A Generic Method for Stamp Detection and Classification using Machine Learning Object Detection Frameworks

Malte Zietlow

Department of Computer Science
Nordakademie
Köllner Chaussee 11
25337 Elmshorn

malte.zietlow@nordakademie.de

24th January 2021

Abstract

In this paper, the well-known SSD framework [Liu+16] for object detection is thoroughly introduced. For many of its building blocks, motivations are provided that were omitted in its original formulation. The framework is implemented, using MobileNetV1 as its base network without pre-training. The model is then applied to a well known public data set for stamp detection (StaVer). As a metric, mean Average Precision (mAP) is reported and visual examples of prediction results are discussed. The method produced generally favorable results and was able to overcome most of the flaws reported by related work.

Contents

List of Figures

Glossary

Bounding Box The smallest rectangle that can be drawn around an object, s.t. all pixels from that object are contained. ix, x, 1–12

COCO Common Objects in Context (COCO). A well known dataset for object detection, segmentation and captioning. ix

Convolutional Layer Composition of the convolutional operation and nonlinearity, partially referring to the 'complex layer terminology' in [GBC16, p. 341]. 4, 7–9

Deconvolutional Layer Composition of the deconvolutional operation and nonlinearity, partially referring to the 'complex layer terminology' in [GBC16, p. 341]. 3

Dense Layer Also: fully connected layer. Composition of the matrix operation and nonlinearity, partially referring to the 'complex layer terminology' in [GBC16, p. 341]. 2, 3, 7, 8

Feature Map Output of a convolutional layer after applying an activation function like Rectified Linear Unit (ReLU). i, 4–9

Ground Truth The correct labels for an example. i, vii–ix, 4–12

Open Images Google Open Images is a well known dataset/challenge for object detection and classification. ix

Pascal VOC Pascal VOC is a well known dataset/challenge for object detection and classification. ix, 3

Acronyms

AP Average Precision. ix, x

CNN Convolutional Neural Network. 4

FCN Fully Convolutional Neural Network. 3

FN False Negative. ix

FP False Positive. ix, x, 2, 13

IoU Intersection over Union. i, viii, ix, 6, 10, 12, 13

mAP mean Average Precision. vii, ix, x, 11–13

PCA Principal Component Algorithm. 2

ReLU A common activation function for neural neworks. $f(x) = \max(0, x)$. ii

SSD Single Shot MultiBox Detector. vii, 4, 8, 9, 11–13

StaVer Stamp Verification Dataset [MvS15]. i, xii, 1, 11–13

SVM Support Vector Machine. 3

TP True Positive. vii, ix, x

YOLO You Only Look Once. 4

1 Introduction

Digitalization is a recurring topic in German politics. Especially in governmental contexts however, formal letter writing is still sometimes a requirement.

This paper is motivated by a use case identified within the author's corporation, regarding incompletely filled out billing documents. While the respective document contains rows explicitly for the name and address of the contractor, it is common practice that contact details are not provided via the fields. Instead, contractors use an ink stamp. Unfortunately, this practice excludes the document from currently applied simple automation pipelines. Therefore, a multistage plan has been forged, to enable automated document processing. Its first stage comprises the detection of a bounding box around the stamp.

The paper is structured as follows. In section 2, related work on the topic of stamp detection is collected and discussed. In section 3, the method that will be applied in this paper is thoroughly discussed. In section 4.2, a public dataset (StaVer) for stamp detection is introduced, and results are discussed. Finally, in section 4.3, the paper is wrapped up and future research directions are presented.

Unfortunately, introduction to fundamental concepts from statistical learning is beyond the scope of this paper. The reader is therefore expected to be roughly familiar with basic terminology, regarding especially the usage of neural networks.

2 Related Work

In the past, a variety of methods for stamp detection has been proposed. One patent on stamp detection, for example, dates back as far as 1963 [SE63].

A typical strategy found in most publications is to first separate a set of stamp-candidates from text and background, followed by a *verification* of those candidates by applying thresholds on multiple previously chosen features.

To provide an overview of the topic, recent work on stamp detection is split into two major- and respective sub-categories in the following.

Restricted Approaches

`Color restricted` Micenková and van Beusekom [Mv11] present an approach based on color clustering and geometric features. To extract a set of stamp-candidates from a given image, they assume stamps to be chromatic objects. In consequence, all achromatic parts are dropped off the image. To separate the remaining candidates for individual analysis, the XY-Cut algorithm proposed in [Kri+93] is applied. For classification, an ensemble of geometric- (pixel density, skew, etc.) and color-related features (standard deviation of hue) is compared with empirically

set thresholds. By treating stamps as chromatic objects, they are unable to detect black stamps and face problems when colored fonts are used. In [MvS15], Micenková et al. improve upon this method by applying additional geometric features.

A predecessor of the approach by Micenková and van Beusekom [Mv11] is [FF10] which uses fewer features for *verification* and is only applicable to blue or red colored stamps.

`Geometrically restricted` An early attempt of using geometric analysis was formulated by Zhu, Jaeger and Doermann [ZJD06], proposing an ellipse detection algorithm inspired by Hough transform [DH72]. However, this method is only applicable to elliptical stamps.

Generic Approaches

`Geometric features` A more versatile method than [ZJD06] was proposed in [Ahm+13; Ahm16]. Ahmed et al. make use of advances in research of *feature detectors* and *feature descriptors*. In a first step, the image is binarized [TM20] and connected components are extracted. For every connected component, *feature key points* and *feature descriptors* are computed via FAST [RD05] and ORB [Rub+11]. Further, bounding box height and width of each connected component are extracted. *Verification* is performed by comparing the extracted features of each connected component to a validation set [GBC16, pp. 120 sq.]. However, the authors notice low precision- and recall-rates, caused by their inability to recognize severely overlapping stamps.

In [NMS15a], Nandedkar et al. extend an approach they suggested earlier in [NMS15b]. They analyze spatial frequency components in document images, finding text to be the major contributor. To segment a given image, they first conceal text-components by removing objects with high spatial frequencies and further apply a Gaussian filter. To prevent textual stamps from being removed, a chromaticity threshold is applied on the removal of spatial frequency components. Stamp-candidates are then identified using mean-shift segmentation [FH75]. Small regions are filtered with an empirically set threshold, where the largest region is considered to be the background. For *verification* of stamp-candidates, AlexNet [KSH12] is used, producing three classes (logo, stamp and noise) in the final dense layer. Even though unmentioned in the paper, the presented approach of filtering spatial frequency components faces limitations regarding black textual stamps which might turn out to be removed by the spatial frequency filter.

`Mixed features` Dey et al. propose an outlier-based method in [DMS15]. To segment a given image, it is separated in foreground and background using binarization [TM20]. To retain color information the obtained foreground-values are applied as a mask to the original image. Color information is reduced from three-dimensional *RGB* color space to a one-dimensional principal component using Principal Component Algorithm (PCA) [Hot33]. Stamp-candidates are obtained using a connected-components algorithm on the first principal component (from PCA). To filter out False Positives (FPs), thresholds on features such as stroke width, bounding box width and height are applied. For *verification*, a set of features is extracted and validated using additional empirically set thresholds.

A sliding window approach using the AdaBoost [FS95] on cascaded decision trees has been proposed in [FM16]. To train AdaBoost, a labeled training set is constructed. For each training sample, Haar-like features [VJ01] are extracted. AdaBoost then constructs a set of consecutive decision trees to optimize stamp classification of training samples. Each decision tree takes in a Haar-like feature from a fixed position inside the sliding window. Based on the input feature value, the tree decides whether to *accept* or *reject* a given window as stamp-candidate. A set of consecutive decision trees with such property (*accept* or *completely reject*) is then called a *cascade*. In order for a sliding window to become a valid stamp-candidate it has to survive such a cascade of decision trees. The set of generated candidates is then *verified* using a broader set of features. Several algorithms are compared for *verification* including neural networks, decision trees, and Support Vector Machines (SVMs).

Younas et al. [You+17] propose using a Fully Convolutional Neural Network (FCN), built upon the VGG-16 [SZ15] network architecture. In this approach, the final dense layers of VGG-16 are replaced with a deconvolutional layer. By doing this, a prediction map is received as output instead of classification scores. To reduce training time, transfer learning from the Pascal VOC2011 dataset is applied (although without ablation study of the impact of transfer learning). However, no preprocessing is mentioned. Using 90% of their dataset for training and 10% for *verification* they report "state-of-the-art" results, although noticing difficulties with tabular stamps.

Despite numerous research carried out on stamp segmentation, most of the approaches examined in this chapter either face limitations with overlapping objects [NMS15b; NMS15a; FM16; Ahm+13; DMS15; FM15; FM16] or logo misclassification [Ahm+13; DMS15; Mv11]. Some approaches are limited by runtime, operating for more than 13 Seconds per evaluation [Ahm16; NMS15b].

Besides segmentation, classification of stamps is observed to be a common task in literature. Classes are usually either binary (Stamp and non-Stamp), object-related (Stamp, Logo, Text, Tables) [FM16; NMS15b; NMS15a; DMS15], or shape-dependent (Circle, Ellipse, Square, etc.) [FM15]. However, research on classification of similar stamps i.e. stamps belonging to the same shape-class but differing in text and decor, has so far only been reported on by [PG13].

3 Methodology

To address the challenges identified in section 2, namely evaluation-speed and overlap, recent advances in machine learning are considered. Contrary to most approaches in the field which perform segmentation, this work focuses on detecting bounding boxes.

In this section, details on the applied method is provided. Methods for object detection can coarsely be divided into two groups: one-staged and two-staged approaches [cf. Liu+20]. To reduce complexity, only a single one-stage approach will be considered in this paper.

Single Shot MultiBox Detector

Two common one-stage architectures are You Only Look Once (YOLO) [Red+15; RF16; RF18] and Single Shot MultiBox Detector (SSD) [Liu+16]. While SSD can be extended to any base network (e.g. VGG16 [SZ15]), YOLO is limited to only Darknet [Red16]. Therefore, SSD was chosen over YOLO for this paper, retaining the ability to quickly exchange base networks.

Central concepts of SSD are described in the following.

3.1 Default Boxes

The most fundamental part of understanding SSD is understanding its specific concept of bounding boxes, default boxes and anchors. In order to reduce the very high complexity, this subsection is written in a more colloquial manner.

Slightly anticipating and simplifying section 3.2, a *prediction* from SSD is the output from a set of convolutional layers that are chosen from within the "main" network. Output of a single such layer is, first and foremost, just a feature map. The goal in stamp detection, however, is to find the coordinates of a bounding box.

Consider such a feature map with dimensions $m \times n$. Every pixel from that feature map contains highly condensed information about a specific region from the original image[1]. The feature map is then processed further, s.t. for every pixel (and therefore the related region within the original image) the coordinates of a single bounding box are received (simplified, for details see section 3.2).

To the reader, this might seem confusing. Receiving $m \times n$ single bounding box predictions for every pixel within the feature map for every chosen layer makes the number of predicted bounding boxes much greater than the expected number of ground truth bounding boxes. And it raises a pressing question: **How is training data constructed for such an unusual architecture?**

This question is answered (partly) by the introduction of **anchors**.

Anchors

Every pixel from a feature map is related to a region within the original input image (*receptive field* [cf. GBC16, pp. 331 sq.]). Furthermore, convolutional layers preserve the spatial structure of a convolved image [cf. GBC16, pp. 335 sqq.].

Therefore, every *pixel* from within the feature map can be related (very easily) to a region within the original image. The center of such region is called the **anchor**. The x and y coordinates for every pixel p_{ij} can then be calculated using eqs. (1) and (2). A practical example for an image I of height and width $I_w \times I_h = 300 \times 300$ and a feature map M of height and width $M_w \times M_h = 19 \times 19$ is shown in fig. 1.

[1] This concept is called the *receptive field* of Convolutional Neural Networks (CNNs), [cf. GBC16, pp. 331 sq.]

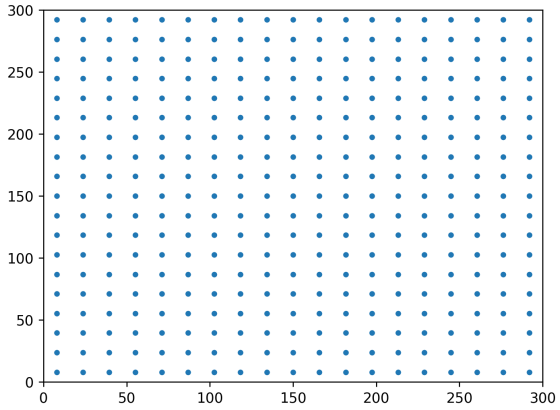


Figure 1: Generated anchors (blue) for an image of dimensions 300 × 300 and feature map of 19 × 19.

$$x_i = 0.5 * \frac{I_w}{M_w} + \sum_{0}^{i-1} \frac{I_w}{M_w} \tag{1}$$

$$y_j = 0.5 * \frac{I_h}{M_h} + \sum_{0}^{j-1} \frac{I_h}{M_h} \tag{2}$$

where:

x_i := x-coordinates of the anchor for pixel p_{ij}
y_j := y-coordinates of the anchor for pixel p_{ij}
w_I := width of input image I
w_M := width of feature map M
h_I := height of input image I
h_M := height of feature map M

Now, training data could be constructed by assigning every ground truth bounding box to its closest anchor (from every chosen layer). An example of this is given in eq. (3), where the first four rows are x- and y-coordinate, height and width of the respective bounding box and the last row is the class (0:

no-object, 1: *object*).

$$\begin{matrix} \dots \\ \begin{bmatrix} \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 \\ 271 & 828 & 182 & 845 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \\ \dots \\ \begin{bmatrix} \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 \\ 3141 & 592 & 653 & 59 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \\ \dots \end{matrix} \tag{3}$$

This approach is flawed in two ways:

1. Assigning ground truth bounding boxes to the closest anchor is entirely agnostic of the expected size of the receptive field for the different layers. A layer with smaller receptive field per pixel would probably not be able to perceive larger objects, while a layer with a larger receptive field might overlook smaller objects.

2. The model has no *a priori* knowledge of *position*, of *x*- and *y-coordinates*, of *width* and *height*. This is especially critical for applications of the model to images with varying resolutions.

A possible solution to these flaws are **default boxes**.

Default Boxes

Default boxes are used to assign spatial representation to anchors. With such a representation, every ground truth bounding box can be matched to a set of anchors more precisely (by overlap[2] with respective default boxes).

As of now, a set of chosen layers $\mathbf{L} = \{L_0, L_1, \dots, L_{n-1}\}$ and a set of anchors for every such layer[3] $\mathbf{A} = \{A_0, A_1, \dots, A_{n-1}\}$ were constructed.

Next, although the exact receptive fields of the chosen layers are uncertain, their sizes, $|recept\ (L)|$, *are* known to be strictly increasing with subsequent layers, i.e. $|recept\ (L_0)| < \dots < |recept\ (L_{n-1})|$.

Fortunately, Liu et al. [Liu+16] claim that the exact size of the receptive field is of subordinate importance. Instead of calculating the exact receptive field, they propose to assign a fixed edge length s_k for every chosen layer per eq. (4):

$$s_k = s_{\min} + k * \frac{s_{\max} - s_{\min}}{n - 1} \tag{4}$$

[2]The overlap could then be computed via IoU for example — as is done in eq. (6).
[3]More precisely, a set of anchors for the feature maps produced by such layer.

where:

$$
\begin{aligned}
n &:= |L| \\
k &\in [0, \text{n-1}] \\
s_{\text{min}} &= 0.2 \\
s_{\text{max}} &= 0.9
\end{aligned}
$$

Equation (4) produces values within $[s_{\text{min}}, s_{\text{max}}]$ or $[0, 9]$ for the default parameters, where the value increase strictly per layer (like their receptive fields do). In the following s_k is used as a percental scaling factor to determine height and width of the default box.

A default box $d_{i,j}$ for a pixel $p_{i,j}$ within a given layer L_k can then be computed as given in eq. (5) (with h: height, w: width, x: x-coordinate of the box's center, y: y-coordinate of the box's center).

$$
\begin{aligned}
d^h_{i,j} &= s_k * h_I \\
d^w_{i,j} &= s_k * w_I \\
d^x_{i,j} &= x_i \text{ (see eq. (1))} \\
d^y_{i,j} &= y_j \text{ (see eq. (2))}
\end{aligned}
\tag{5}
$$

It is now possible to match every ground truth bounding box to a set of default boxes per eq. (6).

$$
match(b, d) = \begin{cases} 1 & \text{if IoU } (b, d) > 0.5 \\ 0 & \text{otherwise} \end{cases}
\tag{6}
$$

where:

$b :=$ coordinates of a given bounding box
$d :=$ coordinates of a given default box

Thereby[4], flaw 1.) from anchor construction can be considered as solved.

Flaw 2.) can now be tackled multiple ways. The most obvious would be to convert ground truth bounding boxes and into the *percental* space (i.e. $[0, 1]$) and predict their percental coordinates. However, this would again require the model to learn different semantics per pixel within a feature map. For example, the upper left pixel/region would produce predictions within $[0, 0.2] \times [0, 0.2]$, while the lower right region would produce predictions within $[0.8, 1.0] \times [0.8, 1.0]$. This is problematic, because convolutional layers share weights[5] between inputs (i.e. such pixels/regions) [cf. Mur12, pp. 564 sqq.]. Requiring different semantics *between* these pixels introduces additional complexity to the model. To alleviate this complexity, rather than coordinates, the offset *between* the coordinates (of ground truth

[4]— by matching ground truth bounding boxes to anchors based on their related default boxes which in turn are coarsely based on receptive fields.
[5]In fact, sharing weights between inputs is the key distinguishing feature between convolutional layers and dense layers.

bounding boxes and the related default boxes) are predicted. This aligns the task between all pixels of a feature map. Normalized offsets o can be computed via eq. (7).

$$
\begin{aligned}
o^h &= \log\left(b^h\right) - \log\left(d^h\right) = \log\left(b^h \,/\, d^h\right) \\
o^w &= \log\left(b^w\right) - \log\left(d^w\right) = \log\left(b^w \,/\, d^w\right) \\
o^x &= \left(b^x - d^x\right) \,/\, d^x \\
o^y &= \left(b^y - d^y\right) \,/\, d^y
\end{aligned}
\tag{7}
$$

where:

b := A ground truth bounding box
d := A default box

Liu et al. [Liu+16] introduce a last quirk inspired by the concept of priors from Bayesian statistics [cf. Mur12, pp. 165 sqq.]. To improve convergence they choose multiple aspect ratios (i.e. 1:1, 1:2, 1:3, 2:1, 3:1) for their default boxes. In an attempt to reduce the extent of this paper, the reader is referred to [Liu+16]. The number of aspect ratios r, and thereby the number of default boxes per anchor, will be omitted from all equations in the following.

This finalizes the introduction to anchors and default boxes. For reference, confer [Liu+16]. Following, the concrete architecture of SSD is discussed.

3.2 SSD Architecture

As mentioned in the introduction to this section, SSD is independent of any specific base network. Exemplary, fig. 2 shows the architecture of SSD for VGG16 [SZ15]. Dense layers of VGG16 are dropped and replaced with additional convolutional layers (blue in fig. 2). Let now Network := Base Network → Additional Layers. Then (as briefly addressed in section 3.1), a set of convolutional

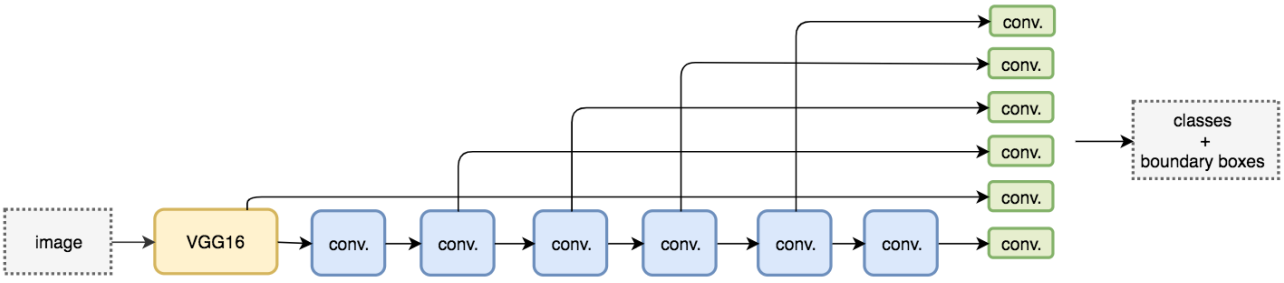


Figure 2: Example of the SSD architecture using VGG16 as its base network [cf. Liu+16].

Blue colored boxes represent the additional convolutional layers that are added to the base network.

Green boxes represent the final additional convolutional layers that produce the class- and bounding box predictions.

layers $\mathbf{L}$ is chosen from the network (i.e. $\mathbf{L} \subseteq$ Network)[6].

To produce class- and bounding box predictions, every chosen layer $L \in \mathbf{L}$ is fed into a final additional convolutional layer (i.e. one additional convolutional layer per layer in $\mathbf{L}$, depicted green in fig. 2).

As discussed in section 3.1, the model is supposed to predict the quadruple $o = (o^h, o^w, o^x, o^y)$. To enable also the prediction of different object classes, c is appended to the quadruple o, where c is a one-hot encoded vector [cf. Mur12, p. 35] of all classes[7]. The output size per chosen layer $L \in \mathbf{L}$, with feature map size $m \times n$ can be determined as $m \times n \times (4 + |c|)$.

The output size of a 2D-convolutional layer is the triplet (width × height × filter size). Width and height are calculated via eq. (8), the filter size is chosen without constraints.

$$W' = (W - F + 2P) \;/\; S + 1 \tag{8}$$

where:

W' := output size
W := input size
K := kernel size
P := padding size
S := stride size

To produce an output of size $m \times n \times (4 + |c|)$, kernel, padding, and stride sizes are chosen $K = 3, P = 1, S = 1$; filter size is chosen as $(4 + |c|)$.

This concludes the overall SSD architecture.

3.3 Loss Function

From previous subsections, a set of layers $\mathbf{L}$ has been chosen from the network $\mathbf{L} \subseteq$ Network. For the output (i.e. the feature map) of every such layer a set of default boxes $D \in \mathbf{D}$ has been generated. Also, for every image a set of ground truth bounding boxes B is given from which the offsets O are computed following eq. (7) and $o_{b,d}$ is the offset related to bounding box b and default box d. From the model architecture section 3.2 it becomes apparent, that the number of matching (eq. (6)) default boxes is

[6] The motivation behind choosing multiple layers $\mathbf{L}$ from the network is as follows: the feature maps produced by the convolutional layers get smaller with every additional layer. Colloquially speaking, the relationship between such a small feature map and the input image is, that one pixel within the small feature map is related to multiple pixels within the original input image. Therefore, when looking at the feature maps from subsequent convolutional layers, one is looking at information that is extracted from the image at different scales.

Making this logic concrete, in a larger feature map, one pixel might contain condensed information about objects like wheels or headlights, whereas in a smaller feature map one pixel might contain condensed information about the entire car. This concept is called the *receptive field* [cf. GBC16, pp. 331 sq.].

[7] Actually, c is the one-hot encoded vector of all classes +1. This additional class is the *no-object*-prediction. This is required because obviously most areas within the image contain no classifiable object, and it is desirable that the model is able to identify areas not containing an object.

much smaller than the total number of default boxes. To prevent the model from overfitting [cf. GBC16, pp. 111 sqq.], for example s.t. it detects only the *no-object* class, the *amount* of negatives is pruned[8] to be only three times greater than the number of positives.

This is done by ordering the set of negatives (i.e. not-matched default boxes) by their class confidence score[9] and then cutting it off at index $\lfloor 3 * |Positives| \rfloor$.

The set of matching default boxes ground truth bounding box offsets is built following eq. (9).

$$\begin{aligned} &\text{Let } (D \in \mathbf{D}) \wedge (d \in D) \wedge (b \in B) \\ &\text{Then:} \\ &\quad \text{Pos} = \left\{ \left(d, o_{b,d}\right) \middle| \, \text{IoU}\left(d,b\right) > 0.5 \vee \forall D' \in \mathbf{D}. \nexists d' \in D' : \text{IoU}\left(d,b\right) > \text{IoU}\left(d',b\right) \right\} \end{aligned} \tag{9}$$

That is, $\left(d, o_{b,d}\right)$ are added to the set, if their IoU is either greater than 0.5 or if no other default box (out of all default boxes over all layers) has a larger IoU. The set of negative default boxes is then simply the complement as per eq. (10).

$$\text{Neg} = \{d \mid (D \in \mathbf{D}) \wedge (d \in D) \wedge d \notin \text{Pos}\} \tag{10}$$

Finally, the loss is the weighted sum of the confidence loss (eq. (11)) and the location loss eq. (12). Let $\mathbb{1}_d^p = \{0,1\}$ indicate, that the ground truth bounding box b (which was matched to default box b) is of class p.

$$L_{\text{conf}}(l) = -\sum_{p=1}^{|C|} \left[\sum_{(d,o) \in \text{Pos}} \mathbb{1}_d^p * \log\left(\sigma\left(l_d^p\right)\right) \right] - \sum_{d \in \text{Neg}} \log\left(\sigma\left(l_d^0\right)\right) \tag{11}$$

$$L_{\text{loc}}(l) = \sum_{(d,o) \in \text{Pos}} \sum_{m \in \{h,w,x,y\}} \text{smooth}_{\text{L1}}\left(l_d^m - o^m\right) \tag{12}$$

$$\sigma\left(l_d^p\right) = \frac{\exp\left(l_d^p\right)}{\sum_{p'} \exp\left(l_d^{p'}\right)} \tag{13}$$

where:

C := is the set of all n-classes $[1,n]$ without the *no-object* class 0
l := Prediction produced by the model
l_d^p := Class prediction associated with default box d and class p
l_d^m := Bounding box prediction associated with default box d and dimension $m \in \{h,w,x,y\}$
o^m := Offset (see eq. (7)) associated with the current default box d and dimension $m \in \{h,w,x,y\}$

The final loss is now given in eq. (14).

$$L(l) = \begin{cases} \frac{1}{|\text{Pos}|}\left(L_{\text{conf}(l)} + \alpha L_{\text{loc}}(l)\right), & \text{if } \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

[8] This process is also called hard negatives mining, please confer [Liu+16].
[9] In order to spare the reader, the exact derivation of the relation which specifies the order required for pruning is omitted.

4 Experimental Results

This chapter gives an overview of the applied dataset (section 4.1) and experimental setup (section 4.2). Finally, the results are discussed in section 4.3. For the interested reader, an introduction to the applied metric (mAP) is provided in appendix A.

4.1 StaVer

SSD, as presented in section 3 will be evaluated on the publicly available dataset StaVer[10] [MvS15].

From direct examination, the dataset contains 427 images in total. However, 27 images are not annotated and can therefore not be used for train/test splits. Out of the remaining 400 images, 320 images contain colored and the remaining 80 images contain black stamps. To reduce computational complexity, a resolution of 200 dpi is chosen, although 300 and 600 dpi are also available. The dataset comprises several stamp-categories. Most distinctly, graphical and textual stamps are included. As an addition to the original dataset, the author has produced class annotation ("Text", "Image").

For every image, two different ground truth types are available. The former contains a segmentation map, the latter ground truth bounding boxes. Unfortunately, both given ground truth types are provided only in form of images figs. 3b and 3d.

Coordinates (as required for eq. (7)) are extracted from the ground truth bounding box images programmatically. OpenCV [Bra00] is used to find connected components and convert them into rectangles. Finally, coordinates are determined using `cv2.boxPoints(…)`.

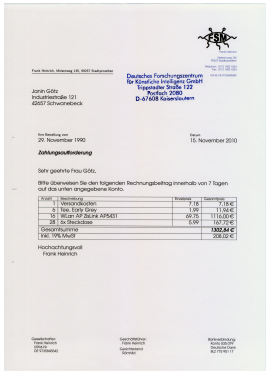
4.2 Experimental Setup

For image preprocessing, a set of manipulations

{Brightness-jitter, Color-jitter, Saturation-jitter, Horizontal-flip, Vertical-flip, Random-crop, Black-and-White}

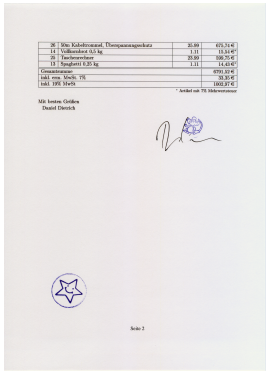
are applied randomly.

[10] https://madm.dfki.de/downloads-ds-staver
[10] Meaning that every pixel belonging to a stamp is marked.



(a) Textual stamp example from StaVer (stampDS-00251)

(b) Image containing the ground truthbounding box for the textual stamp (stampDS-00251)



(c) Two image-stamps from StaVer (stampDS-00138)

(d) Image containing the ground truthbounding box for the two image-stamps (stampDS-00138)

Figure 3: Example images from StaVer and respective ground truth bounding boxes. Unfortunately, bounding box ground truth is sometimes corrupted, e.g. (d).

SSD is implemented by the author using Python 3.7 and TensorFlow 2.0 [Aba+15]. As base network, MobileNet [cf. How+17] is adopted without pre-training[11]. Hyperparameters are chosen as per the original paper, confer [Liu+16]. Training and inference are run using `NVIDIA DGX A100` infrastructure.

Out of the 400 available and annotated images, 65% (240) are chosen for training and 35% (100) are chosen for testing. A validation set is not constructed, because no hyperparameter optimization is conducted [cf. GBC16, pp. 120 sq.].

4.3 Results & Discussion

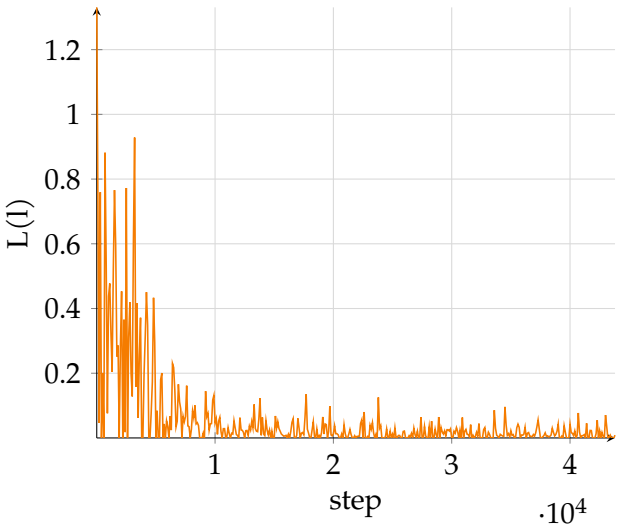
Results seem generally competitive with the original implementation by Liu et al. [Liu+16][12]. Besides that, the results from this work unfortunately cannot be compared to related approaches in stamp detection which, as noted in section 2, were mostly concerned with image segmentation.

As demonstrated in figs. 4a and 4b, the model converges after about a 10.000 training steps, reaching its maximum mAP@0.75IoU of 0.593 after 9.000 training steps.

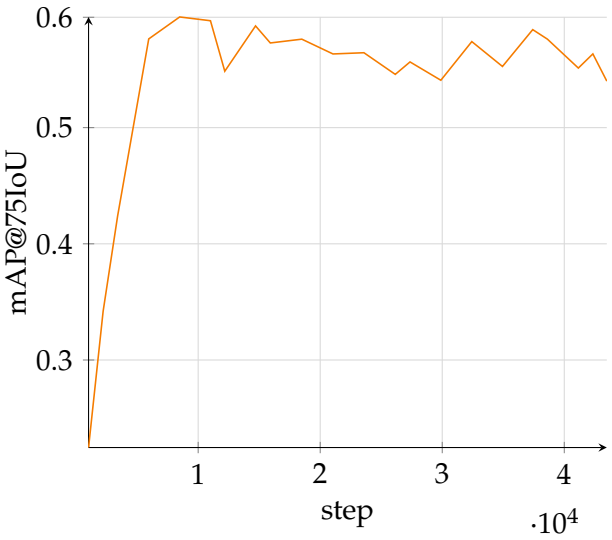
From visual evaluation, the model is able to overcome most issues noted in section 2. In most cases, black and white stamps could be identified (see fig. 8e) as well as stamps with significant overlap (figs. 8d and 9b).

[11] Common datasets like e.g. ImageNet [Den+09] come from a substantially different distribution. Note please that pre-training and thereby heterogenous transfer learning *might* be useful nonetheless [DK17], but would require additional ablation studies which is beyond the scope of this work.

[12] Comparing apples and oranges, the best implementation of SSD by Liu et al. [Liu+16] reached a mAP@.75IoU of 30.3[Table 6, `COCO test-dev2015`][Liu+16].



(a) Model loss over time



(b) mAP@.75IoU computed on the test set

Figure 4: Model loss and test metrics over time

As a downside it is noted that in few cases, SSD was vulnerable to FP detections of logos, which look very similar to graphical stamps (see fig. 8c). Also, sometimes multiple stamps are detected as a single stamp (see fig. 8b).

Interestingly, also stamps from different distributions could be identified without significant increase in FPs (see fig. 9).

5 Conclusion

In this paper, SSD [Liu+16] has been thoroughly explained, implemented by the author and applied on a publicly available dataset (StaVer [MvS15]). The achieved results were generally good and the method was able to overcome several issues reported in previous work (see section 2).

5.1 Future Research

While the results were generally good, interesting ablation studies could include the use of more powerful base networks (NASNetLarge [Zop+18], InceptionResNetV2 [SM17], ...). It would also be interesting to observe how well SSD performs on a dataset with exclusively black-and-white images.

Finally, hyperparameters could be tuned using e.g. grid search [GBC16, pp. 432–434] or Bayesian optimization [AB20]. Default box aspect ratios could be picked by observing the histogram of all aspect ratios and pick the n most common aspect ratios.

Another interesting area of research would be the adaptation of heatmaps [Mar+21] to explain the predictions produced by the SSD framework.

6 Bibliography

[AB20] Apoorv Agnihotri and Nipun Batra. 'Exploring Bayesian Optimization'. In: *Distill* (2020).

[Aba+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: `https://www.tensorflow.org/`.

[Ahm+13] Sheraz Ahmed et al. 'A Generic Method for Stamp Segmentation Using Part-Based Features'. In: *12th International Conference on Document Analysis and Recognition, ICDAR 2013, Washington, DC, USA, August 25-28, 2013*. IEEE Computer Society, 2013, pp. 708–712.

[Ahm16] Sheraz Ahmed. 'A Generic Framework for Information Segmentation in Document Images: A part-based Approach'. Dissertation. Kaiserslautern: Technische Universität Kaiserslautern, 2016.

[Bha+16] Yash Bhalgat et al. *Stamp processing with examplar features*. 2016.

[Bra00] G. Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000).

[Den+09] Jia Deng et al. 'ImageNet: A large-scale hierarchical image database'. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255.

[DH72] Richard O. Duda and Peter E. Hart. 'Use of the Hough Transformation to Detect Lines and Curves in Pictures'. In: *Commun. ACM* 15.1 (1972), pp. 11–15.

[DK17] Oscar Day and Taghi M. Khoshgoftaar. 'A survey on heterogeneous transfer learning'. In: *J. Big Data* 4 (2017), p. 29.

[DMS15] Soumyadeep Dey, Jayanta Mukherjee and Shamik Sural. 'Stamp and logo detection from document images by finding outliers'. In: *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*. IEEE, 2015, pp. 1–4. ISBN: 978-1-4673-8564-0.

[FF10] Paweł Forczmański and Dariusz Frejlichowski. 'Robust Stamps Detection and Classification by Means of General Shape Analysis'. In: *Computer Vision and Graphics*. Ed. by David Hutchison et al. Vol. 6374. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 360–367. ISBN: 978-3-642-15909-1.

[FH75] Keinosuke Fukunaga and Larry D. Hostetler. 'The estimation of the gradient of a density function, with applications in pattern recognition'. In: *IEEE Trans. Inf. Theory* 21.1 (1975), pp. 32–40.

[FM15] Paweł Forczmański and Andrzej Markiewicz. 'Stamps Detection and Classification Using Simple Features Ensemble'. In: *Mathematical Problems in Engineering* 2015.9 (2015), pp. 1–15. ISSN: 1024-123X.

[FM16] Paweł Forczmański and Andrzej Markiewicz. 'Two-stage approach to extracting visual objects from paper documents'. In: *Machine Vision and Applications* 27.8 (2016), pp. 1243–1257. ISSN: 0932-8092.

[FS95] Yoav Freund and Robert E. Schapire. 'A decision-theoretic generalization of on-line learning and an application to boosting'. In: *Computational Learning Theory, Second European Conference, EuroCOLT '95, Barcelona, Spain, March 13-15, 1995, Proceedings*. Ed. by Paul M. B. Vitányi. Vol. 904. Lecture Notes in Computer Science. Springer, 1995, pp. 23–37.

[GBC16] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[Hot33] Harold Hotelling. 'Analysis of a complex of statistical variables into principal components'. In: *Journal of educational psychology* 24.6 (1933), pp. 417–441.

[How+17] Andrew G. Howard et al. 'MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications'. In: *CoRR* abs/1704.04861 (2017).

[JWD11] Craig A. James, D. Weininger and J. Delany. 'Daylight Theory Manual. Daylight Chemical Information Systems'. In: *Inc., Irvine, CA* (2011).

[Kri+93] M. Krishnamoorthy et al. 'Syntactic segmentation and labeling of digitized pages from technical journals'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.7 (1993), pp. 737–747. ISSN: 01628828.

[KSH12] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett et al. 2012, pp. 1106–1114.

[Liu+16] Wei Liu et al. 'SSD: Single Shot MultiBox Detector'. In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*. Ed. by Bastian Leibe et al. Vol. 9905. Lecture Notes in Computer Science. Springer, 2016, pp. 21–37.

[Liu+20] Li Liu et al. 'Deep Learning for Generic Object Detection: A Survey'. In: *Int. J. Comput. Vis.* 128.2 (2020), pp. 261–318.

[Mar+21] Arturo Marban et al. 'Explaining the Decisions of Convolutional and Recurrent Neural Networks'. In: *Mathematical Aspects of Deep Learning*. Ed. by Philipp Grohs and Gitta Kutyniok. Cambridge University Press, 2021.

[Mur12] Kevin P. Murphy. *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, 2012. ISBN: 0262018020.

[Mv11] Barbora Micenková and Joost van Beusekom. 'Stamp Detection in Color Document Images'. In: *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 1125–1129. ISBN: 978-1-4577-1350-7.

[MvS15] Barbora Micenková, Joost van Beusekom and Faisal Shafait. 'Stamp Verification for Automated Document Authentication'. In: *Computational Forensics*. Ed. by Utpal Garain and Faisal Shafait. Vol. 8915. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 117–129. ISBN: 978-3-319-20124-5.

[NMS15a] Amit V. Nandedkar, Jayanta Mukherjee and Shamik Sural. 'A spectral filtering based deep learning for detection of logo and stamp'. In: *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*. IEEE, 2015, pp. 1–4. ISBN: 978-1-4673-8564-0.

[NMS15b] Amit Vijay Nandedkar, Jayanta Mukhopadhyay and Shamik Sural. 'Text-graphics separation to detect logo and stamp from color document images: A spectral approach'. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 571–575. ISBN: 978-1-4799-1805-8.

[Pad19] Rafael Padilla. *Object Detection Metrics*. 2019.

[PG13] Pjero Petej and Sven Gotovac. 'Comparison of stamp classification using SVM and random ferns'. In: *2013 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2013, pp. 000850–000854. ISBN: 978-1-4799-3755-4.

[RD05] Edward Rosten and Tom Drummond. 'Fusing Points and Lines for High Performance Tracking'. In: *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*. IEEE Computer Society, 2005, pp. 1508–1515.

[Red+15] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2015.

[Red16] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. 2016.

[Ren+15] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015.

[RF16] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016.

[RF18] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. URL: `http://arxiv.org/pdf/1804.02767v1`.

[Rub+11] Ethan Rublee et al. 'ORB: An efficient alternative to SIFT or SURF'. In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. Ed. by Dimitris N. Metaxas et al. IEEE Computer Society, 2011, pp. 2564–2571.

[SE63] Karl Steinbuch and Hermann Endres. 'Postage stamp detecting circuit arrangement'. US3087141A. 1963.

[SM17] Satinder P. Singh and Shaul Markovitch, eds. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. AAAI Press, 2017.

[SZ15] Karen Simonyan and Andrew Zisserman. 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann Lecun. 2015.

[TM20] Chris Tensmeyer and Tony R. Martinez. 'Historical Document Image Binarization: A Review'. In: *SN Comput. Sci.* 1.3 (2020), p. 173.

[VJ01] P. Viola and M. Jones. 'Rapid object detection using a boosted cascade of simple features'. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. IEEE Comput. Soc, 2001, pp. I-511-I–518. ISBN: 0-7695-1272-0.

[You+17] Junaid Younas et al. 'D-StaR: A Generic Method for Stamp Segmentation from Document Images'. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 248–253. ISBN: 978-1-5386-3586-5.

[ZJD06] Guangyu Zhu, Stefan Jaeger and David Doermann. 'A robust stamp detection framework on degraded documents'. In: *Document Recognition and Retrieval XIII*. Ed. by Kazem Taghva and Xiaofan Lin. SPIE Proceedings. SPIE, 2006, 60670B-60670B–9.

[Zop+18] Barret Zoph et al. 'Learning Transferable Architectures for Scalable Image Recognition'. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 8697–8710.

A Introduction to Common Metrics

To evaluate the results achieved by SSD (see section 3) and make them comparable to diverging approaches, a set of metrics is introduced in this section. While the central metric used in this paper is Class-wise Mean Average Precision, also Pixel-wise Metrics are introduced.

A.1 Pixel-wise Metrics

As a prerequisite to mAP, pixel-wise metrics such as Precision & Recall, F-Scores and Intersection over Union will be briefly explained in the following.

Precision & Recall

To the best of the author's knowledge, the most influential metric used in stamp detection is pixel-wise evaluation of the precision and recall tuple [NMS15a; You+17; Ahm+13; DMS15; Mv11; Bha+16; MvS15; NMS15a]. Often, recall and precision are used in settings with class imbalance where they provide a more sensible measure than accuracy. In the stamp detection, every image pixel can be considered to be from one of two classes, either *stamp* (i.e. positive) or *non-stamp* (i.e. negative). In training and evaluation images, *non-stamp*-pixels greatly outnumber *stamp*-pixels, thereby showing obvious class imbalance.

A definition of precision and recall is given in [GBC16, p. 423] as follows. Precision is the fraction of *true* positive (TP) pixels over all detected pixels (see eq. (15)). Recall is the fraction of *true* positive detections over all ground truth pixels (see eq. (16)). This relationship is visually depicted in

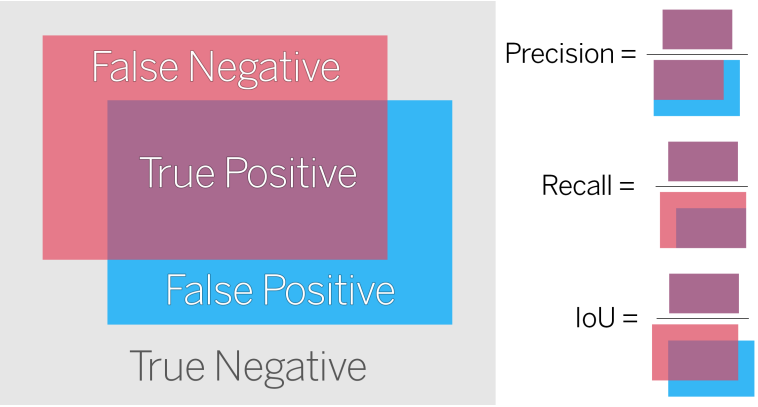


Figure 5: Visual example of pixel-wise recall, precision and IoU, with ground truth (red) and detected pixels (blue)

fig. 5.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{15}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{16}$$

Intuitively, precision is the "ability of a classifier to distinguish a negative sample from [a] positive one" [You+17], while recall is "the ability of a classifier to classify all positive samples" [You+17].

F-Scores

Precision and recall can then be combined into a harmonic average, which is usually called the family of F-Scores [e.g. Mur12, p. 183]. F_1 is given in [Mur12, p. 183] as per eq. (17).

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}$$

However, almost all works reviewed in section 2 give only precision & recall and omit F-Scores.

Intersection over Union

IoU, also known as the Jaccard-Index or Jaccard-Similarity is a metric closely related to F-Scores via the Tversky-Index (for details on this [cf. JWD11, Section 6.3 Similarity Measures]). IoU is given by

$$\text{IoU} = \frac{TP}{TP + FN + FP} \tag{18}$$

Visually, this relation is shown also in fig. 5.

A.2 Class-wise Mean Average Precision

Mean Average Precision (mAP) is a widely used metric in object detection and was adopted by well known tournaments such as COCO, Pascal VOC, and Open Images. In context of a multi-classification setting, mAP represents the arithmetic mean of Average Precision (AP) over the set of all classes. To understand mAP, TPs, FPs & FNs will be redefined for the object detection setting in the following, then AP will be explained.

TP, FP & FN for Bounding Boxes

Unlike in A.1 Pixel-wise Metrics, TPs, FPs & FNs are attributed not per individual pixel, but per bounding box. A predicted box is considered TP, if its (pixel-wise) IoU with a ground truth bounding box is greater than some threshold, FP otherwise. This threshold is often appended to the metric with an @-symbol. A False Negative (FN) is recorded when for a given ground truth bounding box no prediction had an overlap greater than the threshold.

For example, assume minimum IoU $= 0.5$(@.50IOU). Then in fig. 6, bounding boxes A and B would be considered FPs due to their low IoU, while bounding box B is a TP. We count 1 TP, 2 FP and 1 FN and thus observe Precision $= \frac{1}{3}$, Recall $= \frac{1}{2}$ (following eqs. (15) and (16)).

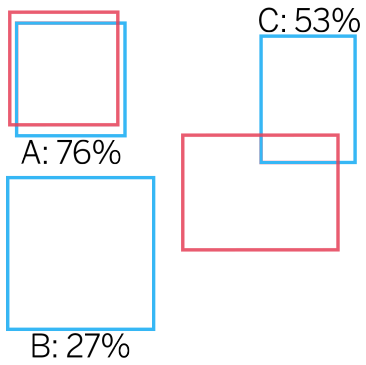
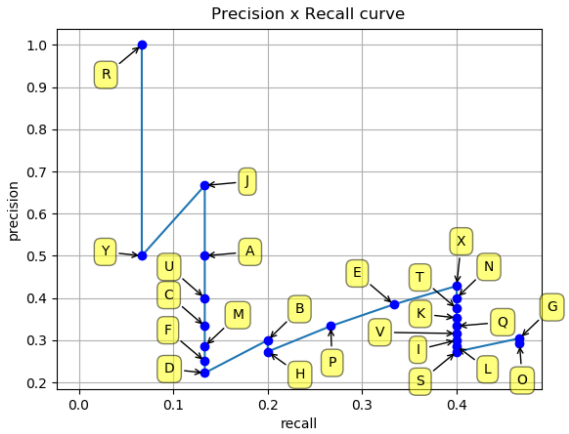


Figure 6: Visual example for bounding-box-wise confusion matrix, with ground truth bounding boxes (red) and predicted bounding boxes (blue). For each predicted bounding box a confidence score (percent) is given.

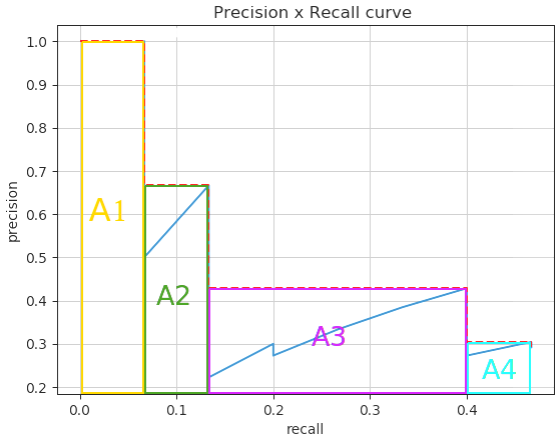
Precision-Recall-Curves and Average Precision

The Precision-Recall curve is constructed by first ordering the set of predicted bounding boxes for a class c at a given threshold (e.g. 0.5IoU) by their confidence[13]. Next, for every prediction in the ordered set $\mathbf{G}_c$@.50IoU determine whether it is a TP or FP. Finally, step through $\mathbf{G}_c$@.50IoU, accumulate TPs and FPs and report recall and precision for every element $e_c \in \mathbf{G}_c$@.50IoU, and plot them. An example of this is shown in fig. 7a.

[13] A prediction has high confidence in the predicted class, if confidence $= \max_{p \in [1,|C|]} \sigma(l_d^p)$ is high, e.g. confidence ≥ 0.9 (excluding the *no-object*)



(a) Precision-Recall-Curve. Each capital represents an inferred bounding box. Recall & Precision are plotted by accumulating TP and FP over the list of inferred bounding boxes sorted by confidence.



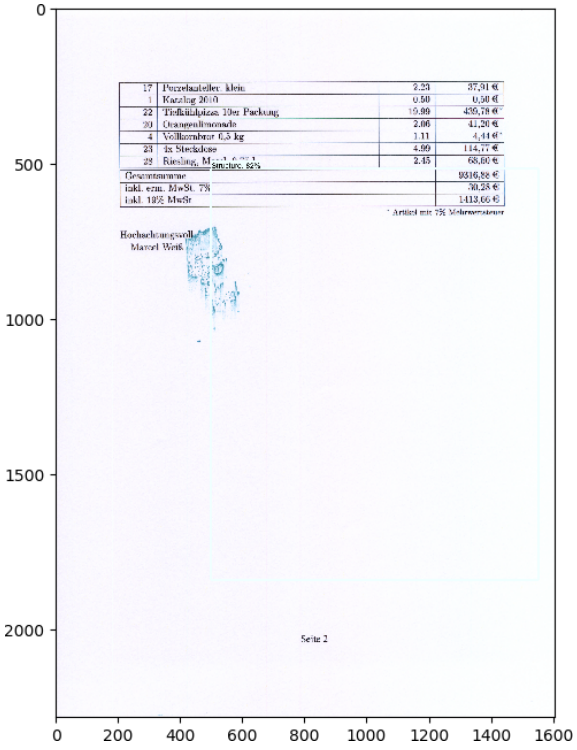
(b) Interpolated Precision-Recall-Curve with drawn-in area under the curve, i.e. AP.

Figure 7: Precision-Recall-Curve and its interpolation. Both figures where adopted from [Pad19].

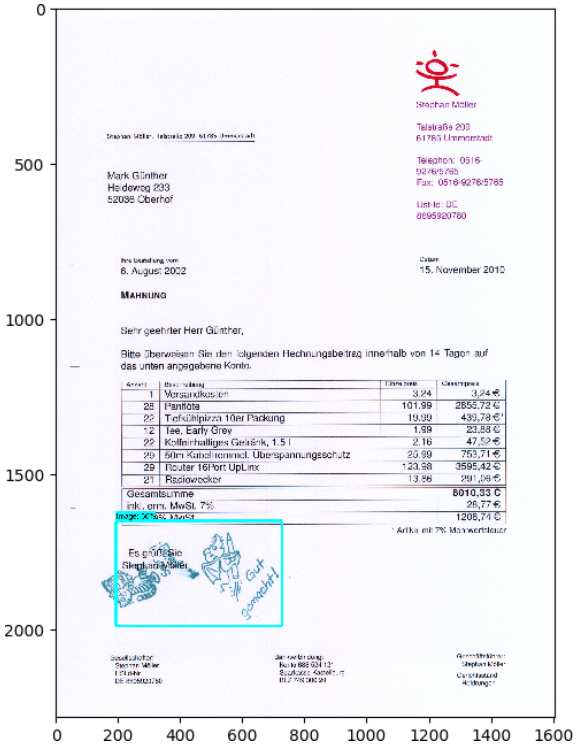
To improve comparability, the Precision-Recall-Curve $f_c(r)$ for class c can be smoothed by applying $g_c(r) = \max(f(r), f(r')), \forall r' > r$, as shown in fig. 7b. The resulting curve is then also called the *Interpolated-Precision-Recall-Curve*. AP is now defined as the curve underneath the interpolated Precision-Recall-Curve $\text{AP}_c = \int_{i=0}^{1} g_c(i)\, dx$.

Finally, mAP is the mean of AP over all classes C. That is $\text{mAP@.50IoU} = \frac{\sum_{c=1}^{|C|} \text{AP}_c\text{@.50IoU}}{|C|}$ Despite its extensive use in the object detection community [Liu+16; Ren+15], mAP was only applied in a single work [ZJD06] from 2006. This is, because i.) usually ranks are not generated in most approaches, therefore the underlying metric of average precision, which *is* computed over ranks, lacks meaning and ii.) oftentimes stamp detection is framed as a binary classification problem.

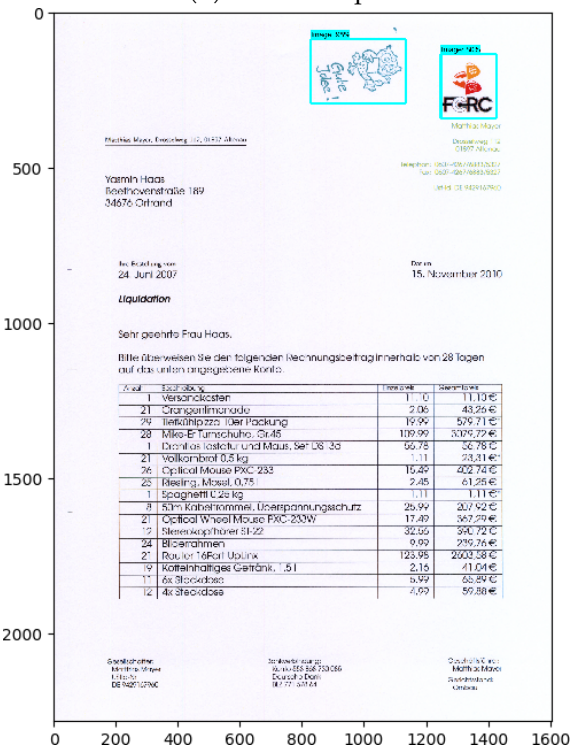
B Additional illustrations for Section 4.3 Results & Discussion



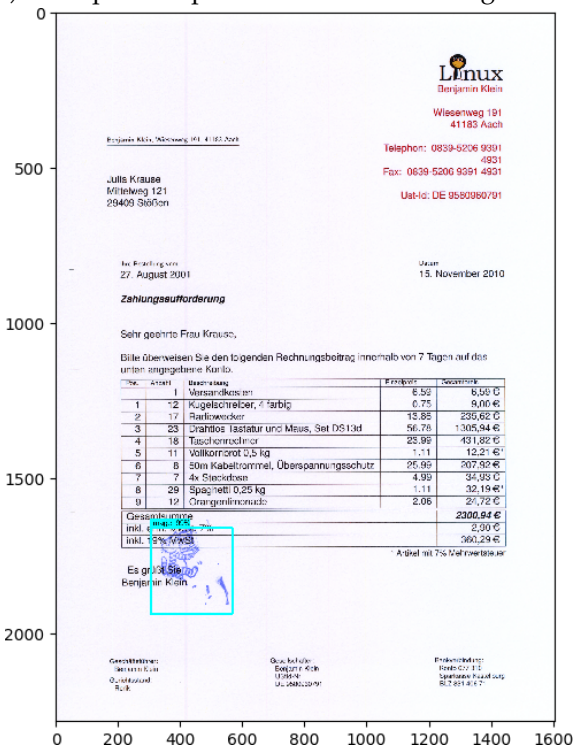
(a) Faint Stamp



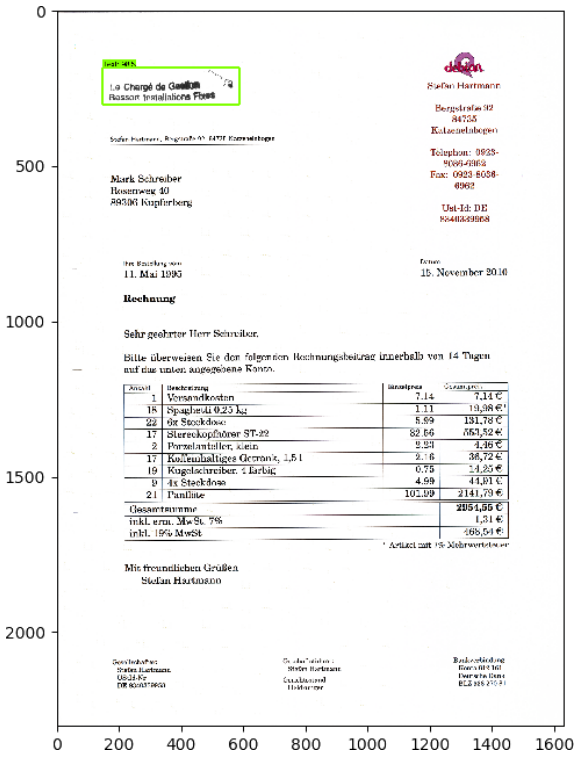
(b) Multiple stamps were detected as a single stamp



(c) Logo detected as stamp

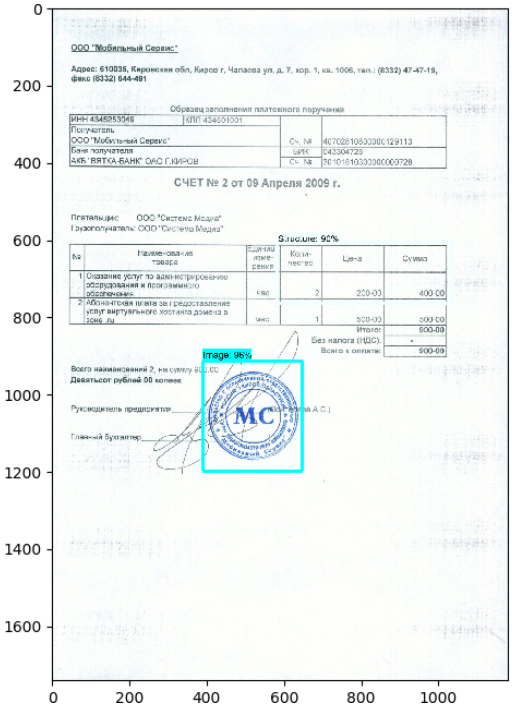


(d) A stamp that was detected despite its overlap with text.

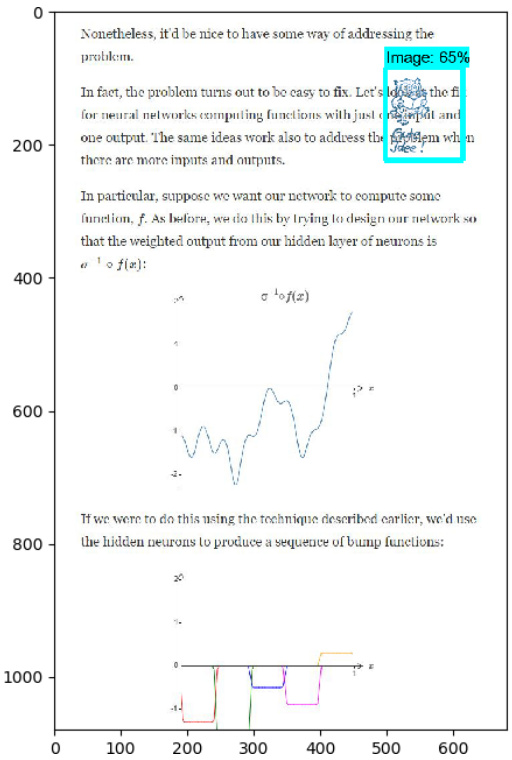


(e) A black and white textual stamp.

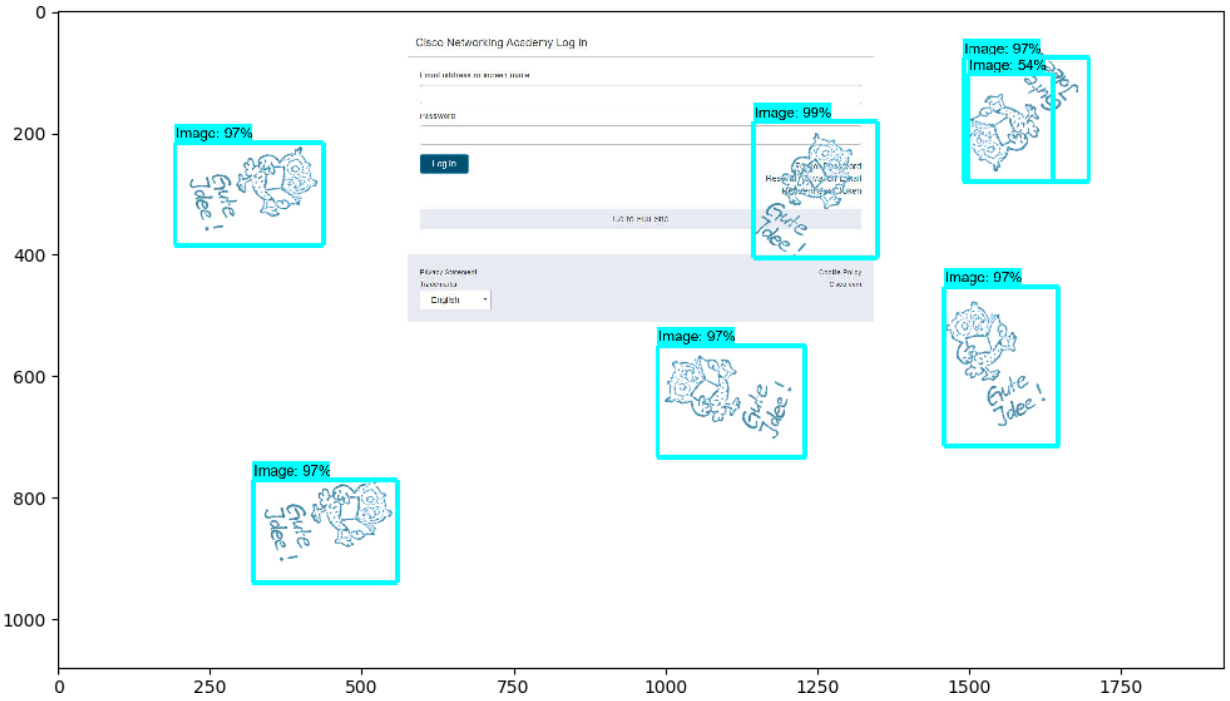
Figure 8: Examplary results



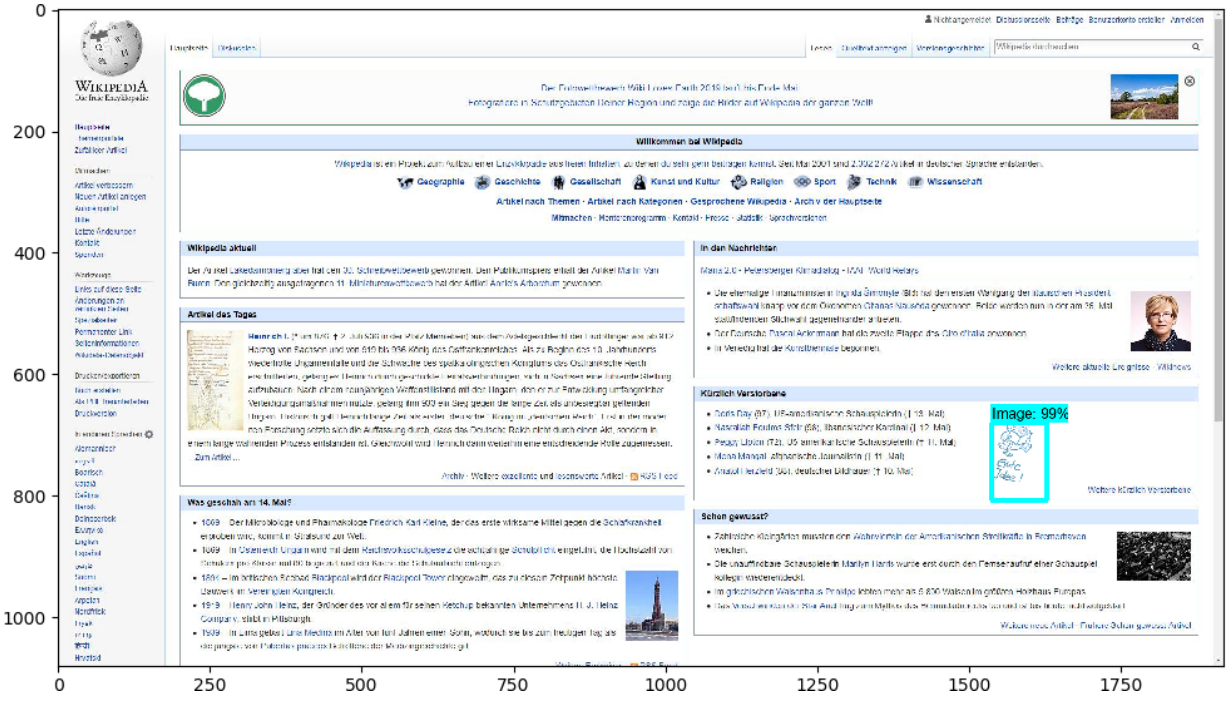
(a) A stamp from a different distribution



(b) Although the stamp is extracted from StaVer, it was placed in a different document by the author. Also see the significant overlap.



(c) An example with many stamps in one image.



(d) A difficult different distribution.

Figure 9: Examples from different domains