A Fine-Tuned YoloV5 Deep Learning Approach for Real-Time House Number Detection

Murat Taşyürek[1] **and Celal Öztürk**[2]

[1]Department of Computer Engineering, Kayseri University, Kayseri, Turkey
[2]Department of Computer Engineering, Erciyes University, Kayseri, Turkey

Corresponding author:
Murat Taşyürek[1]

Email address: murattasyurek@kayseri.edu.tr

ABSTRACT

Detection of small objects in natural scene images is a complicated problem due to the blur and depth found in the images. Detecting house numbers from the natural scene images in real-time is a computer vision problem. On the other hand, convolutional neural network (CNN) based deep learning methods have been widely used in object detection in recent years. In this study, firstly, a classical CNN-based approach is used to detect house numbers with locations from natural images in real-time. Faster R-CNN, MobileNet, YOLOV4, YOLOV5 and YOLOV7, among the commonly used CNN models, models were applied. However, satisfactory results could not be obtained due to the small size and variable depth of the door plate objects. A new approach using the fine-tuning technique is proposed to improve the performance of CNN-based deep learning models. Experimental evaluations were made on real data from Kayseri province. Classic Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 methods yield f1 scores of 0.763, 0.677, 0.880, 0.943 and 0.842, respectively. The proposed fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5, and YOLO V7 approaches achieved f1 scores of 0.845, 0.775, 0.932, 0.972 and 0.889, respectively. Thanks to the proposed fine-tuned approach, the f1 score of all models has increased. Regarding the run time of the methods, classic Faster R-CNN detects 0.603 sn, while fine-tuned Faster R-CNN detects 0.633 sn. Classic MobileNet detects 0.046 sn, while fine-tuned MobileNet detects 0.048 sn. Classic YOLO V4 and fine-tuned YOLO V4 detect 0.235 and 0.240 sn, respectively. Classic YOLO V5 and fine-tuned YOLO V5 detect 0.015 sn, and classic YOLO V7 and fine-tuned YOLO V7 detect objects in 0.009 sn. While the YOLO V7 model was the fastest running model with an average running time of 0.009 seconds, the proposed fine-tuned YOLO V5 approach achieved the highest performance with an f1 score of 0.972.

INTRODUCTION

The quality of geographic information systems (GIS) developed to store, analyze, and display spatial data depends on the accuracy of the data it contains (Cooperative and Collins, 1988; Tasyurek, 2022). The quality and readability of the image data sets used in creating an address map are very important (Ulutaş Karakol et al., 2021). Detecting house numbers from natural scene images containing spatial location information (Visin et al., 2015) and processing them with their locations accelerates the address infrastructure (Öztürkçü and Leyla, 2020). The natural scene image is the raw form of the momentary image of nature or the environment. The most common source used to obtain house numbers from images is Google Street images, which consist of coordinated panoramic images taken with 360° (Vandeviver, 2014). Door numbers from street views detecting and reading (Asif et al., 2021) is a computer vision problem (Zuo et al., 2019; Kulikajevas et al., 2021) that falls under the category of natural scene text recognition (Fischler and Firschein, 2014). Character recognition in images in natural scenes is a complicated problem due to the variability of light, background clutter, severe blur, inconsistent resolution, and many other factors. In addition to these properties, there are deteriorations in the characters and numbers in street view photographs with the effect of natural events.

In recent years, deep learning method has been widely used in image classification, object tracking,

pose estimation, text detection and recognition, visual salience detection, action recognition, and scene tagging (Alzubaidi et al., 2021; Bashir et al., 2021; Pal and Pradhan, 2022; Atasever et al., 2022). Deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks are the methods frequently used in deep learning (Garcia-Garcia et al., 2018). Among these methods, it has been found that convolutional neural networks (CNN) show high performance in image classification (Khan et al., 2020; Dönmez, 2022). The CNN model takes its name from the linear mathematical operation between matrices called convolution (O'Shea and Nash, 2015; Maass and Storey, 2021; Terzi and Azginoglu, 2021). The CNN model consists of a multi-layer structure including a convolutional layer, non-linear layer, pool layer and fully connected layer (Albawi et al., 2017).

Identifying characters and numbers from natural images is one of the classification problems in computer vision. In the literature, studies on detecting house numbers from street images with CNN models show very high performance in image classification (Goodfellow et al., 2013; Visin et al., 2015). In this study, classic CNN models such as Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 were applied in a CNN-based system designed to detect house numbers from images obtained in real-time with spatial location. However, sufficiently successful results could not be obtained, especially due to the small and variable depths of the house number objects in the image.

Training on more datasets is a solution to improve the performance of CNN-based deep learning models, but collecting large amounts of data imposes a time and financial burden. On the other hand, a fine-tuning method has been widely used in recent years to improve the performance of deep learning models (Amisse et al., 2021). Fine-tuning is to increase the model's success by making adjustments on deep learning models (Subramanian et al., 2022). One of the commonly used fine-tuning methods in the literature is to remove the last layer of the model, the softmax layer, and replace it with its classifier layer. Another fine-tuning method is to change the value of the parameters, also called hyperparameters, which affect the performance of the models (Öztürk et al., 2023). On the other hand, freezing the layers' weights in the previously trained model is a common fine-tuning practice. In this study, a new fine-tuning technique is proposed to improve the performance of deep learning-based models. The proposed technique includes updating the softmax layer, multi-scale training (Rath, 2022) and performing the training process with a low learning rate (Yu, 2016) rate. The proposed approach's main contributions within this study's scope are presented below.

Contributions

- A new CNN-based approach is proposed for house number detection with the location in real-time.

- The proposed approach has been tested on real natural scene images taken from Kayseri Metropolitan Municipality.

- In the proposed approach, the performances of Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models, which are widely used as CNN models, are examined.

- A fair evaluation was made by comparing Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models designed in different structures on a single platform (PyTorch).

- A new fine-tuning technique is proposed to improve the performance of classical CNN-based deep learning models in house number detection.

- The proposed fine-tuned YOLO V5 approach can detect house numbers from natural scene images with a high f1 score of 0.972 in an average of 0.015 seconds.

Scope and Outline

- Hyperparameter optimization to improve accuracy performance in house door number detection is out of the scope of this study.

The rest of this paper is organized as follows: Section 2 presents the related work. Section 3 gives about basic concept with CNN models. Section 4 presents the proposed approach. In Section 5, experimental evaluations are presented. Section 6 presents conclusions and future works.

RELATED WORKS

CNN method, which is one of the deep learning methods, has been widely used in different fields such as computer networks (Gu et al., 2018), image detection (Chauhan et al., 2018) and disease classification (Lu et al., 2021) in recent years. The image classification process with CNN can be done by creating a custom CNN structure or using CNN models with a fixed structure. As an example of custom CNN models, Wei et al. (2018) proposed a new technique using the CNN model to effectively and robustly detect multifaceted text in natural scene images. He et al. (2016) presented a system for scene text detection by proposing the Text-CNN model, which focuses on extracting text-related regions and features from image components. Jia et al. (2018) proposed a CNN-based approach to detect handwritten texts from images of whiteboards and handwritten notes. Garg et al. (2019) stated that they detected high performance in MNIST dataset by creating an efficient CNN model with multiple convolutions, ReLu and Pooling layers. Athira et al. (2022) suggested using a special CNN model for character classification in container identity detection and recognition.

The model developed by LeCun et al. (1999) as LeNet-5 for handwriting and machine-printed character recognition in the 1990s is considered the first successful application of Convolutional Networks. LeNet-5, a 7-level convolutional network, was developed to recognize handwritten numbers in 32x32 pixel grayscale input images. When it is desired to analyze higher resolution images with the LeNet-5 method, the level of the convolutional network is insufficient (Paul and Singh, 2015). AlexNet (Krizhevsky et al., 2012) (ImageNet) developed in 2012 produced more successful results than all previous CNN models. CNN models have been continuously developed to achieve higher accuracy and faster results (Alom et al., 2019). ZFNet (Fu et al., 2018) in 2013, GoogLeNet (Sam et al., 2019) and VGGNet (Simonyan and Zisserman, 2014) in 2014, ResNet (Gao et al., 2021) in 2015 were developed.

The developed CNN models are successful in feature extraction and classification in single-object image analysis but not sufficiently successful in multi-object image analysis. For this reason, Girshick et al. (2014) proposed the R-CNN method to overcome the multi-object problem. The R-CNN divides the image into approximately 2000 regions and searches within the region with CNN. The computational cost of the R-CNN method is high in terms of time. Girshick (2015) developed the Fast R-CNN method that works faster to eliminate the problem of R-CNN running slow. Julca-Aguilar and Hirata (2018) suggested using the Faster R-CNN algorithm as a general method for detecting symbols in handwritten graphics. Nagaoka et al. (2017) developed a model for text detection based on Faster R-CNN that can be trained in an end-to-end coherent manner. R-CNN algorithms use regions to localize the object within the image. The CNN-based YOLO (You Only Look Once) method, which examines parts of the image likely to contain the object rather than thinning the region, was developed by Redmon et al. (2016). The YOLO method has produced more successful results than many object detection methods used in real-time object tracking. For example, Li et al. (2018) used the YOLO model to detect steel strip surface defects in real-time. Rahman et al. (2020) suggested using the YOLO model for an automatic reverse vehicle detection system from road safety camera images. Pei and Zhu (2020) developed the YOLO model for real-time text detection and recognition. Taşyürek and Öztürk (2022) proposed a two-stage deep learning model using only the YOLO V4 model to detect house numbers from natural scene images. However, in the approach, real-time object detection was not performed, and the location data of the objects on the earth was not captured.

In addition, YOLO models have been constantly being improved. YOLO V5 was developed by Jocher et al. (2020). Kim et al. (2022) examined the object detection and classification performances of YOLO V4 and V5 models on the Maritime Dataset and showed that the YOLO V5 model showed superior object detection performance compared to the YOLO V4 model. On the other hand, Taşyürek (2023) has proposed a new approach called ODRP, which uses map-based transformation and deep learning models to detect street signs with their real locations on Earth from EXIF format data. In the proposed ODRP approach, the YOLO V5 model outperformed the YOLO V6 model in object detection.

In recent years, the fine-tuning technique has been widely used to increase the classification and segmentation performance of CNN-based deep learning methods (Pham, 2021; Xu et al., 2021). For example, Kaya and Gürsoy (2023) proposed a transfer learning-based deep learning approach with fine-tuning mechanisms to classify COVID-19 from chest X-ray images. They used the MobileNet V2 version as the CNN model, and the proposed model achieved an average accuracy of 97.61% with fine-tuning. Akshatha et al. (2022) examined the performance of the Faster R-CNN and SSD models fine-tuned for human detection from air thermal images. After fine-tuning, the mAP metric of the Faster R-CNN model

increased by 10%, while the mAP metric of the SSD model increased by 3.5%. Salman et al. (2022) proposed the fine-tuned YOLO model for an automated prostate cancer grading and diagnosis system. Thanks to the fine-tuning technique they suggested, the proposed method achieved 97% detection and classification success.

In this study, firstly, classic Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models were applied for a CNN-based system that detects house numbers with spatial locations from natural images in real-time. However, satisfactory results could not be obtained due to the small size and variable depth of the house plate object in the raw images. A new approach using the fine-tuning technique is proposed to improve the object detection performance of the CNN-based system.

BASIC CONCEPTS

Deep Learning has become a prevalent subset of machine learning because of its high classification performance across many data types(Raschka and Mirjalili, 2017; Zhang et al., 2017). One of the most impactful deep learning methods for image classification is the convolutional neural network (CNN) method. CNN is a deep learning algorithm generally used in image processing and takes images as input (Wang et al., 2017; Nasir et al., 2021). This algorithm, which captures and classifies the visual features with different operations, has been widely used in recent years(Barzekar and Yu, 2022). CNN-based Faster R-CNN, MobileNet and YOLO models used in this study are presented below.

R-CNN

R-CNN architecture detects classes of objects in images and their bounding boxes. In the R-CNN model, features that are candidates to be objects in the visual are determined by selective search. In selective search, which works with the hierarchy from small to large, small regions are determined first. Then, two similar regions are merged, and a new larger region emerges. This process continues recursively. In each iteration, more significant regions occur, and the objects in the image are clustered. After about 2000 regions are determined, each is individually entered into a CNN model, and their classes and bounding boxes are estimated. Specific region candidates for R-CNN are determined by selective search. These district candidates each enter the CNN networks as inputs. At the end of this region nomination process, approximately 2000 regions emerge, and 2000 CNN networks are used for these 2000 regions. The object class in SVM models and bounding boxes in regression models are determined using the features obtained from CNN networks. The R-CNN model has the following disadvantages:

- Each image needs to classify 2000 region suggestions. Therefore, it takes a lot of time to train the network.

- It also requires a lot of Disk space to store the feature map of the region recommendation.

The backbone of R-CNN models can be changed. AlexNet, VGG 16 or ResNet 50 can be selected as the backbone of the R-CNN. The default backbone of the R-CNN model developed in the PyTorch is ResNet 50 (Rath, 2021). The ResNet 50 model consists of 50 layers, including 1 MaxPool layer, 1 average pool layer and 48 convolutional layers.

R-CNN architecture (Girshick et al., 2014) has been developed since it cannot be easily detected with CNN in images with multiple objects. Ross Girshick developed the Fast R-CNN method, which works faster, to eliminate the problem of R-CNN running slow(Girshick, 2015). The fast R-CNN model takes all image and region suggestions as input in feed-forward CNN architecture. Also, the Fast R-CNN model combines the ConvNet, Role Pool, and classification layer of the R-CNN model in a single structure. This eliminates the need to store a feature map and saves disk space. It also uses the softmax layer instead of the SVM method in region recommendation classification, which has proven faster and produces better accuracy than the SVM method.

On the other hand, Faster R-CNN were introduced by Ren et al. (2015). In the Fast R-CNN model, the bottleneck is the selective search method for the R-CNN architecture. The region proposal network (RPN) is used instead of the selective search method in the Faster R-CNN model. In this model, the image is first transferred to the backbone network. This backbone network creates a convolutional feature map. This feature map is forwarded to the region recommendation network (RPN). Returns object candidates along with candidate scores objectness using the RPN feature map. Then, The ROI pooling layer resizes the regions to a fixed size. Finally, it feeds the regions to the fully connected layer for classification.

Regarding computational cost, Faster R-CNN is faster than R-CNN and Fast R-CNN (Ren et al., 2015). In addition, the Faster R-CNN model achieves better mean average precision value than R-CNN and Fast R-CNN models. This study used the Faster R-CNN model, a more successful method than R-CNN and Fast R-CNN methods.

MobileNet

MobileNet is a CNN-based deep learning model designed for mobile and embedded computer vision applications. The MomileNet (V1) was introduced by Howard and Zhu (2017). MobileNet is a simple and efficient deep learning model (Michele et al., 2019). It is widely used in real-time applications due to its low computational cost (Verma and Srivastava, 2022; Edel and Kapustin, 2022).

The basis of MobileNetV1 is deeply detachable convolutional structures to create lightweight deep neural networks. Deep convolution applies a single filter to each input channel in this release. Point convolution then uses the 1×1 convolution to combine the outputs of the deep convolution. A standard convolution filters the inputs and combines them into a new set of outputs in a single step. MobileNet has 28 layers. The model takes an image with dimensions $224 \times 224 \times 3$ as input. On the other hand, the MobileNet model continued to be developed by adding new features. In 2018, the MobileNet V2 was introduced by Sandler et al. (2018). The MobileNet V2 has been developed to overcome the bottlenecks in the intermediate inputs and outputs of the V1 model. Thanks to the improvements made, the Mobilenet V2 model has achieved faster training and better accuracy than the V1 model. On the other hand, the following model version, MobileNet V3, is widely used in the image analysis capabilities of many popular mobile applications.

In this study, the MobileNet V3 version was used because it stands out with its low computation cost in real-time systems.

YOLO

The YOLO approach takes its name from the words "You Only Look Once", which means you only look once (Redmon et al., 2016). The YOLO approach can predict at a glance what the objects in the image are and where they are (Sarkar and Gunturi, 2021). With the YOLO method, high accuracy can be achieved most of the time, and it also works in real-time, which has been frequently preferred in recent years due to its capabilities (Du, 2018). The algorithm "looks only once" at the image in the sense that it only requires one forward propagation pass through the neural network to make the prediction. After non-maximum suppression (which allows the object detection algorithm to detect each object only once), it outputs the recognized objects along with the bounding boxes. With YOLO, a single CNN simultaneously predicts multiple bounding boxes and the class probabilities for those boxes. YOLO can work on full images and directly optimize detection performance.

The YOLO algorithm performs these operations using the CNN model. The architectural structure of the YOLO model consists of 24 convolutional layers, followed by 2 fully connected layers (Redmon et al., 2016). The architecture uses the $7 \times 7(S \times S)$ grid structure. It takes $448 \times 448 \times 3$ images as input data. Architecture produces output in size $7 \times 7 \times 30$.

The YOLO approach has been continually developed. YOLO V1 architecture, the first version developed by Redmon et al. (2016), because the output layer is a fully linked layer, the YOLO training model only supports the exact input resolution during testing as the training image. To eliminate the shortcomings of the YOLO V1 version and continue its success, the more accurate, faster, and more powerful YOLO v2 architecture, which can recognize 9,000 objects, was introduced by Redmon and Farhadi (2017). Developed by Redmon and Farhadi (2018) in 2018, the YOLO V3 model is more complex than the previous model. The YOLO V3 architecture allows changing the size of the model's structure, allowing the speed and accuracy of the model to be changed. In 2020, the YOLO V4 version was introduced by Bochkovskiy et al. (2020) as an object recognition method with optimum speed and accuracy. A practical and powerful object detection model is proposed in the YOLO V4 release. YOLO V4 aims to find the best balance between input network resolution, number of convolutional layers, number of parameters, and number of layer outputs (filters).

On the other hand, Jocher developed the YOLO V5 model in 2020 (Jocher et al., 2020). Unlike the V4 model, the YOLO V5 model is run in Pytorch. Studies (Jiang et al., 2022; Fang et al., 2021) have shown that the YOLO V5 model produces more successful estimations and less computational cost than the V4 model. While previous versions of YOLO were written in the C programming language, YOLOv5 was written in the Python programming language. Thus, installing and integrating YOLOv5 into IoT

254 devices has become more accessible. YOLOv5 is only 27 MB, while YOLOv4 using Darknet is 244
255 MB. Compared to YOLOv4's Darknet community, YOLOv5's Pytorch community is more populated,
256 indicating that more contributions will be made and more significant potential for future growth. It is
257 challenging to accurately compare the performance of the YOLO V4 and YOLO V5 methods, which
258 use two different languages and frameworks. But over time, under the same conditions, the YOLO V5
259 method has proven itself by showing higher performance than the YOLO V4 method and receiving more
260 support from the computer vision community.

261 In addition, a new version of the YOLO model, the YOLO V7, was released in 2022 (Wang et al.,
262 2022). YOLO V7 uses anchor boxes to detect a broader range of object shapes and sizes than previous
263 versions. YOLO V7 also has a higher resolution than previous versions. While other models process
264 images at 416×416 resolution by default, the YOLO V7 model processes images at 608×608 by default.
265 Thanks to this default image size, the YOLO V7 model detects smaller objects and gives it higher accuracy
266 overall (Kundu, 2023).

267 In this study, the performances of the YOLO V4, V5 and V7 models were examined.

PROPOSED CNN BASED DEEP LEARNING APPROACH FOR HOUSE NUM-
BER DETECTION WITH SPATIAL LOCATION IN REAL-TIME

270 The quality of geographic information systems developed to store, analyze and display spatial data depends
271 on the accuracy of the data it contains. Since address data has been created using natural scene images in
272 recent years, the legibility of the house number characters in the images is very important (Taşyürek and
273 Öztürk, 2022). In addition, detecting house numbers from natural images containing location information
274 and processing them with their locations accelerates the address infrastructure. The instance natural scene
image containing the number plate is shown in Figure 1.

Figure 1. Example House Number

275
276 The plate with blue background in Figure 1 is the door number plate. There is the letter 5A on the
277 plate. Address plates are produced in the same standard color and format. The images of the Kayseri
278 used within the scope of this study, obtained in real-time, also contain the location information of the
279 point where the photo was taken. When the house number in these images is detected, the location of
280 the house number is automatically detected. The location of the point where the photo was taken is
281 accepted as the location of the house number. Determining the door number, the essential component of
282 the address infrastructure, and its correct positioning on the map is essential for vital services such as
283 education, hospital and pharmacy. However, when door numbers are determined from natural images
284 with classical methods, errors occur due to eye strain or typing on the keyboard incorrectly. In this study,

285 a new CNN-based approach is proposed to overcome these problems and to detect house numbers with
286 their locations in real-time. The flowchart of the proposed system is presented in Figure 2.
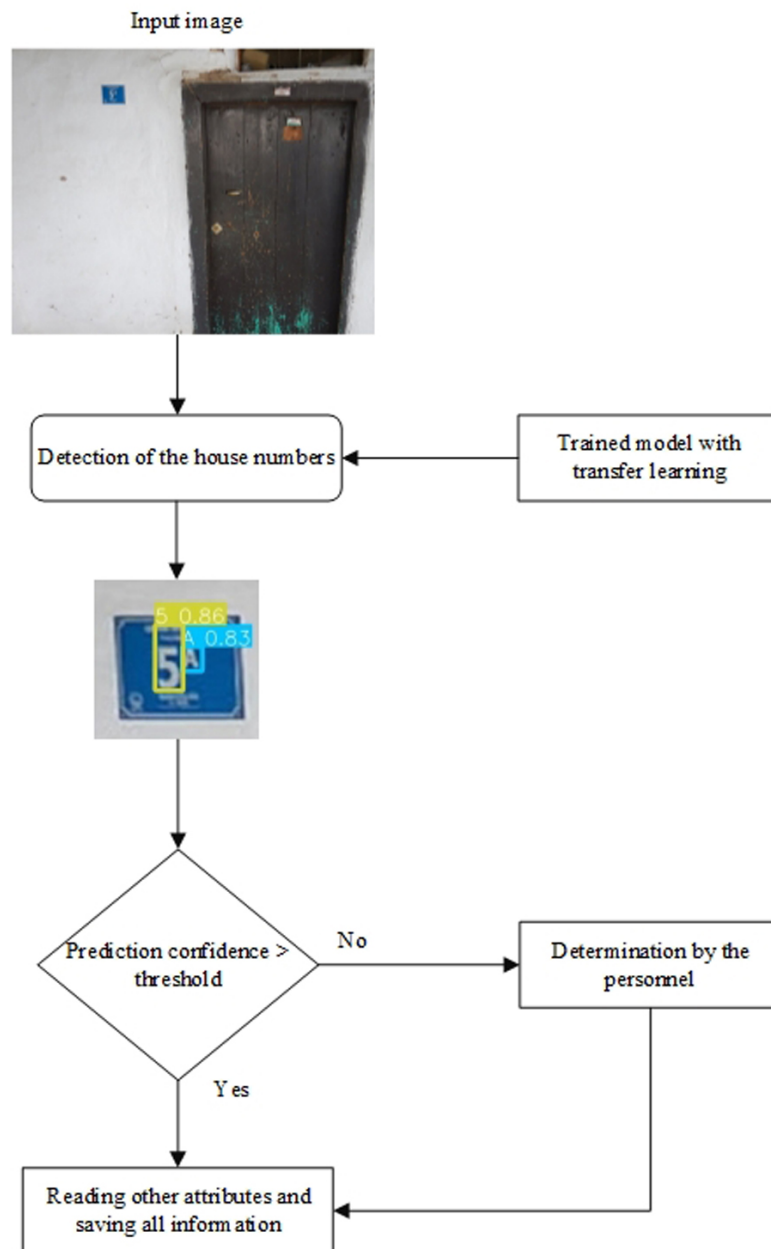
Figure 2. Deep learning-based system architecture

287 As seen in Figure 2, firstly, the model must be trained in CNN-based object detection systems. In
288 order to increase the performance of the proposed system, the transfer learning technique was used within
289 the scope of this study. The transfer learning method is frequently used during the training process of
290 CNN-based models (Zhuang et al., 2020). Transfer learning model can be expressed as transferring the
291 previously trained and high-performance weights to the new model to be trained (Weiss et al., 2016).
292 This way, models that show higher success and learn faster with less training data are obtained using
293 previous knowledge. In the system presented in Figure 2, the picture containing the house numbers with
294 spatial location is input for door number determination. After the picture is given to the system, the door
295 number in the picture is estimated with the CNN-based deep learning method.Suppose the confidence
296 score of the door number estimated by the deep learning method is above the threshold value. In that case,

the system reads the estimated door number, location information in the picture and other attributes and saves this information to the database. '5' was estimated with a confidence score of 0.86, and 'A' was estimated with a confidence score of 0.83 in the sample plate detection presented in Figure 2. Suppose the confidence score of the door number estimated by the deep learning method is below the threshold value (0.5 was selected for this study). In that case, the system ensures that the user enters the door number, reads the other attribute information from the picture, and saves the data to the database.

Within the scope of this study, firstly, Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models, which are widely used as CNN-based deep learning models, were applied in the proposed system. The computational costs of YOLO-based models were low, as expected for real-time systems. However, all models could not detect the house numbers and characters sufficiently due to the depth and resolution found in natural images. In order to overcome these problems and improve the object detection performance of CNN-based models, the fine-tuning technique, which has been widely used in recent years, was proposed. Fine-tuning is expressed as increasing the model's success by adjusting deep learning models. There are many fine-tuning types. However, the common and easy-to-use ones can be listed as changing the last layer, reducing the learning rate and multi-resolution training (Yu, 2016; Rath, 2022). In this study, these three processes were applied. As the first fine-tuning process, the softmax layer of the previously trained network (transferred with the transfer learning technique) was truncated, and a new softmax layer with 14 classes was added instead. As the second fine-tuning process, the learning rate of the models was reduced, and the models were trained with a learning rate of 0.001. As the final fine-tuning process, the models are trained in multi-resolution. For multi-resolution training, images are automatically resized by +-50% during training with the –multi-scale parameter in YOLO V4, V5 and V7 models. However, this feature is not available on Faster R-CNN and MobilNet models. For Faster R-CNN and MobileNet models, images were resized before fine-tuned training. Results of the classical Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models and fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models in the proposed approach in following section has been presented.

EXPERIMENTAL EVALUATIONS

In this section, the experimental performances of Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 methods are compared for both classical and proposed fine-tuned learning. In the experimental evaluations, the answers to the following questions were examined.

- What are the door number detection performances of approaches using classical CNN models?

- What are the door number detection performances of approaches using fine-tuned CNN models?

- What are the run-times of the approaches?

Data Sets

In this study, natural scene images containing the house numbers with the location were used. 2664 images were used as training data, and 626 images were used as validation data. To examine the performance of the methods, real images containing 3627 door numbers and location information in Kayseri province, Sarioglan-Ciftlik district, were used. Detailed information about the images used for testing purposes is presented in Table 1.

The images presented in Table 1 also include locations of door numbers. In other words, while there is a house number on the image, its attributes contain the information at which location the image is taken. The location data in the attribute information is positioned on the map as shown in Figure 3 using the open-source leaflet library and the open street map base.

The spatial distribution of the dataset is shown in the map image presented in Figure 3. Since the settlements are more in the town centre, the blue dots showing the location of the house number are more concentrated in the settlements.

Model settings and performance metrics

The YOLO V5 (Jocher et al., 2020) and YOLO V7 (Wang et al., 2022) models were developed using the PyTorch library. Faster R-CNN (Rath, 2021), MobileNet (Wang, 2019) and YOLO V4 (Yiu, 2021) versions developed with Pytorch architecture were used to compare the methods under equal conditions.

Table 1. Dataset Summary

Dataset ID	Street Name	Count of Images	Size (Megabyte)
1	AHMET TATAR	54	24.4
2	BAĞLAR BAŞI	270	122
3	BELEDİYE	212	95.9
4	CECELİ	60	27.4
5	COŞKUN	34	15.3
6	CUMHURİYET MEYDANI	94	42.7
7	ESEN	130	59.0
8	FATİH	100	45.5
9	GEMEREK	254	115
10	GÖKTEPE	278	125
11	GÜLOVA	160	72.6
12	HACILAR	44	16.2
13	KANUNİ	72	32.5
14	KAYSERİ	1082	488
15	KOYUN ABDAL	58	26
16	KÖMEVİRAN	119	53.7
17	MEHMET AKİF ERSOY	52	23.6
18	MEHMET KOÇER	68	30.4
19	MESCİT	48	21.6
20	NİLÜFER	30	13.6
21	NURİ DEĞERLİ	100	45.0
22	ORHAN GAZİ	28	12.6
23	ORUÇ REİS	24	10.8
24	PAPATYA	44	20
25	SAĞLIK	212	95.4
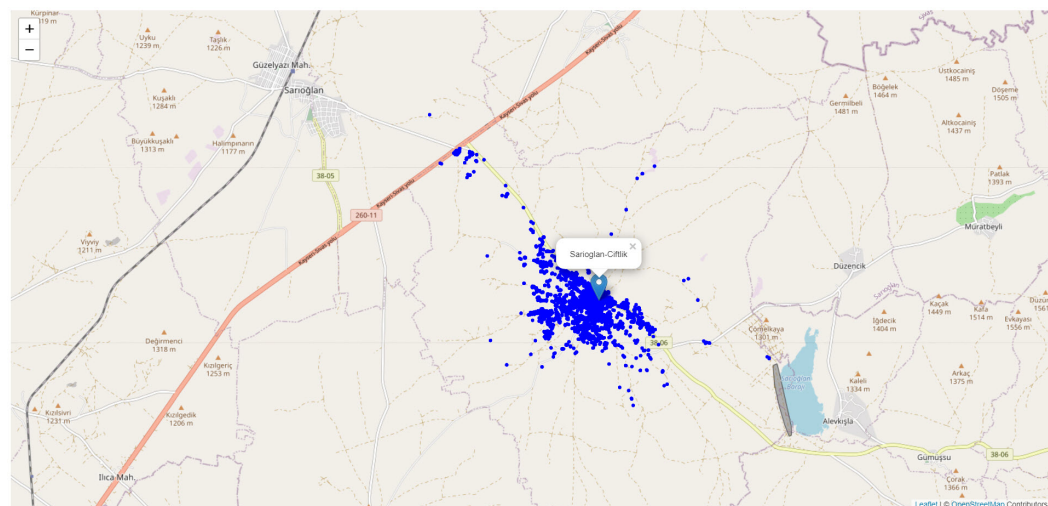
Figure 3. Spatial locations of the dataset

All methods were trained by setting the epoch value to 300. Experimental studies have been analyzed using Python 3.9 version on the computer with Intel Core i7-9700 3.0 GHz, 32 GB RAM and 12GB NVIDIA. The loss value produced by deep learning models is used to examine the success of the training (Chung et al., 2020). The decrease in the Loss value during the training and approaching zero indicates the success of the training. The training graphs of the classical and fine-tuned Faster R-CNN, MobileNet, YOLO V4,

YOLO V5 and YOLO V7 model are presented in Figure 4 (a) and (b), respectively. In Figure 4, as the epoch value increases, the decrease in the loss value and gradually approaching zero shows the success of the training process. When the classical CNN models were examined in terms of training times, the training time of the Faster R-CNN model was 77.065 hours, the MobileNet training time was 11.241 hours, the YOLO V4 training time was 32.133 hours, the YOLO V5 training time was 12.133 hours and the YOLO V7 training time was 7.891 hours. As the fine-tuned CNN models were examined in terms of training times, the training times of Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 were164.123, 23.413, 63.234, 25.149 and 15.982 hours, respectively. The YOLO V5 performed a better training loss value at the same epoch value than the other models for both classical and fine-tuned training, as presented in Figure 4. The loss value decreases for a long time since the fine-tuning process reduces the learning rate. In addition, the multi-resolution training increased the training times of the models.
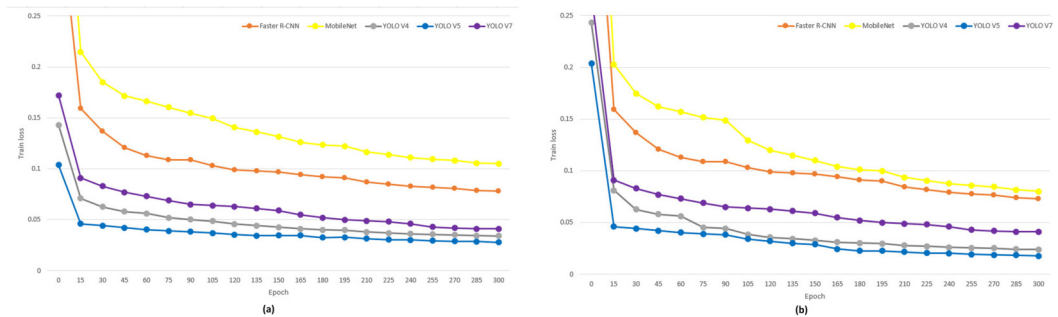
Figure 4. Train loss of classical and fine-tuned CNN models

The labelling (annotation) process was done with the LabelImg (Talin, 2018) program. The door numbers were analyzed as 14 classes. '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '/', 'A', 'B' and 'C' were defined and labelled for classes. The YOLO V4, V5 and V7 use .txt files as labelling files, while the Faster R-CNN and MobileNet use .xml files. A single labelling process was made, and the same labels were used for all models by selecting the export formats .txt and .xml.

Performance metrics are used to examine the performance of deep learning models (Bacchi et al., 2020; Teplitzky et al., 2020). These metrics are accuracy, precision, recall and F 1 score. However, in order to calculate these values, true positive (TP), true negative (TN), false positive (FP) and false negative (FN) values must be calculated. If there is an object and detection, this value is accepted as TP. The number of door numbers that the proposed approach detects correctly is the TP value. If there is no object and no detection, this situation is evaluated as TN. If there is a detection by the model even though there is no object, it is expressed as FP. An object count that cannot be detected by the deep learning model even though it is in the image is referred to as FN.

Accuracy shows how successful the model is in all classes in general and is calculated with Equation 1.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{1}$$

Precision represents the ratio of the number of correctly classified positive samples to the total number of positive samples and is calculated with Equation 2.

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

Recall measures the model's ability to detect positive samples and is calculated with Equation 3.

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

F1 score is one of the most widely used metrics. F1 score is obtained as a result of using almost all metrics. F1 score is calculated with Equation 4.

$$F_1 - score = \frac{2 \times Precision \times Recall}{Precision+Recall} \tag{4}$$

378

379

Experiments

In this section, experimental comparisons of the approach designed with the CNN-based deep learning model for real-time house number detection are presented. First, the door number detection performances of the classic Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models were investigated. Then, the port number detection performances of fine-tuned classic Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models were investigated. Finally, the real times of the proposed approaches are presented.

Door number detection performances of classical CNN models

Within the scope of this experiment, the performance of detecting house numbers in natural scene images of the classic Faster R-CNN, MobileNet, YOLO V4, YOLO V4 and V7 models was compared. Test operations were carried out on 3627 images. These images contain 20722 characters (numbers) in total. In order to better examine the performance of CNN models, all benchmark metrics obtained are presented in Table 2.

Table 2. All Metrics of CNN Models

CNN Model	TP	TN	FP	FN	Acc.	Pre.	Rec.	F1 S.
Classic Faster R-CNN	14849	0	3339	5873	0.617	0.816	0.717	0.763
Classic MobileNet	12532	0	3752	8190	0.512	0.770	0.605	0.677
Classic YOLO V4	17875	0	2020	2847	0.786	0.898	0.863	0.880
Classic YOLO V5	**19302**	0	**924**	**1420**	**0.892**	**0.954**	**0.931**	**0.943**
Classic YOLO V7	17078	0	2780	3644	0.727	0.860	0.824	0.842

When the metrics presented in Table 2 are examined, the classical Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 approaches were able to detect 14849, 12532, 17875, 19302 and 17078 as TP, respectively. The TN value was 0 in all models because there was no image without the door number in the dataset. Regarding models' FP values, Faster R-CNN has 3339, MobileNet has 3752, YOLO V4 has 2020, YOLO V5 has 924, and YOLO V7 has 2780 FP values. On the other hand, Faster R-CNN, MobiletNet, YOLO V4, YOLO V5, and YOLO V7 models have FN values of 5873, 8190, 2847,1420 and 3644, respectively. While TP, FN, FP and FN metrics are used to calculate accuracy, precision and recall values, precision and recall values are used to obtain the F1 score value. For this reason, it is necessary to look at the F1 score values to examine the performance of the models. The classic YOLOV5 model has the highest f1 score, with 0.943. The MobileNet model, on the other hand, has the lowest f1 score with 0.677. Faster R-CNN, YOLO V4, and YOLO V7 models achieved f1 scores of 0.763, 0.880 and 0.842, respectively. However, although the YOLO V5 model achieves a better f1 score than other models, the f1 score of other models is lower. In order to better analyze the metric values obtained by classical Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7, house number detection of methods on an image was examined and presented in Figure 5 (a), (b), (c), (d), and (e), respectively.

When Figure 5 is examined, there is the original version of the image in Figure 5 (b). This is because the MobileNet model cannot detect any digits and characters in Figure 5 (b). Due to such situations, the performance metric of the MobileNet model was lower. As seen in Figure 5 (a), the Faster R-CNN model detected the number '6' with a confidence score of 0.73 but failed to detect the character 'A'. The detection result of the YOLO V4 model is presented in Figure 5 (c). The YOLO V4 model could not detect the 'A' character but detected the '6' with a confidence score of 0.66. As seen in Figure 5 (d), the YOLO V5 model could not detect the 'A' character but detected the '6' with a confidence score of 0.86. The result of detecting the house number of the YOLO V7 model is presented in Figure 5 (e). The YOLO V7 model could not detect the 'A' character like other models, but it did detect the '6'. When Figure 5 is examined, the model that detects '5' with the highest confidence score is YOLO V5. Because it detects with such a high confidence score, the metric values of the YOLO V5 model are higher than the others. However, none of the classical CNN models could detect the 'A' character. In this study,

Figure 5. House number detections by classic CNN Models

the fine-tuning technique is proposed to detect undetectable characters, such as the 'A' character and to detect door numbers with higher performance rates. The results of the proposed fine-tuning technique are presented in the following experiment.

Door number detection performances of fine-tuned CNN models

In this experiment, the performance of fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V4 and V7 models to detect house numbers in natural scene images was compared. As shown in the previous section, classical CNN-based models could not detect house numbers in images with variable depths. The fine-tuning technique has been proposed to overcome these problems and to detect door numbers with higher performance rates. The success of the proposed method was examined on 3627 real images. All benchmark metrics showing the performance of the proposed fine-tuned CNN models are presented in Table 3.

When the metrics presented in Table 3 are examined, fine-tuned Faster R-CNN determined 17122,

Table 3. All Metrics of CNN Models with fine-tuning

CNN Model	TP	TN	FP	FN	Acc.	Pre.	Rec.	F1 S.
Fine-tuned Faster R-CNN	17122	0	2682	3600	0.732	0.865	0.826	0.845
Fine-tuned MobileNet	14977	0	2968	5745	0.632	0.835	0.723	0.775
Fine-tuned YOLO V4	19318	0	1401	1404	0.873	0.932	0.932	0.932
Fine-tuned YOLO V5	**20037**	0	**485**	**685**	**0.945**	**0.976**	**0.967**	**0.972**
Fine-tuned YOLO V7	18345	0	2224	2377	0.799	0.892	0.885	0.889

MobileNet determined 14977, YOLO V4 determined 19318, YOLO V5 determined 20037, and YOLO V7 18345 determined the door numbers as TP. Fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models increased their TP values 2273, 2445, 1443, 735 and 1267, respectively, compared to their classics. The fine-tuned MobileNet model increased the TP value at the highest rate, with a value of 2445. Since the classical MobileNet model has a very low TP compared to other models, the highest increase was observed in this model after fine-tuning. The lowest increase in TP values was observed in the fine-tuned YOLO V5 model. This is because the classic YOLO V5 model is also successful. On the other hand, the TN value of all fine-tuned models was 0. When fine-tuned CNN models are analyzed according to FP value, fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5, and YOLO V7 models have FP values of 2682, 2968, 1401, 485 and 2224, respectively. If the model finds a number or character, even though there is no number or character, it is considered FP. A low FP value or a decrease in this value compared to the previous model indicates the success of the recommended fine-tuning technique. Thanks to the proposed fine-tuning technique, these models reduced their FP values by 657, 784, 619, 439 and 556, respectively. In addition, these models decreased their FN values amount of 2273, 2445, 1443, 735 and 1267, respectively, thanks to the proposed method. When the fine-tuned models were examined according to their F1 score values, the order of performance was the same as the classical CNN models. Fine-tuned YOLO V5 has the highest f1 score with 0.972. The fine-tuned MobileNet model, on the other hand, has the lowest f1 score with 0.775. Fine-tuned Faster R-CNN, YOLO V4 and YOLO V7 models achieved f1 scores of 0.845, 0.932 and 0.889, respectively. Thanks to the proposed fine-tuning technique, all CNN models have increased the f1 score performance. In order to better analyze the performances of the proposed fine-tuned CNN models, the method's house number detection on the same image used in classical CNN models was examined. The detection results of the fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 are presented in Figure 6 (a), (b), (c), (d), and (e), respectively.

As the figure 6 is examined, all models except the fine-tuned MobileNet model detected the number '6' and the character 'A' correctly (TP). The fine-tuned MobileNet model caught only 6'. While the classical MobileNet model could not find any object in the same image, the fine-tuned MobileNet model could detect the number '6' thanks to the suggested fine-tuning technique. Fine-tuned Faster R-CNN, YOLO V4, YOLO V5 and YOLO V7 models detected the 'A' character, which they could not detect in their classical state, thanks to the fine-tuning technique. In the input image, the depth of the door plate is high. In other words, the character's size on the door sign is small. Due to variable depth, classical CNN-based models cannot detect the house number successfully enough. In the proposed fine-tuned technique, the models are trained in multi-resolution by changing the size of +-50. Thanks to this multi-resolution training, fine-tuned models can detect more successful house numbers in natural scene images with varying depths than classic CNN models. In addition, as with classical CNN models, the fine-tuned YOLO V5 model is the model that detects house numbers with the highest confidence score. Due to such successful detections, the performance of the fine-tuned YOLO V5 model is superior to other models.

Run Time of the approaches
In real-time object detection, the computational cost is as important as the estimation performance of the methods. For this reason, the object detection times of the classic Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models and the recommended fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7 models were investigated. The PyTorch version of Faster R-CNN (Rath, 2021), MobileNet (Wang, 2019), YOLO V4 (Yiu, 2021), YOLO V5 (Jocher et al., 2020) and YOLO

Figure 6. House number detections by CNN Models with fine-tuning

475 V7 (Wang et al., 2022) models were used to evaluate the models under equal conditions. Models were run
476 for 3627 images in the dataset. The total running times of models are presented in seconds in Figure 7.
477   As seen in Figure 7, the total run times of the classic Faster R-CNN, MobileNet, YOLO V4, YOLO
478 V5 and YOLO V7 are 2187, 167, 852, 54 and 33 seconds, respectively. The total working time of these
479 models with the fine-tuning technique is 2296, 174, 869, 55 and 33 seconds, respectively. The Fine-tuned
480 Faster R-CNN model has the highest calculation cost with 2296 seconds. Also, the runtime of the classic
481 Faster-RCNN model is higher than the MobileNet and YOLO models. On the other hand, the classic
482 YOLO V7 and fine-tuned YOLO V7 models have the lowest runtime. Classical CNN models detect
483 the house numbers for which an image is found in approximately 0.603, 0.046, 0.240, 0.015 and 0.009
484 seconds, respectively. Fine-tuning CNN models detect about 0.633, 0.048, 0.240, 0.015 and 0.009 seconds,
485 respectively. As a result of the fine-tuning process, the computational cost of the Faster R-CNN model
486 increased by only 0.030 seconds in object detection. The proposed fine-tuning technique added only
487 0.020 seconds to the MobileNet model. This extra computational cost to the YOLO V4 model is 0.005
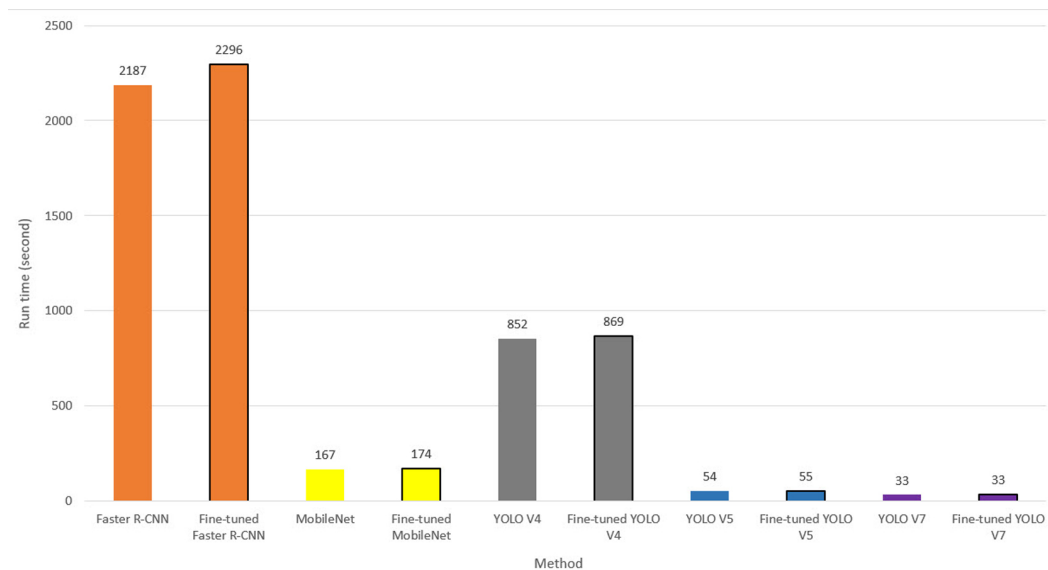
Figure 7. Run time of CNN models

seconds. The fine-tuning technique did not affect the average running time of the YOLO V5 and YOLO V7 models. In real-time door number detection, the YOLO V7 method works at least 66 times faster than the Faster R-CNN method, 5 times faster than the MobileNet model, 26 times faster than the YOLO V4, and at least 1.5 times faster than the YOLO V5 model. The YOLO V5 model operates approximately 40 times faster than the Faster R-CNN model, about 3 times faster than the MobileNet model, and about 15 times faster than the YOLO V4 model.

CONCLUSION

In this study, a CNN-based approach is proposed to detect house numbers with location information from natural images obtained in real-time. The performance of the proposed system has been tested on real images of Kayseri province. In the proposed method, classical Faster R-CNN, MobileNet, YOLO V4, YOLO V5 and YOLO V7, which are widely used as CNN models, were used. However, since the depths vary in natural scene images, sufficient successful results could not be obtained. In other words, the distance of the door plate in the image varies. In cases where the door plate is deep, the characters on the plate become challenging to read. The fine-tuning technique has been proposed to achieve higher performance in images with variable depths. The suggested fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5, and YOLO V7 methods obtained f1 scores of 0.845, 0.775, 0.932, 0.972 and 0.889, respectively. Thanks to the fine-tuning technique of these methods, the f1 score value increased by 0.082, 0.098, 0.052, 0.029 and 0.047, respectively, compared to the classical methods. Among the proposed approaches, the fine-tuned YOLO V5 achieved the highest performance with an f1 score of 0.972. On the other hand, regarding the run time of the proposed fine-tuned based methods, fine-tuned Faster R-CNN, MobileNet, YOLO V4, YOLO V5, and YOLO V7 detect objects about 0.633, 0.048, 0.240, 0.015 and 0.009 seconds, respectively. The YOLO V7 model is the model that makes the door number the fastest, with an average working time of 0.009 seconds.

In future studies, it is planned to perform hyperparameter optimization of CNN-based deep learning models with artificial intelligence optimization algorithms.

DECLARATIONS

Data availability

The datasets generated during and/or analyzed during the current study are not publicly available due to the owner of the data is Kayseri Metropolitan Municipality. The data set can be obtained by requesting it from the address https://kayseri.bel.tr/beyaz-masa.

518 **Competing interests**

519 The authors declare that they have no known competing financial interests or personal relationships that
520 could have appeared to influence the work reported in this paper.

521 **Authors' contributions**

522 **Murat Tasyurek:** Conceptualization, Formal analysis, Methodology, Resources, Software, Visualization,
523 Writing - original draft, Writing - review & editing.
524 **Celal Ozturk:** Conceptualization, Formal analysis, Methodology, Software, Supervision, Writing -
525 original draft, Writing - review & editing.

526 **Acknowledgements**

527 We want to thank Kayseri Metropolitan Municipality for sharing the raw images obtained with their
528 locations in real-time.

529 # REFERENCES

530 Akshatha, K., Karunakar, A. K., Shenoy, S. B., Pai, A. K., Nagaraj, N. H., and Rohatgi, S. S. (2022).
531   Human detection in aerial thermal images using faster r-cnn and ssd algorithms. *Electronics*, 11(7):1151.
532 Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network.
533   In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.
534 Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen,
535   B. C., Awwal, A. A., and Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and
536   architectures. *Electronics*, 8(3):292.
537 Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel,
538   M. A., Al-Amidie, M., and Farhan, L. (2021). Review of deep learning: Concepts, cnn architectures,
539   challenges, applications, future directions. *Journal of big Data*, 8(1):1–74.
540 Amisse, C., Jijón-Palma, M. E., and Centeno, J. A. S. (2021). Fine-tuning deep learning models for
541   pedestrian detection. *Boletim de Ciências Geodésicas*, 27.
542 Asif, M., Bin Ahmad, M., Mushtaq, S., Masood, K., Mahmood, T., and Ali Nagra, A. (2021). Long
543   multi-digit number recognition from images empowered by deep convolutional neural networks. *The
544   Computer Journal*.
545 Atasever, S., Azgınoglu, N., Terzı, D. S., and Terzı, R. (2022). A comprehensive survey of deep learning
546   research on medical image analysis with focus on transfer learning. *Clinical Imaging*.
547 Athira, M., Antony, J., Jacob, L., and Lakshmi, K. (2022). Container id detection and recognition. In *Soft
548   Computing and Signal Processing*, pages 173–183. Springer.
549 Bacchi, S., Zerner, T., Oakden-Rayner, L., Kleinig, T., Patel, S., and Jannes, J. (2020). Deep learning in
550   the prediction of ischaemic stroke thrombolysis functional outcomes: a pilot study. *Academic radiology*,
551   27(2):e19–e23.
552 Barzekar, H. and Yu, Z. (2022). C-net: A reliable convolutional neural network for biomedical image
553   classification. *Expert Systems with Applications*, 187:116003.
554 Bashir, S. M. A., Wang, Y., Khan, M., and Niu, Y. (2021). A comprehensive review of deep learning-based
555   single image super-resolution. *PeerJ Computer Science*, 7:e621.
556 Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object
557   detection. *arXiv preprint arXiv:2004.10934*.
558 Chauhan, R., Ghanshala, K. K., and Joshi, R. (2018). Convolutional neural network (cnn) for image
559   detection and recognition. In *2018 First International Conference on Secure Cyber Computing and
560   Communication (ICSCCC)*, pages 278–282. IEEE.
561 Chung, H., Ko, H., Lee, H., and Lee, J. (2020). Deep learning for heart rate estimation from reflectance
562   photoplethysmography with acceleration power spectrum and acceleration intensity. *IEEE Access*,
563   8:63390–63402.
564 Cooperative, G. and Collins, F. (1988). The unique qualities of a geographic information system: a
565   commentary. *Photogramm. Eng. Remote Sens*, 54:1547–1549.
566 Dönmez, E. (2022). Enhancing classification capacity of cnn models with deep feature selection and
567   fusion: A case study on maize seed classification. *Data & Knowledge Engineering*, 141:102075.
568 Du, J. (2018). Understanding of object detection based on cnn family and yolo. In *Journal of Physics:
569   Conference Series*, volume 1004, page 012029. IOP Publishing.

570 Edel, G. and Kapustin, V. (2022). Exploring of the mobilenet v1 and mobilenet v2 models on nvidia
571   jetson nano microcomputer. In *Journal of Physics: Conference Series*, volume 2291, page 012008. IOP
572   Publishing.

573 Fang, Y., Guo, X., Chen, K., Zhou, Z., and Ye, Q. (2021). Accurate and automated detection of surface
574   knots on sawn timbers using yolo-v5 model. *BioResources*, 16(3).

575 Fischler, M. A. and Firschein, O. (2014). *Readings in computer vision: issues, problem, principles, and
576   paradigms*. Elsevier.

577 Fu, L., Feng, Y., Majeed, Y., Zhang, X., Zhang, J., Karkee, M., and Zhang, Q. (2018). Kiwifruit detection
578   in field images using faster r-cnn with zfnet. *IFAC-PapersOnLine*, 51(17):45–50.

579 Gao, M., Qi, D., Mu, H., and Chen, J. (2021). A transfer residual neural network based on resnet-34 for
580   detection of wood knot defects. *Forests*, 12(2):212.

581 Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P., and Garcia-
582   Rodriguez, J. (2018). A survey on deep learning techniques for image and video semantic segmentation.
583   *Applied Soft Computing*, 70:41–65.

584 Garg, A., Gupta, D., Saxena, S., and Sahadev, P. P. (2019). Validation of random dataset using an efficient
585   cnn model trained on mnist handwritten dataset. In *2019 6th International Conference on Signal
586   Processing and Integrated Networks (SPIN)*, pages 602–606. IEEE.

587 Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*,
588   pages 1440–1448.

589 Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object
590   detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and
591   pattern recognition*, pages 580–587.

592 Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2013). Multi-digit number recognition
593   from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*.

594 Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al.
595   (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377.

596 He, T., Huang, W., Qiao, Y., and Yao, J. (2016). Text-attentional convolutional neural network for scene
597   text detection. *IEEE transactions on image processing*, 25(6):2529–2541.

598 Howard, A. G. and Zhu, M. (2017). Mobilenets: Open-source models for efficient on-device vision.
599   *Google AI blog*.

600 Jia, W., Zhong, Z., Sun, L., and Huo, Q. (2018). A cnn-based approach to detecting text from images of
601   whiteboards and handwritten notes. In *2018 16th International Conference on Frontiers in Handwriting
602   Recognition (ICFHR)*, pages 1–6. IEEE.

603 Jiang, P., Ergu, D., Liu, F., Cai, Y., and Ma, B. (2022). A review of yolo algorithm developments. *Procedia
604   Computer Science*, 199:1066–1073.

605 Jocher, G., Nishimura, K., Mineeva, T., and Vilariño, R. (2020). Yolov5.

606 Julca-Aguilar, F. D. and Hirata, N. S. (2018). Symbol detection in online handwritten graphics using
607   faster r-cnn. In *2018 13th IAPR international workshop on document analysis systems (DAS)*, pages
608   151–156. IEEE.

609 Kaya, Y. and Gürsoy, E. (2023). A mobilenet-based cnn model with a novel fine-tuning mechanism for
610   covid-19 infection detection. *Soft Computing*, pages 1–15.

611 Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep
612   convolutional neural networks. *Artificial intelligence review*, 53(8):5455–5516.

613 Kim, J.-H., Kim, N., Park, Y. W., and Won, C. S. (2022). Object detection and classification based on
614   yolo-v5 with improved maritime dataset. *Journal of Marine Science and Engineering*, 10(3):377.

615 Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional
616   neural networks. *Advances in neural information processing systems*, 25.

617 Kulikajevas, A., Maskeliunas, R., and Damaševičius, R. (2021). Detection of sitting posture using
618   hierarchical image composition and deep learning. *PeerJ computer science*, 7:e442.

619 Kundu, R. (2023). Yolo: Algorithm for object detection explained [+examples].

620 LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning.
621   In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

622 Li, J., Su, Z., Geng, J., and Yin, Y. (2018). Real-time detection of steel strip surface defects based on
623   improved yolo detection network. *IFAC-PapersOnLine*, 51(21):76–81.

624 Lu, J., Tan, L., and Jiang, H. (2021). Review on convolutional neural network (cnn) applied to plant leaf

625   disease classification. *Agriculture*, 11(8):707.

626   Maass, W. and Storey, V. C. (2021). Pairing conceptual modeling with machine learning. *Data &*
627   *Knowledge Engineering*, 134:101909.

628   Michele, A., Colin, V., and Santika, D. D. (2019). Mobilenet convolutional neural networks and support
629   vector machines for palmprint recognition. *Procedia Computer Science*, 157:110–117.

630   Nagaoka, Y., Miyazaki, T., Sugaya, Y., and Omachi, S. (2017). Text detection by faster r-cnn with multiple
631   region proposal networks. In *2017 14th IAPR international conference on document analysis and*
632   *recognition (ICDAR)*, volume 6, pages 15–20. IEEE.

633   Nasir, J. A., Khan, O. S., and Varlamis, I. (2021). Fake news detection: A hybrid cnn-rnn based deep
634   learning approach. *International Journal of Information Management Data Insights*, 1(1):100007.

635   O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint*
636   *arXiv:1511.08458*.

637   Öztürk, C., Taşyürek, M., and Türkdamar, M. U. (2023). Transfer learning and fine-tuned transfer learning
638   methods' effectiveness analyse in the cnn-based deep learning models. *Concurrency and Computation:*
639   *Practice and Experience*, 35(4):e7542.

640   Öztürkçü, T. and Leyla, S. (2020). Adres bilgi sistemlerinin oluşturulmasi. *İstanbul Ticaret Üniversitesi*
641   *Teknoloji ve Uygulamalı Bilimler Dergisi*, 2(2):25–34.

642   Pal, A. and Pradhan, M. (2022). Survey of fake news detection using machine intelligence approach.
643   *Data & Knowledge Engineering*, page 102118.

644   Paul, S. and Singh, L. (2015). A review on advances in deep learning. In *2015 IEEE Workshop on*
645   *Computational Intelligence: Theories, Applications and Future Directions (WCI)*, pages 1–6. IEEE.

646   Pei, S. and Zhu, M. (2020). Real-time text detection and recognition. *arXiv preprint arXiv:2011.00380*.

647   Pham, T. D. (2021). Classification of covid-19 chest x-rays with deep learning: new models or fine
648   tuning? *Health Information Science and Systems*, 9:1–11.

649   Rahman, Z., Ami, A. M., and Ullah, M. A. (2020). A real-time wrong-way vehicle detection based on
650   yolo and centroid tracking. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 916–920. IEEE.

651   Raschka, S. and Mirjalili, V. (2017). Python machine learning: Machine learning and deep learning with
652   python. *Scikit-Learn, and TensorFlow. Second edition ed*, 10:3175783.

653   Rath, S. (2022). Fine tuning yolov7 on custom dataset.

654   Rath, S. R. (2021). Custom object detection using pytorch faster rcnn.

655   Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time
656   object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*,
657   pages 779–788.

658   Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE*
659   *conference on computer vision and pattern recognition*, pages 7263–7271.

660   Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint*
661   *arXiv:1804.02767*.

662   Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with
663   region proposal networks. *Advances in neural information processing systems*, 28:91–99.

664   Salman, M., Cakar, G., Azimjonov, J., Kosem, M., and Cedimoglu, I. (2022). Automated prostate cancer
665   grading and diagnosis system using deep learning-based yolo object detection algorithm. *Expert*
666   *Systems with Applications*, 201:117148.

667   Sam, S. M., Kamardin, K., Sjarif, N. N. A., and Mohamed, N. (2019). Offline signature verification using
668   deep learning convolutional neural network (cnn) architectures googlenet inception-v1 and inception-v3.
669   *Procedia Computer Science*, 161:475–483.

670   Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted
671   residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern*
672   *recognition*, pages 4510–4520.

673   Sarkar, D. and Gunturi, S. K. (2021). Online health status monitoring of high voltage insulators using
674   deep learning model. *The Visual Computer*, pages 1–12.

675   Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image
676   recognition. *arXiv preprint arXiv:1409.1556*.

677   Subramanian, M., Shanmugavadivel, K., and Nandhini, P. (2022). On fine-tuning deep learning models
678   using transfer learning and hyper-parameters optimization for disease identification in maize leaves.
679   *Neural Computing and Applications*, 34(16):13951–13968.

680 Talin, T. (2018). Labelimg.

681 Tasyurek, M. (2022). A novel approach to improve the performance of the database storing big data with
682   time information. *Balkan Journal of Electrical and Computer Engineering*, 10(4):388–396.

683 Taşyürek, M. (2023). Odrp: a new approach for spatial street sign detection from exif using deep learning-
684   based object detection, distance estimation, rotation and projection system. *The Visual Computer*, pages
685   1–21.

686 Taşyürek, M. and Öztürk, C. (2022). Ddl: A new deep learning based approach for multiple house
687   numbers detection and clustering. *Journal of the Faculty of Engineering and Architecture of Gazi*
688   *University*, 37(2).

689 Teplitzky, B. A., McRoberts, M., and Ghanbari, H. (2020). Deep learning for comprehensive ecg
690   annotation. *Heart rhythm*, 17(5):881–888.

691 Terzi, R. and Azginoglu, N. (2021). A novel pipeline on medical object detection for bias reduction:
692   preliminary study for brain mri. In *2021 International Conference on INnovations in Intelligent SysTems*
693   *and Applications (INISTA)*, pages 1–6. IEEE.

694 Ulutaş Karakol, D., Ataman, S., and Cömert, Ç. (2021). Mekansal adres kayıt sistemi üzerine bir inceleme:
695   Ordu ili örneği.

696 Vandeviver, C. (2014). Applying google maps and google street view in criminological research. *Crime*
697   *Science*, 3(1):1–16.

698 Verma, A. and Srivastava, M. K. (2022). Real-time face mask detection using deep learning and mobilenet
699   v2. In *VLSI, Microwave and Wireless Technologies: Select Proceedings of ICVMWT 2021*, pages
700   297–305. Springer.

701 Visin, F., Kastner, K., Cho, K., Matteucci, M., Courville, A., and Bengio, Y. (2015). Renet: A recurrent
702   neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*.

703 Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). Yolov7: Trainable bag-of-freebies sets new
704   state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

705 Wang, D., Zhang, M., Li, Z., Li, J., Fu, M., Cui, Y., and Chen, X. (2017). Modulation format recognition
706   and osnr estimation using cnn-based deep learning. *IEEE Photonics Technology Letters*, 29(19):1667–
707   1670.

708 Wang, K. (2019). A pytorch implementation of mobilenetv3.

709 Wei, Y., Shen, W., Zeng, D., Ye, L., and Zhang, Z. (2018). Multi-oriented text detection from natural
710   scene images based on a cnn and pruning non-adjacent graph edges. *Signal Processing: Image*
711   *Communication*, 64:89–98.

712 Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big data*,
713   3(1):1–40.

714 Xu, Y., Souza, L. F., Silva, I. C., Marques, A. G., Silva, F. H., Nunes, V. X., Han, T., Jia, C., de Albu-
715   querque, V. H. C., and Rebouças Filho, P. P. (2021). A soft computing automatic based in deep learning
716   with use of fine-tuning for pulmonary segmentation in computed tomography images. *Applied Soft*
717   *Computing*, 112:107810.

718 Yiu, W. K. (2021). Pytorch yolov4.

719 Yu, F. (2016). A comprehensive guide to fine-tuning deep learning models in keras (part i).

720 Zhang, L., Tan, J., Han, D., and Zhu, H. (2017). From machine learning to deep learning: progress in
721   machine intelligence for rational drug discovery. *Drug discovery today*, 22(11):1680–1685.

722 Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive
723   survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.

724 Zuo, L.-Q., Sun, H.-M., Mao, Q.-C., Qi, R., and Jia, R.-S. (2019). Natural scene text recognition based on
725   encoder-decoder framework. *IEEE Access*, 7:62616–62623.