Mask Recognition Using Faster R-CNN

Abdul Bhutta
Electrical and Computer Engineering
Toronto Metropolitan University (TMU)
Toronto, Canada
abdul.bhutta@torontomu.ca

***Abstract*—In 2020, we learned that a pandemic could occur at any point in time. COVID-19 brought significant changes and impacted everyone's daily lives. The mandatory mask policy was implemented worldwide, allowing customers access to stores and university buildings. However, many were against it. Much funding went towards an additional workforce to implement the policy, and many resources were used. In this paper, A Faster R-CNN model will be implemented to detect if multiple people are wearing their mask or not or if it is incorrectly worn. Faster R-CNN is used for object detection and can be tuned to detect specific objects. There are many applications where Faster R-CNN has been implemented for various objects in related works. Examining and analyzing related work will help design a generalized model for optimal efficiency and detection. Lastly, the training and testing metrics will be presented with the model's accuracy in detecting face masks.**

I. Problem statement

The rapid increase of cases of COVID-19 throughout the world has been alarming and overwhelming. Hospitals during the pandemic were overrun with patients, and the wait time was extensive. Most businesses sought to employ multiple individuals to ensure everyone entering the building wore a mask. This research project aims to develop a model to automatically detect numerous people with masks on, off, or worn incorrectly with high accuracy recognition. The model is proposed to be used in various applications such as real-time detection or to inform businesses when an individual is breaching protocols.

II. Introduction

In 2020, a pandemic broke out worldwide and affected everyone's daily life, leading to a crisis. Most businesses and services were shut down, and a country-wide lockdown occurred. When the lockdowns were lifted, all businesses and services required you to wear a mask. However, the ban has been lifted today, but it must be stated that a pandemic can occur at any time. One primary application of this model can be the integration of real-time detection of masks when a customer enters a business. It detects whether a person is wearing a mask, and based on that, the person will be allowed entry into the building. The research will aim to create a model to detect multiple users with masks on from an image using a dataset provided by Kaggle. The model will implement a pre-trained Faster R-CNN (Region-Based Convolutional Neural Network) model to determine which of the three classes (Mask on, Mask off, and Mask worn incorrectly) each user belongs to in the image. The paper looks at other pre-trained models implemented for object detection and various ways to improve the faster R-CNN algorithm based on speed and object recognition.

Artificial Neural network-based segmentation mimics the human brain for decision-making and is a two-step process involving feature extraction and image segmentation using the neural network. Image segmentation is one of the primary processes for image processing, such as image compression or object detection, where the image is divided or separated into segments or regions. It is the foundation for many applications, such as content-based image retrieval, medical imaging, video surveillance and object detection. Segmentation can be classified into two types: local and global segmentation. Local segmentation mainly focuses on a specific area, and global focuses on segmenting the whole image. Region-based segments the image into regions by iteratively expanding from initial pixels (seeds) that can be selected manually or automatically. Region splitting and merging are combined to segment the image with the same characteristics and merge similar adjacent regions with similar attributes [1]. One of the popular techniques in image segmentation is verification codes (CAPTCHA) on the internet to help distinguish humans from AI. As the level of difficulty increases, the difficulty of verification also increases. Image segmentation plays a big role in detecting what each character is. Optical character recognition based on segmentation allows the use of limited computational resources and requires a shorter time to train a model. The segmentation is dependent on the creation complexity of the verification code. If the initial recognition of the code is low, the model needs to be retrained [2].

The main algorithms for object detection can be classified into two main categories: single-stage and two-stage detectors. The single-stage detector performs detection for bounding boxes and labels in a single pass through a neural network. An example of a single-stage detector is YOLO (You Only Look Once), which segments the image into a grid and performs position regression and classification. The two-stage detector involves two main stages: region-based proposals and object classification. An example of a two-stage detector is Faster R-CNN, which promoted the two-stage detectors [3]. Most object detection algorithms allow you to use a pre-trained model for feature extraction rather than create a CNN from scratch. Transfer learning is a technique applied in machine learning to use pre-trained weights and parameters for a particular task and modify it to a second task using a smaller dataset. Creating a CNN from scratch for a task such as image recognition is only sometimes ideal and feasible. A study on CNN transfer learning for image classification compared a CNN model (Inception-v3) and a model created from scratch [4]. Multiple tests were conducted: Test one was conducted to verify if the number of images for training had any impact on the accuracy, and test two was used to confirm that the increase in epochs also increases accuracy. Test 3 was used to verify if objects within an image impacted the accuracy. It was concluded that the number of images in a class and epochs significantly increased accuracy while the image content also played a role. Additionally, the accuracy of the pre-trained model (70%) outperformed the CNN from scratch (38%). The study concluded that a pre-trained model works well through transfer learning by adapting to a new dataset and achieving a high accuracy rate for image classification [4].

III. System Model

The model implemented in this research will be a pre-trained Faster R-CNN model because it provides high accuracy and fast speed in detecting an object within an image. To understand how Faster R-CNN works, we must explore its predecessors and how they functioned.

A. R-CNN

Region-Based Convolution Neural Network (R-CNN) is one of the early models that succeeded at object detection using region-based methods, a two-stage process. The first stage proposed regions containing an object, and the second stage classified the proposed regions. It uses a selective search algorithm to detect objects and divides the image into segments based on colour and surface [5]. It extracts 2000 region proposals from top to bottom in an image. Each selected region is 227 by 227 pixels and is sent to a Convolution Neural Network input. A fully connected layer (fc7) output is passed through a support vector machine for classification. The model is trained to identify the image's background and the objects. The main concerns for R-CNN are the computational cost and time consumption [6]. It takes 40 to 50 seconds for the model to predict, making it not feasible for real-time applications.

B. Fast R-CNN

Fast R-CNN was the next model created to battle the issues with the predecessor model to address the computational cost and time-consuming issues of R-CNN. It uses a single network to perform both region proposals and classification. In this model, a CNN is used only once to find the region of interest for each object. The model utilizes multiple steps simultaneously, such as extracting features, separating them into distinct classes and providing bounding boxes, making it nine times faster than R-CNN. However, model training still leads to high computational costs and can be time-consuming [5].

C. Faster R-CNN

Faster R-CNN is one of the state-of-the-art object detection algorithms on the market while taking advantage of GPU to analyze an image using a region-based method. It combines the two networks, Fast R-CNN and Region proposals, into one network to achieve high a ccuracy and speed. The RPN and object detection share the same convolution features and are trained together. It integrates the RPN directly into the architecture and allows for end-to-end training compared to Fast R-CNN, a two-stage process. The combination of the two networks acts as an attention in the model and is the primary unit to identify and detect objects in an image. The Faster R-CNN architecture consists of four main components: Backbone layer, Region Proposal Network (RPN), Region of Interest (ROI) pooling, and Classification. The backbone or convolution layer extracts features from the image using convolution layers with activation functions (RELU) and max pooling. The generated maps are sent to the next layer, RPN, to create bounding boxes that may contain an object with a likelihood score. Lastly, the region of interest locates specific locations in the image that may have the object and produces a fixed-sized feature map. These fixed-sized feature maps are sent to the output layer with an activation function, SoftMax, to produce probabilities for each class.

The pre-trained image classification model that will be used is ResNet50. The models extract features from the image which allow for object detection of a specific task (whether each person is wearing a mask, no mask, or worn incorrectly) through fine-tuning. The pre-trained model weights are adjusted through backpropagation to detect the new object within the image. Once fine-tuning is finished, it will become specialized in detecting a particular object. The model applies convolution layers and pooling while capturing low and mid-level features of an image. The generated feature maps are shared across the entire image and are passed to the region proposal network (RPN) that proposes bounding boxes or regions where the object may reside. The ROI (region of interest) pooling takes the shared features and aligns them with the generated region proposals while ensuring they are consistent for further analysis. The features are then passed down to the additional layers in the faster R-CNN model to classify between the three classes.

It is a difficult choice between YOLO and Faster R-CNN algorithms as both are current state-of-the-art models. An experiment was conducted using a dataset that included a person wearing a mask or not with annotations using both models. The objective was to create a bounding box on the image to determine whether a person was wearing a mask. In both models, all the layers were frozen except the last layer in training to produce a generalized model. It was concluded that the faster R-CNN had better accuracy in detecting the mask than YOLOv3. However, YOLOv3 was the ideal solution for real-life scenarios as the faster R-CNN required more computational resources and a longer time to generate predictions [7].

IV. Related Works

This section will explore four different Faster R-CNN experiments conducted by other researchers for object detection using various backbones and compare the results. In the first experiment, the primary task was to help drivers identify streetlights while driving. The dataset includes 3000 points divided into four classes: green, red, intermittent, and stop signs. The data was split into 80 percent for training and 20 percent for testing. The training involved capturing features from each frame, including anchors and ground truth regions. The model produced satisfactory results and could detect the signs even with a cheap camera. The results were divided into morning, afternoon, and evening. The results for the evenings compared to the other two were low, where intermittent light was detected only 30 percent of the time. The 4K camera produced much better results, where it could detect all the signs in the evening at an accuracy of 84% or greater. In comparison, morning and evening resulted in 89% or higher accuracy [9].

Experiment two aimed to detect and localize mask occluded faces by transfer learning using the Faster R-CNN model. It used a pre-trained CNN (ResNet 101) to extract facial features from an image and pass it to the ROI layer to recognize the distinct faces even when hidden or in a complex environment. It was concluded that the faster R-CNN-optimized model could detect masks in real-life images containing complex environments with various lighting and backgrounds. The results achieved a high accuracy while the running time was reduced by 11.46% compared to other state-of-the-art models [8].

The objective of the third experiment was to distinguish between healthy tomatoes and diseased tomatoes. It used three different backbones for feature extraction: VGG16, ResNet50, and ResNet 101. These three models are currently the primary model being used extensively for feature extraction with a faster R-CNN model. It involved a dataset of 1035 images divided into 60% for training, 20% for validation, and 20% for testing. The model's accuracy with the three different models was calculated based on the mean average precision, where the VGG16 model gave an accuracy of 88.28%, resnet50 was 89.53%, and resnet101 was around 90.87% [10].

In experiment four, the PASCAL VOC2007 dataset and Faster RCNN with various backbone layers. However, the results from each model were underwhelming, where the first model (Faster R-CNN + VGG16) achieved a 69.4% mAP, the second model (Faster R-CNN + ResNet101) around 72.5%, and the final model (Faster R-CNN + PVANET) was around 84.9% [6].

V. Methodology

A. Pre-Proccessing

The current dataset (853 images) provided by Kaggle is in an XML format for the annotations, and the images are in a PNG format. The model requires two different inputs: one is a list of tensors in the format of [Channel, Height, Width], and the other input is a dictionary (list) of target values for the bounding boxes. The bounding boxes must be in the format of [x1, y1, x2, y2], and the labels are in a tensor with integer values. The labels in this project are with mask (1), without mask (2), and mask worn incorrectly(3). When creating the labels for the ground truth box for the dictionary, the first label must start from 1 and not 0. The zero label is used for the background, and this was one of the first mistakes encountered throughout the project.

The model allows the images to be of different sizes but performs generalized normalization and resizing of the image. The normalization is performed at the pixel level using the standard values of ImageNet, a common practice in machine learning. The values for the mean are [0.485, 0.456, 0.406], and for standard deviation, it is [0.229, 0.224, 0.225]. Using bilinear interpolation, the resize transformation is applied to have a minimum size of 800 pixels and a maximum length of 1333 pixels. Bilinear interpolation is a technique that allows estimating the value of a pixel when it is resized and it doesn't fit the original pixel grid because the pixel values are continuous.

In most images, there are multiple people with masks or no masks, and each image has a variation in the total bounding boxes. Therefore, a collate function was required to convert the list of samples into batches of tensors suitable for training the network. A collating function is optional if each image's total number of bounding boxes is equal. Each image in the dataset is sent through a transform function to convert it into a PyTorch tensor with values ranging from 0 to 255, and normalization is performed to scale them from 0 to 1. Once the image has been converted and a dictionary of targets is created, the dataset will be split up for training and testing. The training dataset contained 95% of the images from the initial dataset, and the remaining were split into the testing dataset. The purpose of a higher dataset for training will allow the model to generalize well, as each image contains a diverse set of objects and complex backgrounds.

B. Model

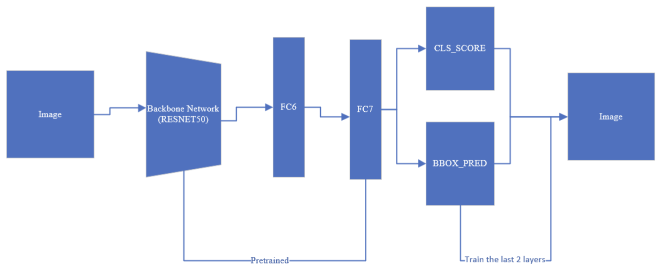

Fig. 1. Faster R-CNN model

The model implemented in this project will be a pre-trained Faster R-CNN model because it provides high accuracy and fast speed in detecting an object within an image. The Faster RCNN networks consist of multiple stages to detect objects.

The image goes through a backbone network to generate features for the objects. These generated feature maps are sent to a region proposal algorithm to create bounding boxes of possible locations of an object in the image. The proposed regions are sent to the ROI layer to create a fixed-size feature map. The feature maps are sent to the Fast R-CNN detector, which consists of two parts: Classification (`CLS_SCORE`) and Bounding Box Regression (`BBOX_PRED`). The classification part is used to classify the object in the image in 3 different classes: with mask, without mask, and mask worn incorrectly. The bounding box regression is used to indicate the coordinates of the object's bounding box. Figure 1 shows the model implemented in this project.

Initially, the model is downloaded from PyTorch with the pre-trained default values from the COCO dataset. The Resnet-50 will be the backbone of our network for feature extraction within all the images. All the parameters before layer seven will be frozen as they are good at extracting features at low and mid-level. The fully connected layer 7 (fc7) is connected to the CLS Score layer and the BBOX Prediction layer but will be removed to create two new layers with customized features and four classes (3 for object detection and 1 for background).

C. Training

The trained model uses stochastic gradient descent with the default values given in the documentation from Torchvision. It uses a learning rate of 0.005 and a weight decay 0.0005, which is a regularization technique to prevent overfitting. A momentum value of 0.09, a parameter to speed up gradient descent in the relevant direction, dampens oscillations to help converge faster and get past the local minima to reach the global minimum. A learning scheduler was used with the default values provided by TorchVision. The training ran for 50 epochs, and the results for the model are shown in the results sections under Tables 1 and 2. The training evaluation classes were extracted using the PyTorch references as it was easier to use methods already created for training one epoch or multiple epochs rather than implementing your own functions. All the training references are provided in the Jupyter Notebook.

The loss function in the model is generated by two different loss functions: regression loss and classification loss. The classification loss uses cross-entropy, and the bounding box uses regression loss, which is the smooth L1 loss. The total classification loss is computed using the difference between predicted and actual classes. The regression loss is used to get the difference between the predicted and actual bounding boxes. Lastly, the total loss is the sum of the classification and regression losses.

An anchor is used as a reference box to determine the bounding box for an object, and it is a box centred on a pixel with various fixed heights and widths at different scales. The `cls_ layer` outputs a vector of 2 values for each anchor. If the values are (`p0, p1`), then `p0` is the probability that the anchor contains an object, and `p1` is the probability that the anchor does not have an object. Each anchor is given a positive or negative score during training to determine whether it contains an object based on Intersection over Union (IoU). The IoU is the ratio of the area of overlap between the predicted and the actual bounding box. The range is between 0 and 1, where 0 is no overlap, and 1 is complete overlap. The anchors with an IoU greater than 0.7 are considered positive objectness score, and if the IoU is less than 0.3, a negative score is assigned, which means it is a background. The scores between 0.3 and 0.7 are discarded.

D. Testing

The testing dataset evaluates the model's performance and accuracy on unseen data. The testing of the model was complicated as the evaluation method from the TorchVision reference included Mask segmentation and required additional changes to the tensors as some of the methods have been deprecated. Unfortunately, it was too time-consuming to implement a mask segmentation format, and it was easier to import evaluation methods for various metrics using another utility object detection class [11].

VI. Results

A. Backbone Layer Results

The backbone layer (ResNet50) extracts features from the image using convolution layers with activation functions (RELU) and max pooling. In this section, we will look at the results of the various backbone layers, as shown in Figures 2 to 8. Each figure has the last five feature maps created at each layer with the tensor size (Filters, Height, Width) displayed on the top of each image and Figure 8 shows a summary of the output of all the layers with max pooling.

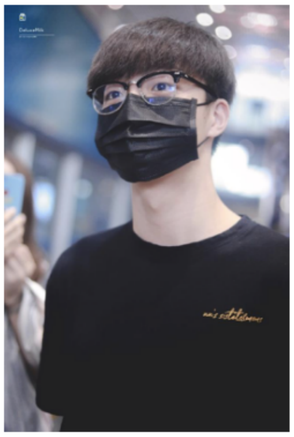

Fig. 2. Image sent to backbone layer

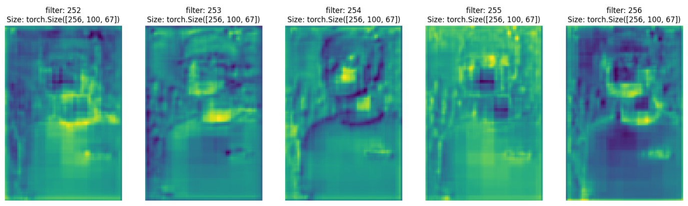

Fig. 3. Layer 1

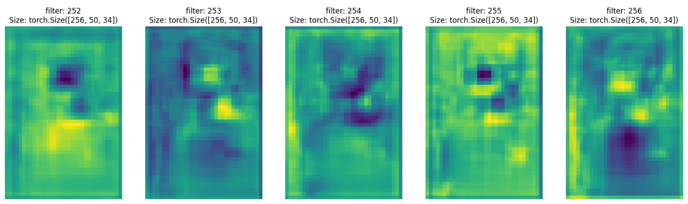

Fig. 4. Layer 2

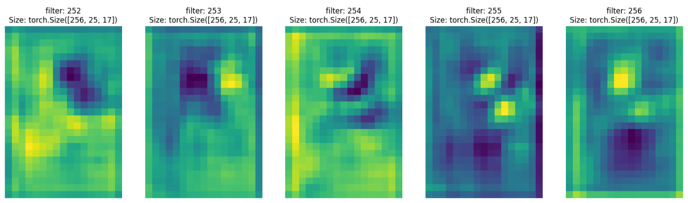

Fig. 5. Layer 3

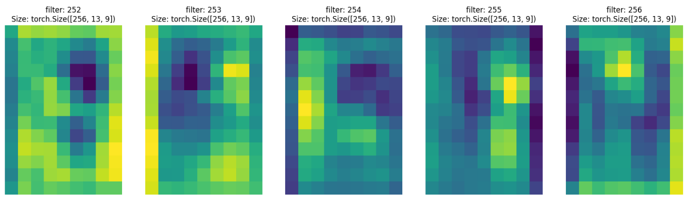

Fig. 6. Layer 4

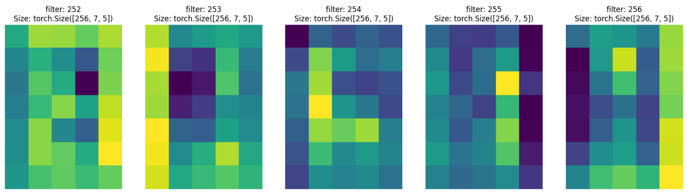

Fig. 7. Max Pooling

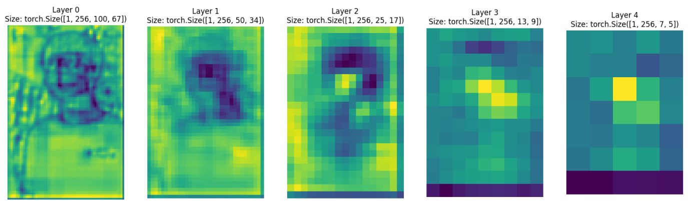

Fig. 8. Summary of all layers with max pooling

B. *Region Proposal Network Results*

The region proposal network (RPN) creates bounding boxes that may contain an object with a likelihood score. The RPN generated proposals for the test image are shown in Figure 9. The RPN generates 1000 bounding boxes for the image, which are then sent to the ROI layer depending on the score. The top 30 ROI proposals are shown in Figure 10.

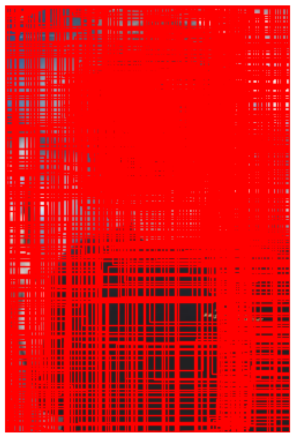

Fig. 9. RPN Generated for the image

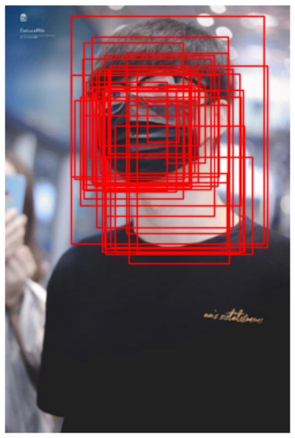

Fig. 10. Top 30 ROI Proposal Generated for the image

C. *Test Image Results*

The results of the model for a single test image are shown in Figure 11 and was able to detect the mask on the person's face with a confidence score of 1.00.

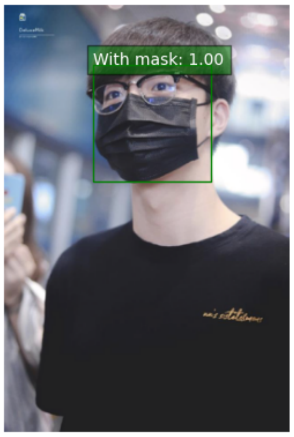

Fig. 11. Final Result with classification score and bounding box

D. Training Results and Testing Results

In this section, we will look at the model's training and testing results. The training results are shown in Tables 1 and 2, while the testing results are in Tables 3 and 4. The testing dataset is evaluated using the Mean Average Precision, Average Precision, Precision, Recall, and F1-score. The training results show the loss for the classifier, box regression, and RPN box regression. The training results show that the model converged as the loss value gradually decreased. The model was able to predict the correct classes for the test dataset with an accuracy of 86% and an mAP of 73%. It detected the mask on the person's face with an average precision of 0.9997, no mask with an average precision of 1.0000, and the mask worn incorrectly with an average precision of 0.2000. The reason for the low accuracy in detecting masks worn incorrectly is due to the lack of images in the dataset or the model was not trained enough to detect the mask worn incorrectly. The evaluation methods were taken from the utility object detection class and may be outdated leading to the low accuracy. When plotting the actual image with the predicted bounding box, the model was able to detect the mask worn incorrectly but the evaluation metrics did not reflect that. Due to time constraints, the evaluation metrics were not explored further.

TABLE I
TRAINING RESULT

Epoch	loss classifier	loss box reg	loss rpn box reg
10	0.0646	0.1220	0.0034
20	0.0540	0.1118	0.0025
30	0.0514	0.0959	0.0024
40	0.0427	0.0793	0.0016
50*	0.0424	0.0660	0.0035

*Batch 0 had best results

TABLE II
TRAINING RESULT

Epoch	Total Loss	Loss Objectness
10	0.1896	0.0009
20	0.1758	0.0009
30	0.1523	0.0004
40	0.1446	0.0002
50*	0.1121	0.0002

*Batch 0 had best results

TABLE III
TESTING RESULT

Metric	With mask	Without Mask	Mask Worn Incorrectly
AP	0.9997	1.0000	0.2000
Precision	0.9704	0.9394	0.5000
Recall	1.0000	1.0000	0.3333
F1-score	0.9535	0.8085	0.2000
mAP	0.7332	0.7332	0.7332

TABLE IV
TESTING RESULT

Total Correct Predictions	Total Incorrect Predictions	Accuracy
37	6	86%

VII. CONCLUSION

Image segmentation was discussed in this paper, including the various techniques involved. Region-based segmentation was introduced, and the main algorithm used for region-based is Faster R-CNN. Faster R-CNN and its predecessors were examined in detail, as how they are used for object detection. Furthermore, we explored various experiments performed by other researchers using Faster R-CNN, which showed that the Faster R-CNN model is one of the best models for object detection. In this paper, a generalized model was implemented using Faster R-CNN and ResNet50 as the backbone layer to detect whether a person is wearing a mask, not wearing a mask, or if the mask is worn incorrectly. The model produced a low loss value in training and a decent mAP for the testing dataset while verifying that the Faster R-CNN model was the correct choice. The accuracy of the test set for correctly predicting the correct classes was approximately 86%.

REFERENCES

[1] Kaur, D., Kaur, Y. (2014). International Journal of Computer Science and Mobile Computing. Various Image Segmentation Techniques: A Review, 3(5), 809–814. https://doi.org/10.47760/ijcsmc

[2] Zhou, Z., Wang, H., Shang, W., Zhang, L. (2018). Image segmentation algorithms based on Convolutional Neural Networks. 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS). https://doi.org/10.1109/icis.2018.8466419

[3] Hu, H., Duan, J., Bi, J., Fang, L. (2022). Target recognition technology based on improved faster RCNN. 2022 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI). https://doi.org/10.1109/iwecai55315.2022.00103

[4] Hussain, M., Bird, J. J., Faria, D. R. (2018). A study on CNN transfer learning for image classification. Advances in Intelligent Systems and Computing, 191–202. $https://doi.org/10.1007/978-3-319-97982-3_16$

[5] Hmidani, O., Ismaili Alaoui, E. M. (2022). A comprehensive survey of the R-CNN family for object detection. 2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet). https://doi.org/10.1109/commnet56067.2022.9993862

[6] Liu, B., Zhao, W., Sun, Q. (2017). Study of object detection based on faster R-CNN. 2017 Chinese Automation Congress (CAC). https://doi.org/10.1109/cac.2017.8243900

[7] Singh, S., Ahuja, U., Kumar, M., Kumar, Sachdeva, M. (2021). Face mask detection using yolov3 and faster R-CNN models: COVID-19 environment. Multimedia Tools and Applications, 80(13), 19753–19768. https://doi.org/10.1007/s11042-021-10711-8

[8] H N, S., H N, L., H N, P., Uma, B. (2021). Detection and localization of mask occluded faces by transfer learning using faster RCNN. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3835214

[9] Gavrilescu, R., Zet, C., Fosalau, C., Skoczylas, M., Cotovanu, D. (2018). Faster R-CNN:an approach to real-time object detection. 2018 International Conference and Exposition on Electrical And Power Engineering (EPE). https://doi.org/10.1109/icepe.2018.8559776

[10] Wang, Q., Qi, F. (2019). Tomato diseases recognition based on faster RCNN. 2019 10th International Conference on Information Technology in Medicine and Education (ITME). https://doi.org/10.1109/itme.2019.00176

[11] Pseudo-Lab, "Pseudo-lab/tutorial-book-utils: Data loader function for Tutorial book," GitHub, https://github.com/Pseudo-Lab/Tutorial-Book-Utils (accessed Dec. 15, 2023).

[12] "Torchvision Object Detection Finetuning Tutorial." TorchVision Object Detection Finetuning Tutorial - PyTorch Tutorials 2.2.0+cu121 Documentation, $pytorch.org/tutorials/intermediate/torchvision_tutorial.$ Accessed 12 Dec. 2023