FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Historical Document Analysis

Ph.D. Study Report

Josef Baloun

Technical Report No. DCSE/TR-2022-02
June 2022
Distribution: public

Technical Report No. DCSE/TR-2022-02
June 2022

Historical Document Analysis

Ph.D. Study Report

Josef Baloun

Abstract

Scanned documents are a rich source of various information that can be processed utilizing a document analysis system. Such a system covers the areas of Machine Learning, Computer Vision and Natural Language Processing. In the thesis, these areas are covered with a focus on common and state-of-the-art approaches applicable to historical document analysis, which is still challenging due to several difficulties such as handwritten text. Finally, the current research results and aims of the future doctoral thesis are presented.

Contents

Abbreviations

BERT	Bidirectional Encoder Representation from Transformer. 33–36
BiLSTM	Bidirectional Long Short-Term Memory. 29
CER	Character Error Rate. 49
CNN	Convolutional Neural Network. 8, 9, 24, 28
CTC	Connectionist Temporal Classification. 12, 13, 21, 28–30
CV	Computer Vision. 2, 43, 51
DIA	Document Image Analysis. 1, 2, 15, 16
EDT	Euclidean Distance Transform. 19
ELMo	Embeddings from Language Models. 33
FCN	Fully Convolutional Network. 9, 10, 18, 20, 21, 39, 40, 43, 44, 47
FgPA	Foreground Pixel Accuracy. 20, 21, 42
GPT	Generative Pre-trained Transformer. 33
GT	Ground-Truth. 4, 5, 12, 21, 24, 34, 49
HTR	Handwritten Text Recognition. 2, 12, 15, 26, 28–30, 36, 48–50
IoU	Intersection over Union. 42
LSTM	Long Short-Term Memory. 11
ML	Machine Learning. 2, 3, 16, 28, 43, 50
MLP	Multilayer Perceptron. 3, 6
NLP	Natural Language Processing. 2, 3, 13, 31, 50
NN	Neural Network. 1–6, 48

OCR Optical Character Recognition. 2, 15, 26, 28–30, 36, 47

PCA Principal Component Analysis. 32, 37

PHOC Pyramidal Histogram Of Characters. 28, 29

PPG Perpendicular Peak Group. 45, 46

QbE Query by Example. 28, 29

QbS Query by String. 28, 29

R-CNN Region-based Convolutional Neural Network. 24–26, 36, 47

ReLU Rectified Linear Unit. 4, 8

RNN Recurrent Neural Network. 11, 14, 21, 33

RoI Region of Interest. 22, 25, 26

RPN Region Proposal Network. 25

SGD Stochastic Gradient Descent. 5

SPP Spatial Pyramid Pooling. 24, 25, 28

YOLO You Only Look Once. 26, 47

Chapter 1

Introduction

Considerable efforts are currently being made to digitize documents and make them available electronically in most areas, such as business or archives. The aim is usually to make searching in these documents easier, reduce storage costs or make documents available to the general public. They are a rich source of information not only for historians, since they record various events including weather conditions for example. The volume of scanned historical documents in the archives around the world is still increasing and at the same time there is a growing need to process them into a more searchable form allowing information retrieval or data mining.

The input of document analysis are usually scanned pages. The output can be a set of detected key points, transcribed text or page representation. Compared to modern printed documents, the historical ones are more challenging due to handwriting, different authors, scanning quality and also paper degradation. Moreover, methods that work for modern documents usually do not work well for historical ones. Thus, there is a need for new approaches. These new approaches benefit from machine learning and are usually based on deep Neural Networks (NNs).

The main part of historical document analysis is Document Image Analysis (DIA). It can be divided into textual and graphical processing which deals with textual and graphical regions, respectively. [42] An important part is layout analysis, where the goal is to describe the layout of the page and to obtain the areas like text blocks, text lines, characters or images for example.

Once the text and graphic components are obtained, they can be further utilized. As depicted in Fig. 1.1, the conventional document processing workflow focuses

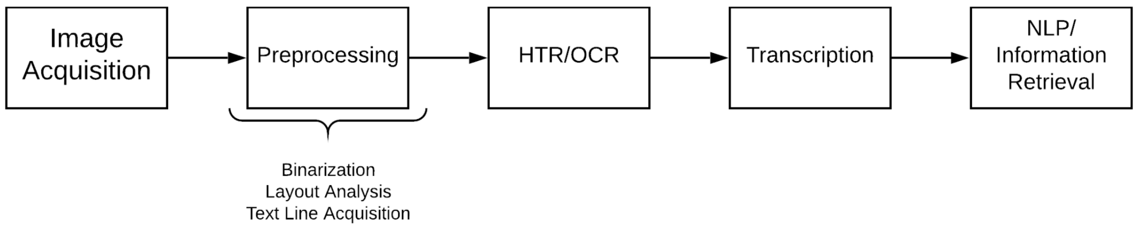


Figure 1.1: Historical document analysis conventional workflow [46]

mainly on the text information. The crucial step for textual components is Optical Character Recognition (OCR) or more challenging Handwritten Text Recognition (HTR). It allows transcription into a full-text representation.

Nowadays, the need for information present in graphic areas is also growing. Obviously, this information may supplement the information present in the text but it could be also the main source of information. Generally, the historical document analysis can include tasks like object detection, full-text transcription or image captioning. Therefore, it covers the areas of Machine Learning (ML), Computer Vision (CV) and Natural Language Processing (NLP) and is thus a hybrid field. [46] For example, NLP methods can analyze the text itself and provide more advanced functionality such as named entity recognition or multilingual searching. The graphic components can be processed by CV classification methods for example. Further, graphic and text components can be utilized in multi-modal approaches that seems promising and have not yet been sufficiently explored.

The goals of this work are to introduce the topic of historical document analysis, summarize state-of-the-art approaches in the area, present the research results and aims of the doctoral thesis.

This work starts with an introduction to NN and frequently used approaches in the area of document analysis in order to simplify the understanding of methods presented further. The next chapter is related to DIA including layout analysis, textual and graphical processing. It is followed by chapters related to NLP and multi-modal processing. After that, own work is presented. Finally, a summary and the aims of the doctoral thesis are given.

Chapter 2

Neural Networks

NNs are currently the mainstream in the area of ML and state of the art in many tasks including general classification, image segmentation, object detection, NLP and also historical document analysis. They are able to extract features directly from the input, so no preprocessing is needed. This is a great advantage not only in historical document analysis. In this chapter, I will firstly introduce the basic model and training. Further, I will highlight the main and frequently used approaches in the area.

2.1 Multilayer Perceptron

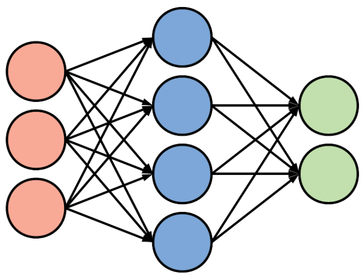
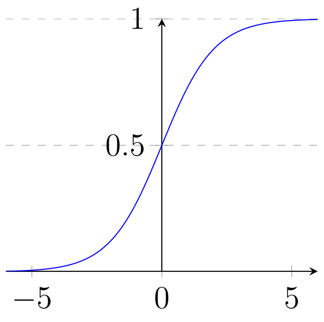


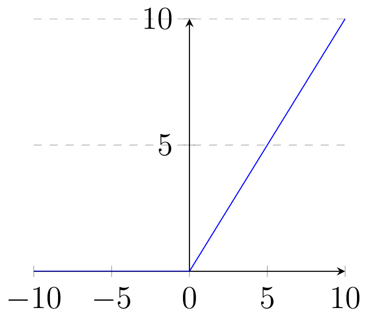
Figure 2.1: MLP visualization: input (red), hidden fully-connected layer (blue), output fully-connected layer (green)

Multilayer Perceptron (MLP) is the basic model of NN. As can be seen in Fig. 2.1, it consists of fully-connected layers, where each neuron is connected with all neurons from the previous layer. The input of the neuron is generally a tensor, which can be the input vector or the outputs of the neurons from previous layer.

In this type of network, the neuron output z for the input x of length M is computed according to Eq. 2.1, where w are the weights of the neuron and φ stands for the activation function (see Fig. 2.2). Bias w_0 is a special weight that allows the



(a) Sigmoid



(b) ReLU

Figure 2.2: Basic activation functions

x-axis shift of the activation function and thus $x_0 = 1$.

$$z = \varphi(\sum_{i=0}^{M} w_i x_i) \tag{2.1}$$

Another important part is the loss function (also denoted as cost or error function). It is crucial for the training of NNs, since it computes the error of the prediction (p of length K) compared to the Ground-Truth (expected output y of length K). Generally, the combination of the loss function and activation function in the last output layer depends on the task. [6]

Linear activation function (Eq. 2.2) and mean squared error (Eq. 2.3) can be used for regression problems.

$$\varphi_l(x) = x \tag{2.2}$$

$$C_{MSE}(p, y) = \sum_{i=1}^{K} (p_i - y_i)^2 \tag{2.3}$$

Sigmoid activation function (Eq. 2.4) and a binary cross entropy loss function (Eq. 2.5) are used for binary classification or multiple binary classifications, e.g. multi-class multi-label classification.

$$\varphi_s(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

$$C_{BCE}(p, y) = -\sum_{i=1}^{K} y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i) \tag{2.5}$$

Softmax outputs (Eq. 2.6) and corresponding categorical cross-entropy loss function (Eq. 2.7) are used for multi-class single-label classification for example.

$$\varphi_{sm}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.6}$$

$$C_{CCE}(p, y) = -\sum_{i=1}^{K} y_i \cdot log(p_i) \tag{2.7}$$

Presented losses are element wise, thus it sums the error between elements of prediction p and Ground-Truth (GT) y. For multiple data samples (e.g. batches), average term is used in order to have the gradient independent of the number of samples as presented in Sec. 2.2.1.

2.2 Training

The goal of NN training is to properly set the weights in order to minimize the error on training data. It is usually done by backpropagation which consists of forward pass, error backpropagation and weights update. [6] The forward pass was described in the previous section. It is followed by error computation using the loss function and error backpropagation, which obtains the gradients of the error function with respect to the weights. Finally, the weight update is done based on these gradients.

2.2.1 Stochastic Gradient Descent

The straightforward approach for weight (w) update is gradient descent that uses the error gradient information (∇E) to do a small step in the direction of the negative gradient according to Eq. 2.8, where η is known as learning rate.

$$w^{(t+1)} = w^{(t)} - \eta \nabla E(w^{(t)}) \tag{2.8}$$

The error is defined with respect to the whole training set accordingly to Eq. 2.9, where the average term is used due to learning rate parameter in order to have the gradient independent of the size of the dataset (or the batch size).

The evaluation and also the weight update in gradient descent is done based on processing the whole training set. It is not beneficial, since it tends to get stuck at local minima and it is problematic with large datasets.

$$E = \frac{1}{N} \sum_{n=1}^{N} C(y_n, p_n) \tag{2.9}$$

$$E = \frac{1}{N} \sum_{n=1}^{N} E_n \tag{2.10}$$

Stochastic Gradient Descent (SGD) is the basic optimizer for NN training. The error function can be re-defined as in Eq. 2.10, where E_n is the error with respect to one training sample or a batch of training samples. Then, the update step is done for each E_n (after every sample or batch). This approach also allows to escape from local minima since the stationary points should be different for E, E_0 or E_n.

For the batch version of E_n, the average term is added to pass Eq. 2.10 in order to have the gradients independent of the batch size.

2.2.2 Error Backpropagation

The goal of the error backpropagation is to compute the gradients of the error function with respect to each weight in the NN that is needed for weights update. It relies on the chain rule, which is the formula for computing the derivative of the composed function according to Eq. 2.11.

$$\begin{aligned} f(x) &= g(h(x)), \\ f'(x) &= g'(h(x))h'(x), \\ \frac{df}{dx} &= \frac{dg}{dx} = \frac{dg}{dh} \cdot \frac{dh}{dx} \end{aligned} \tag{2.11}$$

The forward pass in MLP from Eq. 2.1 can be divided into computing the weighted sum of the inputs according to Eq. 2.12 and applying activation function according to Eq. 2.13. The notation $w_{ji}^{(l)}$ represents the i^{th} weight of the j^{th} neuron that is present in the l^{th} layer.

$$a_j^{(l)} = \sum_{i=0}^{M} w_{ji}^{(l)} z_i^{(l-1)} \tag{2.12}$$

$$z_j^{(l)} = \varphi(a_j^{(l)}) \tag{2.13}$$

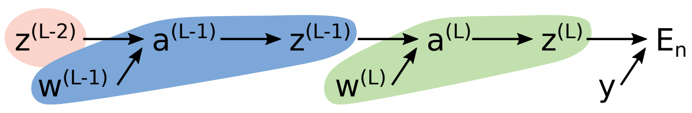


Figure 2.3: MLP computational graph

The computational graph of the MLP is illustrated in Fig. 2.3. Using the chain rule, the partial derivative for the weights in the last layer can be reformulated according to Eq. 2.14. This allows to directly compute the partial derivatives for each weight in the last layer of the MLP.

$$\frac{\partial E_n}{\partial w_{ji}^{(L)}} = \frac{\partial E_n}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial w_{ji}^{(L)}} \tag{2.14}$$

An example for regression task with mean squared error and linear activation function can be seen in Eq. 2.15.

$$\frac{\partial E_n}{\partial z_j^{(L)}} = \frac{\partial}{\partial z_j^{(L)}} \sum_{i=1}^{K} (z_i^{(L)} - y_i)^2 = 2(z_j^{(L)} - y_j), \tag{2.15a}$$

$$\frac{\partial z_j^{(L)}}{\partial a_j^{(L)}} = \frac{\partial}{\partial a_j^{(L)}} \varphi_l(a_j^{(L)}) = \frac{\partial}{\partial a_j^{(L)}} a_j^{(L)} = 1, \tag{2.15b}$$

$$\frac{\partial a_j^{(L)}}{\partial w_{ji}^{(L)}} = \frac{\partial}{\partial w_{ji}^{(L)}} \sum_{k=0}^{M} w_{jk}^{(L)} z_k^{(L-1)} = z_i^{(L-1)} \tag{2.15c}$$

Then, the error for the last output layer is given according to Eq. 2.16, where $\delta_j^{(L)}$ stands for the error of the j^{th} neuron in L^{th} layer.

$$\delta_j^{(L)} = \frac{\partial E_n}{\partial a_j^{(L)}} = \frac{\partial E_n}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial a_j^{(L)}} \tag{2.16}$$

For the precedent layers l ($L-1$, $L-2$ etc.), the error is backpropagated accordingly to Eq. 2.17. Compared to forward pass visualized in Fig. 2.1, the error backpropagation is in reversed direction. Therefore, the sum runs over the k connections to which the output of j^{th} neuron in l^{th} layer is connected. Note that the $\delta_k^{(l+1)}$ is already computed.

$$\begin{aligned}\delta_j^{(l)} = \frac{\partial E_n}{\partial a_j^{(l)}} &= \sum_k \frac{\partial E_n}{\partial a_k^{(l+1)}} \frac{\partial a_k^{(l+1)}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial a_j^{(l)}} \\ &= \sum_k \delta_k^{(l+1)} \frac{\partial}{\partial z_j^{(l)}} \Big(\sum_{i=0}^{M} w_{ki}^{(l+1)} z_i^{(l)}\Big) \frac{\partial}{\partial a_j^{(l)}} \varphi(a_j^{(l)}) \\ &= \varphi'(a_j^{(l)}) \sum_k w_{kj}^{(l+1)} \delta_k^{(l+1)}\end{aligned} \tag{2.17}$$

Finally, the partial derivative with respect to weights in l^{th} layer can be computed according to Eq. 2.18. This can be iterated for additional layers providing all necessary gradients for weights update.

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \tag{2.18}$$

It should be clear, that the combination of the specific loss function and activation function does not matter for the backpropagation, as long as their derivatives can be obtained. On the other hand, the combination is usually done to satisfy Eq.

2.19 since a linear gradient in the output layer is beneficial for convergence reasons.

$$\delta_j^{(L)} = \frac{\partial E_n}{\partial a_j^{(L)}} = z_j^{(L)} - y_j \tag{2.19}$$

This can be verified for mean squared error and linear activation function from Eq. 2.15a and 2.15b.

In summary, error backpropagation can be done in three steps as follows:

1. Compute the error of the output layer using Eq. 2.16 or 2.19.
2. Backpropagate the error using Eq. 2.17.
3. Compute the derivatives using Eq. 2.18.

For the batch training, only the average term is added. It results in averaging the gradients of data samples in the batch. It is not equal to computing the gradients of average error due to non-linear activation functions.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are widely used especially for image classification, but can also be useful in other tasks. Main parts of CNN are convolutional, pooling and fully-connected layers which allow to extract features, propagate the important ones and then classify, respectively.

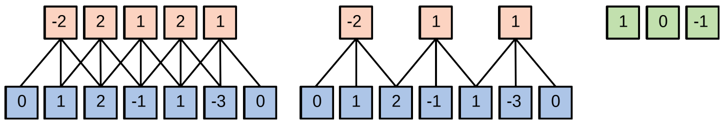


Figure 2.4: Convolutional layer without activation function ignoring the bias for simplicity with stride 1 (left) and 2 (right): input (blue), output (red), kernel (green)

As depicted in Fig. 2.4, the neurons in the convolutional layers have a limited field of view and shared weights. Thus, they are computationally and memory efficient and they are able to extract the features independently of their position in the input. The shared weights are referred to as kernel. Based on stride parameter, the computation briefly moves the kernel over the input and performs the dot product. Usually, the Rectified Linear Unit (ReLU) activation function (see Fig. 2.2b) follows. Frequently, more kernels are used and their results are stacked together. The result of the convolutional layer is referred to as a feature map and its number of channels (sometimes referred to as depth) depends on the number of kernels used.

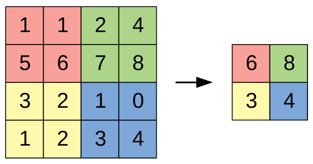


Figure 2.5: 2x2 max-pooling operation

Pooling layer usually follows after convolutional ones to filter or aggregate the information. The max-pooling layer selects the maximal activation from the area as illustrated in Fig. 2.5. Alternatively, the average can be computed. Finally, the fully-connected layers are employed.

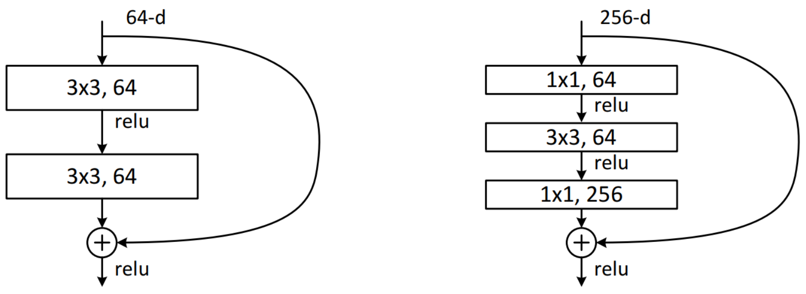


Figure 2.6: ResNet residual block (left) and bottleneck residual block (right) [26]

These layers are crucial for CNNs such as VGG-19 [54] which is a 19 layers deep architecture. On the other hand, the increasing number of layers is not always beneficial as presented in ResNet paper [26]. An 18-layer CNN was better than a 34-layer on the ImageNet dataset. The probable explanation discussed in the paper is that “the deep plain nets may have exponentially low convergence rates”. To solve this problem, they used residual blocks (see Fig. 2.6) which consist of convolutional layers and provide shortcuts for the information and also the gradient flow resulting in 152-layer CNN and stunning results.

2.4 Fully Convolutional Networks

Fully Convolutional Networks (FCNs) consist of convolutional and pooling layers. They evolved from CNNs as a much more efficient model for “sliding window” classification tasks. For example, the task of image segmentation can be solved as a pixel labeling problem, where each pixel is classified into corresponding classes. The classification of neighbouring pixels can be done with CNN in a sliding window manner. But, it results in huge overlap of the input windows and also the computed features. Moreover, the neuron in fully-connected layer has limited field of view and “shared” weights for each window so it could be replaced with convolutional layer. The benefit is that the overlapped features are computed only once.

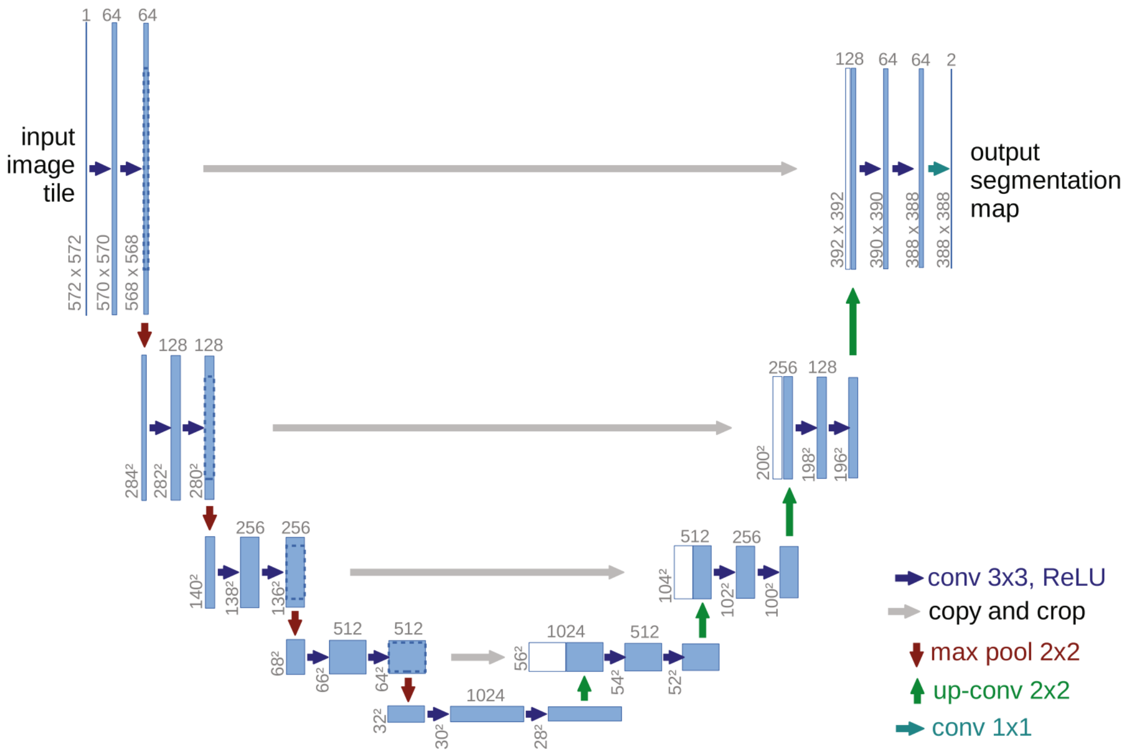


Figure 2.7: Architecture of U-Net [51]

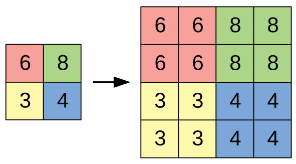


Figure 2.8: 2x2 upsampling operation

An example of FCN is a well-known U-Net [51] for biomedical image segmentation. It has significantly affected general image segmentation including the document image segmentation. It is used in a sliding window manner providing segmentation mask for each tile of the input image. According to Fig. 2.7, the architecture consists of an encoder (left part) and a decoder (right part). Since it is FCN, it consists mainly of convolutional layers. They are used to extract features and to process these features to reconstruct the segmentation mask in the encoder and decoder part, respectively. Skip-connections (see "copy and crop" in Fig. 2.7) are an important part that allows combination of local and contextual features and thus improve localization accuracy in the result. They work similarly to residual blocks discussed in the previous section. According to Fig. 2.8, the upsampling is used as a reverse operation to max-pooling in order to match dimensions of encoder and decoder parts. This is followed by a convolutional layer reducing the depth of the feature map (noted as "up-conv 2x2" in Fig. 2.7). An alternative for upsampling is a transposed convolutional layer, which can be understood as the reverse operation for the convolutional layer. According to Fig. 2.9, it basically outputs the kernel multiplied by the input value.

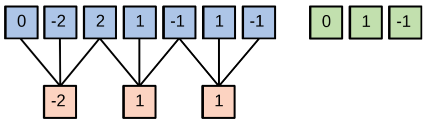


Figure 2.9: Transposed convolutional layer without activation function with stride 2: input (red), output (blue), kernel (green)

2.5 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are adapted for data sequences such as time series or text where both global (long-term) and local (short-term) dependencies usually play an important role. For example, the information provided by speech depends on the currently spoken words but also on the context of the conversation.

There is a concept of "memory" or, in other words, storing the information. The information can be stored for example in recursive connection where the output of the neuron at time step t_n is taken as the input of the same neuron at time step t_{n+1}. But this type of a recursive network has problems with long-term dependencies and is prone to vanishing and exploding gradient problem. [27]

These problems are adressed by Long Short-Term Memory (LSTM) [27, 19, 56] which can memorize both long-term and short-term dependencies. According to Fig. 2.10, the LSTM cell visualization contains input x, output h, time step t, cell state C, cell state update candidate g, forget gate f, input gate i and output gate o.

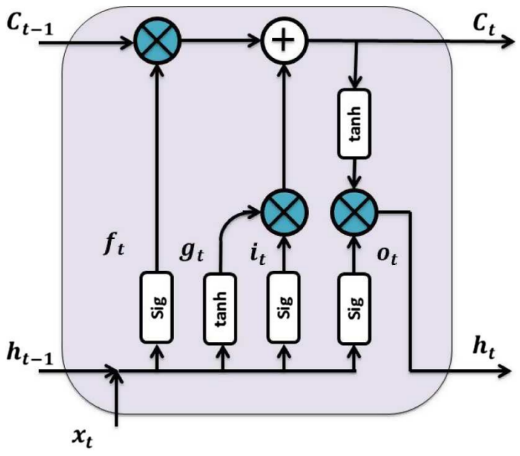


Figure 2.10: Long Short-Term Memory (LSTM) cell [56]

$$g_t = tanh(W^{(g)}x_t + U^{(g)}h_{t-1} + b^{(g)}) \tag{2.20a}$$
$$f_t = sigmoid(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{2.20b}$$
$$i_t = sigmoid(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \tag{2.20c}$$
$$o_t = sigmoid(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \tag{2.20d}$$

Forget gate f_t decides which data are forgotten since the output ranges from 0 to 1 and Hadamard multiplication is performed with cell state C_{t-1}.

For the input gate, new candidate values to update the current state are provided by g_t. These are filtered by input gate i_t in the same way as in forget gate. The result is added to the current cell state resulting in C_t.

Similarly, the output gate o_t decides which parts of the cell state C_t to output.

2.6 Connectionist Temporal Classification

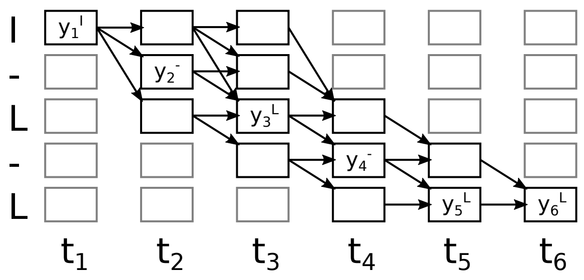


Figure 2.11: The idea of CTC for HTR of labeling *ILL*: All possible paths are visualized

Connectionist Temporal Classification (CTC) [22] trains the network to label unsegmented sequences. The input can be speech signal but also the image of a text line. It has made a huge impact in the field of HTR. Since there is no need for character alignment, GT can be the transcription of the line.

The output is the sequence (further referred as *path*) that consists of an alphabet and a blank. Blank is a special label (usually noted as "-") which is used for repeated labels. For example, `ILL` labeling can be represented as `IL-L` or `IIII-LL-LLL` depending on the output sequence length. As depicted in Fig. 2.11, the crucial step is that the network outputs are transformed into a conditional probability distribution over label sequences. Given the labeling, we know the order of desired labels (`I-L-L`) and thus all the possible paths. According to Fig. 2.11 given the inputs x and predictions y, the probability estimate of path `I-L-LL` can be computed according to Eq. 2.21.

$$P(I - L - LL|x) = y_1^I \cdot y_2^- \cdot y_3^L \cdot y_4^- \cdot y_5^L \cdot y_6^L \tag{2.21}$$

The goal is to find the most probable labeling which can be found dynamically using Viterbi algorithm as the best path. Once it is found, it can be used as the labeling for weights update of the network.

But most probable path is not the most probable labeling. Thus, a many-to-one mapping β can be used alternatively (e.g. $\beta(I-L-LL) = \beta(ILL--L) = ILL$). It sums the probabilities of all the paths corresponding to certain labeling accordingly to Eq. 2.22 and it can be computed dynamically using the CTC forward-backward algorithm [22].

$$P(ILL|x) = \sum_{path \in \beta^{-1}(ILL)} P(path|x) \tag{2.22}$$

2.7 Transformer

The architecture known as Transformer is presented in paper "Attention is all you need" [63]. Transformer-based networks are now considered a state of the art in many tasks and have a major impact on solving NLP tasks such as machine translation.

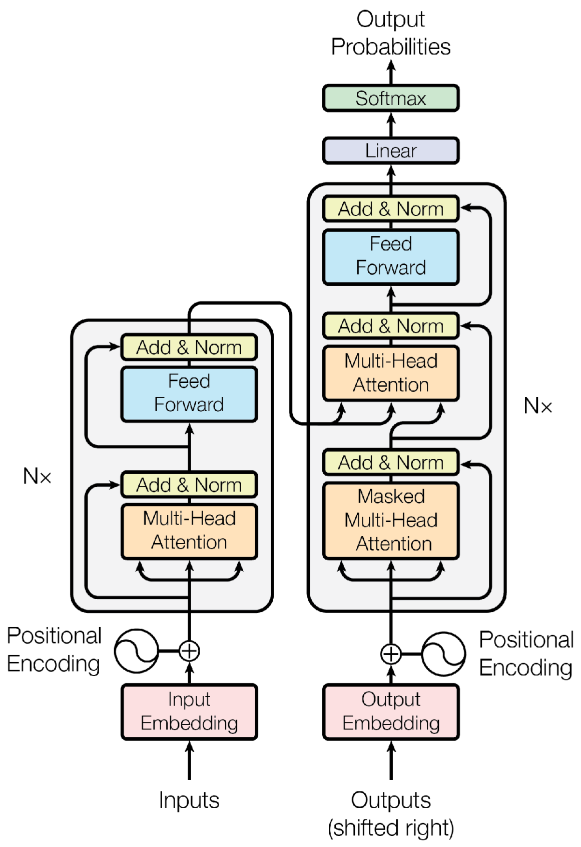


Figure 2.12: Transformer model architecture [63]

According to Fig. 2.12, it consists of an encoder and decoder part which encodes the input and decodes it into the target domain, respectively. The crucial blocks are Multi-Head and Masked Multi-Head Attention. Both blocks utilize scaled dot-product attention according to Eq. 2.23.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.23}$$

The scaling term $\sqrt{d_k}$ is used for gradient reasons. If Q and K components are independent random variable with mean 0 and variance 1, the QK^T has mean 0 and variance d_k, but we prefer the variance of 1.

Q, K and V are matrices representing the query, key and value feature vectors. These vectors are learned from the input vectors (word embeddings or other feature vectors) via fully connected layer. For each query vector, the $softmax$ part basically weights the key vectors based on the similarity with query vector. This is visualized in Fig. 2.13, where the dot-product of query vector q with key vector k_1 will be greater than with k_2. Multiplication with V then provide the weighted sum of related learned value vectors.

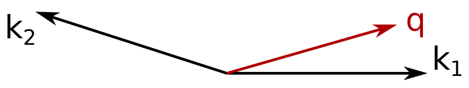


Figure 2.13: Dot-product attention idea

For every input vector (represented by query vector), this scenario allows to aggregate information (represented by value vector) about other input vectors (represented by key vector). If the input is a word, its vector is able to aggregate information about other words. Thus, it can provide contextual representation. Moreover, it can be done parallelly and thus the computational time can be reduced compared to RNNs due to input time steps dependencies.

On the other hand, the information about its position in the input is lost. Therefore, the positional encoding is made. It adds the position vector to the embedding vector so the information about position can be provided. In [63], the position vector is based on sin and cos functions. Alternatively, the learned position vectors can be used.

The network is initially proposed for machine translation, where the encoder takes the whole source sentence and encodes it. The decoder then decodes it into the target language in the next word prediction scenario. Therefore, the decoder input is shifted target sentence. The attention block of the decoder takes the key and value from the encoder. The query vector is obtained from target domain (decoder). To allow parallel computation during training, the decoder attentions $QK^T/\sqrt{d_k}$ in Masked Multi-Head Attention are masked so the model can not see the word it should predict. Otherwise, it will only copy that word to the output.

During the inference, the decoder input starts with special SOS token. The prediction is then the first translated word. Then, the new word extends the decoder input until the EOS token is predicted.

Chapter 3

Document Image Analysis

Document Image Analysis (DIA) is the main part of historical document analysis and "its objective is to recognize the text and graphics components in images, and to extract the intended information as a human would" [42]. It includes tasks like segmentation, object detection, OCR or HTR. As depicted in Fig. 3.1, these tasks are usually divided into textual and graphical processing which deals with textual and graphical regions, respectively. Usually, the processing is task-dependent and therefore the processing structure presented in Fig. 3.1 can be modified.

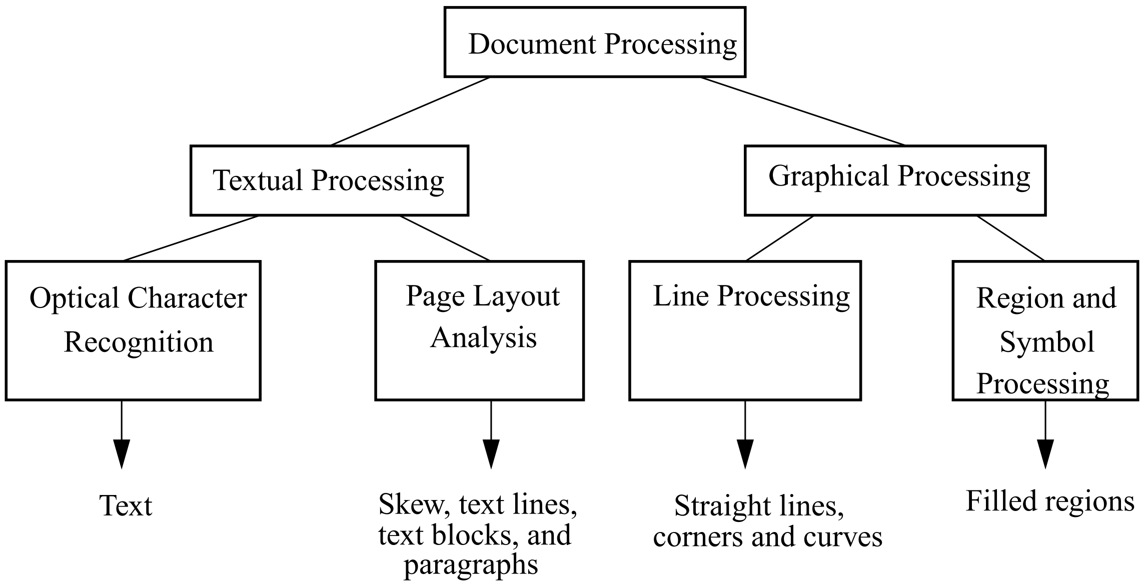

Document Processing
Textual Processing
Graphical Processing
Optical Character Recognition
Page Layout Analysis
Line Processing
Region and Symbol Processing
Text
Skew, text lines, text blocks, and paragraphs
Straight lines, corners and curves
Filled regions

Figure 3.1: Document Image Analysis areas [42]

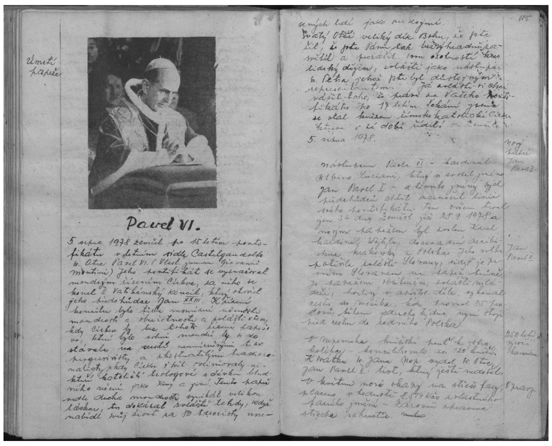
Since the input is usually the image of a scanned page, the first section is related to graphical processing. The layout analysis follows in a separate section since it benefits from graphical processing methods and it can help to distinguish between textual and graphical regions. Finally, the textual processing approaches are presented.

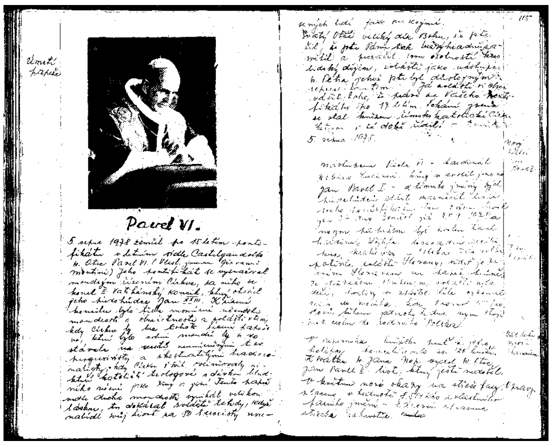
3.1 Graphical Processing

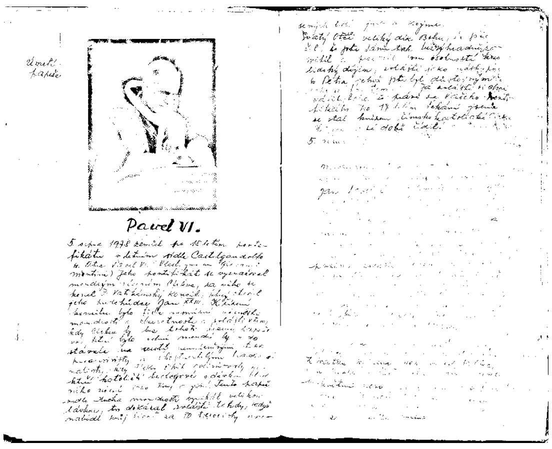
The task of historical DIA is usually very specific and problematic for conventional methods thus it is beneficial to use (deep) ML. On the other hand, annotated data are rare and the conventional methods can help.

For example, drastic improvement in localization accuracy or amount of data needed can be achieved utilizing binarization and conventional image post-processing methods as shown in [10].

3.1.1 Binarization


(a) Input grayscale image


(b) Result of Otsu's method [44]


(c) Result of Adaptive Document Binarization method [53]

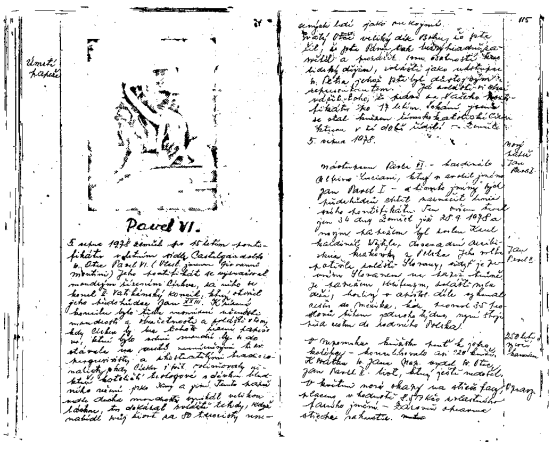

(d) Result of Recursive Thresholding for Documents [41]

Figure 3.2: Examples of binarization

Image binarization is a process that converts the image into black and white image. A basic approach for image binarization is to select an adequate threshold from gray level histogram. Then, pixels of gray level image are divided into two classes.

Generally, the approaches can be divided into global and local (also referred to as adaptive) thresholding. The global methods use a single threshold value which is not appropriate for historical documents due to degradation, brightness inconsistency and inhomogeneous objects like text. The adaptive methods use a local features that allow them to overcome this problem. The quality of the result depends on the input and selected binarization method as depicted in Fig. 3.2.

Otsu's Method

Otsu's method [44] is a straightforward non-parametric binarization method that provides an automatic threshold selection.

In an ideal image histogram, there is a well-recognizable valley between two peaks representing objects and background so the threshold can be easily selected. Nevertheless, real pictures do not have such a histogram. Otsu's method deals with this problem by selecting the optimal threshold from histogram by maximizing the separability of the two classes.

The gray-level histogram is normalized and regarded as a probability distribution according to Equation 3.1, where p_i represents the probability of the pixel with intensity i, n_i denotes the number of pixels with intensity i and N total number of pixels. L stands for the number of gray levels.

$$p_i = \frac{n_i}{N}, \qquad p_i \geq 0, \sum_{i=1}^{L} p_i = 1 \tag{3.1}$$

Then, the optimal threshold $k*$ is computed by maximizing the between-class variance $\sigma_B^2(k)$ according to Equation 3.2.

$$\begin{gathered} \sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]} \\ \mu_T = \mu(L) = \sum_{i=1}^{L} i p_i, \qquad \mu(k) = \sum_{i=1}^{k} i p_i, \qquad \omega(k) = \sum_{i=1}^{k} p_i \end{gathered} \tag{3.2}$$

Adaptive Document Binarization

Sauvola et al. [53], as a representative of a group of methods based on local mean and standard deviation of pixel values, address the problems related to historical documents such as brightness inconsistency and differing quality. The goal of the method is to provide the optimal threshold for each pixel. For the speed improvement, the threshold is computed for every n^{th} pixel and other thresholds are interpolated.

The text binarization method utilizes the local mean m and local standard deviation s. A threshold T is computed for each pixel according to the Equation 3.3. According to [53], user defined parameters are set experimentally as $R = 128$ and $k = 0.5$.

$$T = m[1 + k(s/R - 1)] \tag{3.3}$$

Recursive Thresholding for Documents

The method for binarization of scanned documents that can deal with variable writing style, stroke intensity and noise is presented in [41]. The authors stated that: "Global thresholding suffers from the need to find a single threshold value that will satisfactorily binarize the handwriting in a document."

Selecting a single threshold value could lead to several problems in historical documents due to degradation and brightness inconsistency. On the other hand, locally adaptive methods can amplify noise that is usually present in historical documents. To deal with these problems, the method composes a set of pixels iteratively.

There is a great idea of background estimation and removal that significantly improves binarization in handwritten documents. This idea also provides functionality similar to locally adaptive methods, although the thresholding is performed globally. Since the background is removed, the brightest component in the image simply represents the background.

To deal with noise, bilateral filtering and hysteresis are employed. So, new pixels are accepted only if they are connected to the previously identified text pixels. The disadvantage of the method is the number of user defined parameters.

FCN Binarization

The document image binarization can be solved as a pixel-labeling task with FCN accordingly to Sec. 2.4. It can significantly reduce noise or even complete the missing parts. The drawback is that the training data are needed.

For example in [61], the input is a grayscale image together with relative darkness features. The output is the mask of foreground probabilities that can be easily thresholded resulting in a binarized image.

3.1.2 Segmentation

In image processing, it is important to extract objects of interest for further processing. The task of image segmentation, or semantic segmentation, could be understood as the task of extracting homogeneous components from an image. In document image, homogeneous components may represent text characters, text blocks, lines of text, tables or pictures. The goal is to provide the segmentation mask.

Terminologically, the task of segmentation does not include the classification of components, but it is important to understand that these tasks are difficult to separate since the segmentation basically classifies pixels.

A simple example of image segmentation can be the result of a binarized image since the foreground and background are segmented. Conventional segmentation approaches used for printed documents like Page Segmentation Based on Thinning of Background [30] are now surpassed by deep learning approaches.

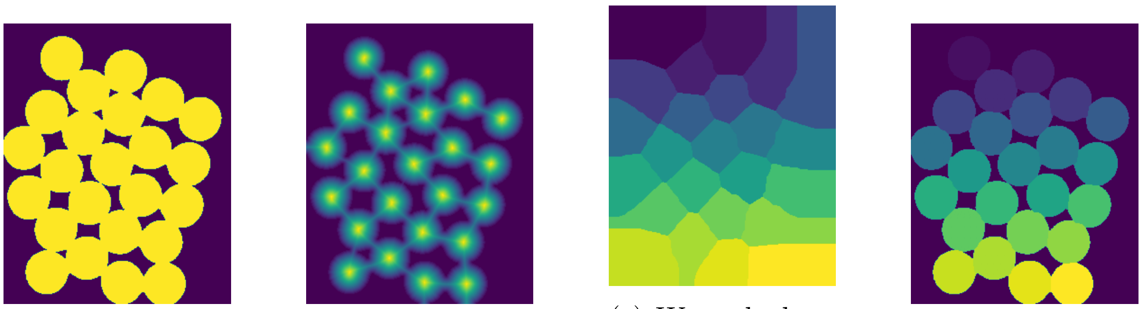
Thinning of Background

An interesting approach for page segmentation is based on thinning of background [30]. It allows segmentation of an arbitrarily rotated page with a non-overlapping layout.

The input of the method is a binarized image of the page. Firstly, the background thinning is applied. Then, terminal pixels are filtered out, so only closed chains remain. The next step removes unnecessary chains and merges the areas. Their removal is done if they satisfy Eq. 3.4, where D represents the minimum distance from chain to black pixels and W stands for the difference of average line widths of two neighboring regions. Parameters t_D and t_W are set to 6 and 31, respectively.

$$t_D \cdot W + t_W \cdot D \leq t_D \cdot t_W \tag{3.4}$$

Watershed



(a) Input (b) EDT (c) Watershed result (d) Result

Figure 3.3: Coins segmentation with watershed example: different coins in different colors

The watershed [5] segmentation can be used on a grey level image or on a binarized image using Euclidean Distance Transform (EDT) according to Fig. 3.3. The basic idea is placing the water source in a local minimum (Fig. 3.3b). The regions are decided according to the positions where different water sources meet (Fig. 3.3c).

It can be also useful for instance segmentation (Sec. 3.1.3) since it can separate the instances from segmentation mask in some cases. For example, the mask of coins (Fig. 3.3a), which have the similar shape and size, can be multiplied by watershed segmentation result (Fig. 3.3c) resulting in separate coins (Fig. 3.3d). On the other hand, this is hard to achieve on general shape because the water source position has to be decided correctly.

Graph-Based Image Segmentation

A graph-based image segmentation is proposed in [16]. It provides the region segmentation as depicted in Fig. 3.4. Firstly, it divides an image into regions. Boundary between two regions is then evaluated based on the inter-region and within-region

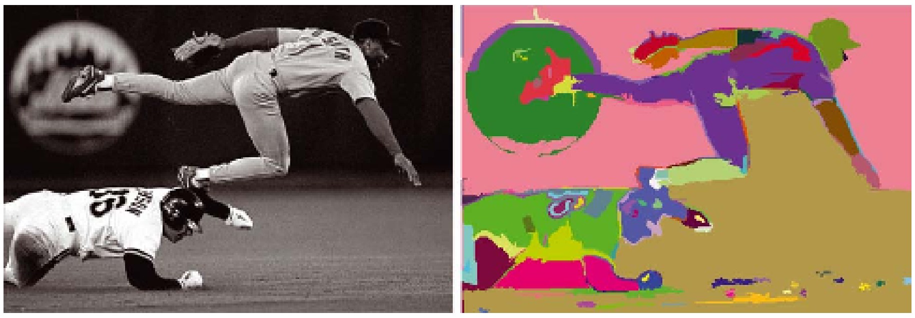


Figure 3.4: An example of graph-based image segmentation results [16]

differences comparison. The method is able to distinguish between low and high variability regions.

dhSegment

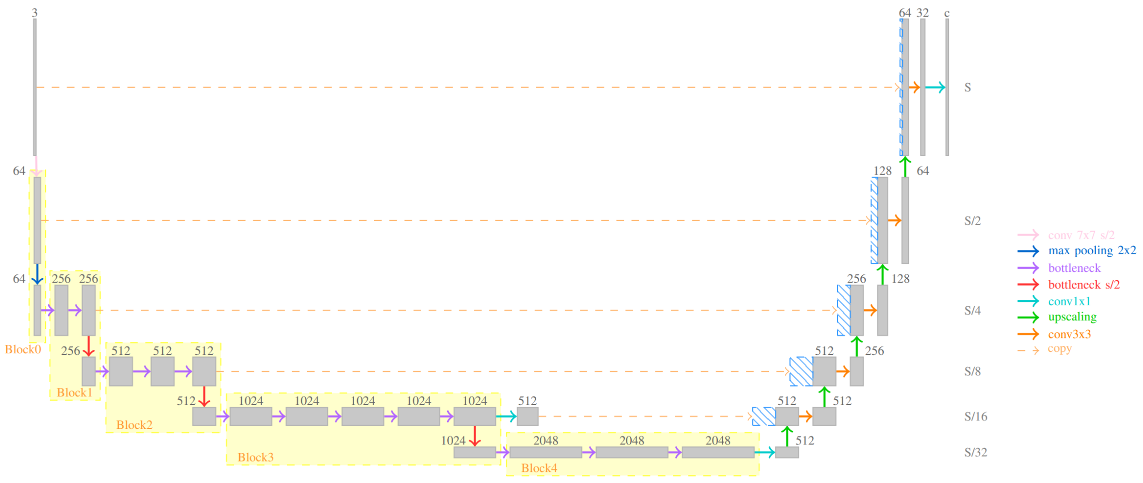


Figure 3.5: Architecture of dhSegment: The yellow blocks correspond to modified ResNet-50 [43]

The architecture combining U-Net [51] and ResNet-50 [26] is presented in [43] as dhSegment. According to Fig. 3.5, the ResNet-50 is used as an encoder part of the U-Net. It consists of the general mask prediction and task related post-processing for fine-tuning. It is successfully used for page extraction, baseline detection and document layout analysis.

FCN for Historical Document Segmentation

FCN for page segmentation of historical document images is proposed in [64] together with evaluation score Foreground Pixel Accuracy (FgPA). It comes with the idea that only the foreground pixels are crucial for the document segmentation (see

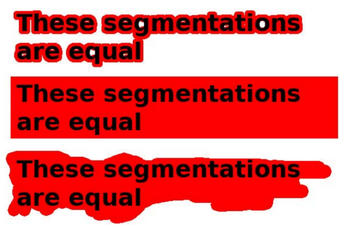

These segmentations are equal
These segmentations are equal
These segmentations are equal

Figure 3.6: Idea of Foreground Pixel Accuracy [64]

Fig. 3.6). The architecture (see Fig. 3.7) consists of encoder and decoder parts. Compared to U-Net, there are no skip-connections and instead of up-sampling followed by convolutional layer, transposed convolutional layers are used in the decoder. The input is a downsampled document image and the output is in a form of segmentation mask. The paper presents also the post-processing based on connected components. The connected component label is selected as the most frequent one.

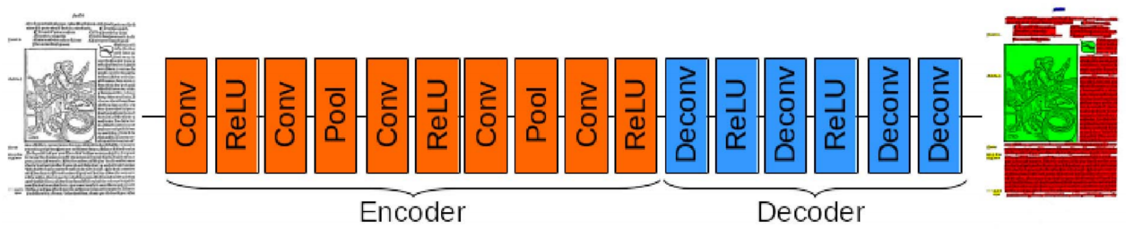


Figure 3.7: Architecture of FCN for page segmentation [64]

ARU-Net

A two-stage method for text line detection in historical documents is presented in [23]. ARU-Net architecture is used for baseline mask prediction in the first stage. The second stage consists of post-processing based on superpixel extraction and their clustering.

The ARU-Net architecture extends the U-Net architecture by residual blocks (R) and spatial attention (A). Residual blocks provides shortcuts for the information flow. The attention briefly multiplies the RU-Net output to focus on related parts of the input as depicted in Fig. 3.8. Since the input is provided at multiple resolutions, this allows the network to focus on image content at different positions and scales.

Paragraph line segmentation with RNN

Another interesting method that allows the paragraph segmentation into the text lines is proposed in [40]. It requires only the number of lines in the input paragraph as GT. To do that it utilizes RNN and CTC. The network learns to label vertical coordinates for the presence of the baselines. Thus, the solution is not ideal for curved or rotated baselines.

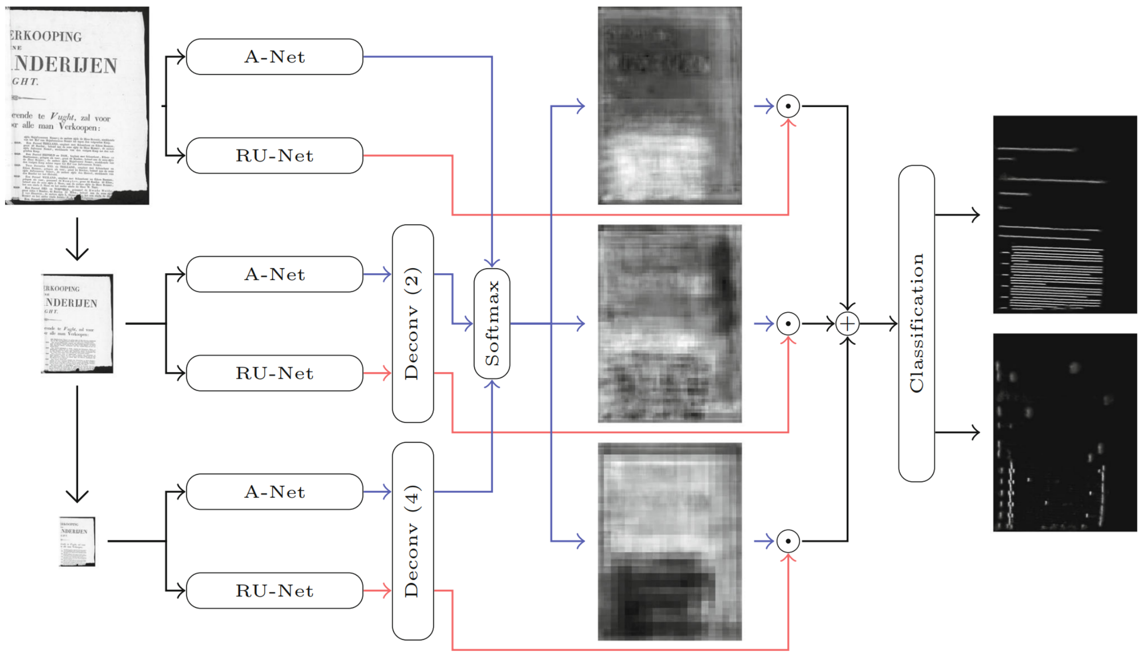


Figure 3.8: ARU-Net architecture [23]

3.1.3 Object Detection

The goal of object detection, or instance segmentation, is to detect and delineate each distinct object of interest in the input image. This is usually achieved by providing the bounding boxes and their classes which allows to access the separate Regions of Interest (RoIs).

This is hard to achieve with the segmentation mask provided by semantic segmentation since it does not distinguish between objects. This can be illustrated in Fig. 3.3a where we are interested in separate coins. Contrary to semantic segmentation, the instance segmentation should provide bounding box for each coin.

Template Matching

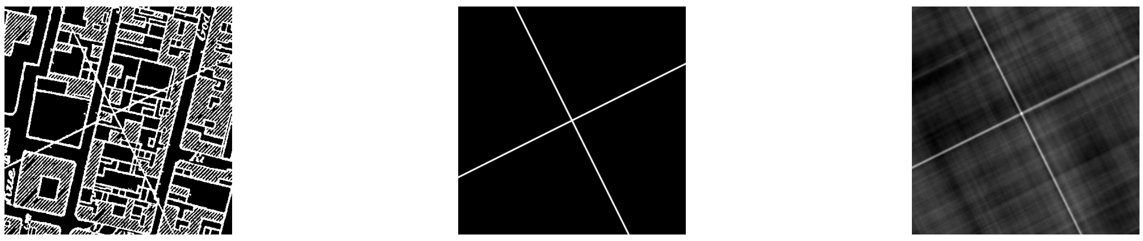

(a) Image (b) Template (c) Result

Figure 3.9: Template matching example: The match as the maximal value in the result

Template matching is used to detect a given template in the image. It compares directly the pixel values of the input image and the template in a sliding window manner. One of the possibilities is the normalized correlation coefficient [7] which is used for the result in Fig. 3.9c. It is related to Pearson correlation coefficient and computed accordingly to the Eq. 3.5, where I and T denote the image and template pixel values, respectively. Image and template coordinates are represented by x, y and x_T, y_T, respectively.

$$R(x,y) = \frac{\sum_{x_T,y_T} T(x_T,y_T) \cdot I(x+x_T, y+y_T)}{\sqrt{\sum_{x_T,y_T} T(x_T,y_T)^2 \cdot \sum_{x_T,y_T} I(x+x_T,y+y_T)^2}} \tag{3.5}$$

It can also be beneficial during post-processing. In [10], it was used for correction of detected graticule lines intersections as illustrated in Fig. 3.9.

Hough Line Transform

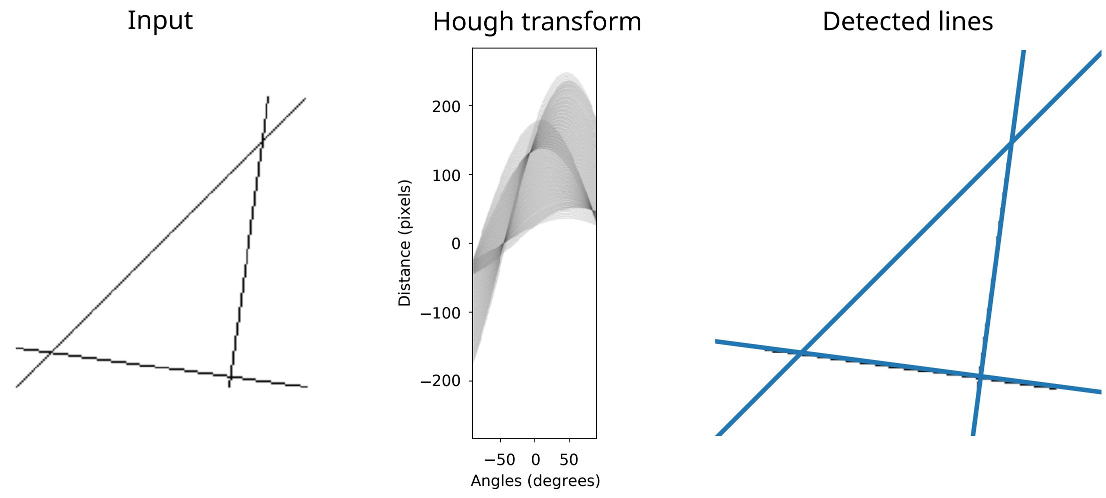


Figure 3.10: Lines detection using Hough line transform (inverted colors)

Hough line transform [15] is a powerful algorithm for detection of lines. It is based on Hesse normal form of a line according to Eq. 3.6. It transforms the image from x, y space into θ, r parameter space by a voting procedure where each pixel votes for a set of lines. The lines can be then detected as local maxima as depicted in Fig. 3.10.

$$r = x\cos(\theta) + y\sin(\theta) \tag{3.6}$$

Selective Search

The problem of generating possible object locations is addressed in [62]. The selective search algorithm for region proposal starts with initial segmentation into regions using [16] (see Sec. 3.1.2) at different scales (see Fig. 3.11). These regions are then hierarchically grouped resulting in region proposals. Finally, region proposals are

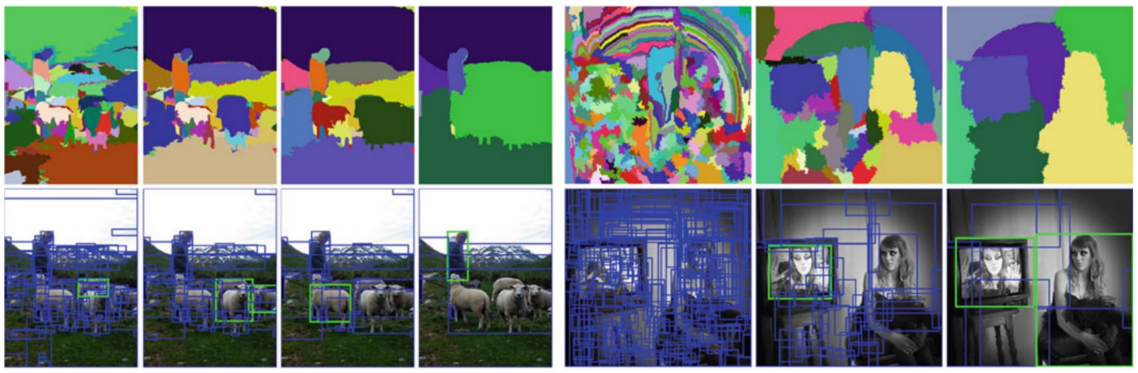


Figure 3.11: Example of selective search at different scales [62]

classified using support vector machines classifier to filter the regions according to GT.

Region-Based Convolutional Neural Networks

Currently, the mainstream methods in object detection are deep neural network models like Region-based Convolutional Neural Networks (R-CNNs). These networks typically use pre-trained backbones (e.g ResNet-50 [26]) to extract features and region proposals.

Similar to selective search [62], an approach for efficient object detection using deep CNN has been proposed in [21]. R-CNN system consists of region proposal, feature extraction and classification modules. Proposed regions are separately processed

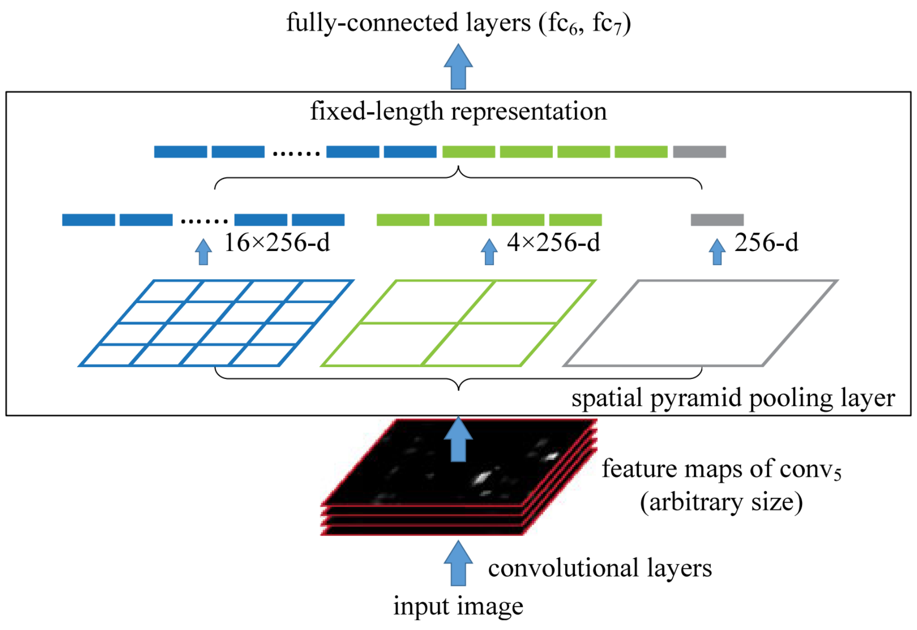


Figure 3.12: The idea of Spatial Pyramid Pooling [25]

and classified for presence of the desired objects. The system achieved promising results but the drawbacks are multi-stage training and the computational inefficiency due to the number of RoIs and thus classifications per single image.

An approach based on Spatial Pyramid Pooling (SPP) has been proposed by He et al. [25]. The main strength of this approach is the capability to use a specific pooling operation that generates a fixed-length representation regardless of image size or scale (also the bounding box size). According to Fig. 3.12, the pooling regions are defined by the grid. Since convolutional layers can process varying input resolutions thanks to shared parameters, the pooling is made on resulting feature maps providing fixed-length representation for fully connected layers. The contribution is that there is no need to resize and distort the input image.

In 2015, Ross Girshick went further in his research proposing Fast R-CNN [20] with several innovations to deal with the drawbacks and speed-up. Compared to R-CNN, the image features are calculated only once. Based on the regions, the region features are pooled into a fixed size feature space using RoI pooling (which can be understood as 1-level SPP) or RoI align (which employs also bilinear interpolation). This way, the image features does not have to be computed per each RoI separately.

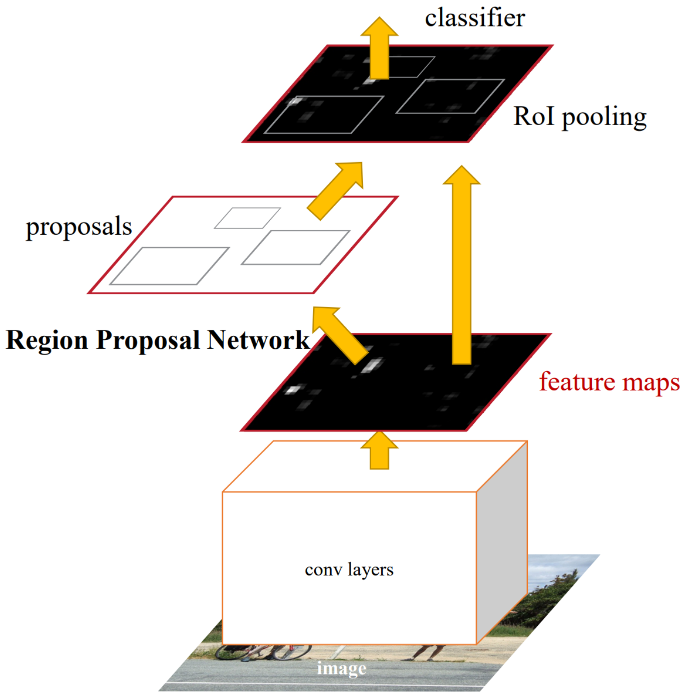


Figure 3.13: Feature sharing in the Faster R-CNN model [50]

The selective search algorithm for region proposals was still considered a drawback resulting in an end to end Faster R-CNN model [50]. Instead of selective search algorithm, the model is extended by Region Proposal Network (RPN). RPN is implemented by convolutional layers since it works in a sliding window manner. It predicts the coordinates of the box and estimates the object's presence probability. As illustrated in Fig. 3.13, RPN shares the image features with the RoI classification module and is faster than selective search.

Additionally, the mask R-CNN [24] combines semantic and instance segmentation. It adopts the Faster R-CNN [50] scenario and outputs also segmentation mask for each RoI.

You Only Look Once

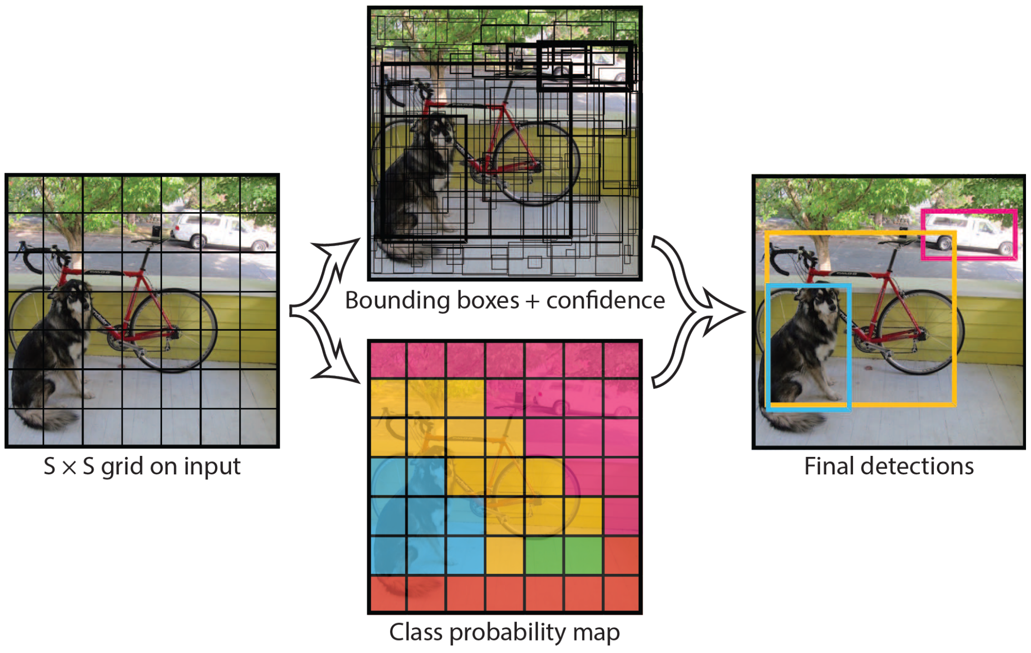


Figure 3.14: You Only Look Once (YOLO) model detections using a grid [49]

You Only Look Once (YOLO) [49] is even faster than Faster R-CNN and allows real-time object detection. Briefly, as says the network name, you only look once at the image to predict the bounding boxes, confidences and the class probabilities for these boxes as depicted in Fig. 3.14. It uses the grid over the image and predicts the boxes in each cell of the grid. The trade-off is that each cell in the grid predicts a limited amount of boxes and has only one class. Compared to Faster R-CNN, this could cause wrong classifications and problems if there are more nearby objects.

3.2 Layout Analysis

Document layout analysis is crucial for further processing of the documents. The goal is basically to describe the input document page and separate it into RoIs. Such a region can be a block of text, image, paragraph or text line. Sometimes more detailed information is needed. For example, the text region can be of heading, page number or main text type. Then, different processing can be made according to the block type. It can take into account also reading order of the text blocks which is beneficial in assembling the transcriptions of document by OCR/HTR methods.

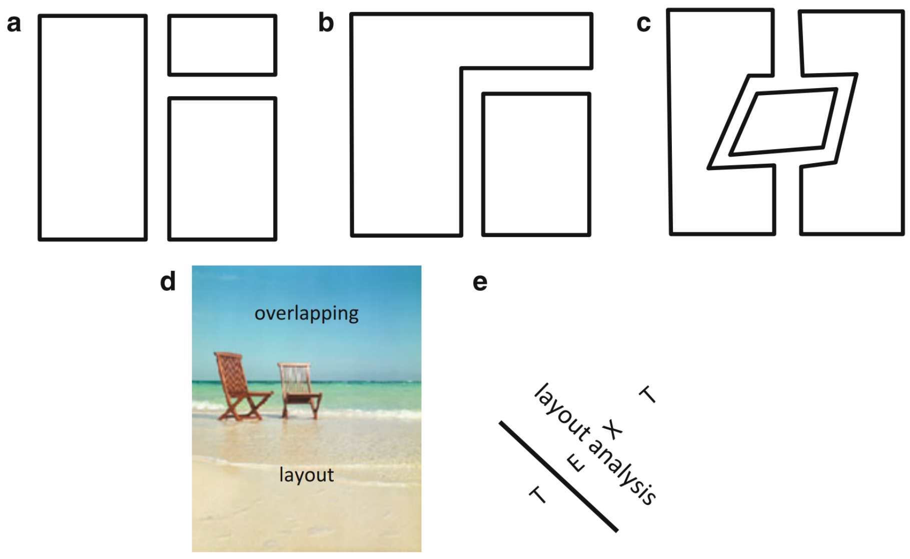


Figure 3.15: Classes of layout: (a) rectangular, (b) Manhattan, (c) non-Manhattan, (d, e) overlapping layout [31]

These blocks can be distributed differently in the input page as depicted in Fig. 3.15. The layout classes are rectangular, Manhattan, non-Manhattan and overlapping [31]. The layout of historical handwritten documents is usually the most challenging overlapping layout where the blocks can overlap. It is often the case in handwritten chronicles where the text can overlap the image (Fig. 3.15.d) or text (Fig. 3.15.e).

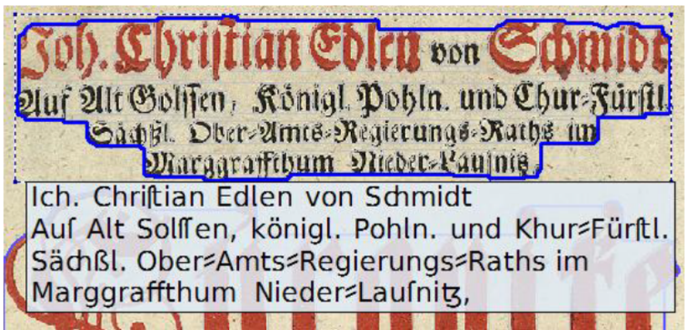

Ich. Chriſtian Edlen von Schmidt
Auf Alt Solſſen, königl. Pohln. und Khur=Fürſtl.
Sächßl. Ober=Amts=Regierungs=Raths im
Marggraffthum Nieder=Lauſnitʒ,

Figure 3.16: Aletheia text region annotation example [13]

Alehteia [13] is a helpfull tool for time consuming annotation of document pages which works with xml-based PAGE (Page Analysis and Ground-truth Elements) format. It allows precise and detailed annotation of several regions including text, image or separator. The region is represented by boundary polygon as depicted in Fig. 3.16. Additionally, there are several metadata options related to specific regions. For example, the text region can contain text transcription, text and background color, font, subtype and so on. There is also an option to specify reading order, baselines and other objects related to layout and document analysis. Additionally to manual tools, it provides automatic tools for layout and document analysis which, unfortunately, do not work well for handwritten documents.

The tasks of layout analysis can be successfully solved by semantic (Sec. 3.1.2) or instance segmentation (Sec. 3.1.3) as in the OCR system for historical documents [36]. It employs semantic segmentation models for text, baseline and separator segmentation. The separator is basically a line which divides the page and separates the text blocks. The reading order is then determined recursively using the separator information. Finally, the text recognition is done with CTC-based model (see Sec. 3.3.2 for more details).

3.3 Textual Processing

Transcription of a text image into a full-text representation is one of the most wanted and usable parts of document analysis. It allows to automate processes and significantly reduces the time of searching for related information. It can be solved with OCR, HTR or keyword spotting approaches.

According to [39], the first patent on OCR is from 1929, so OCR is quite an old task and it is practically solved. Several fonts like OCR-B [18] were proposed for easier OCR. The solution can be made even with conventional approaches based on template matching for example.

Compared to OCR for printed text, HTR is more challenging due to connected characters in handwritten text. The visual differences occur even between same characters written by the same author. There are even more problems in historical documents caused mainly by document degradation. Moreover, there is not defined layout and the baselines are not even straight and often rotated. Because of these challenging problems, the number of specialties and dependencies, it is beneficial to learn these using ML approaches. This may be the reason why (deep-)learning methods are much better than conventional approaches in HTR area.

3.3.1 Keyword Spotting

The early approaches for processing of handwritten text were based on keyword spotting and solved as an image similarity problem. In Query by Example (QbE), the input is a template image and the desired result is a list of similar images, usually word images. There is also Query by String (QbS) scenario, where the input is a string and a result is a list of word images containing corresponding text.

The PHOC-Net [58] approach employs CNN and Spatial Pyramid Pooling [25] thus it accepts different input resolutions. The input is an image of a word for which the Pyramidal Histogram Of Characters (PHOC) vector is predicted. As depicted in Fig. 3.17, the PHOC vector for the training phase is created from the string transcription according to the selected levels. In level 2 for example, it provides 2 binary vectors for the presence of the character in the first and second half of the word. The resulting vector is the concatenation of these vectors.

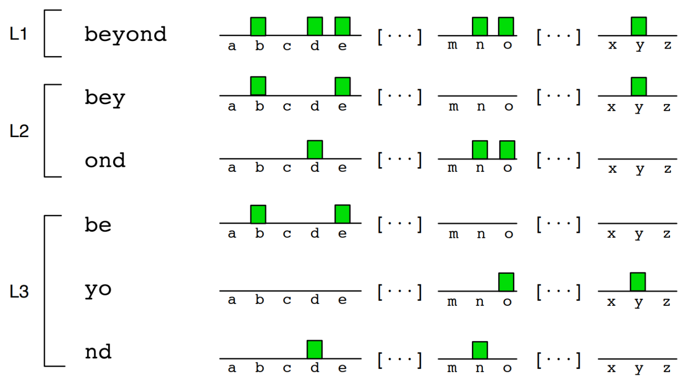


Figure 3.17: Example of PHOC vector for 1, 2 and 3 levels [58]

Since all word images have their own PHOC vector predicted by PHOC-Net, this approach allows both, QbE and QbS, scenarios. For QbE, the PHOC vector of the input template image is predicted. For QbS, the PHOC vector is created in the same way as in the training phase. Finally, the spotting is performed by comparing the input PHOC vector with other PHOC vectors.

3.3.2 Text Recognition

Compared to keyword spotting, the transcription into full-text provides more options and better usability.

OCR and HTR can be successfully solved as a labeling task with CTC. It is used for example in [36]. The typical architecture (Fig. 3.18) takes text line image as an input and extracts visual features using convolutional and max pooling layers. Then, fully-connected and Bidirectional Long Short-Term Memory (BiLSTM) layers are employed as a character-level language model. Finally, the sequence is labeled using the CTC (see Sec. 2.6).

An example of the Transformer-based (see Sec. 2.7) approach is presented in [34]. The vanilla Transformer [63] is used for "translation" of text line image into its transcription. The result is new state of the art in both OCR and HTR. According to Fig. 3.19, the encoder input consists of image patches. The decoder then works with a sequence of ground truth tokens from text transcription.

The modification of the Transformer for HTR is presented in [28]. Contrary to [34], it uses ResNet-50 [26] as an image feature extractor providing the input for Transformer encoder. The ground-truth, decoder input and output is character-level.

In [65], the authors warn about the usage of Transformers for HTR since the Transformer requires a larger training dataset in order to overcome CTC-based approaches. On the other hand, it is able to learn a strong language model. The approach to reduce the amount of data needed is presented in the same paper. It consist of forward and backward Transformer and voting procedure. The forward Transformer works in reading order and the backward in reverse order. Voting is based on matching the predictions and resolving the conflicts using several heuristics.

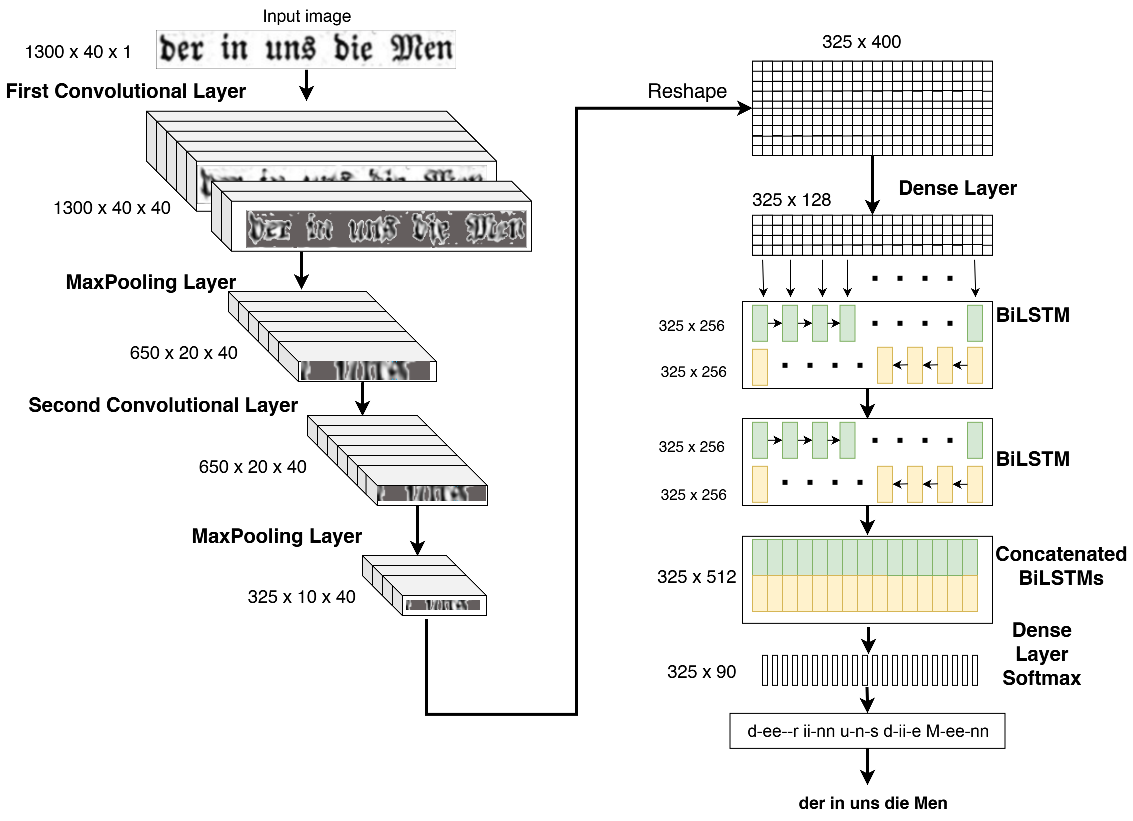


Figure 3.18: Architecture for OCR/HTR with CTC [36]

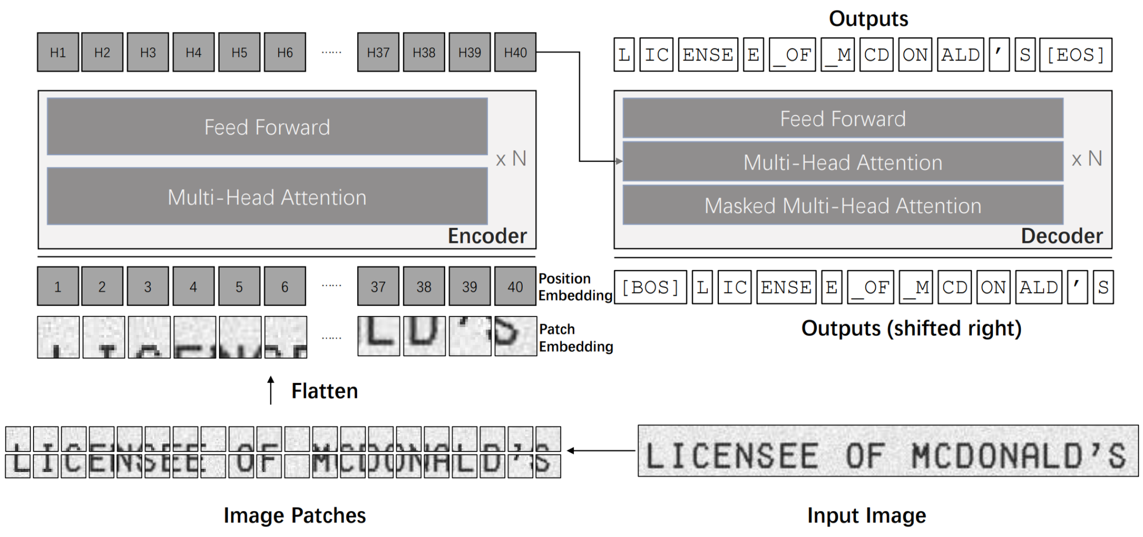


Figure 3.19: Transformer-based approach for OCR [34]

Chapter 4

Natural Language Processing

NLP goes further and its added value is natural language "understanding". According to [12], it faces three major problems: thought process, the representation and meaning of the linguistic input and the world knowledge.

Therefore, an NLP system, usually, starts at word level representations. Naturally, the word has a meaning and represents a part of speech or a lnamed entity. Moreover, there are words like "orange" that have several meanings. That meaning depends on the context (surrounding words) which should be taken into account. Therefore, the next step are contextual word representations. The combination of words can further result in the meaning of a sentence or overall environment.

Usually, the vector representation in multidimensional space is used and is further discussed in Sec. 4.1 and 4.2. These representations help to solve the tasks such as sentence classification, question answering or sentence tagging as presented in Sec. 4.3. The result can be utilized for information retrieval, more detailed analysis and additional functionality such as semantic or multilingual searching.

4.1 Word Representation Using Word2Vec

Breakthrough Word2Vec [37] word representations utilize the idea that the word meaning (and representation) corresponds to the context it usually appears in. According to Fig. 4.1, the goal of the Skip-gram model is to predict the surrounding words of the input word and its training objective is to maximize the average log probability as defined in Eq. 4.1 where the seguence of T words (w) is given.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j} \mid w_t) \tag{4.1}$$

The word representations that are useful for surrounding words prediction are obtained as a "side product". These representations have interesting properties and may encode linguistic regularities which can be represented as linear translation or other basic operations as depicted in Fig. 4.2. For example as presented in [37], the combination of `vec(Berlin)` − `vec(Germany)` + `vec(Poland)` is a vector close to `vec(Warsaw)` and the combination of `vec(Spain)` + `vec(capital)` is close to

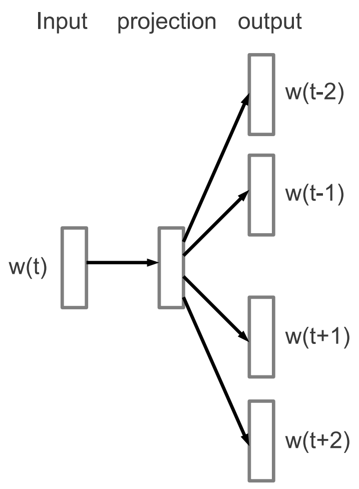


Figure 4.1: The Skip-gram model architecture [37]

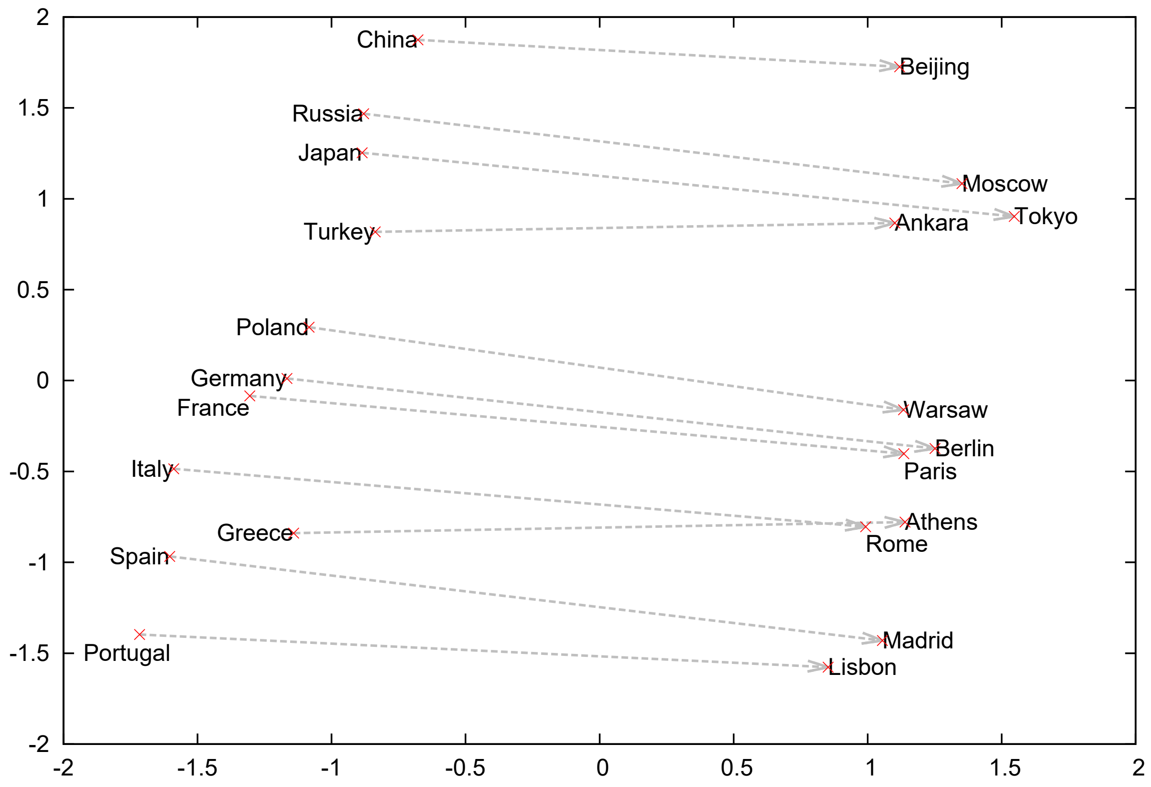


Figure 4.2: Learned Word2Vec representation of countries and their capitals projected by PCA [37]

`vec(Madrid)`. These examples show the possibility of learning the relations and meaning of the words.

4.2 Contextual Representation

The contextual representation takes into account also the context of the word. For example, the word *orange* can represent the color or fruit. Contrary to Word2Vec representation which will result in the same vector per both situations, we expect different representations based on the context.

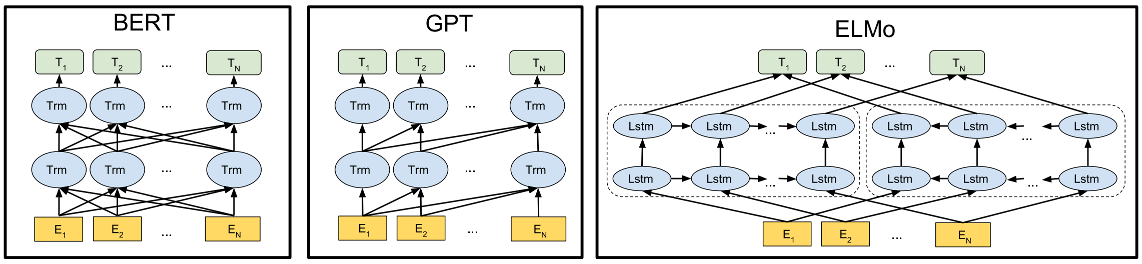


Figure 4.3: Contextual representation approaches [14]

According to Fig 4.3, this can be achieved with RNN as in [45] which is known as Embeddings from Language Models (ELMo). It is able to gather context as a combination of left and right context. There are also Transformer-based (see Sec. 2.7) approaches such as Bidirectional Encoder Representation from Transformer (BERT) [14] and Generative Pre-trained Transformer (GPT) [47] which utilize the encoder and decoder part of the Transformer, respectively. GPT utilizes only the left context, whereas BERT can utilize the left and right context directly which is beneficial but it also prevents efficient standard conditional language model training for the next word prediction (e.g. text generation task).

The idea is that the model is pre-trained on a huge amount of text and then fine-tuned on a specific task. As depicted in Fig. 4.4, the BERT input consists of two sentences. It is pre-trained as a masked language model so it predicts the masked words. At the same time, it classifies if sentence B follows sentence A. This way, the training data can be easily created from general text and general contextual representations are obtained. These days, there is al plethora of pre-trained models available for many languages and tasks. The pre-training can be understood as a "good initialization of weights" for a wide range of tasks.

4.3 Task Dependent Fine-Tuning

Given the pre-trained model that contains good general language understanding, the various tasks can be solved using fine-tuning on a specific task. It is expected that using pre-trained model allows faster training, using smaller dataset and obtaining good results.

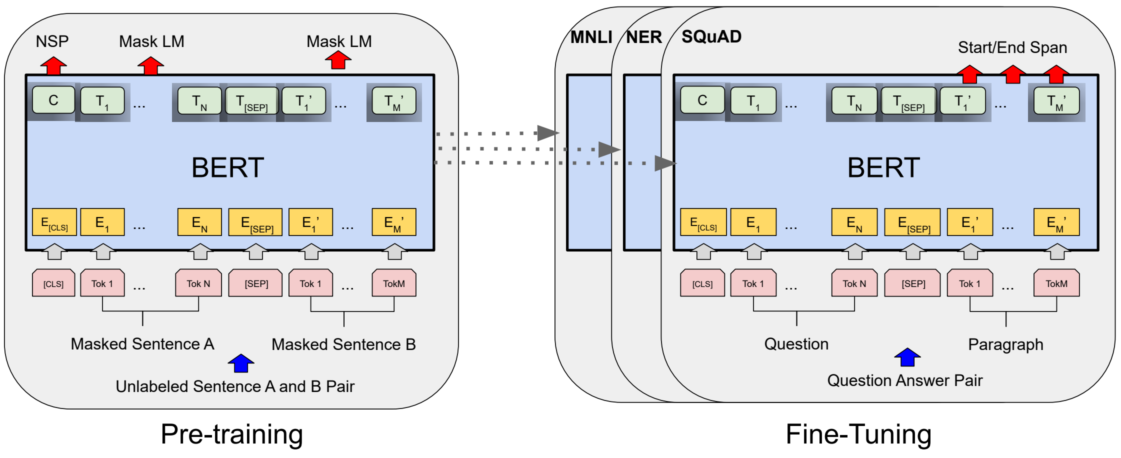


Figure 4.4: BERT pre-training and fine-tuning for specific task such as question answering [14]

The BERT model, discussed in previous section, can be fine-tuned for tasks like classification, question answering or sentence tagging as illustrated in Fig. 4.5.

For the classification (Fig. 4.5.a and 4.5.b), the CLS token representation, that is pre-trained to classify if sentence A follows sentence B, is utilized as an input for classification head (e.g. fully connected layer).

The fine-tuning for sentence tagging is straightforward as depicted in Fig. 4.5.d. Given the corresponding GT, each input token (or word) representation is classified into corresponding class.

The question answering highlights the answer in the input paragraph as depicted in Fig. 4.5.c. To do that, the idea of dot-product attention can be used. Two learned vectors are used as a query for start and end span. The token representations of the input paragraph are considered as keys. Then, the dot-product of the query vector with all key vectors is performed and followed by softmax. It results in a sequence for both start and end span queries. The start and end sequence is then utilized for deciding the start and end span position of the answer, respectively.

4.4 Multilingual Approaches

A multilingual system basically works for data related to two and more languages. Usually, the biggest part of available training data relates to English but a model trained for English will not work properly for other languages without adjustments. There are also poorly-resourced languages with rare training data which can share similar grammar rules with other languages. Therefore, the multilingual models are trained on several languages as in case of mBERT [14] or SlavicBERT [2]. These models are usually fine-tuned on concrete language and task.

Alternatively, the cross-lingual linear transformations can be used as in [8] where bilingual dictionaries are used to project monolingual semantic spaces into a shared space.

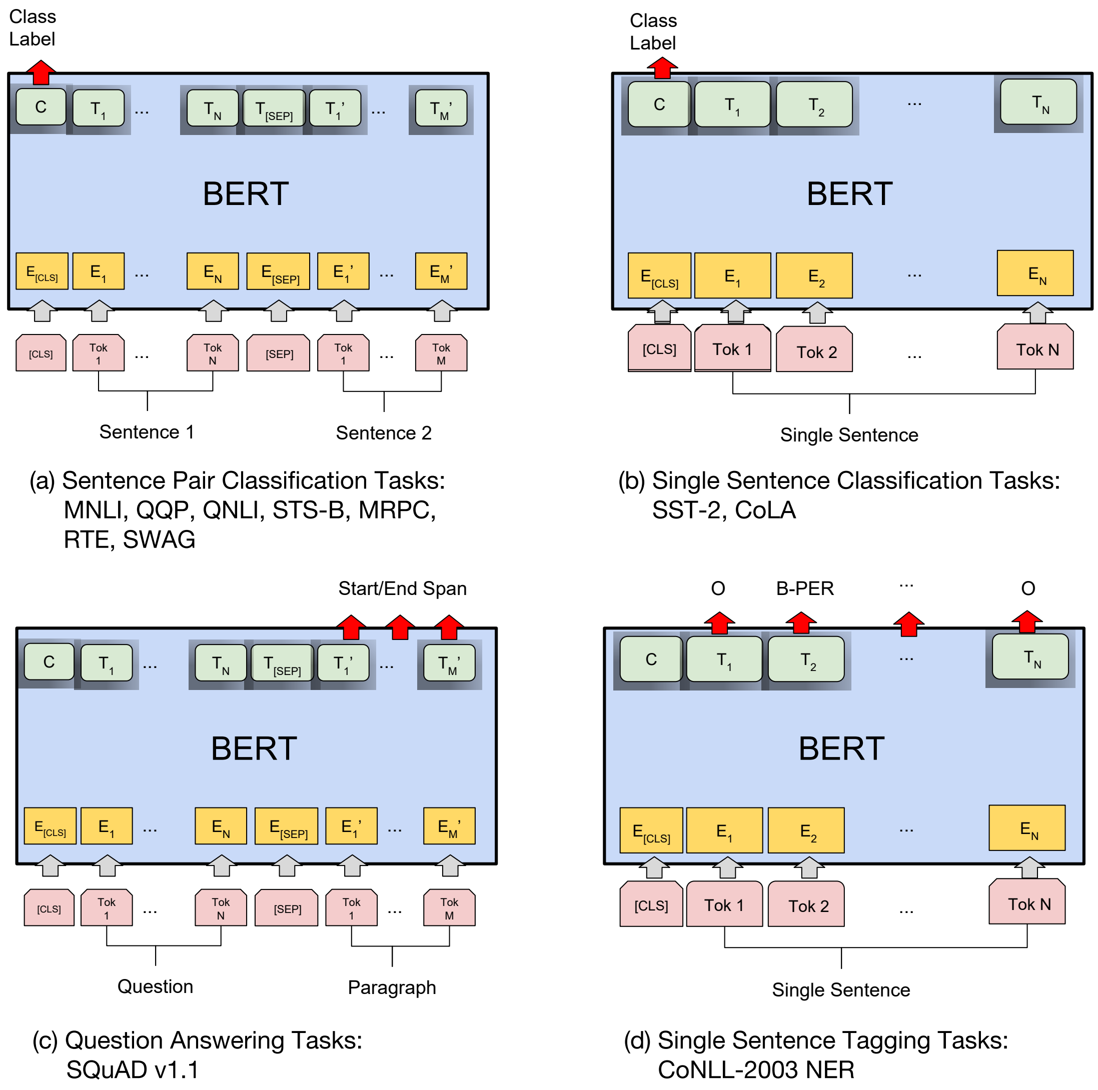


Figure 4.5: BERT task dependent fine-tuning and available datasets [14]

Chapter 5

Multi-Modal Processing

Multi-modality is a natural property in many tasks. However, its usage in machine learning for historical document analysis is not fully explored and the tasks are mostly solved using only one modality. Intuitively, layout analysis should benefit from text transcription. Text transcription can benefit from graphical areas and layout information related to the text. The graphical areas should benefit from text around etc. Therefore, several related research areas and methods where multi-modality is already employed are summarized in this chapter.

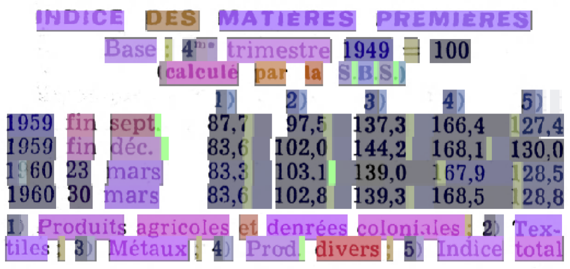
Although there is relatively little work in the field of multi-modal document analysis, it has been shown in many studies that multi-modal processing is beneficial and gives better scores than methods using only one modality as for instance in the case of named entity recognition [38], sentiment analysis [59] or emotion recognition [48]. Multi-modality is also often utilized in video data as proposed in [11] where the authors solve the task of lecture retrieval.

One of the recent papers [17] utilizes EfficientNet [60] in combination with BERT [14] for the document image classification task. The approach uses BERT for capturing the semantic information from recognized text and combines it with the image input handled by the EfficientNet. The two networks are used separately and the final result is obtained from their combination. There is thus room for designing better/joint models to improve the results in these areas.

A Transformer-based approach for joint handwriting and named entity recognition in historical documents is presented in [52]. The architecture is similar to [28] which is discussed in Sec. 3.3.2. Firstly, the model is trained only for HTR. Then, the named entity tags are added and the model is fine-tuned.

Visual and textual features are used for semantic segmentation of historical newspapers in [29]. It utilizes dhSegment architecture (see Sec. 3.1.2) and adds the text embedding map as depicted in Fig. 5.1 which is the only modification. It improves the results significantly.

Li et al. [35] proposed a self-supervised document representation pre-training approach. It utilizes textual, visual and layout information to capture the context of individual blocks in the document. Trained Faster R-CNN (see Sec. 3.1.3) is used as an object detector and visual feature extractor. Then, OCR and BERT are applied to provide textual features. The Cross-Modality Encoder taking into



INDICE DES MATIERES PREMIERES

Base : 4me trimestre 1949 = 100

(calculé par la S.B.S.)

			1)	2)	3)	4)	5)
1959	fin	sept.	87,7	97,5	137,3	166,4	127,4
1959	fin	déc.	83,6	102,0	144,2	168,1	130,0
1960	23	mars	83,3	103,1	139,0	167,9	128,5
1960	30	mars	83,6	102,8	139,3	168,5	128,8

1) Produits agricoles et denrées coloniales ; 2) Textiles ; 3) Métaux ; 4) Prod. divers ; 5) Indice total

Figure 5.1: Text embedding map visualization using PCA projection (R, G, B) [29]

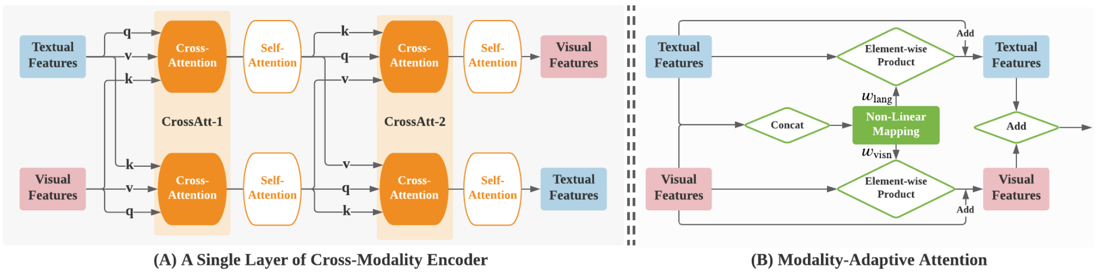


Figure 5.2: Cross-Modality Encoder layer (A) and Modality-Adaptive Attention (B) [35]

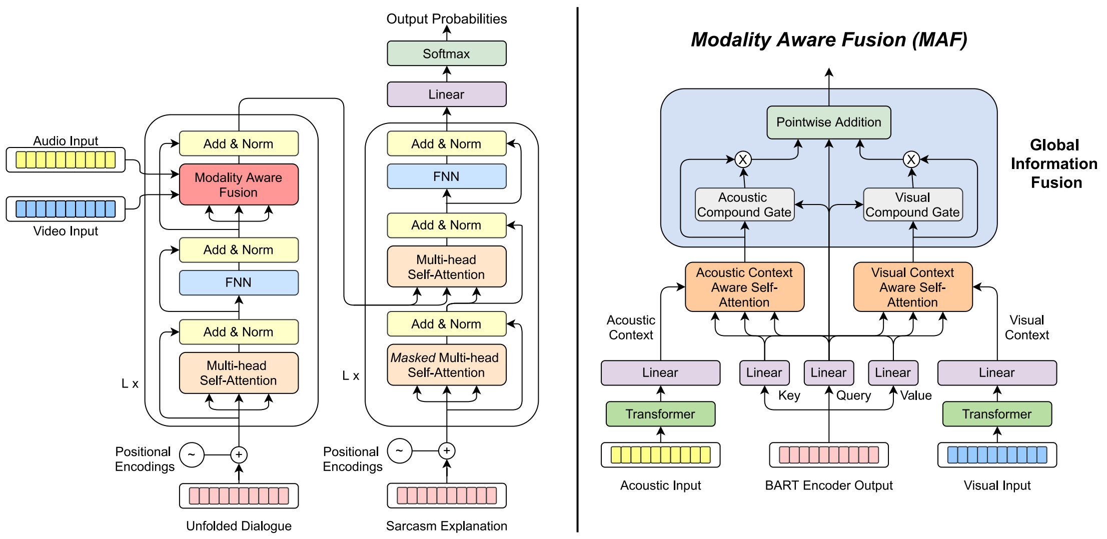


Figure 5.3: Multi-modal Transformer architecture and Modality Aware Fusion block [32]

account both features is proposed accordingly to Fig. 5.2. Pre-training is based on masking strategies that are often used in the Transformer-based architectures. The model is then fine-tuned and evaluated for document entity recognition and document classification tasks using the Modality-Adaptive Attention (Fig. 5.2) to fuse the visual and textual features.

Interesting approach for multi-modal sarcasm explanation is presented in [32]. According to Fig. 5.3, it uses the Transformer architecture for "transcribing the sarcasm into its explanation". It utilizes the text and also the audio and video input. To do that, the encoder is extended by Modality Aware Fusion block that allows to use contextual audio and video inputs. These contextual inputs are processed separately with Context-Aware Self-Attention [67] block and then fused in Global Information Fusion block.

Context-Aware Self-Attention [67] allows to encode contextual information into vector representation. Given the context vector C, it basically modifies the query Q and key K vector of Multi-Head Attention block (see Sec. 2.7) according to Eq. 5.1 where U_Q and U_K are trainable weight matrices and λ_Q and λ_K weights the importance of context representation.

$$\begin{bmatrix}\hat{Q}\\ \hat{K}\end{bmatrix} = (1 - \begin{bmatrix}\lambda_Q\\ \lambda_K\end{bmatrix}) \begin{bmatrix}Q\\ K\end{bmatrix} + \begin{bmatrix}\lambda_Q\\ \lambda_K\end{bmatrix} (C \begin{bmatrix}U_Q\\ U_K\end{bmatrix}) \tag{5.1}$$

Finally, it uses scaled dot-product attention $Attention(\hat{Q}, \hat{K}, V)$ according to Eq. 2.23 as in Multi-Head Attention block.

Chapter 6

Own Work

The proposed approaches and the results encountered on the topic of historical document analysis are presented in this chapter.

6.1 Handwritten Historical Chronicles Segmentation

This section focuses on the segmentation of historical handwritten documents, namely chronicles, for the purposes of layout analysis. We take image, text and background classes into account. For this goal, a new dataset including precise pixel-level annotations in PAGE format is created in [4].

Further in [4], we build on that database but also utilize other datasets for transfer learning in order to improve the results. We discuss a series of experiments, including a novel data augmentation method which creates artificial pages, that evaluate possibilities how to train FCN for image, text and background segmentation.

Based on the experiments, we can say that high resolution is not crucial for the chronicle segmentation into text, image and background. FCN model can generalize well on the documents that are similar but it is hard to create one generalized FCN model that can segment pages of different types and characteristics (e.g. modern printed magazines and historical handwritten documents). In such a case, the model tends to output more noise than the specialized one.

The outcome, in a form of segmentation method with relatively low computational costs and great results, is integrated into the Porta fontium portal to improve its possibilities of searching and publication of the documents.

6.1.1 Dataset Description

The dataset is composed of scanned pages from several chronicles of varying styles provided by Porta fontium portal[1]. The main part of the dataset consists of 5 chronicles in a total of 38 double-sided pages from which 18 contain images.

[1] http://www.portafontium.cz/

There is also an experimental part that contains 20 printed pages from documents of different types. This set contains rare pages that can be used in different application areas. Additionally, this part contains 29 standalone images of old photographs. The experimental part is further used in experiments.

Totally, there are images of 58 good-quality pages of documents and their dimensions vary from 2000 to 5777 pixels.

6.1.2 FCN Architecture

The U-Net-based [51] architecture is designed to segment the entire input page at once. It uses padding in the convolutional layers which can help to suppress noise at the borders of scanned document pages and preserves dimensions so that the input resolution matches the output resolution.

Shared parameters in the convolutional layers allow variable input dimensions. In order to prevent skip-connection dimension inconsistency, the model input dimension has to be multiple of $2^4 = 16$ (given by four 2×2 max-pooling layers).

If there is a high-resolution input, the memory limitations appear. Then, there is again the need to trade-off between localization accuracy and the use of context as discussed in [51]. The high resolution input can be processed in the sliding window manner using small context of the page or it can be down-sampled and processed with less details, bigger context but worse localization accuracy. To reduce computational costs, the input image size is limited to 512×512 pixels as depicted in Fig. 6.1. This setup has been identified based on our preliminary experiments and it is also supported by the work of Wick and Puppe [64] where the authors used input of 260×390 pixels.

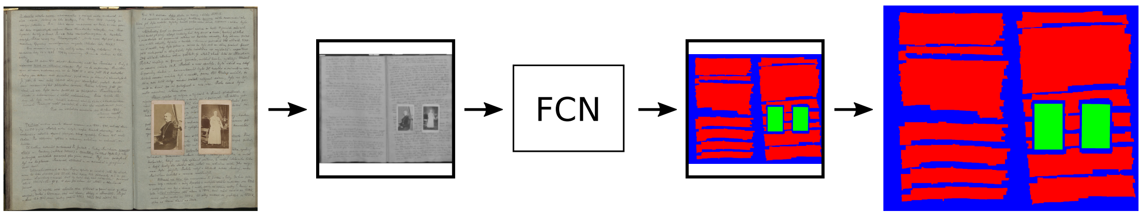


Figure 6.1: Input image segmentation process: The input limit of 512×512 pixels is represented by squares before and after FCN box. [4]

6.1.3 Experimental Results

We have designed a set of experiments for techniques that are used to enhance the recognition results if only small amount of data is available. Namely, we experiment with extending the training data, transfer learning and loss function weighting (Fig. 6.2).

We also evaluate the influence of input resolution (Fig. 6.3) and a post-processing step. The results are reported in Table 6.1 and compared to the *baseline* setup which represents the model trained only on the 6 pages of chronicles that contains an image. Based on the experiments the *combined* setup of the model is made.

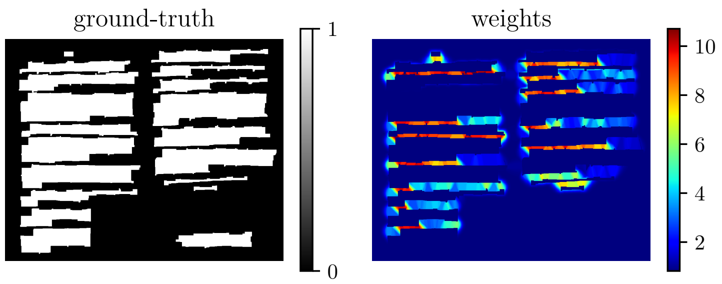


Figure 6.2: Calculated weights for loss function weighting to improve the separation of the components. [4]

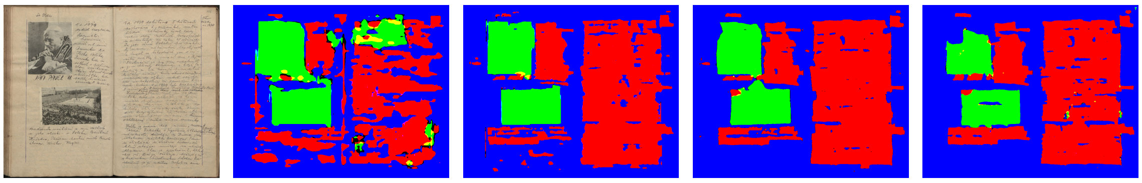


Figure 6.3: Example predictions with different input limits (from left: input image, 128×128, 256×256, 512×512 and 1024×1024 input limits)

For transfer learning, the model is pre-trained on the printed documents from other dataset and then fine-tuned as the baseline setup. The results are slightly worse and the model mispredicts the handwritten text as image more likely. On the other hand, it works better for the glued printed text blocks. This is probably due to the learned features for printed documents during pre-training. The fine-tuning is very fast and takes about 20 epochs compared to 160 epochs for the *baseline* setup. If the model is trained further for roughly the same number of epochs as *baseline*, the results are comparable.

The combined setup is also used for the transfer learning and results are reported for *pre-trained* and *fine-tuned* model separately. For the fine-tuned model, the characteristics are the same as in the previous case. The predictions of pre-trained model are not directly usable as can be seen in Figure 6.4.

The automatic creation of artificial pages from the existing ones is also presented as a data augmentation approach that deals with the problem of class imbalances and brings significant improvements. It allows to utilize annotated pages without images

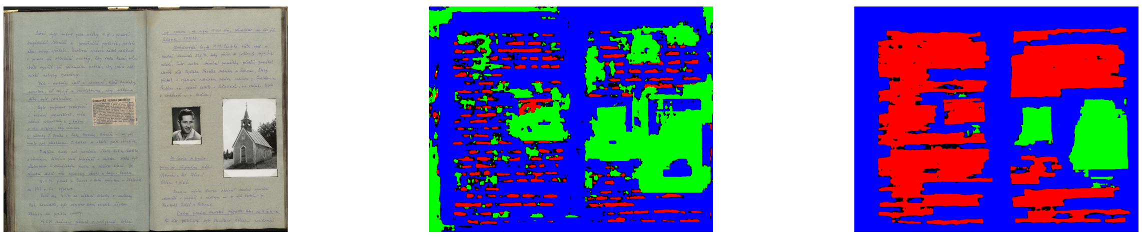


Figure 6.4: From left: the input image, prediction of pre-trained model, prediction after fine-tuning (23 epochs)

Table 6.1: Average results (in %) of the experiments on the validation part: *Baseline* is a referential setup with 512×512 input limit and the model is trained only on 6 pages that contain images. Baseline setup modifications are presented in next three blocks (different input size, loss function weighting, training data). Based on the experiments, the *combined* setup is reported in the next block. The last block contains modifications to the combined setup using post-processing, transfer learning and extended training data.

	Accuracy	**Precision**	**Recall**	**F1 score**	**IoU**	**FgPA**
Baseline	95.3	91.8	92.6	92.0	85.5	98.5
128×128 input	86.6	79.9	82.7	80.7	68.0	93.4
256×256 input	93.9	89.9	91.8	90.7	83.1	98.4
1024×1024 input	95.5	94.1	91.6	92.6	86.5	98.8
Weighted sep. areas	95.3	94.6	90.7	92.3	85.9	99.0
Weighted classes	94.9	92.5	91.4	91.8	85.0	98.3
Augmentation	95.5	93.2	92.7	92.8	86.7	98.4
Artificial pages	96.1	94.0	94.3	94.0	88.9	99.2
Printed pages	95.5	94.2	92.1	93.0	87.1	98.8
Transfer learning	94.8	94.0	89.8	91.6	84.6	98.5
Combined	96.4	94.5	94.3	94.2	89.2	99.2
Post-process	95.9	93.4	94.6	93.9	88.6	99.0
Pre-trained	82.4	73.6	73.8	65.8	52.0	90.1
Fine-tuned	95.8	94.7	91.9	93.1	87.2	99.0
Extended	96.3	95.1	93.3	94.1	89.0	99.4

from the experimental part of the dataset. These no-image pages are problematic for training because of class imbalances. On the other hand, they contain specialties like different writing styles and decorations that are useful for training. To be able to use them, the images are added randomly into a no-image page as depicted in Figure 6.5 with reasonable size and position restrictions.

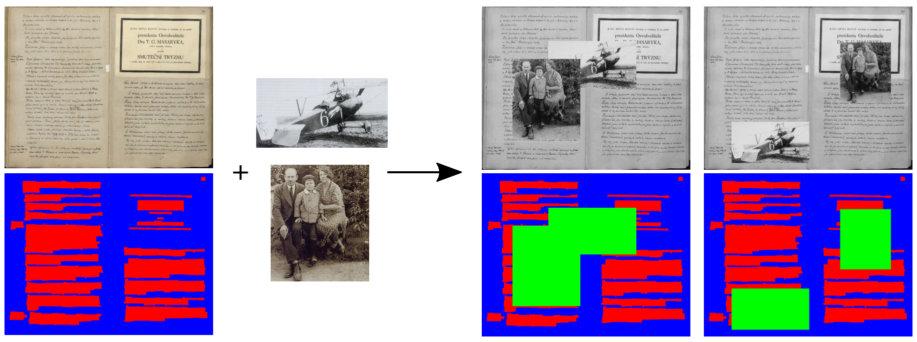


Figure 6.5: Creation of artificial pages: Images are added randomly into document page [4]

6.2 Historical Map Processing

This section relates to processing of historical maps and contains winning approaches of "MapSeg" competition [10] task 2 (segmentation of map content) and 3 (localization of graticule lines intersections).

6.2.1 Segmentation of Map Content

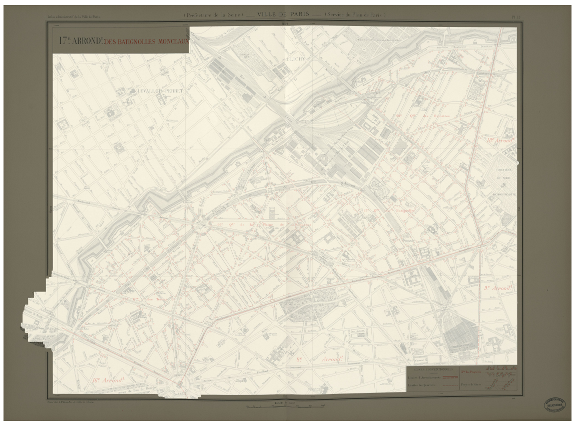


Figure 6.6: Segmentation of map content result example: The map image overlayed with map content area segmentation mask

We propose an efficient approach for automatic map segmentation which combines conventional CV techniques with deep learning methods. It utilizes results of FCN, binarization and post-processing to obtain map content area. ML can easily deal with hardly definable specialties in the documents. Whereas the conventional CV approach can improve the results and further reduce the amount of data needed. Compared to using solely ML or traditional CV techniques, this combination is able to provide excellent results with a small amount of training data as illustrated in 6.6.

We have identified experimentally that predicting only border areas (Fig. 6.7.d) is a much easier learning objective than predicting the whole map content area. In our opinion, the reason for that is given in the following example. If we train the FCN to predict the whole map content area (Fig. 6.7.e), we want to predict every pixel there as positive (e.g. roads, buildings or text). There are also similar objects outside the map content that we want to predict as negative (legends for example). We find this conflict problematic for training. On the other hand, it is much easier to predict only the border areas. The network can focus for example on lines, transitions between "empty" and "non-empty" areas, border decorations or legends. The border contours also appear close to image borders, thus the network can use that information provided by padding as discussed in [4]. Therefore, we proposed the segmentation process accordingly to Fig. 6.7.

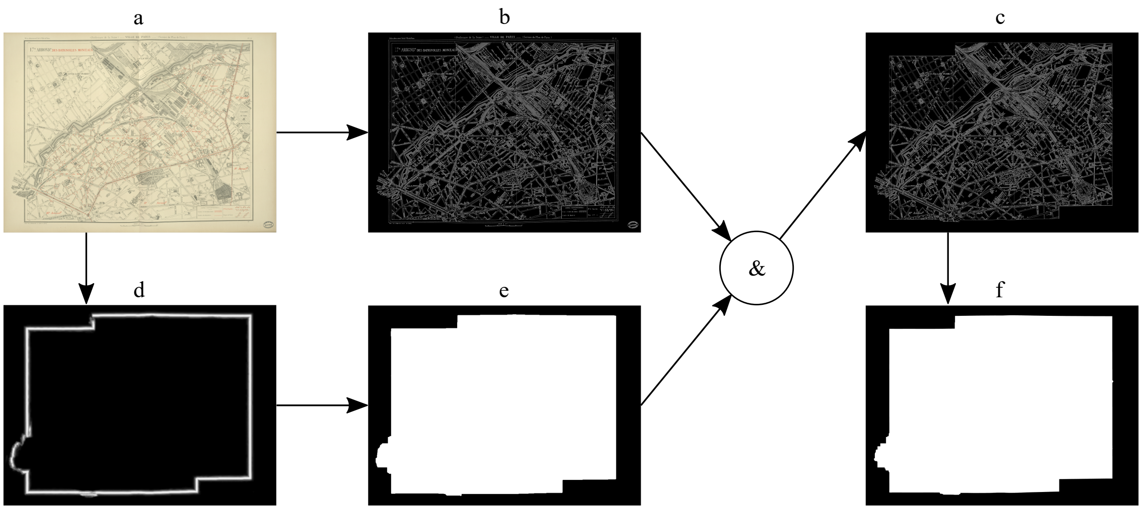


Figure 6.7: Map content area segmentation process: *(a)* input image, *(b)* binarized input image, *(c)* binarized image *b* masked with estimated mask *e*, *(d)* FCN border prediction, *(e)* estimated mask as post-processed border prediction *d*, *(f)* result as post-processed *c*

Our approach significantly outperforms other approaches (CMM [10], IRISA [3] and L3IRIS [10]) as can be seen in Fig. 6.8 where the 95th percentile variant of Hausdorff distance is used as error measure.

Rank	Team	Hausdorf 95 (pix.) ↓
1	UWB	19
2	CMM	85
3	IRISA	112
4	L3IRIS	126

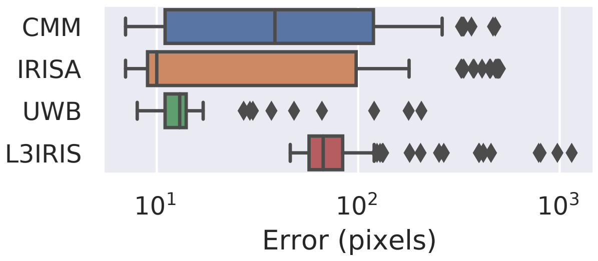


Figure 6.8: Test images errors (left) and error distribution (right) for map content area segmentation task (UWB is ours)

6.2.2 Localization of Graticule Lines Intersections

Important features that can be used for geo-referencing the maps are graticule lines indicating the North/South/East/West major coordinates and their intersections. Another use case is assembling the single map sheets into a larger seamless map. We proposed a novel method that detects a grid as a whole. Based on the results we deduce that focusing on bigger part of the grid (whole grid, cross, longest line etc.) brings more correct detections and is thus more robust.

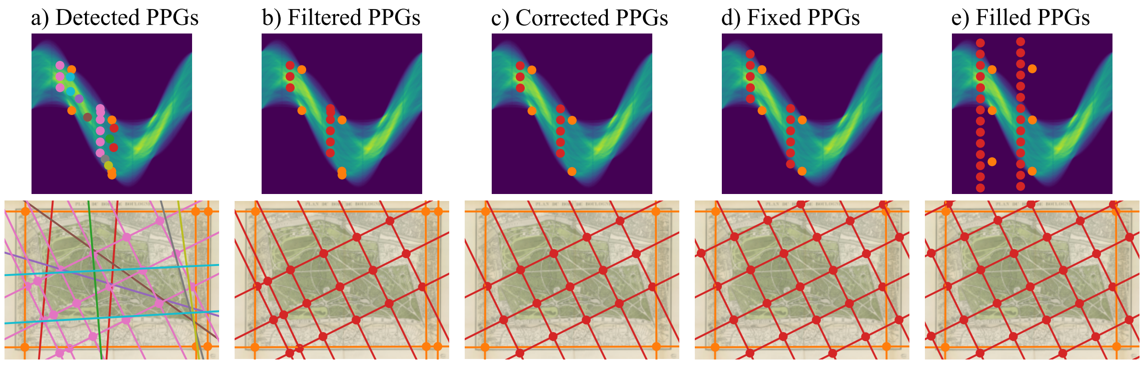


Figure 6.9: Graticule candidate generation process: Hough accumulator (top) peaks from the same PPG and their visualization (bottom) have the same color. Note that image origin is placed at top left corner and y-axis is inverted so the angle goes clockwise.

The approach is based on Hough transform accompanied with a sophisticated post-processing. It works without any training and does not require any annotated data. The proposed approach is very efficient in detecting the rectangular grid and the intersection points. The robustness of the proposed method is demonstrated by evaluating it on another dataset composed of significantly different cadastral map images with excellent results.

Based on formulated presumptions, the grid-forming lines are detected in Hough accumulator in several steps as depicted in Fig. 6.9. The proposed method firstly detects peaks in the Hough accumulator and selects angle candidates. Based on the angle candidates, peaks are grouped into Perpendicular Peak Groups (PPGs).

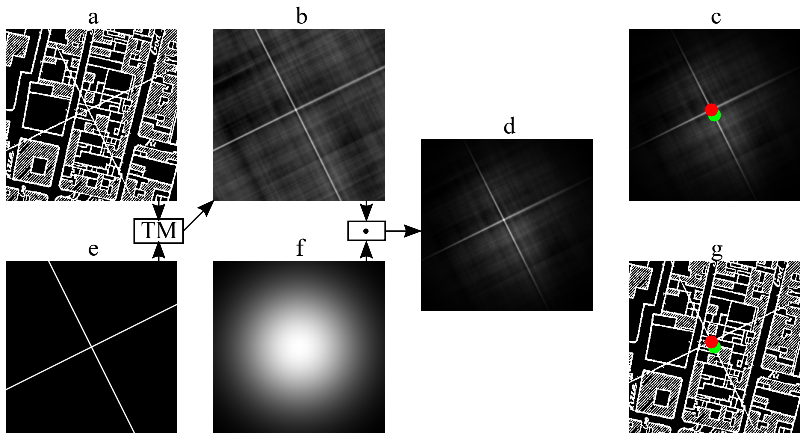


Figure 6.10: Graticule lines intersection refinement process: *(a)* cropped binarized input image, *(b)* cropped result of template matching, *(c)* red refinement of green estimated point using maximal value of *d*, *(d)* pixel-wise product of *b* and *f*, *(e)* template for template matching, *(f)* Gaussian hill with estimated point as center, *(g)* red refinement of green estimated point visualization.

Further steps include filtering, correcting, fixing, rating and filling the PPG. The rating is inspired by heart rate variability analysis [9]. The bigger amount and more stable intervals results in lower rating. The lower the rating, the better it is. As a result, we have several graticule candidates that are represented by the PPGs and contain information about their rating, angle and distance between the lines. Based on these, we can select the best candidate, detect intersection points and also refine them.

Intersections are determined using homogeneous coordinates [55]. It is straightforward if we represent lines in normal form and detect peaks with angle and distance coordinates. The presumption on straight lines is not usually satisfied so detected intersections are rather estimates of intersections and their localization can be improved as can be seen in Fig. 6.10. To do that, we use template matching where the template is a cross rotated accordingly to the grid (Fig. 6.10.e). The result of the template matching is multiplied with Gaussian hill to limit the refinement distance and also to approximate error distribution. The center of the Gaussian hill is positioned at the intersection estimate. Finally, the maximal value determines the position of refined intersection.

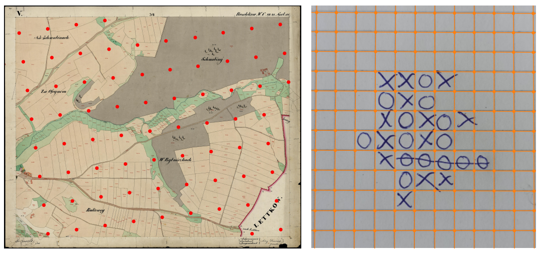


Figure 6.11: Grid detection example in different images.

The proposed method shows excellent results even on noisy historical map images. Moreover, it can be applied to general image as illustrated in Fig. 6.11. As can be seen in Fig. 6.12, our method surpassed the other methods (CMM [10], IRISA [3] and L3IRIS [10]) by a significant margin.

Rank	Team	Detection score (%) ↑
1	UWB	92.5
2	IRISA	89.2
3	CMM	86.6
4	L3IRIS	73.6

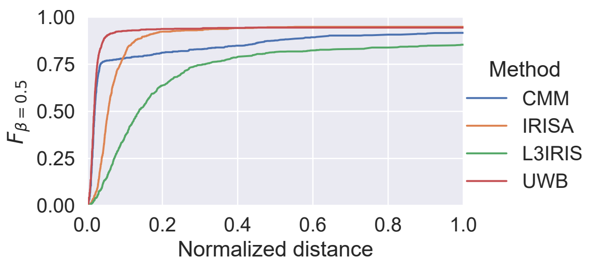


Figure 6.12: Final detection score (left) and results for each distance threshold (right) [10]

6.2.3 Historical Map Toponym Extraction

In [33], we deal with detection, classification and recognition of toponyms in hand-drawn historical cadastral maps. Toponyms are local names of towns, villages and landscape features such as rivers, forests etc. The detected and recognized toponyms are utilized as keywords in an information retrieval system that allows intelligent and efficient searching in historical map collections. We create a novel annotated dataset that is freely available for research and educational purposes. Then, we propose a novel approach for toponym classification based on KAZE descriptor. Next we compare and evaluate several state-of-the-art methods for text and object detection including FCN, YOLO or R-CNN. We further show the results of toponym text recognition using popular Tesseract [57] engine.

The overall pipeline is depicted in Fig. 6.13. Three blue boxes represent models for particular sub-tasks: text detection, toponym classification and OCR.

The proposed algorithm for toponym classification into printed and handwritten categories is inspired by the algorithm for writer identification based on image descriptors from [66] and it relies on KAZE [1] descriptor.

As described in Fig. 6.14, the first step of the proposed classification algorithm is a codebook generation. It is based on a training set with known labels. The KAZE detector is applied on all preprocessed regions. We thus obtain a set of key-points and corresponding descriptors for each region. All descriptors are put together and the resulting set of descriptor vectors is clustered with K-means algorithm.

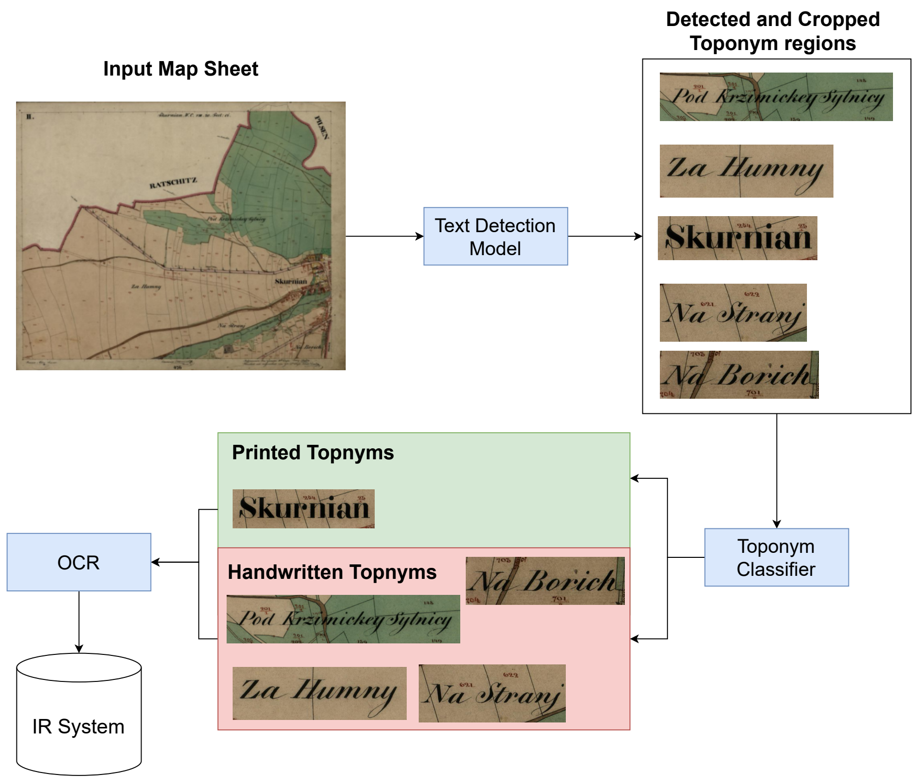


Figure 6.13: Historical map toponym extraction overall processing pipeline

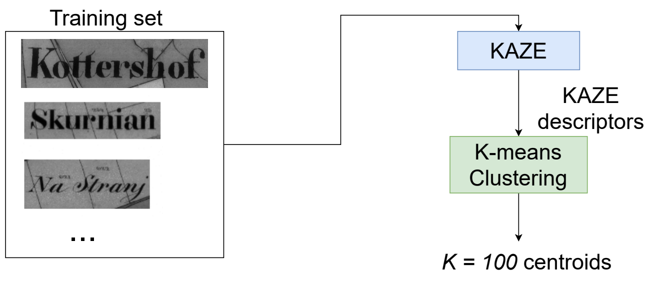


Figure 6.14: Codebook generation process

Image representation is calculated based on the codebook. For each descriptor vector, we find the closest cluster. The representation is then a histogram of size K where each bin represents how many times the given centroid was the closest to a descriptor vector. Finally, the histograms are size-normalized.

The prediction of an unknown text region is based on comparison of its representation with representations of known samples (training set). We first preprocess the image and obtain its histogram representation. Then, we find N most similar histograms from the training set using Bhattacharyya distance. Based on our experiments, this distance is more suitable for the comparison of histograms than other traditional distance measures. The predicted class is determined as the majority class occurring in the N most similar histograms. Fig 6.15 shows the prediction phase of our approach.

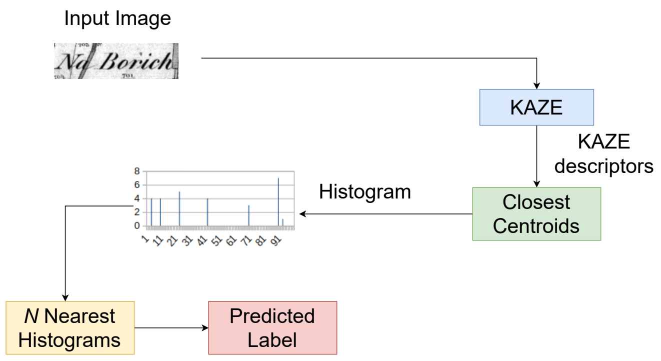


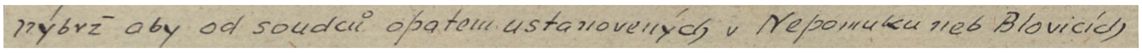
Figure 6.15: Test image prediction process

The biggest advantage of this algorithm is the fact that only a small amount of training examples is sufficient for reasonable results (comparing to the NN models).

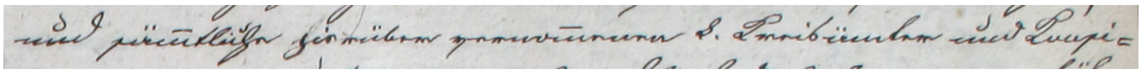
6.3 Multilingual HTR

This section presents preliminary results in the area of multilingual HTR. For the experiments, we utilized the model from [36] (see Sec. 3.3.2) and a private subset of approx. 9 thousand lines from 13 chronicles (10 pages from each) containing two languages, namely Czech and German.

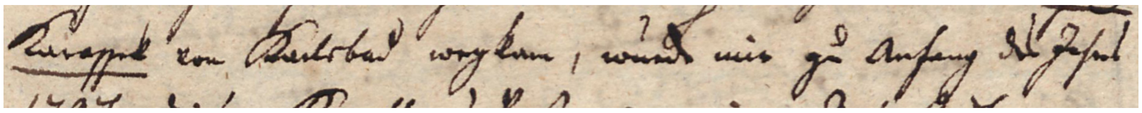
As illustrated in Fig. 6.16, the multilingual model trained in both languages and all chronicles produced surprisingly good results despite the high variety of writing styles. On the test part, we achieved 0.18 Character Error Rate (CER) in average. The results on separate chronicles ranged from 0.13 to 0.29 CER.



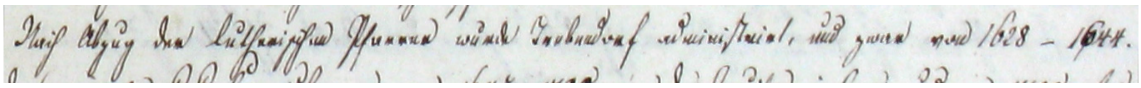
(a) Example from Blovice:
nýbrž aby od soudců opatem ustanovených v Nepomuku neb Blovicích (GT)
nýbrž aby od soudců apatem ustanovených v Nepomuku neb Blovicích (predicted)



(b) Example from Svojšín:
und sämmtliche hierüber vernommenen k. Kreisämter und Konsi- (GT)
zund sämmtliche hierüber vernwnenen k Kreisinter un d Jonsi- (predicted)



(c) Example from Karlovy Vary:
Karassek vom Karlsbad wegkam wurde mir zu Anfang des Jahres (GT)
Karesek von Karlsbad wergkann wurde mir zu Anfang des Juhnes (predicted)



(d) Example from Třebeň:
Nach Abzug der lutherischen Pfarrer wurde Trebendorf administrirt und zwar von 1628 1644 (GT)
dNach Azug der lutherischen Pfarer wurdl Trebendorf administrirt und zwar von 1628 Ex4r (predicted)

Figure 6.16: HTR examples

Chapter 7

Conclusions and Aims of the Doctoral Thesis

This work provides summary of beneficial and state-of-the-art approaches in the area of historical document analysis. It presents also our contribution in the area. Finally, the conclusions and aims of the doctoral thesis are discussed further in this chapter.

Based on the presented methods and their analysis, we can say that the analysis of modern printed documents works quite well. On the other hand, the analysis of historical documents is still challenging due to handwriting, different authors, scanning quality and also paper degradation. Moreover, the methods that work for modern documents usually do not work well for historical ones. Thus, there is a need for new approaches. These days, new approaches benefit from ML and are usually based on deep learning. An important limitation is a significant amount of annotated training data needed. At the same time, the annotated data are still rare and very time consuming to create and thus expensive. So there are two conflicting requirements on the amount of data.

Since we need to learn due to specificity of documents, it is beneficial to use ML approaches and it is problematic to replace them. On the other hand, the models could use easier learning objective and post-processing or they could be optimized for lower amount of data as was partially shown in our research (see Chapter 6). Another possibility to deal with the data is to make annotation easier utilizing current models and improving them iteratively with a growing amount of data.

There are languages that share common grammar and rules like Slavic or Germanic languages. Therefore, transfer learning and multilingual approaches look very promising especially for poorly resourced languages. An interesting opportunity lies in employing pre-trained models from related areas such as NLP for multi-modal processing. But there are problems like different historical language or not negligible character error rate in HTR. These problems should be carried out in order to fully utilize current progress in NLP.

Therefore, the aims of the doctoral thesis relate to document analysis and dealing with low amount of data. From the above mentioned, the aims are as follows:

1. Proposing new methods dealing with the possibilities of combining deep learning with conventional CV approaches to improve the results of individual methods for document analysis.

2. Analyzing the weaknesses of the current single-modal document analysis methods and proposing novel multi-modal approaches using pre-trained models and transfer learning to improve the final results.

3. Handling the issue of small amount of annotated data for model training by proposing novel document analysis methods dealing with weak supervision.

Bibliography

[1] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. Kaze features. In *Computer Vision – ECCV 2012*, pages 214–227, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[2] Mikhail Arkhipov, Maria Trofimova, Yuri Kuratov, and Alexey Sorokin. Tuning multilingual transformers for language-specific named entity recognition. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93, Florence, Italy, August 2019. Association for Computational Linguistics.

[3] Lemaitre Aurelie and Camillerapp Jean. Segmentation of historical maps without annotated data. In *The 6th International Workshop on Historical Document Imaging and Processing*, pages 19–24, 2021.

[4] Josef Baloun, Pavel Král, and Ladislav Lenc. Chronseg: Novel dataset for segmentation of handwritten historical chronicles. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pages 314–322. INSTICC, SciTePress, 2021.

[5] Serge Beucher. Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing.* CCETT, 1979.

[6] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[7] Roberto Brunelli. *Template matching techniques in computer vision: theory and practice.* John Wiley & Sons, 2009.

[8] Tomáš Brychcín. Linear transformations for cross-lingual semantic textual similarity. *Knowledge-Based Systems*, 187:104819, 2020.

[9] A John Camm, Marek Malik, J Thomas Bigger, Günter Breithardt, Sergio Cerutti, Richard J Cohen, Philippe Coumel, Ernest L Fallen, Harold L Kennedy, Robert E Kleiger, et al. Heart rate variability: standards of measurement, physiological interpretation and clinical use. task force of the european society of cardiology and the north american society of pacing and electrophysiology. *Circulation*, 1996.

[10] Joseph Chazalon, Edwin Carlinet, Yizi Chen, Julien Perret, Bertrand Duménieu, Clément Mallet, Thierry Géraud, Vincent Nguyen, Nam Nguyen, Josef Baloun, Ladislav Lenc, and Pavel Král. Icdar 2021 competition on historical map segmentation. In *Proceedings of the 16th International Conference on Document Analysis and Recognition (ICDAR'21)*, Lausanne, Switzerland, 2021.

[11] Huizhong Chen, Matthew Cooper, Dhiraj Joshi, and Bernd Girod. Multi-modal language models for lecture video retrieval. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1081–1084, 2014.

[12] Gobinda Chowdhary. *Natural language processing*, volume 37. Information Today, Inc., 2003.

[13] Christian Clausner, Stefan Pletschacher, and Apostolos Antonacopoulos. Aletheia-an advanced document layout and text ground-truthing system for production environments. In *2011 International Conference on Document Analysis and Recognition*, pages 48–52. IEEE, 2011.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[15] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[16] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[17] Javier Ferrando, Juan Luis Domínguez, Jordi Torres, Raúl García, David García, Daniel Garrido, Jordi Cortada, and Mateo Valero. Improving accuracy and speeding up document image classification through parallel systems. In *International Conference on Computational Science*, pages 387–400. Springer, 2020.

[18] Adrian Frutiger. Ocr-b: A standardized character for optical recognition. *The Journal of Typographic Research*, 1:137–146, 1967.

[19] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[20] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[22] 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.

[23] Tobias Grüning, Gundram Leifert, Tobias Strauß, Johannes Michael, and Roger Labahn. A two-stage method for text line detection in historical documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 22(3):285–302, 2019.

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[26] 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*, pages 770–778, 2016.

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

[28] Lei Kang, Pau Riba, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *arXiv preprint arXiv:2005.13044*, 2020.

[29] Frédéric Kaplan, Sofia Ares Oliveira, Simon Clematide, Maud Ehrmann, and Raphaël Barman. Combining visual and textual features for semantic segmentation of historical newspapers. *Journal of Data Mining & Digital Humanities*, 2021.

[30] K. Kise, O. Yanagida, and S. Takamatsu. Page segmentation based on thinning of background. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 3, pages 788–792 vol.3, 1996.

[31] Koichi Kise. *Page Segmentation Techniques in Document Analysis*, pages 135–175. Springer London, London, 2014.

[32] Shivani Kumar, Atharva Kulkarni, Md Shad Akhtar, and Tanmoy Chakraborty. When did you become so smart, oh wise one?! sarcasm explanation in multi-modal multi-party dialogues. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5956–5968, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[33] Ladislav Lenc, Jiří Martínek, Josef Baloun, Martin Prantl, and Pavel Král. Historical map toponym extraction for efficient information retrieval. In *International Workshop on Document Analysis Systems*, pages 171–183. Springer, 2022.

[34] Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, 2021.

[35] Peizhao Li, Jiuxiang Gu, Jason Kuen, Vlad I Morariu, Handong Zhao, Rajiv Jain, Varun Manjunatha, and Hongfu Liu. Selfdoc: Self-supervised document representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5652–5660, 2021.

[36] J. Martínek, L. Lenc, and P. Král. Building an efficient OCR system for historical documents with little training data. *Neural Computing and Applications*, pages 1–19, 2020. Received: 25 December 2019, Accepted: 06 April 2020, Published: 09 May 2020.

[37] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[38] Seungwhan Moon, Leonardo Neves, and Vitor Carvalho. Multimodal named entity recognition for short social media posts. *arXiv preprint arXiv:1802.07862*, 2018.

[39] Shunji Mori, Ching Y Suen, and Kazuhiko Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80(7):1029–1058, 1992.

[40] Bastien Moysset, Christopher Kermorvant, Christian Wolf, and Jérôme Louradour. Paragraph text segmentation into lines with recurrent neural networks. In *2015 13th international conference on document analysis and recognition (ICDAR)*, pages 456–460. IEEE, 2015.

[41] Oliver Nina, Bryan Morse, and William Barrett. A recursive otsu thresholding method for scanned document binarization. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 307–314. IEEE, 2011.

[42] Lawrence O'Gorman and Rangachar Kasturi. *Document image analysis*, volume 39. Citeseer, 1995.

[43] Sofia Ares Oliveira, Benoit Seguin, and Frederic Kaplan. dhsegment: A generic deep-learning approach for document segmentation. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 7–12. IEEE, 2018.

[44] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[45] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[46] James P Philips and Nasseh Tabrizi. Historical document processing: A survey of techniques, tools, and trends. *arXiv preprint arXiv:2002.06300*, 2020.

[47] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[48] Hiranmayi Ranganathan, Shayok Chakraborty, and Sethuraman Panchanathan. Multimodal emotion recognition using deep learning architectures. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[49] 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.

[50] 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*, 28:91–99, 2015.

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[52] Ahmed Cheikh Rouhou, Marwa Dhiaf, Yousri Kessentini, and Sinda Ben Salem. Transformer-based approach for joint handwriting and named entity recognition in historical document. *Pattern Recognition Letters*, 155:128–134, 2022.

[53] Jaakko Sauvola, Tapio Seppanen, Sami Haapakoski, and Matti Pietikainen. Adaptive document binarization. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 1, pages 147–152. IEEE, 1997.

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

[55] Václav Skala. Intersection computation in projective space using homogeneous coordinates. *International Journal of Image and Graphics*, 8(04):615–628, 2008.

[56] Kamilya Smagulova and Alex Pappachen James. A survey on lstm memristive neural network architectures and applications. *The European Physical Journal Special Topics*, 228(10):2313–2324, 2019.

[57] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[58] Sebastian Sudholt and Gernot A Fink. Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 277–282. IEEE, 2016.

[59] Zhongkai Sun, Prathusha K Sarma, William Sethares, and Erik P Bucy. Multimodal sentiment analysis using deep canonical correlation analysis. *arXiv preprint arXiv:1907.08696*, 2019.

[60] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[61] Chris Tensmeyer and Tony Martinez. Document image binarization with fully convolutional neural networks. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 99–104. IEEE, 2017.

[62] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[64] Christoph Wick and Frank Puppe. Fully convolutional neural networks for page segmentation of historical document images. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 287–292. IEEE, 2018.

[65] Christoph Wick, Jochen Zöllner, and Tobias Grüning. Transformer for handwritten text recognition using bidirectional post-decoding. In *International Conference on Document Analysis and Recognition*, pages 112–126. Springer, 2021.

[66] Yu-Jie Xiong, Ying Wen, Patrick S P Wang, and Yue Lu. Text-independent writer identification using sift descriptor and contour-directional feature. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 91–95, 2015.

[67] Baosong Yang, Jian Li, Derek F Wong, Lidia S Chao, Xing Wang, and Zhaopeng Tu. Context-aware self-attention networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 387–394, 2019.