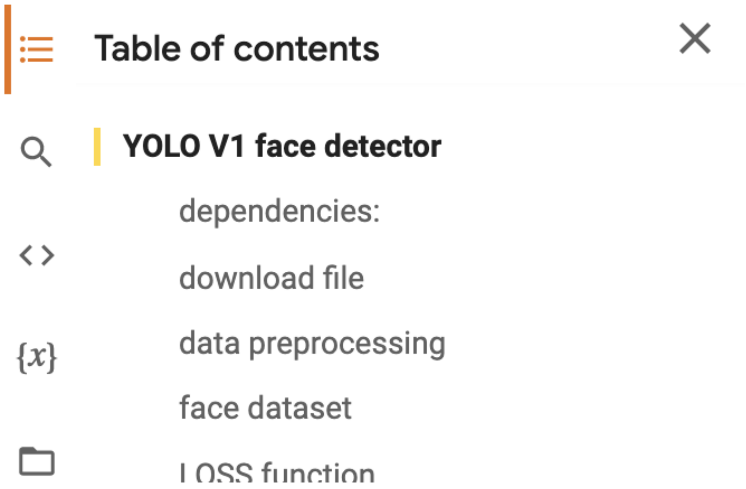
USER MANUAL

Github: https://github.com/zhukson/YOLO-Face-mask-detector

Table of Contents

User manual for two-stage YOLO v1 face mask detector

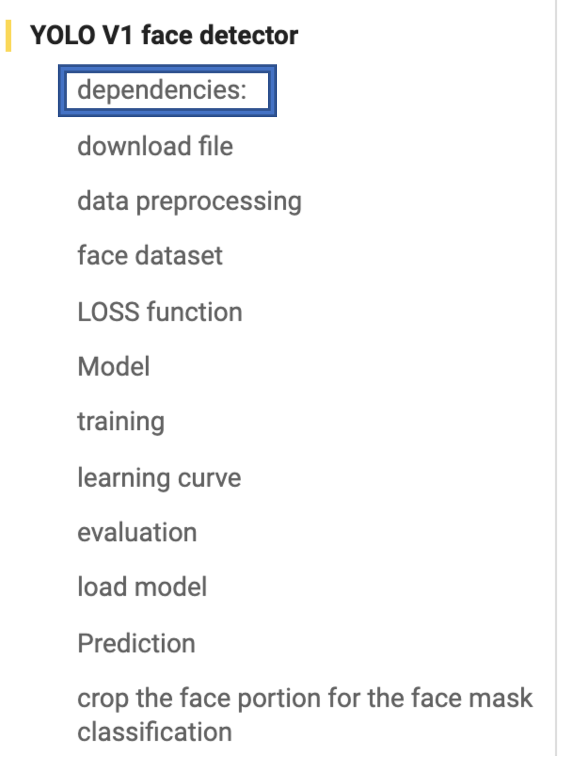
After you downloaded and open the "two_stages_yolo_v1_face_mask_detection! (1).ipynb" through google colab, you would see a menu in the left side of the page.



Part1: YOLO v1 face detector

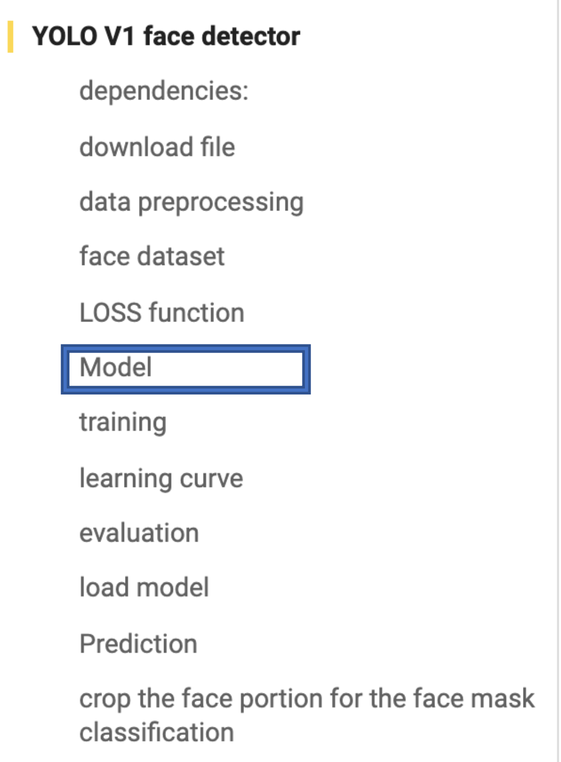
Step 1: import dependencies:

Click on the "dependencies" in the menu and run the according cell.



Step2: Load model

(1)Click on the following “Model” shown below and run the according cell.



YOLO V1 face detector

dependencies:

download file

data preprocessing

face dataset

LOSS function

Model

training

learning curve

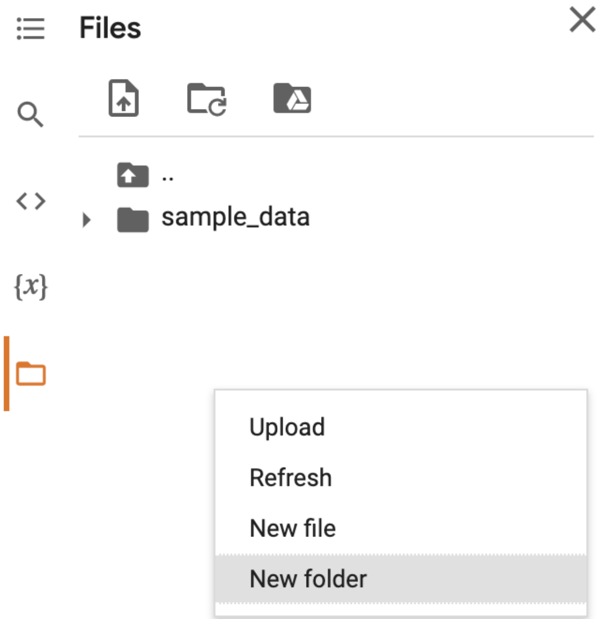
evaluation

load model

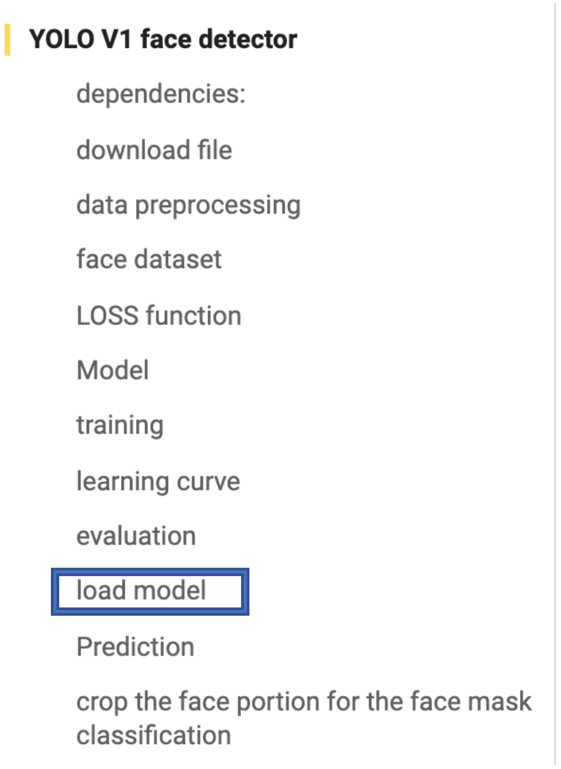
Prediction

crop the face portion for the face mask classification

(2) create a new folder named “saved_models”, and put the pretrained model into it



(3)click on the “load model” from the menu and run the according cell.



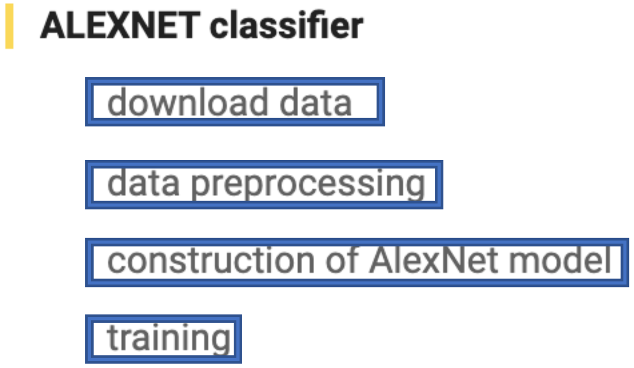
Step3: load required classes

click on the “prediction” from the menu and running the according cell.

Part 2. training ALEXNET classifier

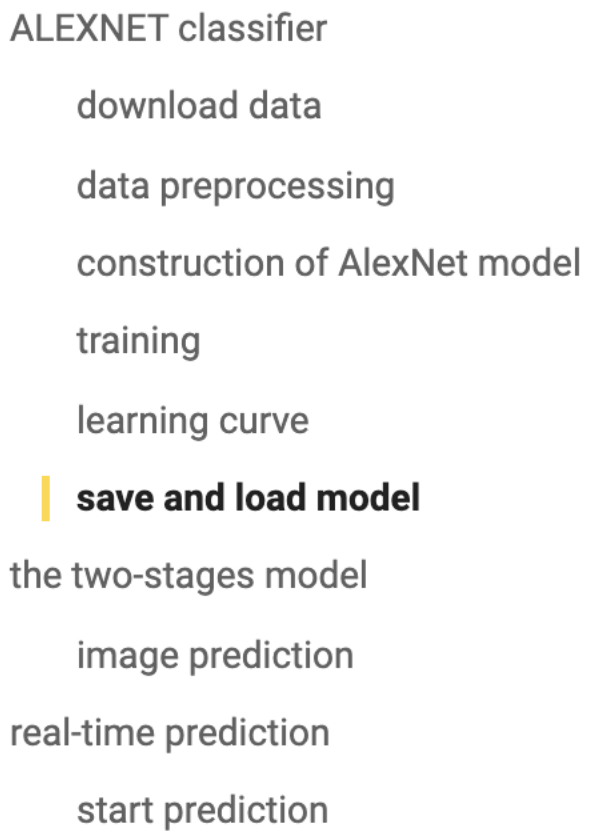
The pretrained weight for alexnet model is too big, it is very time consuming to load it. However, It would only take you less than 5 minutes to trained it from scratch.

Step 1: train the model
You only need to run the sections shown below, and the model would automatically be trained.



Step 2: save and load the model

Running the “save and model” section to save and load the model.



ALEXNET classifier

- download data
- data preprocessing
- construction of AlexNet model
- training
- learning curve
- **save and load model**

the two-stages model

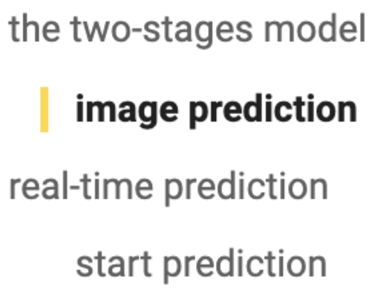
- image prediction

real-time prediction

- start prediction

Part 3. Prediction

3.1 image prediction:
You only need to click on the “image prediction section”, upload the image you want to make prediction and enter the path of this image as the input of function one_stop(file_path).

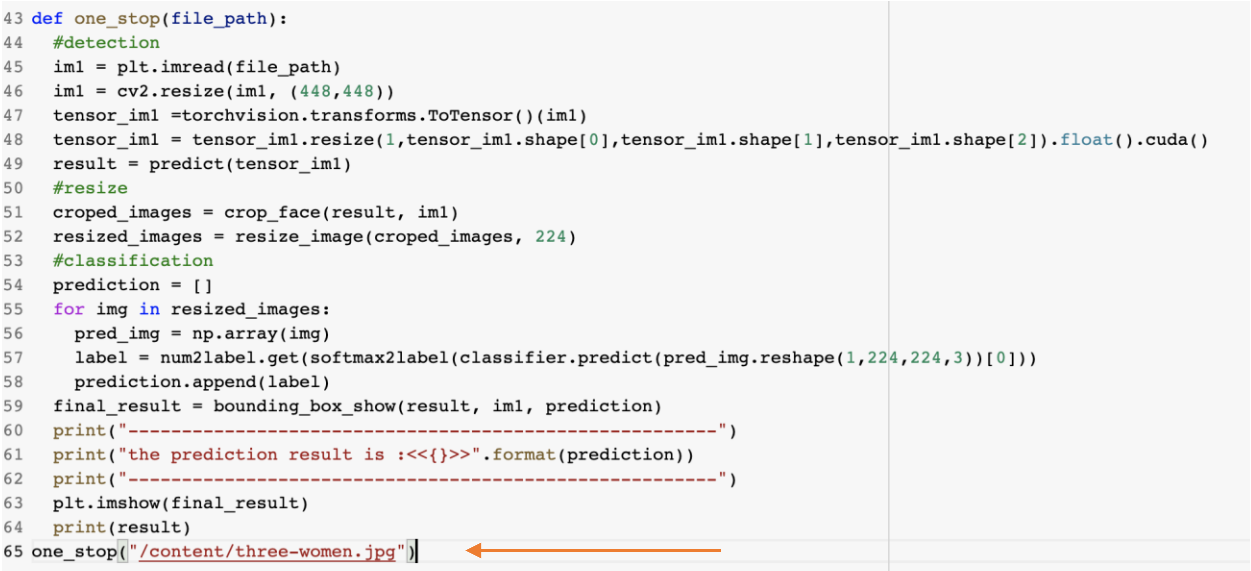


the two-stages model

- **image prediction**

real-time prediction

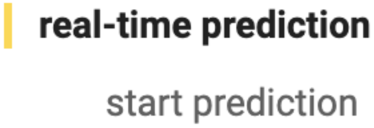
- start prediction



```
def one_stop(file_path):
  #detection
  im1 = plt.imread(file_path)
  im1 = cv2.resize(im1, (448,448))
  tensor_im1 =torchvision.transforms.ToTensor()(im1)
  tensor_im1 = tensor_im1.resize(1,tensor_im1.shape[0],tensor_im1.shape[1],tensor_im1.shape[2]).float().cuda()
  result = predict(tensor_im1)
  #resize
  croped_images = crop_face(result, im1)
  resized_images = resize_image(croped_images, 224)
  #classification
  prediction = []
  for img in resized_images:
    pred_img = np.array(img)
    label = num2label.get(softmax2label(classifier.predict(pred_img.reshape(1,224,224,3))[0]))
    prediction.append(label)
  final_result = bounding_box_show(result, im1, prediction)
  print("-------------------------------------------------------")
  print("the prediction result is :<<{}>>".format(prediction))
  print("-------------------------------------------------------")
  plt.imshow(final_result)
  print(result)
one_stop("/content/three-women.jpg")
```

3.2 camera cam prediction:

you need to make sure your computer has camera. Then run the “real time prediction” to load the required functions and run “start prediction” section to open your camera and start the prediction.



real-time prediction

start prediction

YOLO V5 USER MANUAL

1. Clone the source code
2. Download the preprocessed dataset
3. Ddownload the pretrained weight available in the GitHub link and put it into any directory in YOLO V5
4. Running code to make prediction:

   3.1 image inference

   ```
   !python detect.py --source FILE_PATH.jpg --weights WEIGHT_PATH
   ```

   3.2 video inference

   ```
   !python detect.py --source FILE_PATH.mp4 --weights WEIGHT_PATH
   ```