Racing Bib Number Recognition Method using Deep Learning

Muhammad Aditya Rayhan, Kemas Muslim Lhaksmana

School of Computing, Telkom University, Bandung, Indonesia
Email: ‘m.adityarayhan@yahoo.com, *kemasmuslim@telkomuniversity.ac.id
Email Correspondence: m.adityarayhan@yahoo.com

Abstract—A racing bib number is often produced in mass to enable the identification of a person from recorded images. This paper presents a method to automatically annotate images with bib numbers using a CNN-based framework. The proposed method utilizes face detection to approximate RBN position, and combined stroke width transform (SWT) and optical character recognition (OCR) to identify numerical digit within the RBN position. The experiment result shows 83.0% precision, 81.5% recall, and 82.2% F1 score as a result of our proposed method on running event dataset. Therefore, this implemented method will promote efficiency to solve the image annotation problem because it doesn’t require manual annotation over thousand of running event images.

Keywords: Racing Bib Number, Object Detection, Automatic Annotation, CNN, Object Recognition

1. INTRODUCTION

Running originally only practiced by competitive athletes through extracurricular school and university programs. Recreational running becomes common since the late 1960s because more people enjoy running as a leisure-time pursuit and could be characterized as a mass movement [1]. From its root in the US, recreational running has been widespread across the continents with millions of growth in popularity as participants increased from time to time. This popularity boost encourages the organizer of running events to attract more people to participate with various benefits such as a medal, shirt, and documentation as memorabilia for participants to take home. Documentation take a big part to showcase the event after it was done. With a huge amount of documentation generated from one event, the participant is unlikely to find a picture of themselves as it scattered along with thousands of images.

To address this problem, we could annotate the image with participant attributes such as RBN. RBN is a unique number composed of multiple numerical digits used as participant identity. RBN then could be used as a tag to allow the participant to find their related images with ease. Automatic image annotation could be used as a way to eliminate the inefficiency of labor and time consumption caused by manual annotation on an enormous amount of image data. The idea to recognize numerical digit within RBN and to use them as an annotation for running event images could be used as the problem base. With the rapid growth of research in machine learning topic, this problem could be solved using a state-of-the-art method.

A lot of research has been done using multi-digit recognition for a variety of similar problems. Liu et al. designed an R-CNN based framework for an all-in-one solution for person detection, body keypoints prediction and jersey number recognition to identify players within sport match videos [2]. Goodfellow et al. used CNN based model to recognize the multi-digit number from street numbers using street view house numbers (SVHN) dataset [3]. To solve the jersey number recognition problem, Li et al. used a CNN model along with Spatial Transformer Network (STN) to further boost the CNN model performance [4].

Because our problem is mainly about RBN recognition, there are numerous RBN recognition research topics also has been done with various method. Ben-Ami et al. being the earliest to do RBN recognition research, used face detection to approximate RBN position, and combined stroke width transform (SWT) and optical character recognition (OCR) to identify numerical digit within the RBN position [5]. One of its limitations is when the participant’s face is not detected in the image due to the photo’s inconsistent angle hence the RBN recognition cannot be done. Inspired by Ami et al. following work, Roy et al. proposed a method using a multi-modal technique that combines face, skin and text detection to extract text candidate region as participant generally displays skin and then used it as a cue to identify a person body which gives the general position about RBN whereabouts within
an image [6]. Extracted text candidate region was applied with a text detection technique which utilizes a combination of wavelet and color, and k-mean clustering to detect text line. The detected text line then was applied with the binarization technique to get the binarized image result. The final binarized result then processed into the Tesseract OCR engine to recognize RBN. Shivakumara et. al, also proposed a multi-modal technique that combines torso and text detection for RBN detection and eliminates the need for face detection as many marathon image doest work well at showing face features [7]. Torso detection technique that was used is a combination between a linear support vector machine (SVM) classifier to extract histogram of oriented gradients (HOG) and an appearance model based on pictorial structural model (PSM) which used the result of processed SVM result and then output the needed torso region. The region output from torso detection then applied with HOG based text detection method to extract candidate text region height and width information using SVM based on Fourier, Pseudo-Zernike moments, Polar descriptor, and THOG. Extracted text region then used as an input for scene text segmentation based on inverse rendering method that was used for binarizing the RBN which preserve character shapes to improve recognition result by the tesseract OCR engine. Apap et al. proposed a two stages method for RBN detection and recognition based on a deep learning approach [8]. The first stage of the model is the detection stage used an image segmentation based on a convolutional neural network to localize RBN in complex marathon images and extract a segmentation map as the result. The second stage is the recognition stage, using a convolutional recurrent neural network (CRNN) composed by a convolutional layer for feature extraction, 2 bi-directional gated recurrent units (GRUs) as a recurrent layer, and transcription module which used bi-directional GRUs output to produce number sequence output [9]. Apap et al. proposed a deep learning model that eliminates the need for torso and face detection but still achieves a good F-score in complex images. Wong et al. proposed a deep learning cascade model based on a deep learning model that implements the YOLOv3 algorithm and an RCNN model to solve the RBN recognition problem as work to increase efficiency in sorting marathon images [10]. In Wong et al. proposed model, the YOLOv3 algorithm was used to detect runner, racing bib, and number within images to produce required input for the CRNN model. The result YOLOv3 then used by CRNN as a recognition process to generate a label sequence containing RBN. The proposed network prediction performance were evaluated using YOLOv3 algorithm output. The evaluation was done by calculating the correct class prediction and mean average precision (mAP) to detect how well predicted bounding box produced by object detection model compared to the ground truth bounding box.

Our proposed method used a state-of-the-art object detection model based on the YOLOv3 algorithm to solve the RBN recognition problem. You only look once (YOLO) is a unified object detection model proposed by Redmon et al. which used a single convolutional network to predict multiple bounding boxes and class probabilities of boxes at the same time [11]. At test time, YOLO inference time is fast because the model is not composed of a complex pipeline unlike the other method such as R-CNN which used region proposal method [12]. However, the original YOLO model is still lacking in accuracy to localize small objects. But over the last years, the author of YOLO has made an incremental improvement for the YOLO model called YOLO 9000 (YOLOv2) and YOLOv3 [13][14]. This incremental improvement gives YOLO mAP improvement over smaller-size images. The use of their latest network architecture also contributed to improvement over YOLO mAP. Darknet-53 is a network architecture used for feature extraction that has a deeper network compared to Darknet-19 used on YOLOv2 and was used for YOLOv3 as one of many improvements to the original YOLO model. Darknet-53 has 53 convolutional layers with the addition of shortcut connection between layers and performs much more powerful than Darknet-19 but still efficiently compared to ResNet-101 or ResNet-152 on ImageNet dataset. In this work, we tried to eliminate the use of CRNN and OCR as a number recognition technique in RBN recognition as previous works often use. This is done without sacrificing the prediction accuracy by using YOLOv3 state-of-the-art model with an addition of algorithm to produce a numerical digit sequence as RBN label for image annotation. In this work, we proposed a model optimized for racing bib design and font that only used in our dataset. Different design and font used in other racing bib design are not considered in this work.

2. RESEARCH METHODOLOGY

In this work, the research method consisted of several steps as shown in Figure 1. Our proposed RBN recognition method used two YOLOv3 models that detect racing bib and numerical digit separately. The output from each model then combined in the RBN recognition step to produce a complete RBN label for image annotation.
2.1 Dataset collection

As previously mentioned in the introduction section, the dataset in this research consisted of images from a marathon event. We collect 2194 raw images taken from various angles with two types of width and height image resolution at 3068 x 2048 and 1367 x 2048 in jpg format from an event called BNI ITB Ultra Marathon 2019. We choose this event as it provides us with a sufficient amount of image data to train our YOLOv3 model. The image gathered for our dataset has at least one visible racing bib that mostly not obstructed by any other obstruction. This was done to provide more data for racing bib and numerical digit detection model training. A subset of the image we collected is shown in Figure 2.

2.2 Dataset annotation

Because YOLOv3 training input needs a ground-truth bounding box coordinate and class label for each object in the dataset, we manually annotate each image in our dataset using a python based graphical image annotation tool called labelImg that is specifically used to annotate bounding box and class for image dataset used in YOLO model. We split the dataset into two types of annotation, for racing bib detection and numerical digit detection.

For racing bib detection, we annotate each image with class and bounding box over a fully identified and obstructed racing bib with ‘valid’ and ‘invalid’ label respectively and could be turned into a set of class which is $C_{\text{bib}}=$\{‘valid’, ‘invalid’\}. This could be used to train the model to avoid obstructed racing bib getting mixed in the detection process. For numerical digit detection, we annotate each image with class and bounding box over the identified numerical digit within the RBN with a numerical digit from 0 to 9 and could be turned into a set of class which is $C_{\text{numdigit}}=$\{‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’\}. After the annotation process was done, each image will have a list containing ground truth bounding box annotation of each object within the image in a text file with the name
respective to which image belongs. The component of bounding box annotation output consisted of five components as shown in Table 1.

**Table 1. Bounding box annotation component**

| Component   | Description                                                                 |
|-------------|-----------------------------------------------------------------------------|
| <class-id>  | A representation of predicted bounding box class order from “class.txt” in integer form |
| <x-center>  | The x-center coordinate of object bounding box relative to the image resolution |
| <y-center>  | The y-center coordinate of object bounding box center relative to the image resolution |
| <width>     | The width of the object bounding box relative to the image resolution       |
| <height>    | The height of the object bounding box relative to the image resolution       |

### 2.3 Model Training

The detection model training was done in a Windows framework implementation of the YOLOv3 model. This implementation is capable of utilizing tensor cores from our GPU making it faster to train our model. Because we have two different specific use of models, models are trained separately with different network model parameters involved. The main difference of network model parameter between models is the count of the convolutional filter before each YOLO layer in the network as it influenced by the number of classes in each model and the number of iteration during training. To calculate the number of convolutional filters and recommended iteration needed for each network model parameter, we used equation (1) and (2) from the YOLO author:

\[
\# \text{ of filter} = (\# \text{ of class} + 5) \times 3 \quad (1)
\]

\[
\# \text{ of iteration} = \# \text{ of class} \times 2000 \quad (2)
\]

For the YOLOv3 model to make a prediction, we need to pre-defined several anchor boxes size as dimension cluster needed for the model to generate prediction at the network parameter. Anchor boxes can be generated using k-means clustering from object bounding box size within our image dataset. Before each model training, we have to initialize the initial weight of our network model. This was done by utilizing pre-trained weight for Darknet-53 network architecture. By using already a pre-trained weight as a starting point, we eliminated the work to get the optimum weight initialization method. As mentioned before, Darknet-53 has 53 convolutional layers with the addition of a shortcut connection at the residual layer. The network used leaky rel-u and linear activation unit for convolutional and residual layer respectively, the function for leaky rel-u activation unit shown in equation (3) where α is a small constant.

\[
f(x) = 1(x < 0)(ax) + 1(x \geq 0)(x) \quad (3)
\]

The model will perform a backpropagation using stochastic gradient descent at each convolutional layer to computes the gradient needed to adjust the weight of our network model. This was done using binary cross-entropy loss function calculated at each iteration. To further generalize our model to a more variety of data, the YOLOv3 network model will use a random network size at every 10th iteration. Bounding box and class prediction in the YOLOv3 network model were made by using logistic regression at the end of a fully connected network.

### 2.4 Training Evaluation

Training evaluation is key to measure how well our model performs towards the test data. To evaluate our network model training performance, we used an object detection metric called mAP, the mAP is the mean value of average precision (AP) for each object class in our model. AP is an object detector evaluation metric used at the PASCAL VOC Challenge [15]. AP is a way to summarize the shape of the precision/recall curve and calculated using the mean of interpolated precision over 11 recall levels \([0,0.1,...,1]\). Precision is the ratio between the true object prediction and the total number of object prediction by our model. Recall is the ratio between the true object prediction and total number object in the ground truth label. Precision \((p)\), recall \((r)\), and AP calculation can be shown in equation (4), (5), and (6):

\[
p = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (4)
\]

\[
r = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (5)
\]

\[
\text{AP} = \frac{\sum_{i=1}^{11} \text{AP}(i)}{11} \quad (6)
\]
To determine whether an object prediction is defined to be true or false, intersection over union (IoU) was used to measure a ratio between the intersection and the union of the predicted boxes and the ground truth boxes, which also known as Jaccard index [16]. We then set how much IoU threshold for an object to be defined as true. How IoU was calculated in this work is shown in equation (7):

\[ IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}} \]  

On the bottom line, the mAP could be used as a model training performance indicator over all object classes. If the mAP is low, it means that the model training is not going well due to low generalization over test data or incorrect use of network parameters.

**Figure 3. Our RBN Recognition Method Scheme**

**Figure 4. RBN Recognition Method Illustration**

RBN recognition is the core of our method to recognize RBN within an image. We predict the RBN annotation label for each racing bib detected from our test data based on trained racing bib and numerical digit detection model inference result. The following RBN recognition method scheme used in this work is shown in Figure 3 and Figure 4 to further illustrate on how our RBN recognition method works. First, we used the trained weight that produced the best mAP score during model training for model inference step. The model inference will output a list of prediction consisted of class and bounding box coordinate and size from the detected object within an image, this output component is the same as when we annotate our image dataset. To further increase mAP of our prediction, we then tuned the network size of our model into much bigger value to increase mAP of our model inference prediction as we only process one image at a time unlike during model training step. As
mentioned in the paper the use of larger network sizes can increase mAP [15] and the further reason is that most of our object sizes are relatively small compared to the size of original image resolution.

After we got the output prediction from both models, we need to arrange the output prediction into a correct RBN label. This is because the output of both models which are, racing bib and numerical digit bounding box coordinate in form of text file, not representing the correct number sequence in RBN. To arrange numerical digit and racing bib bounding box into a correct RBN label, we created an algorithm specifically for this problem on Python implementation as shown on Algorithm 1.

Algorithm 1. RBN Bounding box arrangement algorithm

| Input: numdigit_label, runbib_label |
|------------------------------------|
| Output: list_rbn                    |
| Initialization i, j, run, rbn, list_rbn |
| Get numdigit_label, runbib_label |
| sorted_num = sortByX(numdigit_label) |
| sorted_bib = sortByX(runbib_label) |
| len_bib= sorted_bib.shape[0] |
| len_num= sorted_num.shape[0] |
| while j < len_bib and i<len_num do |
| r_bound = getRightX(sorted_bib[j]) |
| l_bound = getLeftX(sorted_bib[j]) |
| if !isOverlap(sorted_num[i], r_bound, l_bound) |
| rbn=rbn+str(sorted_num['x-center'][i]) |
| i++ |
| end if |
| else |
| if nomor!='': |
| list_rbn.append(number) |
| j=0 |
| rbn='' |
| end if |
| if j==len_bib |
| i=i+1 |
| j=0 |
| end if |
| end else |
| end while |
| if number!='' |
| list_rbn.append(number) |
| rbn='' |
| end if |

return list_rbn

2.5 RBN Recognition Evaluation

Our main goal in this work is to create a model that could perform automatic image annotation over detected RBN within an image efficiently. To ensure the model is capable to predict RBN label from an image, we need to evaluate the model RBN prediction based on the ground-truth RBN label from our image dataset. Precision, recall, and F1 score are used to measure how RBN predictions were accurately predicted and how well true predictions were found. F1 score is used to describe how well the balance between precision and recall from RBN prediction was made, thus this evaluation will be the final evaluation for our RBN recognition method.

3. RESULT AND DISCUSSION

3.1 Dataset and Hardware Specification

We perform dataset annotation on 2194 images from various angles obtained from the running event. After the dataset annotation was done, the object annotated within our dataset consisted of 11788 numerical digits and 2733 racing bib bounding box annotations. A subset of bounding box annotation and image from the numerical digits dataset is shown in Table 2 and Figure 4.

Table 2. A subset of bounding box annotation component

| <class-id> | <x-center> | <y-center> | <width> | <height> |
|------------|------------|------------|---------|----------|
| 1          | 0.332520   | 0.738266   | 0.016602| 0.033898 |
| 0          | 0.349121   | 0.737125   | 0.023438| 0.034224 |
| 3          | 0.369629   | 0.735007   | 0.022461| 0.035202 |
The main hardware that is very significant for our research is the use of RTX GPU which is capable of running tensor cores to speed up model training and inference time than other GPU. The computer testing setup is listed in Table 3.

### Table 3. Research computer setup

| PC Component | Name of the Component |
|--------------|-----------------------|
| GPU          | NVIDIA RTX 2060 6 GB  |
| CPU          | RYZEN 1600x @3.6 GHz  |
| RAM          | Corsair 2x8 GB DDR4   |

#### 3.2 Model Training and Evaluation

First, we train our numerical digit and race bib detection model separately. Both models were trained on the same 80%-20% training and test dataset ratio to measure mAP(mean average precision) for evaluating the network model performance. Before training, we customized some network parameters to achieve the best result for our dataset. These network parameters including the number of convolutional filters, number of iterations, network size, number of images within a batch, learning rate, IoU threshold, anchor size, and network architecture. The number of classes that were used in racing bib and numerical digit detection models was 2 and 10 classes respectively. As mentioned before we need to calculate the number of convolutional layers and the number of iteration based on total class used in a model. For the racing bib detection model, we used 21 convolutional layers before the YOLO layer and 4000 iterations to train the model. And for the numerical digit detection model, we used 45 convolutional layers before the YOLO layer and 20000 iterations to train the model. However, there are other parameters tuned for both models to achieve the best mAP for our dataset. We used a 640x640 network size, meaning that our image dataset will be resized to fit the network size. Then batch was used to define how many images are needed in each iteration to perform backpropagation mainly to speed up and generalize the training, we used 64 images in a batch. The adjustment of both network size and the number of images in each batch was influenced by GPU memory capacity to prevent it from being overloaded. We used 0.001 as a learning rate value because the use of bigger and smaller didn't improve our model. For anchor size, we tried to use our calculated anchor size from running k-means clustering over our object bounding box in the dataset. But, the training done in customized anchor size was not performing well so instead we used the default YOLO anchor size. We set the IoU threshold to 0.5 during model training. We also tested a few different network architectures to see how Darknet-53 compared to the others. While testing on ResNeXt-50 architecture, the usage of GPU capacity was huge so we had to tuned the batch size down to 16. At every iteration, the weight of the network model will be updated. Trained weight will be saved at every 1000th iteration as a backup weight. The training time for racing bib and numerical digit detection model consumed roughly around 5 hours and 49 hours for each model.
respectively due to the difference in the number of classes and iterations. We tested the numerical digit detection model on a few different settings as shown at Table 4.

Table 4. Numerical digit detection model training evaluation across different settings

| Architecture   | Anchor Size | Batch Size | Iteration | learning rate | Peaked mAP |
|----------------|-------------|------------|-----------|---------------|------------|
| Darknet-53     | default     | 64         | 7500      | 0.001         | 83%        |
| Darknet-53     | default     | 32         | 7500      | 0.001         | 82%        |
| Darknet-53     | k-means     | 64         | 7500      | 0.001         | 64%        |
| ResNeXt-50     | default     | 16         | 7500      | 0.001         | 60%        |
| Darknet-53-5L  | default     | 64         | 7500      | 0.001         | 55%        |

The graph results of training for the racing bib and numerical digit detection model with its evaluation are shown in Figure 6. The graphs showed the model training performance where blue dot as model average loss and red graph as model mAP calculation on the test data. The numerical digit model and racing bib detection performance peaked at 84% and 79% mAP with our customized network settings parameter. Despite the decrease of average loss throughout the end of model training, the mAP toward the test data was not increased significantly. To avoid model overfitting, we then take the trained weight that produces the best mAP towards the test data for model inference.

3.3 RBN Recognition and Evaluation

At RBN recognition, the first thing we do was performing model inference for both models on the test data. Before we do the model inference, we increased the network size of our model by two different sizes for comparison, 864x864, and 1280x1280. Figure 7. shown a subset of both model inference result over test data. After we performed model inference with both models on test data, we take the detected class label and bounding box coordinate from the inference result of both models and combine them with the RBN bounding box arrangement algorithm to produce a complete RBN label. Figure 8. shown the result of our RBN recognition method annotated into the image. To evaluate the proposed method inference result combined with the RBN bounding box arrangement algorithm, we calculated recall, precision, and F1 score using ground truth RBN label and predicted RBN label using the different network for comparison size as shown in Table 5.

Figure 6. Numerical Digit and Racing Bib Model Training Performance Graph

Figure 7. Inference Result From Both Models

Table 5. RBN Recognition and Evaluation

| Architecture   | Recall | Precision | F1 score |
|----------------|--------|-----------|----------|
| 864x864        | 0.78   | 0.79      | 0.79     |
| 1280x1280      | 0.74   | 0.75      | 0.75     |
3.4. Discussion

As a result, we could achieve 83.0% Precision, 81.5% recall 82.2% F1 Score on RBN recognition with our proposed method. However our model cannot be implemented on other racing bib design and font dataset as we used different dataset during training and testing of detection model, thus we cant compare the performance of our model apple-to-apple with other RBN recognition model as the use of a different dataset. As stated in the introduction, we tried to eliminate the use of CRNN and OCR as number recognition techniques in RBN recognition. With our model using only YOLOv3 and RBN bounding box arrangement algorithm, we achieved a higher F1 score performance in RBN recognition compared to the previous works which got 66% on using multimodal [7], 69% on using CNN + CRNN [8], and precision 64.25% on using YOLOv3 + CRNN [10]. The attempt to tweak the network size into two different sizes shows a minor increase in precision and decrease in recall, this is due to the decrease of chance false positive being detected within the racing bib and numerical digit detection model as the network size increased. Based on the result, our proposed method still suffers from the same problem as previous works did which is incorrect RBN prediction result such as “1745” which it should be “61745” as shown in Figure 9.

Figure 8. RBN Prediction Result From Our RBN Recognition Method

Table 5. Our method evaluation result

| Network Size | Precision | Recall | F1 Score |
|--------------|-----------|--------|----------|
| 840x840      | 81.9%     | 81.6%  | 81.7%    |
| 1280x1280    | 83.0%     | 81.5%  | 82.2%    |

Figure 9. An Example of Incorrect RBN Prediction

Undetected racing bib region also occurs during racing bib model inference, this resulted in detected numerical digit within the undetected racing bib region is not being processed bib in the RBN bounding box.
REFERENCES

[1] J. Scheerder, K. Breedveld, and J. Borgers, “Who Is Doing a Run with the Running Boom?,” in Running across Europe, 2015.

[2] H. Liu and B. Bhanu, “Pose-guided R-CNN for jersey number recognition in sports,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2019, doi: 10.1109/CVPRW.2019.00301.

[3] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multi-digit number recognition from street view imagery using deep convolutional neural networks,” in 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings, 2014.

[4] G. Li, S. Xu, X. Liu, L. Li, and C. Wang, “Jersey number recognition with semi-supervised spatial transformer network,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2018, doi: 10.1109/CVPRW.2018.00231.

[5] I. Ben-Ami, T. Basha, and S. Avidan, “Racing bib number recognition,” in BMVC 2012 - Electronic Proceedings of the British Machine Vision Conference 2012, 2012, doi: 10.5244/C.26.19.

[6] S. Roy, P. Shivakumara, P. Mondal, R. Raghavendra, U. Pal, and T. Lu, “A new multi-modal technique for bib number/text detection in natural images,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2015, vol. 9314, pp. 483–494, doi: 10.1007/978-3-319-24075-6_47.

[7] P. Shivakumara, R. Raghavendra, L. Qin, K. B. Raja, T. Lu, and U. Pal, “A new multi-modal approach to bib number/text detection and recognition in Marathon images,” Pattern Recognit., 2017, doi: 10.1016/j.patcog.2016.08.021.

[8] A. Apap and D. Seychell, “Marathon bib number recognition using deep learning,” in International Symposium on Image and Signal Processing and Analysis, ISPA, 2019, doi: 10.1109/ISPA.2019.8868768.

[9] B. Shi, X. Bai, and C. Yao, “An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition,” IEEE Trans. Pattern Anal. Mach. Intell., 2017, doi: 10.1109/TPAMI.2016.2646371.

[10] Y. C. Wong, L. I. Choi, R. S. S. Singh, H. Zhang, and A. R. Syafeeza, “Deep learning-based racing bib number detection and recognition,” Jordanian J. Comput. Inf. Technol., 2019, doi: 10.5455/jjcit.71-1562747728.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, doi: 10.1109/CVPR.2016.91.

[12] R. Girshick, J. Donahue, T. Darrell, U. C. Berkeley, and J. Malik, “R-CNN,” 1311.2524v5, 2014, doi: 10.1109/CVPR.2014.81.

[13] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017, doi: 10.1109/CVPR.2017.690.

[14] J. Redmon and A. Farhadi, “Yolo3,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2017, doi: 10.1109/CVPR.2017.690.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” Int. J. Comput. Vis., 2010, doi: 10.1007/s11263-009-0275-4.

[16] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019, doi: 10.1109/CVPR.2019.00075.