**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-of-the-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 YOLOv4-tiny 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.

CONT...

**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**

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

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

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

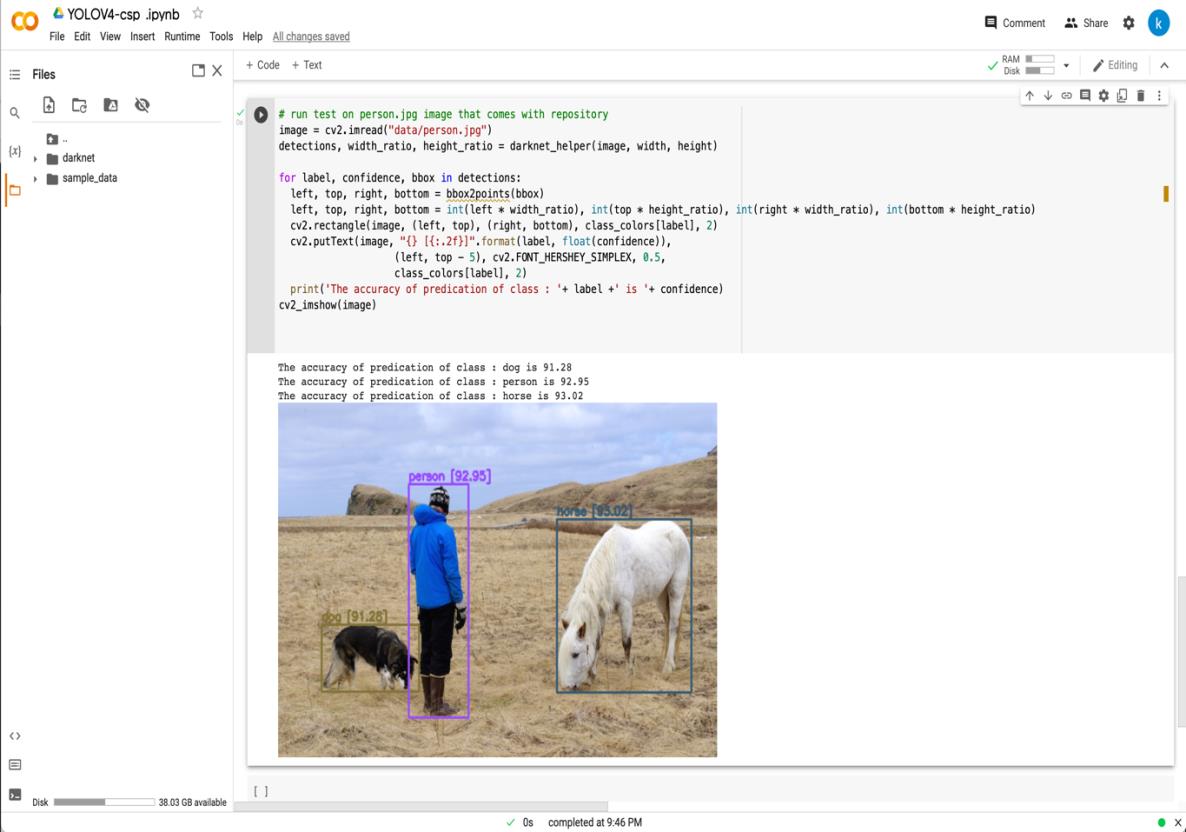
o # import darknet functions to perform object detections o darknet helper function to run detection on image  
o get image ratios to convert bounding boxes to proper size

o run model on darknet style image to get detections

**Result :**

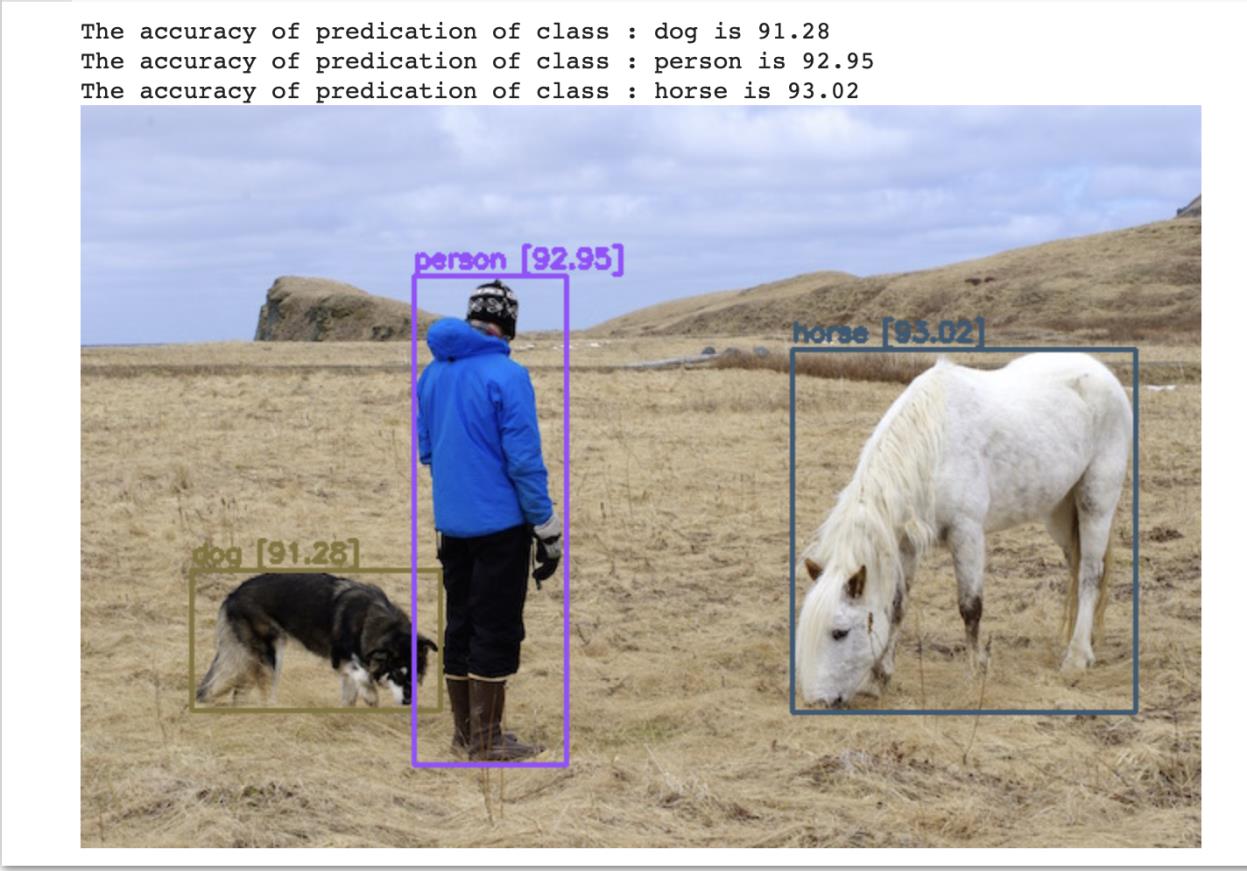


**Pretrained model using Darknet**



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

**ZOOMED IMAGE OF RESULT :**



**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 .