

Detecting signatures in scanned document images

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Marcel René Hauri, BSc.

Matrikelnummer 01355940

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Allan Hanbury

Wien, 10. Jänner 2020

Marcel René Hauri

Allan Hanbury



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



Detecting Signatures in scanned document images

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Marcel René Hauri, BSc.

Registration Number 01355940

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Allan Hanbury

Vienna, 10th January, 2020

Marcel René Hauri

Allan Hanbury



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Marcel René Hauri, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. Jänner 2020

Marcel René Hauri



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to thank my advisor Professor Allan Hanbury for the opportunity to write my master thesis in collaboration with the City of Vienna within the scope of the BRISE project, for continuously providing me with feedback and his great support over the course of this thesis. I would also like to thank Bernd Pinter from the City of Vienna for his constructive feedback and for the good collaboration.

Last but not least, I would like to thank my parents and my girlfriend, Sandra, their encouragement and support kept me focused on writing.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Ziel dieser Arbeit ist es Unterschriften und Unterschriftfeldern in Bauanträgen der Stadt Wien zu erkennen und zu aggregieren um diese auf Vollständigkeit zu überprüfen, eine Serienverarbeitung von Dokumenten soll möglich sein. Eine Reihe von maschinellen Lernmodellen wurde trainiert und evaluiert um einen robusten und schnellen Algorithmus zu finden. Zur Gewährleistung der Reproduzierbarkeit und Vergleichbarkeit wurde der frei verfügbare Tobacco 800 Datensatz initial zum Training verwendet. Der Datensatz beinhaltet eine Vielzahl von Dokumenten, die von Tabakproduzenten im Rahmen des Master Settlement Agreements freigegeben wurden. Jedes Dokument setzt sich aus maschinell gedrucktem Text, Unterschriften und handgeschriebenen Notizen zusammen und jede visuelle Entität ist annotiert. Aufgrund des vergleichbaren Aufbaus von Bauanträgen und Dokumenten aus dem Tobacco 800 Datensatz, eignet sich dieser als initiales Substitut. Die Stadt Wien stellt Bauanträge nur als Rohdaten zur Verfügung, ohne weitere Annotationen von visuellen Entitäten. Um einen zeitaufwändigen manuellen Annotationsprozess zu vermeiden, wurde eine Transfer Learning Pipeline implementiert. Dabei wurde das, auf Basis des Tobacco 800 Datensatz trainierte, Modell verwendet um Unterschriften im Datensatz der Stadt Wien zu detektieren. Die erkannten Bounding Boxes der Unterschriften wurden danach manuell evaluiert und verfeinert um schnell den neuen Datensatz zu annotieren. Im letzten Schritt wurde das Modell verwendet um einen Prototyp zu implementieren, der es Benutzern ohne fundierte technische Kenntnisse ermöglicht schnell Unterschriften aus Dokumenten zu aggregieren, um die Vollständigkeit von Dokumenten in Bezug auf die benötigten Unterschriften zu überprüfen.



EUROPÄISCHE UNION
Europäischer Fonds für regionale Entwicklung

Das Projekt wird aus Mitteln des Europäischen Fonds für regionale Entwicklung
im Rahmen der Urban Innovative Actions Initiative kofinanziert.





Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

The goal of this thesis is to provide a means for detecting and aggregating signatures and signature fields to check for completeness of building applications of the City of Vienna in bulk, i.e. it should be possible to process sets of documents. In order to find a robust algorithm, several machine learning models were trained and evaluated. For reproducibility and comparability with published results, the freely available Tobacco 800 data set was used initially for training. This data set is composed of a variety of documents released by tobacco companies under the Master Settlement Agreement. Each document image contains machine printed text, signatures and handwritten notes, whereas the position of each visual entity is annotated. Therefore, the Tobacco 800 data set is suitable as an initial substitute for the building applications data set from the City of Vienna. The City of Vienna only supplies raw scans of building applications without any annotations of visual entities. To avoid a tedious and time-consuming completely manual annotation process, a transfer learning pipeline was established, where the model trained with the Tobacco 800 data set was applied to the building applications data set from the City of Vienna. The model trained on the Tobacco 800 data set was used to accelerate the manual annotation process of the building applications data set. It was employed to predict signatures in the building applications data set, the predicted bounding boxes were then manually refined. Finally, this model was used to implement a prototype, accessible to users without technical knowledge, that facilitates the rapid aggregation of signatures and assessment of completeness with regard to required signatures for building applications in bulk.



This project is co-financed by the European Regional Development Fund through the Urban Innovative Actions Initiative.





Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Goal and research questions	3
1.2 Scope	4
1.3 Structure of the Thesis	4
2 Background	5
2.1 Computer Vision	5
2.2 Basic image analysis techniques	7
2.3 Machine Learning	10
2.4 Convolutional Neural Networks (CNN)	15
2.5 PyTorch	21
2.6 Performance metrics	21
2.7 Data augmentation	23
2.8 GIoU as a loss function	23
2.9 COCO data set	23
2.10 Summary	23
3 Related Work	25
3.1 Feature extraction based approaches	25
3.2 Deep learning based approaches	31
3.3 Conclusion	34
4 Experiment Structure	35
4.1 Data sets	35
4.2 Experiment setup Torchvision	40
4.3 Experiment setup Detectron	41
4.4 Experiment setup YOLO	44
4.5 Hardware	45
	xiii

5	Experiment Results	47
5.1	Experiments on Tobacco 800	47
5.2	Experiments on City of Vienna data set	60
6	Conclusion	73
6.1	Research Questions	73
6.2	Contributions	74
6.3	Limitations	75
6.4	Future Work	75
A	Appendix	77
A.1	Training, Validation and Testing split of Tobacco 800	77
A.2	Implementation of Signature Detection System	80
	List of Figures	83
	List of Tables	85
	Bibliography	87

Introduction

The City of Vienna handles numerous building applications every year. Proprietors, eager to build, have to pass a formal building approval process at the responsible municipality. Part of this process is the building application, which is required to gain a building permit. It consists of building plans, building descriptions, evidence of property ownership, static calculations, the energy performance certificate and various other documents depending on the type of structure. Each of these documents is composed of several pages containing an assortment of paragraphs of machine-printed text concerning the legal requirements for a specific type of building, often complemented by handwritten notes. Building applications have to be physically signed by a variety of legal entities (e.g. proprietor, stakeholders) to be valid. After receiving a building application, the City of Vienna has to perform a check for completeness, i.e. asserting that all required signatures are present, before legally verifying the request. The civil servants of the municipality MA 37 (Baupolizei) are responsible for examining the documents. As applications for bigger projects can easily amount to hundreds of pages full of paragraphs and signatures being spread out through the whole document, checking for completeness and aggregating all involved legal entities is a tedious and error-prone task. The BRISE Vienna (Building Regulations Information for Submission Involvement) project is a research and development project funded by the EU initiative Urban Innovative Actions. Its goal is the digitization of the administrative building approval process, in which about 13000 building submissions are analyzed yearly. Automating the process of extracting signatures of documents is in the project's scope so that civil servants do not have to open each scanned document manually. Experts are then able to see on a report, which signatures are present in which document. A variety of challenges has to be overcome, starting with the great diversity of documents, ranging from pre-defined forms over handwritten letters up to plans exported from CAD software and signed by the proprietor or architect. Another difficulty is posed by stamps frequently overlapping signatures, which render the detection of signatures more difficult. Figure 1.1 shows the challenges of signature detection based on, for privacy reasons, an

1. INTRODUCTION

artificially created document, mimicking a document of the building application approval process.

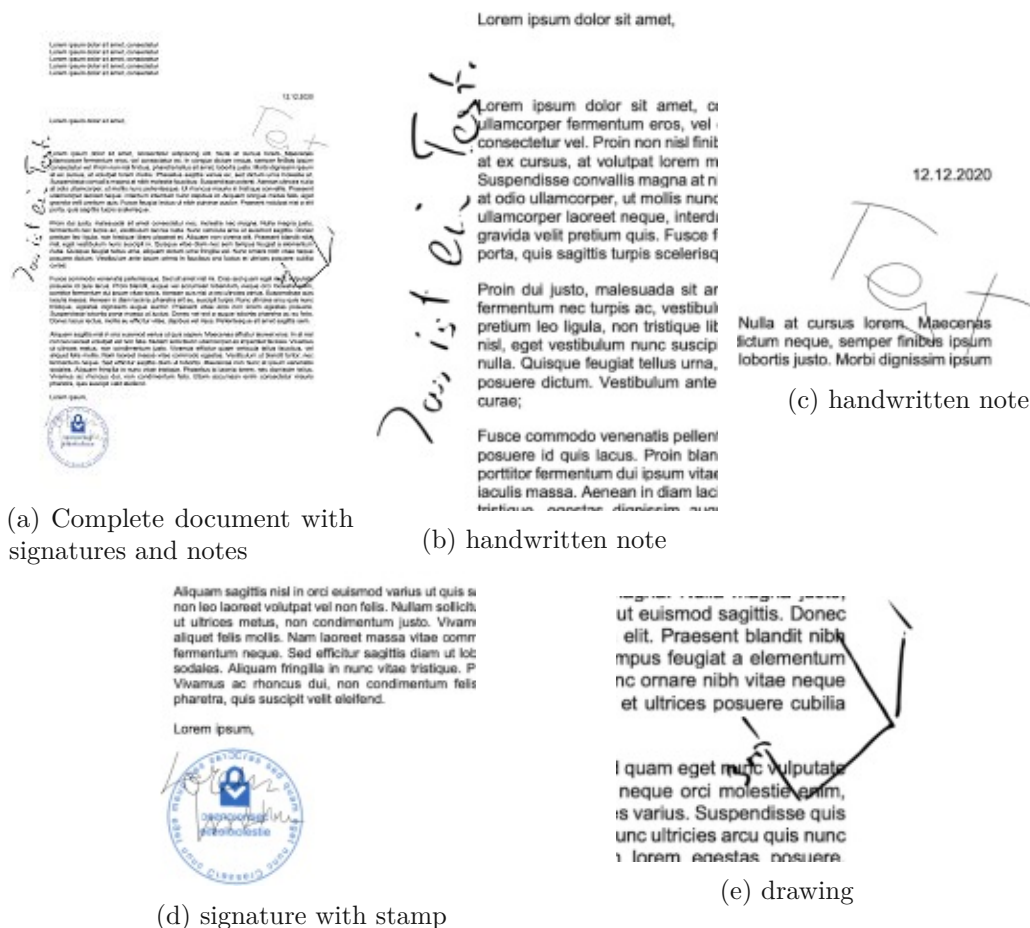


Figure 1.1: Challenges of signature detection (artificially created document)

From a computer vision perspective, the problem of identifying signatures in documents can be addressed as the detection of free-form objects from varying backgrounds [62]. As building applications do not contain only signatures, but other types of free-form objects, e.g. handwritten notes from the applicant, the detection process is more challenging than simply identifying free-form objects on machine-printed text. Mandal et al. [32] mention that interpreting signatures and machine printed text as two separate layers is difficult in many cases due to overlapping elements, which have to be clearly identified before performing detection. Furthermore, Zhu et al. [62] describe the problem of overfitting, as free-form objects are subject to large intra-class variation. Zheng et al. [60] address the problem of noise in scanned document images, which occurs in the scanning

process. Noise is described as blocks of black pixels not found in the original image but in the digital image, i.e. artefacts produced by document scanners. It is stated that in detecting signatures based on joint features, noise can interfere with model performance. To overcome this issue noise is treated as a separate prediction class (in addition to signatures). Finally, signatures have to be differentiated from other handwritten on the page.

1.1 Goal and research questions

The goal of this thesis is to provide a tool for detecting and aggregating signatures to check for completeness of building applications of the City of Vienna in bulk, i.e. it should be possible to process sets of documents. In order to find a robust algorithm, several machine learning models were trained with different parameter settings and data set splits. Afterwards, a standardized testing and evaluation procedure with objective performance metrics was used to find the best performing model. This model was then employed to build a prototype that allows users without technical knowledge in regard to neural networks to detect signatures in documents.

1.1.1 Research Questions

- RQ1: What is the best approach to detect handwritten signatures from varying, noisy backgrounds with machine learning?
- RQ2: What effect has transfer-learning on model accuracy in handwritten signature detection?
- RQ3: How can human annotators be supported with transfer learning?

1.1.2 Contributions

This thesis contributes a diverse data set of building application documents, annotated with bounding boxes of signatures, in a standardized format (YOLO darknet format). The data set can be used to train object detection models and evaluate their performance. Due to privacy, this data set cannot be released to the public, but it can be used by staff of the City of Vienna to train new object detection models, released in the future. Furthermore, as the Tobacco 800 data set was used initially for training, a new baseline of performance for the data set of, in 2020, state of the art object detection approaches is given. By providing a detailed list of files used for training, evaluation and testing, which to the author's knowledge has not been released by any publication analyzing the data set, allows others to reproduce and compare performance of the proposed approaches in detail. In addition, a detailed evaluation of performance of different networks, parameters used for training, as well as the specific network architecture are stated, to provide the interested reader with information about the best approach for signature detection in documents. Finally, a convolutional neural network model, which facilitates the prediction

of signatures in scanned document images, is provided and integrated into a prototype enabling the users without technical knowledge in regard of neural networks to utilize the model.

1.2 Scope

The goal is to provide a prototypical implementation serving as a proof of concept, rather than developing and integrating a production-ready signature detection system. For the purposes of the City of Vienna, detection but no segmentation or verification of signatures is sufficient. As the amount of data currently available for training and testing the object detection model is limited, its performance is potentially affected as well. If over time a greater number of document instances can be gathered, expanding the data set and retraining the proposed object detection model might result in a better validation score.

1.3 Structure of the Thesis

In the current chapter, Chapter 1 of this thesis, the building application approval process at the City of Vienna, the occurring documents as well as its issues and inefficiencies were introduced. These issues were subsequently mapped to the computer vision problem of object detection and research questions and goals presented. Secondly, background knowledge, required to understand related work, is stated in Chapter 2. Besides explaining deep learning concepts, this section contains feature extraction methodologies, due to being the approach of choice in many signature detection publications. Thirdly, related work in the area of signature detection is listed and aggregated in Chapter 3, to define acquire a state of the art, the thesis can build on. Next, the two analyzed data sets, Tobacco 800 and building applications from the City of Vienna in addition to different experiment setups for deep learning-based signature detection are introduced in Chapter 4. Afterwards, the results of the experiments with different parameter settings and data set splits are displayed and discussed in Chapter 5. Based on the best performing model, a prototypical signature detection pipeline was implemented facilitating extraction of signatures from documents in bulk, it is described in Chapter A.2 in the appendix. Finally, the presented approaches and achievements are summarized and future research possibilities shown in Chapter 6.

Background

The foundational concepts and theoretical background of computer vision with focus on object detection are introduced. The discussed topics range from traditional feature extraction based approaches, which are the method of choice in most signature detection publications and therefore required to understand related work, to state of the art machine- and deep learning algorithms. Furthermore, basic edge detection and clustering algorithms as well as the concept of convolution are introduced. Feature extraction based object detectors rely on these concepts and techniques. Moreover, convolutional neural networks (YOLO and Faster R-CNN), which are used for signature detection in the thesis, are introduced. Finally, metrics used to assess the performance of neural networks used for objection detection are stated.

2.1 Computer Vision

The research area of computer vision is a subfield of artificial intelligence and machine learning and focuses on enabling machines to "see" and interpret still images and videos like a human. While most problems of computer vision seem trivial to humans, they are demanding tasks for a computer program [12]. Computer vision is applied in a variety of fields, e.g. autonomous navigation of robots and vehicles, object detection and tracing, augmented reality and medicine. Due to the broad spectrum of problems, a wide array of methodologies ranging from statistical and probabilistic models over computer graphics to artificial intelligence, are used. The common goal is to distill information of varying types from digital images [50]. The earliest experiments in computer vision date back to the 1960s where edge detection was used to understand the content of images, in the 1980s image pyramids and the concept of scale-space were popular topics of research, which build the foundation of methodologies such as Scale Invariant Feature Transform (SIFT) [54]. Common Tasks in computer vision are detection of objects, i.e. locating objects in an image, classification of objects, i.e. determining the kind of object displayed

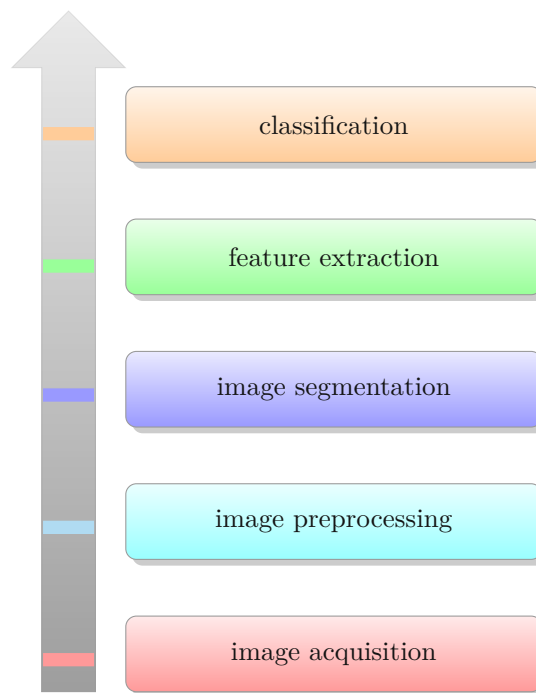


Figure 2.1: Image recognition pipeline

in an image, and segmentation of objects, i.e. how pixels can be grouped based on coherence in an image.

2.1.1 Architecture of computer vision systems

From a technical point of view, Jiang [23] defines that computer vision systems can be typically divided into two building blocks, image acquisition, often performed by scanners or cameras, and image interpretation. Image interpretation is heavily dependent on pattern recognition, where a pattern is defined as any form of repeating information that needs to be analyzed. A wide variety of tasks, e.g. analysis of x-rays or MRI images in the health sector, object detection, pose recognition and even biometrical authentication systems build upon image recognition. Most image recognition systems follow the structure outlined in Figure 2.1.

During the acquisition phase of a digital image recognition system a scene is captured and digitized. In the preprocessing step, artefacts like noise are eliminated by filtering and the image is optimized for the subsequent stages, e.g. by improving contrast. Image segmentation refers to the merging of neighbouring and coherent parts of the image into groups. During the feature extraction phase, dimensionality is reduced, i.e. a smaller set of numerical values characterizing the analyzed entity is distilled from the data. Finally, the extracted features are used to classify the entity [54]. Modern deep learning based object detection approaches fuse many of these steps. Knowledge about these

basic concepts is, therefore, required for understanding the filtering functions of layers in convolutional neural networks. After training, layers model feature extraction functions, e.g. initial convolutional layers tend to perform basic feature extraction tasks like edge detection.

2.2 Basic image analysis techniques

Traditional feature extraction in terms of computer vision refers to the extraction of features from images. Features should be robust in terms of small changes in the image such as translation, rotation or varying color and brightness. SIFT was selected as a representation for advanced, generalized feature extraction algorithms, used in related work [34]. Image segmentation algorithms aim at reducing the pixels in an image to groups for easier analysis [54]. Borders like edges of objects can be used as markers of segmentation for grouping pixels. Image segmentation algorithms are typically divided into three groups: pixel-, edge- and region-based approaches. The output of such an algorithm is an array of segments for the image, where each segment contains similar pixels, based on a criterion. Despite the thesis focusing on object detection, knowledge about basic image analysis techniques is required for understanding related work in the area of feature extraction based signature detection algorithms, where mostly segmentation is performed as well. The list of image analysis algorithms is not exhaustive, but is composed of selected examples that allow the reader to understand related work in subsequent chapters. Explaining all in related work used feature extraction methodologies in depth is considered out of scope for the thesis, as it is focused on deep learning based approaches. Furthermore, the mathematical concept of convolution is introduced, it is not only used in feature extraction and image processing but convolutional neural networks as well.

2.2.1 Thresholding

In this very basic approach, a greyscale image is turned into a binary image by a threshold. Madasu et al. [31] use thresholding as a preprocessing step in a signature detection pipeline. Every pixel above the threshold is converted to white, every pixel below is set to black. The key element of this method is a proper choice of the threshold. It can be extended by multiple thresholds, resulting in a finer granularized image. Several algorithms exist for finding thresholds, e.g. Otsu's method and Balanced histogram thresholding. Otsu's method [38] focuses towards minimizing intra-class variances. First the histogram of the image is calculated, then the intra-class variance is minimized. The following formula specifies intra-class variance: $\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t)$, $w_1(t)$ and $w_2(t)$ refer to the class probabilities under threshold t . These are calculated by $w_1(t) = \sum_{i=1}^t P(i)$ and $w_2(t) = \sum_{i=t+1}^I P(i)$. By minimizing the function, an optimal threshold can be determined. An Otsu's method based binarization approach is employed by Mandal et al. [32] for preprocessing in a signature detection system.

2.2.2 Clustering

Clustering algorithms try to group pixels in an image by common characteristics, e.g. colour, coordinates and textures. From a mathematical perspective clustering can be defined as follows, considering an image I composed of M elements $I = x_1, x_2, x_3, \dots, x_M$, I can be divided in a series of not overlapping subsets I_j , $I = \bigcup_{j=1}^k I_j$ where $I_i \cap I_j = \emptyset \wedge \forall i \neq j$. Each element of I_j shares a common characteristic [12][55]. The most common clustering algorithms are K-Means, Gaussian Mixture Model and Mean-Shift-Clustering [22]. In K-Means-Clustering first the number of clusters (K) that the data should be grouped into is selected [6]. Then K random tuples are selected to form the initial clusters, afterwards for every remaining data point the distance to each of the K clusters is calculated, subsequently the tuple is added to the nearest cluster. Next the mean element of each cluster will be calculated, the remaining elements are then clustered again based on the mean element. This process is repeated until the clusters do not change further when the mean element is selected. As the K initial tuples were selected randomly, this process has to be repeated with different initial values to find the optimal clusters. With respect to the total number of types and number of clusters K , quality can be measured by inner cluster variation. Nandedkar et al. [37] use K-Means-Clustering in a document retrieval pipeline that detects signatures and logos in scanned document images.

Convolution

Convolution, noted as $a * b$, is a frequently used mathematical operation in computer vision and signal processing in general. Due to mostly discrete functions in computer vision, it usually occurs in discrete form. Broughton and Bryan [9] define discrete convolution formally by: Let $a, b : D \rightarrow \mathbb{C}$ be functions with a discrete domain $D \subseteq \mathbb{Z}$, then the following holds: $(a * b)(n) = \sum_{k \in D} a(k)b(n - k)$. The domain of summation is the domain of a and b . Domain refers to the mathematical domain, which indicates for which input values of a function corresponding output values are defined. In case of finite domains (aperiodic function) a and b are usually padded with zeros and interpreted as vectors[40]: $\vec{a} \in \mathbb{C}^{n_a}$ and $\vec{b} \in \mathbb{C}^{n_b}$, which allows more efficient computation by matrix vector multiplication: $(a * b)(n) = \mathbf{B}\vec{a}$

$$\mathbf{B} = \begin{bmatrix} \vec{b} & 0 & 0 & \dots & 0 \\ 0 & \vec{b} & 0 & \dots & 0 \\ \vdots & 0 & \vec{b} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \vec{b} \end{bmatrix}$$

$$\mathbf{B} \in m \times n_a$$

$$m = n_a + n_b - 1$$

Edge detection

Edge detection algorithms are based on the close relation between regional boundaries and edges. Assuming function f specifying the intensity of each pixel of an image via $f(x)$, then the first discrete derivative of f , $f'(x)$ often shows local maxima at borders of regions due to dramatic changes in intensity. Two often used edge detection algorithms are Sobel and Canny edge detection, Canny edge detection is based on Sobel. Gimel'Farb and Delmas [13] describe Sobel edge detection as an algorithm that computes the gradient of intensity for each pixel and finds the orientation of most dramatic intensity changes between pixels. After this step every pixel has been assigned a probability of how likely it is part of an edge, including the orientation of the edge. The following kernels are used to perform the Sobel transformation on the image I .

$$B_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * I \quad B_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * I$$

B_x represents horizontal changes in intensity, i.e. their derivatives, and B_y represents vertical respectively. After the derivatives have been calculated, their values are merged via $B = \sqrt{B_x^2 + B_y^2}$ and the direction of the edge is then computed by $\Theta = \arctan\left(\frac{B_y}{B_x}\right)$. Zhu et al. [62] uses Canny edge detection in a Multi-scale Structural Saliency for Signature Detection approach.

2.2.3 Scale Invariant Feature Transform (SIFT)

Lowe [30] introduces the feature extraction algorithm SIFT. It is translation, scaling, rotation and luminance invariant thus allowing the robust extraction of features. The algorithm is used in [34] to detect handwriting in scanned document images. In Figure 2.2 the main steps of the algorithm are outlined. First of all potential features are collected by employing image pyramids. A Gaussian filter kernel is applied via convolution multiple times with increasing strength [46], strength is defined by standard deviation σ . Afterwards the image is shrunk by the factor 2. Then Gaussian blur is applied again multiple times, followed by downsampling. This process is repeated usually four times, a stage is called octave, each with five applications of Gaussian blur with different standard deviation. The resulting image of this stage is called scale space, its purpose is to attenuate smaller structures. The next step is to calculate the difference between Gaussian blurred images within each octave. The extrema found are then marked, where extrema with low absolute values are suppressed. Then, a direction is assigned to each extremum by determining the edge, it is located on. This is performed by analyzing the gradients of pixels, surrounding the extremum. During this step all points without a distinct orientation are truncated. Finally, descriptors for all extrema are calculated, by gathering histogram data for the surroundings of each point. The resulting descriptors are robust to most changes in the image, due to being computed via image pyramids, with varying levels of blur and resolution. Mandal et al. [32] uses SIFT features for signature-based document retrieval.

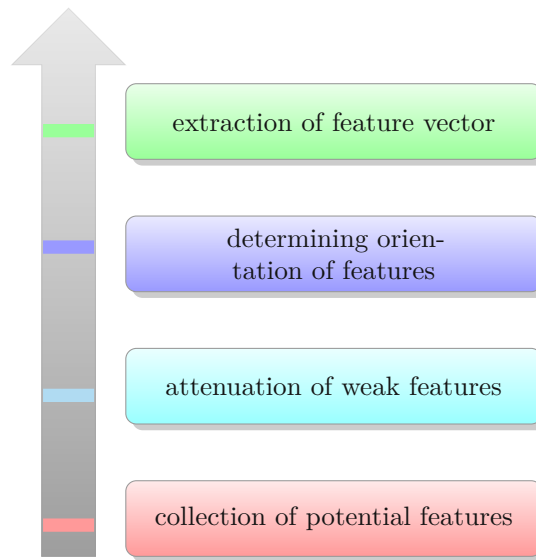


Figure 2.2: SIFT pipeline

2.3 Machine Learning

In machine learning, generalized algorithms are employed in order to extract information from a data set without customizing the algorithm to be suitable to the data set [5]. Alignment of the algorithm towards the problem rather happens via learning, i.e. the algorithm is trained with tuples from the data set and adapts autonomously accordingly. In general lots of data is necessary to facilitate good adaption. In general, machine learning algorithms can be divided into two groups following different paradigms: supervised learning and unsupervised learning.

2.3.1 Supervised Learning

During training, supervised learning algorithms need to be fed with the correct output values for each tuple of input they are trained on [5]. A tuple in machine learning is a vector of fixed size, i.e. an ordered, fixed length series of values. In neural network based machine learning approaches the tuple defines the size and structure of network's input layer. The necessary customization of the algorithm's internal logic, to convert input tuples to the correct output values is determined by the learning algorithm. Oversimplified, a machine learning algorithm processes and manipulates input values with a set of numbers, called weights, to compute the output values. Input and output values are known while training, therefore weights can be adjusted to achieve optimal output values for the training samples. This contrasts dramatically with traditional approaches, where human experts would define rules for inference in a tedious and time consuming process [5]. The general machine learning pipeline is shown in Figure 2.3

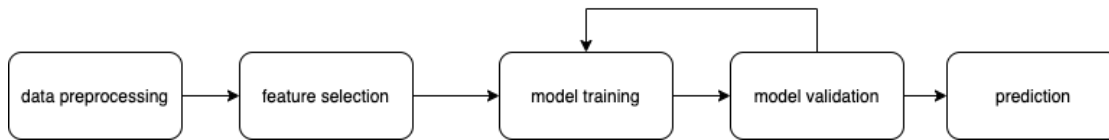


Figure 2.3: Machine learning pipeline

2.3.2 Unsupervised learning

Unsupervised learning in contrast to supervised learning, uses algorithms that process input data without a supervisor specifying expected output values for the input data [5]. These algorithms are used to find hidden patterns, relationships and clusters in the input data set. A frequently used algorithm is the self-organizing map (SOM) [26]. This dimensionality reduction technique processes the input space of data samples into a two dimensional map. Therefore, SOMs are very useful visualization techniques for high dimensional data.

2.3.3 Artificial Neural Networks

Artificial neural networks are modelled on their real counterparts, neurons in the nervous system of human beings. Artificial neural networks [20] are built from several layers, each of which contains one to many nodes. The layers between input and output layer are called hidden layers [20]. Nodes compute values, that are passed to subsequent nodes in the network. Despite research and theoretical concepts of deep learning dating back to the 1960s, decades had to pass until recent advances in computing power made it possible to turn theory into practice. Neurons in deep learning are mimicked after neurons in human brains, this idea originates from the McCulloch-Pitts Model [35], where McCulloch and Pitts introduce the relationship of neurons in brains with transistor gate logic with binary output. In the brain dendrites receive signals which are passed to the neuron, where they are gathered. Upon surpassing a specific level, an output signal is transferred to the axons [41]. Figure 2.4 illustrates the signal flow in biological neurons.

Artificial neurons accept one to many input values from the data set or other neurons from previous layers in the network. After they calculate a value, it is passed further in the network. Connections between inputs, outputs and neurons are called synapses. Synapses are associated with weights, that represent the previous neurons significance in relation to the complete network. The calculation of values in neurons happens as follows:

1. Neurons sum up the products of inputs and weights of the previous layers neurons.
2. Depending on the architecture, sometimes an additional bias is used to further manipulate this sum.
3. Afterwards an activation function is used to process the sum and create the input for the next layer.

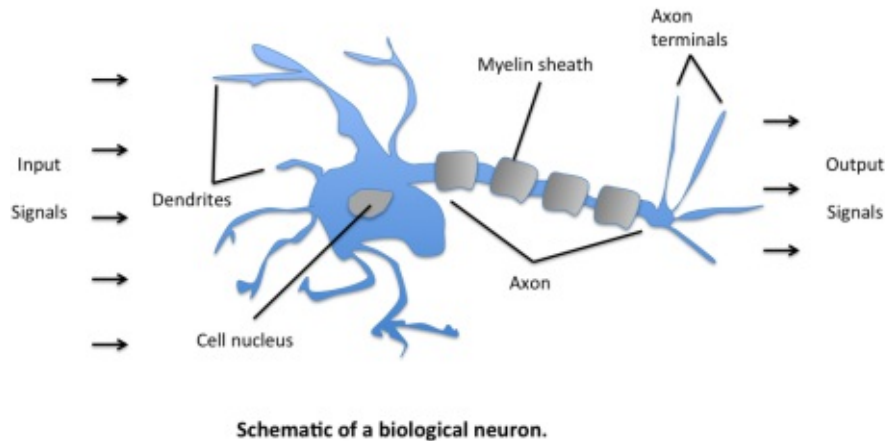


Figure 2.4: Biological neuron (Raschka and Mirjalili [41])

During training, the expected results of tuples fed into the network are compared with the actual results, these are aggregated into a cost function. Minimizing the cost function via gradient descent or other optimization algorithms and by adapting weights and biases by backpropagation is how neural networks learn.

2.3.4 Artificial neuron

In Figure 2.5 the structure of an artificial neuron used in modern neural networks is shown. In the forward pass of the network, a neuron processes input x specified via the values x_1 to x_n as follows. First, each element of input x is multiplied by the corresponding weight from the set of weights (w_1 to w_n), then these products and the bias are summed up. This result is subsequently processed by the activation function. The bias can therefore be seen as a means of shifting the activation function $f(x)$ on the x-axis. During training the output of a tuple is calculated and compared to the target value. The weight update is calculated by $\Delta w_j = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}$ [41].

Learning process

Widrow and Lehr [56] introduce Adaptive Linear Neurons (Adaline) and the Widrow-Hoff rule, which employ a linear activation function to update weights. This allows the usage of gradient descent to update weights. Tuples, for which the expected output is known, are passed through the neural network. Via a cost function $J(w)$, e.g. sum of squared errors, the difference between expected and actual outputs is measured over all samples, passed through the network in one training round (epoch). Batch size refers to the number of samples fed to the network to in one iteration. The number of iterations required for one epoch is defined by the following relation: $\text{iterations} = \frac{\text{samples in data set}}{\text{batch size}}$. Provided a continuous activation function is used in the network, the cost function can

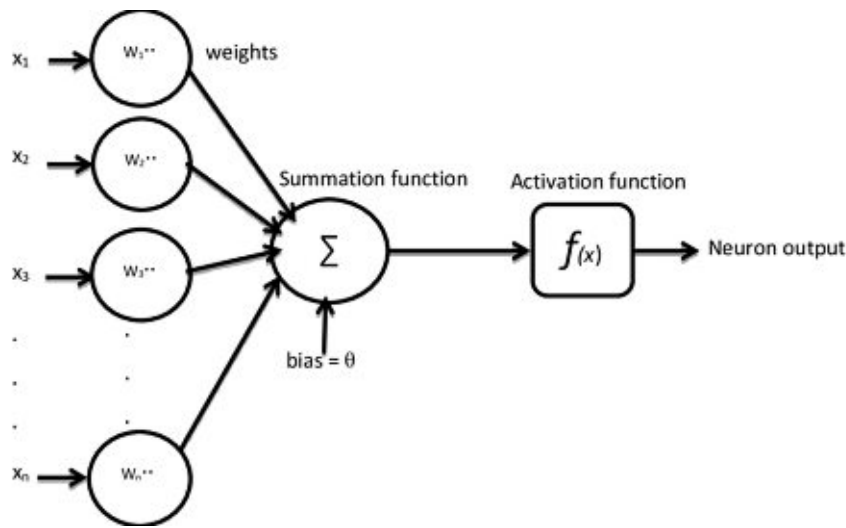


Figure 2.5: Artificial neuron [57]

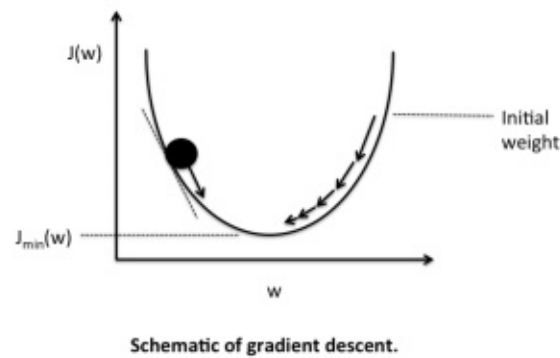


Figure 2.6: Gradient descent perceptron Raschka and Mirjalili [41]

be minimized by differentiation, which is a requirement for gradient descent [41].

$$J(\mathbf{w}) = \frac{1}{2} \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2, \quad \text{output}^{(i)} \in \mathbb{R}$$

Raschka and Mirjalili describes gradient descent in [41] intuitively like "climbing down a hill" to the point of a local minimum. Steps in the size of the learning rate in the negative direction of the gradient are performed until a minimum of the cost function $J(w)$ is reached, this is illustrated in Figure 2.6. Mathematically, Raschka and Mirjalili [41] defines the calculation of the weight update as $\Delta \mathbf{w} = -\eta \nabla J(\mathbf{w})$. $\Delta \mathbf{w}$ is the value that will be added to the weight vector \mathbf{w} . By partially derivating $J(w)$, the weight update for a particular weight Δw_j is determined.

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$
$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = -\eta \sum_i (t^{(i)} - o^{(i)}) (-x_j^{(i)}) = \eta \sum_i (t^{(i)} - o^{(i)}) x_j^{(i)}$$

The learning rate η determines how quickly the weights adapt and is usually a value between 0 and 1, if it is set too low, many epochs are necessary to find a minimum, with the additional risk of getting caught in a local valley of the graph without the possibility to escape, due to the learning rate being too small. Employing higher learning rates results in shorter training periods, but carries the risk of missing the global minimum, due to jumping behaviour on the cost function. Carefully selecting the learning rate is necessary. Finally, the weights are updated all at once via $\mathbf{w} := \mathbf{w} + \Delta\mathbf{w}$. For more complex networks with multiple layers an algorithm called backpropagation is used, which is described by Rumelhart et al. in [49]. Backpropagation is an algorithm to update weights in a neural network via the chain rule, it starts at the last layer of the network and updates its weights first and then traverses through the network, updating one layer after another until the first is reached. Backpropagation uses the gradient of the cost function to find the amount of change in a particular weight or bias required to minimize the cost function.

Activation functions

There are many different activation functions suitable for varying applications, e.g. step, linear, sigmoid, hyperbolic tangent, rectified linear unit and softmax. Step functions output binary values and employ a threshold, i.e. if the calculated sum of the neurons is above the threshold, 1 is passed to the next neuron, otherwise 0. Linear activation functions multiply the sum of a neuron by a predefined factor that does not change during training. These functions have two disadvantages, gradient descent is not possible for the models with more than one neuron, because of a constant derivative. Furthermore, they merge all layers of a network into one, because the output is simply a linear combination of the input values, this results in a regression model not a complex neural network. The current state of the art in activation functions is of non-linear type, these enable the network to reproduce complex relationships required to process high dimensional data occurring in computer vision tasks. Non-linear functions such as Sigmoid, TanH, ReLU, Softmax and Swish have two main advantages, their derivative is related to the input values and therefore allows backpropagation and they allow for combination of several layers of neurons [41].

Types of artificial neural networks

Zell [59] describes different types of neural networks. The simplest form of neural network is called feedforward neural networks, neurons of a layer are only connected

in the forward direction to the neurons of the next layer without any backloops. The multilayer perceptron is an example for a feedforward neural network. It consists of several perceptrons, arranged in multiple layers. Each neuron is connected to neurons in the next layer, i.e. during each forward step, information is only passed in one direction through the layers of the networks and never in the other. A more complex form of neural network are recurrent networks, which in contrast to feedforward networks, allow connections to preceding layers and looping. This reflects the behaviour of neurons in most areas of the human brain. There are four categories to divide recurrent networks: direct feedback networks, which use the output of a neuron as an additional input for it, indirect feedback networks, where the output of neurons is fed back to neurons of preceding layers, lateral feedback networks, where the output of a neuron is used as input for neurons on the same layer and fully recurrent networks, where the output of each neuron is connected to the input of every other in the network. Fully connected networks differ from fully recurrent networks as they only contain connections between every neuron of one layer to the following, but no feedback loops where the output of one neuron is used as input for a preceding neuron (preceding in terms of data has already flown through the preceding node to reach the succeeding). Recurrent neurons enable networks to detect temporal patterns in the processed data, naturally their application lies in areas that require processing sequences of data, e.g. detection of handwriting, speech recognition and translation. It is possible to convert recurrent neural networks into feedforward networks if they do not contain any loops (e.g. lateral feedback). Popular examples are Long Short-term memory (LSTM) [21], the Elman [11] and the Jordan network [25]. Training recurrent networks is more challenging than simple feedforward networks, if they have finite impulse response, a common approach is to convert them to feedforward networks for training, in LSTM networks this is done via Backpropagation-Through-Time.

2.4 Convolutional Neural Networks (CNN)

Goodfellow et al. [17] mention that convolutional networks are especially useful in image, video and audio processing. The most basic convolutional neural networks are usually composed from one or more convolutional layers, after which a pooling layer is placed, this architectural pattern can be repeated multiple times. In Figure 2.7 this architecture is illustrated. Neurons of convolutional layers for image processing are usually aligned as 2D matrices and receive two dimensional matrices as input. Despite other layouts, e.g. 1D for audio and 3D for volumetric image processing (e.g. MRI and CT), being used as well from now on a 2D convolution in the context of image processing is assumed. Each neuron performs discrete convolution over parts of the input matrix via a filter kernel to calculate its activation. Locally close neurons process overlapping areas of the input matrix, for border areas different padding strategies are used. Considering a stacked hierarchy of convolutional layers, one can observe that each neuron is affected only by neighbouring areas of the previous layer. Translation invariance is an important property of CNNs and guaranteed by neurons of one layer always sharing the same weights. Due

2. BACKGROUND

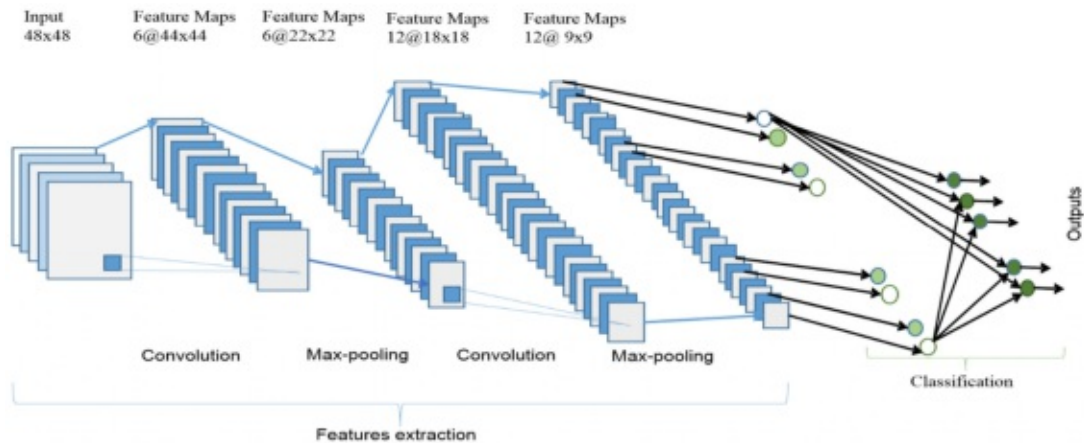


Figure 2.7: Convolutional Neural Network architecture [4]

to shared weights, the first layer of a CNN always performs edge detection, its activation shows the likelihood of an edge in the close vicinity of the neuron's input. The most commonly used activation function is rectified linear unit (ReLU), which is approximated by $f(x) = \ln(1 + e^x)$, as backpropagation requires differentiable functions. The pooling layer in convolutional neural networks is responsible for dimensionality reduction, resulting in 1/4 of information from the previous layer. This dramatic reduction of data increases inference performance and subsequently allows the creation of deeper networks while minimizing overfitting. Dropout is another possibility to reduce the dimensions in a network, a number of randomly selected neurons are removed from the network. The most frequently used approach for pooling is max pooling, where only the highest value of activation is transferred to the next layer, all others are truncated [17]. After a multiple of convolutional and pooling layers, a fully-connected layer is used for classification, the number of neurons in the last layer reflects the number of classes the network should classify. By employing a softmax function, the output of the last layer is transformed into a probability distribution over all classes. Convolutional neural networks follow the principles of supervised learning, where for each input the expected output, i.e. class is specified via backpropagation the weights are updated subsequently. LeCun and Bengio [27] categorizes convolutional neural networks as a special kind of multi layer perceptron (MLP), with fully connected layers of CNNs offering a similar performance as MLP. The success of convolutional neural networks is based on their efficient processing of image data. While shared weights result in low memory footprint, the number of neurons necessary is dramatically reduced by convolution with respect to multilayer perceptrons. Another problem of MLPs is that they scale badly for high resolution images. Furthermore shared weights in layers result in shift-, translation-, scaling- and luminance-invariance, a property that is not found in multilayer perceptrons.

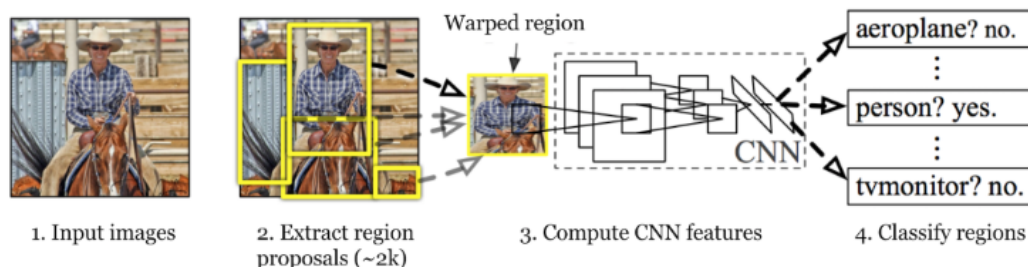


Figure 2.8: R-CNN Architecture[15]

Region-based Convolutional Neural Networks (R-CNN)

Girshick et al. [16] introduces region-based convolutional neural networks. This advancement in the area of CNNs is originally employed for object detection only. Subsequently, improvements of R-CNN were presented in 2015 with Fast R-CNN [14] by Girshick, Faster R-CNN [45] by Ren et al. and Mask R-CNN [19] by He et al., the latter being not only an object detector but object segmentor. The architecture of R-CNN is displayed in Figure 2.8, its process of object detection can be divided into four sequential tasks.

1. The segmentation algorithm selective search is used to identify areas of interest. Roughly 2000 areas of interest are identified, these areas are not uniform in shape but rather polygons as they are output of a segmentation algorithm. By surrounding them with bounding boxes, these areas serve as regional proposals, containing possible objects of interest, which are the input for the next step in the algorithm. This approach is an improvement to the classic sliding window, which is very time-consuming, requires analyzing a greater number of areas and is not adaptive to not square objects of interest.
2. Regions of interest are stretched or shrunk to square boxes, which is the required format for a pre-trained convolutional neural network, that is subsequently used to reduce dimensionality by extracting 4096 features.
3. Several support vector machines, one for each category to classify into, are used to process the features of each region of interest.
4. Regression is employed to improve accuracy of the proposed bounding boxes, i.e. precisely locate the object of interest in the box.

This in 2014 novel approach carries a few pitfalls. First of all, the algorithm, selective search, initially used to identify regions of interest is fixed and does not contain any means for adaption through learning, i.e. the quality of regional proposals varies dramatically depending on the input image. Moreover, training is very time consuming, because of the great number of 2000 region proposals even in small images. On the other hand, due to different parts of the network not supporting in simultaneous training, e.g. the support

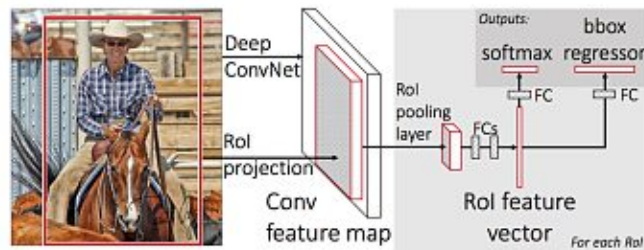


Figure 2.9: Fast R-CNN Architecture [14]

vector machines, the convolutional neural networks and the final regression step. Finally, inference time is very slow with about 45 seconds per image and therefore not suitable for realtime image processing, which might be necessary for camera feed analysis.

Girshick [14] addresses the drawbacks of R-CNN by directly using the test image as input for the Convolutional Neural Network, instead of regional-proposals, which were a time-consuming fixed step in the original algorithm. The changes in architecture of Fast R-CNN are displayed in Figure 2.10. The CNN reduces the image to a feature map, on which afterwards regional proposals are established. Then a process, similar to R-CNN, starts, each regional proposal is stretched or shrunk to fit a square and dimensionality further reduced by region of interest pooling, which is the required format for the fully connected layer. Finally, by applying softmax the region is classified. To improve the accuracy of bounding boxes, bounding box regression is used. By skipping the time-consuming step of analyzing 2000 region proposals, Fast R-CNN offers a speed improvement compared to R-CNN.

Ren et al. [45] propose an improvement of Fast R-CNN. In contrast to R-CNN and Fast R-CNN, the new approach is not dependent on selective search, but replaces it with a region proposal network. The architecture of Faster R-CNN is displayed in Figure 2.9. This is faster as there is no need for time-consuming selective search and a smaller number of region proposals. The only alterations of the original Fast R-CNN are the exchange of selective search for a region proposal network, which first employs a convolutional layer to extract features from the image, based on the feature multiple anchor boxes with varying shapes are set up. An important hyperparameter of Faster R-CNN are anchors scales and ratios. By fitting the aspect ratio and scales of anchors to the object of interest accuracy can be increased. The region proposal network (RPN) then determines the likelihood of containing an object of interest for each anchorbox. This is done by aligning the center of the anchorbox with the centerpoint of the sliding window. The sliding window is of square shape, by employing anchorboxes, unsquare or bigger objects, partially exceeding the boundary of the sliding window, can be detected. Finally, all boxes are compared and only the most promising ones are passed to the next layer. This is called Non Maximum Suppression (NMS). Apart from that, Faster R-CNN is same as Fast R-CNN.

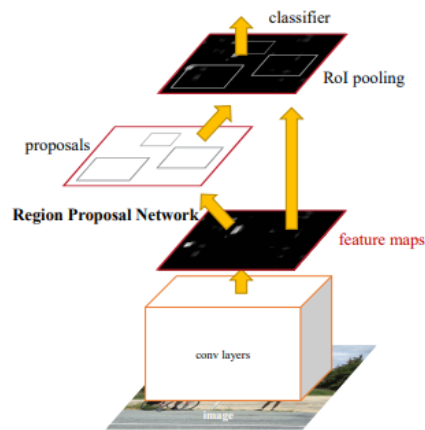


Figure 2.10: Faster R-CNN Architecture [45]

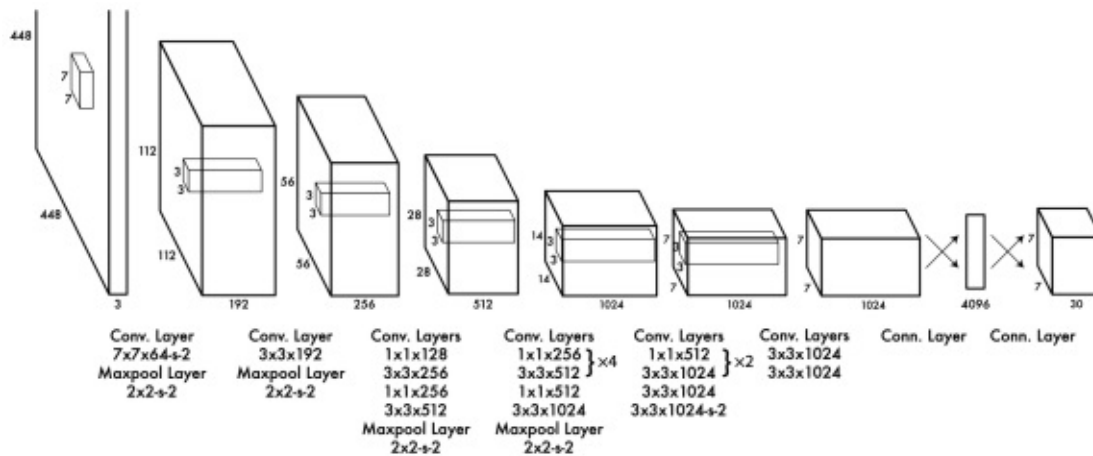


Figure 2.11: YOLO Architecture [44]

YOLO

The algorithm YOLO, short for You Only Look Once, introduced by Redmon et al. [44], takes a different approach and relies on a single pass of the image through the network for object detection. The YOLO algorithm was improved continuously over the last years [42][43][8][24]. This makes YOLO much faster than Faster R-CNN and allows real time object detection with more than 30 frames per second on state of the art hardware, rendering it particularly useful for autonomous driving and camera feed processing, but also analyzing massive amounts of stills benefits from its fast inference time. The architecture of YOLO is shown in Figure 2.11. YOLO only uses one neural network and is therefore straightforward to train. Images are used as inputs for the network, bounding boxes are the output.

Starting from YOLOv2, released in 2017, the input image is divided into a number of prediction areas, e.g. 3x3. Each of these areas is in charge of predicting up to five objects. Naturally most areas do not contain any objects of interest, to avoid a tedious and useless prediction process for empty areas, the "object containment" probability is calculated and subsequently areas with of low probability of object occurrence truncated. The centerpoint of objects determines the area responsible for its prediction, in case of one object overlapping multiple areas, the most central area is selected. Each prediction contains a bounding box, its location within the area and a confidence value. The network itself is composed of a sequence of convolutional and pooling layers, topped up with two fully connected layers at the end. In 2018, YOLOv3 [43] was introduced, which offer higher precision than its predecessor and allows the user to choose model size and subsequently adjust the trade-off between inference speed and precision. In April 2020, Bochkovskiy et al. [8] introduced YOLOv4, which offers further gains in inference speed and precision compared to YOLOv3. When tested on a Telsa V100 GPU with the COCO data set, an about 10% higher precision and 12% faster inference time was measured by the authors of the paper. Shortly after YOLOv4 was released, YOLOv5 was introduced in June 2020 by Jocher et al. [24]. In contrast to previous YOLO implementations, which always originated from Darknet, an open source machine learning framework, and were ported to other frameworks later, YOLOv5 was initially implemented in PyTorch. The framework offers several enhancements, such as automated calculation of suitable anchors and new strategies for data augmentation, e.g. mosaic augmentation. Another advantage of YOLOv5 is its model size, which is about one tenth of the size of a YOLOv4 model with comparable inference time and precision.

2.4.1 Transfer learning

Transfer learning refers to using pre-trained neural network models and adapting them to the new use case. The approach of transfer learning is not exclusively used in the context of convolutional neural networks, but can be used with other deep learning techniques as well. The main reason for its popularity with CNNs is their extremely time and resource consuming training process, if performed from scratch. Models, used for transfer learning, are usually trained on general purpose data sets with numerous instances such as COCO (330k images, 80 classes, 1.5 million object instances) or ImageNet (1000 classes and millions of images). This process lasts sometimes for weeks until it achieves satisfying performance, even if executed on GPU clusters. Then the model is adapted to classify a smaller, different data set, where it often achieves performance levels, unreachable with traditional learning approaches, due to small number of instances. For custom object detection tasks, creating a large, well distributed data set with varying instances is a challenge. By employing transfer learning this issue can be overcome to a certain extent. In general, this process relies on the concept of efficiently transferring knowledge (e.g. which and how to extract features from an image) from one model to another. It has been observed that the first layers of convolutional neural networks generally adapt to perform similar tasks (e.g. detecting edges in images) independently of the data set used. Two approaches are used for transferring these to other networks, feature extraction and

fine-tuning. One approach of transfer learning is fixing weights of all layers and using the network as a feature extractor, whose output is further processed in a new neural network for classification. Another option is to partially fix the weights in the network, usually the first layers, which perform the more general tasks like blob forming and edge detection and allowing last layer(s) weights to be adapted during training, although at a much lower learning rate than during the original training process, on the new data set. Depending on the size of the data set the first or the second approach can be more successful [58].

2.5 PyTorch

All algorithm implementations used in this thesis are based on PyTorch. PyTorch is a machine learning framework and implements Torch (machine learning framework originally for the language Lua) for Python. The framework allows performing matrix operations via tensors on the GPU and automatic differentiation for construction and learning process of networks. In contrast to other frameworks (e.g. Tensorflow), PyTorch builds upon the concept of dynamic graphs, which encourages the definition of the computation graph via Python functions. In TensorFlow the implementation has to be compiled first to be run. In PyTorch, errors are shown as Python native exceptions. Furthermore, control structures such as loops and if-else-blocks can be used while performing backpropagation. Tensors are the most important data structures in PyTorch. A tensor in PyTorch is a multi-dimensional array, that contains data of a specific type, the main difference to a traditional numpy array is that tensors can be used for computation on CPU and GPU, for each tensor type there exists a CPU and GPU variant [41].

2.6 Performance metrics

In machine learning, performance is measured by comparing the prediction of a network to the expected value in the ground truth of a test data set. The most common metrics are listed in this section. Input tuple in the aspect of performance metrics refers to an instance of the ground truth, that is used for training, validation or testing and for which the expected output value is defined. In a binary classification task, true positive refers to a prediction for an input tuple that is of positive class in the ground truth and correctly classified as positive by the model. False positive refers to a prediction that is of negative class in the ground truth, but classified as positive by the model. False negative is a prediction, that is of positive class in the ground truth, but is predicted as negative by the model. Based on these definitions a series metrics was established [41].

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive} + \text{False Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

In object detection, not only the predicted occurrence of an object, but their predicted position and size is of interest. Usually ground truths of data sets used for object detection are annotated with bounding boxes, referring to the area of interest. These boxes contain x and y coordinates of the object in the image and the width and height of the box. To evaluate how well a model performs, the predicted bounding box is compared to the bounding box in the ground truth. The metric Intersection over Union is calculated by computing the area of the image that lies both in the predicted and in the annotated bounding box, i.e. the intersection between those two, and dividing that by the area of the union of both bounding boxes.

$$\text{IoU} = \frac{\text{overlapping area of the bounding boxes}}{\text{union of bounding box areas}}$$

In competitions and research an intersection over union (IoU) greater than 0.5 is considered as well performing. As exactly matching the annotated bounding boxes is extremely unlikely, comparing the coordinates is not a good indicator, Intersection over Union is used instead to gather a model's performance. Based on IoU a few metrics were established, the Pascal VOC metric (AP[0.50]) which measures average precision based on an IoU greater than 0.50 and the COCO metric (AP[0.50:0.95]). COCO defines average precision by the average over a series of intersection over union. IoU in the interval from 0.5 to 0.95 with a step size of 0.05, i.e. 9 different IoU, is averaged to compute COCO AP.

$$\text{COCO mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i$$

Rezatofighi et al. [47] introduce generalized intersection over union. Intersection over union does not provide a metric to compare bounding boxes not overlapping the ground truth, IoU of predictions not overlapping are always zero. This leads to a problem as bounding boxes nearer to the ground truth bounding box should achieve a higher score than those further away. By measuring generalized intersection over union (GIoU) this problem is solved.

$$\text{GIoU} = \frac{|A \cap B|}{|A \cup B|} - \frac{|C \setminus (A \cup B)|}{|C|} = \text{IoU} - \frac{|C \setminus (A \cup B)|}{|C|}$$

When measuring GIoU, the predicted bounding box (A) and the bounding box from the ground truth (B) is surrounded by the smallest possible area C. By integrating hull C in the calculation, predictions in closer proximity to the ground truth are encouraged and those further away are penalized.

2.7 Data augmentation

A frequent problem in object detection are too small training data sets, data augmentation can fix this issue to a certain extent. By performing transformations on the training image before feeding them to the network a bigger data set can be simulated. When performing data augmentation, not only the image, but the annotation (bounding boxes in object detection) has to be transformed as well. Commonly used techniques are randomly flipping or cropping the image as well as altering brightness, contrast, hue and saturation.

2.8 GIoU as a loss function

Major requirement for backpropagation is a differentiable loss function. The metric IoU is not differentiable, as it is not defined in case of no overlaps between prediction and ground truth. GIoU, on the other hand, is always defined and can be differentiated. GIoU has a value lower than 0, if hull C, surrounding ground truth and predicted bounding box, is greater than IoU. By employing GIoU as a loss function, predictions not overlapping the ground truth can be penalized according to their distance to the ground truth. This is not possible with IoU, which is not defined in case of no intersection with the ground truth.

2.9 COCO data set

Machine learning models have to be trained on large data sets, to compare performance between different approaches publicly available data sets are used as a benchmark. COCO (Common Objects in Context)¹ is such a data set, it provides a variety of feature, instances feature pixel masks for object segmentation, 330000 images, 1500000 million object instances, 80 object categories as well as 250000 people instances with posture keypoints [29]. The data set contains common objects such as vehicles (car, train, bike), humans, animals, household items, sport utilities, etc.

2.10 Summary

In this chapter, the research area of computer vision, common problems stated and the typical architecture of computer vision systems were introduced. Basic feature

¹<https://cocodataset.org>

2. BACKGROUND

extraction based algorithms for clustering, edge detection were described as they build the foundation of modern object detection approaches. Furthermore, an overview about machine learning and artificial neural networks with a focus on convolutional neural network, as these are commonly used for computer vision tasks, was given. State of the art models model architectures such as Faster R-CNN and YOLOv5 were mentioned. Finally, metrics to assess performance of object detectors as well as performance markers specific to artificial neural network such as loss are presented.

Related Work

This chapter gives an overview about the state of the art in free-form object detection in the context of signature detection and segmentation from varying backgrounds. Due to detecting signatures in documents being a relatively narrow field of research, some publications presenting approaches and algorithms for the detection of handwriting are included as well. The selected work serves as a baseline and foundation, that allows comparison of performance of the proposed detection pipeline. As mentioned in previous sections, the majority of publications employs a feature extraction based approach, despite the thesis' focus on deep learning based algorithms for signature detection. Including these methodologies is required, as they represent the state of the art in signature detection. The majority of analyzed articles evaluated their approaches for signature detection on publicly available data sets. The most commonly used is the Tobacco 800 data set. A large proportion of approaches are based on feature extraction algorithms, only two papers employ deep learning to detect signatures. Therefore, this section is divided into two parts, first different feature extraction algorithms are introduced, then deep learning based approaches are described.

3.1 Feature extraction based approaches

Many signature detection algorithms are based on feature extraction. The most significant one for the thesis, as it was used to create the ground truth for the Tobacco 800 data set and therefore serves as a reference point for the experiments, is [62]. Zhu et al. [62] propose an approach to jointly detect and segment signatures from varying backgrounds based on multi-scale saliency, which was used in a subsequent publication [63] to implement an end to end document retrieval system. Zhu et al. [62] describe that cluttered backgrounds are a big challenge in signature detection and segmentation. The proposed framework works both for offline and online applications, offline referring to signatures embedded in scanned document images, online for smart devices where stroke directions from the

3. RELATED WORK

stylus are available. The outlined methodology is divided into two tasks, identification of salient structures and grouping of components. Signatures and their most salient structures are illustrated in figure 3.1. Saliency of a structure is defined by dynamic structural curvature and proximity, e.g. large connected, cursive objects like signatures.



Figure 3.1: Examples of detected signatures from the Tobacco-800 data set, together with their saliency maps. The top three most salient parts are shown in red, green, and blue, respectively Zhu et al. [62]

Instead of employing local features with large variance, the most salient parts of a free-form object are used to identify the signature. According to Zhu et al. [62], employing saliency measures referring to high-level knowledge of the objects results in a better performance in object detection, compared to low-level vision constraints. After salient areas are determined, grouping is performed by less complex constraints such as neighbourhood. By using multi-scale saliency the detection of not continuous objects like signatures is improved. Handwritten signatures feature higher in-class variance than other free-form objects due to being unique to each individual. To detect signatures at a given scale, first a Gaussian blur is applied, then a Lanczos filter is used to re-sample the image. Finally, a Canny edge detector is used to determine the edges. This procedure ensures a connected curvature of the signature, by subsequent re-sampling after filtering. Afterwards saliency is calculated for each component by dynamic curvature. The most salient structures of the image are then used to segment the signature, the methodology is evaluated on the Tobacco 800 data set with a detection rate of 92.8% among all signatures in the data set. No machine learning algorithms are used here, the approach relies only on feature extraction.

Mandal et al. [32] describe a methodology to segment signatures from varying backgrounds by employing a Conditional Random Field (CRF). First, signature blocks are identified by word level feature extraction. Then overlappings with machine printed text are segregated with gradient based features and support vector machines. The proposed system is illustrated in Figure 3.2. Finally, a CRF is used to separate the detected strokes in signature and machine printed text. The approach is evaluated with the Tobacco 800

data set. Signature segmentation is performed here on box level, underlying printed text, seals or other free-form objects are not removed completely. This is potentially a problem for signature verification algorithms, that use these bounding boxes as input. Box level refers to predicting a bounding box surrounding the signature, instead of its exact strokes. A precision of 0.8572 and a recall of 0.8186 could be achieved, which is not fairly comparable to [62] as they only state detection rate. The biggest issue of this methodology is the great number of samples required during the training process.

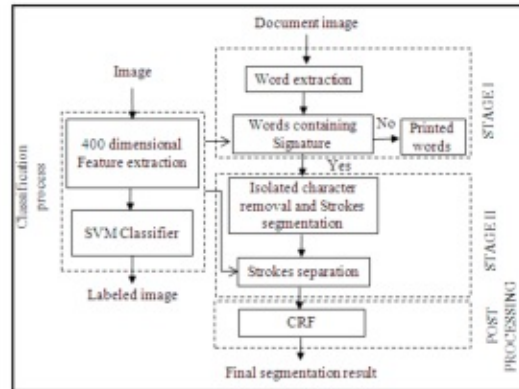


Figure 3.2: Block diagram of system proposed by Mandal et al. [32]

Singh et al. [53] introduce an algorithm to render signatures and seals on digital document images tamper evident, by augmenting documents with watermarking bits. The detection of seals and signatures is performed via feature extraction. A two-step approach is employed to distinguish between the object of interest (signatures) and the cluttered background (printed textual characters). First the scanned document is divided into word patches based on the machine printed text and spacing between words. Bounding boxes of patches are identified by using morphological dilation and connected component labelling. Then signature blocks are separated from machine printed word blocks via gradient based features and a Support Vector Machine. This approach results in imperfect signature blocks, due to possibly isolated printed characters and overlapping strokes. To further refine the hypothetical zones of signatures, machine printed text lines are evaluated via analyzing close machine printed word blocks. The publication provides a framework for end to end watermarking and verifying documents based on seals and signatures and relies in the signature detection part of the work on [32]. The approach is evaluated on the Tobacco 800 data set but no isolated metric concerning the signature detection performance is given.

Mandal et al. [33] describe a methodology to segment signatures from machine printed documents via feature extraction. This new approach solves the issue of patch level only segmentation that was featured in [32]. The two step approach relies on first dividing the image into blocks by a morphological operation, which is followed by block-level analysis to distinguish signatures from machine printed text. 400-dimensional gradient-based

features are used with a Support Vector Machine for classification. The algorithm was then evaluated on the Tobacco 800 data set, where an accuracy of 0.955 was achieved, errors occurred predominantly in areas where the segmentation failed. The achieved accuracy is higher than the accuracy of 0.928 achieved by [62] with a multi-scale saliency approach. Mandal et al. [33] state a potential enhancement of the algorithm to be compatible with mathematical formulae, seals and annotations.

Nandedkar et al. [37] introduce an algorithm for detection of logos, stamps and signatures in documents, furthermore the SPODS data set is presented. Spectral Filtering and Part-based Features (SFPPF) is used to detect these objects. The algorithm is divided into three parts, first potential machine printed text is attenuated in the image by frequency selective filtering, afterwards graphical and textual regions are separated and finally the object detection performed. Different spectral properties in graphical areas of documents build the foundation of the approach. The proposed methodology was evaluated in an experiment with the proposed SPODS data set, which provides pixel level ground truth about the visual entities under test. During the experiments a performance with a precision of 0.93, a recall of 0.86 and an F-Score of 0.9 in the signature class was achieved.

Zheng et al. [60] propose a framework for identification and segmentation of handwriting and machine printed text in noisy scanned document images. The approach uses the novel technique of mapping this two class problem to three classes (noise is treated as a separate class). First, connected patches in documents are identified and combined with neighbouring at block level. Afterwards the following features are extracted for each pattern unit, structural, Gabor filter, run-length histogram, crossing count histogram, bi-level co-occurrence and 2x2 gram. These are subsequently classified by Fisher, SVM and k-NN classifiers into one of the three classes: noise, text or handwritten notes. The Fisher classifier was the best performing with an accuracy of 0.94 in the handwriting class on the Tobacco 800 data set. No deep learning networks are used here.

Ahmed et al. [3] employ Speeded Up Robust Features (SURF) to segment signatures from machine printed text. Using SURF and relying on local features ensures compatibility of the algorithm to visually varying of documents. The approach relies on computing sets of keypoints for the object of interest, each of which specified by a 128 bit descriptor and used to subsequently calculated SURF features with a Hessian threshold of 400, i.e. all keypoints with lower thresholds are truncated. All linked components are detected from the image during segmentation, for each connected component SURF features are computed and the descriptors of the corresponding keypoints matched by Euclidean distance to either the signature or the machine printed text class. The selection of keypoints is illustrated in Figure 3.3. After all keypoints have been processed, the patch is assigned to the class with the greater number of keypoints. The approach introduced by Ahmed et al. [3] suffers from the same problem as the approach from Mandal et al. [32], signatures are segmented on patch level only, this potentially interferes with using the algorithm in a signature verification pipeline. The feature-extraction algorithm is evaluated and trained on the Tobacco 800 data set. Compared to other approaches it manages training with only 10 samples and achieves a precision of 0.5652 with a recall of

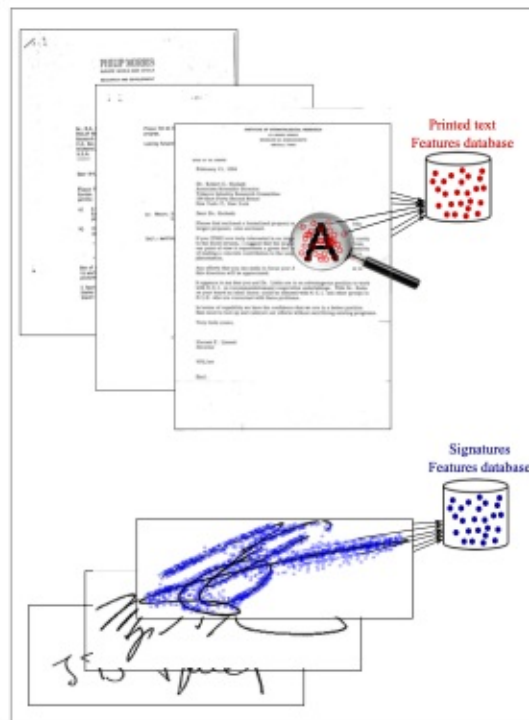


Figure 3.3: Extracted signature keypoints (blue) and printed text keypoints (red) during training Ahmed et al. [3]

1. A fair comparison to Mandal et al. [32] and Zhu et al. [62] is not possible, as both did not state accuracy or exact filenames used for testing. Ahmed et al. [3] describe the reason for low precision to be logos in the documents, adding a logo class should solve this problem.

Madasu et al. [31] employ a sliding window approach to detect handwritten signatures in scanned bank checks. Bank checks, as well as general documents, feature large in class variance. The proposed system relies on a multi-step procedure. The approach requires basic knowledge about the layout of the checks that starts with a preprocessing step via a B-spline filter, that separates a checks background from handwriting. Next the signature is extracted via a sliding window, traversing horizontally over the image and calculating the entropy. Areas of high entropy, i.e. rectangular boxes in the check, are used as markers to find the area of interest. The isolated boxes are then further processed via cropping, shown in Figure 3.4. A sliding window starts in each edge of the isolated bounding box and decreases or increases either the x or y coordinate until a black pixel is reached (the start of a signature). An experiment was performed to evaluate the algorithm, 45 bank checks were analyzed where 97.78% signatures could be extracted successfully. Djeziri et al. [10] introduce a method to extract signatures from check backgrounds based on human perception. A topological criterion called filiformity

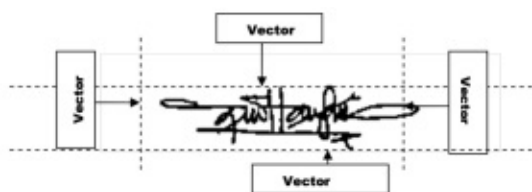


Figure 3.4: The crop method Madasu et al. [31]

is defined. They define edges in images as filiform objects, which are typically longer than wider. To detect signatures, different types of lines are analyzed. Bank checks usually contain a series of boxes, in which customers write information regarding the transaction, e.g. amount and purpose. The authors of the paper name the borders of these boxes layout defining lines or contouring lines. According to them, these lines typically feature dramatic contrast of intensity. On the other hand, handwritten lines, e.g. signatures or handwriting in general, are characterized by a much softer contrast of intensity. Via this criterion, it is possible to distinguish between a machine printed, contouring line and a handwritten line. When following a contouring line it intersects another orthogonal contouring line at some point. These contouring lines form boundaries that divide a check into multiple images. Handwritten lines, e.g. signatures, are part these images. This heuristic is mapped to a function $G(s)$, that refers to the set of pixels, whose intensity is greater than the threshold s . $E(s)$ refers to the derivative of $G(s)$. Finding the optimal threshold S_{opt} , is key to isolate all handwriting in the image under test.

Shetty et al. [52] employ Conditional Random Fields to segment objects in scanned document images. Different visual entities such as machine-printed text, handwriting and noise are labelled. The proposed algorithm is used in signature verification pipelines, which frequently rely on segmented signatures. Initially region growing is applied to the document to divide it into areas of interest. Then each area of interest is labelled via a Conditional Random Field model. The authors of the article state that the model allows classification of signatures as a subtype of handwriting. Conditional Random Fields perform exceptionally well in tagging sequences, which carries over to classifying regions of interest that are often categorized by spatial relationships, e.g. handwritten notes, that are composed of sequential patches from left to right. For each region of interest several state features are extracted, e.g. height, average width of internal connected components, density, aspect ratio, number of components and average run length. Parameters for the Conditional Random Field are approximated by conjugate gradient descent by analyzing the features of 3500 regions of interest from the training data set. The proposed system is illustrated in Figure 3.5. The approach is tested on the Tobacco 800 data set, analyzed in many other publications as well. 53 documents were analyzed, divided into 26 training and 27 testing documents. The algorithm divides each document into approximately 150 areas of interest, which are classified. Using the ground truth of the Tobacco 800 data set as a baseline, an accuracy of 0.957 could be achieved, machine printed text achieved 0.980, handwriting 0.950 and noise 0.900. In contrast to the deep learning based

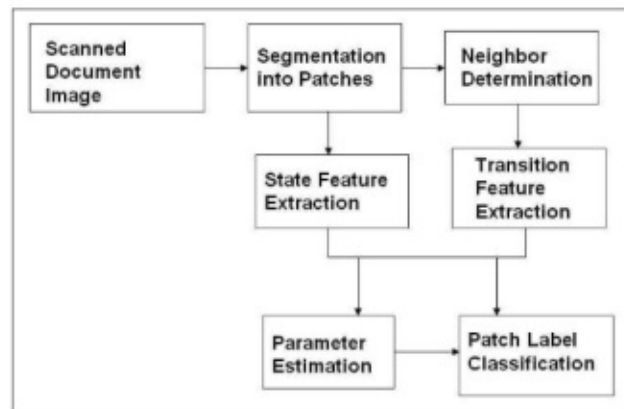


Figure 3.5: A block diagram describing the steps involved in labeling scanned documents, from Shetty et al. [52]

approaches, the relatively low number of training samples is observed as an advantage, while extracting features for 150 patches of interest seems inefficient, due to the actual number of signatures on a single page amounting up to only a fraction of that. Pre-selection could be improved to reduce the number of patches for which features have to be extracted and analysed.

3.2 Deep learning based approaches

Only two publications could be found in which where the authors performed an experiment to detect signatures with convolutional neural networks. In one of which the Tobacco 800 data set is analyzed as well. It is related to the thesis, but as the publication is from 2018, the used network architectures and models are obsolete or newer better performing versions were released. Therefore, it can be used as a starting point concerning the achievable precision, as with newer better performing models higher precision should be achievable. Sharma et al. [51] propose a methodology to detect signatures and logos in scanned document images with convolutional neural networks. The authors describe high intra-category variance to be one of the main challenges in signature detection. The approach was evaluated with the Tobacco 800 data set. In the article four different network models were used for signature and logo detection. Due to limited computational resources and a comparably small data set, transfer learning with fine tuning was performed, i.e. training of models was not performed from scratch but from pre-trained states, where not the complete model but only layers were updated. The following convolutional neural network architectures were tested, Faster R-CNN with different backbones (ZF, VGG, VGG16) and YOLOv2. Non-maxima suppression was applied to distill the most accurate predictions from the models. ZF and VGG were trained with 60000 iterations and achieved a mean average precision (mAP) of 0.882

3. RELATED WORK

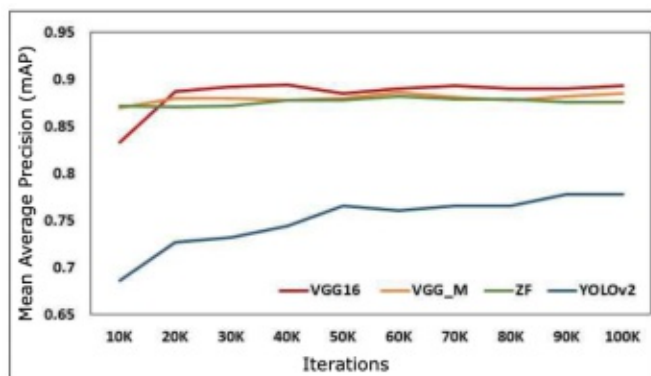


Figure 3.6: Mean Average Precision analysis of all the CNN networks on Set 2 (Train 30%), from Sharma et al. [51]

and 0.884 in the signature class and 0.886 and 0.892 in the signature class, VGG16 was trained with 40000 iterations achieving a mAP of 0.894 and 0.896 in the signature class. YOLOv2 was performing worse with a mAP of 0.778 and 0.788 in the signature class, while being trained for 900000 iterations. The data was split into 30% for training, 10% for validation and 60% for testing. The performance change of mean average precision during the training process of the models is shown in Figure 3.6.

Ribeiro et al. [48] introduces a deep learning based approach for off-line handwritten signature recognition. The approach is focused on signature verification instead of detection and assumes already extracted signatures from documents. A two-step approach is described for signature verification, first the owner is determined by the identification classifier, trained on all signatures and then verified by specific classifiers, trained for each individual. The model is trained and evaluated on the Grupo de Procesado Digital de Senales (GDPS) data set¹. It is composed of signatures of 300 legal entities, with 24 valid and 30 forged signatures per instance. Support Vector Machines were chosen as the classifier with the following features: MDF, Width, Six-Fold Surface and Wavelet Transform. Modified Direction Feature (MDF) showed the strongest impact on classifier performance, the feature is based on Direction Feature (DF) and extracts direction information of the structure of character contours [7]. Width refers to the characteristic of average width in pixels of signatures. Six-Fold Surface splits a signature into three equal cells and computes the center of mass for each of it and splits each area in an upper and lower part. Area of upper and lower part for each cell then calculated, which results in six features [39]. Wavelet Transform extracts horizontal, vertical and diagonal pixels variations of signatures. The error distribution per feature is illustrated in Figure 3.7. The evaluation was performed with signatures from 121 individuals and achieved a precision of 0.86 and a recall of 0.85. Concerning deep learning a neural network with 100 visible units and two layers with 100 hidden units each was employed to compute representations of

¹<http://www.gpds.ulpgc.es/download/>

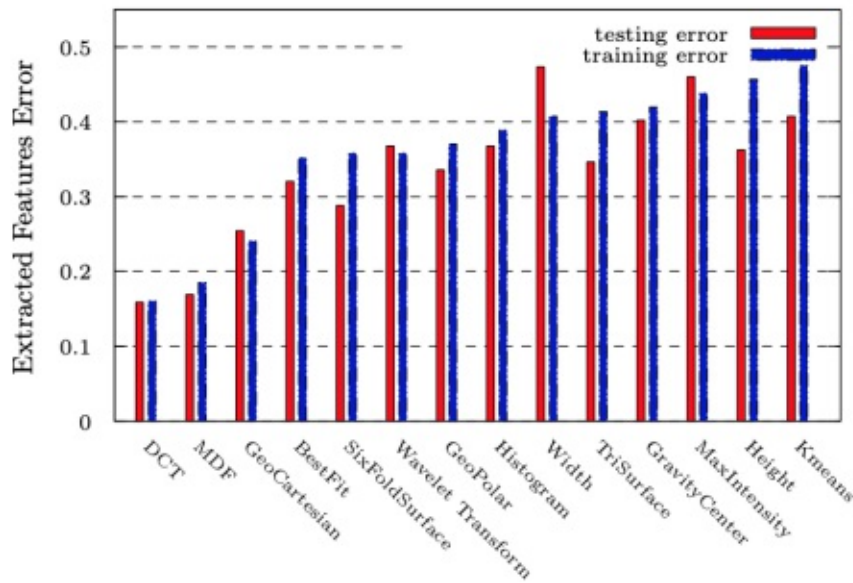


Figure 3.7: Errors per feature, from Ribeiro et al. [48]

signatures. Ribeiro et al. [48] demonstrated that an accurate representation of signatures can be extracted with neural networks, but mentioned limited computational resources with led to a training of only 5000 epochs.

Mondal et al. [36] introduce a new data set for detecting and segmenting objects in scanned document images and create a baseline for performance comparison with convolutional neural networks. The data set is composed of 13000 pages of publicly accessible annual reports of companies. Graphical entities of interest, namely 13000 tables, 3000 figures, 3000 natural images, 500 logos and 600 signatures are annotated with bounding boxes in the ground truth. Due to the relatively small number of signatures, the data set is primarily used useful for general document layout analysis or table and figure detection. Mondal et al. [36] describe the data set to be perfectly usable for training convolutional neural networks for object detection purposes - comparability is maximized by providing the filenames used during training, validation and testing of the baseline models. Furthermore, it is stated that smaller manually annotated data sets are better for training object detection models than massive automatically created data sets. In order to provide a baseline for future object detection approaches, a baseline is created by using Faster R-CNN and Mask R-CNN. Transfer learning, with COCO pre-trained ResNet-101, was employed to speed up the learning process. The Faster R-CNN model is based on PyTorch and was trained with a dynamically reducing learning of 0.001, decaying every five epochs by reducing it to a tenth of the previous value. Dynamically reducing learning allows early epochs of the training process to have more impact on the model, than latter ones where the model has already learned basic features. On the other hand, Mask R-CNN was implemented with Tensorflow and Keras, training was not done

from scratch but again via a ResNet-101 backend, pre-trained on COCO. In contrast to the Faster R-CNN training process, a fixed learning rate of 0.001 was employed, as well as 0.9 momentum and 0.0001 weight as weight decay. The baseline for signature detection in the data set for Faster R-CNN and unseen test data is a mean average precision of 0.898, with a precision of 0.775 and a recall of 0.886. Mask R-CNN resulted in a mean average precision of 0.911, a precision of 0.787 and a recall of 0.917.

3.3 Conclusion

There is a lot of research in the area of signature detection in documents, but most of the proposed algorithms are based on feature extraction, including many approaches that rely on contextual information and are outdated at the time of writing this thesis. A few deep learning based approaches were analysed, as well, but again these use deprecated algorithms and models. Furthermore, the hardware the models were trained on is no longer state of the art. With current GPUs a greater number of epochs and iterations could be achieved while training. Moreover, traceability and comparability are limited in the analyzed publications processing the Tobacco 800 data set, due to not stating the exact file names of images used for training, testing and validation respectively, but only the percentage split. Additionally the approaches were not tested on the complete data set, as some images need be to used for training. This leads to several issues, the performance of the presented algorithms presented cannot be compared in a fair way as well as the performance might be impacted by the selection of easier files to detect. Sometimes the authors did not even state comprehensive metrics, i.e. missing recall or precision. In Figure 3.8 the distribution of publications with respect to the analyzed data set is illustrated. It can be observed that the Tobacco 800 data set was used in most publications. Provided a large and diverse data set with scanned document images with signatures could be gathered and sufficiently powerful GPU cluster allocated, a promising experiment would be to train a convolutional neural network from scratch. Another research opportunity would be to analyze a potential performance impact in case of the Tobacco 800 data set, augmented with two more classes: date and handwritten note.

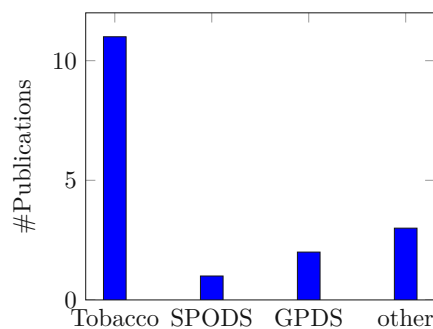


Figure 3.8: Distribution of datasets used in publications

Experiment Structure

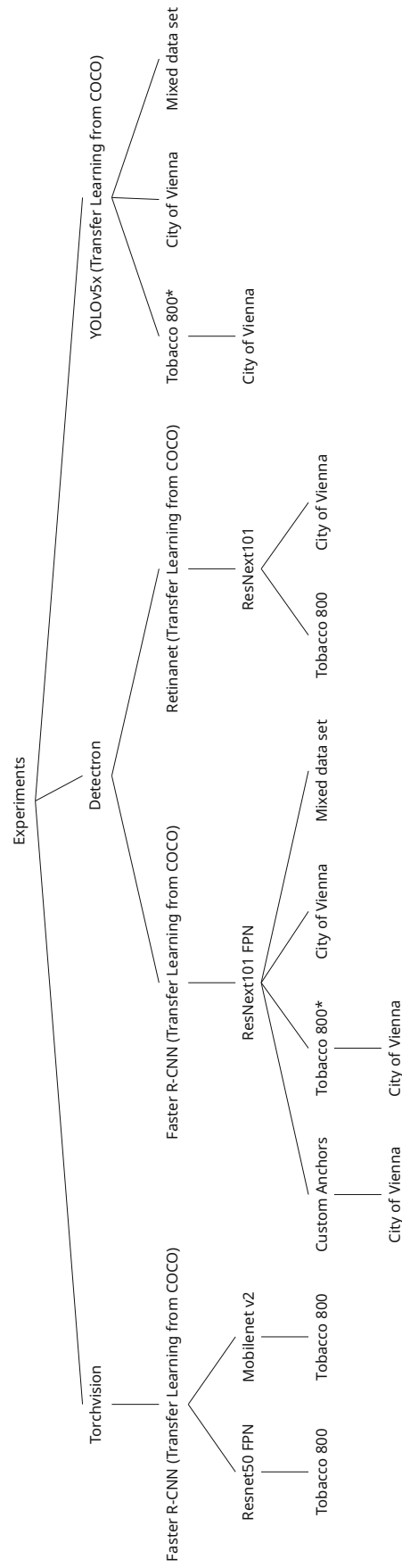
This chapter describes a series of different experimental setups used to evaluate the proposed methods for detecting signatures in scanned document images. It gives the reader an overview about the employed algorithms, implementations and frameworks. In addition, two data sets, mentioned in earlier chapters, are introduced in detail, as well as how the data sets had to be processed to serve as training data for the models. Finally, the hardware configuration, that the training of convolutional neural networks was performed on, is stated. While all implementations of signature detection algorithms used in this thesis are based on the framework PyTorch, the utilized convolutional neural networks (Faster R-CNN, YOLOv5, Retinanet) and backbones (Resnet, MobileNet, ResNeXt) differ. Different frameworks (Torchvision, Detectron, YOLOv5) build on top of PyTorch and implement these convolutional neural networks. The structure of this chapter follows the type of CNN implementing framework instead of the type of network. Subsequent sections refer to each of these experiment setups by a unique identifier, specifying the network, data set and type of model. Torchvision was chosen as it allows rapid prototyping with minimal configuration overhead to initially assess if the proposed approaches suffice. After successful experiments further CNN implementing frameworks such as Detectron and YOLOv5 were included. These have more configuration overhead, but provide a wider array of pre-trained models for transfer learning, more flexibility and better customization options for the learning process.

4.1 Data sets

The following section introduces two data sets, used to evaluate the performance of the proposed signature detection approaches. Their structure as well as samples are described. Furthermore, different data set splits into training, validation and testing partitions, required for the training of deep learning models, are mentioned. Experiment labels, introduced in this section, follow a specific format: <data set abbreviation>_

4. EXPERIMENT STRUCTURE

<architecture>_<model name>_<optional: notes>. For data set abbreviations, "T" is used for the Tobacco 800 data set and "W" for the City of Vienna data set. Due to the City of Vienna data set being rather small, the Tobacco 800 data set was merged with the City of Vienna data set for some experiments for training, evaluation was performed only on instances of the City of Vienna data set, this is indicated by the "MIX" prefix in some experiments. In total 14 different experiment configurations were defined, these can be observed in the experiment tree. All leafnodes as well as nodes, marked with a star "*", are experiment setups. Star marked nodes served as a transfer learning base for further experiments.



4.1.1 Tobacco 800 data set

The data set [1][2][28] is compiled from a variety of scanned document images from the Master Settlement Agreement in the tobacco industry. It consists of 1290 documents in TIFF format containing machine printed text, sometimes accompanied by handwritten notes, signatures and logos with dimensions ranging from 1200 x 1600 to 2500 x 3200 pixels, DPI ranging from 150 to 300 DPI. Not all documents contain signatures and logos. The ground truth for the data set is supplied by the Language and Media Processing Laboratory, University of Maryland [62][61]. Logos and signatures in the scanned document images are annotated with location and dimensionality (bounding boxes surrounding the object of interest), stored in XML files via the GEDI (ground truthing editor and document interface) format. In Figure 4.1, the different combinations of signatures and logos, occurring in the data set, can be observed. Great diversity is

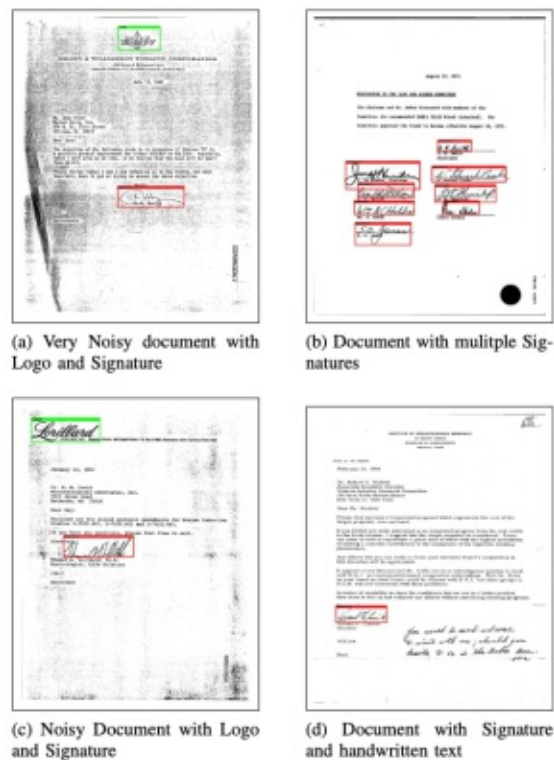


Figure 4.1: Samples Signature and Logo detection results from Tobacco-800 data set. Sharma et al. [51]

one of the biggest advantages of the data set, visual entities do not only occur isolated, but overlapped and in multiple. The instances include various difficulties for signature detection, ranging from noisy document scans and groups of signatures, over handwritten notes (not annotated in the ground truth) to logos. The data set was split with different ratios into a training, testing and validation partition. Subsequent chapters refer to splits

	Training	Validation	Testing
Set 1	44%	10%	36%
Set 2	60%	10%	30%
Set 3	75%	10%	15%
Set 4	80%	10%	10%

Table 4.1: Tobacco 800 data set training, validation and testing splits

listed in Table 4.1 by "set <number>". The validation data set was used to evaluate model performance during training and to select the best performing model after a training session was finished. Then the selected model was evaluated on the testing data set. Scores stated in the results section refer to the performance on the testing partition. Numerous publications (mostly feature detection based approaches) used the Tobacco 800 data set for performance evaluation and benchmarking. This renders it the perfect data set to test the models trained in this thesis on. The filenames for different splits can be found in the appendix.

4.1.2 City of Vienna building applications data set

The municipality MA 37 of the City of Vienna (CoV) covers the whole process of building projects, ranging from the approval of building applications to final checks after building completion. In this process, a variety of documents has to be processed, many of them with signatures and stamps. These documents are collected and serve as a second ground truth for training convolutional neural network models. The achieved performance on this ground truth will ultimately determine the effectiveness of the proposed signature detection system. A diverse and numerous data set is necessary for training convolutional neural networks. Due to building applications containing confidential information and posing the requirement of protecting the identity of applicants by law, the acquired data set cannot be publicly distributed or made accessible by any means. Transmission of the documents is performed via a secured connection, without possibility of access for uninvolved third parties. The main reason for acquiring a new data set and training models on, is a substantial difference between the documents occurring in Tobacco 800 and the building application documents of the City of Vienna. While signatures in the Tobacco 800 data set occur usually isolated without any overlapping entities, signatures in building application documents are often partially covered by stamps, logos or handwritten notes. Furthermore, the aspect ratio, orientation and color-space (black and white or color scans) of documents in the latter is not fixed. All these differences render a model, perfectly suitable for predicting signatures in the Tobacco 800 data set, inefficient and incapable of performing well in the City of Vienna data set. Therefore, further fine-tuning and transfer learning is necessary. The City of Vienna data set was split into several training, validation and testing partitions, these are outlined in Table 4.2. Subsequent chapters refer to splits listed in Table 4.1, in context of the City of

	Training	Validation	Testing
Set 1	60%	10%	30%
Set 2	70%	10%	20%
Set 3	80%	10%	10%
Set 4	90%	5%	5%

Table 4.2: City of Vienna data set training, validation and testing splits

Vienna data set, by "set (W) <number>". Experiments with the "MIX" prefix follow this data set split analogously, except for testing, where only instances belonging to the City of Vienna data set were used.

4.2 Experiment setup Torchvision

The machine learning framework Torchvision for Python offers a variety of state of the art deep learning implementations. In this thesis, its Faster R-CNN implementation was used. First, the Tobacco 800 data set had to be converted into a compatible format. Therefore, the data set was used as is, i.e. without augmenting it with further annotation (e.g. additionally annotating handwritten notes) to maximize traceability and comparability. Logos were neglected during training, due to only signatures being relevant in this thesis. Furthermore, documents without signatures were identified by scanning the ground truth and removed from the training data, as negative examples cannot be used for training Faster R-CNN models. The next step was data set preparation, the Tobacco 800 data set is available in TIFF format with annotations in XML format, both cannot be processed without further conversion by Torchvision's F-RCNN implementation. Therefore, all images were converted to JPEG and the `torch.utils.data.Dataset` class extended and the `init` and `get_item` method overwritten to serve the data to the model. The `init` function prepares the filepaths for annotation and image data to be served to the data loader via `get_item`. The member variables `imgs` and `annotations` hold images and annotations respectively. In the `get_item` method instances are converted to a format that allows them to be fed to the data loader used for training the model. First the id of the image in the list is stored in a tensor and the respective XML file is parsed to read the coordinates and dimensions of the bounding boxes of signatures. Furthermore, the number of signatures in the image as well as the area of the bounding box is calculated. Boxes and areas of boxes are stored in an array, which is then converted to a tensor. Finally, transformations for data augmentation are applied to the image before passing it to the data loader. `RandomCrop` from Torchvision transforms was applied for data augmentation, this transformation randomly crops parts of the image. Both the annotation and the image had to be transformed. Then data was split into various sets of training, validation and testing according to Table 4.1. Instead of training the Faster R-CNN models from scratch, transfer learning was employed, the backbone of the network was pre-trained on the COCO data set. Based on this methodology, a series of models were defined, each

of which was assigned a label for reference in subsequent chapters. During the training process, data was fed to the models via Torchvision's data loaders, data augmentation in form of transformations was only performed on training data, augmenting validation and test data sets would bias the performance metrics. Due to mediocre performance on the Tobacco 800 data set and limited performance evolution tracking in regard of the training process, no experiments with Torchvision were performed on the City of Vienna data set. Limited performance evolution tracking refers to no out of the box support for logging validation accuracy and loss during training in TensorBoard. TensorBoard is a tool for analyzing the machine learning training process.

Faster R-CNN Resnet 50 Feature Pyramid Network

Label: T_TV_FRCNN_resnet50_FPN

Resnet 50 Feature Pyramid Network was used as backbone. Only the classifier in the network was replaced with a new one fitting the Tobacco 800 data set. As the network should only differentiate between background and signatures, logos or other graphical instances were neglected in this step, the number of classes was set to 2 (one for signatures and another one for the rest). The predictor was replaced with a new bounding box predictor.

Faster R-CNN Mobilenet V2

Label: T_TV_FRCNN_mobilenetv2

In another experiment the backbone of the network was changed to Mobilenet v2, including a custom anchor generator and roi pooler, i.e. the anchor generator and roi pooler from the transfer learning base model were overwritten. Custom anchor generator refers to not using the pre-defined anchor generator from the COCO pre-trained model. Anchor generators are responsible to create location proposals of objects of interest. Roi (region of interest) poolers reduce the number of region proposals based on their features. The sizes of the anchor generator were set to 32, 64, 128, 256 and 512, with aspect ratios of 0.5, 1.0 and 2.0. MultiScaleRoIAlign was selected as region of interest pooler, with an output size of 7 and a sampling ratio of 2.

4.3 Experiment setup Detectron

Detectron 2 overcomes a few disadvantages of Torchvision, with a greater selection of pre-trained models, better configuration and tracking options for the training process as well as advanced data set management. Therefore, further experiments were performed with Detectron 2. The tool facilitates a series of computer vision tasks not limited to object detection, but includes object segmentation via Mask R-CNN, Panoptic segmentation and Keypoint detection. The framework relies on the COCO format in data sets, as a consequence the Tobacco 800 annotations had to be converted to a list of Python dictionaries, following the COCO standard and registered in the Detectron 2 data set catalog via the listing. Each data set has associated meta information, referring to class names in case of object detection, for other disciplines more information is stored. For all subsequent experiments with Detectron 2, a routine for repeatable selection of training,

test and validation data sets was established. First all filenames of images in the data set folder are loaded via *os.listdir*, the resulting list is then put in alphabetical order via *sorted*, annotations are treated analogously. This ensures repeatable experiments as files are split afterwards into training, validation and testing partitions. Afterwards, the data set is partitioned by employing scikit-learns *train_test_split* with a fixed random state (seed = 10) following the ratios of Table 4.1. Creating all data set partitions according to this routine ensures maximum comparability among experiments. The configuration of models in Detectron 2 is performed by using the default configuration of a network architecture (hyperparameters and CNNs are defined in Detectron 2 in yaml files) as a base and then fine tuning it via the configuration object *cfg* in Python. Via *cfg*, parameters of the base configuration can be overwritten. Detectron 2 provides a transformation pipeline, building on Torchvision’s transformation package, it facilitates augmentations like *RandomBrightness*, *RandomFlip* and *RandomCrop*. In the following experiments *RandomFlip* and *Resize* were used. These were used to simulate documents with wrong orientation and of varying size.

Faster R-CNN ResNeXt 101 Feature Pyramid Network

Label: T_D_FRCNN_Resnext101_FPN

Label: W_D_FRCNN_Resnext101_FPN

Label: MIX_D_FRCNN_Resnext101_FPN

The first experiment performed is based on Faster R-CNN with transfer learning. Via model zoo, a library of pre-trained neural network models, the base configuration for Faster R-CNN with a Feature Pyramid ResNeXt 101 (*faster_rcnn_X_101_32x8d_FPN_3x*) Backbone is fetched and a checkpoint pre-trained for 270000 iterations on the COCO data set loaded. During the experiment different combinations of frozen layers were evaluated, details are given in the results section. Convolutional Neural Networks are often organized in layers, when performing transfer learning parts of the network can be frozen, i.e. will not be fine tuned on the new data set, but the weights of the base model are kept.

Faster R-CNN ResNeXt 101 Feature Pyramid Network Transfer Learning from Tobacco 800

Label: W_D_FRCNN_Resnext101_FPN_TF

In another experiment, the best performing Faster R-CNN model of the Tobacco 800 data set prediction task was used as a base for transfer learning the City of Vienna data set.

Faster R-CNN ResNeXt 101 Feature Pyramid Network Custom Anchors

Label: W_D_FRCNN_Resnext101_FPN_CA

The default anchors of Detectron’s Faster R-CNN are customized to suit the COCO data set. Objects of the City of Vienna data set differ dramatically from COCO’s objects, especially in regard of aspect ratio. Therefore, an experiment with custom anchors for the City of Vienna data set was performed. Further details on the hyperparameter anchors (scales and ratios) were described in Chapter 2. To acquire suitable anchors the ground truth of the City of Vienna data set was analyzed. First the distribution of instance

dimensions was aggregated and normalized to range of 600 - 1024 pixels. Dimensions, exceeding this range, were either set to 600, if they cross the lower boundary and to 1024, if they exceed to higher boundary. Subsequently bounding boxes were resized, the distribution of their dimensions is displayed in Figure 4.2. Then the size of the base

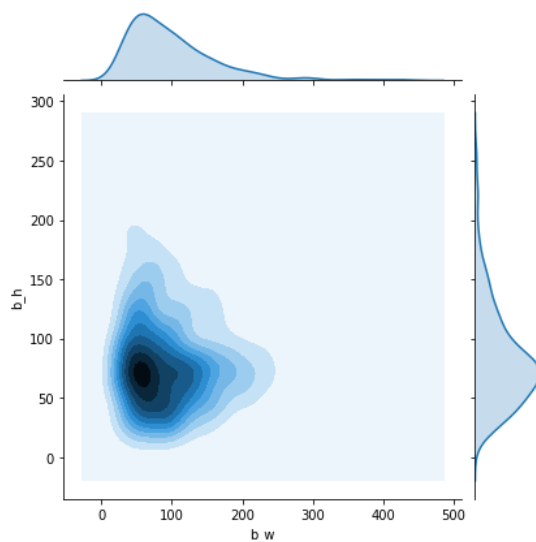


Figure 4.2: Dimensions distribution of normalized bounding boxes

boxes is calculated, their distribution is shown in Figure 4.3. Then the aspect ratios of

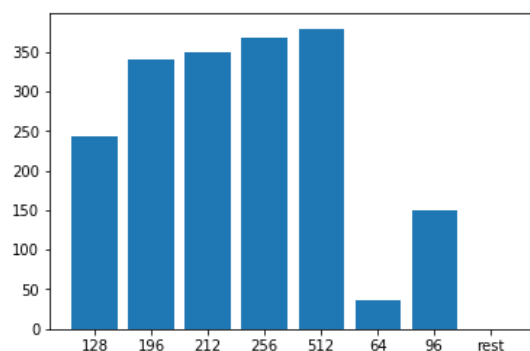


Figure 4.3: Distribution of base boxes

the boxes are clustered via K-Means, resulting in 0.597, 2.454, 4.093 and 1.447, anchor scales were treated analogously, resulting in 0.552, 1.157, 0.874 and 1.553. These new anchors were used in training during this experiment.

Retinanet Resnet

Label: T_D_Retinanet_Resnext101

Label: W_D_Retinanet_Resnext101

Further experiments were performed with Retinanet and ResneXt101 backbone. Retinanet is a one stage object detection model and provides faster inference speed compared to Faster R-CNN. During the experiment different combinations of frozen layers (freezing weights of certain parts of the network) were evaluated, details are given in the results section.

4.4 Experiment setup YOLO

A couple of years ago, using one stage object detectors (especially YOLOV2 and YOLOV3) for high precision demanding tasks such as detecting signatures in documents would not have resulted in good performance, as these models are traditionally employed in video analysis and real time video stream processing. When detecting objects in videos, accuracy was sacrificed for inference speed. Due to recent advancements in one stage object detection approaches, performance comparable to Faster R-CNN can typically be achieved, sometimes even surpassed. Therefore, the newest version of YOLO, YOLOV5 is included in the experiments of this thesis. YOLOV5 relies on a different annotation format than Detectron and Torchvision. Subsequently, the data set has to be converted first into a suitable format. YOLOV5 expects training and annotation data to be in the folders images and labels respectively. For each image in the images folder, the folder labels has to hold a corresponding annotation file, specifying each visual entity the network should be trained on. Annotation files have the same filename and *.txt* as file ending, each visual entity within one image is in a separate line in the annotation file. Each line follows the format: class id, bounding box center x-coordinate, bounding box center y-coordinate, bounding box width and bounding box height. All coordinates and values are normalized on the image dimensions and, therefore, range between 0 and 1. The data set is complemented with a yaml file, specifying the number of classes, class names and paths to the training, validation and test partition of the data set. A python script was used to distribute the instances of the data set in the corresponding directories based on the splitting ratios of Table 4.1. YOLOv5 offers different model sizes, in the subsequent experiments only the largest YOLOv5x is used, as it prioritizes accuracy over speed. As we are not performing real time object detection, prediction speed can be neglected.

YOLOv5x

Label: T_Y_YOLOv5x

Label: W_Y_YOLOv5x

Label: MIX_Y_YOLOv5x

In the first experiment with YOLOV5, a COCO pre-trained YOLOv5x model was used. The architecture of models used in YOLOV5 can be modified in yaml files. The architecture of layers was not changed, only the number of classes was set to 1 (signatures). When using Transfer Learning with YOLOv5x, freezing layers is discouraged by the author of the framework as it results in poorer performance in case of YOLOv5, therefore all layers were trained during learning phase. Adapting anchors is not necessary for

YOLOV5, as the network determines these automatically if the original anchors from the transfer learning base model do not fit.

YOLO5x Transfer Learning from Tobacco 800

Label: W_Y_YOLOv5x_TF

In another experiment, the best performing Faster R-CNN model of the Tobacco 800 data set prediction task was used as a base for transfer learning with the City of Vienna data set.

4.5 Hardware

Training convolutional neural networks is a resource intensive task that can take several weeks to complete. Employing transfer learning reduces training time significantly. Depending on data set composition, usable results can usually be expected within 24 hours. The training process was executed on the GPU cluster of TU Wien. As of 2020 the cluster facilitates 4 Nvidia GTX 1080 TI with 12 GB VRAM and 2 Titan RTX with 24 GB VRAM. To further accelerate the training, distributed learning was performed in longer training sessions.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Experiment Results

This chapter describes the performance achieved with deep learning models in experiments, defined by the experimental structure of the previous chapter. Detailed metrics such as frozen layers as well as hyperparameters of the models are given. Results are described for two data sets, Tobacco 800 and City of Vienna building application data set. In this section, scores of different model architectures are listed separately, in-depth comparison is performed in the subsequent discussion chapter. Concerning training process plots, YOLOv5 uses a combined loss function, which includes `obj_loss` and `giou_loss`. Therefore, both loss functions are plotted. YOLOv5 labels `AP[0.50:0.95]` as `mAP_0.5:0.95` and `AP[0.50]` as `mAP_0.5`. In regard of the learning process of Detectron models, total loss refers to the loss on the training data set and validation loss on the validation data set. Detectron labels `AP[0.50:0.95]` plots as `AP` and `AP[0.50]` as `AP50`.

5.1 Experiments on Tobacco 800

Based on the experiment configurations, identified by experiment labels, in Chapter 4, various experiments were conducted with differing data set splits. This section presents and discusses results experiments concerning the Tobacco 800 data set. The best performing model and data set split of each configuration was selected for analysis and comparison with their counterparts from other architectures. Out of these, the best performing model was chosen and compared with the state of the art approaches, identified in the literature review. In many plots, the graph has been smoothed to enhance readability, the original graph is visible in the background in lighter color.

5.1.1 Torchvision

In this section, results of experiments performed with Torchvision on the Tobacco 800 data set are presented and discussed. Furthermore, selected examples of predictions are illustrated and analyzed.

Backbone	Epochs	LR	Data set	AP[0.50:0.95]	AP[0.50]	R
resnet50	10	0.001	Set 1	0.547	0.622	0.531
resnet50	15	0.005	Set 4	0.531	0.636	0.610
resnet50	20	0.005	Set 2	0.506	0.669	0.665
resnet50	25	0.001	Set 3	0.559	0.729	0.632

Table 5.1: Performance of model T_TV_FRCNN_resnet50_FPN across varying epochs and data splits with decaying learning rate (LR), all layers apart from last are fixed

F-RCNN resnet50

Model T_TV_FRCNN_resnet50_FPN was trained with an optimizer, based on stochastic gradient descent. An initial learning rate of 0.001, a momentum of 0.9, and a weight decay of 0.0005 were defined. The employed scheduler facilitates a constant reduction of learning rate by dividing it every third epoch by a divisor of 10. Reduced learning rates partially preserve the original weights of the model. 10 epochs were selected as a first training duration. In order to find the optimal hyperparameter settings, different epochs, train validation, and test splits, as well as learning rates, were analyzed. Their performance is shown in Table 5.1. Performance of models vary over the period of training, therefore the best performing model was tracked overall epochs and selected as a final model.

Discussion and Comparison Torchvision’s F-RCNN implementation, used in configuration T_TV_FRCNN_resnet50_FPN, does not support detailed logging of performance evolution in graphs during the training process. Therefore, only the final model T_TV_FRCNN_resnet50_FPN trained on Set 1 was analyzed. In this section sample predictions of the model are displayed, as well as common false positives are shown to display areas having poor results. The correct signature bounding box is displayed via a surrounding with a green border, the prediction is visualized with a red border.

In Figure 5.1, examples of the performance of the model can be observed. Single isolated (not surrounded by handwritten notes) signatures are detected with high precision, even multiple occurrences are located without issues. Common false positives are handwritten notes.

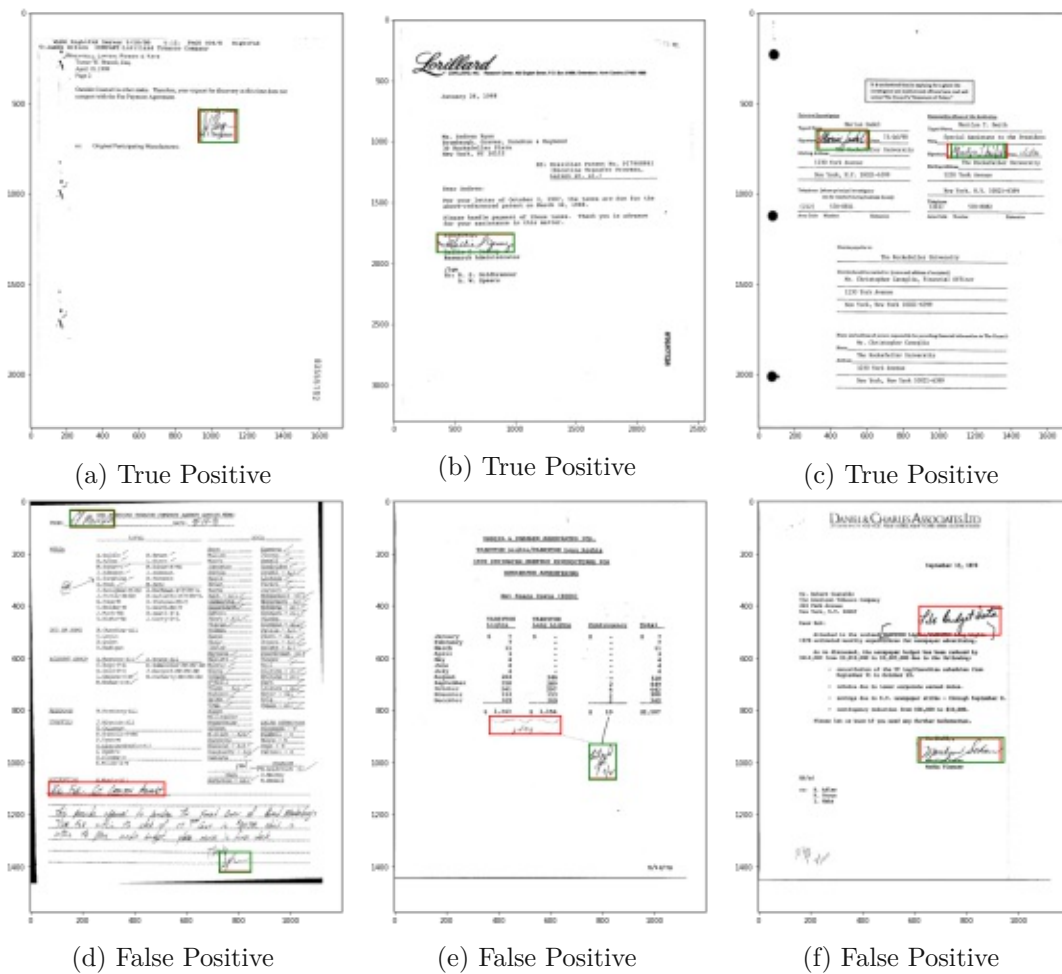


Figure 5.1: Selected predictions of model T_TV_FRCNN_resnet50_FPN Set 1

F-RCNN MobileNet V2

Model T_TV_FRCNN_mobilenetv2 was trained on set 4 with a learning rate of 0.001, a momentum of 0.9 and a weight decay of 0.0005 over 15 epochs. Employing mobilenet with custom anchor generator and roi pooler did not produce good results with an AP[0.50:0.95] of 0.297 with a recall of 0.372. Due to the low score, no further training with different data set splits was performed on this configuration.

5.1.2 Detectron

In this section, results of experiments performed with Detectron on the Tobacco 800 data set are presented and discussed. Furthermore, selected examples of predictions are illustrated and analyzed.

Backbone	IT	LR	Data set	AP[0.50:0.95]	AP[0.50]	R
resnext101	50000	0.00015	Set 1	0.604	0.791	0.662
resnext101	50000	0.00025	Set 2	0.604	0.760	0.650
resnext101	50000	0.00025	Set 3	0.565	0.735	0.653
resnext101	50000	0.00015	Set 4	0.552	0.680	0.598

Table 5.2: T_D_FRCNN_Resnext101_FPN over various iterations (IT), learning rates (LR) and different test train splits, stem + first stage were frozen

F-RCNN ResNext101

In another experiment, model T_D_FRCNN_Resnext101_FPN was trained on different data set splits with a batch size per image of 512 and varying learning rates with a single class classifier (only signatures were predicted). The optimizer used stochastic gradient descent. Intermediate states of the model were saved every 5000 iterations, out of these, the, on the validation data set, best performing model was selected at the end. The results of the experiments are displayed in Table 5.2.

Discussion and Comparison The best performing model of T_D_FRCNN_Resnext101_FPN was trained on Set 1 (44/10/36 split) for 50000 iterations. During training, the performance was evaluated every 5000 iterations which is shown in Figure 5.2. It can be observed that no major improvements of accuracy occurred after the first 5000 iterations and that the best performing model was trained for 24999 iterations. In the loss graph

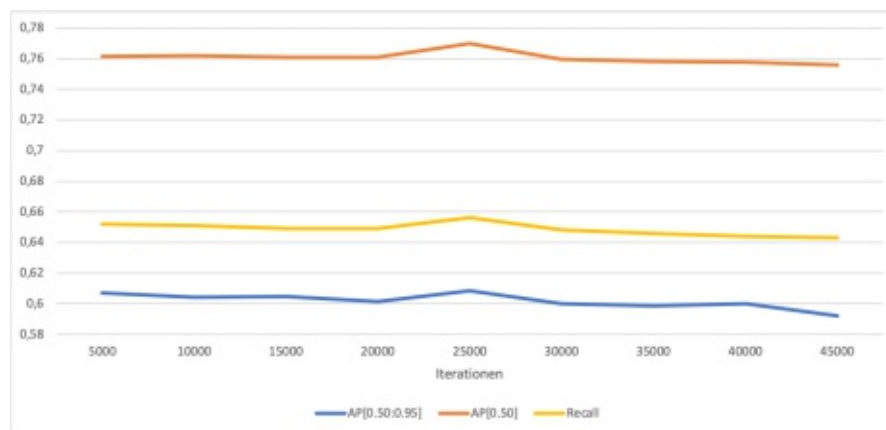


Figure 5.2: Average precision of T_D_FRCNN_Resnext101_FPN validation partition of Set 1

shown in Figure 5.3, a rapid decrease until the 5000 iteration mark is displayed. In combination with the average precision plots presented in Figure 5.2, this indicates that most performance gains were achieved within the first 5000 iterations. In Figure 5.4, a selection of predictions on the test data set is shown. It is observed that in contrast to

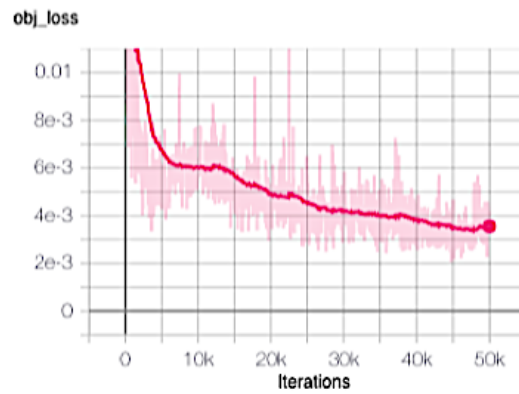
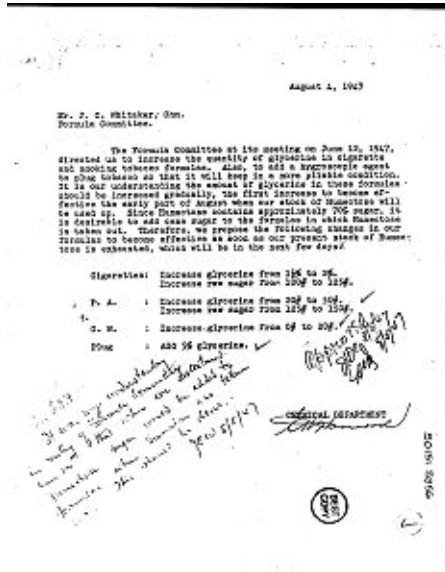


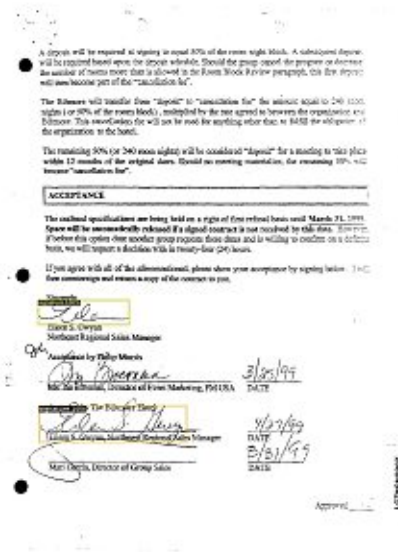
Figure 5.3: Loss graph of training process of T_D_FRCNN_Resnext101_FPN on Set 1

the best performing model of T_Y_YOLOv5x noisy backgrounds do not interfere with prediction performance (displayed in images a and b), while multiple signatures, as well as handwritten notes in close proximity to signatures, deteriorate performance. The best performing model of T_Y_YOLOv5x suffers from problems with handwritten notes as well.

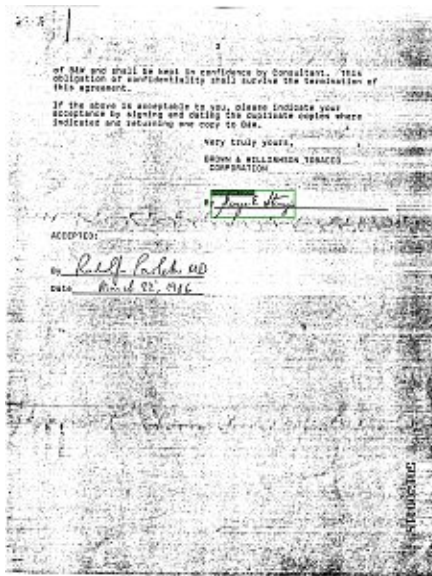
5. EXPERIMENT RESULTS



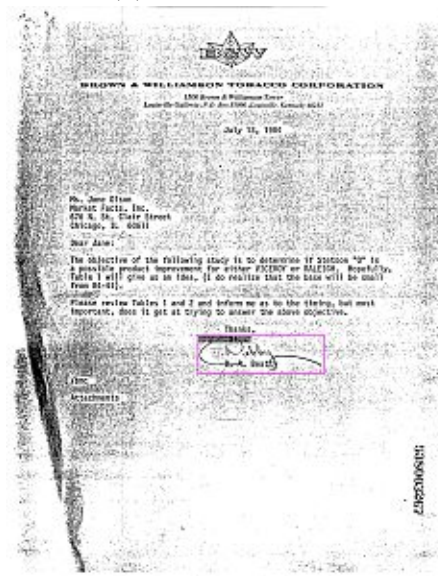
(a) False Negative



(b) partial False Negative



(c) True Positive



(d) True Positive

Figure 5.4: Selected predictions of T_D_FRCNN_Resnext101_FPN (44/10/36 split)

T_D_Retinanet_Resnext101

Further experiments were performed with model T_D_Retinanet_Resnext101 with varying frozen layers over different data set splits. The results are displayed in Table 5.3.

Backbone	frozen layers	LR	IT	Data set	AP[0.50:0.95]	AP[0.50]	R
resnext101	stem+first stage	0.00015	50000	Set 1	0.605	0.792	0.667
resnext101	stem+first stage	0.00025	50000	Set 2	0.578	0.708	0.590
resnext101	stem+first three stages	0.00025	50000	Set 3	0.532	0.685	0.523
resnext101	stem+first three stages	0.00015	50000	Set 4	0.552	0.720	0.598

Table 5.3: T_D_Retinanet_Resnext101 over various iterations (IT), learning rates (LR) and different test train splits

Discussion and Comparison The best performing model of T_D_Retinanet_Resnext101 was trained on set 1 (44/10/36 split). The training process of the model is illustrated in Figure 5.5. The greatest performance increase was achieved in the first 5000 iterations as seen in loss graphs, afterwards gains slowed down. Due to employing the same backbone, the training graphs of T_D_FRCNN_Resnext101_FPN and T_D_Retinanet_Resnext101 show similar behavior.

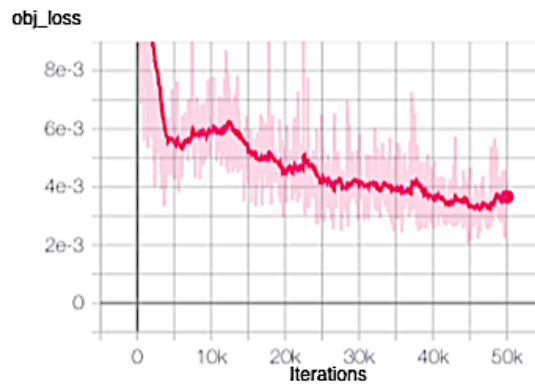


Figure 5.5: Loss graph of training process of model T_D_Retinanet_Resnext101 on Set 1

In Figure 5.6, selected predictions from the model are shown. The signature in the image could not be detected in contrast to previously analyzed models. In sample b and d, not all signatures were recognized, which indicates problems with detecting multiple signatures. Furthermore, it is the only model that misclassifies a date as a signature. Noisy document backgrounds do not seem to interfere with prediction performance.

Model	Data set	LR	Epochs	AP[0.50:0.95]	AP[0.50]	R
YOLOv5	Set 1	0.12	1000	0.697	0.972	0.971
YOLOv5	Set 2	0.12	1000	0.691	0.950	0.950
YOLOv5	Set 3	0.12	1000	0.690	0.963	0.897
YOLOv5	Set 4	0.12	1000	0.692	0.941	0.942

Table 5.4: Performance of model T_Y_YOLOv5x on different data set splits

T_Y_YOLOv5x

Experiments were conducted with the model T_Y_YOLOv5x with a batch size of 16 across various data set splits. The optimizer was based on stochastic gradient descent. The performance of the models trained in the experiments can be observed in Table 5.4.

Discussion and Comparison The best performing model of the YOLOv5 Setup (T_Y_YOLOv5x) was trained on set 1, with a training, validation, and testing split of 44%, 10%, and 36%. In Figure 5.7, the development of its accuracy, recall, and precision regarding the validation set over the training period can be observed. Ultralytics YOLOv5 implementation logs more details during the training process compared to other assessed frameworks. As it is the best performing model, understanding its training process is of great significance. Therefore, this section presents more plots based on the additional available data compared to the discussion section of Detectron’s Retinanet and Faster R-CNN. In the graph displaying the development of the mean AP[0.50:0.95], one can observe, that after 500 epochs only an insignificant performance increase was achieved. Figure 5.8 displays loss over the training period. A stagnation of general intersection over union loss (gIoUL) can be observed in the training data set after epoch 800, in contrast to gIoUL of the validation data set, which slowly starts to increase after epoch 800. This phenomenon indicates beginning overfitting on the training data set, resulting in deteriorating performance on the validation data set. Combined with the observed stagnation of the mean average precision metric, it is concluded that further training would not lead to better model performance.

In Figure 5.9, a series of selected predictions of the model T_Y_YOLOv5x (44/10/36 split) is displayed. Detecting multiple signatures in a document is demonstrated. On the other hand, noisy documents, as well as a large number of handwritten notes in close proximity to signatures, seem to increase the difficulty for the model.

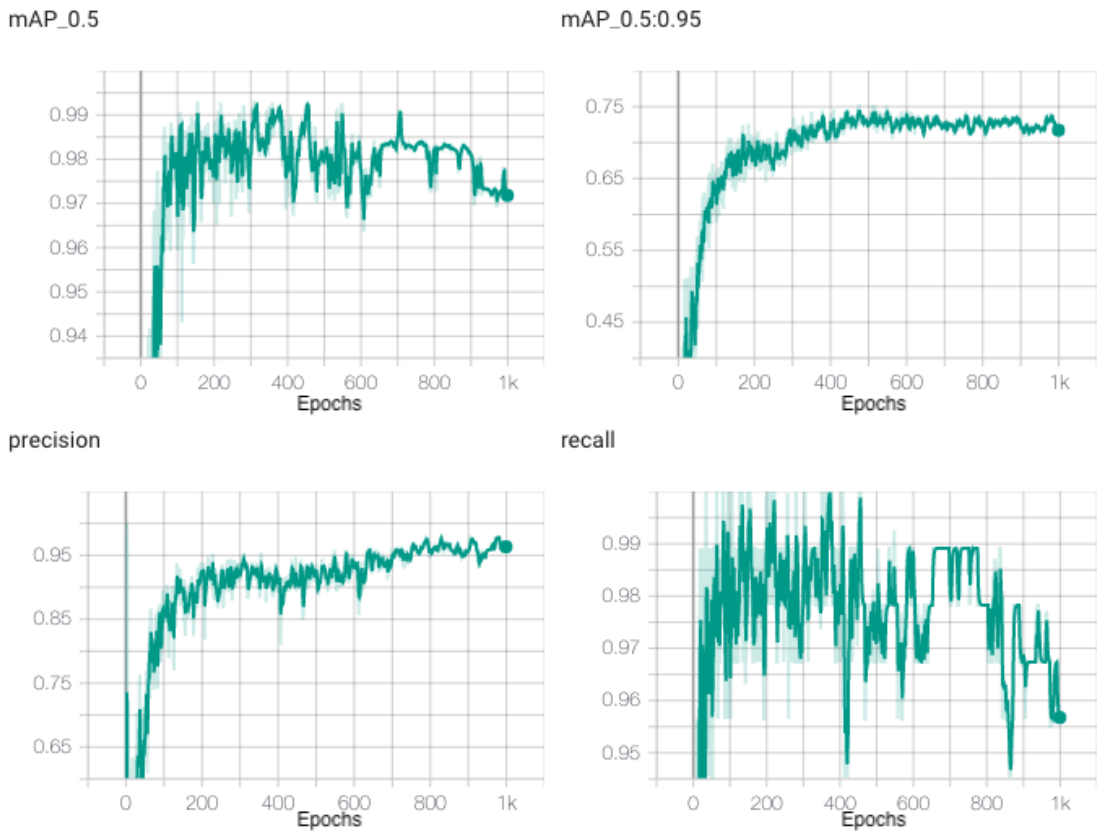
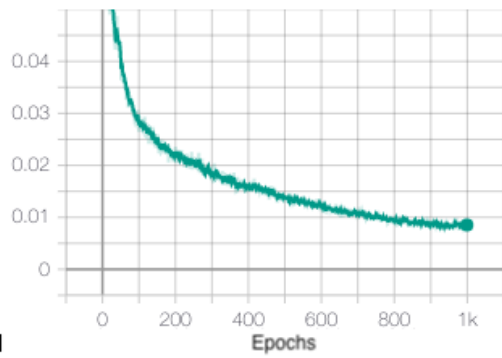


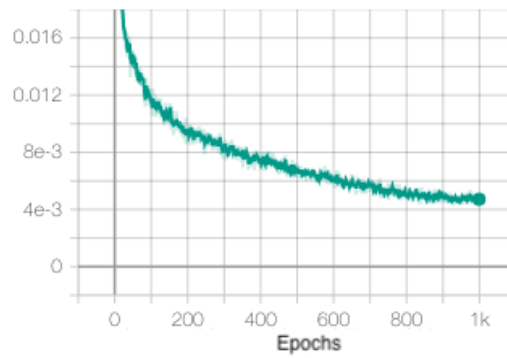
Figure 5.7: Development of performance of model T_Y_YOLOv5x on set 1

train

giou_loss

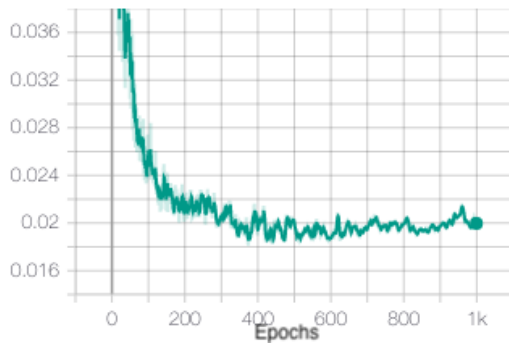


obj_loss



val

giou_loss



obj_loss

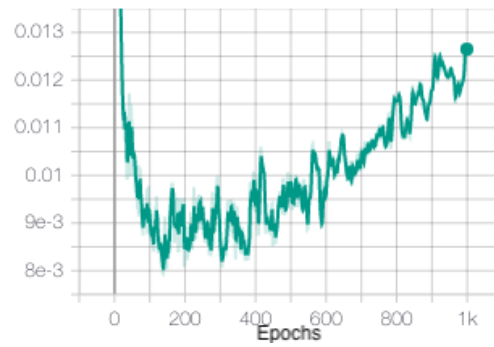
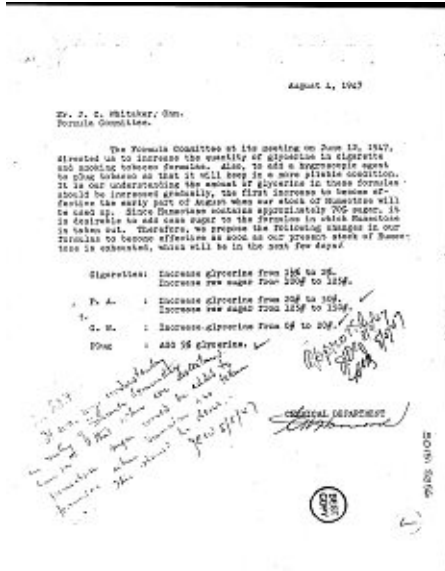
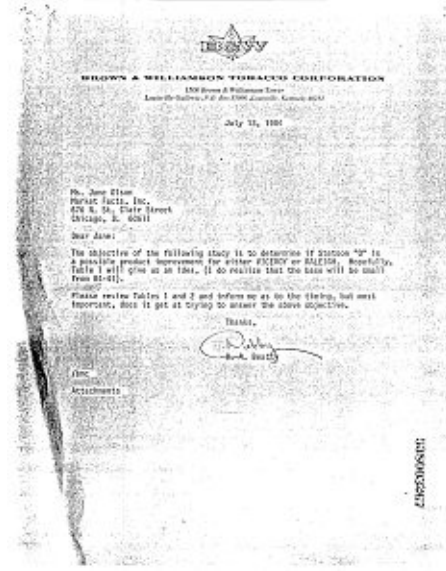


Figure 5.8: Loss of model T_Y_YOLOv5x on set 1 on training and validation set

5. EXPERIMENT RESULTS



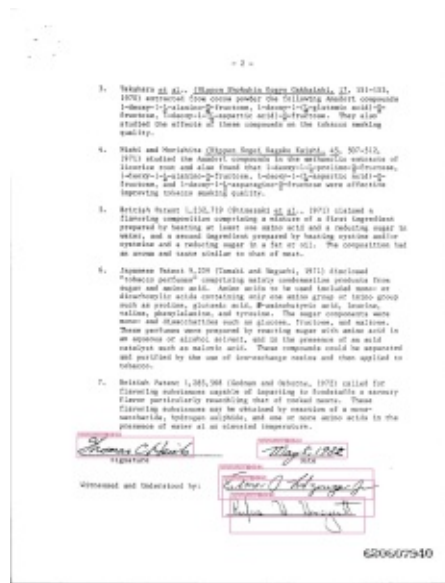
(a) False Negative



(b) False Negative



(c) True Positive



(d) True Positive

Figure 5.9: Selected predictions of T_Y_YOLOv5x (44/10/36 split)

5.1.4 Conclusion of Tobacco 800 models

Table 5.5 shows performance on Set 1 across different model architectures, the best performing configuration was T_Y_YOLOv5x, its performance is therefore used for comparison with the state of the art scores, found in different publications. T_D_Retinanet_Resnext101 and T_D_FRCNN_Resnext101_FPN offer similar performance, this is potentially caused by them relying on the same backbone architecture. The worst performance in this comparison offers the model T_TV_FRCNN_resnet50_FPN. The models T_D_FRCNN_Resnext101_FPN and T_TV_FRCNN_resnet50_FPN differ in performance due to the inferior backbone, Resnet 50 instead to ResNeXt101.

Furthermore, a comparison of performance to approaches in state of the art literature was performed. The default metric in most object detection competitions, as well as in all publications analyzing the Tobacco 800 data set, is an AP[0.50]. Using this looser criterion an average precision of 0.972 was achieved, with a recall of 0.971, which surpasses all deep learning based approaches found in related work. For comparison, the best performing deep learning approach Sharma et al. [51] achieved a mean average precision of 0.882 and a recall 0.884 with Faster R-CNN in a 30% train, 10% validation, and 60% testing data set split. The leading feature extraction based approach [62] [62], achieved an accuracy of 0.928 on the complete data set. The proposed YOLOv5 model of T_Y_YOLOv5x trained on Set 1 surpasses this score as well. However, a fair comparison is not possible, due to not published file names used for train, validation, and test split.

Model	Data set	AP IoU[0.50:0.95]	AP IoU[0.50]	R
T_Y_YOLOv5x	Set 1	0.697	0.972	0.971
T_D_Retinanet_Resnext101	Set 1	0.605	0.792	0.667
T_D_FRCNN_Resnext101_FPN	Set 1	0.604	0.791	0.662
T_TV_FRCNN_resnet50_FPN a	Set 1	0.547	0.622	0.531

Table 5.5: Best performing models

5.2 Experiments on City of Vienna data set

Analogously to the discussion section of the Tobacco 800 data set, the best performing model and data set split of each experiment configuration was selected for analysis and comparison with their counterparts from other architectures. Out of these, the best performing model was selected and used to implement a signature detection system accessible to the end-user, this system is introduced in subsequent chapters. Due to privacy law, predictions of the models can only be described textually, but not illustrated as in the Tobacco 800 discussion section.

5.2.1 YOLOv5

In this section, results of experiments performed with YOLOv5 on the City of Vienna data set are presented and discussed. Furthermore, the training process is analyzed in detail and insights given.

W_Y_YOLOv5x

Model W_Y_YOLOv5x was trained for 3000 epochs with an initial learning rate of 0.0032 and a final learning rate of 0.12, a momentum of 0.843 and a weight decay of 0.00036 on varying data set splits. The performance of the models trained in the experiments can be observed in Table 5.6. As previously mentioned, the YOLOv5 implementation logs more data during the training process of a model. As the YOLOv5x model is the best performing, understanding its training process in depth is important. Therefore, more graphs were plotted by employing these additionally logged metrics compared to other architectures.

Discussion and Comparison The best performing model of the YOLOv5 configuration W_Y_YOLOv5x was trained on set (W) 4. Despite the model trained on set (W) 2 offering a higher AP[0.50], the stronger AP[0.50:0.95] is prioritized. Due to the low number of training samples, a 20% bigger training set (the difference between set (W) 2 and set (2) 4) leads to a better generalizing model. The training process is displayed in Figure 5.10. In the loss graph, displayed in Figure 5.11, it is observed that while the training loss continues to decrease, general intersection over union loss regarding the validation data set starts to increase after about 2200 epoch and object loss increases

Epochs	LR	data set	AP[0.50:0.95]	AP[0.50]	R
3000	0.12	Set (W) 1	0.456	0.821	0.869
3000	0.12	Set (W) 2	0.499	0.891	0.922
3000	0.12	Set (W) 3	0.443	0.809	0.848
3000	0.12	Set (W) 4	0.623	0.875	0.875

Table 5.6: Performance of model W_Y_YOLOv5x

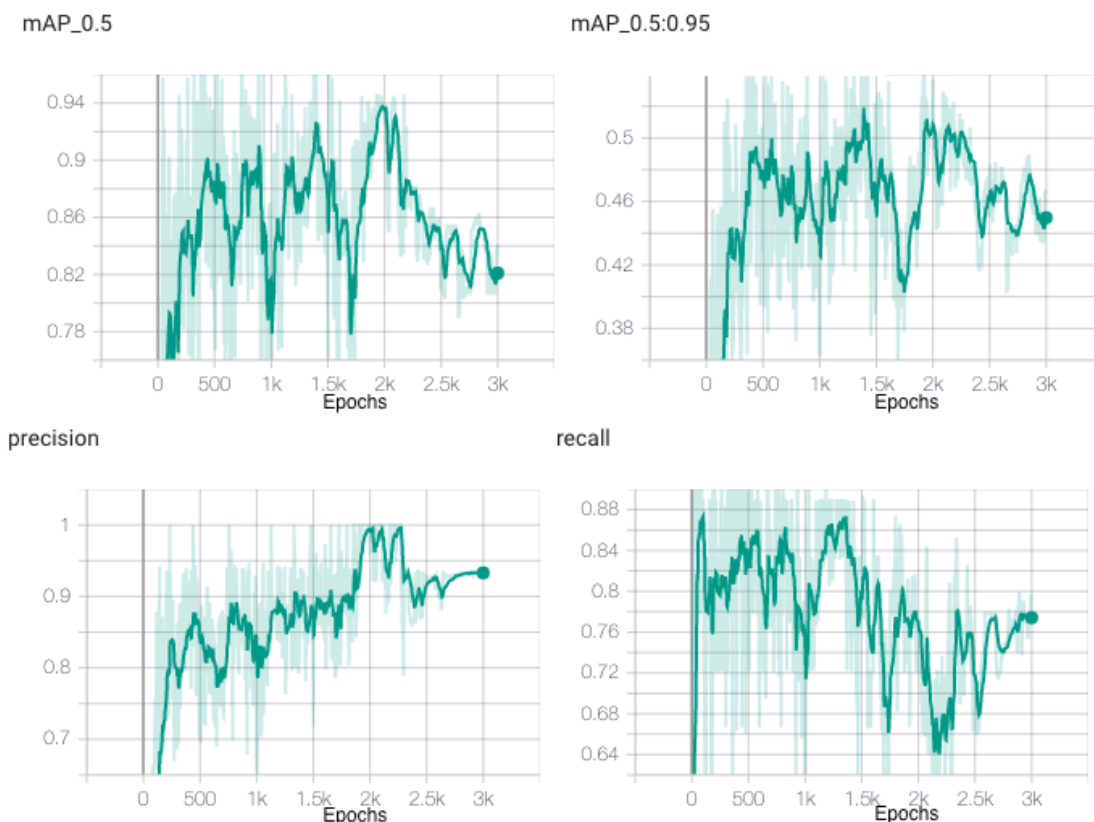


Figure 5.10: Performance evolution of W_Y_YOLOv5x during training

after 1500 epochs. This indicates overfitting of the model to the training data and that further training will not result in a better performing model. The best performing model among all training epochs was selected for final evaluation with the test data set. Multiple signatures in one document, as well as signatures overlapping stamps, were detected without any problem. Dates and handwritten notes were not misclassified as signatures.

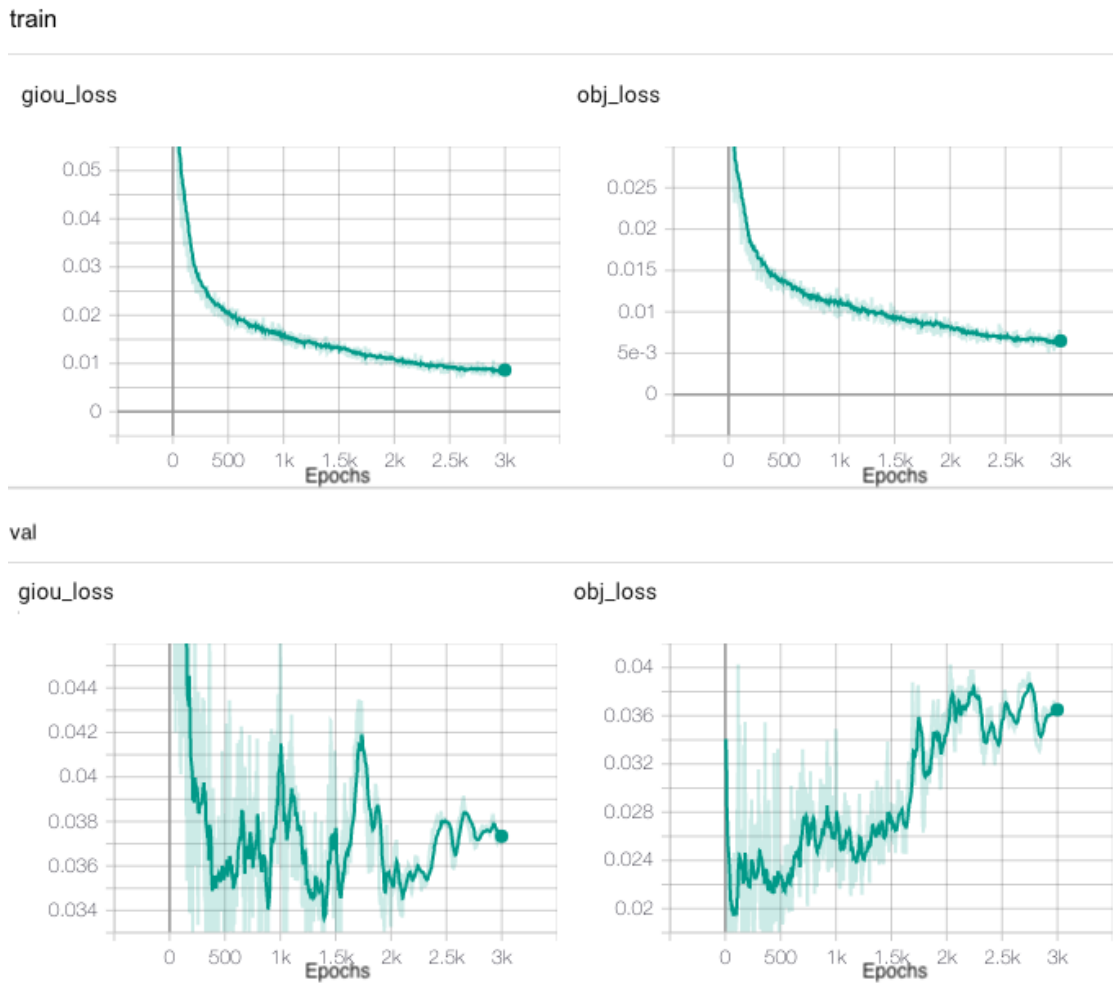


Figure 5.11: Loss on training and validation data set of W_Y_YOLOv5x during training

W_Y_YOLOv5x_TF

In another experiment, model W_Y_YOLOv5x_TF was trained for 1500 epochs with an initial learning rate of 0.0032 and a final learning rate of 0.12, a momentum of 0.843 and a weight decay of 0.00036 on varying data set splits. As transfer learning basemodel T_Y_YOLOv5x was used, it was trained on set 1 for 1000 epochs. The results are displayed in Table 5.7.

Discussion and Comparison The best performing model of the YOLOv5 configuration W_Y_YOLOv5x_TF was trained on set (W) 4. In this experiment transfer learning, building upon the best performing Tobacco 800 model was employed. The training process is displayed in Figure 5.12 and 5.13. One can observe that an AP at an IoU greater than 0.5 of 0.70 was achieved after the first epoch, this indicates that

Basemodel	Epochs	LR	Data set	AP[0.50:0.95]	AP[0.50]	R
T_Y_YOLOv5x	1500	0.12	Set (W) 2	0.469	0.829	0.875
T_Y_YOLOv5x	1500	0.12	Set (W) 4	0.573	0.799	0.812

Table 5.7: Performance of model W_Y_YOLOv5x_TF on different data set splits

basic signature detection is transferable across different data sets. Within the first 500

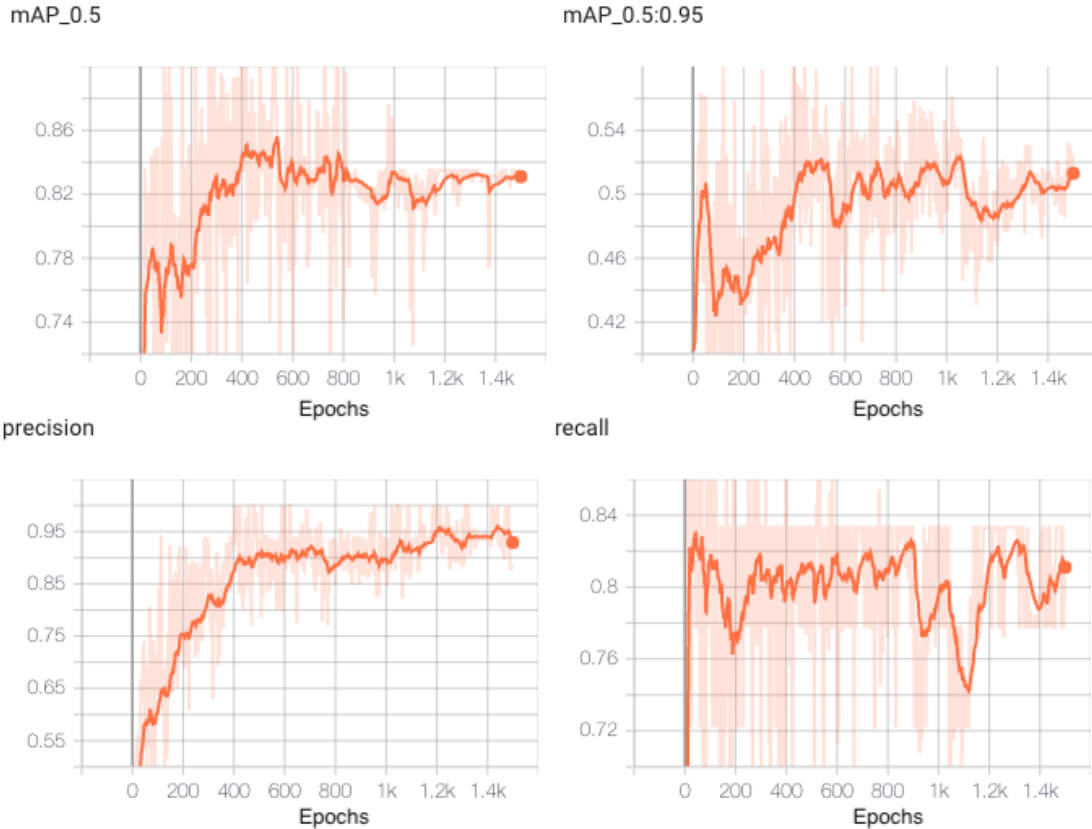


Figure 5.12: Performance evolution of W_Y_YOLOv5x_TF during training

epochs, high AP values of about 0.8 on the training data set were achieved. As displayed in Figure 5.14, W_Y_YOLOv5x_TF reaches higher AP significantly faster than W_Y_YOLOv5x, which uses a generic transfer learning base, compared to the task-specific base of W_Y_YOLOv5x_TF. Despite W_Y_YOLOv5x_TF leading up to the 500 epoch mark, W_Y_YOLOv5x takes over afterwards and achieves a higher accuracy over the whole training period. W_Y_YOLOv5x_TF plateaus earlier at around 500 epochs at an AP of 0.82. It is concluded that some of the features learned from the Tobacco 800 data set are transferable to the City of Vienna data set and accelerate the training process at the beginning, but also prevent learning specialized features that would be

5. EXPERIMENT RESULTS

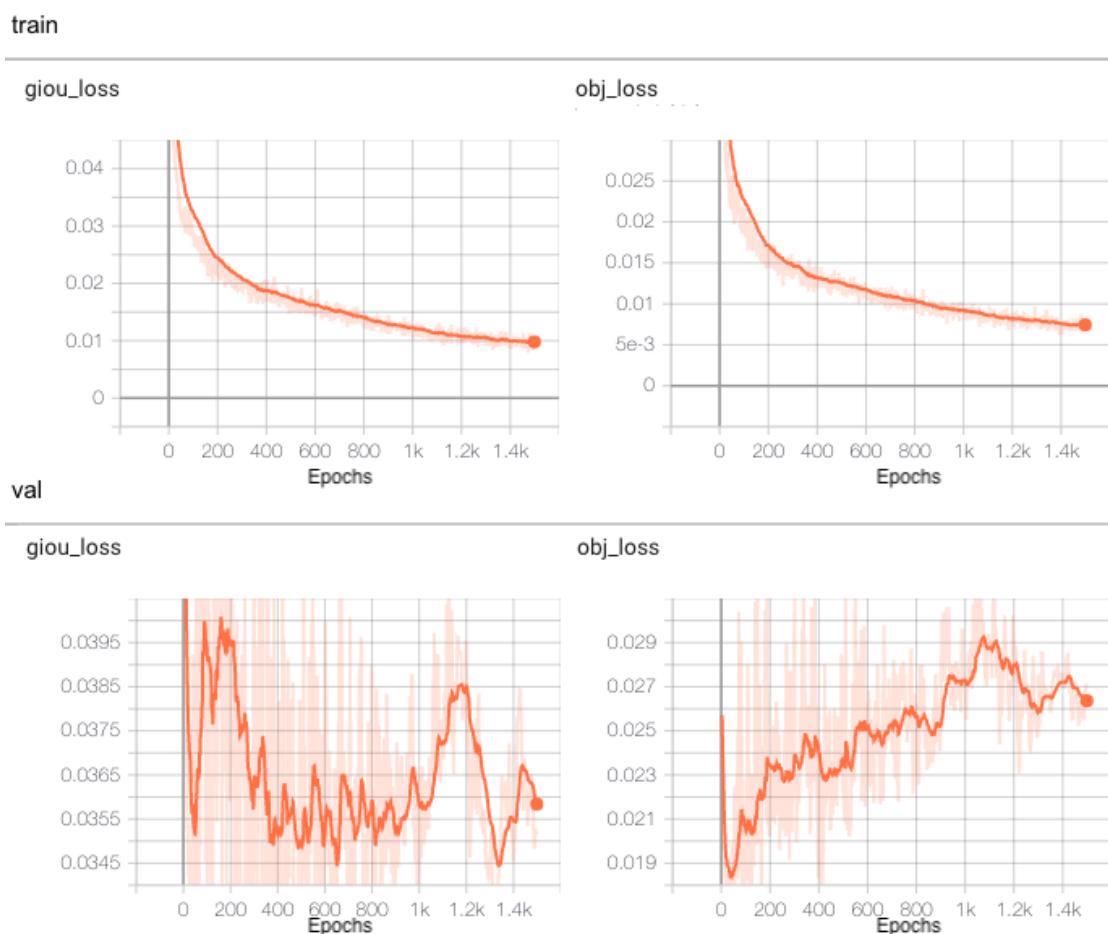


Figure 5.13: Loss on training and validation data set of W_Y_YOLOv5x_TF during training

required to achieve better AP in the whole training process. Therefore, it is concluded that task-specific transfer learning is not beneficial when dealing with small data sets. Both the City of Vienna data set and the Tobacco 800 data set are small compared to the COCO data set. It has to be considered that the Tobacco 800 transfer learning base was trained on COCO data set, therefore a reduced learning rate was employed when training with Tobacco 800. As a result, not all features learned from COCO were overwritten and not the whole network was adapted to Tobacco 800, but only fine-tuned. With a bigger initial data set, training a CNN from scratch on signatures and then fine-tuning it on the City of Vienna data set might have better results. As expected, signatures not overlapping stamps were detected flawlessly by the model of W_Y_YOLOv5x_TF, but not all signatures overlapping stamps could be detected. This is potentially because the Tobacco 800 data set, which served as a base for training, only contains isolated signatures.

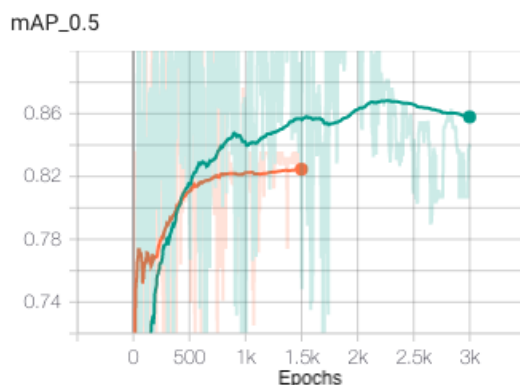


Figure 5.14: Accuracy of W_Y_YOLOv5x_TF (orange) and W_Y_YOLOv5x (green) during training

5.2.2 Detectron

In this section, results of experiments performed with Detectron on the City of Vienna data set are presented and discussed. Furthermore, the training process is analyzed in detail and insights given.

W_D_FRCNN_Resnext101_FPN

In this experiment, models were trained on varying data set splits. The results of these experiments can be observed in Table 5.8.

Discussion and Comparison The best performing model W_D_FRCNN_Resnext101_FPN was trained on set (W) 4 for 50000 iterations with default anchors. The average precision plot in Figure 5.15 for the validation data set shows a performance peak after around 5000 iterations, both in AP[0.50:0.95] and AP[0.50]. Considering the plot of validation and training loss, one can observe a minimum after around 5000 iterations, afterwards validation loss increases, while training loss steadily decreases over the course of training. In combination with the high average precision at this stage of training, it is concluded that the model at this training level offers the best performance. Not all signatures were detected by the model, it had problems both in areas where stamps

LR	IT	Data set	Anchors	AP[0.50:0.95]	AP[0.50]	R
0.00015	50000	Set (W) 2	default	0.349	0.550	0.469
0.00015	50000	Set (W) 3	default	0.469	0.650	0.576

Table 5.8: Performance of model W_D_FRCNN_Resnext101_FPN (default anchors) on different data set splits

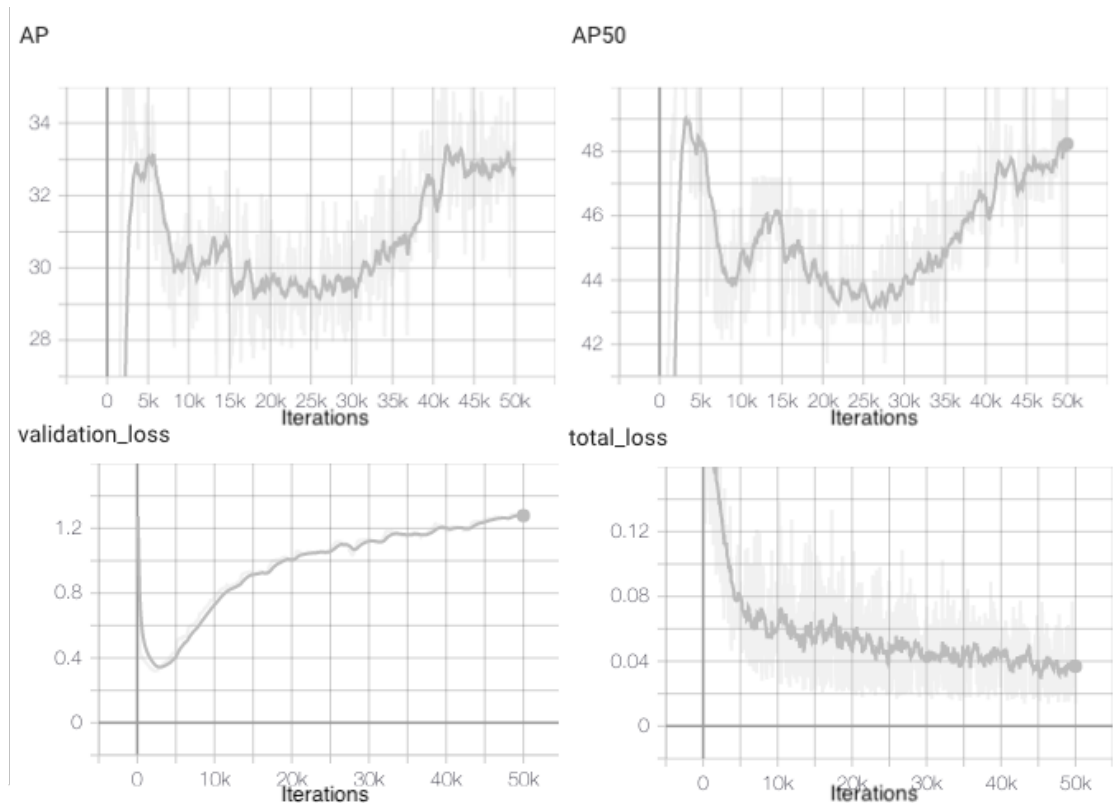


Figure 5.15: Training process of W_D_FRCNN_Resnext101_FPN set (W) 4

LR	IT	data set	Anchors	AP[0.50:0.95]	AP[0.50]	R
0.00015	50000	Set (W) 3	custom	0.398	0.675	0.505
0.00015	50000	Set (W) 4	custom	0.352	0.554	0.455

Table 5.9: Performance of model W_D_FRCNN_Resnext101_CA (custom anchors) on different data set splits

overlap signatures as well where multiple signatures were located in close proximity to each other.

W_D_FRCNN_Resnext101_CA

This model was trained with custom anchors, computed via the methodology shown in Chapter 4 for signatures in the City of Vienna data set, on varying data set splits. A comprehension of results of these experiments can be observed in Table 5.9.

Discussion and Comparison The best performing model of W_D_FRCNN_Resnext101_CA was trained on set (W) 4 for 50000 iterations with custom anchors. The performance

evolution of the training process is displayed in Figure 5.16. The best-performing of

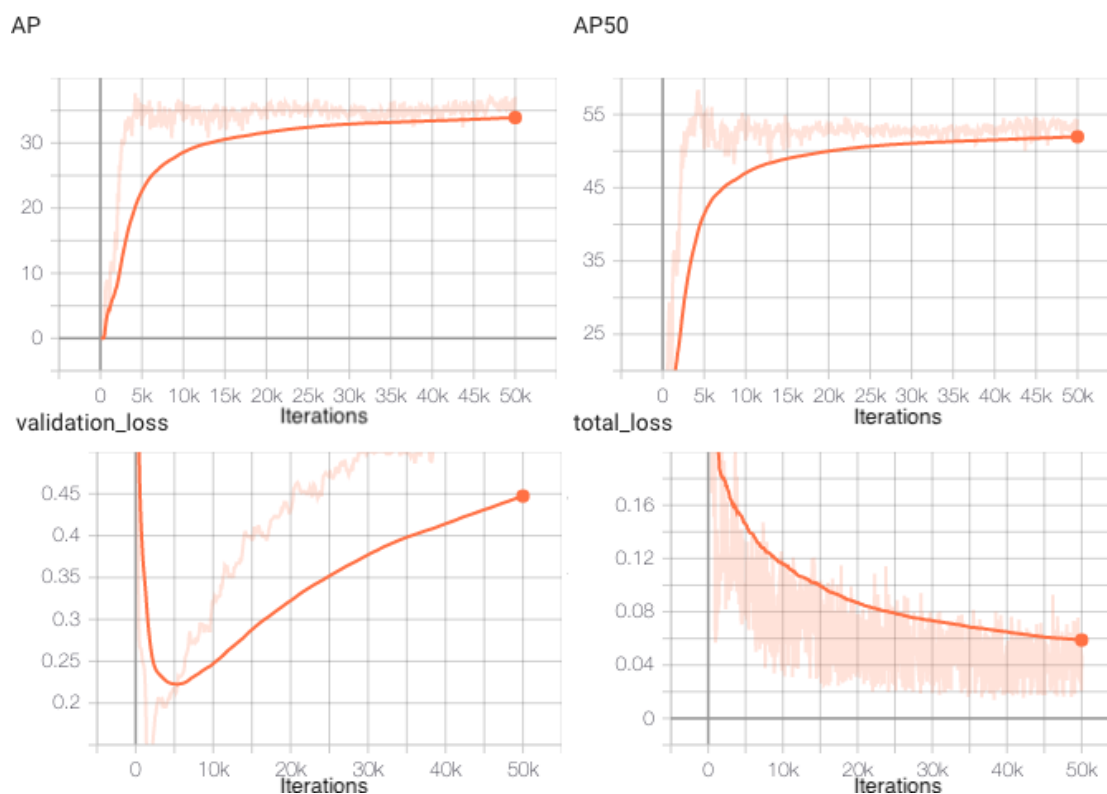


Figure 5.16: Training process of W_D_FRCNN_Resnext101_CA set (W) 4

the models W_D_FRCNN_Resnext101_FPN and W_D_FRCNN_Resnext101_CA state can be found at around 5000 iterations. In the combined plot illustrated in Figure 5.17 of the AP[0.50] one can observe that the model with custom anchors performed worse in initial the 5000 iterations mark, afterwards performance increased, while the model with default anchors plateaued. With regard to prediction performance, the model W_D_FRCNN_Resnext101_CA achieved a higher score on the validation set. But prediction performance overall was weaker on the test set compared to W_D_FRCNN_Resnext101_FPN. In areas where both models could detect a signature, the model with custom anchors detected better fitting bounding boxes.

MIX_D_FRCNN_Resnext101_FPN

Model MIX_D_FRCNN_Resnext101_FPN trained on set (MIX) 3 for 200000 with a learning rate of 0.00025 reached an AP[0.50:0.95] of 0.42, an AP[0.50] of 0.68 with an average recall of 0.5 was achieved, due to low recall, training with further sets was not performed.

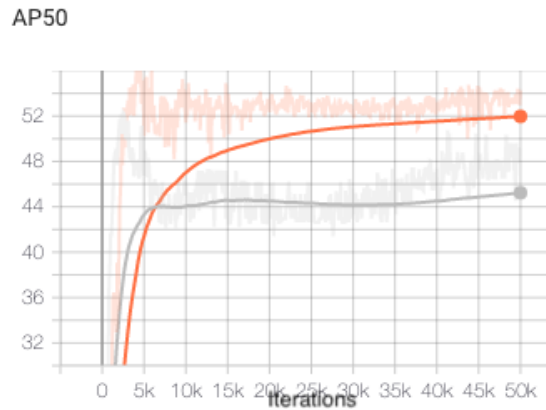


Figure 5.17: AP of W_D_FRCNN_Resnext101_CA (orange) and W_D_FRCNN_Resnext101_FPN (gray) during training

LR	IT	Data set	Basemodel	AP[0.50:0.95]	AP[0.50]	R
0.0001	25000	Set (W) 2	Tobacco 800	0.343	0.579	0.423
0.0001	25000	Set (W) 4	Tobacco 800	0.569	0.768	0.600

Table 5.10: Performance of model W_D_FRCNN_Resnext101_FPN_TF on different data set splits

Discussion and Comparison The performance of MIX_D_FRCNN_Resnext101_FPN when evaluated only on City of Vienna documents is sub-par, this is likely due to the City of Vienna data set generally posing more difficult instances such as overlapping stamps for signature detection, but the network being trained on a data set, in which the vast majority of images contains only isolated signatures (75% from Tobacco 800 and 25% from City of Vienna).

W_D_FRCNN_Resnext101_FPN_TF

Model W_D_FRCNN_Resnext101_FPN_TF was trained on varying data set splits with a Tobacco 800 model as a transfer learning base, i.e. not a generic COCO pre-trained model was used for training, but a model, that had already learned signature specific features. The results of these experiments can be observed in Table 5.10.

Discussion and Comparison The best performing model of W_D_FRCNN_Resnext101_TF was trained on set (W) 4 for 25000 iterations with the best performing Tobacco 800 Faster R-CNN model as transfer learning base with default anchors. The performance evolution of the model over the course of the training process is displayed in Figure 5.18. After about 3000 iterations an AP[0.50] of 0.53, which is also the point where the minimum in the validation loss graph occurs. Afterwards validation loss begins to rise,

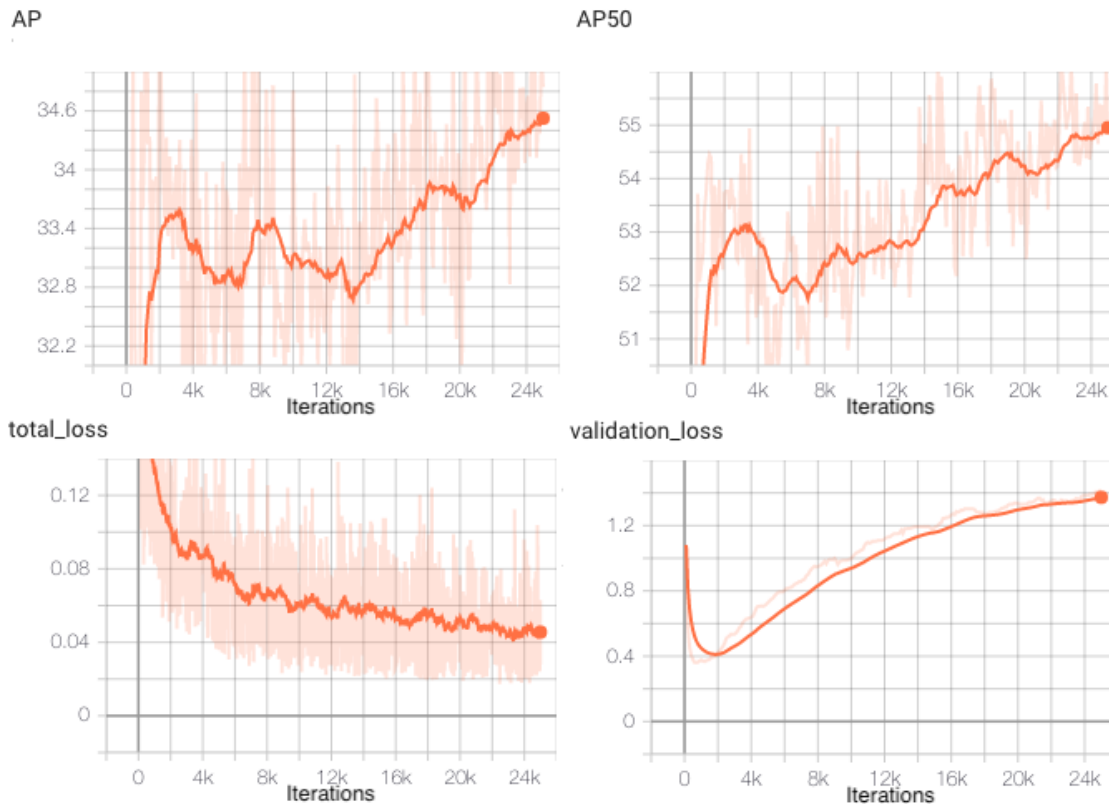


Figure 5.18: Training process of W_D_FRCNN_Resnext101_TF

while training loss slowly continues decreasing. This potentially indicates overfitting to the training data set. After the training process, the best performing model was selected, therefore an overfitted model can be ruled out. In Figure 5.19 the training process of W_D_FRCNN_Resnext101_TF and W_D_FRCNN_Resnext101_FPN is compared. In this case task-specific transfer learning clearly outperforms the COCO pre-trained model. In regard to actual predictions W_D_FRCNN_Resnext101_TF performed better in multi-signature detection and detection of overlapped signatures, but not as well as W_Y_YOLOv5x_TF and W_Y_YOLOv5x.

W_D_Retinanet_Resnext101

The models of experiment W_D_Retinanet_Resnext101 were trained on varying data set splits. The results of this experiment are displayed in Table 5.11.

Discussion and Comparison The best performing model of W_D_Retinanet_Resnext101 was trained on set (W) 4 for 25000 iterations. Figure 5.20 illustrates the performance evolution during the training process. The model reaches an intermediate peak in AP[0.50:0.95] at around 4000 iterations and plateaus with only small outliers

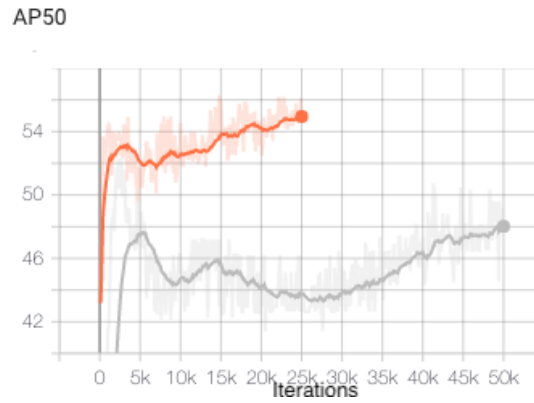


Figure 5.19: Training process of W_D_FRCNN_Resnext101_TF (orange) and W_D_FRCNN_Resnext101_FPN (gray) on set (W) 4

LR	Iterations	Data set	AP[0.50:0.95]	AP[0.50]	R
0.00015	25000	Set (W) 1	0.349	0.529	0.428
0.00015	25000	Set (W) 2	0.510	0.696	0.575
0.00015	25000	Set (W) 3	0.346	0.514	0.406
0.00015	25000	Set (W) 4	0.280	0.440	0.392

Table 5.11: Model W_D_Retinanet_Resnext101 performance across various data set splits

Model	Data set	AP[0.50:0.95]	AP[0.50]	R
W_Y_YOLOv5x	Set (W) 4	0.623	0.875	0.875
W_D_FRCNN_Resnext101_TF	Set (W) 4	0.569	0.768	0.600
W_D_Retinanet_Resnext101	Set (W) 4	0.510	0.696	0.575
W_D_FRCNN_Resnext101_FPN	Set (W) 4	0.469	0.650	0.576
W_D_FRCNN_Resnext101_CA	Set (W) 4	0.398	0.675	0.505

Table 5.12: Comparison of best performing models

after 8000 iterations. Training loss decreases over the whole course of the training process, while validation loss starts to increase after 2000 iterations, this potentially indicates overfitting of the model to the training data set. In regard of predictions, the model W_D_Retinanet_Resnext101 performs slightly better than the Faster R-CNN models.

5.2.3 Conclusion of City of Vienna data set

In Table 5.12, the best performing models of each experiment are listed for comparison. Model W_Y_YOLOv5x trained on Set (W) 4 performed better than every other model. In general, it can be observed that the best performing model of each experiment was

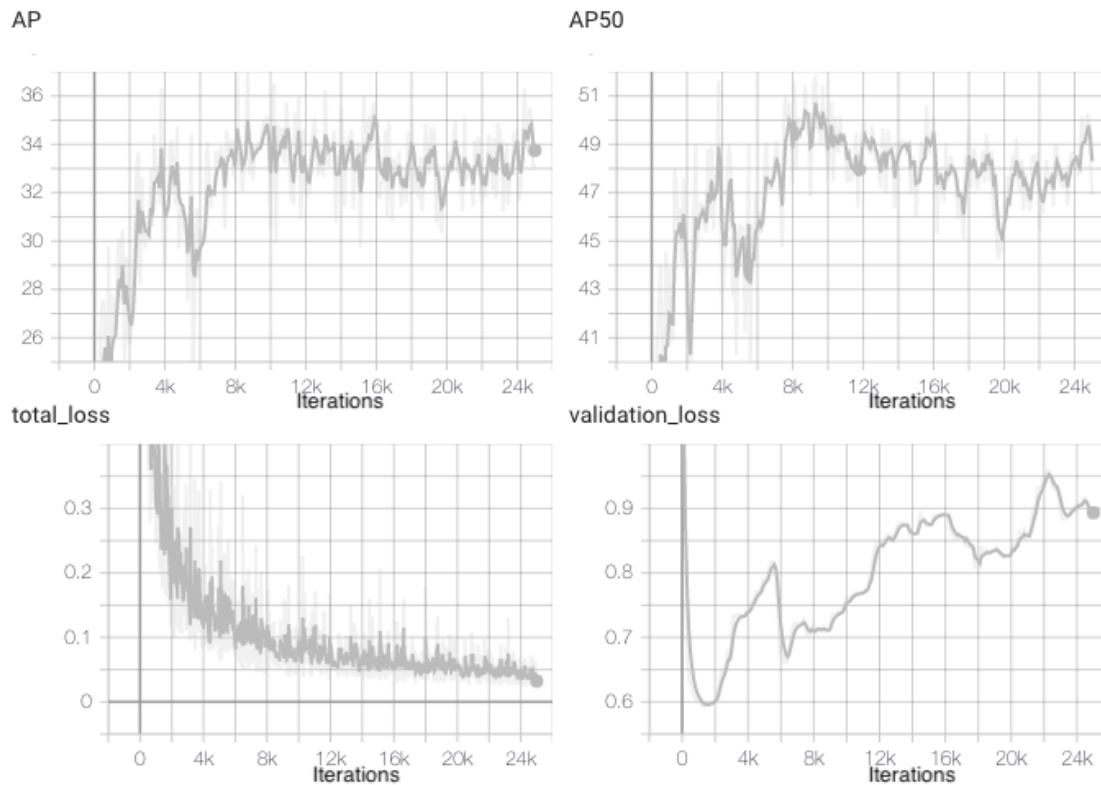


Figure 5.20: Training process of W_D_Retinanet_Resnext101 set (W) 4

trained on Set (W) 4. The small portion of files used for testing and validation in this split allows using most of the data for training, this is important for a good generalizing model especially in small data sets such as the City of Vienna building applications data set. Employing a single-stage object detector for predicting signatures offers faster inference time compared to two-stage object detectors such as Faster R-CNN. Not only inference, but training can be performed more efficiently as well.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Conclusion

In this chapter, the results of this work are aggregated and analyzed. Furthermore, the limitations of the proposed signature detection system are summarized and contributions to the state of the art mentioned. Finally, suggestions for future research in the area of signature detection are given.

6.1 Research Questions

The goal of this thesis was to develop an automated system to detect signatures in scanned document images. The system should enable civil servants at the municipality MA37 (Baupolizei) of the City of Vienna to work more efficiently. Checking every page for signatures in a building application compiled from hundreds of pages was a tedious, time consuming and error-prone task, the proposed signature detection system solves these problems by automatically extracting signatures from the document.

RQ1: What is the best approach to detect handwritten signatures from varying, noisy backgrounds with machine learning? The experiments in this thesis have shown that convolutional neural networks are highly capable of detecting signatures from complicated backgrounds such as scanned document images with a variety of overlapping visual entities with high precision and accuracy. While many different types of models offered good performance, all of them were surpassed by YOLOv5 in terms of accuracy, training time, and inference speed. It was shown that in 2020 one stage object detectors such as YOLOv5 offer better performance than traditional counterparts such as Faster R-CNN. One stage object detectors are algorithms that perform detection and classification of objects in one stage. This is not limited to YOLOv5. Retinanet, another one stage object detector, also provides better performance than Faster R-CNN, although the gap in achieved accuracy was not as large as between YOLOv5 and Faster R-CNN. This was demonstrated by performing experiments on two different data sets,

Tobacco 800 and City of Vienna building applications. In these two data sets, signatures had to be detected, the best performing model was based on YOLOv5 in both cases. Hyperparameters were optimized in all three types of networks, YOLOv5, Retinanet and Faster R-CNN, but the YOLOv5 models T_Y_YOLOv5x and W_Y_YOLOv5x achieved the best overall score with AP[0.50] of 0.875 and recall of 0.875 in the City of Vienna building applications experiment and AP[0.50] of 0.972 and recall of 0.971 in the Tobacco 800 experiment.

RQ2: What effect has transfer-learning on model accuracy in handwritten signature detection? Experiments in this thesis have shown that transfer learning increases model accuracy in handwritten signature detection compared to training from scratch. In fact, training a convolutional neural network for signature detection on a data set as small as the City of Vienna building applications data set, with only about 250 instances from scratch, does not produce any usable results in terms of a well generalizing model within a reasonable number of training epochs. Relying on a two step transfer learning approach, i.e. using a COCO pre-trained model as a base for training it with the Tobacco 800 data set and then fine tuning it again on the City of Vienna building applications data set did only result in small performance gains in one case, in other experiments no gains or even diminished precision was measured. Performance of model W_D_FRCNN_ResNeXt101_TF, which was fine tuned from the Tobacco 800 trained model T_D_FRCNN_ResNeXt101_FPN could be improved. Its accuracy could be increased by 10% and its recall by 3% compared to the pre-trained model variant W_D_FRCNN_ResNeXt101_FPN. On the other hand performance deteriorated in YOLOv5 Models. Model W_Y_YOLOv5x_TF achieved 3% worse accuracy and 6% worse recall.

RQ3: How can human annotators be supported with transfer learning? Annotating images in data sets with bounding boxes is a time consuming and tedious task. This task can be accelerated and performed more efficiently with a half automated approach. Experiments in this thesis have shown that provided a similar data set exists, it is beneficial to train a model first on the existing data set and subsequently use the model to predict the objects of interest, e.g. signatures, in the new data set and save the results as bounding boxes. These bounding boxes are then used as base for further annotations by manual annotators. After the new data set is annotated completely, the model is fine tuned on the new data set.

6.2 Contributions

Concerning contributions of this research to the state of the art in signature detection, a new baseline of accuracy for detecting signatures in the Tobacco 800 data set was established. To the author's knowledge, the proposed methodology surpasses all published approaches. Moreover, the list of files from the Tobacco 800 data set used for training, validating, and testing is stated on the filename level in the appendix of the thesis and was published via Zenodo [18]. This allows the reader to reproduce the proposed results

and encourages fair comparison of signature detection algorithms. No publication listed the exact file names used to train signature detection algorithms with the Tobacco 800 data set. In addition, a new data set was created over the course of this master thesis, the City of Vienna building applications data set. A data set compiled of building application documents of varying kinds, where each scanned document image contains a variety of visual entities such as machine printed text, handwriting, notes, logos and signatures. Precise location and dimension of signatures were annotated in a semi-automated approach via bounding boxes and subsequently refined manually for correctness. This data set cannot be released to the public, but can be employed for further training other deep learning models by the City of Vienna.

6.3 Limitations

The proposed signature detection system is a prototypical implementation. The model used for prediction was trained on a small data set. Therefore, performance in a real world scenario may vary. If a larger data set is available in the future, re-training the model might result in better accuracy. Despite the system being capable of performing inference on CPUs, useful performance can only be guaranteed when inferencing on state of the art GPUs. Furthermore, the thesis was focused on creating an accurate deep learning model for handwritten signature detection, instead of developing a distributed, high performance encapsulation for machine learning models, that allows a large audience of end users parallel access to its inference capabilities. Subsequently the implemented system rather serves as a proof of concept, than a system for productive use. The implementation, accessible to the end user, is therefore neither optimized for serving numerous requests in parallel nor features security measures to restrict unsolicited use. The new baseline established on signature detection in the Tobacco 800 data set was verified by comparing the achieved metrics against scores found in publications. Given the limited information in publications about the training process, especially the exact data set split on filename level, no completely fair comparison could be performed.

6.4 Future Work

Several opportunities exist to improve the proposed system. First, the performance of models in proposed experiments could be analyzed in regard of a larger data set. Therefore, more samples have to be collected and annotated and the models retrained on the new data set. Second, new object detection convolutional neural networks are expected to be release in the future. These new algorithms could be trained and evaluated on the Tobacco 800 data set and the achieved score compared to the new baseline set in this thesis. This is encouraged by providing the exact list of files used for training, validation and testing in the appendix. Third, the proposed system could be evaluated with civil servants and potential productivity gains in comparison to a completely manual process analyzed. If these tests are successful and high acceptance could be achieved, efficacy of the model could be optimized. In this work, accuracy was prioritized over

6. CONCLUSION

inference speed and model size in the experiments performed in this thesis, employing smaller models such as YOLOv5s would highly likely result in reduced accuracy but increase inference speed. Future research could evaluate if the achieved performance is still sufficient to alleviate the signature assertion process performed by civil servants at MA37. If performance suffices, the signature detection system could be ported to mobile devices with low GPU performance such as smartphones and laptop computers. Finally, the created model could be used to develop a distributed inference system optimized for parallel use to serve end users more efficiently.

Appendix

A.1 Training, Validation and Testing split of Tobacco 800

Each filename is followed by the corresponding set <partition tr(aining)|te(sting)|v(alidation)> <portion of training set>, i.e. te44 refers to a file belonging to the testing partition of the data set split 44% training, 10% validation and 36% testing. Images without signatures were excluded from the split. The data set split was published via Zenodo for easier parsing [18].

wau30a00page9_3.jpg tr44 tr60 tr75 tr80 cij01a00page02_2.jpg te44 te60 te75 tr80 dgw64a00.jpg tr44 tr60 tr75 tr80 fny38c00page05_5.jpg te44 tr60 tr75 tr80 fzs20e00page02_2.jpg tr44 tr60 tr75 tr80 vdr55d00.jpg tr44 tr60 tr75 tr80 umw13f00.jpg te44 te60 tr75 tr80 amk00a00.jpg te44 te60 v75 tr80 suh90a00.jpg tr44 tr60 tr75 tr80 mwj41f00.jpg tr44 tr60 tr75 tr80 yeo49d00page02_2.jpg te44 tr60 tr75 tr80 vim53c00.jpg te44 tr60 tr75 tr80 djy33f00page02_2.jpg v44 v60 v75 v80 tma35f00.jpg tr44 tr60 tr75 tr80 cvg13f00.jpg te44 tr60 tr75 tr80 nu124f00.jpg tr44 tr60 tr75 tr80 xyf43a00page02_2.jpg v44 v60 tr75 tr80 njk79d00page02_2.jpg tr44 tr60 tr75 tr80 nzw16e00.jpg tr44 tr60 tr75 tr80 bkz54f00_1.jpg tr44 tr60 tr75 tr80 mmk15f00.jpg v44 v60 tr75 tr80 jcv75f00_1.jpg v44 te60 tr75 tr80 hpz95d00.jpg te44 v60 tr75 tr80 osp65f00.jpg tr44 tr60 tr75 tr80 cjb54c00.jpg v44 v60 tr75 tr80 lje254f00_1.jpg te44 tr60 tr75 tr80 vvq90a00page03_3.jpg te44 tr60 tr75 tr80 hti31a00_1.jpg te44 te60 v75 v80 vam30f00page01_1.jpg te44 tr60 tr75 tr80 vwd93f00first_1.jpg tr44 tr60 tr75 tr80 fhi41f00.jpg v44 v60 te75 te80 amw93e00.jpg te44 te60 te75 te80 eig45f00.jpg tr44 tr60 tr75 tr80 gpq38e00page02_2.jpg te44 te60 v75 tr80 eey54f00_1.jpg te44 te60 te75 te80 rsv90c00first.jpg v44 v60 te75 te80 kmw13f00.jpg tr44 tr60 tr75 tr80 ofy54f00_1.jpg v44 tr60 tr75 tr80 kci90c00.jpg tr44 tr60 tr75 tr80 qit05f00page2_12.jpg te44 te60 te75 v80 mht90f00varfull_1.jpg tr44 tr60 tr75 tr80 ytz94a00.jpg tr44 tr60 tr75 tr80 xtj41f00.jpg tr44 tr60 tr75 tr80 wau30a00page9_2.jpg te44 te60 te75 te80 std20e00page01_1.jpg tr44 tr60 tr75 tr80 oan00d00.jpg v44 v60 v75 v80 pke56d00.jpg tr44 tr60 tr75 tr80 qit05f00page2_38.jpg te44 te60 tr75 tr80 qit05f00page2_10.jpg tr44 tr60 tr75 tr80 vby5aa00.jpg v44 tr60 tr75 tr80 nlk90c00.jpg te44 tr60 tr75 tr80	gdh93f00.jpg tr44 tr60 tr75 tr80 cnk41e00page02_2.jpg te44 te60 v75 v80 thi51a00page02_2.jpg tr44 tr60 tr75 tr80 kfw39d00.jpg tr44 tr60 tr75 tr80 hze996c00first.jpg tr44 tr60 tr75 tr80 dlk65e00.jpg te44 te60 tr75 tr80 ztz52d00page02_2.jpg v44 v60 v75 v80 icw51a00.jpg tr44 tr60 tr75 tr80 bat60e00.jpg te44 tr60 tr75 tr80 bea6aa00.jpg te44 te60 te75 tr80 cel93f00.jpg v44 tr60 tr75 tr80 wfg55f00_1.jpg tr44 tr60 tr75 tr80 hfn24f00.jpg tr44 tr60 tr75 tr80 iom19e00.jpg te44 te60 te75 te80 vfh43f00.jpg te44 te60 tr75 tr80 xmw13f00.jpg te44 tr60 tr75 tr80 gfe210e00.jpg tr44 tr60 tr75 tr80 mxt33d00.jpg te44 te60 v75 v80 yyj15f00page3_3.jpg v44 v60 tr75 tr80 eck05f00_1.jpg v44 tr60 tr75 tr80 iox21c00.jpg tr44 tr60 tr75 tr80 qbh54c00page02_2.jpg te44 te60 tr75 tr80 wtm90c00.jpg tr44 tr60 tr75 tr80 qnj41f00.jpg tr44 tr60 tr75 tr80 zav15e00.jpg tr44 tr60 tr75 tr80 dmy31e00page02_2.jpg te44 te60 tr75 tr80 biz25e00.jpg tr44 tr60 tr75 tr80 iq70f00page02_2.jpg te44 te60 v75 v80 qim54c00_1.jpg tr44 tr60 tr75 tr80 gpq38e00page02_1.jpg v44 v60 te75 te80 nuz52d00.jpg v44 tr60 tr75 tr80 upp41f00.jpg tr44 tr60 tr75 tr80 nrg54f00page02_1.jpg te44 tr60 tr75 tr80 jrk44a00.jpg te44 te60 te75 te80 sik79d00.jpg v44 v60 v75 v80 cnk41e00page2_2.jpg tr44 tr60 tr75 tr80 jrp2aa00page02_2.jpg tr44 tr60 tr75 tr80 khh96d00.jpg te44 tr60 tr75 tr80 bke80e00.jpg tr44 tr60 tr75 tr80 ocw55f00.jpg tr44 tr60 tr75 tr80 pti31a00_1.jpg te44 v60 tr75 tr80 zqs18e00page01_1.jpg te44 te60 te75 te80 ibk15f00page02_2.jpg tr44 tr60 tr75 tr80 uji44a00.jpg te44 te60 v75 v80 jfw98c00.jpg tr44 tr60 tr75 tr80 wfg55f00.jpg te44 te60 tr75 tr80 xlk90c00_1.jpg te44 v60 tr75 tr80 btu54a00.jpg te44 tr60 tr75 tr80 kvw59c00.jpg te44 tr60 tr75 tr80 wau30a00page9_5.jpg te44 te60 te75 v80 bnj00a00.jpg tr44 tr60 tr75 tr80	uzb51a00.jpg te44 te60 te75 tr80 aao54e00_2.jpg tr44 tr60 tr75 tr80 qit05f00page2_29.jpg te44 te60 te75 te80 xme03e00.jpg te44 te60 tr75 tr80 ehz25e00.jpg tr44 tr60 tr75 tr80 hylb45f00.jpg te44 tr60 tr75 tr80 dea05a00.jpg te44 tr60 tr75 tr80 dqn43c00.jpg te44 te60 tr75 tr80 dgi64c00.jpg te44 te60 tr75 tr80 drd89c00page02_2.jpg te44 tr60 tr75 tr80 ixp01f00_1.jpg tr44 tr60 tr75 tr80 ukk18c00page03_3.jpg tr44 tr60 tr75 tr80 dwr29e00_2.jpg te44 tr60 tr75 tr80 qfw98c00.jpg te44 te60 tr75 tr80 whz29d00page03_3.jpg te44 te60 te75 te80 nu100a00.jpg tr44 tr60 tr75 tr80 zkd43f00_3.jpg te44 tr60 tr75 tr80 hcu72e00_2.jpg v44 tr60 tr75 tr80 jcv75f00_6.jpg te44 te60 te75 tr80 eta15e00.jpg v44 v60 v75 v80 crr09c00_1.jpg tr44 tr60 tr75 tr80 bqz95d00.jpg tr44 tr60 tr75 tr80 sxq52e00page02_2.jpg te44 v60 v75 te80 wlv85f00_2.jpg tr44 tr60 tr75 tr80 mba90a00.jpg te44 te60 te75 tr80 acr64d00.jpg tr44 tr60 tr75 tr80 ypz83f00_1.jpg tr44 tr60 tr75 tr80 zkd43f00_2.jpg v44 tr60 tr75 tr80 fki00f00page02_2.jpg te44 tr60 tr75 tr80 xch36e00.jpg te44 te60 te75 te80 wfn74c00.jpg te44 te60 te75 v80 ofw98c00.jpg te44 v60 tr75 tr80 rzz5aa00.jpg tr44 tr60 tr75 tr80 obh31f00_1.jpg tr44 tr60 tr75 tr80 pj44a00.jpg te44 tr60 tr75 tr80 pfv39d00.jpg tr44 tr60 tr75 tr80 kv75f00page02_2.jpg tr44 tr60 tr75 tr80 nuu04e00.jpg v44 tr60 tr75 tr80 jpi68d00.jpg te44 te60 te75 te80 xbk38d00page02_2.jpg te44 tr60 tr75 tr80 gsp65f00.jpg te44 te60 tr75 tr80 rxm78d00page02_2.jpg te44 te60 te75 te80 hcj11c00page02_2.jpg tr44 tr60 tr75 tr80 yte65e00.jpg tr44 tr60 tr75 tr80 wfg15f00full.jpg te44 tr60 tr75 tr80 scv85f00.jpg tr44 tr60 tr75 tr80 zlw44e00page02_2.jpg tr44 tr60 tr75 tr80 wab91d00var.jpg tr44 tr60 tr75 tr80 ukd41a00ernest.jpg tr44 tr60 tr75 tr80 giy01a00page02_2.jpg v44 v60 tr75 tr80 cmw13f00.jpg v44 v60 v75 v80
---	---	---

A. APPENDIX

hst85500.jpg te44 v60 tr75 tr80
qit05f00page2_14.jpg te44 tr60 tr75 tr80
ruu5e00.jpg te44 tr60 tr75 tr80
wau30a00page9_4.jpg te44 tr60 tr75 tr80
ruu90c00.jpg v44 v60 v75 v80
diy01a00page02_2.jpg te44 tr60 tr75 tr80
cjj44a00.jpg te44 tr60 tr75 tr80
kmf72d00_1.jpg v44 tr60 tr75 tr80
wau30a00page2_6.jpg te44 v60 v75 tr80
kgi60e00.jpg te44 tr60 tr75 tr80
qit05f00page2_16.jpg te44 te60 te75 v80
bad45f00.jpg te44 tr60 tr75 tr80
qma35f00.jpg te44 tr60 tr75 tr80
xym00d00.jpg v44 tr60 tr75 tr80
zqc25f00_1.jpg te44 te60 te75 te80
mnq44a00.jpg te44 tr60 tr75 tr80
bwu60f00_1.jpg te44 te60 tr75 tr80
cjj33f00page02_2.jpg te44 tr60 tr75 tr80
dck05f00_1.jpg te44 te60 te75 te80
nr143c00.jpg te44 te60 v75 v80
kdlw38e00page02_2.jpg te44 te60 te75 tr80
xlk00a00.jpg te44 tr60 tr75 tr80
aux55d00.jpg te44 te60 te75 v80
jbs53d00.jpg te44 te60 tr75 tr80
vfe60f00.jpg te44 tr60 tr75 tr80
hlz25e00.jpg te44 tr60 tr75 tr80
mev75d00_2.jpg te44 te60 v75 v80
lku85f00.jpg te44 tr60 tr75 tr80
zqo9e00.jpg v44 v60 v75 v80
lks41a00.jpg te44 tr60 tr75 tr80
pym00d00page14_14.jpg te44 te60 te75 te80
ygs60f00page02_2.jpg te44 tr60 tr75 tr80
bsj50e00.jpg te44 tr60 tr75 tr80
tpz83f00.jpg te44 tr60 tr75 tr80
zrt45f00.jpg te44 tr60 tr75 tr80
odm51f00.jpg te44 te60 te75 te80
cdq65f00.jpg te44 tr60 tr75 tr80
zkd43f00_1.jpg te44 v60 te75 te80
ykh94f00.jpg v44 v60 v75 v80
rks41a00.jpg te44 te60 v75 v80
ebp68d00page02_2.jpg te44 te60 v75 tr80
wej25f00.jpg te44 te60 te75 te80
owe03e00.jpg te44 te60 te75 te80
rta5aa00.jpg v44 v60 v75 v80
ff10f00.jpg te44 te60 te75 te80
agw39d00.jpg te44 tr60 tr75 tr80
tjt59e00var.jpg te44 tr60 tr75 tr80
ecn9aa00.jpg te44 tr60 tr75 tr80
gpi68d00.jpg te44 tr60 tr75 tr80
necn0d00.jpg te44 te60 tr75 tr80
oma35f00.jpg te44 te60 v75 te80
scf65f00.jpg te44 tr60 tr75 tr80
bea69d00.jpg v44 tr60 tr75 tr80
wau30a00page9_7.jpg te44 tr60 tr75 tr80
zny04f00_2.jpg te44 tr60 tr75 tr80
bfr18e00page02_2.jpg te44 tr60 tr75 tr80
enn00a00.jpg te44 tr60 tr75 tr80
cgr9e00.jpg te44 tr60 tr75 tr80
vib56d00.jpg te44 te60 tr75 tr80
eld41a00.jpg te44 tr60 tr75 tr80
cgs54f00_1.jpg te44 tr60 tr75 tr80
bfr18e00.jpg te44 tr60 tr75 tr80
bhw64a00.jpg te44 tr60 tr75 tr80
tum33a00.jpg te44 tr60 tr75 tr80
ixv75f00_1.jpg te44 te60 v75 tr80
wbv90c00page02_2.jpg v44 v60 v75 v80
lkd31a00.jpg te44 tr60 tr75 tr80
jrs9aa00.jpg te44 te60 te75 te80
zss86d00.jpg te44 tr60 tr75 tr80
gzr94e00page01_1.jpg te44 te60 te75 te80
djs24f00.jpg v44 v60 v75 v80
yoi68d00.jpg te44 tr60 tr75 tr80
qpr23f00page02_2.jpg te44 tr60 tr75 tr80
bba45f00.jpg te44 tr60 tr75 tr80
xtm90c00.jpg te44 tr60 tr75 tr80
qcu5aa00init.jpg v44 te60 tr75 tr80
wzt35f00.jpg te44 te60 tr75 tr80
lrm00d00first_1.jpg te44 te60 v75 tr80
qpr23f00.jpg te44 tr60 tr75 tr80
mzm72e00page06_6.jpg te44 tr60 tr75 tr80
eiv46d00.jpg te44 te60 te75 te80
hki32e00.jpg te44 tr60 tr75 tr80
ezr04d00.jpg te44 te60 tr75 tr80
mta00c00.jpg te44 te60 tr75 tr80
ceox05f00_1.jpg te44 te60 te75 te80
wud23f00.jpg v44 v60 tr75 tr80
nir55d00page02_2.jpg te44 tr60 tr75 tr80
lpy35f00first.jpg v44 v60 v75 tr80
bex05f00_1.jpg te44 tr60 tr75 tr80
huz50e00_1.jpg te44 te60 tr75 tr80
qz54f00_1.jpg te44 tr60 tr75 tr80
fja22c00.jpg te44 te60 v75 te80
scv85f00.jpg te44 tr60 tr75 tr80
fam00a00.jpg te44 tr60 tr75 tr80
hdx55e00.jpg te44 tr60 tr75 tr80
mge98e00.jpg te44 tr60 tr75 tr80
qit05f00page2_8.jpg v44 tr60 tr75 tr80
bjn43c00page02_2.jpg te44 tr60 tr75 tr80
cbm67e00page02_2.jpg te44 tr60 tr75 tr80
drn00d00.jpg v44 v60 v75 v80
zhu43d00.jpg te44 tr60 tr75 tr80
lbu31a00.jpg te44 te60 v75 tr80
vyx02f00page02_2.jpg te44 tr60 tr75 tr80
hmv91c00_2.jpg te44 tr60 tr75 tr80
bda6aa00.jpg te44 te60 tr75 tr80
jij01a00page02_2.jpg te44 tr60 tr75 tr80
yaq00e00.jpg v44 te60 tr75 tr80
uvk3aa00.jpg te44 te60 v75 v80
sla48c00.jpg te44 te60 te75 tr80
jzx41a00.jpg v44 v60 v75 v80
kffj00a00.jpg te44 tr60 tr75 tr80
sdx05f00_1.jpg te44 te60 v75 v80
ors51e00page2var_2.jpg te44 tr60 tr75 tr80
dfm53d00.jpg te44 tr60 tr75 tr80
jy335a00.jpg te44 tr60 tr75 tr80
iny31e00.jpg te44 tr60 tr75 tr80
cio55e00page02variant_2.jpg te44 tr60 tr75 tr80
tr80
lrm00d00first_2.jpg te44 tr60 tr75 tr80
orn43d00.jpg v44 v60 tr75 tr80
xbn00d00.jpg te44 te60 v75 tr80
tzs45f00.jpg te44 te60 te75 te80
hfw98c00.jpg te44 te60 te75 tr80
wfv39d00.jpg te44 tr60 tr75 tr80
vvg45f00first.jpg te44 te60 v75 tr80
sia23d00.jpg v44 v60 v75 v80
sfp41a00page5_5.jpg te44 te60 te75 tr80
duz52d00page02var_3.jpg te44 tr60 tr75 tr80
gcf65f00.jpg te44 v60 tr75 tr80
law8aa00.jpg te44 tr60 tr75 tr80
hpj41f00.jpg te44 te60 tr75 tr80
piw13f00.jpg te44 tr60 tr75 tr80
gpx9aa00first.jpg te44 te60 te75 v80
ail70a00.jpg te44 te60 te75 v80
ald41a00ernest.jpg te44 tr60 tr75 tr80
bjn43c00page02_1.jpg te44 tr60 tr75 tr80
cmw44e00.jpg te44 tr60 tr75 tr80
ymp51a00.jpg te44 tr60 tr75 tr80
kqe30a00first.jpg te44 tr60 tr75 tr80
aah97e00page02_2.jpg te44 tr60 tr75 tr80
fks41a00.jpg v44 v60 v75 v80
fh443f00.jpg te44 te60 v75 tr80
qcw00f00page02_2.jpg v44 te60 tr75 tr80
daf01f00first.jpg te44 te60 te75 te80
xjx9aa00first.jpg te44 v60 tr75 tr80
gsr09c00.jpg te44 tr60 tr75 tr80
vjk79d00page04_4.jpg te44 te60 tr75 tr80
znmw13f00.jpg te44 te60 tr75 tr80
bug83d00page02_2.jpg v44 te60 v75 tr80
asx34c00page02_2.jpg te44 tr60 tr75 tr80
pfk90c00.jpg te44 te60 te75 tr80
rj144a00.jpg te44 te60 te75 v80
kjl13f00.jpg te44 te60 v75 v80
abs60f00.jpg te44 te60 te75 tr80
juo75f00_1.jpg te44 tr60 tr75 tr80
cew64a00page02_2.jpg te44 te60 tr75 tr80
miw83f00.jpg te44 tr60 tr75 tr80
ecv85f00.jpg te44 te60 tr75 tr80
pek94c00_2.jpg te44 tr60 tr75 tr80
pqr23f00page07_7.jpg te44 tr60 tr75 tr80
wrv35d00page02_2.jpg te44 tr60 tr75 tr80
ajz31e00.jpg te44 tr60 tr75 tr80
ajj10e00.jpg te44 tr60 tr75 tr80
exl43d00page02_2.jpg te44 te60 tr75 tr80
don15f00_1.jpg te44 tr60 tr75 tr80
zfs4f00_1.jpg te44 tr60 tr75 tr80
ewx61c00.jpg v44 v60 v75 tr80
sfp41a00page5_1.jpg te44 tr60 tr75 tr80
ksf09c00_1.jpg te44 tr60 tr75 tr80
hfr79c00.jpg te44 tr60 tr75 tr80
eaf60e00.jpg te44 te60 v75 v80
gaio1c00.jpg te44 te60 te75 tr80
wgo05c00.jpg te44 tr60 tr75 tr80
fwe69c00.jpg te44 tr60 tr75 tr80
nsr05f00_1.jpg v44 tr60 tr75 tr80
mos87c00page02_2.jpg te44 tr60 tr75 tr80
ghz25e00.jpg te44 tr60 tr75 tr80
gaio1c00.jpg te44 te60 te75 tr80
yav51a00.jpg te44 tr60 tr75 tr80
eoc33a00page02_2.jpg te44 tr60 tr75 tr80
njn54c00.jpg te44 tr60 tr75 tr80
gmkl5f00.jpg te44 te60 tr75 tr80
dny38c00_5.jpg v44 v60 tr75 tr80
fzbl1c00ovar.jpg te44 te60 v75 v80
mub51a00.jpg te44 tr60 tr75 tr80
was45f00.jpg te44 tr60 tr75 tr80
lhw39d00.jpg te44 tr60 tr75 tr80
vbd23f00.jpg te44 tr60 tr75 tr80
lhw39d00.jpg te44 tr60 tr75 tr80
sfw98c00.jpg te44 te60 te75 te80
mzd01a00.jpg te44 tr60 tr75 tr80
fyg65f00.jpg te44 tr60 tr75 tr80
kqd94a00.jpg te44 tr60 tr75 tr80
pkj90c00.jpg v44 v60 v75 v80
kpl36e00.jpg te44 tr60 tr75 tr80
qat01f00.jpg te44 tr60 tr75 tr80
adq65f00.jpg te44 tr60 tr75 tr80
eao90f00.jpg te44 te60 te75 te80
adp7aa00.jpg te44 tr60 tr75 tr80
fpg81e00.jpg te44 te60 te75 te80
rk41f00.jpg te44 tr60 tr75 tr80
eqb46d00.jpg te44 tr60 tr75 tr80
vrr09c00page03_3.jpg te44 te60 te75 tr80
cgv39d00.jpg te44 tr60 tr75 tr80
alz35d00.jpg te44 tr60 tr75 tr80
jgs60f00page03_3.jpg te44 tr60 tr75 tr80
pek94c00_1.jpg te44 te60 v75 v80
mma35f00.jpg te44 tr60 tr75 tr80
yhw13f00_1.jpg te44 tr60 tr75 tr80
hbk41e00.jpg te44 tr60 tr75 tr80
btt85f00page2_2.jpg te44 tr60 tr75 tr80
cpr95d00variant_2.jpg te44 tr60 tr75 tr80
foe33a00page02_2.jpg te44 tr60 tr75 tr80
ogg70f00page3_3.jpg te44 tr60 tr75 tr80
don15f00_2.jpg te44 tr60 tr75 tr80
btc51a00.jpg te44 tr60 tr75 tr80
hlm6aa00init.jpg te44 tr60 tr75 tr80
cfr04f00_1.jpg te44 tr60 tr75 tr80
sfp41a00page5_2.jpg te44 te60 tr75 tr80
avb45e00.jpg te44 tr60 tr75 tr80
uiw28e00.jpg te44 te60 te75 te80
kvb51a00.jpg te44 tr60 tr75 tr80
gwo3aa00.jpg te44 te60 tr75 tr80
ozj00a00.jpg te44 tr60 tr75 tr80
sxjw13f00.jpg te44 tr60 tr75 tr80
sma35f00.jpg te44 tr60 tr75 tr80
rz33aa00.jpg te44 v60 te75 te80
btt85f00page2_3.jpg te44 tr60 tr75 tr80
ama91d00page03_3.jpg te44 tr60 tr75 tr80
hby31f00_1.jpg te44 te60 v75 tr80
pdg62d00page02_2.jpg te44 tr60 tr75 tr80
nyk41e00.jpg v44 v60 tr75 tr80
nff41e00.jpg te44 tr60 tr75 tr80
ikw83f00.jpg te44 tr60 tr75 tr80
gnx21c00page02_2.jpg te44 te60 te75 te80
jhs25e00.jpg te44 tr60 tr75 tr80
kjj24f00.jpg te44 tr60 tr75 tr80
wme03e00.jpg te44 tr60 tr75 tr80
zvs17e00.jpg te44 te60 v75 tr80
jfd45f00.jpg te44 tr60 tr75 tr80
sdm51f00.jpg v44 v60 te75 te80
en41f00.jpg te44 tr60 tr75 tr80
vcv85f00.jpg te44 tr60 tr75 tr80
pmp81f00.jpg te44 tr60 tr75 tr80
ran25f00.jpg te44 te60 te75 te80
iyn03d00.jpg te44 tr60 tr75 tr80
pca6aa00.jpg te44 te60 v75 v80
rtv20a00var.jpg te44 te60 te75 v80
oky33f00var.jpg te44 te60 tr75 tr80
skj51f00_1.jpg te44 tr60 tr75 tr80
djw26d00.jpg te44 te60 te75 te80
ggp21c00var.jpg te44 tr60 tr75 tr80
kan00d00.jpg te44 tr60 tr75 tr80
tkj51f00_1.jpg te44 te60 v75 v80
cry54f00_1.jpg te44 tr60 tr75 tr80
hez90c00.jpg v44 te60 tr75 tr80
wzk36d00page02first_2.jpg v44 tr60 tr75 tr80
fcb93e00.jpg te44 tr60 tr75 tr80
pfd65f00.jpg te44 te60 te75 te80
wbo3aa00.jpg te44 te60 tr75 tr80
zda6aa00.jpg te44 tr60 tr75 tr80
ege23f00_1.jpg te44 te60 te75 te80
mcs39c00.jpg te44 te60 te75 v80
aex05f00_1.jpg te44 tr60 tr75 tr80
wau30a00page9_15.jpg te44 tr60 tr75 tr80
cxi59c00_2.jpg te44 te60 tr75 tr80
uzm55d00page02_2.jpg v44 v60 v75 te80
aeb95e00.jpg te44 te60 v75 v80
aew44e00.jpg te44 tr60 tr75 tr80
xxs58e00.jpg te44 tr60 tr75 tr80
odx05f00_1.jpg te44 tr60 tr75 tr80
nle30a00.jpg te44 tr60 tr75 tr80
rzo50e00.jpg te44 te60 te75 te80
fsc51a00.jpg te44 te60 tr75 tr80
gjl70a00page2_2.jpg te44 tr60 tr75 tr80
pqw02f00page07_7.jpg te44 tr60 tr75 tr80
wdf61f00page2_2.jpg te44 te60 te75 tr80
qaw85f00.jpg te44 tr60 tr75 tr80
bak14d00.jpg te44 tr60 tr75 tr80
wut05f00.jpg te44 te60 te75 v80
pqu51e00.jpg te44 tr60 tr75 tr80
osv63f00_2.jpg te44 tr60 tr75 tr80
men75f00.jpg te44 tr60 tr75 tr80
dlu7aa00_2.jpg te44 tr60 tr75 tr80
itflaa00.jpg te44 tr60 tr75 tr80
lin01f00.jpg te44 te60 tr75 tr80
hna35f00.jpg te44 te60 te75 te80
gaio1c00.jpg te44 te60 v75 te80
fsh23c00page04_4.jpg v44 v60 v75 v80
yqi08e00.jpg te44 v60 tr75 tr80
iivw54f00_1.jpg v44 v60 te75 te80
anv39d00.jpg te44 tr60 tr75 tr80
qjj44a00.jpg te44 tr60 tr75 tr80
qo168d00.jpg te44 te60 v75 te80
hls03f00_1.jpg te44 tr60 tr75 tr80
arz92e00.jpg te44 tr60 tr75 tr80
gxp01f00_1.jpg v44 v60 te75 tr80
aik94a00page02_2.jpg te44 te60 te75 te80
pvs38c00page06_6.jpg v44 v60 v75 v80
bbl11c00.jpg v44 v60 te75 te80
ihz25e00.jpg te44 tr60 tr75 tr80
fbv15e00.jpg te44 tr60 tr75 tr80
kqb30f00.jpg te44 tr60 tr75 tr80
hvb51a00.jpg te44 tr60 tr75 tr80
bjj44a00.jpg te44 te60 v75 tr80
jnm00a00.jpg v44 v60 v75 v80
jfh35f00.jpg te44 te60 te75 v80
xlx69d00page07_7.jpg te44 tr60 tr75 tr80
wso51e00.jpg te44 tr60 tr75 tr80
aex05f00_2.jpg te44 tr60 tr75 tr80
spg81c00.jpg te44 te60 tr75 tr80
vcp7aa00.jpg te44 te60 te75 te80
cxi59c00_1.jpg te44 tr60 tr75 tr80
wau30a00page9_16.jpg te44 tr60 tr75 tr80
yas45f00.jpg te44 te60 te75 te80
wau30a00page9_17.jpg te44 te60 te75 te80
rgj46d00.jpg te44 tr60 tr75 tr80
fpi68d00.jpg te44 tr60 tr75 tr80
mma35f00.jpg te44 tr60 tr75 tr80
bb12f00.jpg te44 tr60 tr75 tr80
dtflaa00.jpg te44 tr60 tr75 tr80
cad45f00page02_2.jpg te44 te60 tr75 tr80
osv63f00_1.jpg te44 tr60 tr75 tr80
hqj59c00.jpg v44 v60 v75 v80
dkn80f00.jpg te44 te60 tr75 tr80
qit05f00page2_2.jpg te44 tr60 tr75 tr80
whg41f00.jpg te44 te60 tr75 tr80
csj50c00.jpg te44 te60 tr75 tr80
zgm69d00_2.jpg te44 tr60 tr75 tr80
hgp21c00var.jpg te44 te60 tr75 tr80
mlk12d00.jpg te44 tr60 tr75 tr80
gko55e00page02_2.jpg te44 tr60 tr75 tr80
sks41a00.jpg te44 te60 te75 tr80
cle30a00.jpg te44 tr60 tr75 tr80
xik94f00.jpg te44 te60 te75 tr80
dsj50c00page03_3.jpg te44 te60 te75 tr80
uma35f00.jpg te44 te60 te75 v80
pnt08d00.jpg te44 tr60 tr75 tr80
asg54f00.jpg te44 tr60 tr75 tr80
grm00d00.jpg te44 te60 te75 tr80
jaw63f00varfull_1.jpg v44 tr60 tr75 tr80
hwz64c00var.jpg te44 tr60 tr75 tr80
qit05f00page2_6.jpg te44 tr60 tr75 tr80
xca6aa00.jpg te44 te60 te75 tr80
alk3aa00.jpg te44 tr60 tr75 tr80
nzy64f00var_1.jpg te44 te60 te75 tr80
yme01e00page01_1.jpg te44 tr60 tr75 tr80
djh5aa00.jpg te44 te60 te75 tr80
avd23f00first_1.jpg te44 te60 v75 te80

aeq93a00.jpg te44 tr60 tr75 tr80
 qe159c00.jpg tr44 tr60 tr75 tr80
 kdb11a00page03_3.jpg tr44 tr60 tr75 tr80
 qx8f00page01_1.jpg te44 te60 tr75 tr80
 zrz94a00page02_2.jpg te44 tr60 tr75 tr80
 ayy01c00.jpg tr44 tr60 tr75 tr80
 icu98c00.jpg te44 te60 tr75 tr80
 fgx54f00_1.jpg te44 te60 te75 te80
 get10c.jpg te44 tr60 tr75 tr80
 fre41e00.jpg te44 te60 tr75 tr80
 fez35f00.jpg te44 te60 te75 te80
 kcn64a00.jpg tr44 tr60 tr75 tr80
 aj132e00page02_2.jpg te44 te60 v75 v80
 hty24f00page02_2.jpg te44 te60 tr75 tr80
 nky33f00page02_2.jpg tr44 tr60 tr75 tr80
 pcw43c00page02_2.jpg tr44 tr60 tr75 tr80
 dme30a00.jpg te44 te60 v75 v80
 wau30a00page9_13.jpg te44 tr60 tr75 tr80
 ank41e00.jpg te44 te60 tr75 tr80
 ewe36d00.jpg v44 v60 v75 v80
 ceo04f00.jpg tr44 tr60 tr75 tr80
 bnp51a00.jpg te44 tr60 tr75 tr80
 zo168d00.jpg tr44 tr60 tr75 tr80
 aww54f00_1.jpg te44 te60 tr75 tr80
 idr55d00.jpg te44 te60 te75 tr80
 wat01f00.jpg tr44 tr60 tr75 tr80
 mln70f00page03_3.jpg tr44 tr60 tr75 tr80
 jrc19c00.jpg tr44 tr60 tr75 tr80
 waq00e00.jpg tr44 tr60 tr75 tr80
 yyo25f00.jpg tr44 tr60 tr75 tr80
 chg15f00full_1.jpg v44 v60 v75 tr80
 cqt45f00page02_2.jpg v44 v60 tr75 tr80
 kna35f00.jpg tr44 tr60 tr75 tr80
 ydx05f00.jpg tr44 tr60 tr75 tr80
 evz52d00page02_2.jpg tr44 tr60 tr75 tr80
 cpi68d00.jpg te44 te60 te75 te80
 cdp7aa00var.jpg v44 v60 tr75 tr80
 yun00a00.jpg te44 tr60 tr75 tr80
 lox9aa00fip.jpg tr44 tr60 tr75 tr80
 who15f00.jpg tr44 tr60 tr75 tr80
 wdf14f00page02_2.jpg te44 tr60 tr75 tr80
 byd23a00.jpg tr44 tr60 tr75 tr80
 osi0f00page02_2.jpg tr44 tr60 tr75 tr80
 bin05a00.jpg te44 tr60 tr75 tr80
 kde44c00page02_2.jpg tr44 tr60 tr75 tr80
 fvz52d00page02_2.jpg v44 tr60 tr75 tr80
 mlr04f00.jpg v44 v60 tr75 tr80
 xrm23c00.jpg te44 tr60 tr75 tr80
 weq76d00.jpg te44 v60 v75 v80
 rin95e00.jpg te44 te60 tr75 tr80
 nnj41f00.jpg tr44 tr60 tr75 tr80
 eys04c00page04_4.jpg tr44 tr60 tr75 tr80
 jaw63f00varfull_2.jpg tr44 tr60 tr75 tr80
 qof23f00.jpg tr44 tr60 tr75 tr80
 adn64a00page02_3.jpg tr44 tr60 tr75 tr80
 nrb01a00.jpg tr44 tr60 tr75 tr80
 uca6aa00.jpg tr44 tr60 tr75 tr80
 afm90c00first_2.jpg te44 te60 tr75 tr80
 juo75f00_36.jpg v44 tr60 tr75 tr80
 khs80e00_2.jpg te44 tr60 tr75 tr80
 cdp9aa00page02_2.jpg te44 tr60 tr75 tr80
 wau30a00page9_10.jpg te44 te60 v75 v80
 bbn00d00first.jpg tr44 tr60 tr75 tr80
 ufw98c00.jpg tr44 tr60 tr75 tr80
 cqcl13f00.jpg te44 te60 tr75 tr80
 abm69c00.jpg te44 te60 tr75 tr80
 wau30a00page9_11.jpg te44 te60 te75 te80
 vaw13f00.jpg v44 v60 tr75 tr80
 vbx83d00.jpg v44 v60 tr75 tr80
 axp01f00_1.jpg te44 te60 te75 v80
 gvz52d00page02_2.jpg te44 tr60 tr75 tr80
 amy31e00.jpg te44 tr60 tr75 tr80
 trv39d00.jpg te44 tr60 tr75 tr80
 trj44a00.jpg tr44 tr60 tr75 tr80
 kecv85f00_1.jpg te44 tr60 tr75 tr80
 qit05f00page2_4.jpg tr44 tr60 tr75 tr80
 tyo25f00.jpg te44 te60 tr75 tr80
 pzb89c00.jpg te44 te60 tr75 tr80
 kie30a00.jpg v44 tr60 tr75 tr80
 fki32e00.jpg te44 te60 v75 v80
 erw85a00.jpg tr44 tr60 tr75 tr80
 fmr56e00.jpg te44 te60 tr75 tr80
 oly45f00.jpg tr44 tr60 tr75 tr80
 egl10a00.jpg te44 te60 te75 tr80
 vff43c00.jpg te44 te60 te75 tr80
 cbp51e00_2.jpg tr44 tr60 tr75 tr80
 ch43f00page02var_2.jpg v44 v60 tr75 tr80
 aki32e00.jpg te44 te60 te75 te80
 rnf51a00.jpg tr44 tr60 tr75 tr80
 qit05f00page2_26.jpg v44 tr60 tr75 tr80
 be173f00.jpg tr44 tr60 tr75 tr80
 oxo21c00.jpg te44 v60 tr75 tr80
 pik10a00.jpg tr44 tr60 tr75 tr80
 rex05f00_1.jpg te44 te60 te75 te80
 arr09c00.jpg te44 tr60 tr75 tr80
 trv03f00page03_3.jpg te44 tr60 tr75 tr80
 kdhs1e00_1.jpg tr44 tr60 tr75 tr80
 oey93e00.jpg tr44 tr60 tr75 tr80
 hua33a00init.jpg te44 te60 te75 v80
 irn90c00page02_2.jpg te44 te60 te75 te80
 izj51a00.jpg te44 tr60 tr75 tr80
 xtr90a00_1.jpg tr44 tr60 tr75 tr80
 oen75f00.jpg te44 te60 te75 te80
 mzt43f00page3_3.jpg tr44 tr60 tr75 tr80
 pfb94f00page02var_2.jpg te44 te60 te75 te80
 nht43d00page02_2.jpg tr44 tr60 tr75 tr80
 xfe44c00page02fullvar_2.jpg tr44 tr60 tr75 tr80
 dna82e00.jpg tr44 tr60 tr75 tr80
 axz34c00page02_2.jpg tr44 tr60 tr75 tr80
 ten21f00full.jpg te44 te60 te75 te80
 sjj44a00.jpg te44 te60 te75 te80
 wfo00a00.jpg v44 v60 te75 v80
 iek79c00.jpg te44 te60 te75 te80
 icr55d00.jpg te44 te60 te75 v80
 iuk21a00.jpg tr44 tr60 tr75 tr80
 mbw13f00page2_2.jpg te44 te60 tr75 tr80
 jvz52d00page02_2.jpg tr44 tr60 tr75 tr80
 mc121e00.jpg te44 tr60 tr75 tr80
 chs6f00_1.jpg tr44 tr60 tr75 tr80
 esk12d00.jpg te44 te60 te75 te80
 cc65f00.jpg te44 te60 tr75 tr80
 nlu01f00.jpg v44 v60 v75 v80
 cix9aa00.jpg te44 te60 tr75 tr80
 oib30f00firstvar.jpg tr44 tr60 tr75 tr80
 csi09c00.jpg tr44 tr60 tr75 tr80
 gsj41f00page02_2.jpg te44 tr60 tr75 tr80
 erk44a00_1.jpg tr44 tr60 tr75 tr80
 fhz25e00.jpg te44 te60 v75 v80
 qit05f00page2_33.jpg te44 te60 te75 te80
 ply60e00.jpg tr44 tr60 tr75 tr80
 hbt66c00.jpg v44 v60 te75 tr80
 yda69d00page02_2.jpg te44 te60 v75 tr80
 adh36e00page2_2.jpg te44 te60 te75 te80
 wau30a00page9_9.jpg tr44 tr60 tr75 tr80
 lin90f00.jpg te44 te60 te75 te80
 isp65f00.jpg v44 v60 v75 v80
 gjw13f00.jpg te44 te60 te75 tr80
 qit05f00page2_31.jpg tr44 tr60 tr75 tr80
 xyr15f00.jpg tr44 tr60 tr75 tr80
 lcm24f00.jpg tr44 tr60 tr75 tr80
 dpl68d00.jpg te44 te60 te75 te80
 djps1a00.jpg tr44 tr60 tr75 tr80
 ovg33f00.jpg te44 te60 te75 te80
 ffl1aa00.jpg tr44 tr60 tr75 tr80
 qpb44c00page03_3.jpg te44 tr60 tr75 tr80
 ihh44a00.jpg te44 te60 tr75 tr80
 chw80e00_1.jpg te44 te60 tr75 tr80
 jdc94f00.jpg tr44 tr60 tr75 tr80
 boa85f00.jpg te44 tr60 tr75 tr80
 vad45f00page03_6.jpg te44 tr60 tr75 tr80
 mcm24f00_1.jpg tr44 tr60 tr75 tr80
 qyt81e00page02_2.jpg tr44 tr60 tr75 tr80
 xyb11c00page2_2.jpg te44 v60 tr75 tr80
 kft94f00.jpg tr44 tr60 tr75 tr80
 qqu85f00.jpg tr44 tr60 tr75 tr80
 edj50e00first_3.jpg tr44 tr60 tr75 tr80
 gpg81e00.jpg tr44 tr60 tr75 tr80
 tyr04d00.jpg te44 te60 te75 v80
 amr29e00page02_2.jpg tr44 v60 tr75 tr80
 ufs60f00.jpg tr44 tr60 tr75 tr80
 bfx94e00.jpg tr44 tr60 tr75 tr80
 khz25e00.jpg v44 tr60 tr75 tr80
 qjf59c00.jpg te44 te60 te75 tr80
 vda05a00.jpg te44 tr60 tr75 tr80
 kef61f00page2_2.jpg te44 te60 v75 v80
 eut41e00.jpg te44 te60 tr75 tr80
 gfg5aa00.jpg tr44 tr60 tr75 tr80
 atz94a00.jpg te44 te60 te75 te80
 xqw35e00.jpg tr44 tr60 tr75 tr80
 evr93c00.jpg te44 te60 tr75 tr80
 aam09c00.jpg v44 v60 tr75 tr80
 ouz52d00page02_2.jpg tr44 tr60 tr75 tr80
 csk72e00page03_3.jpg te44 te60 tr75 tr80
 ppz95d00.jpg te44 te60 te75 te80
 ket24f00.jpg tr44 tr60 tr75 tr80
 dic45f00_1.jpg tr44 tr60 tr75 tr80
 wky60e00.jpg tr44 tr60 tr75 tr80
 wrn67e00.jpg tr44 tr60 tr75 tr80
 qyx61c00.jpg v44 te60 tr75 tr80
 ldh93f00.jpg v44 tr60 tr75 tr80
 qit05f00page2_24.jpg te44 tr60 tr75 tr80
 qit05f00page2_18.jpg te44 te60 tr75 tr80
 pzm00d00.jpg tr44 tr60 tr75 tr80
 xoi68d00.jpg te44 te60 tr75 tr80
 llq11e00page02_2.jpg tr44 tr60 tr75 tr80
 rlf139d00.jpg te44 tr60 tr75 tr80
 biv33c00page02_2.jpg tr44 tr60 tr75 tr80
 hn93f00.jpg tr44 tr60 tr75 tr80
 qit05f00page2_34.jpg v44 v60 v75 v80
 ijt58d00page02_2.jpg tr44 tr60 tr75 tr80
 qit05f00page2_20.jpg te44 te60 tr75 tr80
 nlq86d00.jpg tr44 tr60 tr75 tr80
 pdy5aa00.jpg te44 v60 te75 v80
 fga05a00.jpg tr44 tr60 tr75 tr80
 ndafaa00.jpg te44 te60 te75 v80
 wlp51a00.jpg tr44 tr60 tr75 tr80
 wjf44f00.jpg te44 te60 tr75 tr80
 pmx82f00page04_4.jpg tr44 tr60 tr75 tr80
 cc177c00page02_2.jpg te44 tr60 tr75 tr80
 vad45f00page03_3.jpg tr44 tr60 tr75 tr80
 jtj30e00var.jpg te44 tr60 tr75 tr80
 lwd23f00page02_2.jpg v44 tr60 tr75 tr80
 eez35f00page02_2.jpg tr44 tr60 tr75 tr80
 bmc19d00.jpg v44 v60 tr75 tr80
 tbn00d00.jpg v44 tr60 tr75 tr80
 api68d00.jpg te44 te60 tr75 tr80
 ubd60f00.jpg te44 tr60 tr75 tr80
 cgy54f00_1.jpg te44 te60 te75 te80
 jnx54f00_1.jpg tr44 tr60 tr75 tr80
 zpz83f00_1.jpg te44 tr60 tr75 tr80
 dch31f00.jpg tr44 tr60 tr75 tr80
 ckr93d00.jpg tr44 tr60 tr75 tr80
 ehi41f00.jpg tr44 tr60 tr75 tr80
 vjy5aa00.jpg tr44 tr60 tr75 tr80
 ncp46d00var.jpg tr44 tr60 tr75 tr80
 nav20a00.jpg tr44 tr60 tr75 tr80
 vmk98c00.jpg te44 te60 tr75 tr80
 qlm53d00.jpg tr44 tr60 tr75 tr80
 bhd80a00.jpg tr44 tr60 tr75 tr80
 mcp00d00.jpg te44 tr60 tr75 tr80
 wma35f00.jpg te44 tr60 tr75 tr80
 ouq21c00.jpg v44 tr60 tr75 tr80
 dv41a00.jpg te44 tr60 tr75 tr80
 ceb93e00page03_3.jpg tr44 tr60 tr75 tr80
 nvz52d00page02_2.jpg tr44 tr60 tr75 tr80
 rsj41f00page02_2.jpg te44 tr60 tr75 tr80
 fjg59c00.jpg v44 v60 tr75 tr80
 bht85f00.jpg v44 v60 v75 v80
 fre53c00page02_3.jpg tr44 tr60 tr75 tr80
 men43c00.jpg tr44 tr60 tr75 tr80
 dhr55d00page02_2.jpg tr44 tr60 tr75 tr80
 tjr72f00page02_2.jpg tr44 tr60 tr75 tr80
 adh36e00_2.jpg v44 v60 te75 v80
 vss86d00.jpg te44 te60 v75 v80
 efm53d00.jpg tr44 tr60 tr75 tr80
 qv98d00page02_2.jpg tr44 tr60 tr75 tr80
 keo90c00page02_2.jpg te44 te60 tr75 tr80
 lpi68d00.jpg tr44 tr60 tr75 tr80
 mia62d00.jpg te44 tr60 tr75 tr80
 teb06e00.jpg tr44 tr60 tr75 tr80
 ifw98c00.jpg tr44 tr60 tr75 tr80
 vji44a00.jpg tr44 tr60 tr75 tr80
 cir10f00.jpg tr44 tr60 tr75 tr80
 dat60e00_1.jpg te44 te60 tr75 tr80
 mtq30d00_4.jpg tr44 tr60 tr75 tr80
 gxv75f00_1.jpg tr44 tr60 tr75 tr80
 syi15f00.jpg te44 tr60 tr75 tr80
 izq96c00.jpg v44 v60 te75 tr80
 thi93f00_1.jpg te44 te60 tr75 tr80
 rhr75e00.jpg te44 tr60 tr75 tr80
 dvw54f00_1.jpg tr44 tr60 tr75 tr80
 bfk68c00page03_3.jpg tr44 tr60 tr75 tr80
 jyx55e00.jpg tr44 tr60 tr75 tr80
 wry97e00.jpg tr44 tr60 tr75 tr80
 dfb44f00page4_4.jpg te44 v60 v75 v80
 pgo51e00page04_3.jpg te44 tr60 tr75 tr80
 rkc33a00page02_2.jpg v44 v60 tr75 tr80
 kgr04f00page3_3.jpg v44 tr60 tr75 tr80
 zjy03e00page02_2.jpg te44 te60 tr75 tr80
 gat5aa00.jpg tr44 tr60 tr75 tr80
 wav95e00page03_3.jpg te44 te60 v75 tr80
 lec44c00.jpg tr44 tr60 tr75 tr80
 fjx9aa00.jpg tr44 tr60 tr75 tr80
 kes05f00_1.jpg tr44 tr60 tr75 tr80
 uk143a00page03_3.jpg te44 tr60 tr75 tr80
 lfj35f00.jpg tr44 tr60 tr75 tr80
 sik65f00_1.jpg tr44 tr60 tr75 tr80
 kpu96c00.jpg tr44 tr60 tr75 tr80
 yrz52d00.jpg tr44 tr60 tr75 tr80
 qit05f00page2_36.jpg te44 te60 tr75 tr80
 qit05f00page2_22.jpg tr44 tr60 tr75 tr80
 eqp14e00page02_2.jpg tr44 tr60 tr75 tr80
 hfv39d00.jpg te44 te60 tr75 tr80
 jxp35f00.jpg tr44 tr60 tr75 tr80

A.2 Implementation of Signature Detection System

This section introduces the signature detection system encapsulating the YOLOv5 model to enable end users to perform inference on documents. The architecture, its API as well as its limitations are displayed. Finally, instructions for deployment of the system are given.

A.2.1 Architecture

To provide end users with access to the signature detection system an encapsulation for the YOLOV5 model had to be created. As prediction is a computationally intense task, that demands powerful GPUs, a distributed architecture was chosen. The signature detection system runs on a server with multiple GPUs and hosts a restful web service for access to the system. The YOLOv5 implementation is based on PyTorch and Python, therefore the signature detection system was implemented with the same technologies. FastAPI was employed to create a restful web service. The web framework facilitates rapid development of web services following open web standards such as OpenAPI and JSON Schema, as well as automated generation of documentation and tests. Building applications can be composed of hundreds of pages that need to be analyzed, therefore these documents are not serialized and included as binary data in the http request sent to the prediction server but rather included as a link to the building applications file located in a data center of the City of Vienna. This avoids redundantly transferring data. A prediction request to the server is performed as follows, first, a request including links to documents that should be analyzed as well as a sensitivity threshold is sent to the FastAPI server. This request is performed asynchronously to avoid timeouts as this process can take a couple of minutes depending on the number of files. Second, the FastAPI server downloads all requested documents and strips them in case of PDFs to pages and converts them to jpg files. These files are then passed to the predictor with the specified threshold. Third, the predictor detects signatures in each file, the coordinates of their bounding boxes as well the signature image are saved. Meanwhile, the client receives link to a queue where the status of the predictions can be periodically checked. After inference is finished, the client downloads the predictions from the queue, images of the signatures are included as base64 encoded images.

A.2.2 Prediction Request

For prediction, the client sends a request including the files to predict, specified with their URI on the fileserver, and a confidence threshold to the predict endpoint at the webserver via HTTP POST. Figure A.1 shows an example prediction request. Afterwards the client receives a polling link, where it can periodically check for prediction results. The polling link is shown in Figure A.2.

Now, the client frequently checks via HTTP GET for prediction results at the received URL. As soon as partial results are available, they are published under the URL. After the prediction is done, a flag is set to indicate that the process is done. Figure A.3

```
{
  "documents": [
    "application1.pdf",
    "application2.pdf"
  ],
  "sensitivity": 0.5
}
```

Figure A.1: Prediction request

```
{
  "url": "http://10.0.0.1:8080/results/3093240923"
}
```

Figure A.2: Polling link

displays an example of a prediction result, filename as well as bounding boxes, confidence level and base64 preview of the signature are included in the prediction result. The base64 string is rendered to an image for better illustration. These results can subsequently be included in the building application workflow software of the City of Vienna.

A.2.3 Deployment

The signature detection system is deployed via docker. A docker compose file was created to set up the container and install required dependencies. Therefore, the system can be launched with a single command *docker-compose up*. For useful performance the sever hosting the container should be at minimum equipped with a Nvidia GeForce GTX 1080 TI graphics card. While being capable of performing inference on CPUs, the performance will highly likely not be sufficient for batch processing and serving multiple users simultaneously.

```
{
  "finished": true,
  "documents": [
    {
      "application1.pdf": {
        "bounding_boxes": []
      }
    },
    {
      "application2.pdf": {
        "bounding_boxes": [
          {
            "x_min": 100,
            "y_min": 200,
            "x_max": 400,
            "y_max": 300,
            "confidence": 0.59,
            "base64": Sincerely
          }
        ]
      }
    }
  ]
}
```



Figure A.3: Results

List of Figures

1.1	Challenges of signature detection (artificially created document)	2
2.1	Image recognition pipeline	6
2.2	SIFT pipeline	10
2.3	Machine learning pipeline	11
2.4	Biological neuron (Raschka and Mirjalili [41])	12
2.5	Artificial neuron [57]	13
2.6	Gradient descent perceptron Raschka and Mirjalili [41]	13
2.7	Convolutional Neural Network architecture [4]	16
2.8	R-CNN Architecture[15]	17
2.9	Fast R-CNN Architecture [14]	18
2.10	Faster R-CNN Architecture [45]	19
2.11	YOLO Architecture [44]	19
3.1	Examples of detected signatures from the Tobacco-800 data set, together with their saliency maps. The top three most salient parts are shown in red, green, and blue, respectively Zhu et al. [62]	26
3.2	Block diagram of system proposed by Mandal et al. [32]	27
3.3	Extracted signature keypoints (blue) and printed text keypoints (red) during training Ahmed et al. [3]	29
3.4	The crop method Madasu et al. [31]	30
3.5	A block diagram describing the steps involved in labeling scanned documents, from Shetty et al. [52]	31
3.6	Mean Average Precision analysis of all the CNN networks on Set 2 (Train 30%), from Sharma et al. [51]	32
3.7	Errors per feature, from Ribeiro et al. [48]	33
3.8	Distribution of datasets used in publications	34
4.1	Samples Signature and Logo detection results from Tobacco-800 data set. Sharma et al. [51]	38
4.2	Dimensions distribution of normalized bounding boxes	43
4.3	Distribution of base boxes	43
5.1	Selected predictions of model T_TV_FRCNN_resnet50_FPN Set 1	49
		83

5.2	Average precision of T_D_FRCNN_Resnext101_FPN validation partition of Set 1	50
5.3	Loss graph of training process of T_D_FRCNN_Resnext101_FPN on Set 1	51
5.4	Selected predictions of T_D_FRCNN_Resnext101_FPN (44/10/36 split)	52
5.5	Loss graph of training process of model T_D_Retinanet_Resnext101 on Set 1	53
5.6	Selected predictions of T_D_Retinanet_Resnext101 (44/10/36 split) . .	54
5.7	Development of performance of model T_Y_YOLOv5x on set 1	56
5.8	Loss of model T_Y_YOLOv5x on set 1 on training and validation set . .	57
5.9	Selected predictions of T_Y_YOLOv5x (44/10/36 split)	58
5.10	Performance evolution of W_Y_YOLOv5x during training	61
5.11	Loss on training and validation data set of W_Y_YOLOv5x during training	62
5.12	Performance evolution of W_Y_YOLOv5x_TF during training	63
5.13	Loss on training and validation data set of W_Y_YOLOv5x_TF during training	64
5.14	Accuracy of W_Y_YOLOv5x_TF (orange) and W_Y_YOLOv5x (green) during training	65
5.15	Training process of W_D_FRCNN_Resnext101_FPN set (W) 4	66
5.16	Training process of W_D_FRCNN_Resnext101_CA set (W) 4	67
5.17	AP of W_D_FRCNN_Resnext101_CA (orange) and W_D_FRCNN_Resnext101_FPN (gray) during training	68
5.18	Training process of W_D_FRCNN_Resnext101_TF	69
5.19	Training process of W_D_FRCNN_Resnext101_TF (orange) and W_D_FRCNN_Resnext101_FPN (gray) on set (W) 4	70
5.20	Training process of W_D_Retinanet_Resnext101 set (W) 4	71
A.1	Prediction request	81
A.2	Polling link	81
A.3	Results	82

List of Tables

4.1	Tobacco 800 data set training, validation and testing splits	39
4.2	City of Vienna data set training, validation and testing splits	40
5.1	Performance of model T_TV_FRCNN_resnet50_FPN across varying epochs and data splits with decaying learning rate (LR), all layers apart from last are fixed	48
5.2	T_D_FRCNN_Resnext101_FPN over various iterations (IT), learning rates (LR) and different test train splits, stem + first stage were frozen	50
5.3	T_D_Retinanet_Resnext101 over various iterations (IT), learning rates (LR) and different test train splits	53
5.4	Performance of model T_Y_YOLOv5x on different data set splits	55
5.5	Best performing models	59
5.6	Performance of model W_Y_YOLOv5x	60
5.7	Performance of model W_Y_YOLOv5x_TF on different data set splits	63
5.8	Performance of model W_D_FRCNN_Resnext101_FPN (default anchors) on different data set splits	65
5.9	Performance of model W_D_FRCNN_Resnext101_CA (custom anchors) on different data set splits	66
5.10	Performance of model W_D_FRCNN_Resnext101_FPN_TF on different data set splits	68
5.11	Model W_D_Retinanet_Resnext101 performance across various data set splits	70
5.12	Comparison of best performing models	70



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

- [1] The Legacy Tobacco Document Library (LTDL), University of California, San Francisco. URL <https://www.industrydocuments.ucsf.edu/tobacco/>.
- [2] Gady Agam, Shlomo Argamon, Ophir Frieder, David Grossman, and David Lewis. The complex document image processing (CDIP) test collection project. *Illinois Institute of Technology*, 2006.
- [3] Sheraz Ahmed, Muhammad Imran Malik, Marcus Liwicki, and Andreas Dengel. Signature segmentation from document images. *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pages 425–429, 2012. ISSN 15505235. doi: 10.1109/ICFHR.2012.271.
- [4] Md. Zahangir Alom, Tarek Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Nasrin, Mahmudul Hasan, Brian Essen, Abdul Awwal, and Vijayan Asari. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, 8: 292, 2019. doi: 10.3390/electronics8030292.
- [5] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [6] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007. ISBN 9780898716245.
- [7] Michael Blumenstein, Xin Yu Liu, and Brijesh Verma. An investigation of the modified direction feature for cursive character recognition. *Pattern Recognition*, 40 (2):376–388, 2007. ISSN 00313203. doi: 10.1016/j.patcog.2006.05.017.
- [8] Alexey Bochkovskiy, Chien Yao Wang, and Hong Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020.
- [9] S Allen Broughton and Kurt Bryan. Convolution and Filtering. *Discrete Fourier Analysis and Wavelets*, pages 139–183, 2018. doi: 10.1002/9781119473329.ch4.
- [10] Salim Djeziri, Fathallah Nouboud, and Réjean Plamondon. Extraction of signatures from check background based on a filiformity criterion. *IEEE Transactions on Image Processing*, 7(10):1425–1438, 1998. ISSN 10577149. doi: 10.1109/83.718483.

- [11] Jeffrey Elman. Finding structure in time. *Cognitive Science*, 1990. ISSN 03640213. doi: 10.1016/0364-0213(90)90002-E.
- [12] David Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.
- [13] Georgy Gimel'Farb and Patrice Delmas. *Image Processing And Analysis: A Primer*. 2018.
- [14] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. ISSN 10636919. doi: 10.1109/CVPR.2014.81.
- [16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016. ISSN 01628828. doi: 10.1109/TPAMI.2015.2437384.
- [17] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [18] Marcel René Hauri. Tobacco 800 data set split, 2021. URL <https://doi.org/10.5281/zenodo.4383235>.
- [19] 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.
- [20] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
- [21] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735.
- [22] Jan Erik Solem. *Programming Computer Vision with Python*. *Programming Computer Vision with Python*, 2012. ISSN 1098-6596.
- [23] Xudong Jiang. Feature extraction for image recognition and computer vision. *Proceedings - 2009 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009*, pages 1–15, 2009. doi: 10.1109/ICCSIT.2009.5235014.

- [24] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, STAN Christopher, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomamma, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, 10 2020. URL <https://doi.org/10.5281/zenodo.4154370>.
- [25] Michael Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 1986.
- [26] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- [27] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, page 255–258, 1998. URL <https://dl.acm.org/doi/10.5555/303568.303704>.
- [28] David Lewis, Gady Agam, Shlomo Argamon, Orphir Frieder, David Grossman, and Jefferson Heard. Building a Test Collection for Complex Document Information Processing. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06*, page 665–666, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933697. doi: 10.1145/1148170.1148307. URL <https://doi.org/10.1145/1148170.1148307>.
- [29] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10602-1{_}48.
- [30] David Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 2:91–110, 2004.
- [31] Vamsi Krishna Madasu, Mohd Hafizuddin, Mohd Yusof, and M Hanmandlu. Automatic Extraction of Signatures from Bank Cheques and other Documents. *Techniques*, pages 10–12, 2003.
- [32] Ranju Mandal, Partha Pratim Roy, and Umapada Pal. Signature segmentation from machine printed documents using conditional random field. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 1170–1174, 2011. ISSN 15205363. doi: 10.1109/ICDAR.2011.236.
- [33] Ranju Mandal, Partha Pratim Roy, and Umapada Pal. Signature segmentation from machine printed documents using contextual information. *International Journal of*

Pattern Recognition and Artificial Intelligence, 26(7):1–25, 2012. ISSN 02180014. doi: 10.1142/S0218001412530035.

- [34] Ranju Mandal, Partha Pratim Roy, Umapada Pal, and Michael Blumenstein. Bag-of-visual-words for signature-based multi-script document retrieval. *Neural Computing and Applications*, 31(10):6223–6247, 2019. ISSN 14333058. doi: 10.1007/s00521-018-3444-y.
- [35] Warren McCulloch and Walter Pitts. A logical calculus nervous activity. *Bulletin of Mathematical Biology*, 52(1):99–115, 1990. ISSN 00074985. doi: 10.1007/BF02478259.
- [36] Ajoy Mondal, Peter Lipps, and C. V. Jawahar. IIIT-AR-13K: A New Dataset for Graphical Object Detection in Documents, 2020.
- [37] Amit Vijay Nandedkar, Jayanta Mukherjee, and Shamik Sural. SPODS: A dataset of color-official documents and detection of logo, stamp, and signature. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10481 LNCS:219–230, 2016. ISSN 16113349. doi: 10.1007/978-3-319-68124-5{_}19.
- [38] Otsu Nobuyuki. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man and Cybernetics*, 9(1):62–66, 1996.
- [39] Ashwini Pansare and Shalini Bhatia. Handwritten Signature Verification using Neural Network. *International Journal of Applied Information Systems*, 2012. doi: 10.5120/ijais12-450114.
- [40] Fernando León Puente and Sebastian Bauer. *Praxis der Digitalen Signalverarbeitung*. 2017.
- [41] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning*. Packt Publishing Ltd., 2015. ISBN 1789955750.
- [42] Joseph Redmon and Ali Farhadi. Yolo V2.0. *Cvpr2017*, 2017. ISSN 0146-4833.
- [43] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement, 2018.
- [44] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:779–788, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.91.
- [45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

- [46] Ives Rey Otero and Mauricio Delbracio. Anatomy of the SIFT Method. *Image Processing On Line*, 4:370–396, 2014. ISSN 2105-1232. doi: 10.5201/ipol.2014.82. URL <http://dx.doi.org/10.5201/ipol.2014.82><http://dx.doi.org/10.5201/ipol.2014.82><http://dx.doi.org/10.5201/ipol.2014.82>.
- [47] Hamid Rezatofighi, Nathan Tsoi, Jun Young Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. *arXiv*, 2019.
- [48] Bernardete Ribeiro, Ivo Gonçalves, Sérgio Santos, and Alexander Kovacec. Deep learning networks for off-line handwritten signature recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7042 LNCS:523–532, 2011. ISSN 03029743. doi: 10.1007/978-3-642-25085-9{_}_}62.
- [49] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. ISSN 00280836. doi: 10.1038/323533a0. URL <https://www.nature.com/articles/323533a0>.
- [50] Robert Schalkoff. *Digital image processing and computer vision*, volume 286. Wiley New York, 1989.
- [51] Nabin Sharma, Ranju Mandal, Rabi Sharma, Umapada Pal, and Michael Blumenstein. Signature and logo detection using deep CNN for document image retrieval. *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, 2018-Augus:416–422, 2018. ISSN 21676453. doi: 10.1109/ICFHR-2018.2018.00079.
- [52] Shravya Shetty, Harish Srinivasan, Matthew Beal, and Sargur Srihari. Segmentation and labeling of documents using conditional random fields. *Document Recognition and Retrieval XIV*, 6500(i):65000U, 2007. ISSN 0277-786X. doi: 10.1117/12.704410.
- [53] Priyanka Singh, Balasubramanian Raman, and Partha Pratim Roy. Detection of seal and signature entities with hierarchical recovery based on watermark self embedding in tampered digital documents. *Displays*, 54(January 2017):47–59, 2018. ISSN 01419382. doi: 10.1016/j.displa.2018.09.004. URL <https://doi.org/10.1016/j.displa.2018.09.004>.
- [54] George Stockman and Linda G Shapiro. *Computer Vision*. Prentice Hall PTR, USA, 1st edition, 2001. ISBN 0130307963.
- [55] Richard Szeliski. *Computer Vision: Algorithms and Applications* 2nd Edition. 2020.
- [56] Bernard Widrow and Michael Lehr. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990. doi: 10.1109/5.58323.

- [57] Joseph Yacim and Douw Boshoff. Impact of Artificial Neural Networks Training Algorithms on Accurate Prediction of Property Values. *Journal of Real Estate Research*, 40:375–418, 11 2018.
- [58] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1, 2014. URL <http://arxiv.org/abs/1411.1792>.
- [59] Andreas Zell. *Simulation neuronaler netze*, volume 1. Addison-Wesley Bonn, 1994.
- [60] Yefeng Zheng, Huiping Li, and David Doermann. Machine Printed Text and Handwriting Identification in Noisy Document Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):337–353, 2004. ISSN 01628828. doi: 10.1109/TPAMI.2004.1262324.
- [61] Guangyu Zhu and David Doermann. Automatic Document Logo Detection. In *In Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR 2007)*, pages 864–868, 2007.
- [62] Guangyu Zhu, Yefeng Zheng, David Doermann, and Stefan Jaeger. Multi-scale structural saliency for signature detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (June), 2007. ISSN 10636919. doi: 10.1109/CVPR.2007.383255.
- [63] Guangyu Zhu, Yefeng Zheng, David Doermann, and Stefan Jaeger. Signature detection and matching for document image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2015–2031, 2009. ISSN 01628828. doi: 10.1109/TPAMI.2008.237.