

Improving web user interface element detection using Faster R-CNN

Jiří Vyskočil¹, Lukáš Pícek¹

¹Department of Cybernetics, Faculty of Applied Sciences, University of West Bohemia

Abstract

Several challenges may arise when designing new user interfaces (UIs), e.g., because of communication between designers and developers, to which the detection of UI elements can help. The ImageCLEF DrawnUI 2021 challenge builds on the detection of such elements in two contest tasks: a Screenshot task that contains the website screenshot images with lots of noisy data, and a Wireframe task for detecting UI elements from hand-drawn proposals. This paper describes a simple algorithm based on the edge detection to filter noisy data from the website screenshots, and machine learning method which scored the first place in both tasks while having 0.628 and 0.900 mAP at 0.5 IoU in the Screenshot and Wireframe tasks. This method is based on the Faster R-CNN with a Feature Pyramid Network (FPN) that uses selected aspect ratios of anchor boxes according to the occurrences from the datasets. The code is available at <https://github.com/vyskocj/ImageCLEFdrawnUI2021>

Keywords

Object Detection, Machine Learning, Edge Detection, Faster R-CNN, FPN, CNN, User Interface

1. Introduction

The ImageCLEF DrawnUI challenge [1] was organized as part of the ImageCLEF 2021 workshop [2] at the CLEF conference. The main goal for the two proposed tasks - *Screenshots & Wireframes* - was to create a system capable of automatic detection and recognition of individual user interface (UI) elements on given images. The *Screenshot task* focused on the website screenshot images, and the *Wireframe task* targeted on hand-drawn UI drawings. The motivation for both tasks is to simplify and speed up the Web development process by giving the designers a tool that can visualize the website immediately based on their hand-drawn sketches.

The machine learning techniques have already been applied to the hand-drawn UI elements detection in the last years. Gupta et al. [3] used Mask R-CNN [4] and Multi-Pass Inference technique to boost the viability of the model by passing the input image (without the already detected objects) to the model several times. Narayanan et al. [5] explored Cascade R-CNN [6] and YOLOv4 [7] architectures, and Zita et al. [8] used regular Faster R-CNN [9] architecture and advanced regularization techniques for training the model. In this work, we utilize the Faster R-CNN extended by the Feature Pyramid Network (FPN) [10] that builds high-level semantic feature maps at all selected scales and makes the predictions more accurate. The models were implemented and fine-tuned using the Detectron2 API [11] from publicly available checkpoints


CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

✉ vyskocj@kky.zcu.cz (J. Vyskočil); picekl@ntis.zcu.cz (L. Pícek)

🆔 0000-0002-6443-2051 (J. Vyskočil); 0000-0002-6041-9722 (L. Pícek)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

pre-trained on the COCO dataset [12]. Additionally, we improved the performance by using various augmentations, i.e., Relative Random Resize, Cutout [13], brightness and contrast adjustment, and by selecting bounding box proposals. In case of the Screenshot task, we utilized the use of data filtering algorithm based on the edge detection [14, 15]. The improvements of our method, which won in both contest tasks, are shown by comparing with the others in the benchmark of the DrawnUI challenge.

Besides, we experimented with novel methods [16, 17] based on the Detection Transformer. These approaches remove the need for hand-designed components, e.g., non-maxima suppression (NMS), but requires much more training time to convergence than previous detectors. Given this training issue of the Detection Transformer, we decided to keep the NMS in our model. Using the Transformers on the provided data in the contest tasks led to significantly worse detection performance even with $7.5\times$ more training steps.

2. Challenge datasets

Wireframe task. Provided dataset is a combination of 4,291 hand-drawn high-resolution image templates. The data is divided into 3,218 images for the development and 1,073 images for the testing. For each image in the development set, we have manual annotations with the bounding boxes and their corresponding labels from pre-defined 21 classes. The development set includes all images from last year’s challenge and additional images to re-balance the class distribution. As there is no official training/validation split provided, we did a random 85%/15% split. The detailed statistics covering the class distribution, dataset split, and absolute/relative box number are presented in Table 1.

Screenshot task. In the Screenshot task, the provided dataset includes 9,630 full-page screenshots of websites in several languages. The data comes with labeled bounding boxes of the UI elements. A total of 6 classes is defined, the distribution of ground truth boxes can be found in Table 2. The development set contains 6,840 training images, and 930 manually annotated validation images. The training set includes noisy data: blank images and bounding boxes with shifted positions. The testing set contained a total of 1,860 samples.

3. Methodology

In this section, we cover the noise data filtering algorithm and training the Faster R-CNN [9] detection network based on the ResNet-50 [18] backbone. We also use the FPN [10] extractor to combine semantically strong features thanks to a top-down pathway and lateral connections from the same spatial size. We use SGD optimizer with momentum of 0.9 [19], learning rate warm up of the first epoch reached the value of 0.0025 and a smooth L_1 [20] is applied for regression loss. The detector is implemented and fine-tuned in the Detectron2 API [11] from publicly available pre-trained weights on the COCO dataset [12]. For more details about the hyperparameter settings, see Table 3, and advanced augmentations are listed in Table 4. All experiments are evaluated at mean average precision (mAP) and mean average recall (mAR) with Intersection over Union (IoU) in range of 0.5 to 0.95 with the increment of 0.05, and mean average precision with IoU greater than 0.5 (denoted as mAP0.5).

Table 1
Distribution of training and validation set categories of the Wireframe task.

Wireframe task				
Category	Training set		Validation set	
	# of boxes	fraction [%]	# of boxes	fraction [%]
button	21,787	19.33	3,657	17.80
label	17,348	15.39	3,462	16.85
paragraph	13,884	12.32	2,474	12.04
image	10,328	9.16	1,752	8.53
link	6,359	5.64	1,133	5.52
linebreak	6,208	5.51	1,118	5.44
container	5,425	4.81	953	4.64
header	4,356	3.86	739	3.60
textinput	4,192	3.72	825	4.02
checkbox	3,426	3.04	677	3.30
radiobutton	3,302	2.93	719	3.50
toggle	2,785	2.47	574	2.79
slider	2,668	2.37	524	2.55
datepicker	2,606	2.31	473	2.30
textarea	2,449	2.17	452	2.20
rating	2,372	2.10	437	2.13
dropdown	1,453	1.29	250	1.22
video	810	0.72	155	0.75
list	788	0.70	125	0.61
stepperinput	137	0.12	22	0.11
table	55	0.05	19	0.09

Table 2
Distribution of training and validation set categories of the Screenshot task.

Screenshot task				
Category	Training set		Validation set	
	# of boxes	fraction [%]	# of boxes	fraction [%]
link	106,457	36.72	10,910	30.46
text	82,642	28.51	9,400	26.24
image	52,672	18.17	7,462	20.83
heading	39,330	13.57	5,620	15.69
input	4,448	1.53	605	1.69
button	4,353	1.50	1,823	5.09

3.1. Baseline experiment

For the baseline experiment, the Random Relative Resize augmentation is applied to resize an image to 70-90% of its size and crop it to a maximum of 1,400 px to limit the memory usage. The resize augmentations are deeply examined in Section 3.3.1. The hyperparameters settings are described in Table 3 and advanced augmentations in Table 4. In the Screenshot task, the baseline model reached **0.592** mAP0.5, **0.404** mAP, and **0.603** mAR, while in the Wireframe task, a model with the same settings have **0.969** mAP0.5, **0.703** mAP, and **0.763** mAR.

Table 3

Base parameters for training the models.

Parameter	Value	Parameter	Value
Checkpoint	COCO	Batch size	1
Optimizer	SGD w/ moment. of 0.9	Accumulated grad.	4
Loss	smooth L ₁	Epochs	20 (Screenshot)
Base and min lr	0.0025 - 0.000625		40 (Wireframe)
Decay factor	0.5	Decay in ... epoch	[10, 15] (Screenshot)
Warm up	1 epoch		[20, 30] (Wireframe)

Table 4

Base augmentations for training the models.

Augmentation	Intensity	Probability [%]
Random Brightness	0.5 - 1.5	50
Random Contrast	0.5 - 1.5	50
Random Saturation (RGB only)	0.5 - 1.5	50

3.2. Filtering noisy data in the Screenshot task

Even though the noisy data can be effective for the training [21], we decided to analyze filtering of blank images and wrongly annotated bounding boxes from the Screenshot task dataset. The aim is to remove images or ground truth boxes that contain constant color intensity. For this reason, the data filtering (shown in Algorithm 1) is based on an edge detector [14, 15] to be independent of the intensity of the pixels in the input image.

Algorithm 1 Filtering homogeneous image elements from a dataset.

```

Define threshold values  $T_{img}$  and  $T_{box}$ 
for each image do
  Apply edge detector to the image and compute a mean value  $\mu_{img}$  from the output
  if  $\mu_{img} \leq T_{img}$  then
    Discard this image from the set and continue with the next one
  else
    for each bounding box of the image do
      Apply the  $T_{box}$  threshold in the same way as  $T_{img}$  in the image
    end for
    if all bounding boxes are discarded from the annotations of the image then
      Discard this image from the set and continue with the next one
    end if
  end if
end for

```

To verify the efficiency of data filtering, we manually selected appropriate thresholds for images (see Table 5) and a set of fixed thresholds from 0.2 to 1.8 for bounding boxes. Then

we trained the network with filtered annotations in the Screenshot dataset using the same settings as in Section 3.1. For the results of this experiment, see Table 6. One can observe that filtering the homogeneous images increases the mAP0.5 by **0.012**, mAP by **0.008**, and mAR by **0.009** compared to the case of the original set. Filtering homogeneous images and bounding boxes also increases the detection performance but it detects less precisely in all tested cases of bounding box thresholds than filtering only the images. This behaviour can be caused by eliminating the training data, which is in fact an object, not noise. Therefore, we determined **a new baseline for the Screenshot task** by filtering only the images from the training set (this model was submitted as a baseline for the Screenshot task of DrawnUI challenge [1]).

Table 5

Manually designed thresholds for discarding images from a training set of the Screenshot task.

Img size	$\leq 500 \times 500$	$\leq 600 \times 600$	$\leq 900 \times 900$	$\leq 1200 \times 1200$	$\geq 1200 \times 1200$
Threshold	2.5	3.5	2.8	2.5	0.8

Table 6

Results on the validation set while discarding noisy images (for threshold values see Table 5) and bounding boxes (BBoxes) from the training set of the Screenshot task.

Threshold		mAP0.5	mAP	mAR
Image	BBox			
-	-	0.592	0.404	0.603
Tab. 5	-	0.604	0.412	0.612
Tab. 5	0.2	0.599	0.409	0.610
Tab. 5	0.6	0.599	0.409	0.610
Tab. 5	1.0	0.601	0.408	0.610
Tab. 5	1.4	0.601	0.411	0.612
Tab. 5	1.8	0.593	0.397	0.606

3.3. Augmentations

Image resizing, Cutout [13] augmentation, and color spaces are tested to improve detection performance. The improvement is evaluated as a comparison with baseline models defined in the previous sections, i.e., Section 3.1 is relevant for the Wireframe task and for the Screenshot task, a new baseline was established in the Section 3.2.

3.3.1. Image resize

The basic approach to dealing with various sizes of the input images is to resize it to the desired constant value so that the original aspect ratio is kept. However, various sizes can help the learning algorithm to detect objects at different stages of the network. For example, imagine that we only have small boxes available for the category button in the training set. In the test stage, this network will not expect a large button at the input and will most likely fail to detect

it. In order to use different input image sizes, the backbone network must not contain any fully connected layer.

Two types of resizing images are compared in Table 7. The first one, Resize Shortest Edge (default for Detectron2 [11] software) has a defined set of shortest edge lengths of the image from 640 to 800 px with the increment of 32, which are selected randomly during the training. If the longer edge is larger than 1,333 px, the shorter edge is underscaled so that the longer edge does not exceed this maximum size. We proposed the second type of resizing as a Random Relative Resize. It defines *an interval* for which the image is randomly resized, and *a maximum length* of the edges for cropping image during the training due to memory requirements. The particular aim of this augmentation is to keep the small boxes so that they do not disappear when the image size is reduced, and the network is able to detect them. In the test stage, the image is resized only by the middle value of the specified interval and no image cropping is applied. This augmentation proved to be most suitable for an image enlarging by a random value in the interval [0.6, 1.0] for both tasks, where mAP0.5 and mAP metrics are roughly ranging from **0.034** to **0.045** higher than when using the Resize Shortest Edge.

Table 7

Comparison of two types of resizing augmentation on the validation set for the Screenshot and Wireframe tasks. Resize Shortest Edge (RSE) selects sizes from 640 to 800 px with a step of 32. Random Relative Resize (RRR) defines an interval of [min, max] relative sizes as factor for resizing the image.

Resize type	Screenshot task			Wireframe task		
	mAP0.5	mAP	mAR	mAP0.5	mAP	mAR
RSE	0.563	0.372	0.549	0.929	0.672	0.732
RRR [0.7, 0.9]	0.604	0.412	0.612	0.969	0.703	0.763
RRR [0.6, 1.0]	0.608	0.417	0.619	0.972	0.706	0.765
RRR [0.4, 1.0]	0.608	0.414	0.608	0.954	0.689	0.751

3.3.2. Cutout augmentation

To increase the performance of the network, in addition to brightness and resize augmentations, we also used Cutout [13] from Albumentations library [22] to randomly cuts boxes (denoted also as holes) from the image. This augmentation expects the number of maximum holes and their maximum spatial size as the input. In our experiments, we define the maximum size of holes in the percentage of the image. The results (see Table 8) show that it can increase mAP0.5 by **0.008**, and mAP by **0.004** for the Screenshot task when using 4 holes with max size of 5% of the image, while in the Wireframe task, the detection performance was slightly reduced in all settings of the Cutout augmentation. Even so, we applied this augmentation in our further research (see Section 3.4 and Section 4) to keep the experiments for both contest tasks comparable, and in the Screenshot task, the augmentation shows meaningful improvements.

Table 8

Comparison of using Cutout augmentation on the validation set for the Screenshot and Wireframe tasks. The max size is given as a percentage of the image size.

		Screenshot task			Wireframe task		
max holes	max size	mAP0.5	mAP	mAR	mAP0.5	mAP	mAR
-	-	0.604	0.412	0.612	0.969	0.703	0.763
4	5.0%	0.612	0.416	0.610	0.967	0.699	0.759
8	5.0%	0.603	0.411	0.606	0.968	0.701	0.760
16	5.0%	0.596	0.405	0.608	0.967	0.698	0.759
16	2.5%	0.603	0.410	0.607	0.964	0.697	0.758

3.3.3. Color space

In the next step, converting images to the greyscale, such as in the previous works of this challenge [8, 3], is applied. It results in no improvement against the RGB images (see Table 9) in the Screenshot task. On the other hand, for the Wireframe task, converting the data to grayscale yields up to approximately **0.005** greater mAP0.5, mAP, and mAR. Therefore, both RGB and grayscale images are used for the remaining experiments.

Table 9

Comparison of using RGB and greyscale images on the validation set for the Screenshot and Wireframe tasks.

	Screenshot task			Wireframe task		
Color space	mAP0.5	mAP	mAR	mAP0.5	mAP	mAR
RGB	0.604	0.412	0.612	0.969	0.703	0.763
Greyscale	0.593	0.403	0.604	0.974	0.707	0.767

3.4. Anchor box proposals

We followed up on previous experiments that examines augmentations (see Section 3.3) and we trained new models (parameters for new augmentations are summarized in Table 10). After that we analyzed which aspect ratios of ground truth boxes are included in the datasets. Occurrence of such aspect ratios are visualized in Figure 1 for both the Screenshot and the Wireframe tasks. One can observe that the horizontal boxes are far more frequent than the vertical ones. As a result, the appropriate aspect ratios were selected to generate the box proposals. We added one horizontal aspect ratio of 0.2 to the default ones (i.e., to the set of 0.5, 1, and 2). Then we selected aspect ratios of 0.1, 0.5, 1, and 1.5 according to the distribution from the Figure 1. Eventually, we also reduced the size of the anchors $2\times$ for each output layer from the Feature Pyramid Network [10] of ResNet [18] backbone, i.e., reduced size for semantic feature maps from P_2 to P_6 , see Table 11 for a summary of these settings.

An experiment examining the use of different aspect ratios for anchor box proposals (see Table 12) shows that selecting aspect ratios by frequency in the dataset increases detection

Table 10

Additional augmentations to the training the models (see Table 3 and Table 4 for base parameters and common augmentations used for training).

Augmentation	Parameters	
Random Relative Resize	resize interval = [0.6, 1.0]	crop = 1400 px
Cutout	max holes = 4	max size = 5% of image

performance in most cases. Only for the Wireframe task with RGB images, the default aspect ratios achieved slightly greater mAP and mAR than the ones selected from statistics. The value of mAP0.5 is greater for aspect ratios selected according to statistics with smaller anchor sizes, and this setting proved to be better performing for greyscale images roughly ranging from **0.002** to **0.003** for all measured metrics. Therefore, we selected this setting for comparison with the other backbone models in the Wireframe task. For the Screenshot task, the same aspect ratios were selected but with default sizes of anchor boxes, because it performed better with mAP0.5 and mAP up to **0.006** than the default anchor settings.

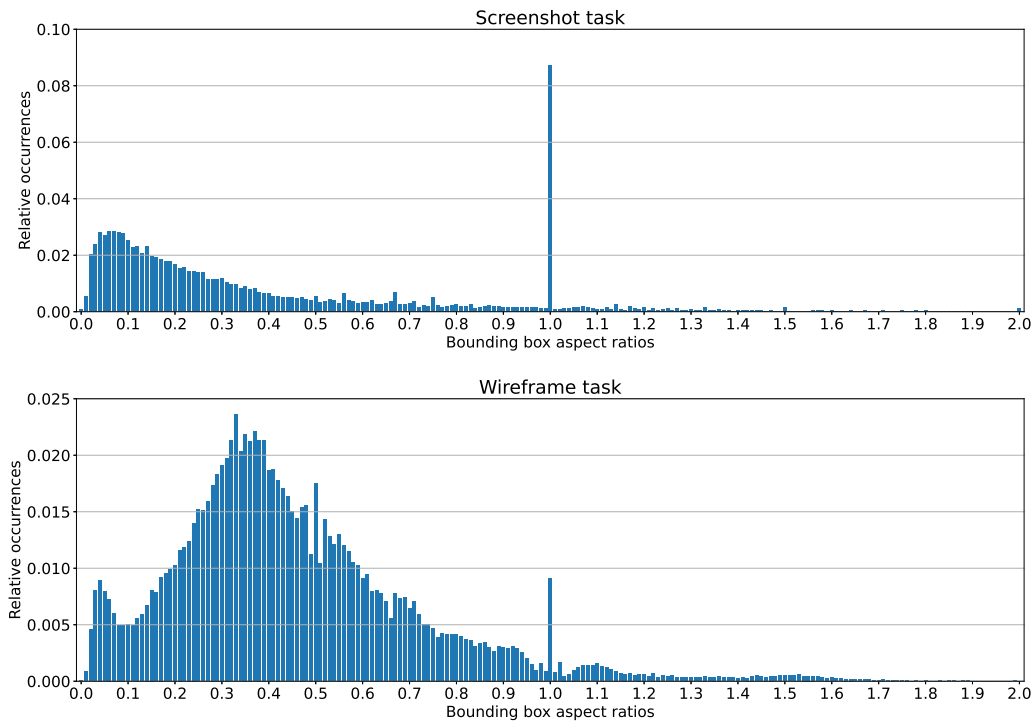


Figure 1: Occurrence of the aspect ratios of the bounding boxes in the Screenshot and Wireframe task datasets.

Table 11

Settings of the selected aspect ratios and sizes for anchor box proposals.

#	Anchor generator settings	Aspect ratios	Anchor sizes
1	default	[0.5, 1.0, 2.0]	[32, 64, 128, 256, 512]
2	default + horizontal	[0.2, 0.5, 1.0, 2.0]	[32, 64, 128, 256, 512]
3	statistical	[0.1, 0.5, 1.0, 1.5]	[32, 64, 128, 256, 512]
4	statistical + smaller sizes	[0.1, 0.5, 1.0, 1.5]	[16, 32, 64, 128, 256]

Table 12

Comparison of using different settings of aspect ratios and sizes for anchor box proposals on the validation set. Comparison is performed on the RGB and greyscale images for the Screenshot and Wireframe tasks. Anchor settings are specified in Table 11.

		Screenshot task			Wireframe task		
Anchor setting	Color space	mAP0.5	mAP	mAR	mAP0.5	mAP	mAR
#1	RGB	0.617	0.421	0.615	0.973	0.705	0.765
#2	RGB	0.621	0.424	0.625	0.974	0.704	0.762
#3	RGB	0.623	0.426	0.627	0.970	0.704	0.763
#4	RGB	0.611	0.414	0.627	0.975	0.700	0.762
#1	Greyscale	0.612	0.417	0.613	0.972	0.702	0.762
#2	Greyscale	0.610	0.416	0.617	0.974	0.703	0.762
#3	Greyscale	0.612	0.419	0.620	0.970	0.703	0.762
#4	Greyscale	0.606	0.410	0.622	0.975	0.704	0.764

4. Backbones comparison

As a last step, several backbone architectures were compared for UI element detection on greyscale and RGB images. We used base parameters and augmentations from Table 3 and Table 4, and additional augmentations described in Table 10. In the Wireframe task, *statistical + smaller sizes* variant of the anchor generator was used, and the *statistical* variant was used for the Screenshot task (for these settings of anchor box proposals see Table 11).

In the comparison of the backbone architectures (see Table 13), the reader can recognise that only for the Wireframe task, the most complex compared architecture achieved better performance for measured metrics in both cases of selected color space. Although we expected better performance with the more complex ResNeXt-101 backbone, superior results were achieved with the ResNet-50 in the Screenshot task. The model with a complex backbone converges slower than ResNet-50, hence more epochs should be ran for better results.

5. Submissions

In the DrawnUI challenge [1], we have created up to 9 submissions using the configuration listed bellow. The configuration is the same for both the Screenshot and the Wireframe tasks, any additional configurations relevant for any of the tasks are also specified. Results on the test

Table 13

Comparison of different backbone architectures on the validation set. Comparison is performed on the RGB and greyscale images for the Screenshot and Wireframe tasks.

		Screenshot task			Wireframe task		
Backbone	Color space	mAP0.5	mAP	mAR	mAP0.5	mAP	mAR
ResNet-50	RGB	0.623	0.426	0.627	0.975	0.700	0.762
ResNet-101	RGB	0.603	0.410	0.614	0.970	0.699	0.758
ResNeXt-101	RGB	0.601	0.408	0.608	0.977	0.705	0.765
ResNet-50	Greyscale	0.612	0.419	0.620	0.975	0.704	0.764
ResNet-101	Greyscale	0.604	0.413	0.612	0.970	0.699	0.760
ResNeXt-101	Greyscale	0.598	0.408	0.605	0.976	0.706	0.766

set can be found in Table 14:

#1: ResNet-50 (baseline, RGB) - model trained according to Table 3 and Table 4 w/ the Random Relative Resize augmentation using image resize interval [0.7, 0.9]. In the Screenshot task, only the images were filtered using thresholds described in Table 5.

#2: ResNet-50 (augmentations, RGB) - baseline trained w/ augmentations from Table 10.

#3: ResNet-50 (anchor settings, RGB) - same as submission #2 w/ anchor settings from Table 11: *statistical* for the Screenshot task, and *statistical + smaller sizes* for the Wireframe task.

#4: ResNet-50 (anchor settings, greyscale) - same as submission #3 but w/ greyscale images.

#5: ResNet-50 (train+val, RGB) - same as submission #3 but trained on the whole development set (w/o any validation data).

#6: ResNeXt-101 (RGB) - trained w/ the same settings as submission #3.

#7: ResNet-50 (train+val, RGB, 2× epochs) - submission #5 trained for 2× more epochs.

#8: ResNet-50 (train+val, greyscale) - same as submission #5 but trained w/ greyscale images.

#9: ResNeXt-101 (RGB, train+val, +5 epochs) - submission #6 fine-tuned w/ 5 more epochs on whole development set (w/o any validation data).

Table 14

Test results obtained from the submissions.

		Screenshot task		Wireframe task		
#	Run ID	mAP0.5	mAR0.5	Run ID	mAP0.5	mAR0.5
1	134207	0.594	0.815	134095	0.794	0.832
2	134214	0.602	0.822	134175	0.830	0.863
3	134215	0.609	0.834	134180	0.882	0.918
4	134217	0.601	0.827	134181	0.889	0.923
5	134224	0.628	0.830	134225	0.888	0.925
6	134603	0.590	0.807	134548	0.900	0.934
7	134716	0.621	0.821	134723	0.894	0.928
8	-	-	-	134728	0.895	0.927
9	-	-	-	134829	0.900	0.933

6. Conclusion

Our method, including data filtering, Cutout augmentation and statistical aspect ratios for anchor box proposals, ended in the first place in both contest tasks of DrawnUI challenge: *Screenshot task* - ResNet-50 backbone trained on whole development set with 0.628 mAP at 0.5 IoU on the test set, and *Wireframe task* - ResNeXt-101 backbone trained with split development set for training and validation, this model achieved 0.900 mAP at 0.5 IoU on the test set. Besides, we explored the State-of-the-Art object detectors based on the transformers, such as a DETR. The DETR did not achieve satisfactory results even after 300 epochs compared with the Faster R-CNN trained up to 40 epochs. Due to time constraints, we will consider the use of transformer in the upcoming research projects.

Acknowledgments

The work has been supported by the grant of the University of West Bohemia, project No. SGS-2019-027. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References

- [1] R. Berari, A. Tauteanu, D. Fichou, P. Brie, M. Dogariu, L. D. Ștefan, M. G. Constantin, B. Ionescu, Overview of ImageCLEFdrawnUI 2021: The Detection and Recognition of Hand Drawn and Digital Website UIs Task, in: CLEF2021 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org <<http://ceur-ws.org>>, Bucharest, Romania, 2021.
- [2] B. Ionescu, H. Müller, R. Péteri, A. Ben Abacha, M. Sarrouti, D. Demner-Fushman, S. A. Hasan, S. Kozlovski, V. Liauchuk, Y. Dicente, V. Kovalev, O. Pelka, A. G. S. de Herrera, J. Jacutprakart, C. M. Friedrich, R. Berari, A. Tauteanu, D. Fichou, P. Brie, M. Dogariu, L. D. Ștefan, M. G. Constantin, J. Chamberlain, A. Campello, A. Clark, T. A. Oliver, H. Moustahfid, A. Popescu, J. Deshayes-Chossart, Overview of the ImageCLEF 2021: Multimedia retrieval in medical, nature, internet and social media applications, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction, Proceedings of the 12th International Conference of the CLEF Association (CLEF 2021), LNCS Lecture Notes in Computer Science, Springer, Bucharest, Romania, 2021.
- [3] P. Gupta, S. Mohapatra, Html atomic ui elements extraction from hand-drawn website images using mask-rcnn and novel multi-pass inference technique, in: CLEF2020 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org <<http://ceur-ws.org>>, Thessaloniki, Greece, 2020.
- [4] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.
- [5] N. Narayanan, N. N. A. Balaji, K. Jaganathan, Deep learning for ui element detection: Drawnui 2020, in: CLEF2020 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org <<http://ceur-ws.org>>, Thessaloniki, Greece, 2020.

- [6] Z. Cai, N. Vasconcelos, Cascade r-cnn: High quality object detection and instance segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (2021) 1483–1498. doi:10.1109/TPAMI.2019.2956516.
- [7] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, *arXiv preprint arXiv:2004.10934* (2020).
- [8] A. Zita, L. Picek, A. Říha, Sketch2code: Automatic hand-drawn ui elements detection with faster r-cnn, in: *CLEF2020 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org* <<http://ceur-ws.org>>, Thessaloniki, Greece, 2020.
- [9] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), *Advances in Neural Information Processing Systems* 28, Curran Associates, Inc., 2015, pp. 91–99.
- [10] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944. doi:10.1109/CVPR.2017.106.
- [11] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, R. Girshick, Detectron2, <https://github.com/facebookresearch/detectron2>, 2019.
- [12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: *European conference on computer vision*, Springer, 2014, pp. 740–755.
- [13] T. DeVries, G. W. Taylor, Improved regularization of convolutional neural networks with cutout, *arXiv preprint arXiv:1708.04552* (2017).
- [14] X. Wang, Laplacian operator-based edge detectors, *IEEE transactions on pattern analysis and machine intelligence* 29 (2007) 886–890. doi:10.1109/TPAMI.2007.1027.
- [15] D. Ziou, S. Tabbone, et al., Edge detection techniques-an overview, *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii* 8 (1998) 537–559.
- [16] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: *European Conference on Computer Vision*, Springer, 2020, pp. 213–229.
- [17] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, J. Dai, Deformable detr: Deformable transformers for end-to-end object detection, *arXiv preprint arXiv:2010.04159* (2020).
- [18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.
- [19] N. Qian, On the momentum term in gradient descent learning algorithms, *Neural networks* 12 (1999) 145–151.
- [20] R. Girshick, Fast r-cnn, in: *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448. doi:10.1109/ICCV.2015.169.
- [21] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, L. Fei-Fei, The unreasonable effectiveness of noisy data for fine-grained recognition, in: *European Conference on Computer Vision*, Springer, 2016, pp. 301–320.
- [22] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, A. A. Kalinin, Alumentations: Fast and flexible image augmentations, *Information* 11 (2020). URL: <https://www.mdpi.com/2078-2489/11/2/125>. doi:10.3390/info11020125.