YOLO_v5_Project_Submission

December 15, 2022

1 Instructions

This document is a template, and you are not required to follow it exactly. However, the kinds of questions we ask here are the kinds of questions we want you to focus on. While you might have answered similar questions to these in your project presentations, we want you to go into a lot more detail in this write-up; you can refer to the Lab homeworks for ideas on how to present your data or results.

You don't have to answer every question in this template, but you should answer roughly this many questions. Your answers to such questions should be paragraph-length, not just a bullet point. You likely still have questions of your own – that's okay! We want you to convey what you've learned, how you've learned it, and demonstrate that the content from the course has influenced how you've thought about this project.

2 Project Name

Project mentor: Yuli Wang

Haoxuan Huang hhuan110@jhu.edu, Yinglong Wang ywang791@jh.edu, Yongrui Qi yqi28@jh.edu, Zhiqing Zhong zzhong19@jh.edu

Link to GitHub

https://github.com/Auroraaa-Qi/yolo-mask-detection

3 Outline and Deliverables

List the deliverables from your project proposal. For each uncompleted deliverable, please include a sentence or two on why you weren't able to complete it (e.g. "decided to use an existing implementation instead" or "ran out of time"). For each completed deliverable, indicate which section of this notebook covers what you did.

If you spent substantial time on any aspects that weren't deliverables in your proposal, please list those under "Additional Work" and indicate where in the notebook you discuss them.

3.0.1 Uncompleted Deliverables

1. "Expect to complete #1": we decided to use dealing with the real time face mask detection, where the original input would be video not the image, we have not written the code to transfer the video to the images yet.

3.0.2 Completed Deliverables

1. "Must complete #1": We discuss our dataset pre-processing [in "Dataset" below].
2. "Must complete #2": We discuss training our logistic regression baseline [in "Baselines" below].
3. "Must complete #3": We use the training model to get the curve of training loss,metrics percision,metricss recall,recall cofidence,precision recall,precision confidence to evualte the training model on this specific task

3.0.3 Additional Deliverables

1. We decided to add a second baseline using the published model from this paper. We discuss this [in "Baselines" below].
2. We add some results of unusual face mask to show the model's ability to recognize the whether people reaaly have the mask to cover the nose and mouth.

4 Preliminaries

What problem were you trying to solve or understand?

What are the real-world implications of this data and task?

How is this problem similar to others we've seen in lectures, breakouts, and homeworks?

What makes this problem unique?

What ethical implications does this problem have?

In the real-world, implication would be the detection of facial masks as an necesssay tool to protect of everyone in covid-19 and some othercommunicable diseases.Especially in places where facial masks are required like hospitals. The problem could simply consider as the CNN classifier problem. The reason why this problem could consider to be unique since the face masks could have different shape but same function which could protect the people. The detector may not be allow to use when the laws prihibit the goverment or company to collect the people's data no matter people agree.

4.1 Dataset(s)

4.1.1 Describe the dataset(s) you used.

Dataset we used contains of 2 parts: images with/without masks, and corresponding annotations in .xml format illustrating image labels, bounding box size and position.

4.1.2 How were they collected

They are open sourced dataset from kaggle.

4.1.3 Why did you choose them

1. Since the annotations are available, we can focus more on model and methods.
2. Each photo in this dataset may have multiple bounding boxes, which is more robust and generalizable than single object detection images.

4.1.4 How many examples in each

We have 853 images in total, with three labels in different amount: - without_mask: 717 - mask_wared_incorrect: 123 - with_mask: 3232

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
# Load your data and print 2-3 examples
import os
os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"

img_names=[]
annotations=[]

for dirname, _, filenames in os.walk("/content/drive/My Drive/MachineLearning/
 ↪archive"):
    for filename in filenames:
        if os.path.join(dirname, filename)[-3:]==("png" or "jpg"):
            img_names.append(filename)
        elif os.path.join(dirname, filename)[-3:]=="xml":
            annotations.append(filename)
```

[ ]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os
os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"


img_names=['maksssksksss110.png','maksssksksss82.png']
annotations=['maksssksksss110.xml','maksssksksss82.xml']

for i in range(2):
  img_path = "/content/drive/My Drive/MachineLearning/archive/images/" +␣
 ↪img_names[i]

  im = mpimg.imread(img_path)
  plt.imshow(im)
  plt.show()
```

[ ]:
```
from xml.etree import ElementTree as ET

for i in range(2):
    file_path = "/content/drive/My Drive/MachineLearning/archive/annotations/"␣
 ↪+ annotations[i]
```

```
    file=ET.parse(file_path)
    objs=file.findall('object')

    for obj in objs:
        for child in obj.getchildren():
            print(child.tag,':',child.text)
```

```
<ipython-input-3-1d5781724ecf>:10: DeprecationWarning: This method will be
removed in future versions.  Use 'list(elem)' or iteration over elem instead.
  for child in obj.getchildren():
```

```
name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
```

```
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
```

```
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :
```

```
name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : without_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :
```

```
name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :

name : with_mask
pose : Unspecified
truncated : 0
occluded : 0
difficult : 0
bndbox :
```

4.2 Pre-processing

4.2.1 What features did you use or choose not to use? Why?

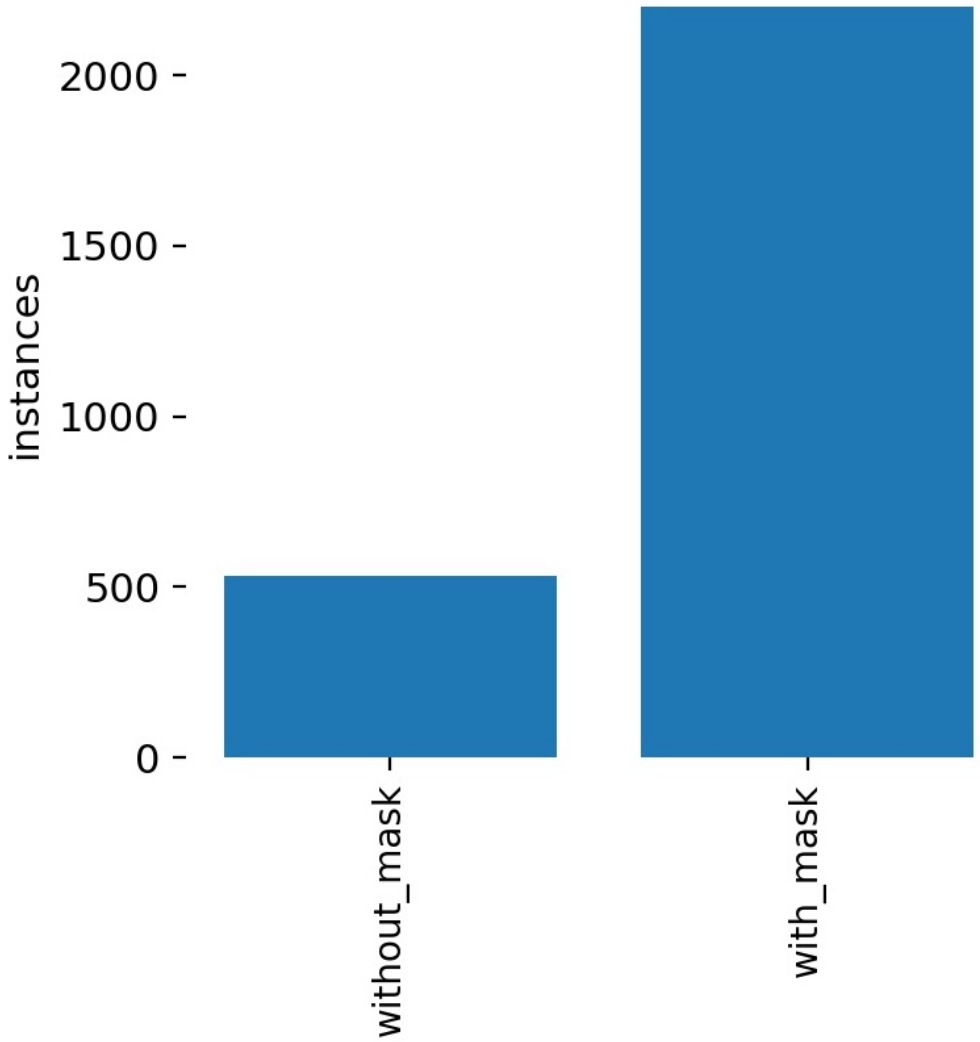
x, y : coordinates of middle of bounding box;

$width, height: $ width, height of bounding box.

Since the first step of successful detection is correctly locating where the mask should be, we need such features to learn labels.

4.2.2 If you have categorical labels, were your datasets class balanced?

No, classes are unbalanced. Due to the unbalanced distribution, we dropped "mask_weared_incorrect" label, and the proportion of "with_mask" is much larger than "without_mask".



4.2.3 How did you deal with missing data? What about outliers?

We delete the specific example if there's missing or outliers for any feature.

4.2.4 What approach(es) did you use to pre-process your data? Why?

1. We dropped "mask_weared_incorrect" rows. Keeping it may cause unwanted noise and reduce the accuracy for "with_mask" class.
2. Features were normalized between 0~1, to make all the features are in the same base unit.

4.2.5 Are your features continuous or categorical? How do you treat these features differently?

After normalization, all features are continuous.

[ ]:
```
# For those same examples above, what do they look like after being␣
↪pre-processed?
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Machine Learning/archive/preprocessed.
↪csv')
df
```

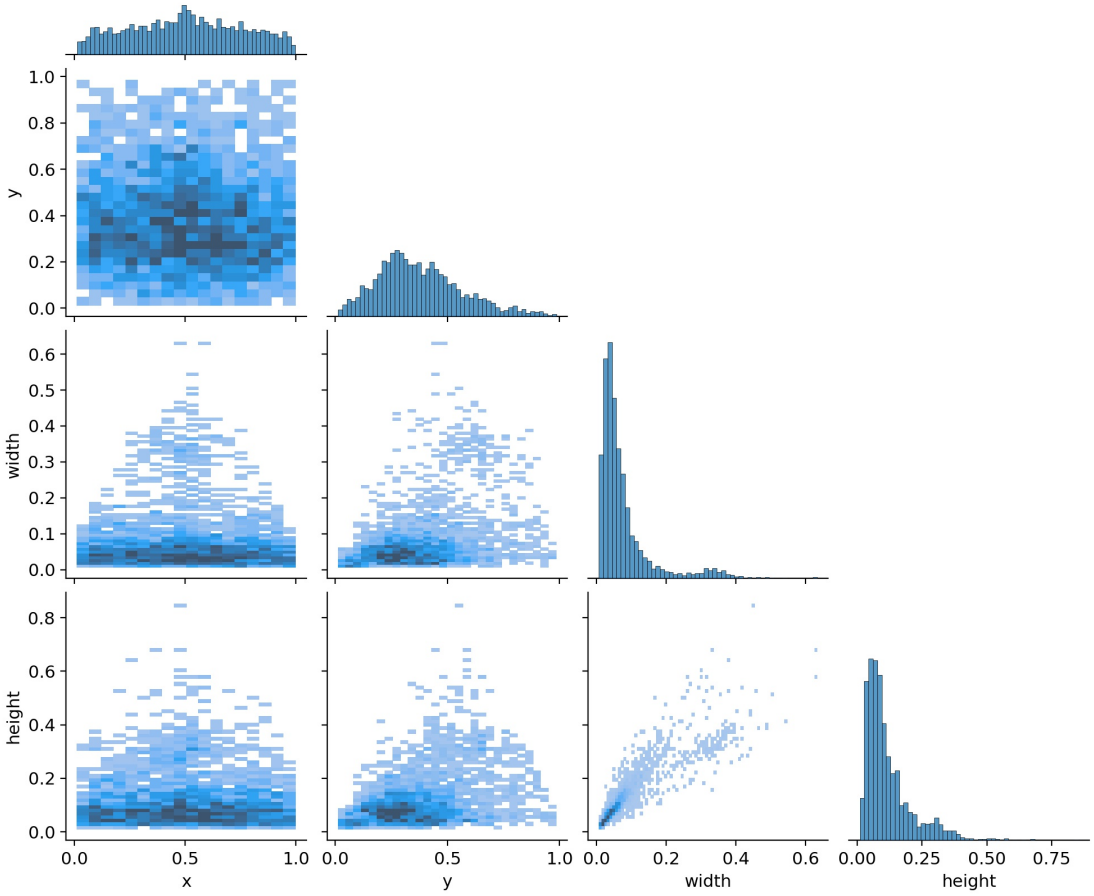
[ ]:

	Unnamed: 0	file	classnames	class_id	width	height	\
0	0	maksssksksss110.png	without_mask	0	400	267	
1	1	maksssksksss110.png	without_mask	0	400	267	
2	2	maksssksksss110.png	with_mask	1	400	267	
3	3	maksssksksss110.png	with_mask	1	400	267	
4	4	maksssksksss110.png	without_mask	0	400	267	
…	…	…	…	…	…	…	
3944	3944	maksssksksss82.png	with_mask	1	400	225	
3945	3945	maksssksksss82.png	with_mask	1	400	225	
3946	3946	maksssksksss82.png	with_mask	1	400	225	
3947	3947	maksssksksss82.png	with_mask	1	400	225	
3948	3948	maksssksksss82.png	with_mask	1	400	225	

	xmin	ymin	xmax	ymax	x_pos	y_pos	frame_width	frame_height
0	6	111	43	148	0.06125	0.485019	0.0925	0.138577
1	22	20	49	46	0.08875	0.123596	0.0675	0.097378
2	48	12	70	37	0.14750	0.091760	0.0550	0.093633
3	113	1	136	24	0.31125	0.046816	0.0575	0.086142
4	78	17	106	46	0.23000	0.117978	0.0700	0.108614
…	…	…	…	…	…	…	…	
3944	57	67	79	89	0.17000	0.346667	0.0550	0.097778
3945	149	45	172	72	0.40125	0.260000	0.0575	0.120000
3946	192	36	215	65	0.50875	0.224444	0.0575	0.128889
3947	241	74	262	97	0.62875	0.380000	0.0525	0.102222
3948	331	58	354	88	0.85625	0.324444	0.0575	0.133333

```
[3949 rows x 14 columns]
```

```
[ ]: # Visualize the distribution of your data before and after pre-processing.
     #   You may borrow from how we visualized data in the Lab homeworks.
```



5 Models and Evaluation

5.1 Experimental Setup

5.1.1 How did you evaluate your methods? Why is that a reasonable evaluation metric for the task?

Since object detection considered to be a regression problem in YOLO, MSE is adopted as the loss function.

5.1.2 What did you use for your loss function to train your models? Did you try multiple loss functions? Why or why not?

Total loss makes up of three parts: coordinate loss, classification loss, and confidence loss, which is defined as below:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}[(x_i-\hat{x_i})^2+(y_i-\hat{y_i})^2]+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}[(\sqrt{w_i}-\sqrt{\hat{w_i}})^2+(\sqrt{h_i}-\sqrt{\hat{h_i}})^2]+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}(C_i-\hat{C_i})^2+\lambda_r$$

To get the error, firstly find the bounding boxes with the highest intersection over union (IoU) and with the correct bounding boxes. Then, calculate the confidence loss, which means the probability of the object being present inside a given bounding box. Next, calculate the classification loss, which indicates the present class of objects within the bounding box. Finally, calculate the coordinate loss for matching the detected boxes.

We didn't try other loss functions, for this loss function takes all aspects into consideration.

link to loss function: https://github.com/ultralytics/yolov5/blob/master/utils/loss.py

5.1.3 How did you split your data into train and test sets? Why?

Since this project can be applied in real world, for better generalization and robustness, we didn't split data into train and test sets.

However, we seperate our data into train and validation sets using split-folders package, to address imbalanced datasets by randomized oversampling.

5.2 Baselines

5.2.1 What baselines did you compare against?

The baseline model we chose was a facial mask detector that could only accomplish very basic facial mask detection task. To elaborate on this, the baseline model that we set was a naive facial mask detector which can detect only one facial mask out of a image. Facial masks, detected by the basline model, refer to the most common facial masks in white or blue, i.e., those medical face masks. We are not planning to detect those more advance facial masks like N95. Additionally, the baseline model we set was not planning to detect many facial masks out of a single image. A single mask is sufficient for our baseline model.

5.2.2 Why are these reasonable?

This is a reasonable baseline since we are completing a facial mask detector and this detector shall at leaset detect a facial mask out of images. So it is reasonable to have a detector that detects only one basic facial mask. In fact, we complete this baseline model easily as we have a facial mask detector that can detect many facial masks out of a single image and the facial masks doesn't need to be basic medical masks in blue or white but can be N95 or more advance masks in black or other colors.

5.3 Methods

5.3.1 What methods did you choose? Why did you choose them?

We choose Yolov5, because

1. Recall CNN learned at class, kernel as a filter has the nature of translation invariance, which just matches the goal of multiple object detection in this project. Also, convolution is a layer of Yolo;
2. The structure of yolo is single way without recurrent, which is easy to get started with and simple to learn;
3. Yolo, You Only Look Once, is fast and accurate in real-time pattern detection.

5.3.2 How did you train these methods, and how did you evaluate them? Why?

We train the method with train-validation split, since splitfoler allows randomized oversampling for imbalanced datasets.

We evaluate on new images from other dataset for better generalization and robustness, meanwhile avoid using test case to fine-tune.

5.3.3 Which methods were easy/difficult to implement and train? Why?

Yolov5 has many versions: YOLOv5n(nano), YOLOv5s(small), YOLOv5m(medium), YOLOv5l(large), etc. Apparently, smaller the model is, easier and faster to train and fine-tune.

However, for the tradeoff between training computation and testing accuracy, we use YOLOv5s rather than YOLOv5n.

5.3.4 For each method, what hyperparameters did you evaluate? How sensitive was your model's performance to different hyperparameter settings?

YOLOv5 has about 30 hyperparameters used for various training settings including model depth, layer channels, layers, neuron size per layers, and anchors. Since we fine-tune on YOLOv5s, we use the default hyperparameter settings.

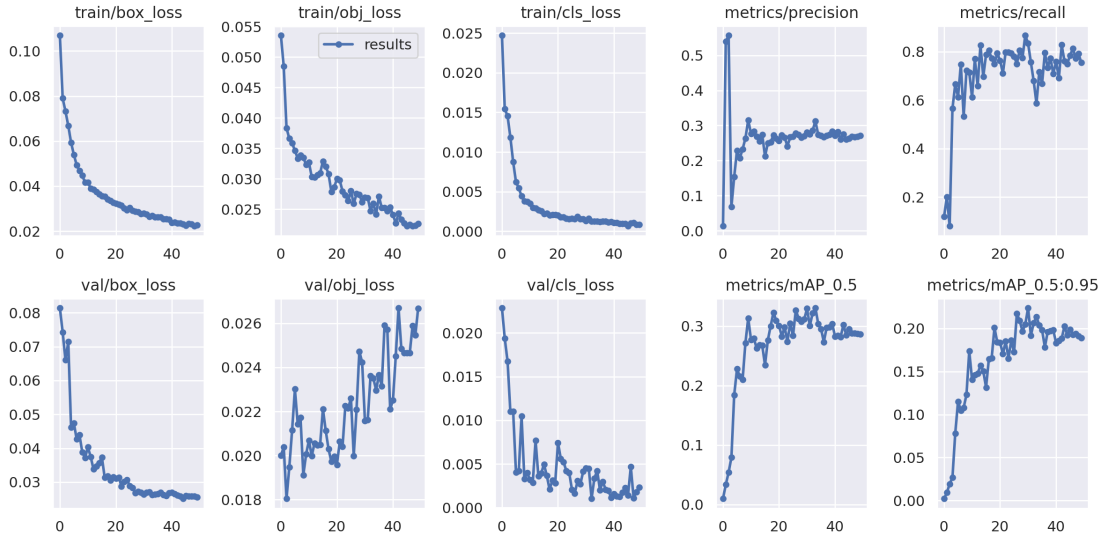
However, according to our experiments and comparison between different YOLOv5 models, YOLOv5 is sensitive to anchors otherwise the it performed badly at pattern matching.

[ ]:
```
# Code for training models, or link to your Git repository

# clone yolov5 repo
!git clone https://github.com/ultralytics/yolov5
# requirements install
%pip install -qr requirements.txt
# training on our dataset
!python3 train.py --workers 2 --img 640 --epochs 50 \
--data "/content/drive/MyDrive/Machine Learning/imagelabel/masks.yaml" \
--weights yolov5s.pt --device {device} --cache
```

Link to GitHub: https://github.com/Auroraaa-Qi/yolo-mask-detection/blob/main/ML_final_project.ipynb

```
[ ]: # Show plots of how these models performed during training.
#  For example, plot train loss and train accuracy (or other evaluation metric)␣
 ↪on the y-axis,
#  with number of iterations or number of examples on the x-axis.
```

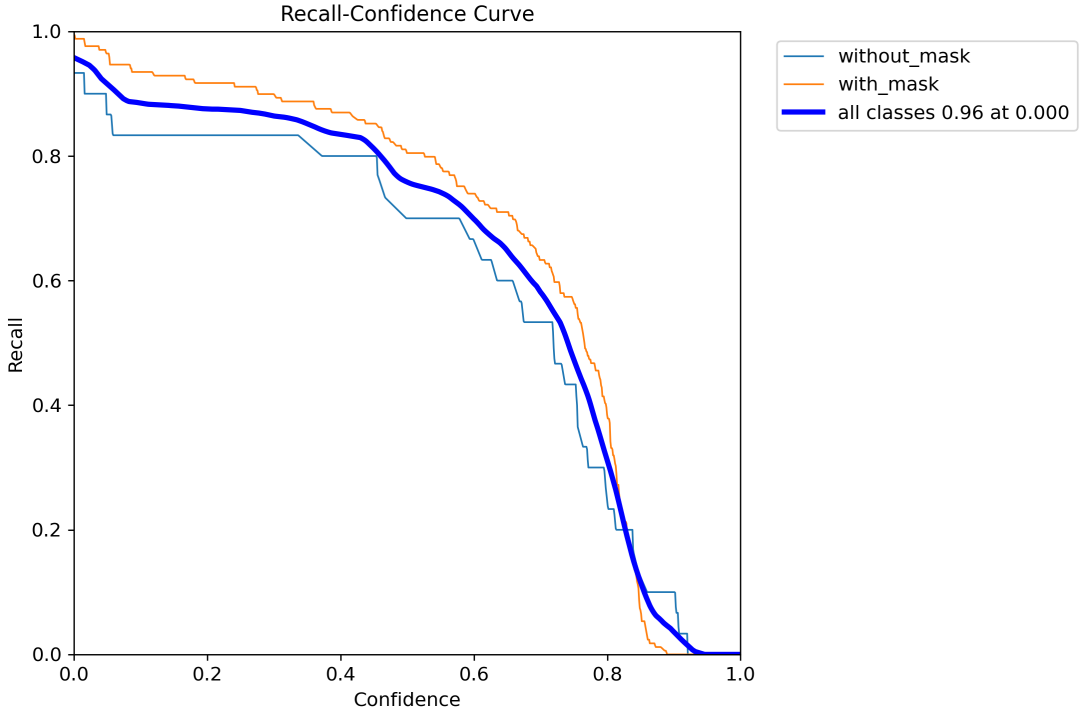


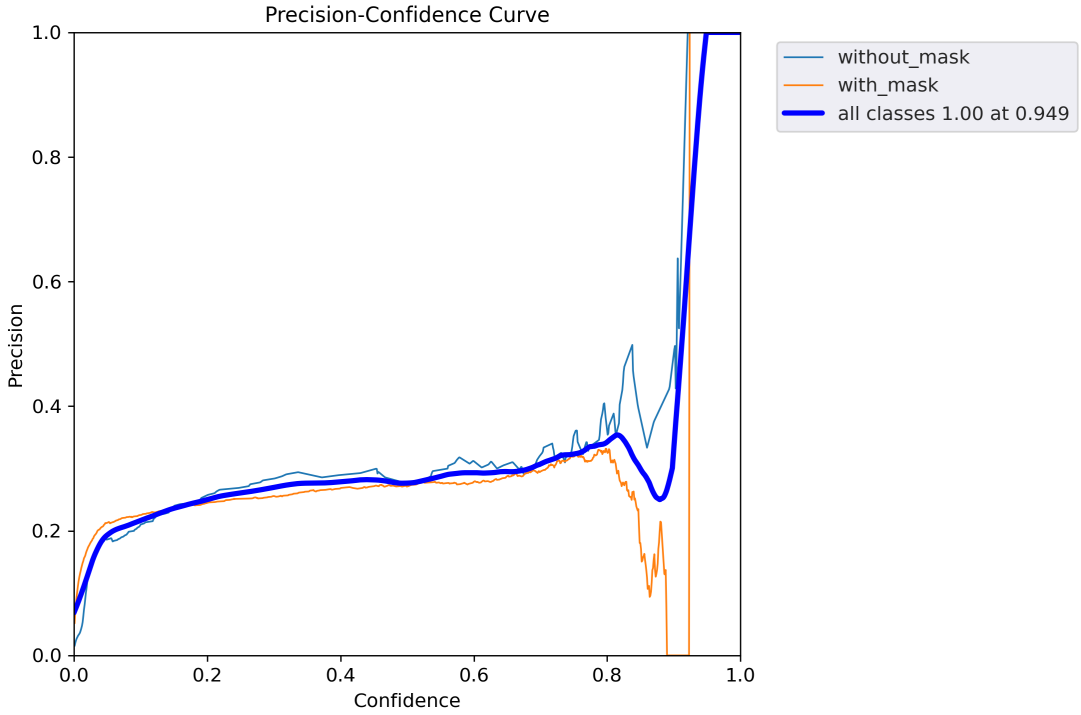
5.4 Results

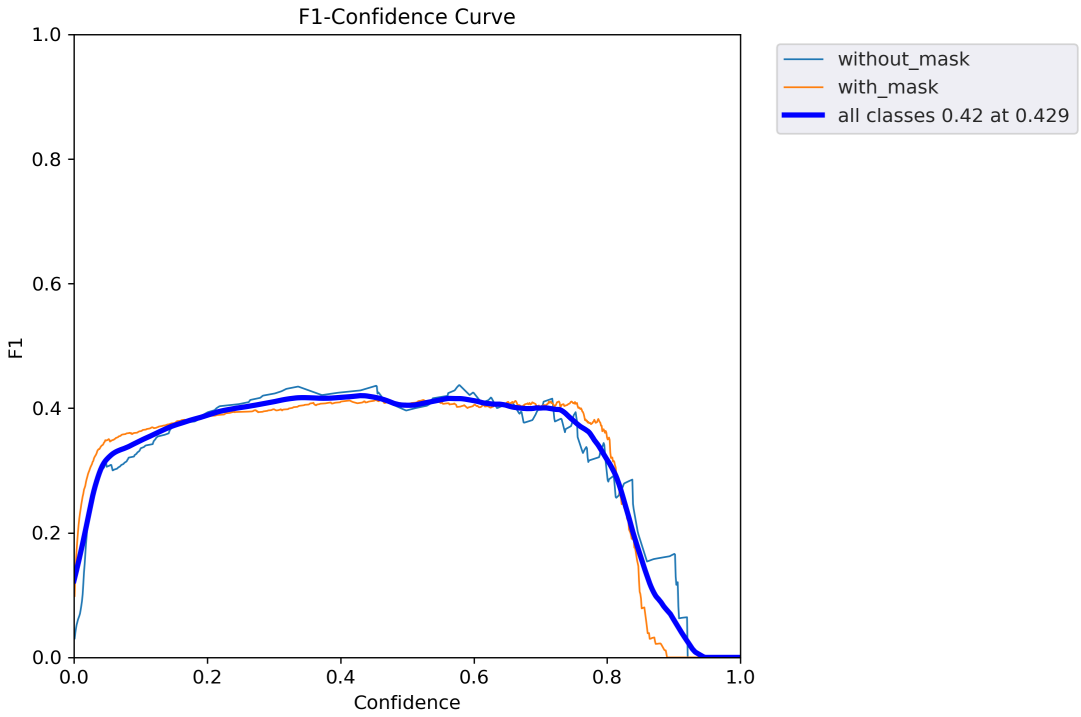
```
[ ]: #   Show plots or visualizations of your evaluation metric(s) on the train and␣
 ↪test sets.
#   What do these plots show about over- or under-fitting?
#   You may borrow from how we visualized results in the Lab homeworks.
#   Are there aspects of your results that are difficult to visualize? Why?
```

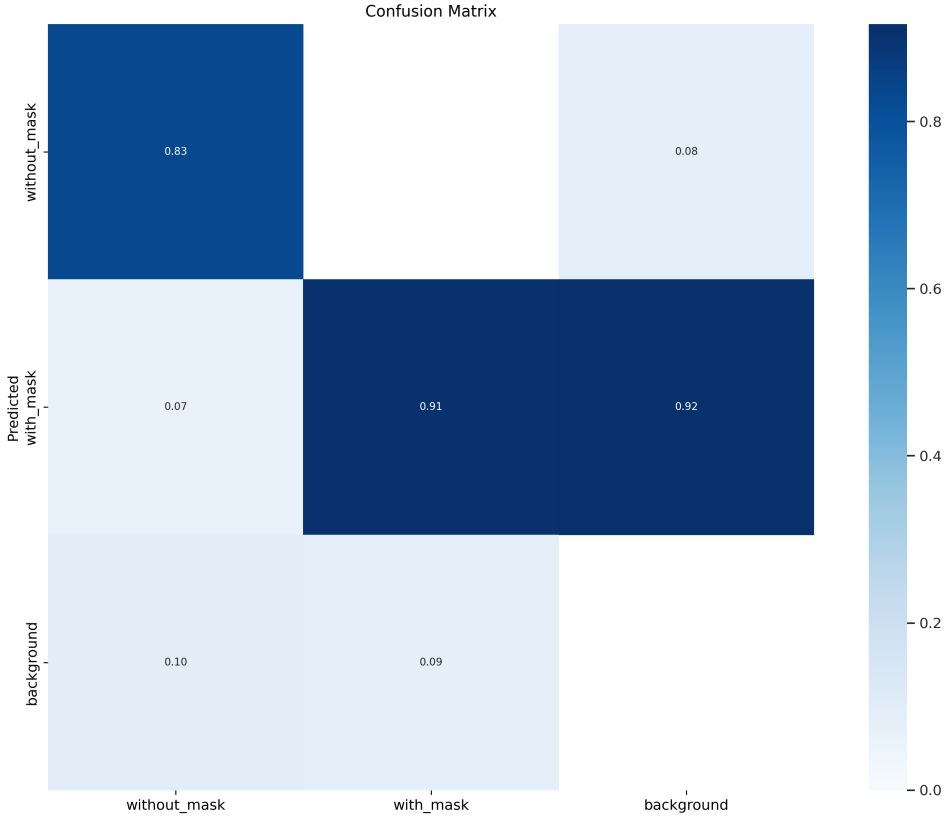
5.4.1 These plot shows the result do not have overfitting or underfitting

The results are not difficult to visualize


Recall-Confidence Curve
Recall
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
without_mask
with_mask
all classes 0.96 at 0.000


Precision-Confidence Curve
Precision
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
without_mask
with_mask
all classes 1.00 at 0.949


F1-Confidence Curve
without_mask
with_mask
all classes 0.42 at 0.429
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix
without_mask
with_mask
background
Predicted
0.83
0.08
0.07
0.91
0.92
0.10
0.09
0.0
0.2
0.4
0.6
0.8

6 Discussion

6.1 What you've learned

Note: you don't have to answer all of these, and you can answer other questions if you'd like. We just want you to demonstrate what you've learned from the project.

What concepts from lecture/breakout were most relevant to your project? How so?

Figuring out bias in data is most relevant to our project. Since our goal is to perform facemask detection, different skin color and facemask color would influence the result. The data needs to be examined to obtain a solid and un-bias model

What aspects of your project did you find most surprising?

Surprisingly, with only a few hours of training, the model obtain a precision of nearly 80% correctness.

What lessons did you take from this project that you want to remember for the next ML project you work on? Do you think those lessons would transfer to other datasets and/or models? Why or why not?

I have learned that when some label is seldomly seen (less than 2%) in the dataset,(e.g. mask_incorrect in our case) then you better drop those cases or find more data with that label. I think this could be transfered to other datasets, because it would be painful to train a model classfying on 3 classes where 2 of them takes up 99% and the rest rarely appeared in the dataset.

If you had two more weeks to work on this project, what would you do next? Why?

Make it real-time so that we can play with it by using cameras on our laptop or even our phone.