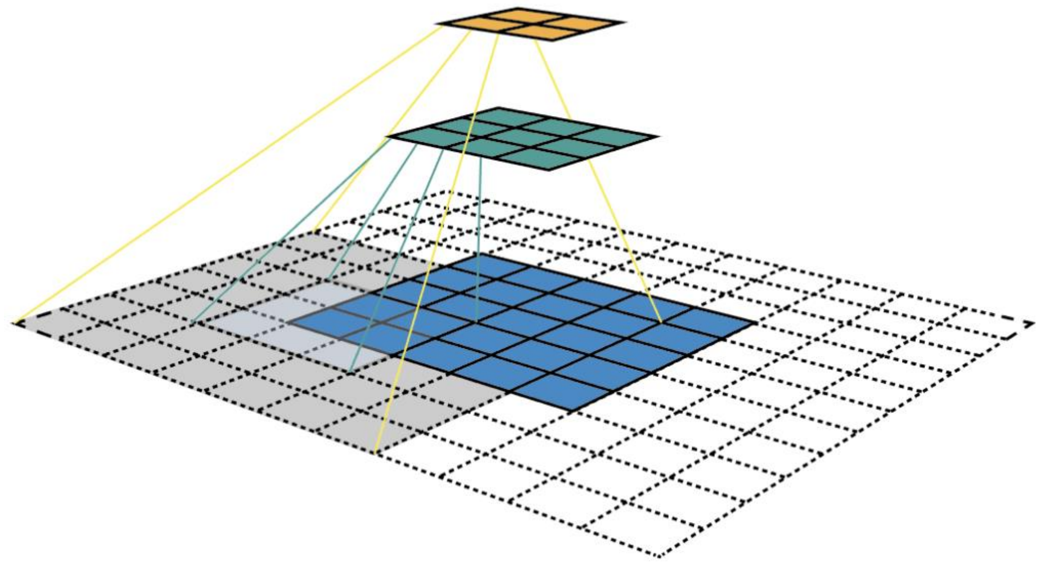


RASPBERRY PI TUTORIAL FOR MACHINE LEARNING

SUBHRANSU S. BHATTACHARJEE

MAY 14, 2025

Australian National University

DUE DILIGENCE:

HAVE YOU SET UP THE PI?

Have you set up your Raspberry Pi's with the SD card?

Does the HDMI port work? Due to the lack of the monitors, we would like everyone to take a round-robin approach to using the Pi for each of the demos.

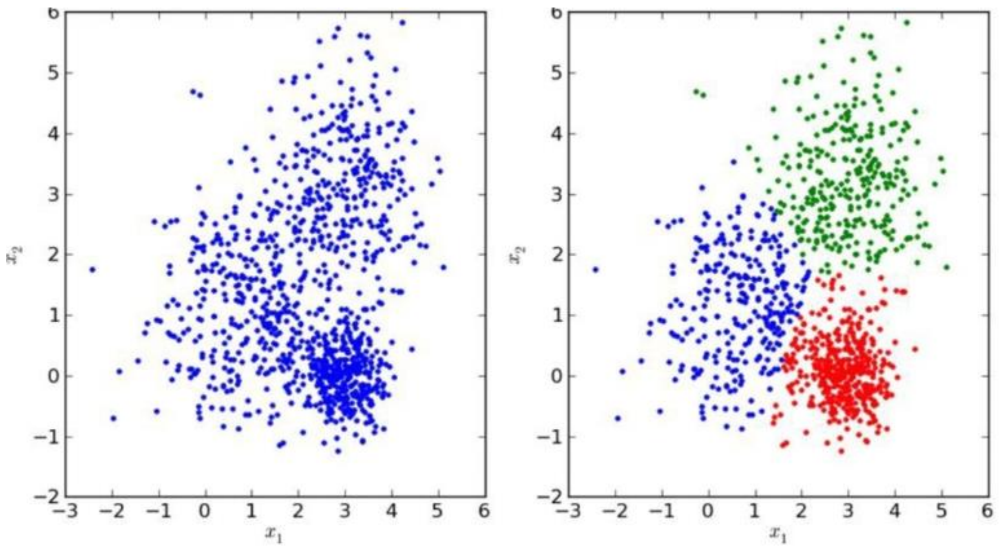
Let's get to the code repo and download the model…..

https://github.com/1ssb/ANU_YOLO.

TYPES OF LEARNING

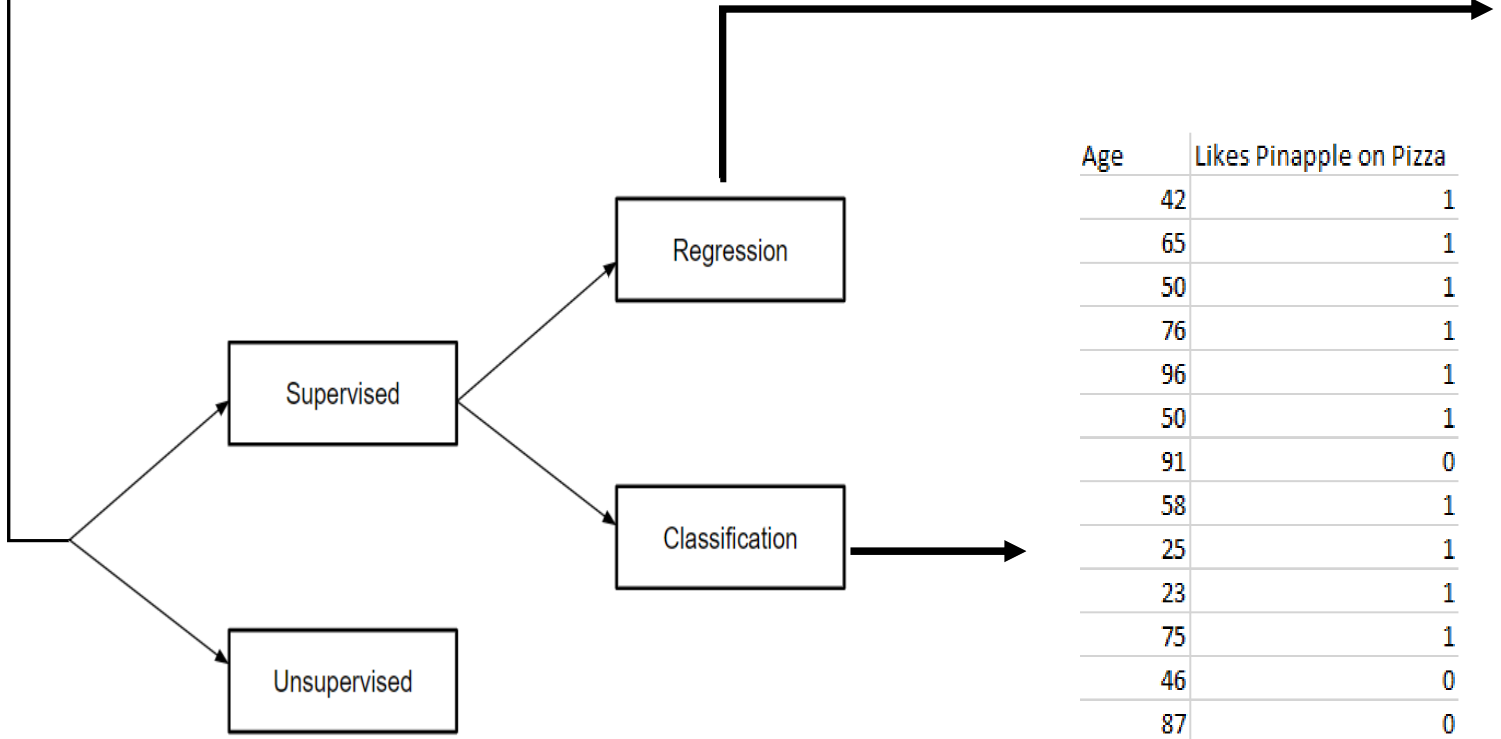
Supervised learning involves learning a function that maps an input to an output based on example input-output pair*.

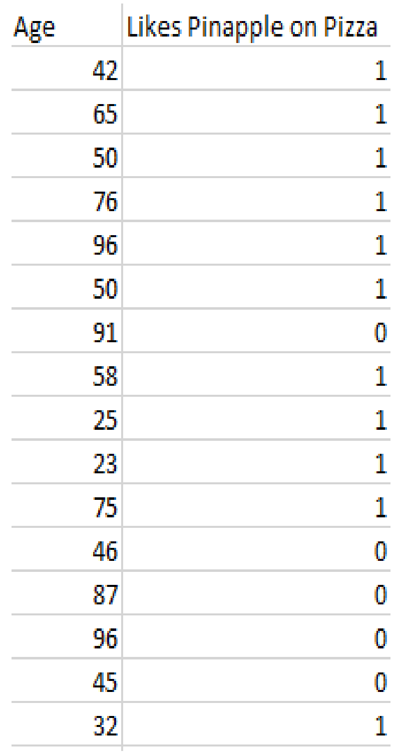
Unsupervised learning is used to draw inferences and find patterns from input data without references to labeled outcomes*. Two main methods used in unsupervised learning include clustering and dimensionality reduction.



*OG AI Book: Stuart J. Russell, Peter Norvig, Artificial Intelligence: A Modern Approach (2010), Prentice Hall

TYPES OF LEARNING

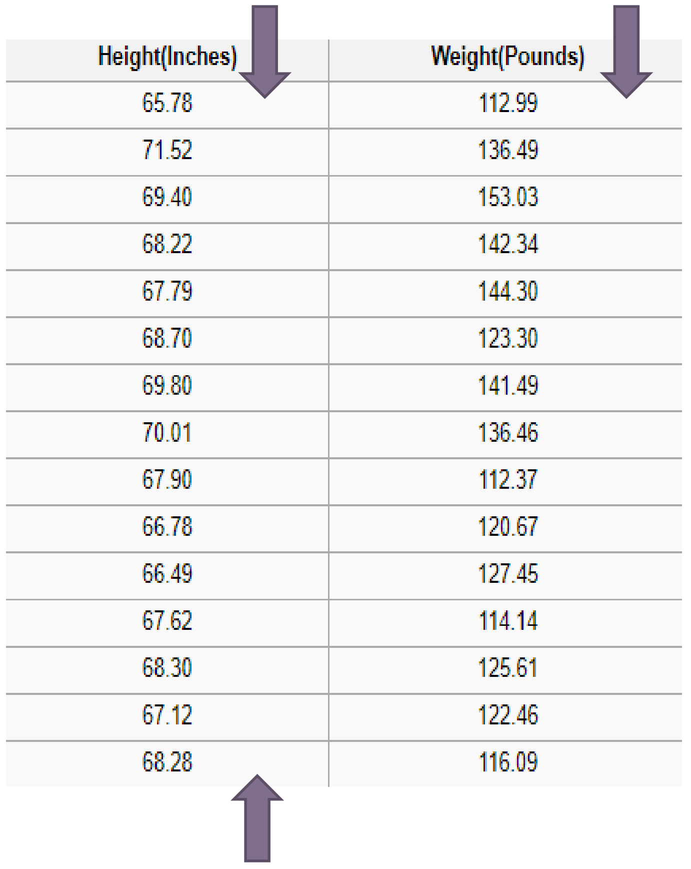




Age	Likes Pinapple on Pizza
42	1
65	1
50	1
76	1
96	1
50	1
91	0
58	1
25	1
23	1
75	1
46	0
87	0
96	0
45	0
32	1

Independent Variable / Predictor/ Feature

Dependent Variable / Label



Height(Inches)	Weight(Pounds)
65.78	112.99
71.52	136.49
69.40	153.03
68.22	142.34
67.79	144.30
68.70	123.30
69.80	141.49
70.01	136.46
67.90	112.37
66.78	120.67
66.49	127.45
67.62	114.14
68.30	125.61
67.12	122.46
68.28	116.09

Data point /Observation /Example

Regression is the idea of finding m & c in **y = mx +c**
Classification is the idea of finding **if x then which y**.

An Introduction to Learning using Computer Vision

The purpose of Computer Vision

• **Automates perception**: Enables machines to "see" and interpret the world—key for robotics, autonomous vehicles, medical imaging, and analytics.

• **Scales human expertise**: Processes vast image/video data in real time, far beyond manual capabilities.

• **Drives innovation**: Powers AR/VR, smart cities, quality control in industry, and new human–machine interfaces.

YOLO: Regression + Classification

1. **Regression:** Predicts precise bounding-box coordinates (x1, y1, x2, y2) directly from each grid cell.

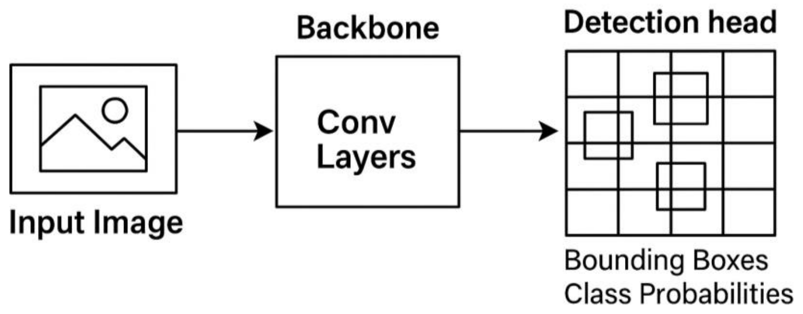
2. **Classification:** Simultaneously outputs class probabilities for each box (what object is inside it) for the class.

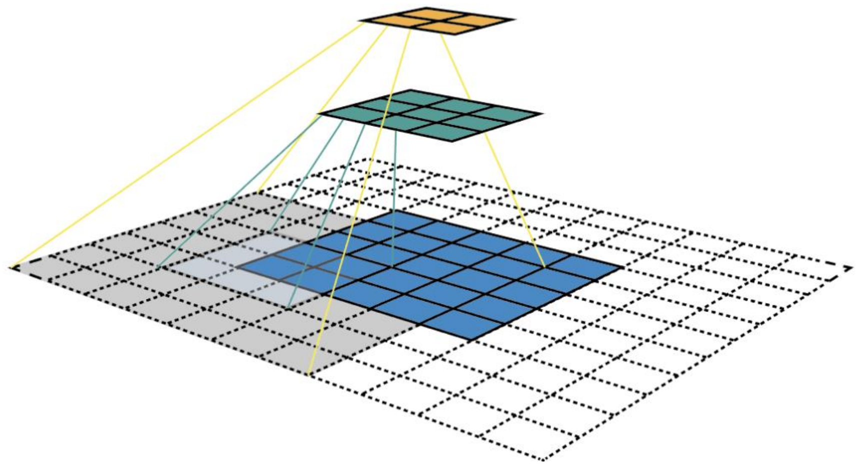
3. **Unified model:** One network does both tasks in a single forward pass—fast and end-to-end trainable.

YOLO: YOU ONLY LIVE ONCE

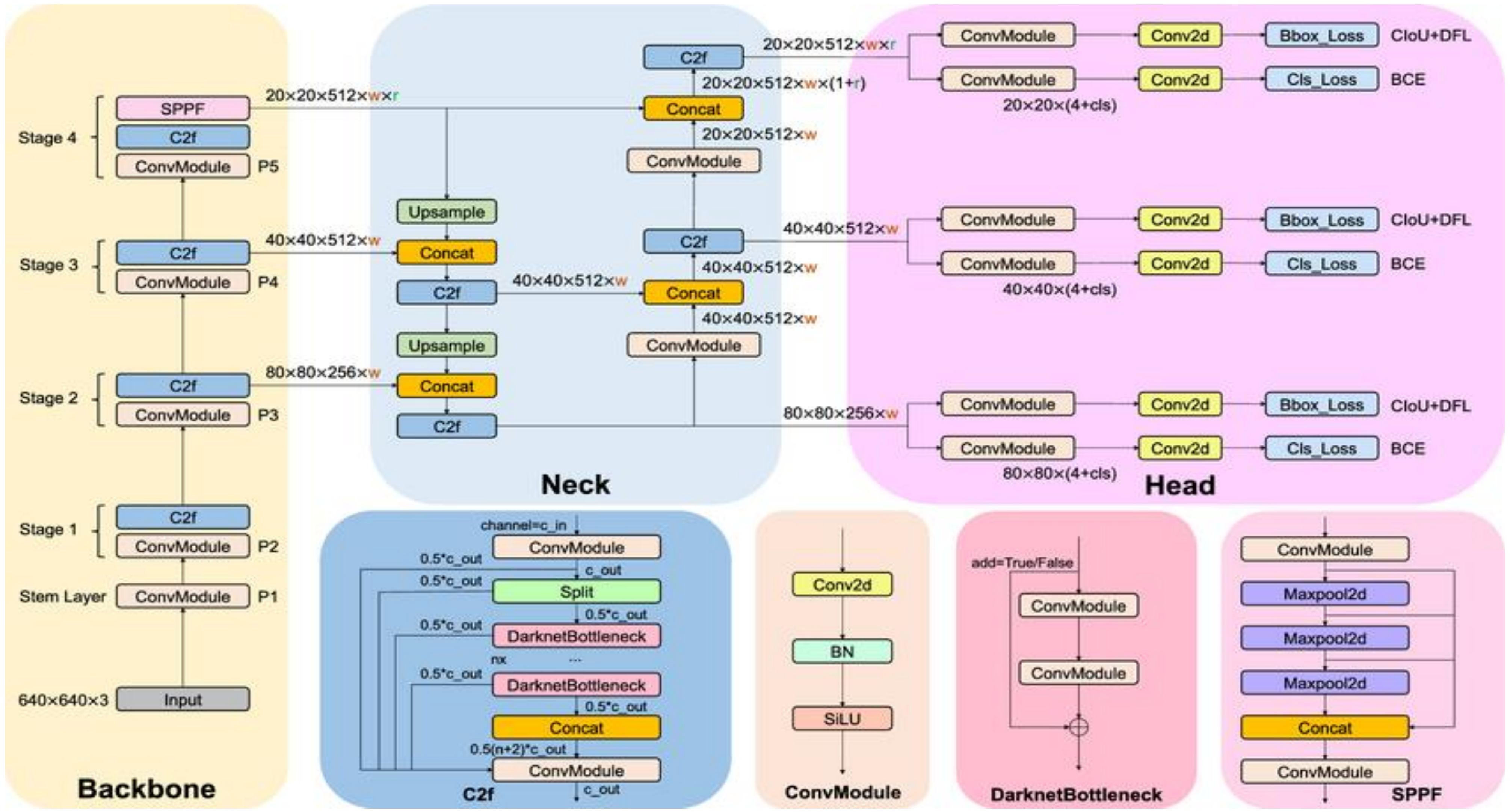
YOLO is a versatile algorithm trained on large-scale datasets to detect and classify objects in real time.

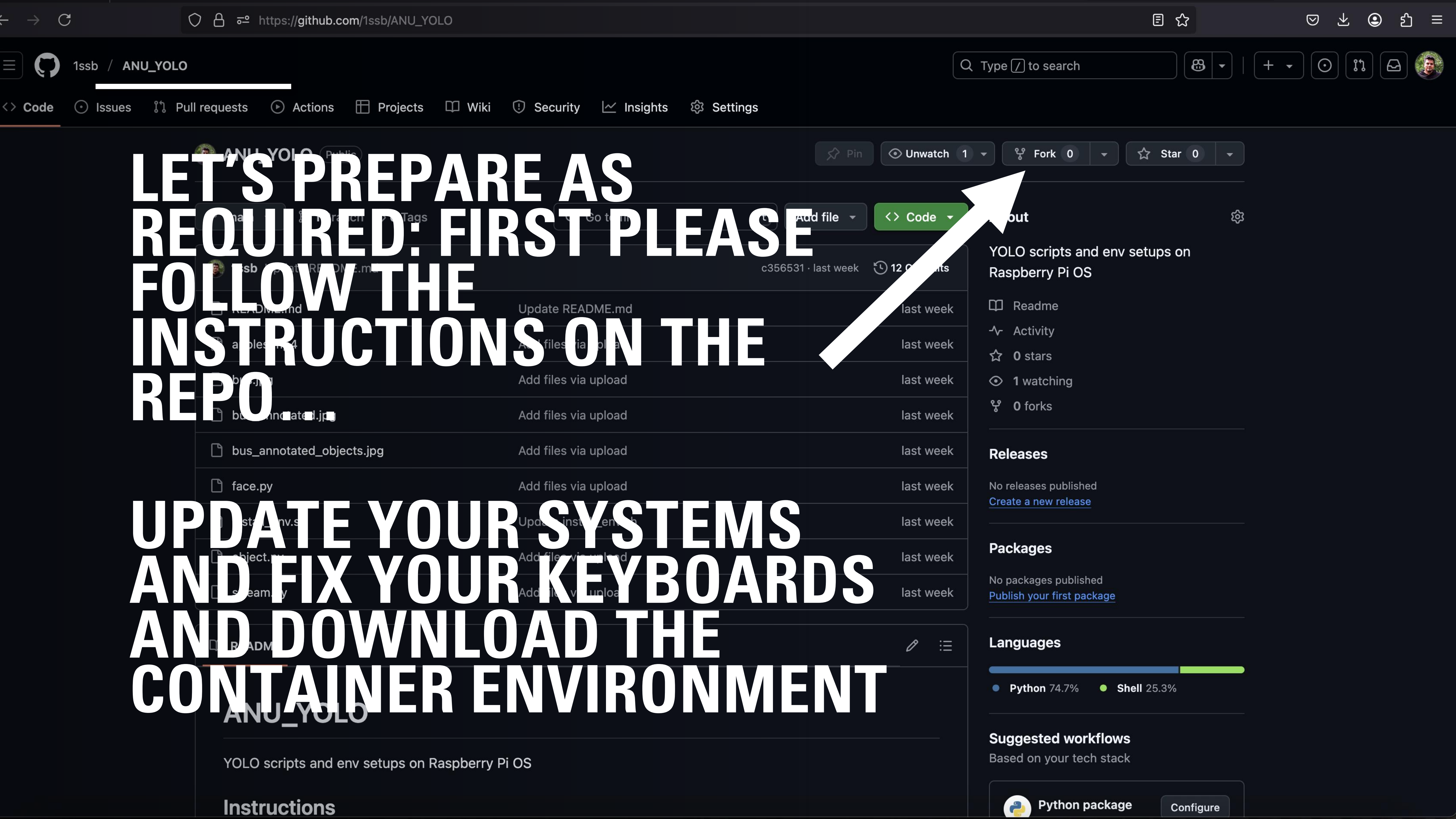
- **Single-pass detection**: Processes the entire image in one forward pass, unifying localization and classification without separate region proposals.
- **Grid-based outputs**: Splits the image into an S×S grid; each cell predicts B bounding boxes (x, y, w, h), a confidence score, and C class probabilities.
- **Real-time performance**: Runs at 30/60+ FPS on modern GPUs—and even on edge devices with the nano/small variants—while maintaining high accuracy and hence our exercise.





YOLOv8 algorithm: Significantly more complex than how it started





LET'S PREPARE AS REQUIRED: FIRST PLEASE FOLLOW THE INSTRUCTIONS ON THE REPO...

UPDATE YOUR SYSTEMS AND FIX YOUR KEYBOARDS AND DOWNLOAD THE CONTAINER ENVIRONMENT

https://github.com/1ssb/ANU_YOLO

1ssb / ANU_YOLO

README.md	Update README.md	last week
bus_annotated_objects.jpg	Add files via upload	last week
face.py	Add files via upload	last week
object.py	Add files via upload	last week
stream.py	Add files via upload	last week

c356531 · last week

About

YOLO scripts and env setups on Raspberry Pi OS

Readme · Activity · 0 stars · 1 watching · 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 74.7% Shell 25.3%

Suggested workflows

Based on your tech stack

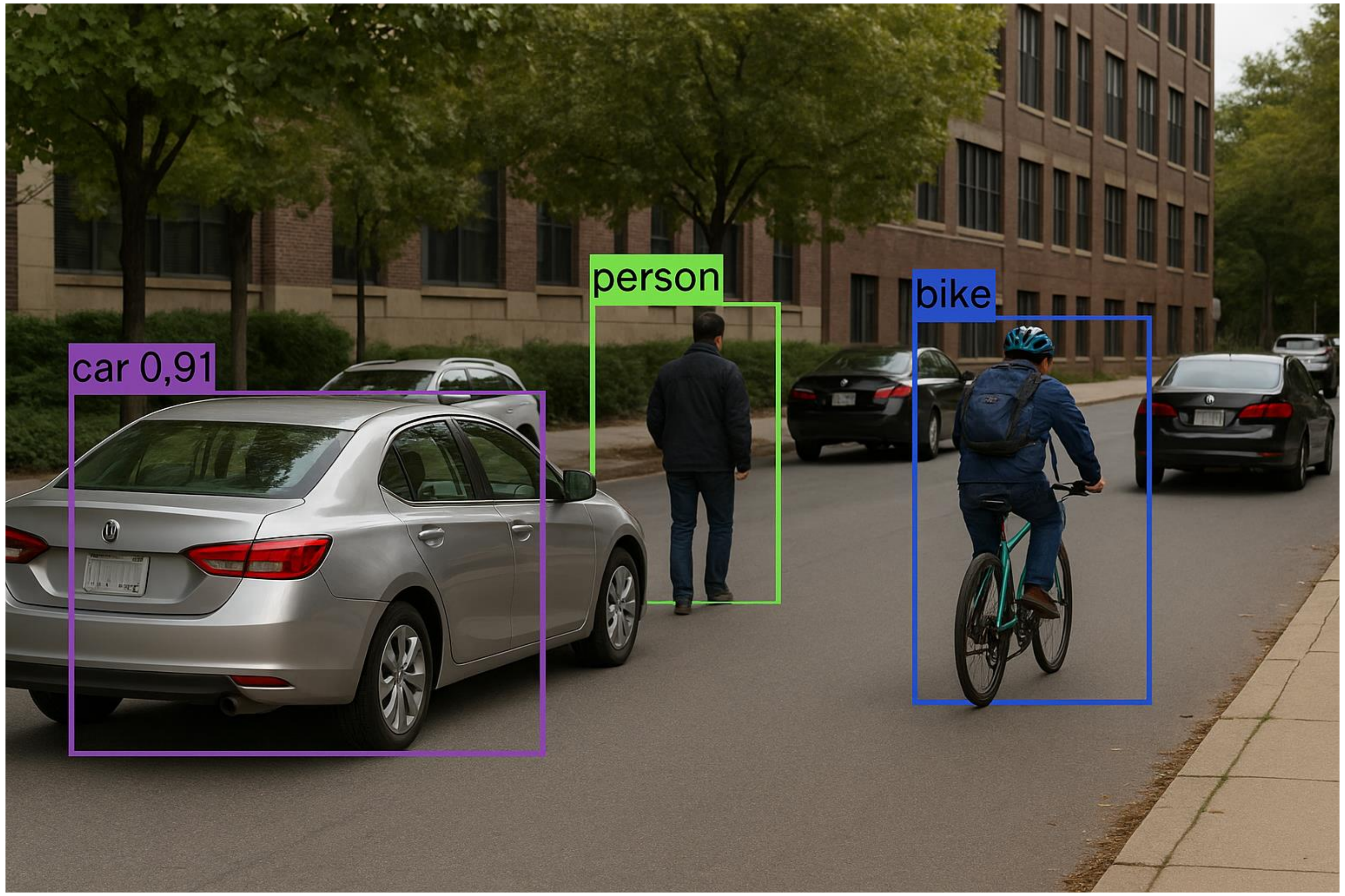
Python package Configure

ANU_YOLO

YOLO scripts and env setups on Raspberry Pi OS

Instructions

DEMO1: OBJECT DETECTION



Goal: Identify and localize objects with bounding boxes and class labels.

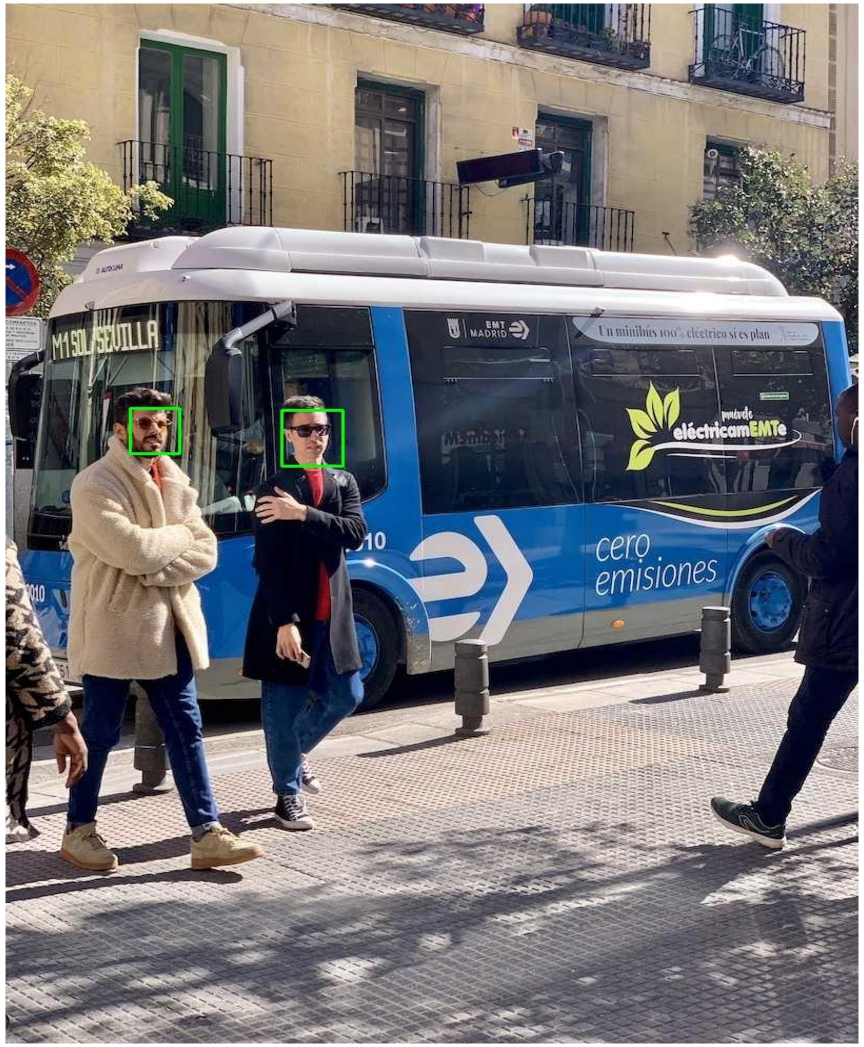
Example (YOLOv8n or Nano):

A street scene with "car", "person", and "bike" boxes labelled with confidence scores.

Key Steps:

1. Input full image.
2. Split into grid.
3. Predict B boxes + confidence + class scores per cell. Maybe threshold the confidence.

BONUS: Will demonstrate the webcam object detection using `stream.py` if we have time



DEMO 2: DETECT FACES

Goal: Locate and identify human faces within an image, returning bounding boxes and confidence scores.

Key Steps:

1. **Preprocessing:** Resize and normalize the input image (e.g., to 416×416).
2. **Feature Extraction:** Use a lightweight CNN backbone (e.g., Tiny-YOLO, MTCNN) to extract facial features.
3. **Confidence Thresholding and NMS suppression:** Necessary in the process of calibration.

GROUP TASK CHALLENGE: DETECT THE APPLES

Task: Count the apples in the video in total.

Option A: Detection + Tracking (HARDER but HIGH ACCURACY)

• **Detect** each frame with a fruit-specialized object detector.
• **Assign IDs** to detections as they move across frames (e.g. via a simple tracker).
• **Aggregate** unique IDs to get the total count—prevents double-counting moving fruits.

Option B: Heuristic Frame Sampling (EASIER but LOW ACCURACY)

• **Sample** one or more representative frames (e.g. every 10th frame or a key frame).
• **Detect** fruits in each sampled frame.
• **Take the maximum** number of detections seen in any single frame as your total—ideal when fruits don't reappear multiple times.

- **JUSTIFY YOUR EXACT CHOICE** and sub-system design choices as comments…..
- Use the Ultralytics documentation, link in repo.

THANK YOU!

For your attention and time.

Questions?

CONTACT ME

W: 1ssb.github.io
E: Subhransu.Bhattacharjee@anu.edu.au