Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 516 – 532
© TÜBİTAK
doi:10.55730/1300-0632.3999

UIBee: An improved deep instance segmentation and classification of UI elements in wireframes

Cahit Berkay KAZANGİRLER[1], **Caner ÖZCAN**[2,*], **Buse Yaren TEKİN**[3]
[1]Department of Computer Engineering, Karabük University, Karabük, Turkey
[2]Department of Software Engineering, Karabük University, Karabük, Turkey
[3]Department of Computer Technologies, Kastamonu University, Kastamonu, Turkey

Received: 05.08.2022 • **Accepted/Published Online:** 06.02.2023 • **Final Version:** 28.05.2023

Abstract: User Interface (UI) is a basic concept in which individuals interact with any computer program or technological device to create a graphical design. In the initial stages of app development, UI prototype is a must. An automatic analysis system for the basic execution of UI designs will considerably speed up the development of designs according to old-fashioned methods. In this approach, it is aimed at saving cost and time by automating the process. For the aforesaid objective, we present a new approach rather than the traditional methods. For this reason, a high amount of elements in wireframes are detected and segmented. Furthermore, with the state-of-the-art methods, one of the machine learning classifiers is expected to give lower performance than deep learning for comparison purposes. In this study, the detection and segmentation of elements, which is the first stage which will eliminate time loss, redundant time, cost, and labor in the communication between designers and front-end developers. To test the classification task of the Mask R-CNN, was designed using transfer learning supported neural networks to compare with other algorithms. As a result, the precision reached 93.15% and the mAP (@IOU > 0.5) reached 96.50%. Then, we improved the algorithm by replacing the convolution blocks in the graphs, adding them, and changing the input units, and the accuracy increased to 98.49%.

Key words: Wireframe, user interface, user experience, object detection, deep neural networks

1. Introduction

Project development refers to all the work in the process from the idea step to the implementation stage. Software projects are canceled before they are completed, as they are in the real world, due to budget, time, and labor issues, or they are not used at all because they do not meet the requirements. Software technology is one of the fastest growing sectors in developed countries [1]. Therefore, the number of software companies or companies that have software departments continues to increase. According to the 2020 CHAOS report of Standish Group International, which publishes comprehensive analysis reports on software projects, the projects were only 23% higher than the skilled level [2]. According to the Comprehensive Human Appraisal for Originating Software (CHAOS) research, as shown in Table 1, 19% revealed failure, while 58% showed a hard challenge.

Projects that exceed the budget, are delivered late and are delivered with less than the initially determined features are considered to be inefficient. According to the report, only 19% of projects were successful. There are projects that are terminated without being completed after starting the project or that are never used, including if they are finished, and they are considered inefficacious.

*Correspondence: canerozcan@karabuk.edu.tr

Table 1. Table of values showing rising project success rates according to the 2020 CHAOS report.

Skilled level	Highly	Skilled	Moderately	Poor
Successful	23%	33%	39%	58%
Challenged	58%	55%	38%	33%
Failed	19%	23%	23%	9%

A design and software development team is involved in a project development process. Depending on the field of the project, there may be different teams such as front-end developer, back-end developer, mobile application team (Android and Iphone Operating System), test and Development Operations (DevOps) team. The physical vector that allows users to engage with software systems is called a UI. While UI designers are in charge of UI design and comprehensive analysis, software engineers are in charge of integrating UI elements into computer packages [3].

User Experience (UX) creates wireframes in the first step of developing applications for users to decide on the UI. Before starting the project, which tool and technique will be used for project management and the teams that will take part in the project are determined. Projects that progress only in the theoretical scope are terminated before they can be implemented to a major extent. In software projects, fundamental technologies that provide equipment, applications, services, and information to support operations, management, analysis, and decision-making functions within an organization are widely applied [4]. As shown in Figure 1, at the beginning of the project, wireframes and prototype models are created by the UX team. The created drawings are sent to the interface designers to be converted into a real design. After the interface designers have completed the design, they send it to their front-end and/or mobile teams for coding, depending on the project. Coding the interface design and successful completion of the project is a very time-consuming, redundant, and costly process. It is very difficult to make changes and arrangements on the project during the development process of the projects. As a result, project management must be established, as well as the software development methodology to be utilized in the project.

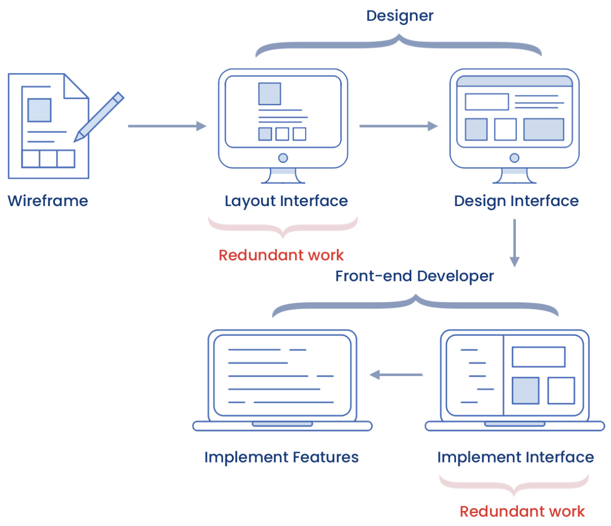

Figure 1. Typical workflow in app and website development.

Wireframes provide users more easily assess the website they want to visit during the planning stage. Furthermore, coding the design and successfully presenting the idea through wireframes is a time-consuming and expensive procedure. The aim of this study is to detect and classify website wireframes as the most efficient on the data registered in the test set. Occasionally, this set of tasks that the system will perform provides well-defined results, which involve complex computation and processing. Managing the entire development process to guarantee that the end-product has a high level of integrity and robustness, as well as user approval, is a difficult and time-consuming task. To accomplish the mentioned features of a successful system, a systematic development method that focuses on comprehending the scope and complexity of the entire development process is required [5]. Conflicts can occur during the project development process between the managers and the development team owing to a variety of disagreements. The software model to be determined within the scope of the project should help the project manager overcome these difficulties. The first step in creating an application after the model is to draw the wireframes that allow the structure of the interface to be determined [6, 7]. As developers decode the generated wireframes and user interface designs, they generate the output according to the designs. This process often takes the developer's time and is therefore costly [8]. Wireframes are generally designed on paper or digital screens, on a white background. For this reason, the data set is restricted to performing on dark-colored wireframes on the ground. The dataset used in the study was drawn with the help of Wacom Bamboo Slate Large CDS-810S model smart agenda. The UI elements drawn on the paper on the Bamboo Slate can be transferred to the cloud server live using the phone, tablet, and computer environment with the help of the Wacom Inkspace application. Our project is *UIBee*, a method for detecting UI elements in hand-drawn wireframe inputs using Convolutional Neural Networks. Our dataset is open to the public in order to encourage future study.[1] The dataset, which consists of 25 classes such as button, image, input, list, checkbox, toggle, video, etc., contains a total of 457 wireframes. There is a wide range of elements, including 3315 UI elements in the training set and 457 in the test set.

Considering the similar studies in the literature, more than one study was able to reach a certain level of accuracy with JavaScript Object Notation (JSON), Domain Specific Language (DSL), or script outputs using traditional machine learning methods. In similar studies, UI elements were detected as a first step, and then the related DSL codes of these detected classes were created and automatic wireframes were programmed. SILK [9] converts digital drawings into code via the application; DENIM [10] enriches drawings to achieve harmony between design tools and code output; REMAUI [11] clipping high-quality screenshots into mobile apps; Sketch2code [12] detects UI elements drawn on paper using Faster R-CNN; Sketch2code [13] detects 5 UI elements trained from wireframes on paper and converts them to Hyper Text Markup Language (HTML) code. Most applications are based on classic computer vision algorithms to recognize and identify. Studies other than these methods are not efficient in terms of the number of UI elements or the estimation mechanism of the model. In Table 2 studies in the literature relating to the detection of UI elements are listed. The use of "-" for the number of UI elements given in Table 2 is due to not specifying the number of UI elements detected in the studies. Most of these studies are based on DSL. DSL outputs are programming languages designed for a specific domain. Compared to full-featured programming languages, DSL outputs are more restrictive. DSL outputs limit the complexity of the programming language, which facilitates automatic programming and makes the special-purpose search algorithm efficient. The hierarchical UI framework was used to detect 25 UI elements in the study by Polozov et al. [14]. In many studies, there is no detailed information about what types of UI elements are detected. To simplify perception in general, some studies have limited the items to be

[1]https://github.com/UIBee-io/Public-Dataset

detected to only a small set [15]. For instance, Asiroglu et al. [16] only considered text, dropdown, button, and checkbox elements. They proposed an approach for automating the code generation process from hand-drawn wireframes using deep learning methods.

Table 2. Related works on detection and segmentation of UI elements.

Literature	Input data	Output data	Classes
Polozov et al., 2015 [14]	UI input	Hierarchical UI	25
Halbe et al., 2015 [22]	Wireframe	HTML	-
Bajammal et al., 2018 [15]	UI input	HTML	-
Beltramelli, 2018 [18]	UI screenshot	DSL	-
Chen et al., 2018[19]	UI input	DSL	-
Han et al., 2018 [24]	Wireframe	HTML	-
Liu et al., 2018 [25]	UI screenshot	DSL	-
Kumar, 2018 [23]	Wireframe	HTML, CSS	16
Kim et al., 2018 [27]	Wireframe	HTML, CSS	5
Robinson, 2019 [13]	UI input	Hierarchical UI	5
Asiroglu et al., 2019 [16]	Wireframe	HTML	5
Beltramelli, 2019 [17]	Wireframe	HTML, CSS	-
Chen et al., 2019 [20]	UI screenshot	DSL, Android, iOS	-
Ge, 2019 [21]	Wireframe	JSON, Android	7
Jain et al., 2019 [26]	Wireframe	HTML	10
Suleri et al., 2019 [28]	UI screenshot	Hierarchical UI	25
Narayanan et al., 2020 [29]	Sketch Wireframe	Hierarchical UI	21
Gupta et al., 2021 [30]	Wireframe	Hierarchical UI	21
Proposed method	**Wireframe**	**Hierarchical UI**	**25**

Asiroglu et al. [16] managed to introduce a total of 5 UI elements on wireframe objects as input and transform them to HTML output. In addition, UI inputs were transformed to HTML output in a 2018 study by Bajammal et al. [15]. Code generation with visual inputs was still an unexplored area of research until another study suggested Pix2code by Beltramelli [17]. Pix2code architecture is similar to some models applied to other domains. The result of Pix2code [18] is beyond expectation as Pix2code's code generation lacks attention mechanism. Chen et al. [19] transformed the UI inputs they took as inputs to DSL output. Chen et al. [20] transform the UI screenshots they take as inputs to DSL, Android, and IOS outputs in other studies. Ge [21] transforms wireframe images into Android and iOS outputs. They managed to detect 7 UI elements in total.

Halbe et al. [22], Han et al. [24], and Jain et al. [26] provided the transformation of wireframe images running on the web platform to HTML output. Also, Jain et al. [26] achieved to detect of a total of 10 UI elements. In the study, 4256 UI elements were used in a total of 457 wireframes. In the study by Kim et al. [27], a total of 5 UI elements were detected on wireframes and transformed to HTML and Cascading Style Sheets (CSS) output. In addition, Kumar et al. [23] have detected a total of 16 UI elements using wireframe inputs. In the study published in 2018, Liu et al. [25] used UI screenshots as input and transformed them into DSL code output. In his research by Robinson [13], the publishing of the data set revealed that the approach of deep learning was superior to the standard way of image processing. Only Jain et al. [26] and Robinson [13] detected the text as separate paragraph and heading. Ultimately, the hierarchical UI framework was used to detect 25 UI elements in the study by Polozov et al. [14]. Narayanan et al. [29] classified and segmented 21 classes using the CNN network derivatives Cascade R-CNN [31] and You Only Look Once-v4 (YOLOv4)[32] algorithms. The results found the overall precision values to be satisfactory, while the performance metrics needed to be considered more successful individually. Gupta et al. [30], preprocessed the open-source dataset in the ImageCLEFdrawnUI 2021 competition by performing contrast enhancement with adaptive histogram equalization on 4291 images. Then, in the object detection step, using the You Only Look Once-v5 (YOLOv5) [33] model, they detected the UI components in the wireframe data with a success rate.

The proposed method in the study aims to improve the performance results in the literature and presented in the studies. Thus, training 12 neural networks improve existing traditional methods, and experimental results are included. The main contribution of our work can be summarized as follows:

- Prepared the dataset used in the study according to real-world data by blending the sketch drawings in the literature and the wireframes drawn by the UI designers using a smart digital device.
- In addition to the Mask R-CNN algorithm, which is the first stage of the study, an improved algorithm has been proposed by updating the pipeline's convolution, activation blocks, and activation functions.
- Individual class objects in the dataset are automatically cropped according to their bounding boxes. The results are obtained by training the segmentation algorithms with state-of-the-art neural networks for comparison purposes.

In summary, in the first part of this study, the critical literature review and comprehensive dataset analysis are included to introduce the paper. For the next section, materials and methods, the preprocessing of the input images was carried out in the first step. In addition, labeling of preprocessed wireframe inputs is also done in this context. Then, in the second part of the study, the methods and segmentation tasks foreseen to be used are explained in the method section. This section gives the neural network parameters used for the segmentation network and the backbone architecture in detail. In addition, the classification studies that we developed, in addition to segmentation networks and structures that will support the academic literature, are also included. The performance values obtained from the experimental studies are included in the third part of the study. In particular, the updates and changes we made on the neural network we used in the article are included in this section. The next section includes conclusions after the experimental findings and future studies. The performance of the study is discussed in detail, and the future studies section is added for the deficiencies revealed.

2. Materials and methods

UI elements are the basic building blocks for all applications. It is the most integral part of a mobile, web, desktop, or virtual reality application [34]. UI elements are divided into 3 major categories. It is responsible for processing different user inputs for input elements. The most used input elements are dropdown, button, combo box, toggle, input, date picker, radio button, and checkbox. Elements are responsible for displaying results against various user inputs for output elements. They also show user information, warning, and successful and incorrect messages. Output items are not impartial in nature. Examples include toast or popups. All other items fall under the category of supplemental items. Commonly used; are notifications, breadcrumbs, icons, progress bars, and tooltips. Supplemental items are divided into 3 navigations, tooltips, and boxes. Navigation elements are responsible for UI navigation and help navigation. Informative is responsible for representing the information. Another element type is boxes/containers.

2.1. Data preprocessing

Image processing means a computer using an algorithm for modifying digital images [35, 36]. It is a method through which some image operations are performed in order to obtain an improved image or to gather some valuable information. The input is an image and the output can be an image or the characteristics that are associated with the image [37]. Image enhancement processes were applied to the data in order to perform a better analysis of the data. Image enhancement is the process of adjusting digital images for further image analysis. The wireframes used in this study contain less noise than the drawings created on paper, as they are created in a digital environment. Fluctuations occur in the sample data in the dataset, as the human component plays an important role in the image and button objects. In such cases, adaptive threshold filtering is applied to the image and a simple threshold filter is used for all pixels. Then, the image was inverted with the *ThreshBinaryInv* module available in the Open Source Computer Vision (OpenCV) library. In adaptive threshold filtering, threshold values are calculated for smaller regions using different threshold values [38]. Adaptive threshold filtering specifies to use Gaussian [39] or Mean [40] filtering with the parameter named adaptiveMethod when thresholding on an image. In adaptive thresholding, firstly, UI elements in wireframes are clipped. Accordingly, preprocessing steps of the image are provided in small images with some UI elements. After the preprocessing steps were completed, the process of inverting the image was started. In this step, the black lines formed as a result of drawing the background and UI elements that were white in the original images were converted to the opposite color tone. When considered in binary, pixels with 0 are inverted as 1, and pixel values with 1 are reversed as 0. In Figure 2, the state of the draft drawing after reversing is given. *ThreshBinaryInv* function is used to bring the image in Figure 2 to this step. Thus, the image was improved and the neural network was able to detect objects with higher performance. When the text in the preprocessed wireframe is examined, the wireframe elements remain smoother than the edge image results. In the preprocessed wireframe, objects are highlighted by converting black lines to white, white lines, and the background to black. In this way, Figure 2 shows images of why preprocessing is important in wireframes.

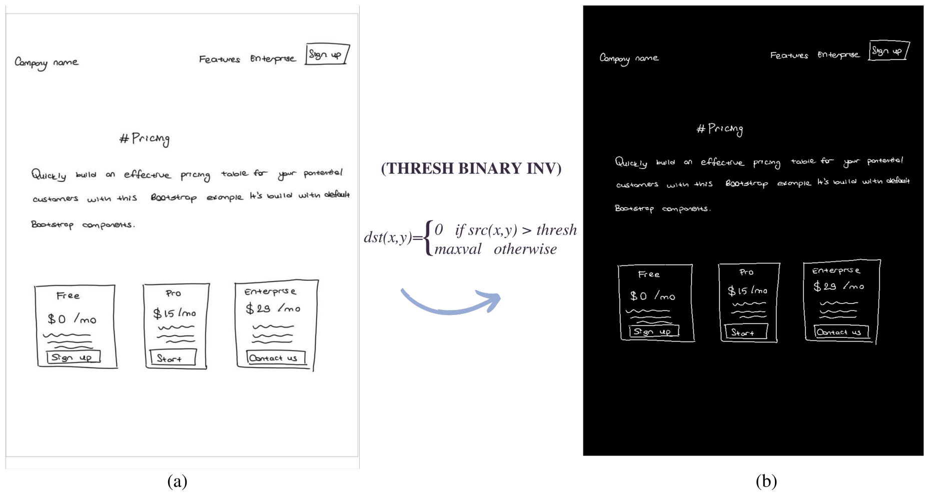
Company name
Features Enterprise Sign up
# Pricing
Quickly build an effective pricing table for your potential customers with this Bootstrap example. It's build with default Bootstrap components.
Free $0 /mo Sign up
Pro $15 /mo Start
Enterprise $29 /mo Contact us
(THRESH BINARY INV)
dst(x,y)= { 0 if src(x,y) > thresh; maxval otherwise }
(a)
(b)

Figure 2. Sample of original and preprocessed wireframe. (a) Original wireframe. (b) Preprocessed wireframe.

2.2. Data labeling and annotations

The ground-truth data, from which the neural network will learn the information, is labeled by selecting the coordinate areas in this step. This step is called data labeling. As seen in Figure 3, each of the UI elements in the wireframes (for a total of 25 class categories) is carefully labeled within the bounding boxes. Considering that in the labeling stage, namely annotations, UI objects are drawn with different nonstraight lines in some cases, it was found appropriate to draw them as segmentation polygon points rather than as rectangular bounding boxes. In the study, one of the many labeling tools in the literature and frequently used VGG Image Annotator (VIA) labeling tool. VIA software provides users with tools such as square, rectangle, circle and polygon for use in labeling [41]. Since UI elements can be found at very close distances, nested, and in irregular shapes, the polygon tool has been preferred. The corner coordinates are stored in the JSON file for each image using the VIA tool during the labeling and annotations stage.

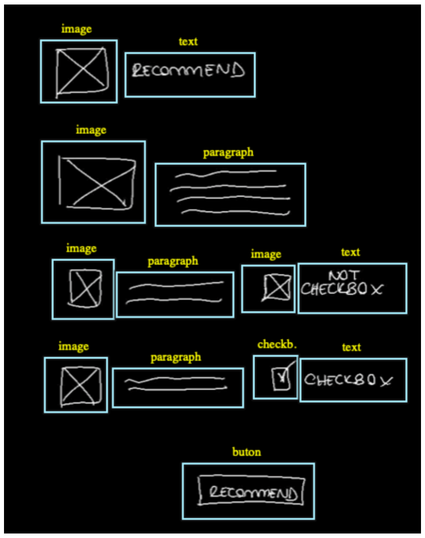

Figure 3. Sample of wireframe labeled with the VIA tool.

Wireframe elements must first be detected before they can be classified. Because a wireframe design will have multiple elements, a procedure for recognizing element boundings is essential [13]. When detecting the boundaries of the elements, various detection methods are used. Deep Learning (DL) and especially image segmentation were preferred in this study. DL method, which is a subbranch of Artificial Intelligence, was preferred for object detection methods [42]. DL allows a computer to generate complex notions out of easier things [43]. DL includes many Artificial Neural Networks (ANNs) in its content. In DL projects, the transfer of ANNs is shown in every existing layer. When put together in a closely interconnected network, a set of processing units provides a remarkable amount of detail that exhibits several unique properties of neurons and networks. It is called "ANNs" [44]. CNNs are a kind of ANNs that work on images. CNNs are designed to resemble a brain with artificial neurons and are comprised of hierarchically replicated layers. These biological neurons use the image as a source, augment it by weight, apply a bias factor, and the activation function. Thus, artificial neurons can perform object detection, identification, and segmentation by performing basic calculations. An efficient and more reliable accurate DL architecture can be acquired by feeding the CNN with more data [45, 46].

2.3. Image segmentation

Image segmentation, which is very important for computer vision, is introduced as the division of an image into regions according to some criteria where regions of an image are meaningful and discrete [47]. Figure 4 shows the segmentation workflow we created in wireframes. According to this workflow, the object detection step in wireframes is image segmentation. Some default parameters were selected for the prepared algorithm, changes were applied in some hyper-parameters, and thus the parameters that provided the most performance were used. As shown in Figure 4, the UI elements in the wireframe were 10 passed through the two-stage Region Proposal Network (RPN). The properties of the UI elements coming from the Residual Network-101 (ResNet-101) backbone network were mapped and sent to the pooling layer. Overlap boxes were scanned according to the Non-maximum Suppression (NMS) algorithm. Regions containing objects were separated into anchors with a certain threshold value with the anchor-based approach in the model. Then, the maximum values in the pooling layer were selected according to the Mask RCNN algorithm. As a result, the segmented masks of the UI elements are shown in the output layer. While Faster R-CNN is generally used in object detection studies in the literature, it was decided to use Mask R-CNN for the formation of masks as well as for detection in this study. Mask R-CNN is a technique for bounding box identification that extends Faster R-CNN by adding a section to predict an item mask in parallel with the current one [48]. Furthermore, the Region of Interest (ROI) estimates segmentation masks in parallel to the current classifier branch and bounding box regression. The purpose of the mask, created in addition to Faster R-CNN, is the intersection between the ROI and the ground-truth mask UI elements to be detected in wireframes are determined as objects as a result of the binary masks created. In order to reduce the computational complexity of the data recorded at various resolutions, the data sizes are reduced to $1024x1024$. The parameters and hyper-parameters in Table 3 have been selected in the residual backbone network to achieve the highest performance. The strides used for the ResNet-101 architecture are in the range of 4-64 and selected the minimum threshold value of the NMS algorithm as 0.3. However, the object tiles in the wireframes will be considered correct if there is a minimum of 90% or more detection. For this reason, the detection minimum confidence value is set at 0.9. Figure 5 shows a wireframe of a sample pricing page taken from the dataset. According to this figure, input and output images are shown side by side. When the figure is investigated, it is concluded that many UI objects are hosted together and collected in a wireframe. For example, when the confidence score of the purple masked “paragraph” object was examined, it was determined with 100% accuracy. On the other hand, it is seen that each box object contains text, paragraph, and button. As a result, the detection of these objects in the box with high accuracy also proves the performance of the model. As shown in Figure 5, the first input represents a data sample from the dataset, while the other input image represents the output result from the test set as a result of the Mask R-CNN approach. As shown in Figure 4, while the original image is given as input to the neural network, the features expected to exist in the object are added to a map by passing through Mask R-CNN stages, while the objects to be detected by the RPN are presented in the background, and the objects are segmented. The segmentation flow shown in Figure 4, step-by-step is performed, and UI elements are assigned to high-accuracy class scores. Although the high-performance Mask R-CNN approach used in the study has contributed satisfactorily to the academic literature, other state-of-the-art classification CNNs have been tested in order to add emphasis to the originality and novelty of the study.

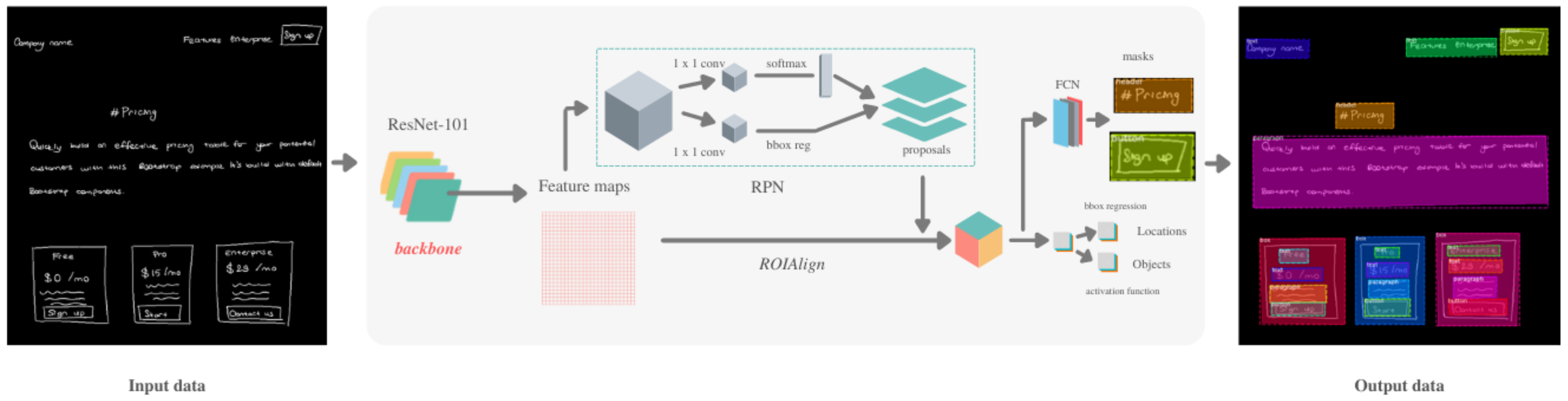

Figure 4. Our segmentation architecture consisting of two-stage RPN in wireframes.

Table 3. Selected parameter properties and values for the ResNet-101 backbone network.

Parameter and hyper-parameter selection	Details
Weight decay	0.0001
Detection NMS threshold	0.3
Learning rate	0.001
Learning momentum	0.9
RPN and NMS threshold	0.7
Detection minimum confidence	0.9
Detection maximum instance	100
Backbone strides	[4, 8, 16, 32, 64]
RPN anchor scales	[32, 64, 128, 256, 512]

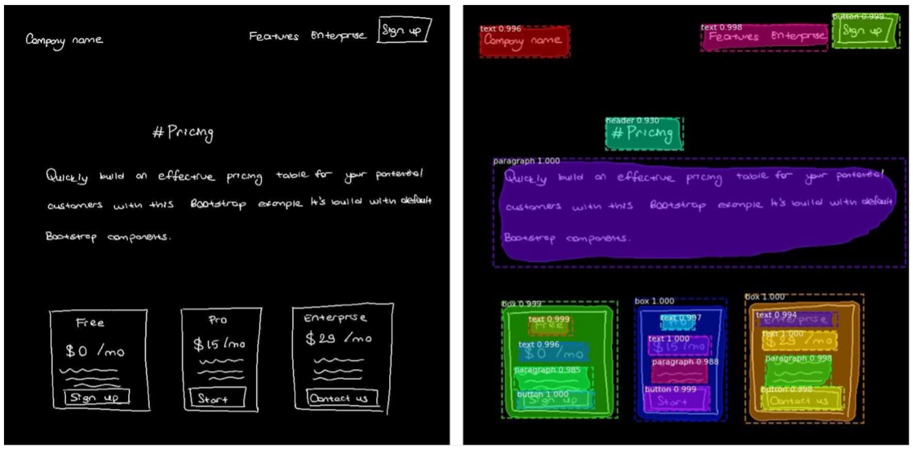

Figure 5. Sample of original and segmented wireframes.

In the process of gathering the data, according to the existing label information in the ground-truth (annotation) file, cropping was provided according to the points of each bounding box. Each data in the dataset, which has different input sizes, has been prepared according to the appropriate size of the neural network to be used (usually $224x224$). In the next classification task of the study, it is aimed to give successful results by training the models of ResNet [49], Alex Network (AlexNet) [50], Visual Geometry Group Network (VGGNet) [51], Densely Network (DenseNet) [52], and Mobile Network (MobileNet) [53], which are CNNs. The Adam and Adamax [54], Stochastic Gradient Descent (SGD) [55], Root Mean Squared Propagation (RMSProp) [56] optimizers and Step Learning Rate (StepLR) scheduler with Categorical Cross-entropy Loss function have been made to perform better in classification for UI dataset. To test the classification task of the Mask R-CNN, a pipeline as shown in Figure 6 was designed using transfer learning supported pretrained neural networks to compare with other classification algorithms.

Figure 7 differs from other classification network architectures. This step aims to improve the ResNet graphs created in the traditional Mask R-CNN algorithm with various changes in the area shown in yellow in the figure. Accordingly, the activation function formed after Batch Normalization in the ResNet graph changed to Softmax. Then duplicate the identity blocks required for Stage 3. Activation function blocks have been edited, and layers have been added. Within the scope of the study, innovation has been made by using state-of-the-art neural networks to handle Mask R-CNN and improving the existing algorithm. Although Mask R-CNN used in the study is a distinctive method in the literature, the assumed parameter values used for neural network architecture may provide low performance in some studies. During the process of obtaining the changed parameters and the change in performance, 12 different experiments were carried out. Thus, determined the most suitable parameters and hyper-parameters for the wireframe data. The variation of the activation function used during normalization is based on the fact that the prediction accuracy of a neural network is determined by the type of activation function used [57]. Therefore, instead of the consecutively placed Rectified Linear Unit (ReLU) function, the inputs were transferred to the positive graph in the range of 0-1 with the Softmax activation. For the identity blocks in the neural network, after examining the number of blocks and shaping an architecture, the C3: (input data, 3, [128, 128, 512], stage=3, block='e', train-bn) layer was added for the ResNet graph.

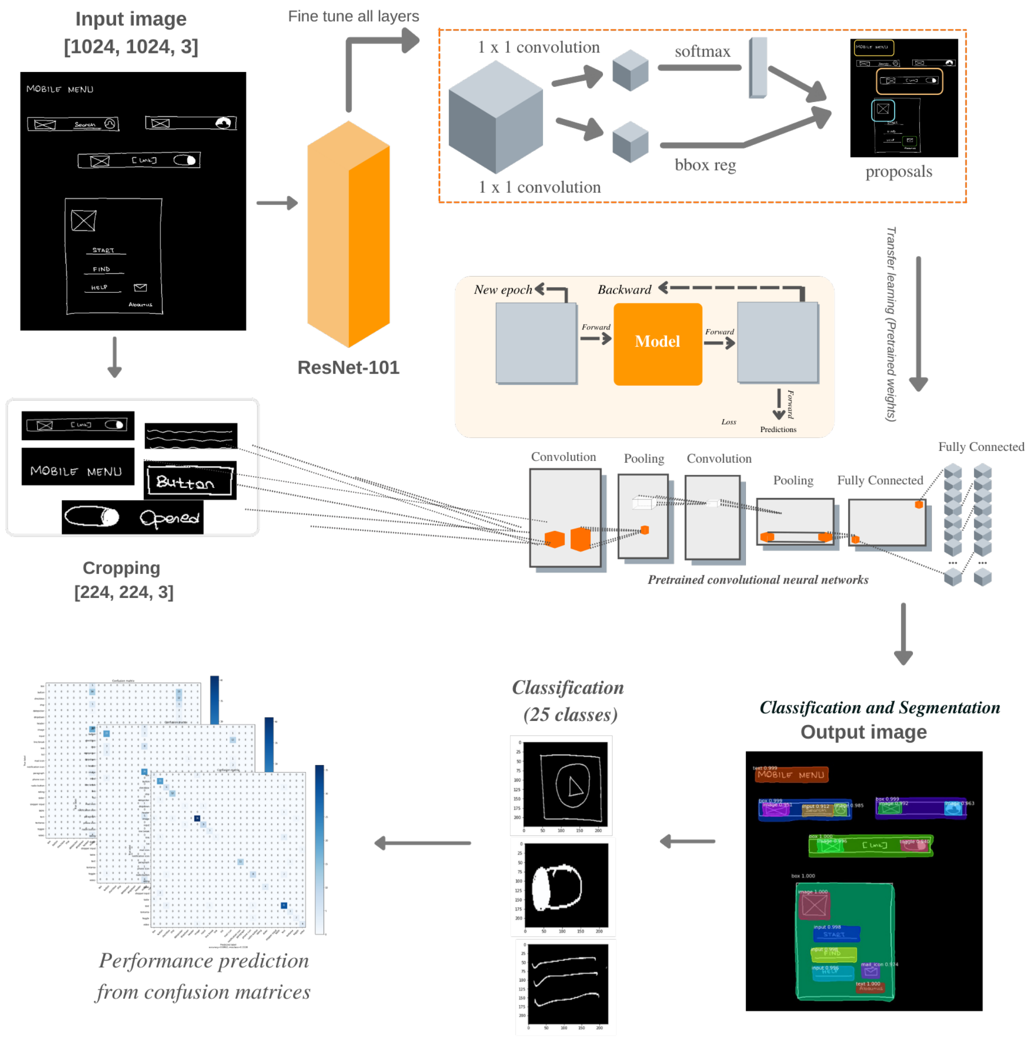

Figure 6. The architecture of detecting UI elements with CNNs as a result of reducing the dataset to be classified and segmented to convenient sizes.

3. Experimental results

In this study, computations were carried out on a computer with i9 10980XE processor and NVIDIA Quadro RTX 5000 graphics card to train the existing neural network through training data. Instead of training to obtain the weights of Mask R-CNN, the objects in the test images with annotations were trained using the pretrained weights with the Microsoft Common Objects in Context (MS COCO) dataset [58], supported by transfer learning strategies. Pretrained high-performance training weights are used to alleviate the workload during training [59]. Therefore, these pretrained weights are used in ANNs without the need for weight training again. Some computational metrics are available to calculate the performance rate of the neural network model during training and additionally to observe the check for network overfitting. The simplest and most common method used to evaluate model success is to look at the accuracy rate. The loss rate is found by dividing the number of misclassified samples by the total number of samples. In other words, the error rate is the value that completes the accuracy rate to 1 [60]. During the training stage of the training data, the epoch was gradually

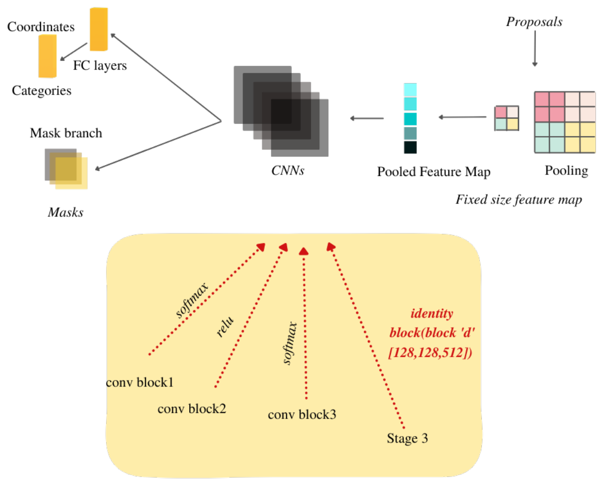

Figure 7. Improved Mask R-CNN algorithm by adding various convolution blocks and activation functions.

increased. After determining the weights to be used while creating the model in the neural network and the backbone network, 10, 50, 100, 150, 250, and 300 epochs of training were provided, respectively. As the training stage gets longer, the learning rate of the neural network increases, and in addition, the segmentation mask areas created highlight the more accurate areas.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

The calculation of the performance metrics used in this study is given in (1), (2), (3), and (4). As a result, the Mean Average Precision (mAP) value is obtained by taking the average of all average precision scores. True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values in the given equations represent the estimation results obtained by taking from the confusion matrix seen in Figure 8 with the data estimated by the model. When the values in the table were examined, it was observed that the increase in training success after 150 epochs decreased over time. For this reason, it was decided that the desired target was achieved as a result of 300 epochs, and there was no need for retraining.

The confusion matrix measures the similarity between actual and predicted data. When Figure 8 is examined, the red colored boxes in the matrix represent the faulty areas. Green boxes represent TP values. When the confusion matrix is checked, it is noticed that the number of erroneous areas is kept to a minimum, and correct predictions are increased. Thus, it is seen that the model achieves successful results in learning the classes. For example, when the actual-prediction results for the paragraph class are examined, it is seen that

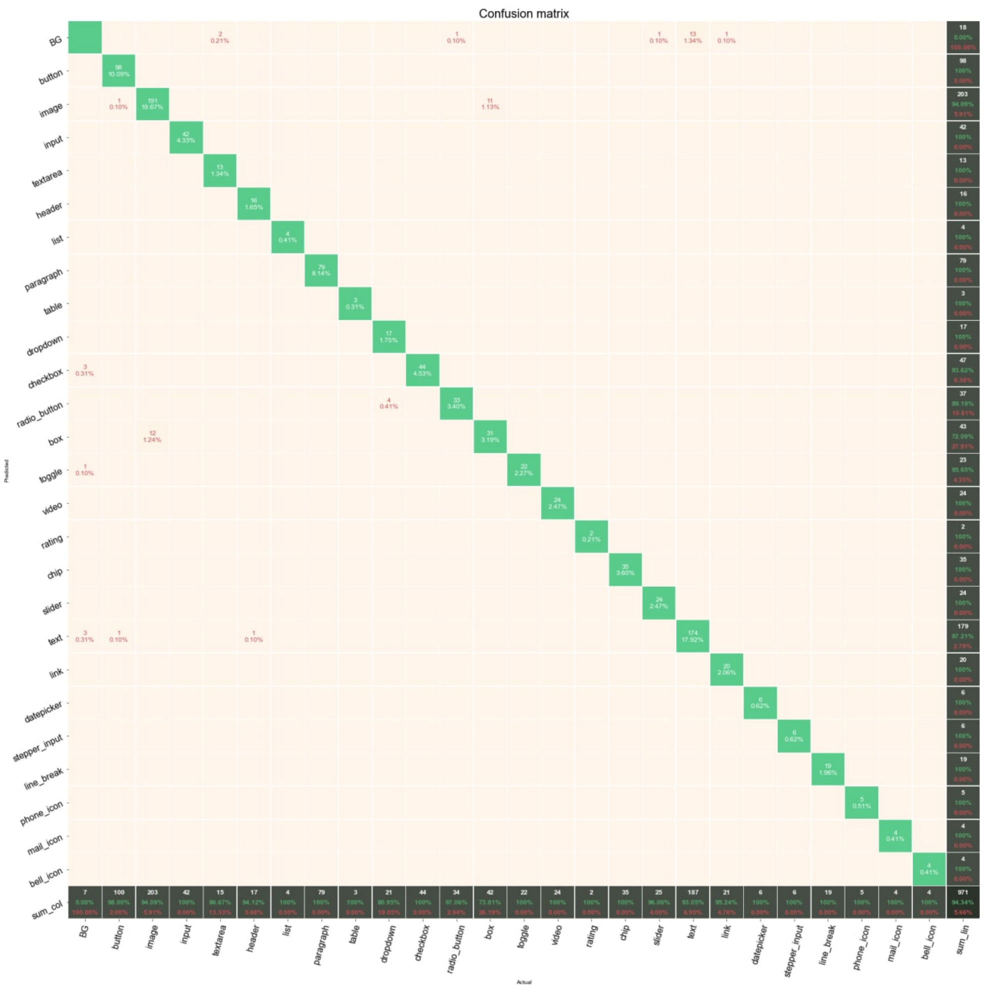

Figure 8. Confusion matrix for UI classes as a result of training the proposed improved Mask R-CNN algorithm.

79 paragraph data results in 100% accuracy. Therefore, negative prediction results will be treated as 0. When the numerical values in the table are examined, it is seen that there is a linear increase. Thus, the Mask R-CNN algorithm allows object detection, and the improved Mask R-CNN algorithm has reached higher accuracy than other models. When Table 4 is examined, the classification algorithms frequently used in the literature are seen. Using different optimizer parameters and momentum coefficients, calculated the results according to the metrics included in the equations. Considering the same neural network architectures and hyper-parameter values, the Adamax optimizer function performs higher than other networks in wireframes required for detecting UI elements. A self-acting comparative analysis is accomplished between multiple optimizers, LR Scheduler, and loss function to achieve the highest accuracy suitable for the proposed system.

4. Conclusion

In this study, Mask R-CNN was used to detect the UI elements in the hand-created wireframe data. The weights used in the activation function for the inputs in the neural network are pretrained MS COCO weights. RPN and Feature Pyramid Network (FPN) architectures existing in the internal structure of Mask R-CNN are used

Table 4. Comparison table of various network models for classification of UI elements based on 25 separate class categories. This table is based on a comparison of findings of different models for the classification study (LR: learning rate).

Algorithm	Opt.	LR	Momentum	Accuracy	Precision	Recall	F1-Score
VGGNet-16	Adam	0.001	0.9	51.97%	51.96%	51.56%	51.37%
VGGNet-16	Adam	0.001	0.5	64.29%	63.58%	64.28%	63.93%
VGGNet-19	Adam	0.001	0.9	64.29%	64.28%	64.64%	64.45%
VGGNet-16	SGD	0.001	0.5	69.10%	69.79%	66.58%	68.14%
AlexNet	Adam	0.001	0.9	72.13%	72.14%	72.13%	72.13%
DenseNet-169	Adam	0.001	0.5	80.00%	80.89%	80.44%	80.66%
ResNet-34	Adamax	0.001	0.9	84.07%	84.53%	83.60%	84.06%
ResNet-18	Adamax	0.001	0.9	88.46%	88.46%	86.56%	87.49%
DenseNet-161	Adamax	0.001	0.9	88.46%	88.95%	89.94%	89.44%
MobileNet-v2	Adam	0.001	0.9	90.66%	87.76%	90.65%	89.13%
Mask R-CNN	Adam	0.001	0.9	96.50%	93.15%	95.43%	94.27%
Improved Mask R-CNN	**Adam**	**0.001**	**0.9**	**98.49%**	**96.06%**	**96.26%**	**96.16%**

for feature extraction. As a result of the training of the neural network, a total of 87 test images included in the data set but not present in the training set was given to the network for testing [61]. After training the neural network up to 300 epochs, respectively, for training, 87 test data were tested and it was determined that the highest value for the mAP value, which was used as a performance metric, was 96.50%. Among the CNNs, MobileNet-v2 gave the highest accuracy with 90.66%. As a result, the precision reached 93.15% and the mAP (@Intersection-over-Union, IOU$>$0.5) reached 96.50%. Furthermore, the algorithm was improved by replacing the conv blocks in the existing Mask R-CNN graphs, adding them, and changing the input units, and the accuracy increased up to 98.49%. In addition, the increase in precision, recall, mAP and f1-score values with various changes for the existing Mask R-CNN algorithm also proved that the study contributed to the scientific literature.

For future studies, wireframes containing UI elements that are frequently included in the literature but not included in the created data set can be included in the study, so that more UI elements can be detected. Due to the drawing differences caused by the designers when creating the wireframes, the expected performance may not be achieved in the testing process of the wireframes. Therefore, it can be expected that the data set will be replicated as much as possible for future studies and that the network will give performance results in outlier data. In addition, in the software and design industry, the UI/UX stages that exist during the creation of the project, but take a lot of time, can be converted into software codes for a project.

Acknowledgment

This work was supported by Scientific Research Projects Unit of Karabük University under project number FYL-2020-2156. The authors appreciate the financial and scientific support.

References

[1] Georgiou S, Rizou S, Spinellis D. Software development lifecycle for energy efficiency: techniques and tools. ACM Computing Surveys (CSUR) 2019; 52 (4): 1-33.

[2] Eveleens J, Verhoef C. The rise and fall of the chaos report figures. IEEE software 2009; 27 (1): 30-36.

[3] Mi Q, Keung J, Xiao Y, Mensah S, Gao Y. Improving code readability classification using convolutional neural networks. Information and Software Technology 2018; 104: 60-71

[4] Strunk Jr W, White EB. The Elements of Style Illustrated. Penguin, 2007.

[5] Leau YB, Loo WK, Tham WY, Tan SF. Software development life cycle AGILE vs traditional approaches. In: International Conference on Information and Network Technology; 2012; 37 (1): 162-167.

[6] Campos P, Nunes NJ. Practitioner tools and workstyles for user-interface design. IEEE software 2007; 24 (1): 73-80.

[7] Landay JA, Myers BA. Interactive sketching for the early stages of user interface design. In: Proceedings of the SIGCHI conference on Human factors in computing systems 1995: 43-50.

[8] Da Silva TS, Martin A, Maurer F, Silveira M. User-centered design and agile methods: a systematic review. In: 2011 AGILE conference 2011; 77-86.

[9] Landay JA, Myers BA. Sketching interfaces: Toward more human interface design. In: Computer 2001; 34 (3): 56-64.

[10] Lin J, Newman MW, Hong JI, Landay JA. DENIM: Finding a tighter fit between tools and practice for web site design. In: Proceedings of the SIGCHI conference on Human factors in computing systems 2000: 510-517.

[11] Nguyen TA, Csallner C. Reverse engineering mobile application user interfaces with remaui (t). In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE) 2015; 248-259.

[12] Zita A, Picek L, Říha A. Sketch2Code: Automatic hand-drawn UI elements detection with Faster R-CNN. 2020.

[13] Robinson A. Sketch2code: Generating a website from a paper mockup. arXiv preprint 2019; arXiv: 1905.13750.

[14] Polozov O, Gulwani S. Flashmeta: A framework for inductive program synthesis. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications 2015; 107-126.

[15] Bajammal M, Mazinanian D, Mesbah A. Generating reusable web components from mockups. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2018; 601-611.

[16] Aşıroğlu B, Mete BR, Yıldız E, Nalçakan Y, Sezen A et al. Automatic HTML code generation from mock-up images using machine learning techniques. In: 2019 Scientific Meeting on Electrical-Electronics, Biomedical Engineering and Computer Science (EBBT); 2019. pp. 1-4.

[17] Beltramelli T. Hack your design sprint: wireframes to prototype in under 5 minutes. 2019.

[18] Beltramelli T. pix2code: Generating code from a graphical user interface screenshot. In: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems; 2018. pp. 1-6.

[19] Chen C, Su T, Meng G, Xing Z, Liu Y. From ui design image to gui skeleton: a neural machine translator to bootstrap mobile gui implementation. In: Proceedings of the 40th International Conference on Software Engineering; 2018. pp. 665-676.

[20] Chen S, Fan L, Su T, Ma L, Liu Y et al. Automated cross-platform GUI code generation for mobile apps. In: 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile); 2019. pp. 13-16.

[21] Ge X. Android GUI search using hand-drawn sketches. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion); 2019. pp. 141-143.

[22] Halbe A, Joshi AR. A novel approach to HTML page creation using neural network. Procedia Computer Science 2015; 45: 197-204.

[23] Kumar A. Automated front-end development using deep learning. Medium Insight, 2018.

[24] Han Y, He J, Dong Q. CSSSketch2Code: An Automatic Method to Generate Web Pages with CSS Style. In: Proceedings of the 2nd International Conference on Advances in Artificial Intelligence; 2018. pp. 29-35.

[25] Liu Y, Hu Q, Shu K. Improving pix2code based Bi-directional LSTM. In 2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEE) 2018; 220-223.

[26] Jain V, Agrawal P, Banga S, Kapoor R, Gulyani S. Sketch2Code: transformation of sketches to UI in real-time using deep neural network. arXiv preprint 2019; arXiv: 1910.08930.

[27] Kim B, Park S, Won T, Heo J, Kim B. Deep-learning based web UI automatic programming. In: Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems 2018; 64-65.

[28] Suleri S, Sermuga Pandian VP, Shishkovets S, Jarke M. Eve: A sketch-based software prototyping workbench. In: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems; 2019. pp. 1-6.

[29] Narayanan N, Balaji NNA, Jaganathan K. Deep Learning for UI Element Detection: DrawnUI 2020. In: CLEF (Working Notes), 2020.

[30] Gupta P, Bansal V. UI element detection from wireframe drawings of websites. In: CLEF (Working Notes); 2021. pp. 1239-1252.

[31] Cai Z, Vasconcelos N. Cascade R-CNN: high quality object detection and instance segmentation. IEEE transactions on pattern analysis and machine intelligence 2019; 3 (5): 1483-1498.

[32] Bochkovskiy A, Wang CY, Liao HYM. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint 2020; arXiv: 2004.10934.

[33] Zhao Y, Shi Y, Wang Z. The Improved YOLOV5 Algorithm and Its Application in Small Target Detection. In: International Conference on Intelligent Robotics and Applications; 2022. pp. 679-688.

[34] Tool RWDP. UXPin newest solution. UXPin.

[35] Chakravorty P. What is a signal? [lecture notes]. IEEE Signal Processing Magazine 2018; 35 (5): 175-177.

[36] Gonzalez RC, Woods RE. Digital Image Processing, 3rd ed, Prentice Hall: Upper Saddle River, NJ, USA, 2008.

[37] Deny J. Digital image processing. Smashwords, Inc, 2016.

[38] Akinbade D, Ogunde AO, Odim MO, Oguntunde BO. An adaptive thresholding algorithm-based optical character recognition system for information extraction in complex images. Journal of Computer Science 2020; 1 (6): 784-801.

[39] Xie X, Huang W, Wang HH, Liu Z. Image de-noising algorithm based on Gaussian mixture model and adaptive threshold modeling. In: 2017 International conference on inventive computing and informatics (ICICI); 2017. pp. 226-229.

[40] Bradley D, Roth G. Adaptive thresholding using the integral image. Journal of graphics tools 2007; 12 (2): 13-21.

[41] Dutta A, Zisserman A. The VIA annotation software for images, audio and video. In: Proceedings of the 27th ACM international conference on multimedia 2019; 2276-2279.

[42] Chollet F. Deep learning with Python. Simon and Schuster, 2017.

[43] McCarthy J. What is artificial intelligence?. 2007.

[44] Yegnanarayana B. Artificial neural networks. PHI Learning Pvt. Ltd., 2009.

[45] Ozdemir MA, Elagoz B, Soy AA, Akan A. Deep learning based facial emotion recognition system. In: 2020 Medical Technologies Congress (TIPTEKNO) 2020; 1-4.

[46] Çelik A, UĞUZ S. A deep learning based system for real-time detection and sorting of earthworm cocoons. Turkish Journal of Electrical Engineering and Computer Sciences 2022; 30 (5): 1980-1994.

[47] Zhang H, Lee K, Chen Z, Kashyap S, Sonka M. LOGISMOS-JEI: Segmentation using optimal graph search and just-enough interaction. In: Handbook of Medical Image Computing and Computer Assisted Intervention 2020; 249-272.

[48] He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision 2017; 2961-2969.

[49] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2016; 770-778.

[50] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Communications of the ACM 2017; 60 (6): 84-90.

[51] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint 2014; arXiv:1409.1556.

[52] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. pp. 4700-4708.

[53] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint 2017; arXiv:1704.04861.

[54] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014; arXiv preprint 2014; arXiv:1412.6980.

[55] Ketkar N. Stochastic gradient descent. Deep learning with Python; Apress, Berkeley, CA 2017; 113-132.

[56] Shi N, Li D. RMSprop converges with proper hyperparameter. In: International conference on learning representation; 2020.

[57] Sharma S, Athaiya A. Activation functions in neural networks. International Journal of Engineering Applied Sciences and Technology, Towards Data Science 2017; 6 (12): 310-316.

[58] Lin TY, Maire M, Belongie S, Hays J, Perona P et al. Microsoft coco: Common objects in context. In: European conference on computer vision 2014; 740-755.

[59] Pan SJ, Yang Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering 2018; 22 (10): 1345-1359.

[60] Bilgin M. Gerçek veri setlerinde klasik makine öğrenmesi yöntemlerinin performans analizi. Breast 2017; 2 (9): 683.

[61] Kayabaş A, Topcu AE, Kılınc Ö. A novel hybrid algorithm for morphological analysis: artificial Neural-Net-XMOR. Turkish Journal of Electrical Engineering and Computer Sciences 2022; 30 (5): 1726-1740.