Project Proposal - 13th April Title:

“Scaled-YOLOv4: Scaling Cross Stage Partial Network “, Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao https://doi.org/10.48550/arXiv.2011.08036

Name – Kishan Kumar Ravikumar **Course** – CS 541 Artificial Intelligence **Mail Id** – kravikumar@stevens.edu

Abstract

The authors show that the CSP-based YOLOv4 object identification neural network scales up and down and can be used to small and large networks while maintaining optimal speed and accuracy. The authors suggest a network scaling method that alters not just the network's depth, width, and resolution, but also its structure. The YOLOv4-large model achieves state-ofthe-art results, with 55.5 percent AP (73.4 percent AP50) for the MS COCO dataset at a speed of 16 FPS on Tesla V100, and 56.0 percent AP with test time augmentation (73.3 AP50). To the best of our knowledge, this is the most accurate study on the COCO dataset that has ever been released. On the RTX 2080Ti, the YOLOv4-tiny model scores 22.0 percent AP (42.0 percent AP50) at 443 FPS, while by using TensorRT, batch size = 4 and FP16-precision the YOLOv4tiny achieves 1774 FPS.

Github link (in abstract): https://github.com/Kishanjr/CS-541-Artificial-intelligence.git

Data:

MS COCO dataset was released in its initial version in 2014. There are 164K photos in total, including 83K in training, 41K in validation, and 41K in testing. In 2015, a new 81K image test set was released, which included all of the previous test images as well as 40K new photographs.

The training/validation split was modified from 83K/41K to 118K/5K in 2017 in response to community feedback. The same images and notes appear in the new split. The 2017 test set is a subset of the 2015 test set, which consists of 41K photos. A new unannotated dataset of 123K photos is included in the 2017 release as well.

CON
T...

Environment Setup - 20th April
Tools & Technologies:

First, you'll need to enable GPUs for the notebook:

- Navigate to Edit→Notebook Settings
- select GPU from the Hardware Accelerator drop-down import dependencies

from IPython.display import display, Javascript, Image from google.colab.output import eval_js
from google.colab.patches import cv2_imshow from
base64 import b64decode, b64encode

import cv2 import
numpy as np import
PIL import io
import html import
time
import matplotlib.pyplot as plt %matplotlib inline

darknet https://github.com/AlexeyAB/darknet

Experiments:

Procedure with steps

○ clone repository - https://github.com/AlexeyAB/darknet o change makefile to have GPU, OPENCV and LIBSO enabled

○ make darknet (builds darknet so that you can then use the darknet.py file and have its dependencies)

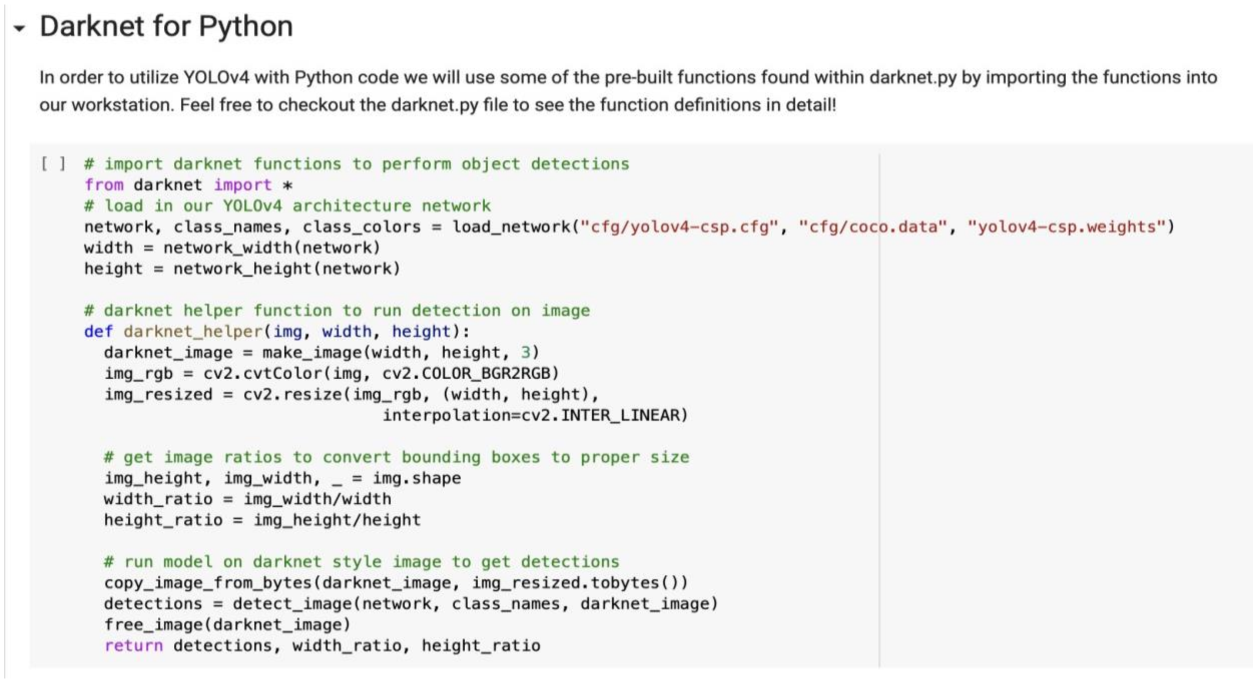
○ get the scaled yolov4 weights file that is pre-trained to detect 80 classes (objects) from shared google drive

○ # import darknet functions to perform object detections o darknet helper function to run detection on image

○ get image ratios to convert bounding boxes to proper size o run model on darknet style

image to get detections

Result :



Darknet for Python

In order to utilize YOLOv4 with Python code we will use some of the pre-built functions found within darknet.py by importing the functions into our workstation. Feel free to checkout the darknet.py file to see the function definitions in detail!

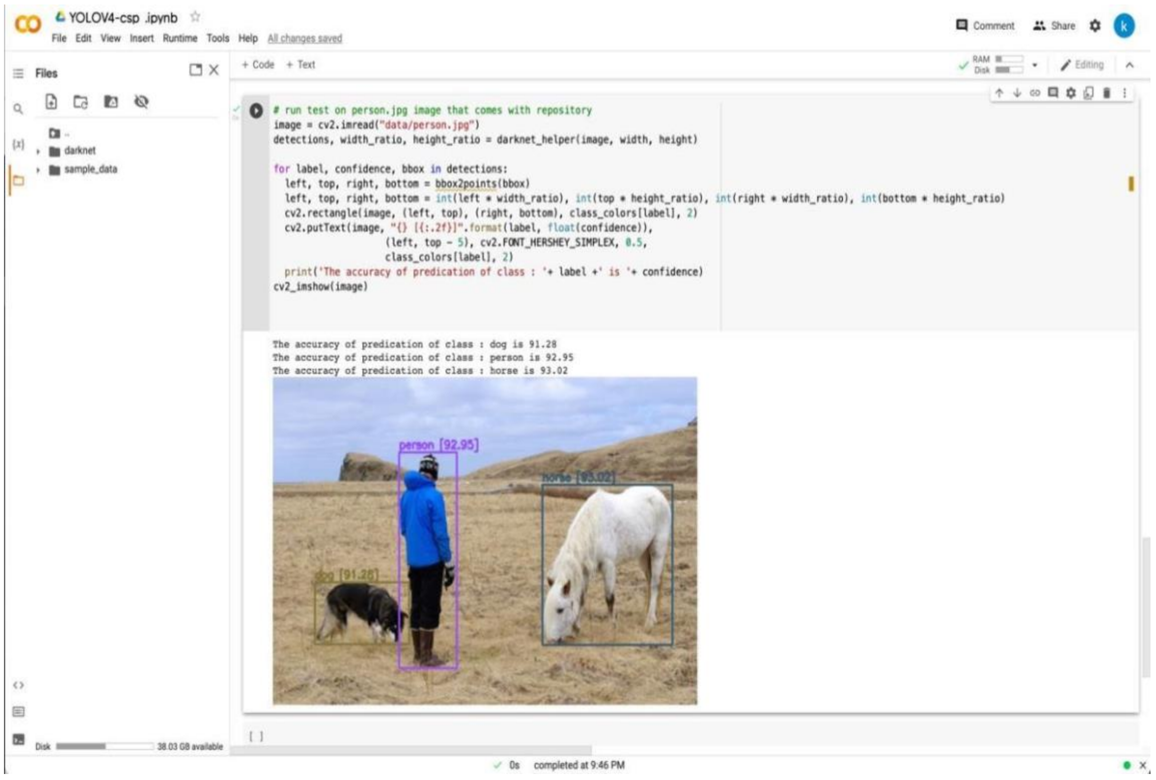
```
[ ] # import darknet functions to perform object detections
    from darknet import *
    # load in our YOLOv4 architecture network
    network, class_names, class_colors = load_network("cfg/yolov4-csp.cfg", "cfg/coco.data", "yolov4-csp.weights")
    width = network_width(network)
    height = network_height(network)

    # darknet helper function to run detection on image
    def darknet_helper(img, width, height):
      darknet_image = make_image(width, height, 3)
      img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      img_resized = cv2.resize(img_rgb, (width, height),
                                  interpolation=cv2.INTER_LINEAR)

      # get image ratios to convert bounding boxes to proper size
      img_height, img_width, _ = img.shape
      width_ratio = img_width/width
      height_ratio = img_height/height

      # run model on darknet style image to get detections
      copy_image_from_bytes(darknet_image, img_resized.tobytes())
      detections = detect_image(network, class_names, darknet_image)
      free_image(darknet_image)
      return detections, width_ratio, height_ratio
```

Pretrained model using Darknet



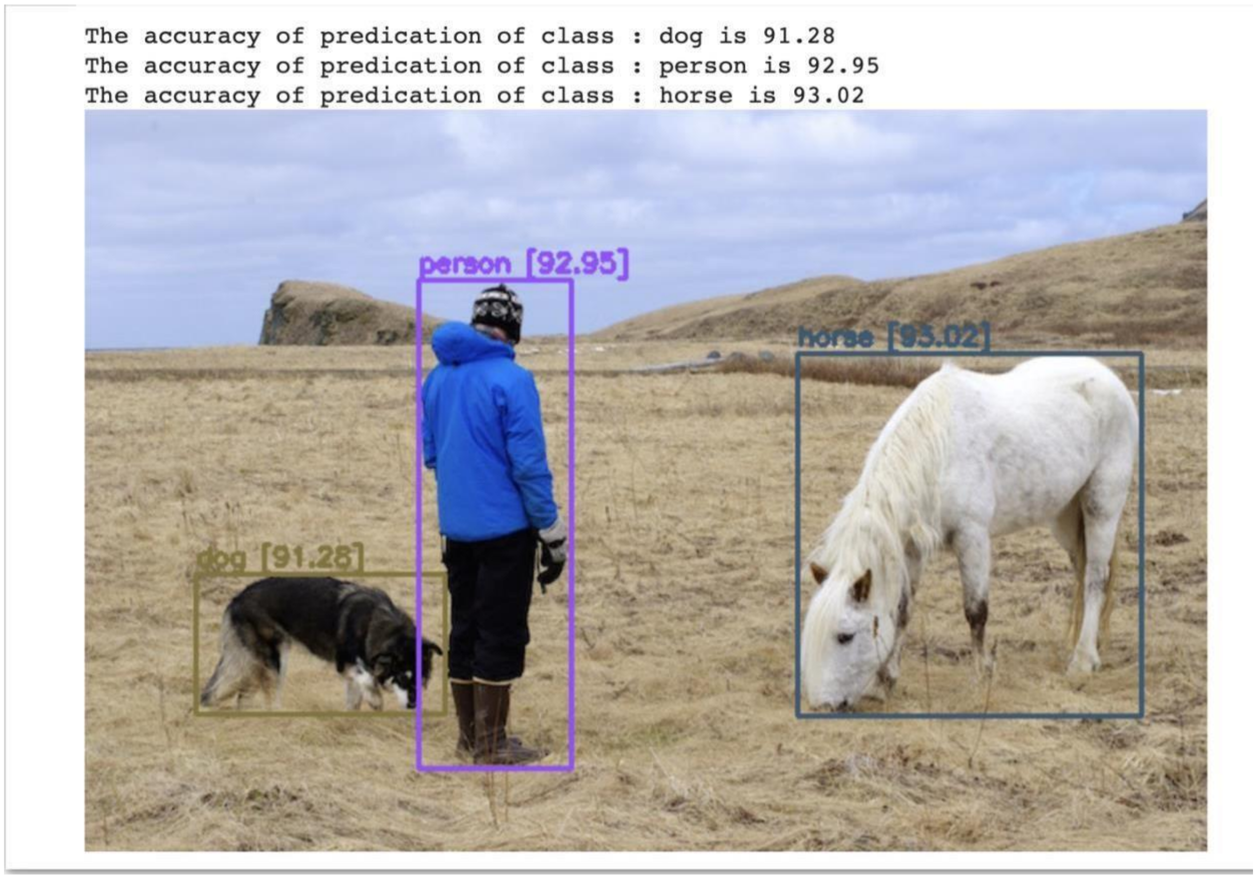
```
# run test on person.jpg image that comes with repository
image = cv2.imread("data/person.jpg")
detections, width_ratio, height_ratio = darknet_helper(image, width, height)

for label, confidence, bbox in detections:
  left, top, right, bottom = bbox2points(bbox)
  left, top, right, bottom = int(left * width_ratio), int(top * height_ratio), int(right * width_ratio), int(bottom * height_ratio)
  cv2.rectangle(image, (left, top), (right, bottom), class_colors[label], 2)
  cv2.putText(image, "{} [{:.2f}]".format(label, float(confidence)),
                    (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                    class_colors[label], 2)
  print('The accuracy of predication of class : '+ label +' is '+ confidence)
cv2_imshow(image)
```

```
The accuracy of predication of class : dog is 91.28
The accuracy of predication of class : person is 92.95
The accuracy of predication of class : horse is 93.02
```

Objects detection work perfect with their label accuracies using the pretrained weights from Yolo v4 csp .

ZOOMED IMAGE OF RESULT :



```
The accuracy of predication of class : dog is 91.28
The accuracy of predication of class : person is 92.95
The accuracy of predication of class : horse is 93.02
```

Problems and issues – No problem or issues until now .

Deliverable -3 27th April

Method:

The following stage is to deal with the quantitative aspects that will vary, such as the number of parameters with qualitative characteristics, after completing model scaling for the suggested object detector. Model inference time, average precision, and other characteristics are among them. Depending on the equipment or database used, the qualitative parameters will have varying gain effects.

Problems and issues : The model run perfectly fine . one problem running the model in Gcp due to lack of 2080 ti rtx gpu . same result are difficult are not expected . Since the gpu provided in colab for free usage is already powerful better results are expected .

Experiments done in paper :

The proposed scaled-YOLOv4 is validated using the MSCOCO 2017 object detection dataset. We don't employ Ima- geNet pre-trained models; instead, all scaled-YOLOv4 models are trained from scratch, with the SGD optimizer as the tool of choice. The training period for YOLOv4-tiny is 600 epochs, whereas the training time for YOLOv4-CSP is 300 epochs. In the case of YOLOv4-large, we initially run 300 epochs before training 150 epochs with a better data augmentation approach. We employ the Lagrangian multi-plier of hyper-parameters, such as anchoring of learning rate and the degree of various data augmentation techniques, for the Lagrangian multi-plier of hyper-parameters.

Result : no differentiable results from previous submission – 2 .

3.4 Final Report - 4th May

Abstract:
In paper, Scaled-YOLOv4: Scaling Cross Stage Partial Network “, Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao https://doi.org/10.48550/arXiv.2011.08036 ,the authors have proposed a new object detection yolo v4 scaled neural network . In the project I have tried to take up the two models namely yolo-v4-csp and yolo -v4-tiny , that the authors compared along with other models . I have verified the results practically in real time testing in image , image capture by webcam and video stream on web cam .

Introduction -

This paper aims to verify the result of yolo-v4-csp and Yolo-v4-tiny practically with help of webcam. This gives clearly understand of how well the scaled object detection network in yolo-v4-csp works, The Fps drop is comparative low in csp model when compared to the tiny model , also the accuracy result in the csp model is much more reliable than the tiny model . Both csp and tiny models are extremely accurate with respect to current day standards .

The Dataset- The COCO Dataset

For assessing object detection methods, the COCO dataset is the gold standard. The COCO dataset (Common Objects in Context) comprises approximately 120,000 photos for training and testing, with 80 object class labels that are widely seen in everyday life. In general, it is assumed that if a model performs well on the COCO dataset, it would generalize well to new bespoke domains - in the spirit of YOLO9000, for 9000 items.

Csp-zing

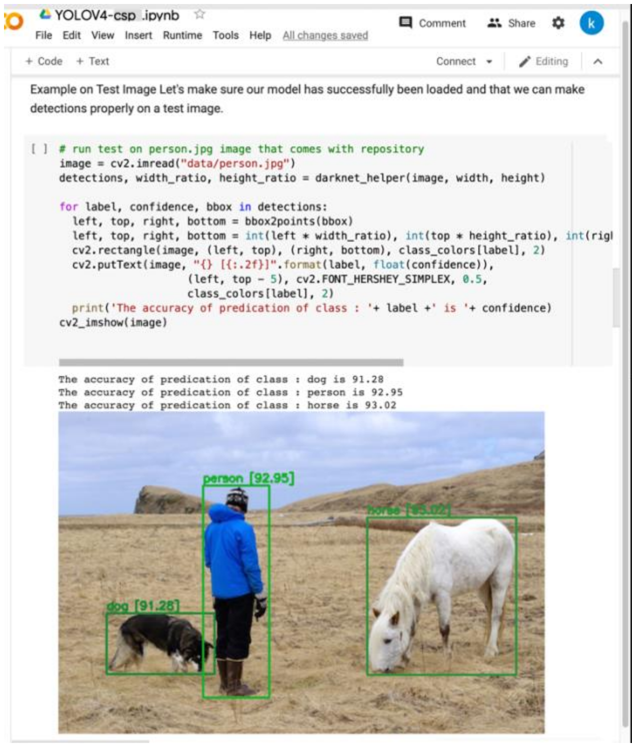
In general, when working on the creation of their models, the Scaled-YOLOv4 writers keep a few scaling aspects in mind: picture size, number of layers, and number of channels, all while optimizing for model performance and inference speed. In their network, they consider ResNet, ResNeXt, and the standard Darknet backbone, as well as a few CSP-ized CNN backbones.

Scaling the YOLOv4-tiny model

Because different constraints, such as memory bandwidth and memory access, come into play on the edge, the YOLOv4-tiny model had different considerations than the Scaled-YOLOv4 model. The authors used OSANet for the YOLOv4-tiny's shallow CNN because of its low processing complexity.

Scaled-YOLOv4 Results

Comparison between the detection accuracy on an jpg image – The result of the csp model are more accurate than that of the tiny model .

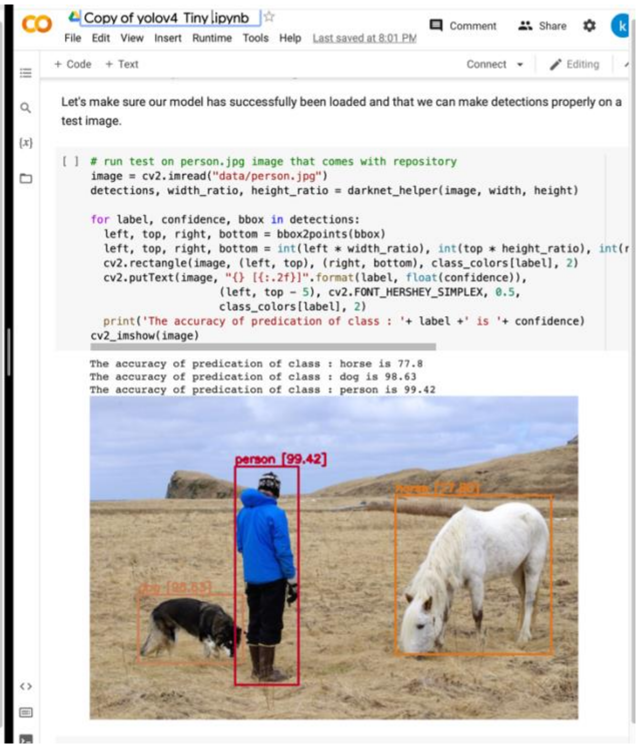

YOLOV4-csp .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text Connect Editing
person [92.95]
horse [93.02]
dog [91.28]

Example on Test Image Let's make sure our model has successfully been loaded and that we can make detections properly on a test image.

```
# run test on person.jpg image that comes with repository
image = cv2.imread("data/person.jpg")
detections, width_ratio, height_ratio = darknet_helper(image, width, height)

for label, confidence, bbox in detections:
  left, top, right, bottom = bbox2points(bbox)
  left, top, right, bottom = int(left * width_ratio), int(top * height_ratio), int(rig
  cv2.rectangle(image, (left, top), (right, bottom), class_colors[label], 2)
  cv2.putText(image, "{} [{:.2f}]".format(label, float(confidence)),
                    (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                    class_colors[label], 2)
  print('The accuracy of predication of class : '+ label +' is '+ confidence)
cv2_imshow(image)
```

```
The accuracy of predication of class : dog is 91.28
The accuracy of predication of class : person is 92.95
The accuracy of predication of class : horse is 93.02
```


Copy of yolov4 Tiny .ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:01 PM
Comment Share
+ Code + Text Connect Editing
person [99.42]

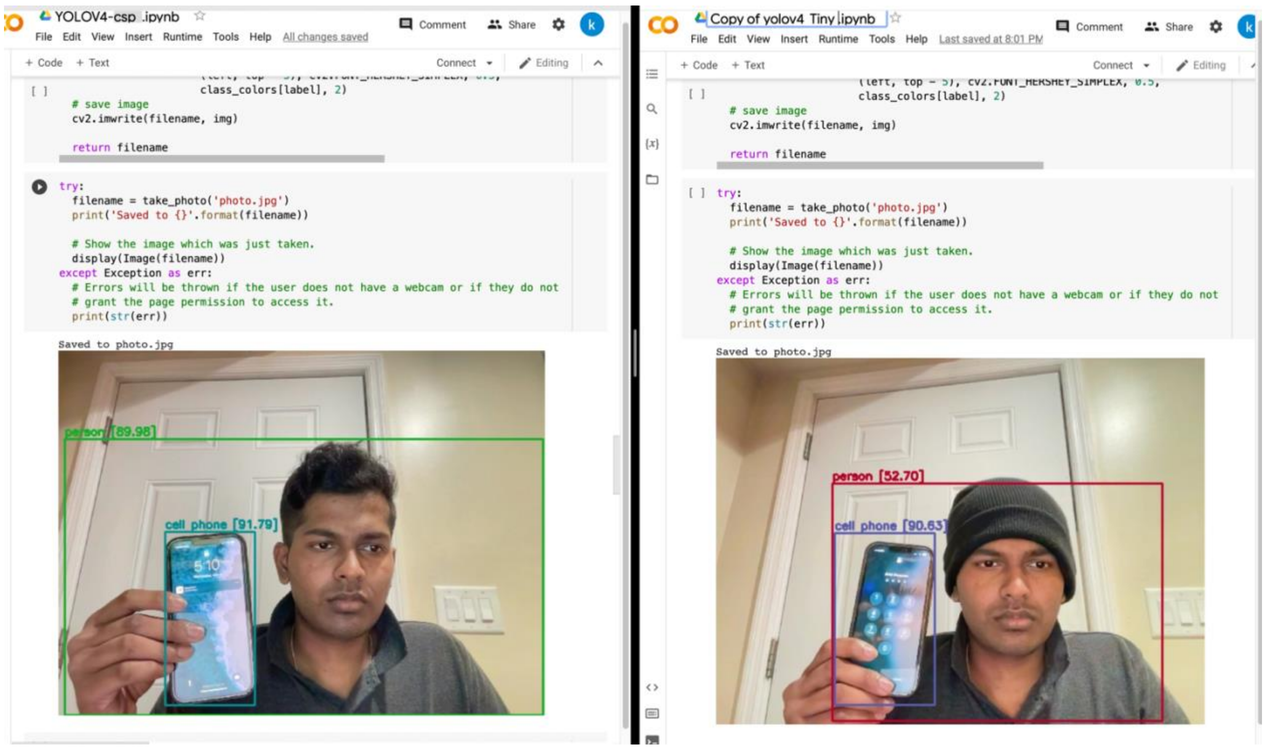
Let's make sure our model has successfully been loaded and that we can make detections properly on a test image.

```
# run test on person.jpg image that comes with repository
image = cv2.imread("data/person.jpg")
detections, width_ratio, height_ratio = darknet_helper(image, width, height)

for label, confidence, bbox in detections:
  left, top, right, bottom = bbox2points(bbox)
  left, top, right, bottom = int(left * width_ratio), int(top * height_ratio), int(r
  cv2.rectangle(image, (left, top), (right, bottom), class_colors[label], 2)
  cv2.putText(image, "{} [{:.2f}]".format(label, float(confidence)),
                    (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                    class_colors[label], 2)
  print('The accuracy of predication of class : '+ label +' is '+ confidence)
cv2_imshow(image)
```

```
The accuracy of predication of class : horse is 77.8
The accuracy of predication of class : dog is 98.63
The accuracy of predication of class : person is 99.42
```

Comparison between the detection accuracy on an image capture from the webcam


YOLOV4-csp .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text Connect Editing
person [89.98]
cell phone [91.79]
Copy of yolov4 Tiny.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:01 PM
Comment Share
+ Code + Text Connect Editing
person [52.70]
cell phone [90.63]

```
                     class_colors[label], 2)
  # save image
  cv2.imwrite(filename, img)

  return filename
```

```
try:
  filename = take_photo('photo.jpg')
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they do not
  # grant the page permission to access it.
  print(str(err))
```

```
Saved to photo.jpg
```

```
                     (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                     class_colors[label], 2)
  # save image
  cv2.imwrite(filename, img)

  return filename
```

```
try:
  filename = take_photo('photo.jpg')
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they do not
  # grant the page permission to access it.
  print(str(err))
```

```
Saved to photo.jpg
```

The results for video streaming is posted in Github. Along with the Zip file

Problems – The google colabs tends crash most of the time and the entire model training need to be re-run from the beginning . This makes the training process even such a model very time consuming . Due to which caused the Tiny model to keep crashing . note- I have attached the model video for csp alone , was unable to record while running the tiny model.

Conclusion

The field of computer vision is more fascinating than it has ever been. In the year 2020, four new standards for object detection (as assessed by the COCO dataset) were established. Faster and more accurate models are making jobs that were previously out of reach for vision developers more feasible, whether it's in terms of delivering the requisite precision for a specific activity or lowering the cost of deploying models to the edge. Unlike many of today's state-of-the-art AI models, Scaled-YOLOv4 raises the bar for object detection thanks to the efforts of several dedicated researchers working on a few cloud GPUs - not a major research institution with a large computational budget. This is a significant triumph for open source's future.