best results are obtained on preprocessed images. The preprocessing includes noise reduction, horizontal alignment of text, elimination of dark borders around text region, conversion to binary black and white picture and other adjustments depending on the nature of the picture. Thus when used on scene text images it gives generally worse results than other OCR softwares.

Computarions with Tesseract are supported for GPU and also CPU. Tesseract uses for recognition Long Short Term Memory (LSTM) model (kind of RNN).

Page Segmentation models By default Tesseract expects a page of text – black letters on white background grouped in horizontal lines, where font type and font size vary only slightly. To deal with differently distributed text over an image Tesseract provides thirteen page segmentation models (PSM). When selecting the right model Tesseract performance can increase from zero up to almost perfect results. Description of all the PSMs can be find directly via Tesseract help command in console application. https://pyimagesearch.com/2021/11/15/tesseract-page-segmentation-modes-psms-explained-how-to-improve-your-ocr-accuracy/

3.3.2 EasyOCR

EasyOCR is a product of Jaded AI for both image text detection and recognition. it supports over 80 languages and various scripts such as Latin, Chinese, Arabic etc. The company offers software with web interface for free and also prepaid version which enables usage of a new model for custom data. However, in addition to the web interface, the company also created a python package under the same name.[2]

The product is still in development and aims for wider functionality. A future idea of EasyOCR package is to provide an easy-to-use tool where one can plug-in already created state-of-art models and use them for annotating. Pipeline of EasyOCR behavior is shown in the image 3.1. As it can be seen in this image, default detection model is CRAFT and for recognition is used CRNN (Convolutional Recurrent

Neural Network) which model is composed of following components: feature extraction (Resnet is used) and VGG (Convolutional Neural Network), sequence labeling (LSTM is used) and decoding (CTC is used).[3]

EasyOCR package by default computes annotation on GPU, however there is a possibility for CPU computations (provided that the selected model supports it).

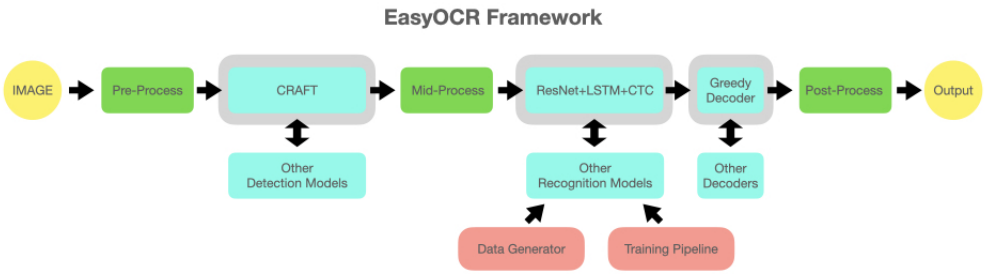


Figure 3.1: Diagram of EasyOCR pipeline. Grey slots are placeholders for models. The mentioned models are the ones used as default. [3]

3.3.3 Keras-ocr

keras-ocr is a python library used for detecting and recognizing text in images created by Fausto Morales. It works with variety of languages and with different writing scripts. It allows computing on CPU as well as on GPU. It unites the CRAFT text detection model and an implementation in Keras python library of CRNN for recognizing text, worth mentioning this is a different implementation of CRNN than in EasyOCR.[12]

On the official website[1] of the package there is a comparison of this method with two other OCR APIs – Google Cloud Vision and AWS Rekognition. Their performance was tested on 1,000 images from the COCO-Text validation set using a basic pretrained model of each method. None of the investigated methods performed poorly; however, AWS Rekognition had the worst precision and recall results. Google's method and keras-ocr has similar results. It is important to mention that no tuning parameters were used in any of these methods. Another candidate for comparison was Tesseract but it performed on very badly on given data, most likely due to the fact that Tesseract is suitable for scanned documents rather than for photos of real life scenery and objects with text. [12]

CRAFT already provides a pretrained model which can be used directly without modification for text detection or it is used as initial model for training a new model on new data. This model was trained on three datasets (SynthText, IC13, IC17) and supports English and multi language text detection.[22] Similarly for recognition, CRNN also has a pretrained model. This model was trained on the synthetic word dataset which consists of 9 million images with vocabulary of 90K English words.[**?**] To use these models in the keras-ocr library one either doesn't specify anything and

[1] `https://pypi.org/project/keras-ocr/`

use the defaults, or pass the value `clovaai-general` for the CRAFT pretrained model or `kurapan` for the CRNN model.

Keras-ocr offers preprocessing for four public datasets though any text image dataset can be examined using this tool. These four datasets are: BornDigital dataset, COCO-Text dataset, ICDAR 2013 dataset, ICDAR 2019 dataset (only Latin-only scripts).[11]

3.3.4 Google Cloud Vision

Google Cloud Vision is software from Google which consist of two products: AutoML Vision and Vision API. Vision API detects objects, faces and text from images with already pretrained model. With AutoML Vision user can train custom model from own data. It has free and paid version with a GUI.[26] API for python

3.3.5 Other

AWS Rekognition

Chapter 4

Datasets

Optical Character Recognition requires data as any other machine learning task. Data are usually divided in two main types - scene and synthetic. Scene datasets contains photographs of real world objects and sceneries where some text occurres, for example shop signs, road signs or car plates. Synthetic dataset are automatically generated images where words are chosen from a extensive dictionary, a font is picked for each word and some sort of deformation is applied. It can be a text distortion to make the text curved or projectively altered, as well as blurring or lighting changes that make the text less obvious for the detector.

Synthetic datasets are usually used for training the models, because it is easier to generate millions of synthetic images rather than to take even one hundredth of a such number of photographs. Needless to say that when generating an image, the ground truth is known and can be saved during the generation process while photographs have to be manually labeled which takes time and might be inaccurate or automatically labeled which also often leads to many mistakes. Scene datasets are then used for testing purposes or for fine tuning a pretrained model.

In the following section sample datasets are introduced. To begin with synthetic dataset MJSynth is a very important dataset because it consists of almost 9 million images covering 90,000 English words. It includes data only for recognition which means that one image has only one word and border of the image represents the word bounding rectangle.[18]. Another synthetic dataset is called SynthText and contains 800 thousand images with approximately 8 million word instances written on the images [24]. Three scene text datasets were created for International Conference on Document Analysis and Recognition (ICDAR) competition. Sets ICDAR03, ICDAR13 and ICDAR15 were used in competitions in years 2003, 2013 and 2015, respectively. First two ICDAR datasets include only horizontal text, text of various orientation appears in set from 2015 [21]. Another widely used dataset is The Street View Text (SVT) which contains images with text harvested from Google Street View [23]. SVT and most other scene text datasets offers mainly frontal text with minimal perspective distorision. However, perspective text is frequent in real life applications of OCR for example previously mentioned street photographs, where it is impossible to capture every visible text from frontal view. Thus Phan et. al [20] created a new StreetViewText-Perspective derived from SVT, it shows the same

places as SVT but from different perspective. Another dataset CUT80 focuses on curved text as well as CTW1500 dataset. For text recognition there exist for example IIIT5k dataset containing 5000 cropped images harvested from Google image search. It combines both scene text images and born-digital images [9]. One of the most widely used dataset is COCO-Text which includes over 60,000 images with almost 250,000 word instances [6].

Datasets that were used for comparission of detection and recognition methods, namely, SCUT-CTW1500 dataset, Kaist Scene Text Database, Born-Digital Images, are described in more detail below.

4.1 SCUT-CTW1500 dataset

SCUT-CTW1500 dataset contains exactly 1500 images of real-world, scene text in English language. Sample images can be seen in Figure 4.1. The key feature of this dataset is that each image contains horizonally aligned text, multioriented text and curved text. There are cases where the curvature is only slight and cases where text forms a circle with letter upside down. Recognizing multi-oriented and curved text is more of a challenge than pure horizontal text. This dataset is split to train and test data. Two thirds of dataset thus one tousand images for training and five hundred for testing. According to the description of this dataset on relevant GitHub repository dataset was manually labeled and lately corrected, therefore labels seem to be very accurate. However for example ground truth for image 1313.jpg misses all occurences of letter I, as the depicted font was probably misread.[28, 17]

The ground truth for train data are in XML format and each file carries information about the file name of respective image file, text information – i.e., words in a text line, 14 coordinates of a bounding polygon and coordinates, height and width of a circumsribed rectangle. Later the authors added coordinates of center point of each English letter to be used as detection ground truth. The ground truth of test data is in simple text file (TXT) and contains only 14 coordinates of the bounding polygon and a text which is within that region. There is a minor issue with labels that it usually contains a full text line with multiple words and coordinates are not assigned to individual words but to text region as whole. Most end-to-end system detect words rather than groups of corresponding words. This fact needs to be taken into account when evaluating results.

4.2 KAIST Scene Text Database

This dataset contains 3000 images of photographed text. It can be divided into three major catgories – text of Korean language, English language and mixed languages. As I concentrate on text in latin script in this paper further information relates to the English language dataset. The number of images is then reduced to less than four hundred images. Figure 4.2 shows few samples of this dataset. Photographed objects are mostly shop banners or parts of magazine front pages. Photographs were

Figure 4.1: Sample images of SCUT-CTW1500 dataset.

either taken by a high-resolution digital camera or a low-resolution mobile phone camera.[10] Each photography has a ground truth description and a bitmap image. In the bitmap file only text is highlighted (by white or red color) and everything else apart from text is set as black. Ground truth files are in XML format and includes a name of an image, its resolution and bounding box for each word and also a bounding box for each letter of the word.

To use this dataset for testing and training the XML ground truth needed to be converted to string and int values. I wrote a parser, that combines letters to form a word that is within a given bounding box. I changed the notation of bounding boxes from one coordinate, width and height attributes to two top left and bottom right coordinates.

Unfortunately this dataset has few errors in filenames of corresponding files or in the content of XML files. Usually these are only typos, however they prevent automatic preprocessing of dataset. Due to this problem these mistakes need to be found and manually corrected. Also there is a small number of ground truth XML file with fully missing data. Despite these shortcomings this dataset is useful because of the bitmap files. This allows to compare results of both images affected by shooting conditions and images dependent only on font and position.

4.3 Born-Digital Images

Born-Digital Images contains data of images with text that can be found on various websites. Samples of this dataset can be found is in Figure 4.3. There are mostly advertisements, company logos or website headers. Such pictures cannot be classified neither as real scene dataset, neither as synthetic one. On one hand this dataset shares with scene datasets the variability in font styles and sizes, different text orientations and complex colour placement. On the other hand it differs in size because low resolution is significant in smooth and fast loading on websites. Also no noise is present due to lighting conditions. Geometrical deformations that result when capturing a real scene with camera also do not appear here. However com-

Figure 4.2: Sample images of KAIST Scene Text Database dataset.

pression to lower resolution can lead to artefacts and aliasing. In general we can say that letters are more clearly visible than in photographed text as easy readability is crucial in successful advertising.[7]

The dataset is available for download from the website of Robust Reading Competition. First version was published in 2011 and revised two years later, it contains separate dataset for text localization, segmentation and then for word recognition. In 2015 they published an end-to-end dataset with ground truth for all tasks. The dataset is split in training and testing data. However, ground truth for testing data contains only a possible vocabulary of words in images and no coordinates. This might be due to the fact that the competition might be still ongoing or there was not a sufficent demand for complete ground truth. As for training data, each image has a corresponding TXT file with coordinates of four vertices of bounding rectangle and a word. Text lines are separated and the text within rectangle is always one word. Unfortunately, there are quite a few missing words, usually words that have two or less characters. This can affect the evaluation when the model finds such a short, missing word.[7]

Figure 4.3: Sample images of Born-Digital Images dataset.

4.4 Wien TU dataset

Chapter 5

Testing methods

I wrote all testing scripts and auxiliary functions in Python programming language. The scripts were written as Jupyter Notebook environment and saved in the corresponding file format `.ypnb`, lately these notebooks were saved as `.py` Python scripts. The set of utility functions is located in a file called `utils.py`.

In the experiments I chose three methods to be tested, namely, Tesseract, EasyOCR and keras-ocr. Each of them has corresponding Python package

5.1 EasyOCR

5.2 Tesseract

Potreba cernobile a threshold tesseract to ma radsi, problem delat centralne - otsu a po castech ale stejne tezke u spousty obrazku s ruznym osvetlenim. Tesseract zkousime u bitmap obr PSM 11 na Kaist a je o hodne horsi nez PSM 6, s 11 je to 9.6. proc. jinak u normlalnich obrazku je lepsi zase 11, taky treba o 9 procent rozdil.. pak jeste 4 to je obcas lepsi U CTW datasetu je to s 6 horsi.

jeste hrani s barvama nebo upraveny , jak kde

verze tesseract 4.0.0-beta.1 leptonica-1.75.3 libgif 5.1.4 : libjpeg 8d (libjpeg-turbo 1.5.2) : libpng 1.6.34 : libtiff 4.0.9 : zlib 1.2.11 : libwebp 0.6.1 : libopenjp2 2.3.0

Found AVX2 Found AVX Found SSE

5.3 Keras

Training took one hour and stopped after 57 epochs ,10.979148864746094,20.5778751373291. Continued training.

Bibliography

[1] D. Abueidda, S. Koric, R. Abu Al-Rub, C. Parrott, K. James, and N. Sobh. **A deep learning energy method for hyperelasticity and viscoelasticity**. *A deep learning energy method for hyperelasticity and viscoelasticity*, 01 2022.

[2] Jaided AI. **Jaided AI - Distribute the benefits of AI to the world**. Last accessed 20 May. 2022. [Online]. Available from: `https://www.jaided.ai/`.

[3] Jaided AI. **Jaidedai/EasyOCR**. Last accessed 13 Jun. 2022. [Online]. Available from: `https://github.com/JaidedAI/EasyOCR`.

[4] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. **Character region awareness for text detection**. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9365–9374, 2019.

[5] X. Chen, L. Jin, Y. Zhu, C. Luo, and T. Wang. **Text recognition in the wild: A survey**. *ACM Computing Surveys (CSUR)*, **54**(2):1–35, 2021.

[6] **COCO-Text V2.0**. Last accessed 22 Jun. 2022. [Online]. Available from: `https://bgshih.github.io/cocotext/`.

[7] Robust Reading Competition. **Overview - born-digital images (web and email)**. Last accessed 21 Jun. 2022. [Online]. Available from: `https://rrc.cvc.uab.es/?ch=1`.

[8] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (4th ed.)*. Pearson, 2018. ISBN 9353062985.

[9] **The IIIT 5K-word — Graviti**. Last accessed 22 Jun. 2022. [Online]. Available from: `https://gas.graviti.com/dataset/graviti/IIIT5KWord`.

[10] **Kaist scene text database**. Last accessed 10 Jun. 2022. [Online]. Available from: `http://www.iapr-tc11.org/mediawiki/index.php/KAIST_Scene_Text_Database`.

[11] **keras-ocr documentation**. Last accessed 29 Apr. 2022. [Online]. Available from: `https://keras-ocr.readthedocs.io/en/latest/index.html`.

[12] **keras-OCR**. Last accessed 29 Apr. 2022. [Online]. Available from: `https://pypi.org/project/keras-ocr/`.

[13] S. Kim, T. Hori, and S. Watanabe. **Joint CTC-Attention based End-to-End Speech Recognition using Multi-task Learning**. 09 2016.

[14] M. Liao, B. Shi, and X. Bai. **Textboxes++: A single-shot oriented scene text detector**. *IEEE transactions on image processing*, **27**(8):3676–3690, 2018.

[15] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. **Textboxes: A fast text detector with a single deep neural network**. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[16] Jingchao Liu, Xuebo Liu, Jie Sheng, Ding Liang, Xin Li, and Qingjie Liu. **Pyramid mask text detector**. *arXiv preprint arXiv:1903.11800*, 2019.

[17] Y. Liu, L. Jin, S. Zhang, C. Luo, and S. Zhang. **Curved scene text detection via transverse and longitudinal sequence connection**. *Pattern Recognition*, **90**:337–345, 2019.

[18] **Text recognition data - Visual Geometry Group - University of Oxford**. Last accessed 21 Jul. 2022. [Online]. Available from: `https://www.robots.ox.ac.uk/~vgg/data/text/#sec-synth`.

[19] **Technology — The technology for converting books and documents into electronic files.** Last accessed 27 Jul. 2022. [Online]. Available from: `https://www.abbyy.co.il/?categoryId=72050&itemId=168963`.

[20] T. Phan, P. Shivakumara, S. Tian, and Ch. L. Tan. **Recognizing Text with Perspective Distortion in Natural Scenes**. pages 569–576, 12 2013.

[21] Z. Raisi, M. A Naiel, P. Fieguth, S. Wardell, and J. Zelek. **Text detection and recognition in the wild: A review**. *arXiv preprint arXiv:2006.04305*, 2020.

[22] Clova AI Research. **Clovaai/Craft-pytorch: Official implementation of character region awareness for text detection (CRAFT)**. Last accessed 29 Apr. 2022. [Online]. Available from: `https://github.com/clovaai/CRAFT-pytorch`.

[23] **The street view text dataset**. Last accessed 22 Jul. 2022. [Online]. Available from: `http://vision.ucsd.edu/~kai/svt/`.

[24] **Synthtext in the wild dataset - Visual Geometry Group - University of Oxford**. Last accessed 21 Jul. 2022. [Online]. Available from: `https://www.robots.ox.ac.uk/~vgg/data/scenetext/`.

[25] Tesseract-Ocr. **Tesseract-OCR/tessdoc: Tesseract documentation**. Last accessed 6 May. 2022. [Online]. Available from: `https://github.com/tesseract-ocr/tessdoc`.

[26] **Vision AI — derive image insights via ML — cloud vision API google cloud**. Available from: `https://cloud.google.com/vision/.[Online]`.

[27] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao. **Shape robust text detection with progressive scale expansion network**. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9336–9345, 2019.

[28] Yuliang-Liu. **Yuliang-Liu/curve-text-detector**. Last accessed 13 Jun. 2022. [Online]. Available from: `https://github.com/Yuliang-Liu/Curve-Text-Detector`.

[29] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. 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.