Construction Site Worker Safety Detection

Udacity Machine Learning Nano-degree

Jun Xian - April 22, 2021

Introduction

Construction site is a place with plenty of hazards and risks. It is the place has much higher probability to cause incidences, and even harms to the people. According to Occupational Safety and Health Administration website, the US Bureau of Labor Statistics reported 5,333 workers died on the job in 2019 in US, and 1061 of worker fatalities in private industry were in construction[1]. This is why safety on a construction site brings much concerns, especially to the workers working there. To ensure safety, workers on site should wear proper body protection including hard helmets, reflective vests, masks, gloves, and other protection wears. Also, the safety managers should make sure every worker on site should wear protection and follow safety protocols.

The old way that safety managers usually do is patrolling the site and scanning every workers on site to check the worker safety. But this is not an easy job for the managers, especially when the site are huge and hundreds of workers are working there. Furthermore, they walking around on the site can be a potential risk for their safety.

There have been many new techniques introduced to construction industry to improve the construction site safety. For example, sensor-based localization technique uses sensors and GPS to locate and monitor workers activities on site and triggers warning when the hazard is detected, and camera-based localization technique puts many cameras at different spots of the site and requires a person to monitor the site through the control centre screens.[2] These techniques are useful but they are not intelligent enough, so the vision-based object detection technique has been introduced into this industry in the recent years.

Define the Problem

How to check worker safety on construction site in a safe and efficient way is the question safety managers frequently ask. They require a solution which is able to help them to detect workers who are not wearing safety protection on site.

Also, they require this type of solution to be fast, accurate, remote, and scalable, so safety managers can immediately get notice if there is any unprotected workers on site and they can react to these workers before any incidents occur. And the solution should be able to correctly

identify the workers who are not wearing safety protection so the safety managers will not waste time to deal with any false alarms. The safety managers would like to have a solution that can be controlled remotely so they do not need to always go to different areas of the construction sites. And lastly, the solution should be scalable so that it can be applied to a small construction site as well as a larger area.

Fortunately, this problem can be easily solved with the help of machine learning and deep learning technologies, specifically, the object detection technique in computer visioning. Similar to how human eyes works, object detection technique will capture an image and then analyze the content, the algorithm then can find out if there are particular objects in the image and try to classify what are these objects individually.

As safety managers, they go to the site, scan the workers and look for any unsafe scenes, then they will take actions to address these unsafe scenes. With this idea, a site safety detection system can use the object detection to check workers safety on site. The system firstly will capture an image of a construction site with workers on it, then the image is passed into the system for analyzing. It will find out if there are particular objects in the image and try to classify what are these objects. If the object is a person, then it will identify that the person wears safety protection or not. The system can evaluate the safety level of the site using some mathematical method and then it will highlight the person without protection wears, and return the result on the output image and warn the safety managers. Furthermore, with object detection technique improving and performing better and better, the system can actually meet the requirement of being fast, accurate, remote, and scalable.

Metrics of Object Detection

The key point in object detection is how to detect there is an object in the image, and this needs the help of a concept called anchor box. The logic of this anchor box is it distributes matrix of anchor dots evenly on the images, and for each anchor dot, the algorithm draws different shape of bounding boxes which use the anchor dots as the centre. And those bounding boxes are called anchor boxes. With all these different shape of anchor boxes, the algorithm will try to classify the image inside each of these boxes. In the training process of this algorithm, the anchor boxes which can successfully be classified (I.e. it can classify the image in the box with label) will be compared with the ground truth bounding boxes and

labels. The more the anchor boxes and labels are close to the ground truth bounding boxes and labels, the more accurate the model is getting. So the shape and the location of the anchor box vs the ground truth bounding box is the key parameter to evaluate the performance of the model. Article "20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics" summarized metrics used in machine learning, it mentioned PSNR, SSIM, and IoU are the popular metrics in Computer Vision[3]. And this project uses the IoU metric, or Intersection over union, which is the ratio of area of overlap to area of union,

IoU = area of overlap / area of union

Obviously, the IoU of an anchor box is closer to 1, the anchor box is closer to the ground truth bounding box. According to "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit" by Rafael Padilla, Wesley L. Passos, and others[4], a metrics can be restrictive on considering correct or incorrect detection with an IoU threshold, and lower threshold means it is more flexible for considering correct detection.

Another metrics to verify the performance of the system is to calculate the precision and recall, more commonly, the average precision (AP) and the average recall (AR). And if the values of these metrics are higher, it indicates the model performance is better.

Faster R-CNN

Object detection uses Convolutional Neural Networks (as known as CNN) architecture as its basic architecture. CNN was firstly applied to solve image recognition by Yann LeCun in 1989. CNN is "a specialized kind of neural network for processing data that has a known grid-like topology"[5]. It is "simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers"[5].

According to Ian Goodfellow in Deep Learning Book, "Convolution leverages three important ideas that can help improve a machine learning system: sparse interactions, parameter sharing and equivariant representations. Moreover, convolution provides a means for working with inputs of variable size" [5]. So CNN can compute image matrix in a faster and more accurate way compared to other neutral network. And that is why CNN is popularly used in image classification problem.

However, if just using CNN, this problem cannot be solved, because an image of a construction site will not always contain a fixed number of objects. To accurately detect variant objects on an image and detect what is the object, Region-based CNN (R-CNN) is used. R-CNN uses the idea to separate an entire image into many regions, and then it uses CNN to classify the image region by region. Up to today, R-CNN has evolved to many advanced version. A few of them can process the detection in a very fast way. One version is called Faster R-CNN, and the other one is called YOLO (You Only Look Once). Both of them are the perfect algorithms for this problem. For this project, Faster R-CNN could be a good fit, because it is fast. Even though it is not as fast as YOLO algorithm which can reach real-time level, Faster R-CNN process time is usually within 5 seconds which can work well in safety detection application. Another reason to choose Faster R-CNN over YOLO is Faster R-CNN can classify small object in a higher accuracy compared to YOLO. This is important because the object inside an image can be very small if the construction site is large and the camera locations are limited.

Anther technique used in this project is transfer learning. "Handbook of Research on Machine Learning Applications" states, "Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned."[6] Because this project used a pre-trained model as the base model, the model training in this project is considered as a second related task to the pre-train model, and so the required of dataset can be smaller for the second training task.

Data Analysis

To train the Faster R-CNN model to classify a specific object that is not seen in the pre-trained model, a data set with this specific is required. In this project, this type of specific object is a person, particularly, a person with safety protection or not. Because there are variety of safety protection wears, a completed dataset with all types of protection wears can be too large to use in this project. For simplicity, this project focuses on detecting if a contraction worker wears a safety helmet, and so the dataset is just a set of people wearing safety helmets or not.

From www.kaggle.com website, a "Helmet_Dataset" dataset provided by Abhishek Annamraju can be downloaded. And the dataset has 3174 jpeg images with people and safety helmet which would be a perfect inputs for this project. But due to the model's transfer

learning ability mentioned in “Faster R-CNN” section, the required amount of the dataset can be less. In this project, 100 images picked from the 3174 images dataset are involved. And the 100 images are randomly selected from the dataset.

Firstly, the samples of such dataset are shown as below:

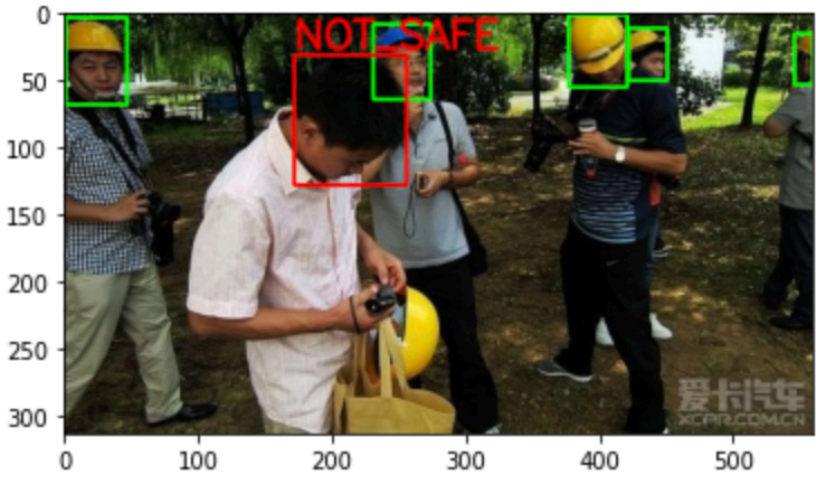
ID	Label
01319.jpg	101 185 127 225 white 123 178 164 225 white 18...
00797.jpg	148 92 175 124 red 196 109 218 138 none 273 11...
00892.jpg	355 241 421 323 red 495 245 575 347 red 568 21...
00537.jpg	350 22 449 158 red 252 31 339 137 red 42 45 20...
01122.jpg	92 71 194 179 blue

The selected 100 images are stored in new folder and the labels are transformed into an array of a dictionary which contains keys of “label”, “box”, and “score” as below:

id	old_id	image_path	boxes	number
helmet_001	00012.jpg	/data/helmet/helmet_001.jpg	[{'label': 'yellow', 'box': [1, 3, 47, 68], 's...	6
helmet_002	00018.jpg	/data/helmet/helmet_002.jpg	[{'label': 'white', 'box': [153, 270, 244, 371...	2
helmet_003	00023.jpg	/data/helmet/helmet_003.jpg	[{'label': 'none', 'box': [344, 77, 391, 132],...	5
helmet_004	00106.jpg	/data/helmet/helmet_004.jpg	[{'label': 'red', 'box': [161, 121, 200, 180],...	5
helmet_005	00113.jpg	/data/helmet/helmet_005.jpg	[{'label': 'white', 'box': [459, 77, 762, 499]...	4

Below is the visualization of one of the images with a bounding boxes on the object.

```
<matplotlib.image.AxesImage at 0x134f37df0>
```



The dataset has labels of workers with no helmet or with different colours of helmets. For this project, the labelling are simplified as 3 categories:

- label = 0 as background,
- label = 1 as person's head with no helmet
- label = 2 as person's head with any colour of helmet.

Additional, the 100 images dataset is divided into 3 sets:

- 60% images are in training dataset
- 20% images are in validation dataset
- 20% images are in testing dataset

Implementation of the System

1. SafetyDetectionDataSet

As the data is collected and preprocessed, the dataset can be packed into a Python class called "SafetyDetectionDataSet" which is holding "x" as a 2D or 3D matrix based on the number of colour channels present in the image, and "y" as a dictionary of: numpy array of "lablels", numpy array of "boxes", and numpy array of "scores". This "SafetyDetectionDataSet" convert the dataset into a ready-to-use format before training the model.

There is a requirements for "torchvision.models.detection.faster_rcnn.FasterRCNN" input[7]:

- The input to the model is expected to be a list of tensors, each of shape [C, H, W], one for each image, and should be in 0-1 range. Different images can have different sizes.

So the SafetyDetectionDataSet needs convert the image matrix to 3 tensors of shape [C, H, W], and then need to normalized the value of the matrix to be in 0-1 range.

2. PyTorch's Faster R-CNN Model

The next step of the implementation is to construct the model. This project uses one of the PyTorch's Faster R-CNN pre-trained model to build to the new safety helmet detection model.

The pre-trained model is a ResNet model (without Feature Pyramid Networks, i.e. FPN). The ResNet model can be constructed by calling the PyTorch function get_fasterRCNN_resnet() and specific parameters: backbone, anchor size, aspect ratios, min_size, max_size.

a. Transformations

PyTorch's Faster R-CNN implementation uses additional transformations: torchvision.models.detection.transform.GeneralizedRCNNTransform

The important arguments for this transformer are: in_size, max_size, image_mean, and image_std. They are generally the standard values of the dataset on which the Faster R-CNN model has been trained on.

PyTorch's Faster R-CNN is trained on ImageNet, this project uses the same normalization values for mean and std when in training: image_mean=[0.485, 0.456, 0.406], image_std=[0.229, 0.224, 0.225].

Also, the training images should be comparable in size, here it uses 1024 for both min_size and max_size. So the image is transformed into a size with maximum height or width with 1024px before it passes to the model training, validation, or testing.

b. Anchor boxes

Anchor boxes is the most important hyper-parameter for training an object detector model. Choose a right anchor sizes and aspect ratios parameters is the essential step to build the Faster R-CNN.

For a transformed image with max size = 1024, i.e. image is [3, 1024, 1024], the anchor sizes can be ((32, 64, 128, 256, 512),), and aspect ratios can be ((0.5, 1.0, 2.0),), according to Johannes Schmidt in "Train your own object detector with Faster- RCNN & PyTorch"[8].

c. Model building

With all parameters defined, a pre-trained faster RCNN ResNet34 model can be constructed. This model is pre-trained on ImageNet which it should be able to detect some general objects from an image. But it may not be able to detect some specific objects or the object it never sees in the pre-training. So that is reason it need to have the 100 images dataset in the new training process to help the model to recognize the specific objects of interest.

d. Training module

This project applies a high level API called PyTorch-lightning to do the training job, because PyTorch-lightning offers many functionalities, such as adding logging, metrics, early stopping, mixed precision training and other functionalities to the training. This saves a lot of time on coding how to train the model and other additional functions. PyTorch-lightning is an easy-to-use API and very similar to PyTorch.

The project creates a class called “SafetyDetectionTraining” class which needs 3 arguments:

- pre-trained Faster-RCNN model
- learning rate
- IoU threshold

The IoU threshold is an important value for the evaluation of the model. Because the threshold will determine what value of boxes IoU can set a prediction to be a True Positive (TP) (from Johannes Schmidt).

PyTorch Lightning module helps creating model training job. The PyTorch Lightning module will simplify the step to create a training script, so in this because the model handles most of the training coding.

In this project, SafetyDetectionTraining class extends PyTorch Lightning module, so this “SafetyDetectionTraining” class has defined all necessary information to do the training, validation and testing. And PyTorch Lightning module also offers callback functions to monitor the training, validating and testing steps and logs the information in a log folder. So it is convenient to use and debug during the process.

e. Training and testing

Next step is to train the model with the PyTorch lightning trainer. In this project it will take a little while because it is running 50 more epochs and is running on CPU. But if it is switches to AWS SageMaker and can have access to GPU then the training can be much faster.

After the training is done, the newly trained model should be able to detect the specific object of interest that is introduced in the dataset. To test the performance of the new model, it goes to test step with the test dataset.

f. Saving the best model

PyTorch lightning also gives an easy way to save the best model's hyper-parameters and parameters. Saving the model is an essential step, especially when the project is still in tuning stage and trying to find the best-fit model overall. The tuning stage needs to train the model with different sets of hyper-parameters and it generates a few set of model parameters. Saving this data helps to retrieve the trained model anytime if it is needed.

3. Model tuning

In this model, there is 3 hyper-parameters that needs to be adjusted.

a. Score threshold

The model will output all detected objects with labels, boxes locations and scores. The score of the each box gives the probability of the object is the true positive or not. In all these output boxes, some of the boxes are overlapping and some are not the object of interest (i.e. a person' head with or without helmet) as shown below . Usually most of the boxes are just the noise with low scores. So setting a threshold of the score can filter out these noise.

```
<matplotlib.image.AxesImage at 0x148d0b8e0>
```

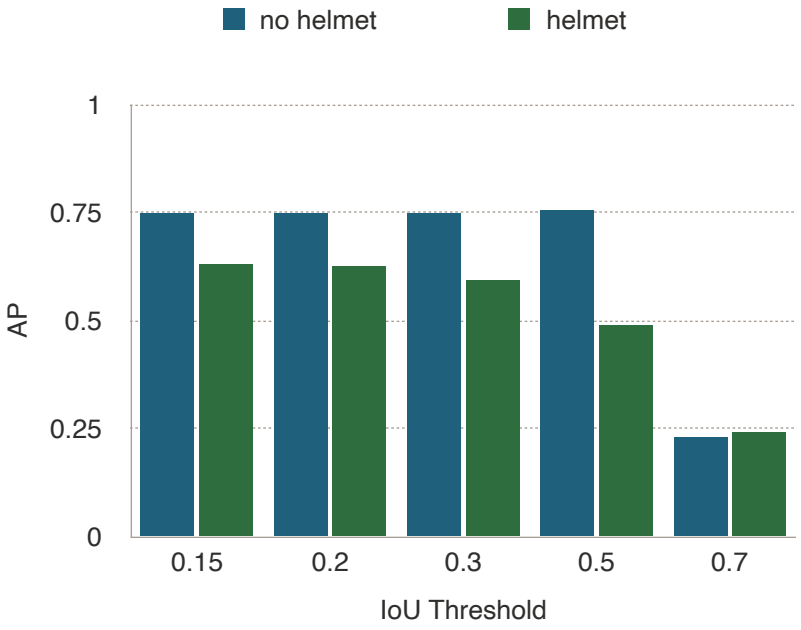
After a few trials with different values, score threshold = 0.6 makes the output only shows the high scoring bounding boxes, and the visualized image looks better than without any score threshold.

<matplotlib.image.AxesImage at 0x1489a1b50>

b. iou_threshold

As mentioned before, iou_threshold is the key metrics to evaluate the performance of the object detection model. In the tuning, the model is trained with the following a few iou_threshold values separately for 50 epochs:

The Benchmark of these iou_threshold vs their AP (Average Precision) is plotted as below:

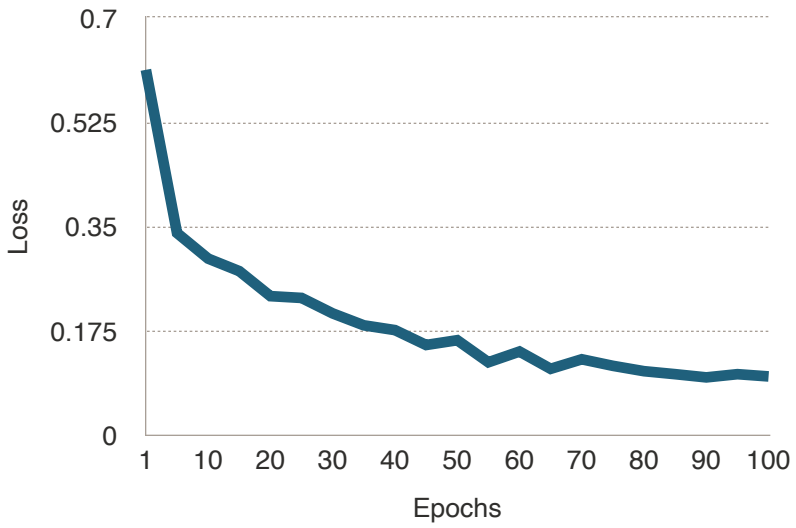


As it shows, AP for both classes is not good when iou_threshold = 0.7, and mAP for class 1 (no helmet) is almost consistent in the other thresholds, but AP for class 2 (has helmet) is

higher when the IoU threshold is lower. At iou_threshold = 0.2, the APs for both classes are at the highest value.

c. training_max_epochs

Below shows the Benchmark of epochs vs loss when IoU threshold is 0.2:



The graph shows that around epochs = 85, the loss function starts decreasing in a very small amount. The loss at epoch = 85 is 0.102, and at epoch = 100, it is 0.0982. The difference in these 15 epochs is 0.0038. So max epochs can be defined at 90.

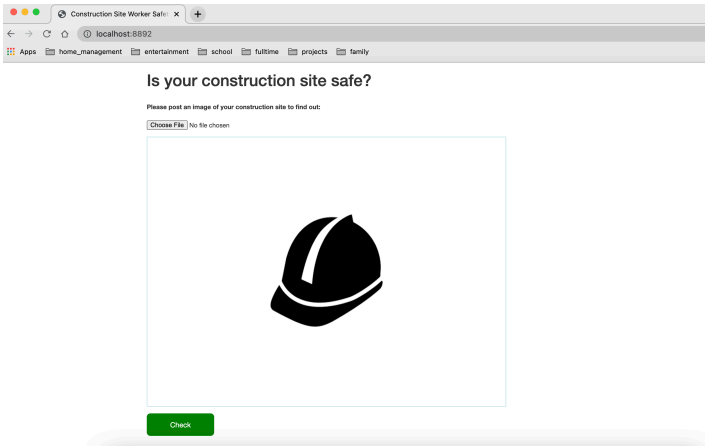
After the tuning, the hyper-parameters with score threshold = 0.6, IoU threshold = 0.2, and max epochs = 90 are selected to produce the best model in this project.

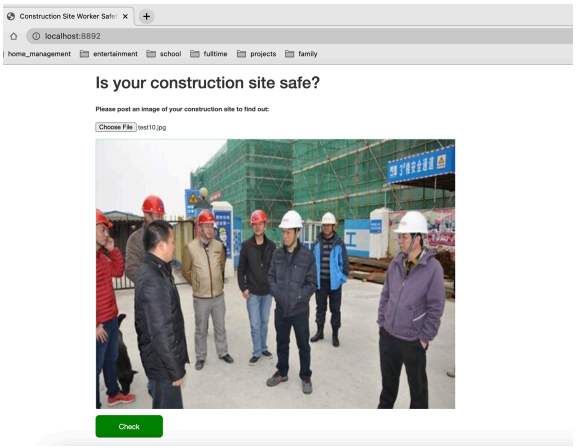
Output of the model

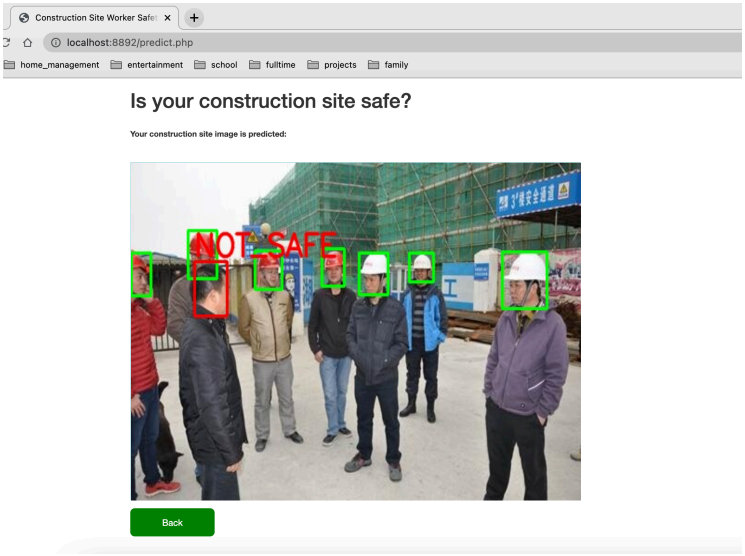
With the best model hyper-parameters are selected, and the best model is trained, the system now can use this model to do prediction on the unseen data or images.

Similar to how to train the model, the unseen image will be convert to an instance of class "SafetyDetectionImageDataSet" which holds the transformed and formatted image matrix, then the "SafetyDetectionImageDataSet" instance will pass to the trained model to go through forwarding path. And the prediction result, the array of boxes, will be visualized back on the image and output to the user end.

This project also uses html and php to create a web app as the user end application. The web app first ask user to choose an image of a construction site from the computer, and it will pass the image to backend which will call a Python script to preprocess the image, build the trained model, run the model prediction code and visualize the result on the image and then display the processed image back on the web app. The following is the screenshot of the web app example:







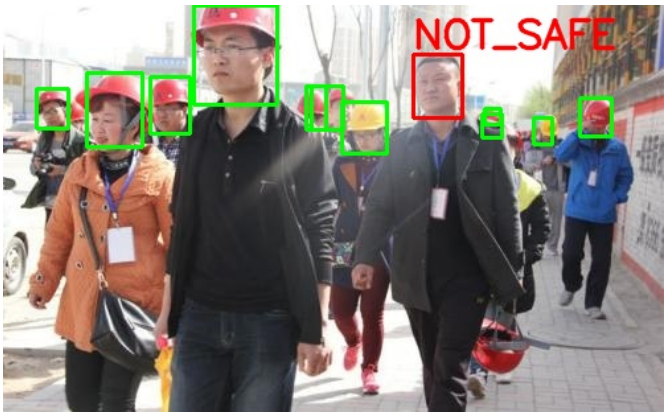
Results

During the model turning, the following hyper-parameters are finalized:

- score threshold = 0.6

- IoU threshold = 0.2
- max epochs = 90

With these hyper-parameters, the model is used to predict 10 unseen images in order to the evaluate the model, see the labeled images as below:



Prediction Average Precision(AP) for class 1(no helmet) = 1.0, and Prediction AP for class 2(with helmet) is 0.947, which is lower than class 1 because there are 2 False Positions labeled, and so Prediction mean AP = 0.974 which is close to 100%.

Prediction Average Recall(AR) for class 1(no helmet) = 1.0, and Prediction AR for class 2(with helmet) is 0.857, which has 6 False Negatives labeled, and so Prediction mean AR = 0.929 which is also close to 100%.

Therefore, the model prediction has mostly good results and high accuracy.

Justification

In order to see if the above model can predict the best result, here is Prediction AP and AR comparison between different IoU thresholds (with score threshold = 0.6, Epoch = 50):

Performances	IoU threshold 0.15	IoU threshold 0.2	IoU threshold 0.3	IoU threshold 0.5	IoU threshold 0.7
Class 1 AP	0.923	0.925	0.923	0.946	0.943
Class 2 AP	1.0	1.0	1.0	0.600	0.500
Mean AP	0.962	0.963	0.962	0.773	0.722
Class 1 AR	0.900	0.902	0.857	0.875	0.846
Class 2 AR	0.750	0.750	0.750	0.750	0.750
Mean AR	0.825	0.826	0.804	0.813	0.798

In this table, mean AP and mean AR are both in highest values when the IoU threshold is 0.2. So it can infer that with a greater max epoch number 90, the same 0.2 IoU threshold value should have even better precision and recall than when max epoch is 50 since the loss is minimum at epoch = 90.

Conclusion

From the above comparison, the system is proved that it is able to have a very accurate result. And during these prediction, the average back-end processing time between clicking the "Check" button and showing the output on the web app is about 5 seconds, which includes fetching the model parameters, rebuilding the model, preprocessing image, executing the model forwarding path and post-processing image, so the actual predicting time is much less than 5 seconds. Also, the model can predict well on image with far and small objects, and can label almost each person's head in a large group on an image. Therefore, this construction site safety detection system can meet the requirements of being fast, accurate, remote, and scalable. This system with Faster R-CNN model should be able to help the safety managers to control the safety at their construction site.

References:

1. Bureau of Labor Statistics (BLS). Occupational Safety and Health Admisinstration (OSHA): Commonly Used Statistics, osha.Gov. 2019. https://www.osha.gov/oshstats/commonstats.html. Accessed: April 26, 2021.
2. Idris Jeelani, Khashayar Asadi, Hariharan Ramshankar, Kevin Han, Alex Albert. Real-time vision-based worker localization & hazard detection for construction. Automation in Construction, Volume 121. ISSN 0926-5805. 2021. https://www.sciencedirect.com/science/article/pii/S0926580520310281.
3. Shervin Minaee. 20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics. Oct 28, 2019. https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce.
4. Padilla, Rafael and Passos, Wesley L. and Dias, Thadeu L. B. and Netto, Sergio L. and da Silva, Eduardo A. B. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. Electronics, Volume 10. 2021.
5. Ian Goodfellow, Yoshua Bengio and Aaron Courville. Deep Learning - An MIT Press book. Chapter 9: Convolutional Network. Pp326.
6. Emilio Soria Olivas, Jose David Martin Guerrero, Marcelino Martinez Sober, Jose Rafael Magdalena Benedito and Antonio Jose Serrano Lopez. Handbook Of Research On Machine Learning Applications and Trends: Algorithms, Methods and Techniques (2 Volumes). Chapter 11: Transfer Learning.
7. TORCHVISION.MODELS. http://pytorch.org/vision/stable/models.html. April 26, 2021.
8. Johannes Schmidt. Train your own object detector with Faster-RCNN & PyTorch. Feb 23, 2021. https://johschmidt42.medium.com/train-your-own-object-detector-with-faster-rcnn-pytorch-8d3c759cfc70.
9. Yinghan Xu, Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images Dataset V4. Nov 19, 2018. https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a.
10. Udacity, Machine Learning Engineer Nanpdegree Program. April, 2021.