Heraldic Symbol Detection

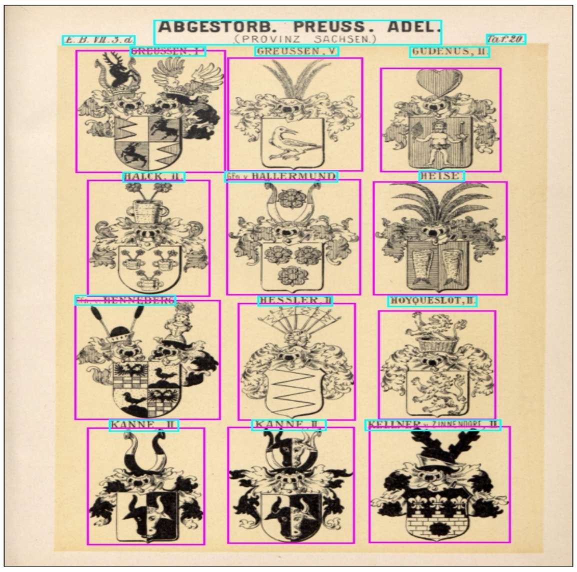
Author: Stefan Schwanecke
Program code: [tehNewton/heraldicSymbolDetection]

Summary

The task was to automatically detect heraldic symbols from images and also localize the textual annotations, belonging to these heraldic symbols.

This task was approached as an object detection problem, that can be tackled with the help of convolutional neural networks (CNNs). For this problem, a faster RCNN network with a resnet-50 backbone was chosen as the faster inference times of single shot detection architectures are not required for this task. To exploit transfer learning, a model, which was pre-trained on the COCO dataset, was fine-tuned.

To be able to fine-tune the model, a set of 77 images was annotated with bounding boxes of heraldic symbols and their class labels comprising a total of 728 symbols and 970 text boxes, with between 1 and 16 symbol boxes and between 1 and 19 text boxes for each image. An annotated version of an image can be seen in the following figure, with purple boxes showing symbols and cyan boxes indicating text.



For the image pre processing, two transformations were made: As the images of the dataset are photographed pages from a book, a noticeable variance in brightness between different images is expected. To prevent the model from overfitting the training set too much in regards of image brightness, the training and validation images were augmented with a random brightness contrast transformation. In addition to that, another transformation was performed afterwards, with the goal of resizing the images to a uniform size, as it is required by the chosen architecture. The resolution of the resized images was selected to be 1024x1024, as it is a decent compromise between retaining the information of the original images, which were around 3000x2000 pixel in size, and reducing the required GRAM for storing them.

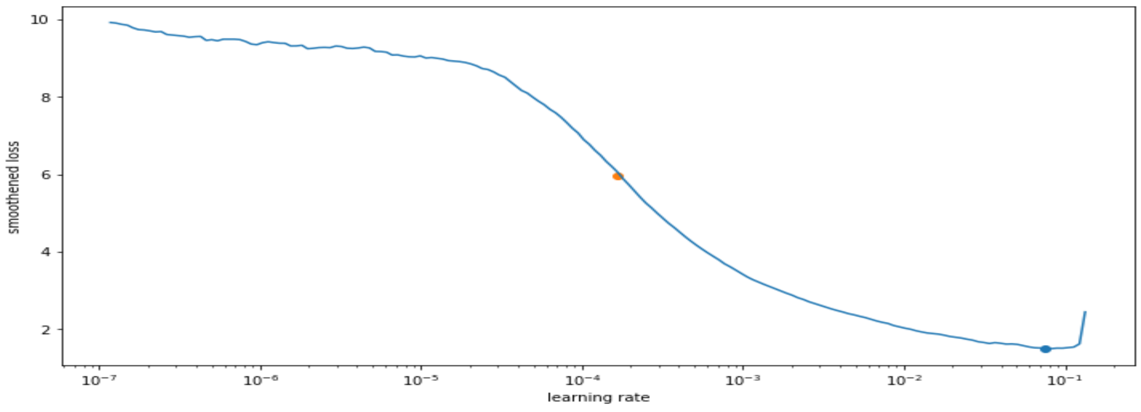
There were two different approaches tried for fine-tuning the pre-trained faster-rcnn model: The first approach was to tune the model for recognizing bounding boxes of symbols and text separately and deriving a merged bounding box afterwards. The second approach was to train the model to

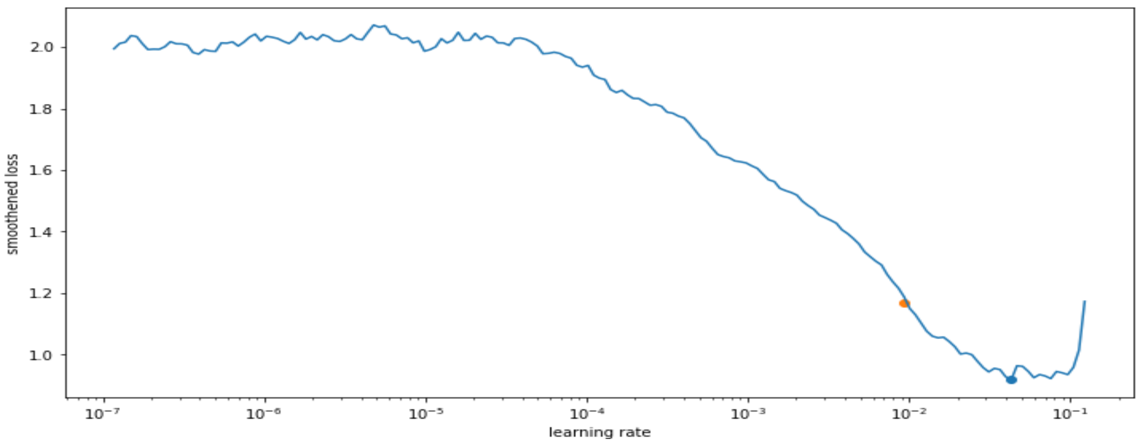
find the already merged bounding boxes of symbols and associated text.
As a result, two different datasets were created. The first containing annotated images with separated text and symbol boxes, the second one containing images, annotated with merged bounding boxes. For that purpose a function for merging bounding boxes was implemented.
The function iterates all annotated bounding boxes and attempts to find the closest text box within a specific distance threshold for each symbol box. If a suitable textbox is discovered, the symbol box and the text box are merged by calculating a bounding box that includes both boxes.

For both approaches, the training/validation split was done by selecting a random set of images for validation. The validation set included approximately 20 percent of the total annotated images with the rest of the images comprising the training set.

For each training method, an instance of a pre-trained faster-rcnn model was created, with a slightly adjusted head comprising two fully connected linear layers that output bounding boxes and class probabilities for three (text, symbol, background) and two (mergedBB, background) classes respectively. For the optimization of model parameters, a simple stochastic gradient descent is used as it outperformed typical optimization algorithms like adam during some test runs. For the learning rate scheduler, a cosine annealing approach with warm restarts was chosen as it can help in preventing the model from getting stuck at local minima. The initial interval was selected at an epoch length of one, with a multiplication factor of two.

To find a suitable initial learning rate, a separate training session was executed. This training session monitored the development of the loss values after each mini batch. In addition, the learning rate got multiplied by a factor between one and two after each mini batch. This lead to the following loss vs learning rate plots (first separated approach, second merged approach):



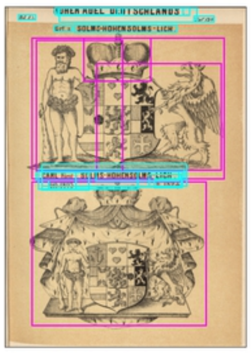


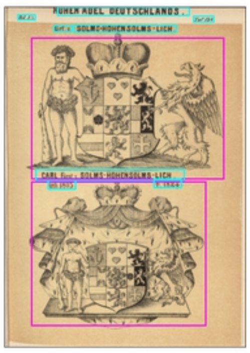
The yellow dots indicate the learning rate for which the steepest descent in loss was detected, the blue dot shows the learning rate for which the minimum loss was detected. This lead to the selection of a learning rate of around 6*10^-4 for the first model and around 4*10^-3 for the second one.

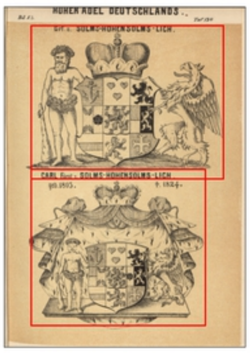
To prevent under and overfitting, the number of epochs for training was not fixed to a set value. Instead, after every epoch, the mean average precision (MAP) for the validation set was logged. If the MAP did not improve after ten epochs or decreased by more than ten percent, compared to the best result, the training was stopped.

For separated bounding boxes, the model has achieved the best results after 48 epochs of training. For merged bounding boxes, the best result on the validation set was achieved after 45 epochs.

For the model, trained to detect merged bounding boxes, the distinct symbols from an unknown image can be recovered by scaling the image size to 1024x1024 pixels, performing inference on the model and scaling the predicted bounding boxes back to the original image size. For the model that detects text and symbol boxes separately, some additional post processing steps need to be performed after inference: Firstly, after the model prediction is recovered, candidate bounding boxes are selected based on a confidence threshold. Secondly, those candidate bounding boxes are inspected for possible duplicate boxes, that refer to the same object as another candidate box. If such a conflict is detected, the box, which the model is less confident in, gets dropped. Finally, after the elimination of duplicate boxes, symbol and text boxes are merged. This process is shown in the following image:

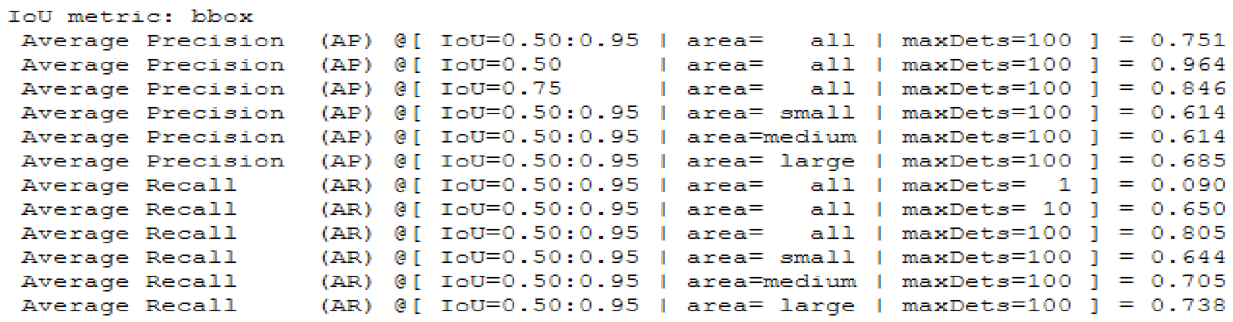






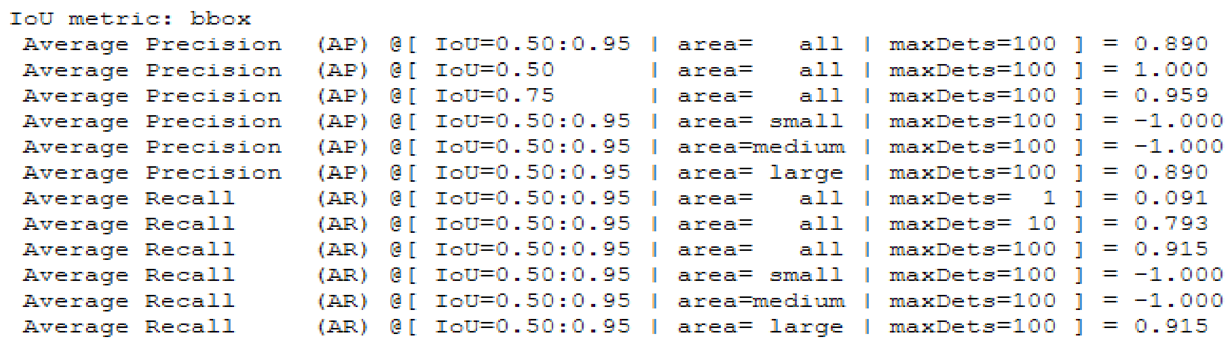
Evaluation

The performance of both trained models on the validation sets is shown in the following images:
Separated:



```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.751
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.964
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.846
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.614
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.614
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.685
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.090
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.805
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.644
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.705
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.738
```

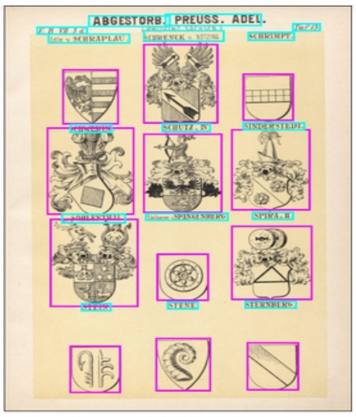
Merged:

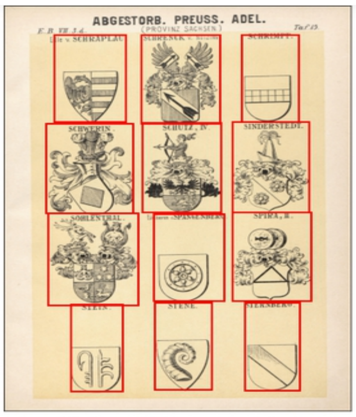


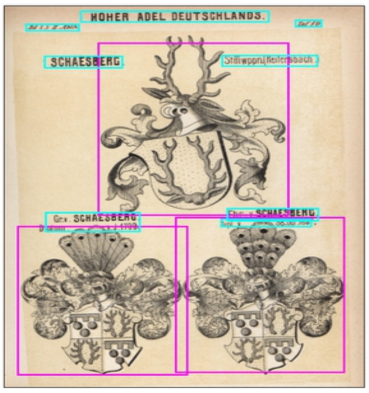
```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.890
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.959
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.890
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.091
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.793
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.915
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.915
```

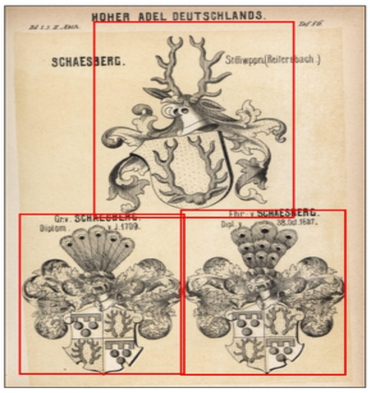
The higher scores in precision and recall for the merged model are expected as the task is reduced to only regressing bounding boxes, instead of a box regression and a classification task for the

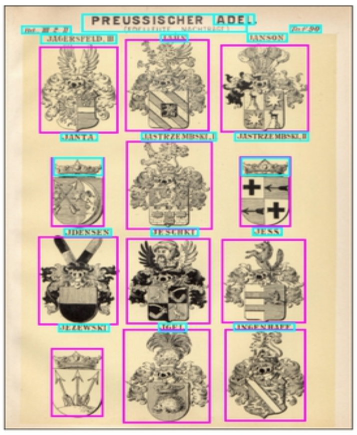
separated model.
In addition to the precision and recall metrics for the validation set, the performance of the models can also be evaluated by looking at their predictions for some previously unknown test images. The following images show some good and bad perfomacnes of the trained models, with the separated models predictions depicted on the left and the merged models predictions depicted on the right side.

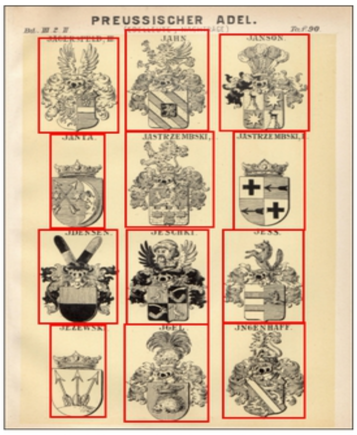


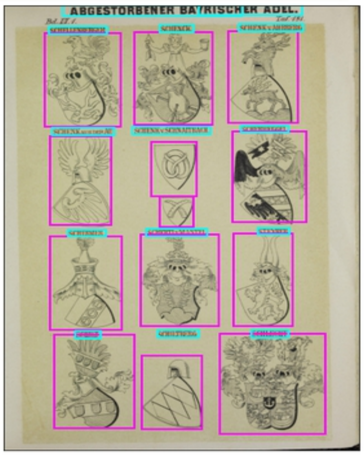


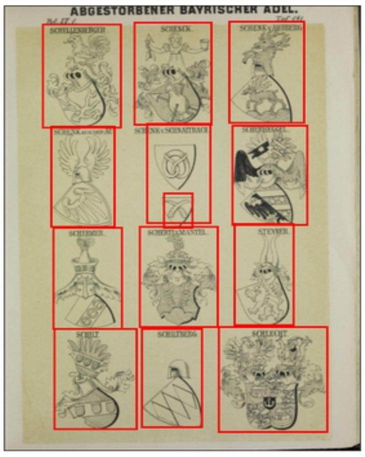












These test images show, that the separated model generally does a good job in recognizing symbols and text boxes, but sometimes fails to seperate neighbouring text boxes or creates multiple small text boxes for each word.
Additionally, sometimes crowns get mistakenly detected as text.
For the detection of heraldic symbols, the separated model does perform better, than for the recognition of text. The only reoccurring issue in symbol recovery is the failure to fully recognize symbols that are divided into multiple smaller parts.
The merged bounding box model does find a suitable merged bounding box for most symbol text combinations. One major issue of this model is its high weight on the bounding boxes of symbols over the bounding boxes of text. This might be due to the fact that usually, the symbol makes up the larger part of the merged bounding box. As a result, only predicting the bounding box of the symbol and disregarding the text can lead to good estimations for merged bounding boxes in most cases.This can cause the model to disregard text that is horizontally larger than the symbol.