Digital Signal Processing

18EC2015

IMAGE PROCESSING USING YOLO

ALDRIN G
URK22EC1019

DSP in Image Processing:

Image processing is a subset of digital signal processing (DSP) that focuses specifically on manipulating digital images. Here's how image processing is related to DSP:

- **Digital Representation:** Images are represented as discrete signals in digital form, where each pixel corresponds to a value representing color intensity or grayscale level. This representation allows for applying DSP techniques directly to images.

- **Mathematical Operations:** DSP techniques, such as filtering, convolution, Fourier transforms, and correlation, can be applied directly to images for tasks like noise reduction, edge detection, smoothing, enhancement and image compression algorithms like JPEG.

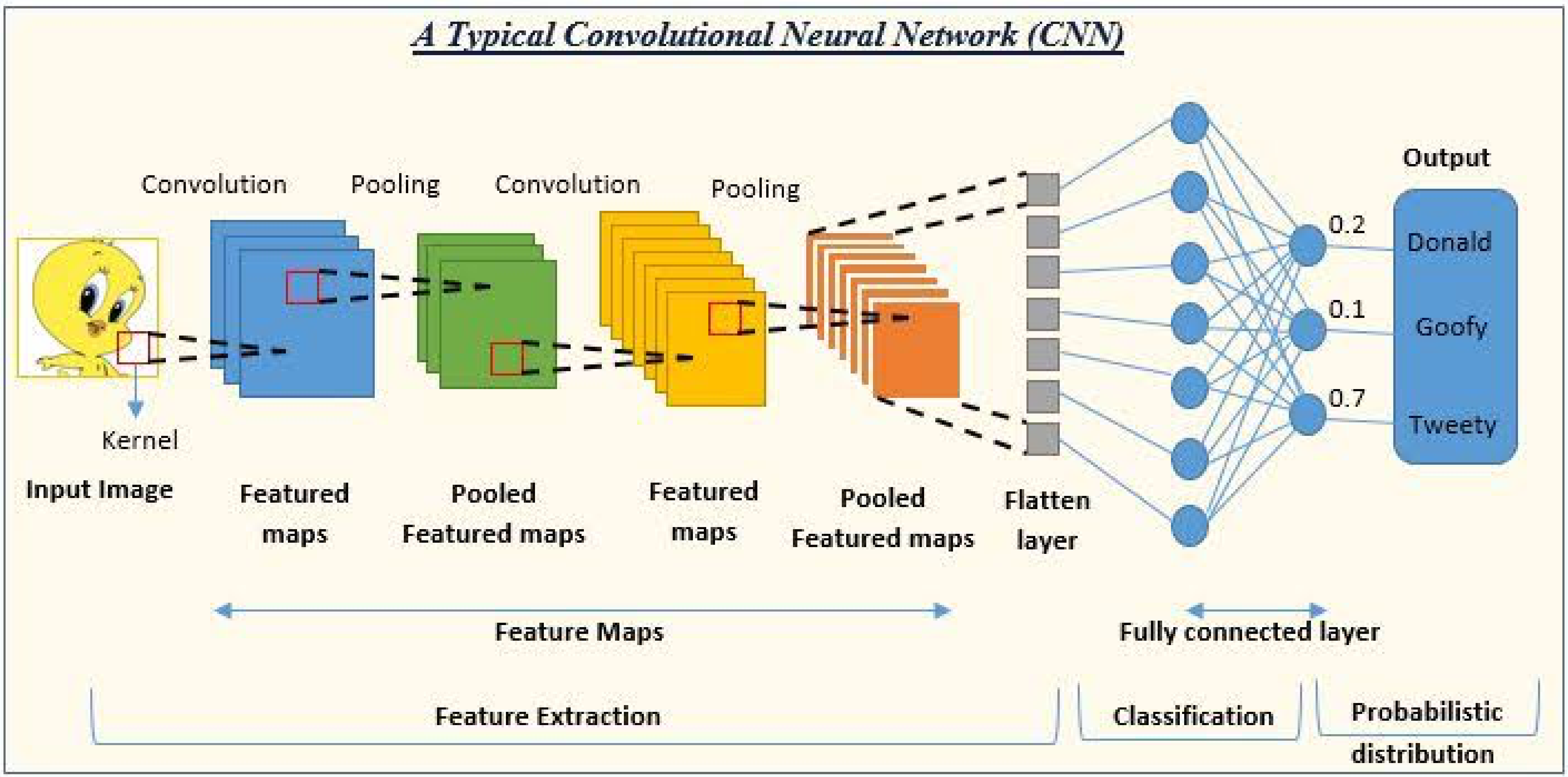
- **Applications:** Its applications include medical imaging, satellite imagery analysis, computer vision, and digital photography. The techniques and algorithms from DSP play a crucial role in enabling various image processing applications.

CNN (Convolution Neural Networks):

CNNs (Convolutional Neural Networks) are a powerful tool in the field of image processing. Here's how they relate:

- **Feature Extraction:** They automatically extract features like edges and textures from images.
- **Classification and Recognition:** Used for identifying objects, scenes, or patterns within images.
- **Object Detection:** Detect and localize objects, vital for tasks like surveillance and medical imaging.
- **Segmentation:** Divide images into meaningful regions, useful in medical imaging and scene understanding.
- **Image Restoration:** Capable of denoising, deblurring, and enhancing image quality.
- **Style Transfer and Generation:** Employed for creative tasks like style transfer and image generation.

CNN (Convolution Neural Networks):



CNN (Convolution Neural Networks):

1. **Input Image:** Imagine you have a big picture, like a photo of a dog. That's our input.

2. **Convolution**: Now, think of a small square on that picture, like a tiny window. We slide this window across the entire picture. At each position, we look at the pixels inside that window and do some math to see if there are any patterns, like edges or textures.

3. **ReLU**: After doing the math, we apply a simple rule: if the result is negative, we make it zero. If it's positive, we keep it the same. This helps us focus on important parts of the picture.

4. **Pooling**: Next, we shrink the picture by taking the maximum value from each small square. It's like zooming out a bit and only focusing on the most important information.

5. **Flattening**: Now, we take all the remaining numbers and put them in a big list. This makes it easier for the computer to understand.

6. **Fully Connected Layers:** Finally, we pass this list through some special layers that learn to recognize patterns and shapes, like dog ears or a tail. These layers help the computer make sense of the entire picture.

7. **Output**: After going through all these steps, the computer gives us an answer, like "Yes, there's a dog in the picture" or "No, there isn't."

YOLOv8 and CNN:

YOLO(You Only Look Once) is actually a specific implementation of a Convolutional Neural Network (CNN) architecture, designed for object detection tasks.

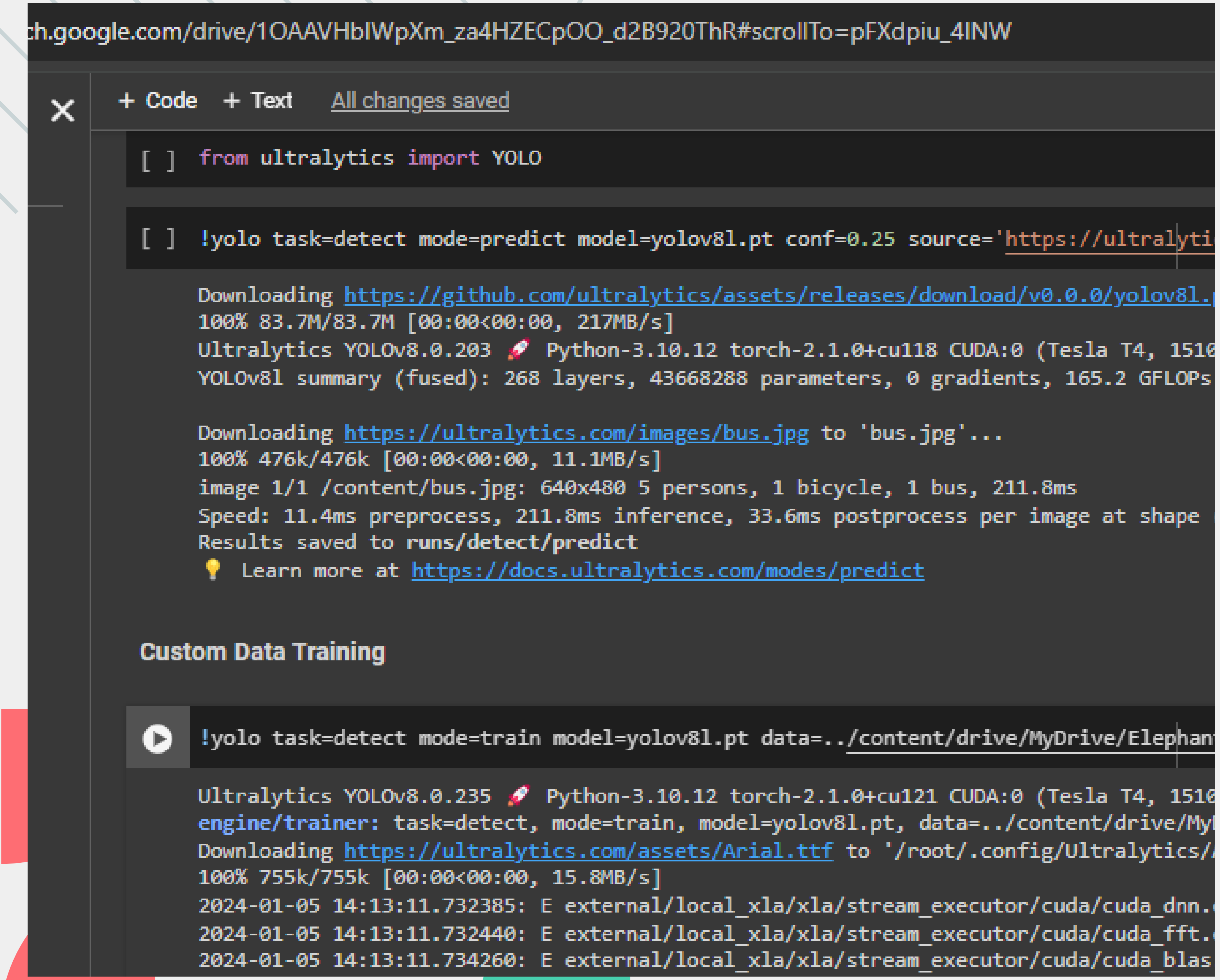
- **YOLOv8 is a specific implementation of a Convolutional Neural Network (CNN) designed for object detection.**
- **focuses on object detection tasks, while CNNs have broader(segmentation,classification,...) applicability.**
- **predicts bounding boxes and class probabilities in a single pass**
- **optimizing for real-time efficiency.**
- **tailored for speed and accuracy in object detection.**

YOLOv8 (Training a Custom Dataset):

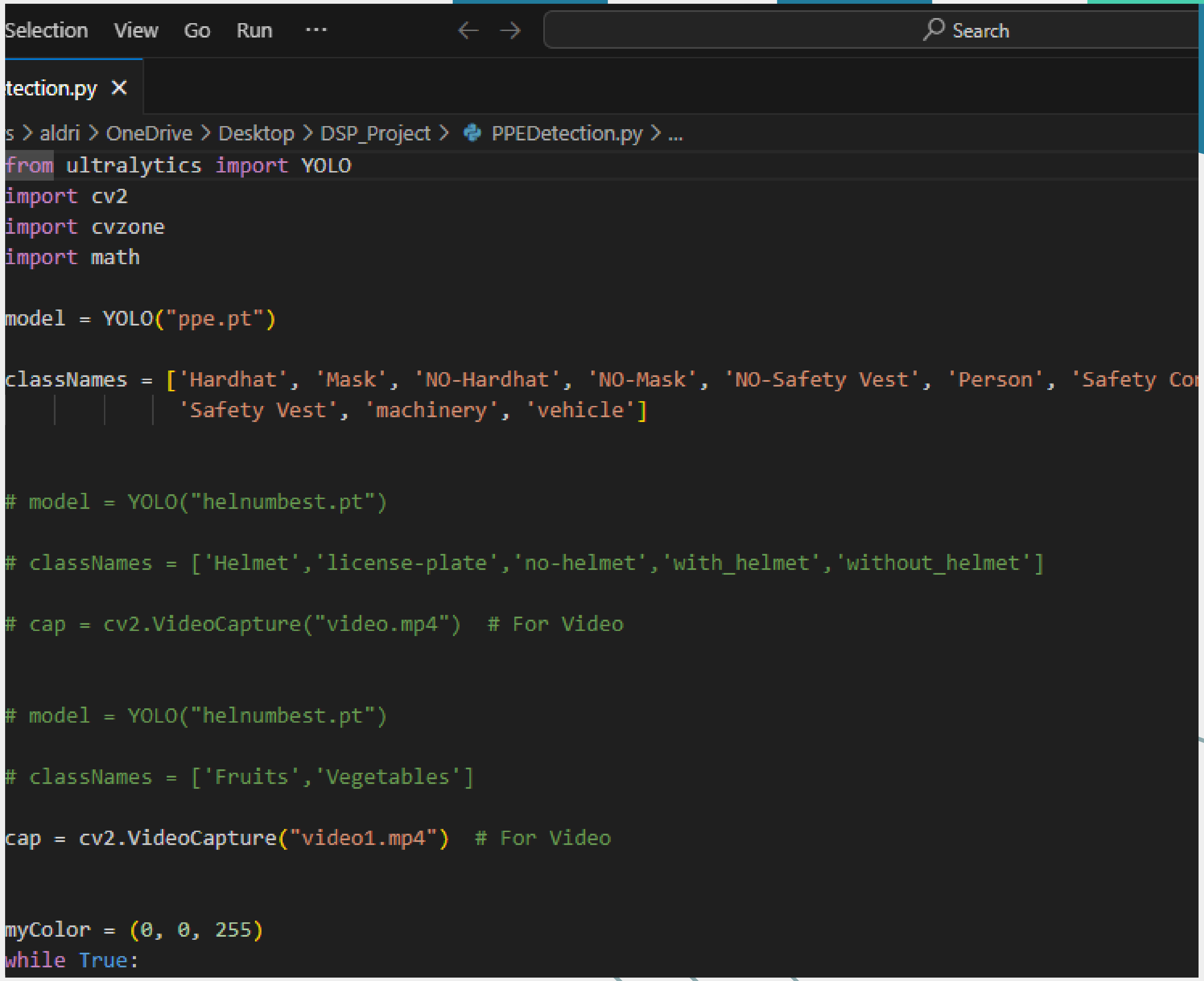
1. **Prepare Dataset:** Organize data into YOLO format (images + annotation files).
2. **Set Up Colab:** Create a new Python 3 notebook and connect to GPU runtime.
3. **GPU:** Connect the runtime to GPU.
4. **Install Dependencies:** Run !pip install -U -r yolov8/requirements.txt to install necessary packages.
5. **Prepare Custom Dataset:** Upload or mount dataset in Colab, and modify YOLOv8 configuration file.
6. **Training:** Execute training script with appropriate arguments, specifying paths to data, configuration, and pre-trained weights.

   !yolo task=detect mode=train model=yolov8l.pt data=xxxx/xxxx/xxxx epochs=30 imgsz=640
7. **Evaluation:** Evaluate model performance using validation or test dataset, measuring metrics like mAP.
8. **Inference:** Use trained model for object detection in new images or videos.

Code:



a) Google Colab

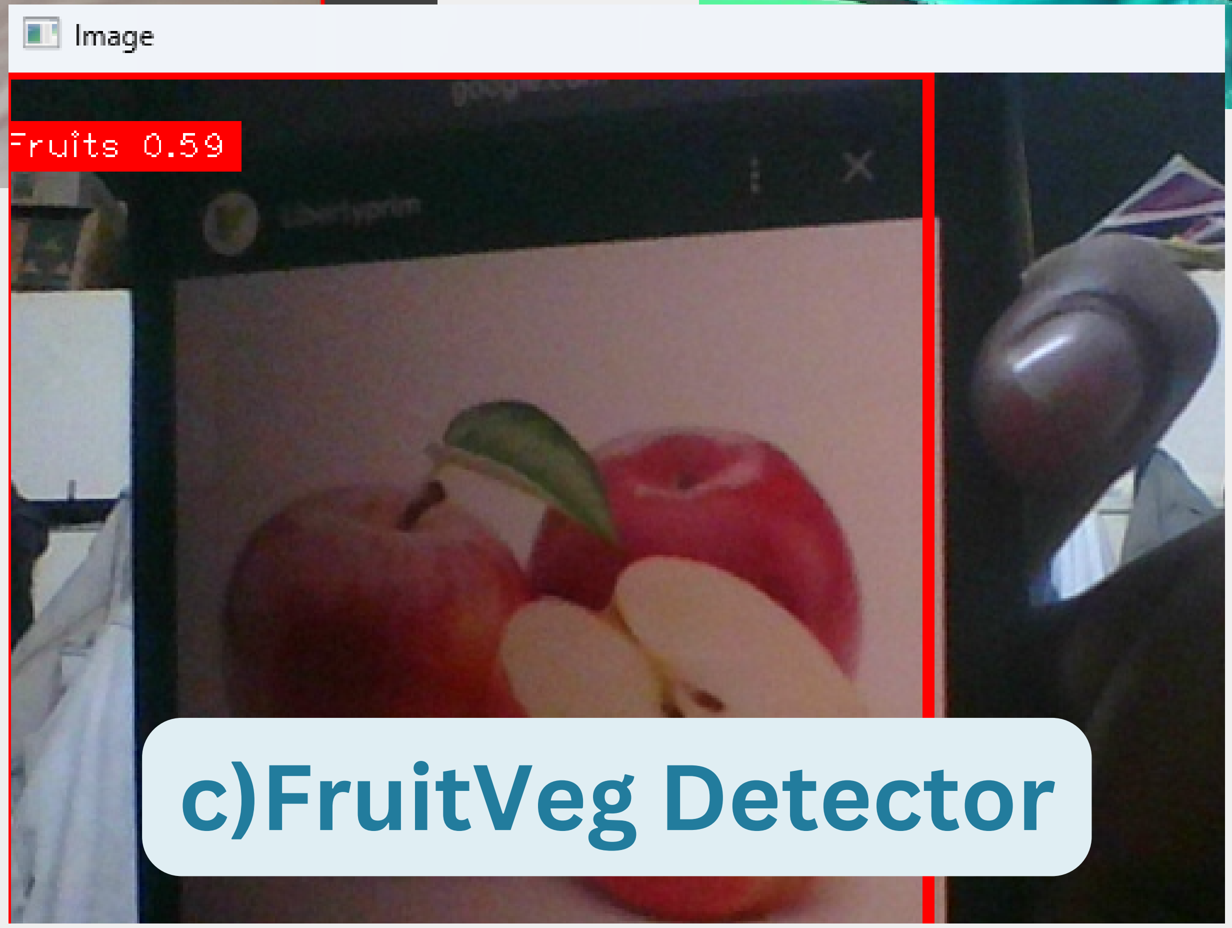


b) VS Code

Results:

a)PPE Detector

b)Helmet Detector



c)FruitVeg Detector

CONCLUSION

In conclusion, integrating DSP techniques with YOLOv8 for object detection can result in more robust, efficient, and accurate systems. By leveraging the strengths of both DSP and deep learning, it is possible to build state-of-the-art object detection solutions that excel in a wide range of applications and environments.

THANK YOU